

**BỘ NÔNG NGHIỆP VÀ MÔI TRƯỜNG
TRƯỜNG ĐẠI HỌC THỦY LỢI**



BÀI TẬP LỚN

**PHÁT TRIỂN ÚNG DỤNG CHO THIẾT BỊ DI ĐỘNG
ĐỀ TÀI: ÚNG DỤNG ...**

Giáo viên hướng dẫn: ThS. Kiều Tuấn Dũng

Sinh viên thực hiện:

STT	Mã sinh viên	Họ và tên	Lớp
1	2251061698	Lê Mạnh Hùng Anh	64CNTT1
2	2251061799	Phan Quang Huy	64CNTT1

Hà Nội, năm 2025

**BỘ NÔNG NGHIỆP VÀ MÔI TRƯỜNG
TRƯỜNG ĐẠI HỌC THỦY LỢI**



BÀI TẬP LỚN

**PHÁT TRIỂN ÚNG DỤNG CHO THIẾT BỊ DI ĐỘNG
ĐỀ TÀI: ÚNG DỤNG**

STT	Mã Sinh Viên	Họ và Tên	Ngày Sinh	Điểm	
				Bảng Số	Bảng Chữ
1	2251061698	Lê Mạnh Hùng Anh	19/06/2004		
2	2251061799	Phan Quang Huy	24/08/2004		

CÁN BỘ CHẤM THI

Hà Nội, năm 2025

LỜI NÓI ĐẦU

Trong thời đại công nghệ số phát triển mạnh mẽ, mạng xã hội đã trở thành một phần không thể thiếu trong đời sống hằng ngày của con người. Trong đó, các nền tảng chia sẻ hình ảnh như Instagram đóng vai trò quan trọng trong việc kết nối, giao tiếp và thể hiện bản thân giữa các cá nhân trên toàn thế giới.

Nhận thức được tầm quan trọng đó, nhóm chúng em thực hiện bài tập lớn với đề tài **“Xây dựng ứng dụng chia sẻ hình ảnh – Instagram Clone”**. Đây là một dự án mô phỏng các tính năng cơ bản của ứng dụng Instagram, nhằm giúp sinh viên hiểu rõ hơn về quy trình phát triển một ứng dụng mạng xã hội từ frontend đến backend, đồng thời áp dụng và củng cố kiến thức đã học như lập trình mobile, thiết kế giao diện, quản lý cơ sở dữ liệu, tương tác với server và các vấn đề liên quan đến bảo mật. Thông qua dự án này, sinh viên không chỉ có cơ hội rèn luyện kỹ năng lập trình mà còn nâng cao khả năng làm việc nhóm, tư duy thiết kế hệ thống và giải quyết vấn đề thực tiễn. Mặc dù ứng dụng chưa thể hoàn thiện đầy đủ như phiên bản gốc, nhưng đó là bước đầu quan trọng để tiếp cận và làm chủ những công nghệ hiện đại trong phát triển phần mềm.

Bố cục báo cáo được trình bày thành 3 chương:

Chương 1: Tổng quan về mạng xã hội chia sẻ ảnh và nền tảng xây dựng ứng dụng.

Trong chương này, báo cáo trình bày tổng quan về xu hướng mạng xã hội hình ảnh, tầm quan trọng của việc chia sẻ thông tin qua hình ảnh trong đời sống hiện đại. Đồng thời, chương này giới thiệu hệ điều hành Android, nền tảng Firebase được sử dụng làm backend, và các chức năng chính của ứng dụng như: đăng nhập, đăng bài, theo dõi, bình luận, xem Reels, v.v.

Chương 2: Hệ thống ứng dụng và thực nghiệm.

Phần này tiến hành phân tích và thiết kế hệ thống thông qua sơ đồ Use Case, mô hình cơ sở dữ liệu và các chức năng chính của ứng dụng. Ngoài ra, chương này còn đưa ra các kết quả thực nghiệm kèm hình ảnh giao diện ứng dụng sau khi hoàn thành.

Chương 3: Kết luận và hướng phát triển.

Chương này tổng kết hiệu quả mà ứng dụng Instagram Clone đã đạt được, đồng thời đưa ra các hướng cải tiến trong tương lai, như tích hợp chức năng nhắn tin, thông báo đẩy và cải thiện giao diện người dùng.

Nhóm chúng em xin chân thành cảm ơn thầy đã hướng dẫn và tạo điều kiện thuận lợi để thực hiện bài tập lớn này. Chúng em rất mong nhận được những góp ý quý báu để hoàn thiện hơn trong các dự án tiếp theo.

