**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**

**NGUYỄN ANH KHOA**

**HỆ THỐNG GIAO LƯU MUA BÁN BẤT ĐỘNG SẢN**

**ĐỒ ÁN NGÀNH**

**NGÀNH CNTT**

**TP. HỒ CHÍ MINH, 2025BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**

**NGUYỄN ANH KHOA**

**HỆ THỐNG GIAO LƯU MUA BÁN BẤT ĐỘNG SẢN**

**Mã số sinh viên: 2251052052**

**ĐỒ ÁN NGÀNH**

**NGÀNH CNTT**

**Giảng viên hướng dẫn: THS. DƯƠNG HỮU THÀNH**

**TP. HỒ CHÍ MINH, 2025**

LỜI CẢM ƠN

Trong suốt quá trình học tại trường, em đã có cơ hội lĩnh hội được rất nhiều kiến thức hay và bổ ích từ các quý Thầy/Cô là giảng viên của khoa Công nghệ thông tin Trường Đại Học Mở Thành phố Hồ Chí Minh. Bằng sự tâm huyết và tấm lòng nhiệt thành mà Thầy/Cô đã kiên nhẫn trong suốt quá trình giảng dạy, em càng thêm trân trọng và biết ơn những tri thức quý giá cùng sự tận tâm mà Thầy/Cô đã dành cho chúng em..

Ngày hôm nay có cơ hội để thực hiện một dự án đặc biệt quan trọng như Đồ Án Ngành, em đã có cơ hội để ứng dụng các kiến thức mà mình đã học được vào một dự án đặc biệt quan trọng mang tính chuyển mình, em nhận thấy công ơn của Thầy cô là vô cùng lớn vì nếu không có sự dìu dắt từ quý thầy cô, chắc có lẽ sẽ không có em của ngày hôm nay.

Để hoàn thành đồ án, em không thể nào quên công ơn và sự giúp sức cực kỳ to lớn từ một người thầy mà em hết mực trân quý: ThS. Dương Hữu Thành. Thầy đã hỗ trợ em tận tình, tận tâm trong suốt quá trình em thực hiện Đồ Án, là người đã đưa ra những đề xuất, định hướng, những phương án khả thi để giải quyết các vấn đề gặp phải. Chính nhờ sự giúp sức đó, em đã kịp thời hoàn thành Báo cáo này đúng hạn và hoàn thiện kĩ lưỡng. Từ tận sâu trong đáy lòng, em xin gửi lời cảm ơn sâu sắc đến Thầy.

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

TÓM TẮT ĐỐ ÁN NGÀNH

**Mục tiêu đề tài:**

Đồ án này hướng đến việc xây dựng một ứng dụng giao lưu mua bán bất động sản minh bạch, hiệu quả bằng cách phát triển nền tảng mobile app tích hợp tìm kiếm, phân tích dữ liệu và gợi ý thông minh. Người dùng có thể đăng tin, tìm kiếm bất động sản theo tiêu chí như vị trí, giá, diện tích, tiện ích,…

**Giải pháp thực hiện:**

Phát triển ứng dụng di động (Flutter) với giao diện hiện đại, tối ưu trải nghiệm.

Xây dựng trên nền tảng Django REST Framework mạnh mẽ cho phép người dùng đăng tải và quản lý thông tin bất động sản một cách chi tiết.

Tích hợp bản đồ số giúp người dùng trực quan hóa vị trí và khu vực xung quanh bất động sản.

Tích hợp các công cụ tương tác thời gian thực để kết nối người mua và người bán một cách hiệu quả.

Gợi ý phong thủy cá nhân hóa dựa trên năm sinh, bản mệnh người dùng.

Xây dựng diễn đàn cộng đồng: đánh giá tin đăng, cảnh báo rủi ro, chia sẻ kinh nghiệm.

Sử dụng kiến trúc bất đồng bộ để xử lý các tác vụ nền, từ đó giảm bớt gánh nặng xử lý nghiệp vụ trong luồng chính và nâng cao hiệu quả vận hành hệ thống.

**Kết quả đạt được:**  
Hệ thống giúp người dùng tiếp cận thông tin bất động sản minh bạch hơn, dễ dàng đưa ra quyết định mua bán nhờ các công cụ phân tích và tư vấn thông minh. Đồng thời, cộng đồng người dùng góp phần tạo nên môi trường đáng tin cậy, giảm thiểu rủi ro, nâng cao hiệu quả giao dịch.

**MỤC LỤC**

[LỜI CẢM ƠN 1](#_Toc208180353)

[NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN 2](#_Toc208180354)

[TÓM TẮT ĐỐ ÁN NGÀNH 3](#_Toc208180355)

[DANH MỤC TỪ VIẾT TẮT 5](#_Toc208180356)

[DANH MỤC HÌNH VẼ 6](#_Toc208180357)

[DANH MỤC BẢNG 7](#_Toc208180358)

[MỞ ĐẦU 8](#_Toc208180359)

[TÀI LIỆU THAM KHẢO 56](#_Toc208180360)

[PHỤ LỤC 57](#_Toc208180361)

DANH MỤC TỪ VIẾT TẮT

DANH MỤC HÌNH VẼ

Không tìm thấy mục nhập nào của bảng hình minh họa.

DANH MỤC BẢNG

Không tìm thấy mục nhập nào của bảng hình minh họa.

MỞ ĐẦU

# TỔNG QUAN ĐỀ TÀI

## Giới thiệu đề tài

Thị trường bất động sản Việt Nam hiện nay đang phát triển mạnh mẽ, với nhu cầu mua bán, đầu tư ngày càng tăng cao tại ở cả khu vực thành thị và nông thôn. Tuy nhiên, bên cạnh tiềm năng, thị trường vẫn tồn tại nhiều **thách thức lớn,** trong đó nổi bật là tình trạng **thiếu minh bạch thông tin.**

Người mua thường gặp khó khăn trong việc xác minh nguồn gốc pháp lý, so sánh giá thị trường hay đánh giá tiềm năng đầu tư của một bất động sản. Trong khi đó, người bán cũng khó tiếp cận đúng đối tượng khách hàng mà mình muốn hướng tới. Nhiều trường hợp **lừa đảo, thổi giá, tin giả**, hoặc **thiếu công cụ hỗ trợ phân tích** khiến quá trình giao dịch trở nên rủi ro, tốn thời gian và chi phí.

Để góp phần giải quyết những vấn đề trên, nhóm đã thực hiện đề tài **“Hệ thống giao lưu mua bán bất động sản”,** với mong muốn xây dựng một nền tảng trực tuyến hỗ trợ người dùng đăng tin, tìm kiếm, phân tích và trao đổi thông tin một cách **hiệu quả, minh bạch và thông minh hơn**. Qua đó, đề tài không chỉ đáp ứng nhu cầu thực tế cấp thiết, mà còn hướng đến việc nâng cao chất lượng giao dịch, góp phần vào sự phát triển bền vững của thị trường bất động sản Việt Nam.

## Lý do chọn đề tài

Việc thiếu minh bạch thông tin trong thị trường bất động sản không chỉ gây trở ngại cho quá trình mua bán, mà còn dẫn đến rủi ro cao như mua phải tài sản không đúng giá trị, dính pháp lý, hoặc bị lừa đảo. Trong khi đó, nhu cầu tra cứu, so sánh và phân tích thông tin trước khi quyết định đầu tư là rất lớn nhưng chưa có nhiều nền tảng hỗ trợ hiệu quả, minh bạch và tin cậy.

Việc xây dựng hệ thống giao lưu mua bán bất động sản không chỉ giúp các bên tham gia dễ dàng tiếp cận thông tin chính xác, mà còn góp phần **nâng cao tính minh bạch của thị trường**, giảm thiểu rủi ro và tiết kiệm thời gian trong quá trình giao dịch. Hệ thống còn mở ra hướng phát triển các công nghệ mới như **AI, dữ liệu lớn và phân tích phong thủy**, mang lại giá trị thực tiễn cao, phù hợp với xu thế chuyển đổi số trong lĩnh vực bất động sản hiện nay.

## Mục tiêu đề tài

Xây dựng một website giao lưu mua bán bất động sản thân thiện, dễ sử dụng.

Cho phép người dùng đăng tin, tìm kiếm bất động sản theo các tiêu chí cụ thể (vị trí, diện tích, giá cả, tiện ích…).

Tích hợp chức năng phân tích, so sánh và tư vấn thông minh giúp người dùng đưa ra quyết định chính xác hơn.

Cung cấp tính năng gợi ý theo phong thủy dựa trên thông tin cá nhân như năm sinh, bản mệnh – yếu tố được nhiều người Á Đông nói chung và người Việt Nam quan tâm.

Xây dựng một mạng lưới người dùng để chia sẻ kinh nghiệm, cảnh báo rủi ro và đánh giá tin đăng.

## Phương pháp thực hiện

Trong quá trình thực hiện đề tài, nhóm đã tiến hành khảo sát một số hệ thống, nền tảng và mô hình đã có trong lĩnh vực giao dịch bất động sản để hiểu rõ cách vận hành, ưu điểm và hạn chế của từng phương pháp. Từ đó, nhóm lựa chọn các hướng tiếp cận phù hợp nhằm đảm bảo hệ thống đáp ứng tốt yêu cầu thực tế.

Cụ thể, nhóm thực hiện các bước như sau:

Khảo sát, phân tích hệ thống thực tế: Tìm hiểu một số trang web và ứng dụng bất động sản đang phổ biến hiện nay như Batdongsan.com.vn, Alonhadat, Nhatot, Guland... để đánh giá về giao diện, tính năng, khả năng tìm kiếm và mức độ minh bạch thông tin.

Thu thập dữ liệu: Tiến hành thu thập dữ liệu thực tế từ các nguồn công khai như bản đồ, thông tin quy hoạch, các tin rao bán thực, đồng thời tạo bộ dữ liệu giả lập để phục vụ quá trình thử nghiệm và mô phỏng.

Thử nghiệm mô hình và đánh giá: Áp dụng một số mô hình thiết kế hệ thống, từ đơn giản đến phức tạp (ví dụ: mô hình MVC, RESTful API) để so sánh hiệu quả hoạt động, tính mở rộng và độ linh hoạt. Qua đó, nhóm lựa chọn mô hình phù hợp nhất với nhu cầu triển khai thực tế.

Lựa chọn công nghệ và triển khai: Dựa trên kết quả đánh giá, nhóm quyết định sử dụng Django REST Framework để xây dựng phần backend API mạnh mẽ, dễ mở rộng và bảo mật tốt; đồng thời sử dụng Flutter để phát triển giao diện người dùng (frontend) linh hoạt, hỗ trợ đa nền tảng (web và mobile). Việc kết hợp hai công nghệ này giúp hệ thống dễ dàng triển khai và nâng cấp trong tương lai.

Thông qua phương pháp này, nhóm có thể xây dựng một sản phẩm vừa bền vững về kiến trúc, vừa đáp ứng yêu cầu về tính năng, hiệu suất và trải nghiệm người dùng trong môi trường thực tế.

## Bố cục báo cáo

Báo cáo đồ ngành này có 4 chương:

Chương 1. Tổng quan đề tài: Nhằm giới thiệu bối cảnh của đề tài, phương pháp nghiên cứu và mục tiêu của đề tài.

Chương 2. Cơ sở lý thuyết: Trình bày nền tảng lý thuyết áp dụng cho đề tài.

Chương 3. Kiến trúc hệ thống: Đi sâu vào việc phân tích các yêu cầu chức năng, xác định các tác nhân và mô hình hóa các luồng hoạt động của hệ thống thông qua biểu đồ Use Case. Chương này cũng trình bày chi tiết về thiết kế cơ sở dữ liệu và các mối quan hệ giữa các thực thể.

Chương 4. Kết luận và hướng phát triển: Tổng kết lại những kết quả đã đạt được, đánh giá các ưu điểm và những mặt còn hạn chế của dự án. Đồng thời, đề xuất các định hướng phát triển và nâng cấp tiềm năng cho hệ thống trong tương lai.

# CƠ SỞ LÝ THUYẾT

## Các mô hình và kiến trúc phần mềm được áp dụng

### Mô hình Client – Server

Hệ thống Landify được xây dựng theo kiến trúc Client-Server. Ứng dụng di động Flutter đóng vai trò Client, là giao diện chính nơi người dùng xem tin, tìm kiếm, nhắn tin hay xác thực danh tính. Mọi thao tác từ Client được gửi dưới dạng HTTP request đến Server - hệ thống backend phát triển bằng Django với Django Rest Framework. Server xử lý logic nghiệp vụ, truy xuất dữ liệu từ PostgreSQL, và trả về JSON response qua API.

Quy trình hoạt động: Từ giao diện Flutter, sự kiện được chuyển qua BLoC, Repository và gửi request HTTP (Dio) đến Server. Django định tuyến request đến ViewSet, dùng ORM để truy vấn dữ liệu, Serializer để chuyển đổi rồi phản hồi JSON. Client nhận dữ liệu, parse thành model và cập nhật giao diện.

Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, hàng

Nội dung do AI tạo ra có thể không chính xác.

Hình . Mô hình Client - Server của Landify

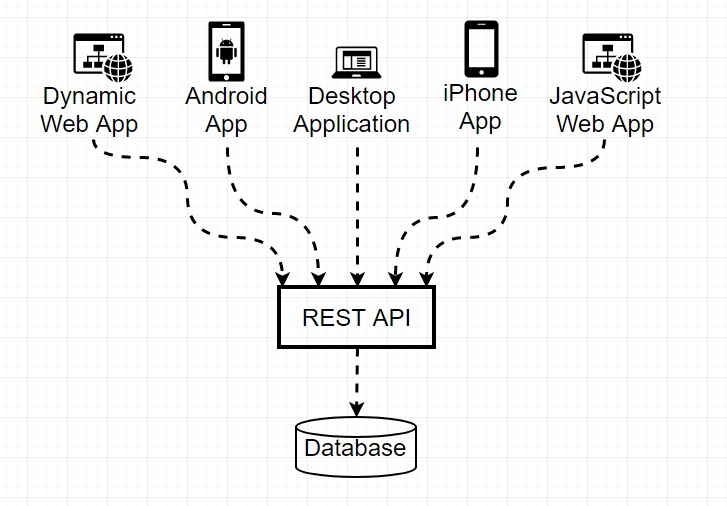
Mô hình Client-Server giúp frontend và backend phát triển độc lập, dễ mở rộng (cùng API có thể phục vụ mobile, web, đối tác), mã nguồn dễ bảo trì và bảo mật hơn nhờ mọi truy cập đều thông qua lớp kiểm soát trên Server.

### Kiến trúc RESTful API

Để đảm bảo sự giao tiếp hiệu quả giữa ứng dụng Flutter (Client) và hệ thống Django (Server), dự án Landify áp dụng kiến trúc RESTful API. Đây được xem như một “hợp đồng” chung, định nghĩa rõ ràng về tài nguyên, hành động và định dạng dữ liệu, nhờ đó hai phía có thể phát triển độc lập nhưng vẫn đảm bảo tương tác chính xác.

Trong dự án, mọi thực thể dữ liệu (tin đăng, người dùng, đánh giá…) đều được coi là tài nguyên, được gắn định danh duy nhất qua URI (endpoint). Client sử dụng các phương thức HTTP như GET, POST, PATCH hay DELETE để thực hiện các thao tác, và phía Server sẽ ánh xạ trực tiếp các yêu cầu này đến logic nghiệp vụ thông qua ViewSet của Django Rest Framework. Dữ liệu trao đổi luôn tuân theo định dạng JSON, được xử lý qua các lớp Serializer để chuyển đổi mượt mà giữa đối tượng Python và dữ liệu mà Flutter hiểu được.

Việc lựa chọn RESTful API mang lại nhiều lợi ích: cho phép frontend và backend phát triển song song, nâng cao tính linh hoạt đa nền tảng (dễ dàng mở rộng sang web, ứng dụng đối tác), và tận dụng tính chất stateless để đảm bảo khả năng mở rộng hệ thống. Đồng thời, nhờ dựa trên các chuẩn web phổ biến như HTTP và JSON, API của Landify dễ dàng tích hợp với các công cụ và dịch vụ khác trong hệ sinh thái công nghệ hiện nay.



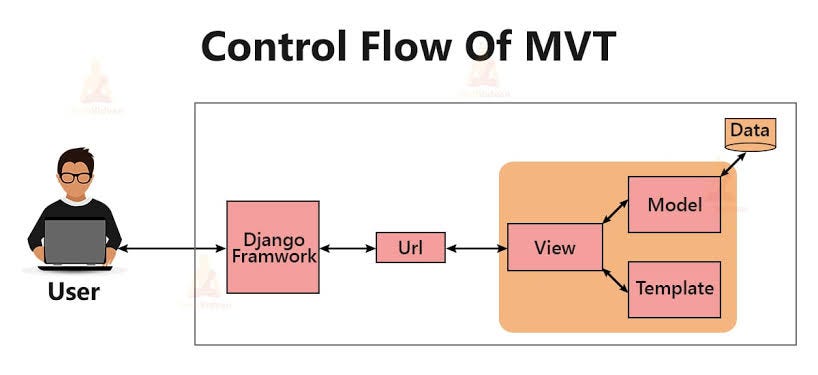
Hình . Tính đa dụng của một hệ thống REST

### Mô hình MVT (Model-View-Template) trong Django

Backend của Landify được phát triển trên Django, theo mô hình MVT (Model–View–Template) – một biến thể của MVC. Cách tiếp cận này giúp tổ chức mã nguồn rõ ràng, dễ bảo trì và mở rộng.

* Model: định nghĩa cấu trúc dữ liệu và tương tác với cơ sở dữ liệu PostgreSQL thông qua Django ORM. Ví dụ: các lớp Listing, User, Review trong models.py.
* View: xử lý logic nghiệp vụ, nhận request từ Client, làm việc với Model rồi trả về response. Trong Landify, điều này do các ViewSet của DRF đảm nhiệm.
* Template (Serializer): thay vì HTML, Landify sử dụng Serializer để chuyển đổi dữ liệu Python thành JSON (và ngược lại) nhằm trao đổi với ứng dụng Flutter.

Luồng xử lý cơ bản: Client gửi request → Django định tuyến đến View → View thao tác với Model → Serializer chuyển đổi dữ liệu → View trả response JSON về cho Client.



Hình . Luồng hoạt động của mô hình MVT

Việc tuân thủ MVT giúp phân tách rõ ràng dữ liệu, logic và trình bày, tăng tính tái sử dụng, cho phép nhiều lập trình viên làm việc song song và giúp hệ thống dễ dàng nâng cấp trong tương lai.

### Kiến trúc BLoC (Business Logic Component) trong Flutter

Trong ứng dụng Landify, việc quản lý trạng thái được thực hiện theo kiến trúc BLoC (Business Logic Component) nhằm tách biệt hoàn toàn logic nghiệp vụ với giao diện người dùng. Cách tiếp cận này giúp mã nguồn dễ kiểm thử, bảo trì và mở rộng.

Nguyên tắc hoạt động: UI phát ra Event → BLoC xử lý logic và gọi đến Repository/API → BLoC phát ra State mới → UI lắng nghe và tự động rebuild để phản ánh trạng thái. Nhờ luồng dữ liệu một chiều, việc dự đoán và gỡ lỗi trở nên đơn giản, rõ ràng.

Dự án sử dụng các thư viện hỗ trợ:

* flutter\_bloc để cung cấp BLoC và rebuild UI khi state thay đổi.
* freezed để định nghĩa Event/State bất biến gọn gàng, an toàn.

Ví dụ: khi người dùng chọn Đăng xuất, AuthBloc nhận Event AuthLogoutRequested, gọi AuthenticationRepository xử lý, rồi phát ra AuthState.unauthenticated. Ứng dụng sẽ tự động điều hướng về LoginPage.

Kiến trúc BLoC được áp dụng xuyên suốt (AuthBloc, WishlistBloc, ListingDetailBloc…), mang lại lợi ích:

* Dễ kiểm thử vì logic tách khỏi UI.
* Dễ bảo trì & mở rộng khi mỗi tính năng có BLoC riêng.
* Dễ dự đoán nhờ luồng dữ liệu đơn hướng, dễ truy vết Event–State.

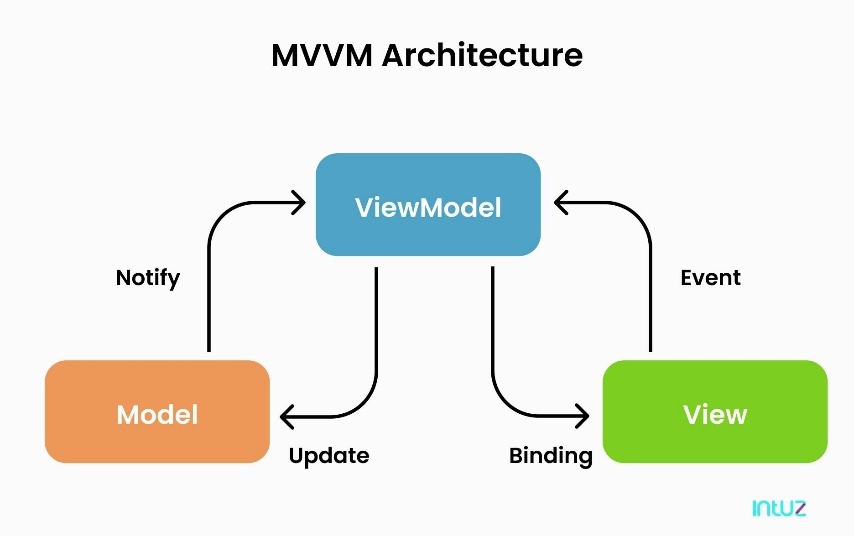
### Mô hình MVVM (Model-View-ViewModel)

Bên cạnh kiến trúc BLoC, dự án Landify còn áp dụng mô hình MVVM (Model–View–ViewModel) để giữ cho tầng trình bày (Presentation Layer) gọn gàng và tách biệt.

Data Models: đại diện dữ liệu thô từ API (JSON), ánh xạ 1–1 với key trả về, chỉ chứa parse từ JSON sang đối tượng Dart, không có logic hiển thị.

View Models: nhận Data Models, “chế biến” và định dạng lại dữ liệu để UI sử dụng trực tiếp. Logic trình bày (ví dụ: đổi createdDate thành “Đăng hôm qua”) được đặt tại đây, thường được tạo trong BLoC trước khi phát State mới.

Luồng dữ liệu: API → Repository (Data Models) → BLoC (View Models) → UI.



Hình . Kiến trúc của mô hình MVVM

Lợi ích:

UI đơn giản: chỉ hiển thị ViewModel, không cần format dữ liệu.

BLoC sạch: tập trung vào nghiệp vụ, không dính logic hiển thị.

Dễ bảo trì & kiểm thử: tách biệt rõ ràng, dễ viết unit test cho ViewModel.

Tái sử dụng tốt: một Data Model có thể phục vụ nhiều ViewModel khác nhau.

Nhìn chung MVVM giúp dự án có cấu trúc rõ ràng, dễ mở rộng và đảm bảo chất lượng khi phát triển lâu dài.

## Công nghệ phía máy chủ (Backend)

Việc lựa chọn công nghệ cho phía máy chủ (backend) là một quyết định quan trọng, ảnh hưởng trực tiếp đến khả năng mở rộng, bảo trì và tốc độ phát triển của dự án. Với dự án Landify, hệ sinh thái Python và Django đã được lựa chọn làm nền tảng cốt lõi.

### Ngôn ngữ Python

Python là ngôn ngữ lập trình bậc cao, thông dịch, hướng đối tượng, nổi tiếng nhờ cú pháp đơn giản, dễ đọc và cộng đồng hỗ trợ rộng lớn. Toàn bộ logic phía máy chủ, từ xử lý HTTP, tương tác cơ sở dữ liệu đến tác vụ nền, đều được viết bằng Python.

Ưu điểm chính:

* Phát triển nhanh nhờ cú pháp rõ ràng.
* Hệ sinh thái phong phú với nhiều thư viện, đặc biệt là Django.
* Khả năng mở rộng tốt, phù hợp cả cho hệ thống lớn.
* Cộng đồng mạnh, dễ tìm giải pháp khi gặp vấn đề.

### Framework Django và Django Rest Framework (DRF)

Để hiện thực hóa logic backend, Landify sử dụng **Django** – web framework bậc cao của Python với triết lý *“batteries-included”*. Django cung cấp:

* ORM: Tương tác cơ sở dữ liệu bằng Python thay cho SQL, vừa dễ đọc vừa an toàn.
* Admin Panel: Tự động sinh giao diện quản trị, hỗ trợ quản lý dữ liệu mà không cần xây dashboard riêng.
* Bảo mật: Tích hợp sẵn các cơ chế chống XSS, CSRF, SQL Injection.

Bên cạnh đó, Django Rest Framework (DRF) được dùng để xây dựng API:

* Serializer: Chuyển đổi dữ liệu giữa đối tượng Python và JSON.
* ViewSet: Gom nhóm logic CRUD, ánh xạ trực tiếp sang các phương thức HTTP, giúp mã ngắn gọn và có tổ chức.

### Hệ quản trị Cơ sở dữ liệu PostgreSQL và PostGIS

#### Giới thiệu PostgreSQL và lý do lựa chọn

PostgreSQL là hệ quản trị cơ sở dữ liệu quan hệ đối tượng mã nguồn mở, nổi bật về độ ổn định, tuân thủ SQL và hỗ trợ nhiều tính năng nâng cao. Landify chọn PostgreSQL vì:

* Độ tin cậy & toàn vẹn dữ liệu: Đảm bảo tính ACID (Atomicity, Consistency, Isolation, Durability); giữ dữ liệu luôn nhất quán kể cả khi lỗi hệ thống.
* Hiệu năng & khả năng mở rộng: Hỗ trợ indexing, partitioning, xử lý đồng thời tốt, phù hợp khi lượng người dùng tăng.
* Hỗ trợ dữ liệu hiện đại: JSONB, Array và extension như PostGIS, giúp hệ thống linh hoạt và mạnh mẽ hơn.

#### Giới thiệu PostGIS và tầm quan trọng trong việc xử lý dữ liệu địa lý

PostGIS là extension quan trọng của PostgreSQL, cung cấp kiểu dữ liệu và hàm chuyên dụng cho xử lý không gian địa lý. Trong Landify, PostGIS giữ vai trò cốt lõi:

* Lưu trữ tọa độ chính xác: Sử dụng PointField (srid=4326) để quản lý vị trí bất động sản theo chuẩn WGS 84.
* Truy vấn không gian mạnh mẽ: Hỗ trợ tìm kiếm lân cận, lọc theo khu vực, tính khoảng cách – các tính năng thiết yếu cho ứng dụng bất động sản.
* Tích hợp với Django (GeoDjango): Cho phép viết truy vấn địa lý bằng ORM Python quen thuộc, vừa tiện lợi vừa ít lỗi.

Nhờ đó, PostgreSQL + PostGIS không chỉ đáp ứng tác vụ dữ liệu thông thường mà còn tối ưu cho các bài toán vị trí – yếu tố then chốt của Landify.

### Hàng đợi tác vụ (Task Queue) với Celery và Redis

Trong một ứng dụng web hiện đại như Landify, không phải tất cả các công việc đều có thể và nên được xử lý ngay lập tức trong luồng yêu cầu-phản hồi chính. Một số tác vụ có thể tốn nhiều thời gian để hoàn thành, ví dụ như gọi đến một API của bên thứ ba, xử lý hình ảnh, hoặc gửi thông báo. Nếu thực hiện các công việc này một cách đồng bộ, người dùng sẽ phải chờ đợi rất lâu để nhận được phản hồi từ ứng dụng, gây ra trải nghiệm người dùng kém.

Để giải quyết vấn đề này, dự án Landify đã triển khai một hệ thống hàng đợi tác vụ bất đồng bộ bằng cách kết hợp Celery và Redis.

#### Vai trò của Celery trong việc xử lý tác vụ bất đồng bộ

**Celery** đóng vai trò hàng đợi tác vụ, nhận công việc từ Django và chuyển cho các Worker xử lý ở nền, giúp API phản hồi nhanh hơn. Trong dự án Landify:

**Kiểm tra spam tin đăng:** check\_listing\_for\_spam.delay(listing.id) chạy nền để không chặn người dùng.

**Gửi thông báo real-time**: notify\_admins\_of\_new\_protest.delay(protest.id) gửi cho nhiều admin mà không làm chậm request.

**Tác vụ định kỳ:** update\_total\_listings\_count chạy 23:00 hằng ngày để cập nhật thống kê.

#### Vai trò của Redis

Redis trong dự án được dùng làm:

Message Broker & Result Backend cho Celery: lưu trữ và phân phối tác vụ đến Worker.

Cache Backend cho Django: lưu tạm kết quả truy vấn/dữ liệu ít thay đổi để tăng tốc phản hồi và giảm tải cho PostgreSQL.

## Công nghệ phía Giao diện Người dùng (Frontend - Mobile App)

Phía Client của hệ thống Landify là một ứng dụng di động, đóng vai trò là điểm tương tác trực tiếp và duy nhất của người dùng cuối với sản phẩm. Để xây dựng một ứng dụng có giao diện hiện đại, trải nghiệm mượt mà và hoạt động ổn định trên cả hai hệ điều hành phổ biến là iOS và Android, hệ sinh thái công nghệ của Flutter đã được lựa chọn.

### Framework Flutter và Ngôn ngữ Dart

Flutter là một bộ công cụ giao diện người dùng (UI toolkit) mã nguồn mở được phát triển bởi Google. Nó cho phép các nhà phát triển xây dựng các ứng dụng đẹp mắt, được biên dịch tự nhiên (natively compiled) cho di động, web, và máy tính để bàn từ một cơ sở mã nguồn duy nhất.

Ưu điểm của Flutter:

* Đa nền tảng: Tiết kiệm thời gian và chi phí nhờ viết một lần, chạy trên nhiều hệ điều hành.
* Hiệu năng cao: Dart được biên dịch thành mã máy gốc ARM, mang lại trải nghiệm mượt mà gần như ứng dụng native.
* UI linh hoạt: Bộ widget phong phú, tùy biến theo Material Design và Cupertino.
* Hot Reload: Giúp phát triển nhanh, thử nghiệm và sửa lỗi tức thì.

Dart là ngôn ngữ hướng đối tượng, cú pháp rõ ràng, hệ thống kiểu tĩnh mạnh mẽ. Nó hỗ trợ cả JIT (phục vụ Hot Reload khi phát triển) và AOT (biên dịch trước khi phát hành để tối ưu tốc độ và hiệu năng).

### Thư viện Quản lý Trạng thái

Trong Landify, quản lý trạng thái theo kiến trúc **BLoC** được hỗ trợ bởi ba thư viện chính:

* **flutter\_bloc**: Thư viện cốt lõi triển khai BLoC, tách biệt logic và UI. Dùng BlocProvider để cung cấp BLoC, BlocBuilder để rebuild UI theo State mới, và BlocListener để xử lý hành động một lần (như điều hướng, hiển thị thông báo).
* **equatable**: Hỗ trợ so sánh đối tượng theo giá trị thuộc tính thay vì tham chiếu, giúp tối ưu việc kiểm tra thay đổi State và tránh rebuild không cần thiết.
* **freezed**: Sinh mã cho các lớp dữ liệu bất biến, cung cấp sẵn copyWith, toString, ==, hashCode, cùng cú pháp map và when để xử lý sealed class an toàn và ngắn gọn.

Sự kết hợp này mang lại kiến trúc quản lý trạng thái rõ ràng, ổn định, dễ bảo trì và giảm thiểu mã thừa cho ứng dụng.

### Các Dịch vụ và Công nghệ của Bên thứ ba

Để xây dựng một ứng dụng hiện đại với đầy đủ các tính năng phức tạp mà không cần phải phát triển mọi thứ từ đầu, dự án Landify đã tích hợp một cách chiến lược các dịch vụ của bên thứ ba. Việc này không chỉ giúp rút ngắn thời gian phát triển mà còn tận dụng được sức mạnh và sự ổn định của các nền tảng đã được kiểm chứng trên quy mô lớn.

### Hệ sinh thái Firebase

Authentication: Là nền tảng xác thực chính. Người dùng đăng nhập/đăng ký qua OTP hoặc Google Sign-In. Sau khi xác thực thành công, Firebase trả về ID Token. Token này được backend Django xác minh bằng Firebase Admin SDK, đồng thời tự động ánh xạ tới tài khoản trong PostgreSQL. Cách làm này vừa an toàn vừa giảm tải cho backend.

Cloud Firestore: Cơ sở dữ liệu NoSQL đồng bộ thời gian thực, dùng cho:

* Thông báo (Notifications): Khi backend cần gửi một thông báo (ví dụ: "Tin đăng của bạn đã được duyệt", "Có người đặt lịch hẹn mới"), nó sẽ ghi một document mới vào collection của người dùng trên Firestore. Ứng dụng Flutter, vốn đang lắng nghe (listen) những thay đổi trên collection này, sẽ nhận được dữ liệu gần như ngay lập tức và hiển thị thông báo trong ứng dụng, tạo ra trải nghiệm liền mạch.
* Tín hiệu cuộc gọi Video (Call Signaling): Firestore đóng vai trò là kênh tín hiệu trung gian cho tính năng gọi video (WebRTC). Khi Người A gọi Người B, ứng dụng sẽ tạo một "phòng chờ" (document) trên Firestore. Ứng dụng của Người B, vốn đang lắng nghe, sẽ nhận được tín hiệu này và hiển thị giao diện cuộc gọi đến. Các trạng thái như chấp nhận, từ chối, hay kết thúc cũng được cập nhật và đồng bộ giữa hai thiết bị thông qua document này.

Remote Config: Cho phép thay đổi nội dung động (như danh mục bất động sản, tiện ích trên trang chủ) từ xa thông qua JSON trên dashboard Firebase, không cần phát hành phiên bản ứng dụng mới.

**2.4.2. Cloudinary**

Cloudinary được lựa chọn làm giải pháp quản lý media (hình ảnh, video) toàn diện cho dự án. Thay vì lưu ảnh/video trực tiếp lên server Django, file được tải thẳng lên Cloudinary và trả về URL tối ưu để lưu trong DB. Ngoài giảm tải lưu trữ và băng thông, Cloudinary còn hỗ trợ resize, crop, nén ảnh và phân phối qua CDN toàn cầu, giúp hình ảnh/video tải nhanh ở mọi nơi.

### Tích hợp API cho các tính năng nâng cao

Để cung cấp các tính năng vượt trội và chuyên biệt, Landify đã tích hợp API từ các nhà cung cấp dịch vụ hàng đầu.

FPT.AI được sử dụng để triển khai tính năng eKYC (electronic Know Your Customer), một quy trình xác thực danh tính người dùng cao cấp.

Bước 1: Người dùng chụp CCCD, API FPT OCR trích xuất thông tin.

Bước 2: Người dùng quay video khuôn mặt, API Liveness Detection xác minh “người thật” và so khớp với CCCD.

Twilio là một nền tảng giao tiếp đám mây, được sử dụng trong dự án cho chức năng gửi tin nhắn SMS OTP (One-Time Password). Khi người dùng đăng ký hoặc đăng nhập bằng số điện thoại, backend Django sẽ gọi đến API của Twilio để gửi một tin nhắn SMS chứa mã xác thực đến số điện thoại của người dùng, đảm bảo rằng người dùng thực sự sở hữu số điện thoại đó.

Agora: Hỗ trợ gọi video thời gian thực. Backend tạo token từ Agora, client dùng token để kết nối vào cùng một kênh và thực hiện cuộc gọi với hạ tầng tối ưu về độ trễ và chất lượng truyền tải.

# HỆ THỐNG GIAO LƯU MUA BÁN BẤT ĐỘNG SẢN

*Chương này trình bày chi tiết về hệ thống giao dịch bất động sản Landify được xây dựng trong khuôn khổ đồ án. Nội dung chương sẽ tập trung vào việc giới thiệu tổng quan đề tài, sau đó đi sâu vào phân tích kiến trúc hệ thống, diễn giải các thành phần công nghệ và cơ chế hoạt động phối hợp giữa chúng.*

## Giới thiệu đề tài

Hệ thống Landify được phát triển với mục tiêu xây dựng một nền tảng bất động sản toàn diện, hiện đại và đáng tin cậy, giải quyết các nhu cầu đa dạng của cả người mua, người bán, và nhà môi giới. Đây không chỉ là một kênh đăng tin đơn thuần, mà còn là một hệ sinh thái tích hợp đầy đủ các công cụ và tính năng cần thiết để nâng cao trải nghiệm người dùng và tối ưu hóa quy trình giao dịch.

* Về phía người dùng, Landify mang đến một trải nghiệm mượt mà và trực quan thông qua ứng dụng di động (xây dựng bằng Flutter). Người dùng có thể dễ dàng truy cập vào một kho tin đăng phong phú, mỗi tin đăng đều được cung cấp thông tin chi tiết, đa dạng từ hình ảnh, video, thông số kỹ thuật, cho đến vị trí chính xác trên bản đồ. Hệ thống tìm kiếm và bộ lọc thông minh cho phép người dùng nhanh chóng tìm thấy bất động sản phù hợp với nhu cầu. Đặc biệt, để xây dựng một môi trường giao dịch an toàn và minh bạch, dự án đã tích hợp các tính năng cao cấp như:
  + Xác thực danh tính (eKYC): Tăng cường sự tin cậy giữa các bên tham gia.
  + Tương tác thời gian thực: Cho phép người dùng Chat, Đặt lịch hẹn và cả Gọi video trực tiếp trên ứng dụng.
  + Xây dựng cộng đồng: Tích hợp các tính năng mạng xã hội như bài đăng, bình luận và đánh giá, giúp người dùng chia sẻ kinh nghiệm và tham khảo ý kiến.
* Về phía quản trị, Landify cung cấp một hệ thống quản lý tập trung và mạnh mẽ (xây dựng trên Django Admin). Đội ngũ quản trị viên có toàn quyền kiểm soát các hoạt động trên nền tảng:
  + Quản lý nội dung: Duyệt, chỉnh sửa, hoặc từ chối các tin đăng không phù hợp.
  + Quản lý người dùng: Theo dõi và quản lý thông tin tài khoản người dùng.
  + Hệ thống kiểm duyệt: Xử lý các báo cáo vi phạm từ người dùng và giải quyết các kháng nghị một cách có hệ thống.
  + Thống kê và báo cáo: Theo dõi các chỉ số quan trọng về hoạt động của hệ thống để đưa ra các quyết định vận hành phù hợp.

## Kiến trúc hệ thống

Hệ thống Landify được xây dựng trên kiến trúc Client-Server phân tán, với sự tách biệt rõ ràng giữa các thành phần frontend, backend và các dịch vụ của bên thứ ba.

Diễn giải các thành phần trong kiến trúc hệ thống & cơ chế hoạt động:

Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, hàng

Nội dung do AI tạo ra có thể không chính xác.

Hình . Sơ đồ kiến trúc tổng quan của hệ thống Landify

Các thành phần chính trong kiến trúc hệ thống bao gồm:

Backend (Server-Side): Là trung tâm xử lý của hệ thống, được xây dựng bằng Python với framework Django và Django Rest Framework (DRF). Chịu trách nhiệm xử lý toàn bộ logic nghiệp vụ, xác thực, và cung cấp RESTful API. Dữ liệu được lưu trữ và quản lý bởi hệ quản trị cơ sở dữ liệu PostgreSQL tích hợp PostGIS để xử lý thông tin địa lý. Các tác vụ nặng và bất đồng bộ được xử lý bởi Celery và Redis.

Frontend (Client-Side): Là ứng dụng di động đa nền tảng, được xây dựng bằng ngôn ngữ Dart và framework Flutter. Đây là giao diện chính mà người dùng tương tác. Ứng dụng áp dụng kiến trúc BLoC (Business Logic Component) để quản lý trạng thái và luồng dữ liệu một cách hiệu quả.

Dịch vụ của bên thứ ba (Third-Party Services): Để tăng tốc độ phát triển và tận dụng các công nghệ chuyên biệt, dự án đã tích hợp nhiều dịch vụ mạnh mẽ:

* Firebase: Được sử dụng cho việc xác thực người dùng (Authentication), các tính năng thời gian thực như Chat (Cloud Firestore), và cấu hình động cho ứng dụng (Remote Config).
* Cloudinary: Chịu trách nhiệm lưu trữ, tối ưu và phân phối toàn bộ media (hình ảnh, video) của dự án.
* Các API chuyên dụng: Tích hợp FPT.AI cho tính năng eKYC, Twilio cho việc gửi SMS OTP, và Agora cho tính năng gọi video.

Sự giao tiếp giữa ứng dụng Flutter (Client) và hệ thống Django (Server) được thực hiện hoàn toàn thông qua RESTful API theo chu trình Yêu cầu – Phản hồi.

Khi người dùng thực hiện một hành động trên ứng dụng Flutter, một yêu cầu HTTP (ví dụ GET, POST) sẽ được tạo ra và gửi đến endpoint API tương ứng trên server Django. Server nhận yêu cầu, tiến hành xác thực (nếu cần), sau đó xử lý logic nghiệp vụ và tương tác với cơ sở dữ liệu PostgreSQL. Dữ liệu sau khi được xử lý sẽ được định dạng thành JSON thông qua các lớp Serializer và gửi ngược lại cho ứng dụng Flutter trong một phản hồi HTTP. Ứng dụng Flutter nhận dữ liệu JSON này, phân tích thành các đối tượng dữ liệu và cập nhật lại giao diện người dùng để hiển thị thông tin mới.

Đối với các tính năng thời gian thực như Chat, luồng hoạt động có sự tham gia của Firestore. Client vẫn gọi API Django để khởi tạo cuộc trò chuyện, nhưng sau đó sẽ lắng nghe trực tiếp các thay đổi trên một document của Firestore để nhận và gửi tin nhắn mới, tạo ra trải nghiệm tức thời cho người dùng.

## Phân tích hệ thống

Ảnh có chứa văn bản, biểu đồ, hình vẽ, hàng

Nội dung do AI tạo ra có thể không chính xác.

Hình . Sơ đồ Use Case

### Đặc tả Use Case

#### Xác thực danh tính (eKYC)

Bảng . Đặc tả UC: Xác thực danh tính (eKYC)

|  |  |
| --- | --- |
| Use case Id | UC001 |
| Name  (Tên use case) | Xác thực danh tính (eKYC) |
| Description  (Mô tả) | Use case này cho phép Người dùng đã đăng nhập thực hiện quy trình xác minh danh tính điện tử (eKYC) hai bước. Bước một bao gồm việc chụp và trích xuất thông tin từ Căn cước công dân (CCCD). Bước hai là quay video khuôn mặt để xác thực người thật và so sánh với ảnh trên CCCD. |
| Primary Actor  (Actor chính) | Người dùng (User) |
| Secondary Actor (Actor phụ) | FPT.AI (Dịch vụ nhận dạng ký tự quang học và xác thực khuôn mặt) |
| Pre-conditions  (Tiền điều kiện) | 1. Người dùng đã đăng nhập thành công vào hệ thống Landify.  2. Người dùng đang ở màn hình lựa chọn phương thức xác thực và đã chọn bắt đầu quy trình. |
| Post-conditions  (Hậu điều kiện) | ***Thành công:***  Trạng thái tài khoản người dùng trong CSDL được cập nhật thành is\_id\_card\_verified = True và is\_identity\_verified = True.  2. Người dùng được cấp quyền truy cập vào các tính năng yêu cầu xác thực (ví dụ: đăng tin, đặt lịch hẹn, đánh giá,…).  3. Hệ thống hiển thị thông báo xác thực thành công.  ***Thất bại:***  1. Trạng thái tài khoản người dùng không thay đổi.  2. Hệ thống hiển thị thông báo lỗi cụ thể. |
| Main flows  (Luồng hoạt động) | ***Bước 1: Quét Căn cước công dân (CCCD)***  1. Use case bắt đầu khi Người dùng chọn "Bắt đầu quét CCCD" trên màn hình xác thực.  2. Hệ thống yêu cầu và kiểm tra quyền truy cập camera.  3. Hệ thống khởi tạo và hiển thị giao diện camera với một khung (overlay) để căn chỉnh CCCD.  4. Người dùng đặt mặt trước của CCCD vào trong khung và nhấn nút chụp ảnh.  5. Ứng dụng gửi tệp ảnh đã chụp đến backend  6. Backend nhận tệp ảnh và gọi đến API FPT.AI ID Recognition để trích xuất thông tin.  7. FPT.AI xử lý và trả về dữ liệu đã trích xuất (số CCCD, họ tên, ngày sinh...).  8. Backend cập nhật trạng thái is\_id\_card\_verified = True cho người dùng và trả về thông báo thành công cho ứng dụng.  9. Ứng dụng hiển thị thông báo thành công cho Bước 1 và tự động chuyển người dùng sang Bước 2.  ***Bước 2: Xác thực khuôn mặt***  10. Hệ thống khởi tạo lại giao diện camera trước với một khung (overlay) hình oval để căn chỉnh khuôn mặt.  11. Hệ thống sử dụng công nghệ Face Detection trên thiết bị để liên tục kiểm tra xem có khuôn mặt trong khung hình hay không.  12. Khi hệ thống phát hiện khuôn mặt nằm ổn định trong khung, một bộ đếm ngược 5 giây sẽ tự động bắt đầu và hệ thống bắt đầu quay video.  13. Sau khi đếm ngược kết thúc, hệ thống tự động dừng quay và có được một tệp video ngắn.  14. Ứng dụng gửi tệp video này đến backend Django.  15. Backend gọi đến API FPT.AI Liveness Detection, gửi kèm video và ảnh CCCD đã được lưu cache từ Bước 1.  16. FPT.AI phân tích video để xác thực người thật (liveness) và so sánh khuôn mặt trong video với ảnh trên CCCD.  17. Nếu cả hai điều kiện đều hợp lệ, FPT.AI trả về kết quả thành công.  18. Backend cập nhật trạng thái is\_identity\_verified = True cho người dùng và trả về phản hồi thành công.  19. Ứng dụng hiển thị thông báo "Xác minh danh tính hoàn tất!" cho người dùng.  20. Use case kết thúc. |
| Alternative flows (Luồng thay thế) | 3a. Người dùng không cấp quyền camera:  1. Tại bước 2, nếu người dùng từ chối cấp quyền camera.  2. Hệ thống hiển thị thông báo yêu cầu cấp quyền để tiếp tục.  3. Use case kết thúc.  12a. Khuôn mặt di chuyển ra khỏi khung:  1. Tại bước 12, nếu khuôn mặt người dùng di chuyển ra khỏi khung oval trong quá trình đếm ngược.  2. Bộ đếm ngược sẽ dừng lại và reset.  3. Hệ thống hiển thị lại hướng dẫn "Vui lòng giữ yên khuôn mặt trong khung".  4. Luồng hoạt động quay lại bước 11.  17a. Xác thực liveness hoặc so khớp khuôn mặt thất bại:  1. Tại bước 17, nếu FPT.AI trả về kết quả không phải người thật hoặc khuôn mặt không khớp.  2. Backend trả về thông báo lỗi cụ thể cho ứng dụng (ví dụ: "Khuôn mặt không khớp với CCCD").  3. Ứng dụng hiển thị lỗi và cho phép người dùng thực hiện lại từ bước 10. |
| Exception flows (Luồng ngoại lệ) | 6a. Không thể đọc thông tin từ CCCD:  1. Tại bước 6, nếu API của FPT.AI không thể trích xuất được thông tin từ ảnh (do ảnh quá mờ, lóa sáng).  2. Hệ thống trả về lỗi "Không thể đọc thông tin từ ảnh CCCD. Vui lòng thử lại với ảnh rõ nét hơn."  3. Luồng hoạt động quay lại bước 3.  7a. API FPT.AI gặp sự cố:  1. Tại bước 7 hoặc 17, nếu dịch vụ của FPT.AI không phản hồi hoặc báo lỗi hệ thống.  2. Backend trả về một thông báo lỗi chung (ví dụ: "Dịch vụ xác thực đang tạm thời gián đoạn. Vui lòng thử lại sau.").  3. Use case kết thúc. |

#### Tạo tin đăng

Bảng . Đặc tả UC: Tạo tin đăng

|  |  |
| --- | --- |
| Use case Id | UC002 |
| Name  (Tên use case) | Tạo tin đăng |
| Description  (Mô tả) | Use case này cho phép Người dùng đã đăng nhập thực hiện quy trình xác minh danh tính điện tử (eKYC) hai bước. Bước một bao gồm việc chụp và trích xuất thông tin từ Căn cước công dân (CCCD). Bước hai là quay video khuôn mặt để xác thực người thật và so sánh với ảnh trên CCCD. |
| Primary Actor  (Actor chính) | Người dùng (User) |
| Secondary Actor (Actor phụ) | Cloudinary |
| Pre-conditions  (Tiền điều kiện) | 1. Người dùng đã đăng nhập vào hệ thống Landify.  2. Tài khoản của người dùng đã hoàn thành xác thực danh tính (eKYC). |
| Post-conditions  (Hậu điều kiện) | **Thành công:**  1. Một bản ghi Property và Listing mới, cùng các bản ghi phụ thuộc (Location, ListingPropertyFeatureValue...) được tạo thành công trong cơ sở dữ liệu.  2. Các tệp media (hình ảnh, video) được tải lên và lưu trữ trên Cloudinary, URL được liên kết với bản ghi Property.  3. Một tác vụ nền được kích hoạt để kiểm tra nội dung tin đăng có chứa spam hay không.  4. Người dùng được điều hướng đến màn hình thông báo đăng tin thành công.  **Thất bại:**  1. Không có bản ghi nào được tạo trong cơ sở dữ liệu.  2. Hệ thống hiển thị thông báo lỗi cụ thể cho người dùng. |
| Main flows  (Luồng hoạt động) | 1. Use case bắt đầu khi Người dùng nhấn vào nút "Đăng tin" trên giao diện chính của ứng dụng.  2. Hệ thống hiển thị giao diện tạo tin đăng, bắt đầu từ ***Bước 1: Thông tin BĐS.***  3. Người dùng chọn loại nhu cầu (Bán/Cho thuê), sau đó nhập các thông tin chi tiết về bất động sản, bao gồm:  - Địa chỉ (thông qua màn hình tìm kiếm hoặc chọn trên bản đồ).  - Loại hình BĐS, diện tích, giá, đơn vị giá.  - Thông tin khác như giấy tờ pháp lý, số phòng ngủ, hướng nhà...  - Tiêu đề và mô tả chi tiết cho tin đăng.  - Thông tin liên hệ.  4. Ứng dụng thực hiện kiểm tra tính đầy đủ của dữ liệu ngay trên thiết bị khi người dùng nhập.  5. Người dùng nhấn nút "Tiếp tục". Hệ thống chuyển sang ***Bước 2: Hình ảnh & Video.***  6. Người dùng chọn và tải lên các hình ảnh từ thư viện hoặc chụp ảnh mới. Ứng dụng cho phép tải lên tối thiểu 3 ảnh.  7. Người dùng có thể sắp xếp lại thứ tự ảnh, đặt một ảnh làm ảnh đại diện, và thêm mô tả cho từng ảnh.  8. Người dùng có thể tùy chọn dán đường dẫn video từ YouTube hoặc TikTok.  9. Người dùng nhấn nút "Tiếp tục". Hệ thống chuyển sang ***Bước 3: Hoàn tất.***  10. Hệ thống hiển thị danh sách các loại tin đăng (ví dụ: Tin thường, VIP Bạc, VIP Vàng...).  11. Người dùng chọn một loại tin đăng. Ứng dụng hiển thị màn hình cấu hình chi tiết cho phép chọn số ngày đăng và ngày bắt đầu.  12. Người dùng có thể tùy chọn áp dụng mã khuyến mãi.  13. Hệ thống hiển thị tóm tắt chi phí cuối cùng.  14. Người dùng nhấn nút "Đăng tin".  15. Ứng dụng tổng hợp toàn bộ dữ liệu từ 3 bước.  16. Ứng dụng gửi một yêu cầu đến server, đính kèm payload và các tệp hình ảnh  17. Backend Django nhận yêu cầu và xử lý nghiệp.  18. Trong một yêu cầu tạo tin đăng, backend sẽ:  a. Tải các tệp media lên dịch vụ Cloudinary.  b. Tạo các bản ghi Location, Property mới.  c. Tạo bản ghi Listing chính và các bản ghi chi tiết liên quan (BuySellDetail, RentalDetail...).  19. Nếu tất cả các bước trên thành công. Backend kích hoạt một tác vụ nền Celery và trả về phản hồi thành công kèm theo dữ liệu của tin đăng vừa tạo.  20. Ứng dụng nhận được phản hồi thành công và điều hướng Người dùng đến màn hình "Đăng tin thành công".  21. Use case kết thúc. |
| Alternative flows (Luồng thay thế) | 5a. Người dùng muốn quay lại bước trước:  1. Tại Bước 2 hoặc Bước 3, người dùng nhấn nút "Quay lại".  2. Hệ thống chuyển về giao diện của bước trước đó, giữ nguyên các dữ liệu người dùng đã nhập.  15a. Dữ liệu không hợp lệ ở phía Client:  1. Tại bất kỳ bước nào, nếu người dùng nhấn "Tiếp tục" nhưng chưa điền đủ các trường bắt buộc.  2. Ứng dụng sẽ hiển thị thông báo lỗi ngay trên các trường bị thiếu và không cho phép chuyển sang bước tiếp theo cho đến khi dữ liệu hợp lệ. |
| Exception flows (Luồng ngoại lệ) | 18a. Lỗi trong quá trình xử lý ở Backend:  1. Tại bước 18, nếu có bất kỳ lỗi nào xảy ra (ví dụ: kiểm tra dữ liệu đầu vào thất bại, lỗi kết nối đến Cloudinary, lỗi ghi CSDL).  2. Phía Backend đảm bảo cơ sở dữ liệu đảm bảo không có dữ liệu nào được lưu lại một phần.  3. Backend trả về một phản hồi lỗi HTTP (ví dụ: 400 Bad Request) kèm theo thông báo lỗi chi tiết.  4. Ứng dụng hiển thị thông báo lỗi cho người dùng.  5. Use case kết thúc trong trạng thái thất bại. |

#### Hợp tác môi giới

Bảng . Đặc tả UC: Hợp tác môi giới

|  |  |
| --- | --- |
| Use case Id | UC003 |
| Name  (Tên use case) | Hợp tác môi giới |
| Description  (Mô tả) | Use case này mô tả quy trình một Người dùng (Nhà môi giới) gửi một lời đề nghị hợp tác chính thức đến Chủ tin đăng ngay bên trong giao diện trò chuyện (chat). Hệ thống sẽ tạo một bản ghi yêu cầu ở backend và đồng thời gửi một tin nhắn đặc biệt vào cuộc trò chuyện trên Firestore để Chủ tin đăng có thể phản hồi |
| Primary Actor  (Actor chính) | Người dùng (User) - trong vai trò Nhà môi giới và Chủ tin đăng |
| Secondary Actor (Actor phụ) | Cloud Firestore |
| Pre-conditions  (Tiền điều kiện) | 1. Cả Nhà môi giới và Chủ tin đăng đều đã đăng nhập vào hệ thống.  2. Nhà môi giới đã bắt đầu một cuộc trò chuyện với Chủ tin đăng về một tin đăng cụ thể. |
| Post-conditions  (Hậu điều kiện) | **Thành công:**  1. Một bản ghi Cooperation được tạo trong CSDL PostgreSQL với trạng thái PENDING.  2. Một tin nhắn đặc biệt (loại cooperation\_request) được tạo trong cuộc trò chuyện tương ứng.  3. Trạng thái của bản ghi Cooperation được cập nhật thành ACCEPTED hoặc REJECTED sau khi Chủ tin đăng phản hồi.  4. Giao diện tin nhắn đặc biệt trên cả hai thiết bị được cập nhật để phản ánh trạng thái mới  **Thất bại:**  1. Yêu cầu hợp tác không được tạo.  2. Hệ thống hiển thị thông báo lỗi. |
| Main flows  (Luồng hoạt động) | ***Phần 1: Nhà môi giới gửi yêu cầu từ trong Chat***  1. Use case bắt đầu khi Nhà môi giới đang trong màn hình chat với Chủ tin đăng, nhấn vào nút menu mở rộng (dấu cộng).  2. Hệ thống hiển thị danh sách các hành động, bao gồm "Ngỏ lời hợp tác môi giới".  3. Nhà môi giới chọn "Ngỏ lời hợp tác môi giới".  4. Ứng dụng Flutter gửi một yêu cầu đến server kèm theo thông tin đăng liên quan đến cuộc trò chuyện.  5. Backend Django nhận yêu cầu và xử lý logic:  a. Xác định agent (người yêu cầu) và owner (chủ tin đăng).  b. Kiểm tra các điều kiện nghiệp vụ (không gửi yêu cầu trùng lặp...).  6. Nếu hợp lệ, backend tạo một bản ghi Cooperation mới trong CSDL với trạng thái chờ xác nhận.  7. Backend trả về phản hồi thành công chứa dữ liệu của yêu cầu vừa tạo.  8. Đồng thời, sau khi nhận được phản hồi thành công từ backend, ứng dụng của Nhà môi giới sẽ gọi đến Cloud Firestore để thêm một tin nhắn mới vào cuộc trò chuyện. Tin nhắn này có dạng đặc biệt, tham chiếu đến bản ghi Cooperation vừa tạo ở bước 6.  9. Giao diện chat của cả hai người dùng ngay lập tức hiển thị một thẻ "Đề nghị hợp tác" đang ở trạng thái "Chờ phản hồi".  ***Phần 2: Chủ tin đăng phản hồi ngay trên giao diện*** ***Chat***  10. Chủ tin đăng nhìn thấy thẻ "Đề nghị hợp tác" trong cuộc trò chuyện và nhấn vào nút "Chấp thuận" hoặc "Từ chối" ngay trên thẻ đó.  11. Ứng dụng của Chủ tin đăng gửi một yêu cầu đến server, kèm theo thông tin chấp nhận hay từ chối yêu cầu hợp tác này.  12. Backend Django nhận yêu cầu, xác thực quyền và cập nhật trạng thái của bản ghi lời mời hợp tác tương ứng  13. Backend kích hoạt tác vụ nền để gửi thông báo đến Nhà môi giới.  14. Backend trả về phản hồi thành công.  15. Nhờ cơ chế real-time của Firestore, giao diện thẻ "Đề nghị hợp tác" trên cả hai máy của người dùng sẽ tự động cập nhật sang trạng thái "Đã chấp nhận" hoặc "Đã từ chối" mà không cần tải lại.  16. Use case kết thúc. |
| Alternative flows (Luồng thay thế) | 4a. Gửi yêu cầu không hợp lệ:  1. Tại bước 4, nếu backend xác định yêu cầu không hợp lệ (ví dụ: đã có yêu cầu đang chờ).  2. Backend trả về lỗi 400 Bad Request.  3. Ứng dụng của Nhà môi giới sẽ không gửi tin nhắn lên Firestore và thay vào đó hiển thị một thông báo lỗi (ví dụ: "Bạn đã gửi yêu cầu này trước đó").  10a. Chủ tin đăng không phản hồi qua Chat:  1. Tại bước 10, nếu Chủ tin đăng không tương tác với thẻ trong chat mà vào khu vực quản lý riêng.  2. Chủ tin đăng vẫn có thể phản hồi yêu cầu thông qua danh sách quản lý chung (như mô tả trong luồng UC-05 trước đó). Hệ thống vẫn hoạt động bình thường, tuy nhiên tin nhắn trong chat sẽ không được cập nhật trạng thái. |
| Exception flows (Luồng ngoại lệ) | 8a. Lỗi kết nối đến Firestore:  1. Tại bước 8, sau khi đã tạo yêu cầu thành công ở backend nhưng ứng dụng gặp lỗi kết nối và không thể gửi tin nhắn lên Firestore.  2. Yêu cầu hợp tác vẫn tồn tại ở backend và có thể được xử lý ở khu vực quản lý riêng, nhưng sẽ không hiển thị trong cuộc trò chuyện.  12a. Người không có thẩm quyền cố gắng phản hồi:  1. Tại bước 12, nếu một người dùng không phải là Chủ tin đăng (ví dụ: Nhà môi giới) cố gắng giả mạo yêu cầu phản hồi.  2. Backend sẽ kiểm tra quyền và trả về lỗi ngăn cấm.  3. Use case kết thúc. |

## Thiết kế hệ thống

### Sơ đồ hoạt động

#### Xác thực danh tính

Ảnh có chứa văn bản, ảnh chụp màn hình, Song song, biểu đồ

Nội dung do AI tạo ra có thể không chính xác.

Hình . Sơ đồ hoạt động: Xác thực danh tính (eKYC)

#### Tạo tin đăng

Ảnh có chứa văn bản, biểu đồ, ảnh chụp màn hình, Song song

Nội dung do AI tạo ra có thể không chính xác.

Hình . Sơ đồ hoạt động: Tạo tin đăng

#### Hợp tác môi giới

Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, Song song

Nội dung do AI tạo ra có thể không chính xác.

Hình . Sơ đồ hoạt động: Hợp tác môi giới

### Thiết kế cơ sở dữ liệu

#### Lược đồ CSDL

Ảnh có chứa văn bản, biểu đồ, Kế hoạch, bản đồ

Nội dung do AI tạo ra có thể không chính xác.

Hình . Sơ đồ lớp (Class Diagram)

#### Giải thích các mối quan hệ trong lược đồ

users (Quản lý Người dùng và Hồ sơ)

* User – UserProfile : Quan hệ 1 – 1
  + Một User chỉ có duy nhất một UserProfile để lưu trữ thông tin mở rộng.
  + Một UserProfile chỉ thuộc về một User.
* User – Subscription (Người theo dõi - follower): Quan hệ 1 – n
  + Một User có thể theo dõi nhiều người khác (tạo ra nhiều bản ghi Subscription mà họ là follower).
  + Một bản ghi Subscription chỉ có một follower.
* User – User (Người được theo dõi - following) qua Subscription: Quan hệ n – n
  + Một User có thể được theo dõi bởi nhiều người khác (tạo ra nhiều bản ghi Subscription mà họ là following).
  + Một bản ghi Subscription chỉ có một following.

properties (Quản lý Bất động sản)

* User – Property : Quan hệ 1 – n
  + Một User (chủ sở hữu) có thể sở hữu một hoặc nhiều Property.
  + Một Property chỉ thuộc về một User.
* Property – PropertyMedia : Quan hệ 1 – n
  + Một Property có thể có nhiều hình ảnh/video (PropertyMedia).
  + Một PropertyMedia chỉ thuộc về một Property.
* PropertyType – Property : Quan hệ 1 – n
  + Một PropertyType (ví dụ: "Căn hộ") có thể được áp dụng cho nhiều Property.
  + Một Property chỉ thuộc về một PropertyType.
* Location – Property : Quan hệ 1 – 1
  + Một Property chỉ có một Location duy nhất.
  + Một Location chỉ được gán cho một Property.
* Direction – Property : Quan hệ 1 – n
  + Một Direction (ví dụ: "Hướng Đông") có thể được áp dụng cho nhiều Property.
  + Một Property chỉ có một Direction.
* LegalStatus – Property : Quan hệ 1 – n
  + Một LegalStatus (ví dụ: "Sổ hồng") có thể được áp dụng cho nhiều Property.
  + Một Property chỉ có một LegalStatus.
* PropertyType – PropertyFeature : Quan hệ n – n
  + Một PropertyType có thể áp dụng nhiều PropertyFeature (ví dụ: "Căn hộ" có "Số phòng ngủ", "Hướng ban công").
  + Một PropertyFeature có thể được áp dụng cho nhiều PropertyType (ví dụ: "Số phòng ngủ" có thể áp dụng cho cả "Căn hộ" và "Nhà phố").

listings (Quản lý Tin đăng)

* Property – Listing : Quan hệ 1 – n
  + Một Property có thể có nhiều Listing được đăng tải theo thời gian.
  + Một Listing chỉ liên quan đến một Property.
* User – Listing : Quan hệ 1 – n
  + Một User có thể tạo một hoặc nhiều Listing.
  + Một Listing chỉ do một User tạo ra.
* ListingType – Listing : Quan hệ 1 – n
  + Một ListingType (ví dụ: "Mua bán") có thể được áp dụng cho nhiều Listing.
  + Một Listing chỉ thuộc một ListingType.
* UnitPrice – Listing : Quan hệ 1 – n
  + Một UnitPrice (ví dụ: "tỷ") có thể được sử dụng trong nhiều Listing.
  + Một Listing chỉ có một UnitPrice.
* Listing – PropertyFeature : Quan hệ n – n (thông qua bảng ListingPropertyFeatureValue)
  + Một Listing có thể có nhiều PropertyFeature với các giá trị cụ thể.
  + Một PropertyFeature có thể xuất hiện trong nhiều Listing.
* Listing – ListingVip : Quan hệ 1 – 1
  + Một Listing chỉ có tối đa một trạng thái VIP (ListingVip) tại một thời điểm.
  + Một bản ghi ListingVip chỉ thuộc về một Listing.
* VipType – ListingVip : Quan hệ 1 – n
  + Một VipType (ví dụ: "VIP Kim Cương") có thể được áp dụng cho nhiều lần mua VIP (ListingVip).
  + Một bản ghi ListingVip chỉ thuộc một VipType.
* Listing – BuySellDetail / RentalDetail / ProjectDetail : Quan hệ 1 – 1
  + Một Listing chỉ có thể có một bộ thông tin chi tiết duy nhất (hoặc Mua/Bán, hoặc Cho thuê, hoặc Dự án).
  + Một bản ghi chi tiết chỉ thuộc về một Listing.
* PromotionRule – VipType : Quan hệ n – n
  + Một PromotionRule có thể áp dụng cho nhiều VipType.
  + Một VipType có thể nằm trong nhiều PromotionRule.
* User – UserPromotion: Quan hệ 1 – n
  + Một User có thể nhận được nhiều UserPromotion.
  + Một UserPromotion chỉ được cấp cho một User.
* PromotionRule – UserPromotion : Quan hệ 1 – n
  + Một PromotionRule có thể được dùng để tạo ra nhiều UserPromotion cho nhiều người dùng khác nhau.
  + Một UserPromotion chỉ thuộc về một PromotionRule.
* Listing – UserPromotion : Quan hệ 0..1 – n
  + Một Listing có thể đã sử dụng một UserPromotion.
  + Một UserPromotion có thể đã được sử dụng trên một Listing (trường used\_on\_listing có thể là null).

interactions (Quản lý Tương tác)

* User – Review : Quan hệ 1 – n
  + Một User có thể viết nhiều Review.
  + Một Review chỉ do một User viết.
* Property – Review : Quan hệ 1 – n
  + Một Property có thể có nhiều Review.
  + Một Review chỉ đánh giá cho một Property.
* User – Wishlist : Quan hệ 1 – n
  + Một User có thể có nhiều tin đăng trong Wishlist của mình.
  + Một bản ghi Wishlist chỉ thuộc về một User.
* Listing – Wishlist : Quan hệ 1 – n
  + Một Listing có thể được lưu bởi nhiều User (có trong nhiều Wishlist).
  + Một bản ghi Wishlist chỉ lưu một Listing.
* User – Appointment : Quan hệ 1 – n
  + Một User có thể đặt nhiều Appointment.
  + Một Appointment chỉ do một User đặt.
* Listing – Appointment : Quan hệ 1 – n
  + Một Listing có thể có nhiều Appointment được đặt.
  + Một Appointment chỉ được đặt cho một Listing.
* Chat – User : Quan hệ n – n
  + Một Chat có nhiều User tham gia.
  + Một User có thể tham gia nhiều Chat.
* Listing – Chat : Quan hệ 0..1 – n
  + Một Listing có thể có nhiều cuộc Chat liên quan (từ nhiều người mua khác nhau).
  + Một cuộc Chat có thể (hoặc không) thuộc về một Listing cụ thể.
* User (agent) – Cooperation : Quan hệ 1 – n
  + Một User (trong vai trò môi giới) có thể gửi nhiều yêu cầu Cooperation.
  + Một Cooperation chỉ có một agent.
* User (owner) – Cooperation : Quan hệ 1 – n
  + Một User (trong vai trò chủ tin) có thể nhận nhiều yêu cầu Cooperation.
  + Một Cooperation chỉ có một owner.
* Listing – Cooperation : Quan hệ 1 – n
  + Một Listing có thể nhận được nhiều yêu cầu Cooperation từ các môi giới khác nhau.
  + Một Cooperation chỉ được tạo cho một Listing.

social (Quản lý Mạng xã hội)

* User – Post : Quan hệ 1 – n
  + Một User có thể tạo nhiều Post.
  + Một Post chỉ do một User tạo.
* Post – Comment : Quan hệ 1 – n
  + Một Post có thể có nhiều Comment.
  + Một Comment chỉ thuộc về một Post.
* User – Comment : Quan hệ 1 – n
  + Một User có thể viết nhiều Comment.
  + Một Comment chỉ do một User viết.
* Post – Reaction : Quan hệ 1 – n
  + Một Post có thể có nhiều Reaction.
  + Một Reaction chỉ thuộc về một Post.
* User – Reaction : Quan hệ 1 – n
  + Một User có thể có nhiều Reaction trên các bài đăng khác nhau.
  + Một Reaction chỉ do một User thực hiện.

moderation (Quản lý Kiểm duyệt)

* User (reporter) – Report : Quan hệ 1 – n
  + Một User có thể gửi nhiều Report.
  + Một Report chỉ do một User (reporter) gửi.
* User (protester) – Protest : Quan hệ 1 – n
  + Một User có thể gửi nhiều Protest.
  + Một Protest chỉ do một User (protester) gửi.
* Listing – Protest : Quan hệ 1 – n
  + Một Listing có thể bị Protest nhiều lần (sau khi đã được giải quyết).
  + Một Protest chỉ liên quan đến một Listing.
* User (admin) – Protest : Quan hệ 1 – n
  + Một User (admin) có thể xử lý nhiều Protest.
  + Một Protest chỉ do một User (admin) xử lý.

### Thiết kế giao diện

## Kết quả đạt được

Dự án đã được hoàn thành và đưa vào vận hành với những chức năng trọng yếu, hình thành một nền tảng bất động sản đáng tin cậy. Bốn chức năng nổi bật nhất gồm có:

*Chức năng 1: Xác thực Danh tính Người dùng Toàn diện (eKYC)*

Đây là tính năng nền tảng giúp xây dựng sự tin cậy và minh bạch cho toàn bộ hệ thống.

*Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

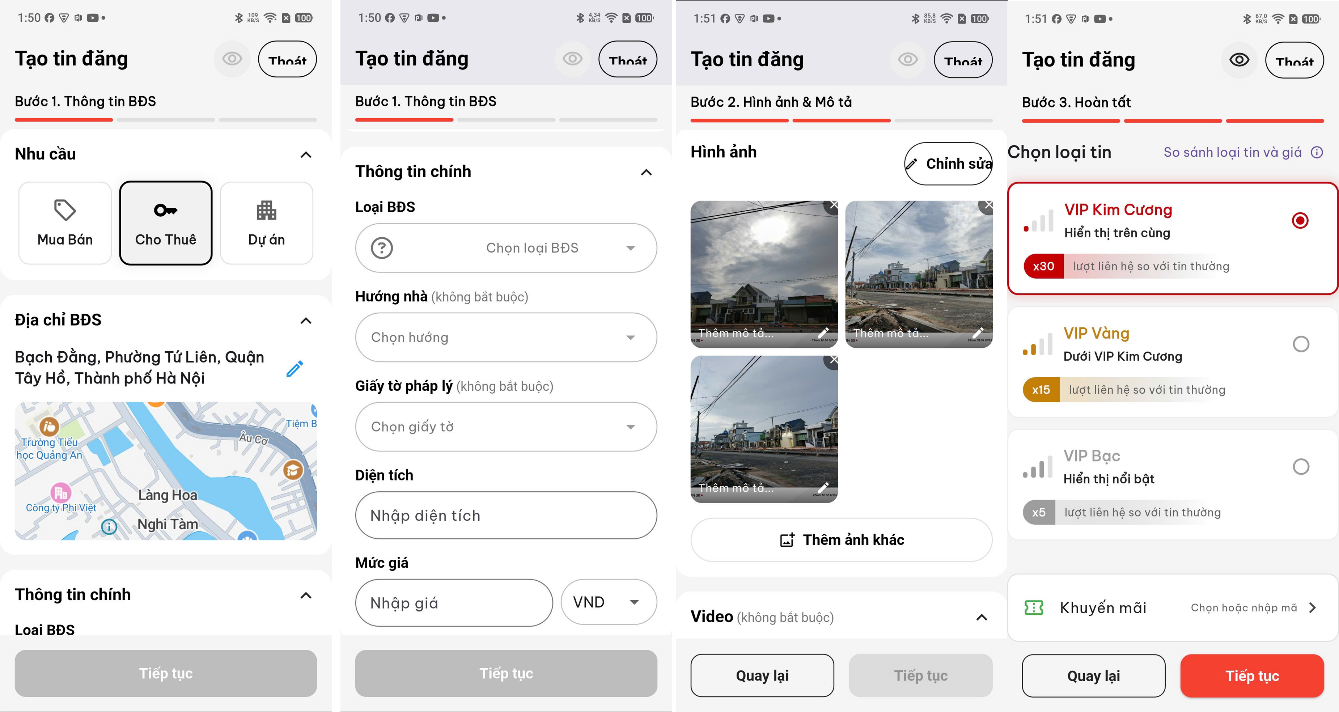
Nội dung do AI tạo ra có thể không chính xác.*

Hình . Chức năng: Xác thực danh tính

* **Quy trình sử dụng:**
  1. Người dùng truy cập vào trang cá nhân và chọn mục "Xác thực danh tính".
  2. Bước 1 - Quét CCCD: Ứng dụng mở giao diện camera. Người dùng đưa mặt trước của Căn cước công dân (CCCD) vào trong khung hình và chụp ảnh.
  3. Hệ thống tự động gửi ảnh đến server, sau đó gọi API của FPT.AI để trích xuất các thông tin như Họ tên, Ngày sinh, Quê quán... và cập nhật vào hồ sơ người dùng.
  4. Bước 2 - Xác thực khuôn mặt: Ứng dụng chuyển sang camera trước. Người dùng đưa khuôn mặt vào trong khung hình oval.
  5. Hệ thống tự động quay một video ngắn, sử dụng AI để kiểm tra đây là người thật (liveness check) và so sánh sự trùng khớp giữa khuôn mặt trong video với ảnh trên CCCD.
* Kết quả: Sau khi hoàn tất, tài khoản người dùng sẽ được gắn huy hiệu "Đã xác thực", tăng cường độ tin cậy khi thực hiện các giao dịch hoặc đăng tin trên nền tảng.

*Chức năng 2: Đăng đa dạng loại tin Bất động sản*

Chức năng cho phép người dùng, đặc biệt là các môi giới, cung cấp thông tin về bất động sản một cách chi tiết và đầy đủ nhất thông qua một quy trình 3 bước rõ ràng.

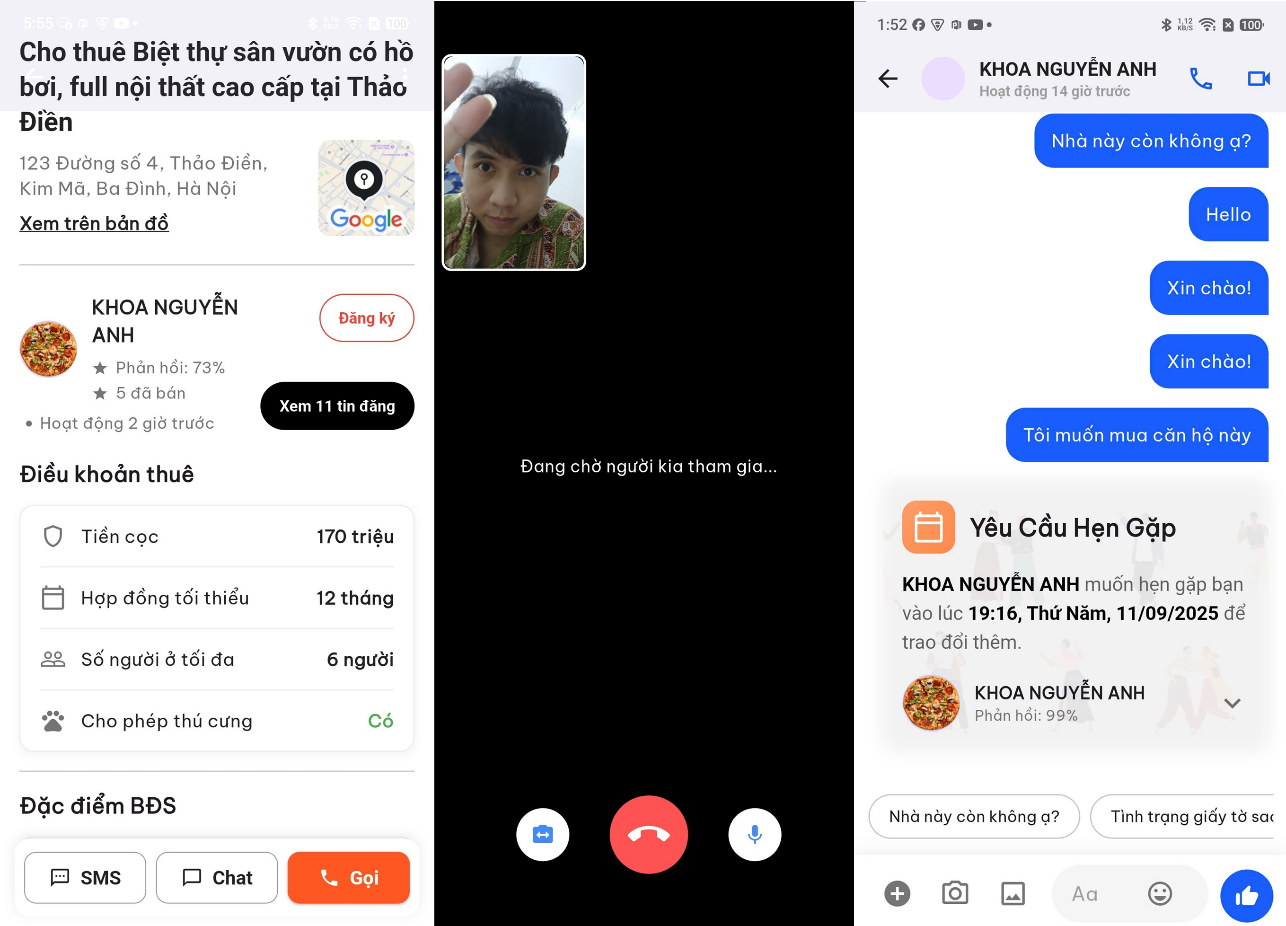


Hình . Chức năng: Tạo tin đăng

* Quy trình sử dụng:
  1. Người dùng chọn chức năng "Đăng tin" từ màn hình chính.
  2. Bước 1 - Thông tin BĐS: Người dùng nhập các thông tin cơ bản và bắt buộc như nhu cầu (Bán/Cho thuê), địa chỉ (hỗ trợ tìm kiếm và chọn trên bản đồ), loại hình BĐS, diện tích, mức giá, và các đặc điểm chi tiết (số phòng, hướng nhà, pháp lý...).
  3. Bước 2 - Hình ảnh & Mô tả: Người dùng tải lên nhiều hình ảnh (hỗ trợ chọn từ thư viện hoặc chụp mới), dán đường dẫn video Youtube/Tiktok, và soạn thảo mô tả chi tiết bằng trình soạn thảo văn bản giàu tính năng (rich text editor).
  4. Bước 3 - Hoàn tất: Người dùng lựa chọn loại tin đăng (Tin thường, VIP), áp dụng các mã khuyến mãi (nếu có), và tiến hành thanh toán để hoàn tất.
* Kết quả: Một tin đăng mới được tạo trên hệ thống với đầy đủ thông tin, hình ảnh trực quan. Ngay sau khi đăng, một tác vụ nền (Celery) sẽ tự động quét nội dung để kiểm tra spam, đảm bảo chất lượng cho toàn bộ nền tảng.

*Chức năng 3: Tương tác Trực tiếp qua Chat và Video Call*

Chức năng này phá bỏ rào cản giao tiếp, cho phép người mua và người bán kết nối tức thì, đẩy nhanh quá trình ra quyết định.



Hình . Chức năng: Chat & Video Call

* Quy trình sử dụng:
  1. Tại màn hình chi tiết một tin đăng, người dùng nhấn nút "Chat".
  2. Ứng dụng gọi API đến server Django để khởi tạo hoặc lấy thông tin phòng chat. Server trả về ID của phòng chat trên cả PostgreSQL và Firebase Firestore.
  3. Ứng dụng mở màn hình chat và bắt đầu lắng nghe (subscribe) vào document tương ứng trên Firestore để nhận/gửi tin nhắn theo thời gian thực.
  4. Trong màn hình chat, người dùng có thể nhấn vào biểu tượng Video Call.
  5. Ứng dụng tiếp tục gọi API Django để yêu cầu một token truy cập tạm thời cho dịch vụ Agora.
  6. Sau khi nhận được token, ứng dụng khởi tạo một phiên gọi video trực tiếp giữa hai người dùng thông qua Agora SDK.
* Kết quả: Người dùng có thể trao đổi thông tin, thương lượng và thậm chí là xem nhà từ xa một cách liền mạch và hiệu quả ngay trên ứng dụng.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

### Những điều đã hoàn thành

### Những điều chưa hoàn thiện

Hiệu năng tải dữ liệu ban đầu: Mặc dù đã áp dụng các kỹ thuật tối ưu, việc tải đồng thời nhiều nguồn dữ liệu trên một số màn hình phức tạp (như trang chủ, trang chi tiết) đôi khi vẫn còn độ trễ, cần được tiếp tục tối ưu hóa.

Giao diện người dùng (UI/UX) trên một số thiết bị: Giao diện đã được xây dựng để đáp ứng đa dạng kích thước màn hình, tuy nhiên vẫn cần thêm các bước kiểm thử và tinh chỉnh để đảm bảo trải nghiệm hoàn hảo trên các thiết bị đặc thù (ví dụ: máy tính bảng, điện thoại màn hình gập).

Tính năng Chat và Gọi video còn ở mức cơ bản: Chức năng chat và gọi video đã hoạt động nhưng vẫn thiếu các tính năng nâng cao như gửi hình ảnh/file trong chat, chia sẻ màn hình, hay ghi âm cuộc gọi.

Hệ thống gợi ý và cá nhân hóa: Hệ thống hiện tại chủ yếu hiển thị dữ liệu dựa trên các bộ lọc thủ công của người dùng, chưa có cơ chế tự động gợi ý các tin đăng phù hợp dựa trên hành vi và lịch sử tìm kiếm.

## Hướng phát triển

Tích hợp hệ thống thanh toán trực tuyến: Tích hợp các cổng thanh toán như VNPay, MoMo để cho phép người dùng thanh toán các dịch vụ trực tiếp trên ứng dụng, ví dụ như mua các gói tin VIP, đặt cọc giữ chỗ bất động sản.

Ứng dụng Trí tuệ nhân tạo (AI) và Học máy (Machine Learning):

Xây dựng hệ thống gợi ý (Recommendation System) để tự động đề xuất các tin đăng phù hợp với sở thích và hành vi của người dùng.

Phát triển công cụ định giá bất động sản tự động dựa trên việc phân tích dữ liệu lịch sử và các đặc điểm của bất động sản.

Nâng cấp chatbot, tích hợp các mô hình ngôn ngữ lớn (LLM) để có thể trả lời các câu hỏi phức tạp hơn về pháp lý, quy hoạch.

TÀI LIỆU THAM KHẢO

(Theo chuẩn IEEE – *bỏ dòng này khi viết báo cáo*)

tối thiểu 40 trang, tôi đa 80 trang

PHỤ LỤC