MỤC LỤC

LỜI NÓI ĐẦU	3
DANH MỤC HÌNH ẢNH	5
DANH MỤC BẢNG BIỂU	6
DANH MỤC CÁC TỪ VIẾT TẮT	7
Chương 1: TỔNG QUAN VỀ ĐỀ TÀI	8
1.1. Giới thiệu về đề tài	8
1.2. Mục tiêu của đề tài	8
1.3. Phạm vi đề tài	8
1.4. Phân chia nhiệm vụ	9
Chương 2: KIẾN TRÚC VÀ CÔNG NGHỆ	10
2.1. Kiến trúc hệ thống	10
2.1.1. Biểu đồ Use Case	
2.1.2. Mô tả Use Case (UC01 – UC16)	
2.2. Giới thiệu về công nghệ sử dụng	27
Chương 3: XÂY DỰNG ỨNG DỤNG	28
3.1. Thiết kế giao diện (Figma)	28
3.2. Thiết kế cơ sở dữ liệu	28
3.3. Giao diện ứng dụng	30
- Màn hình truy cập, đăng ký, đăng nhập	
- Màn hình chính, tìm kiếm, đăng bài	
- Add Post, Add Reels, Profile, Reels	
3.4. Code minh họa chức năng chính	43
3.4.1. Đăng ký	
3.4.2. Đăng nhập	
3.4.3. Màn hình chính	
3.4.4. Hồ sơ cá nhân	
3.4.5. Đăng bài	
3.4.6. Tìm người dùng	
3.4.7. Tải lên Reels	
3.4.8. Follow người dùng	
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	56
4.1. Kết quả đạt được	56
4.2. Nhược điểm	56
4.3. Hướng phát triển	56
TÀI LIỆU THAM KHẢO	57

DANH MỤC HÌNH ẢNH

STT	Hình ảnh	Trang
1	Hình 2.1.1.1: Biểu đồ Use Case	11
2	Hình 3.2.1 Lược đồ CSDL	28
3	Hình 3.2.2 Minh họa CSDL của User	29
4	Hình 3.2.3 Minh họa CSDL của Post	29
5	Hình 3.2.4 Minh họa CSDL của Reel	30
6	Hình 3.3.2 Giao diện truy cập	33
7	Hình 3.3.3 Giao diện đăng ký	34
8	Hình 3.3.4 Giao diện đăng nhập	35
9	Hình 3.3.5 Giao diện màn hình chính	36
10	Hình 3.3.6 Giao diện tìm kiếm người dùng	37
11	Hình 3.3.7 Giao diện đăng bài	38
12	Hình 3.3.8 Giao diện chức năng Add Post	39
13	Hình 3.3.9 Giao diện chức năng Add Reels	40
14	Hình 3.3.10 Giao diện Profile	41
15	Hình 3.3.11 Giao diện hiển thị Reels	42

DANH MỤC BẢNG BIỂU

STT	Bảng biểu	Trang
1	Bảng phân chia nhiệm vụ	9
2	Bảng 2.1.2.1: Use Case Đăng ký	11
3	Bảng 2.1.2.2: Use Case Đăng nhập	12
4	Bảng 2.1.2.3: Use Case Quên mật khẩu	13
5	Bảng 2.1.2.4: Use Case Đăng bài	14
6	Bảng 2.1.2.5: Use Case Chính sửa bài đăng	15
7	Bảng 2.1.2.6: Use Case Xóa bài đăng	16
8	Bảng 2.1.2.7: Use Case Bình luận bài đăng	17
9	Bảng 2.1.2.8: Use Case Thích bài đăng	18
10	Bảng 2.1.2.9: Use Case Theo dõi người dùng	19
11	Bảng 2.1.2.10: Use Case Nhắn tin trực tiếp	20
12	Bảng 2.1.2.11: Use Case Nhận thông báo	21
13	Bảng 2.1.2.12: Use Case Tìm kiếm người dùng	22
14	Bảng 2.1.2.13: Use Case Xem nội dung đề xuất	23
15	Bảng 2.1.2.14: Use Case Cập nhật Avatar	24
16	Bảng 2.1.2.15: Use Case Chính sửa thông tin cá nhân	25
17	Bảng 2.1.2.16: Use Case Xem danh sách người theo dõi	26

DANH MỤC CÁC TỪ VIẾT TẮT

STT	TỪ VIẾT TẮT	VIẾT ĐẦY ĐỦ
1	UI	User Interface (Giao diện)
2	UX	User Experience (Trải nghiệm)
3	CRUD	Create Read Update Delete
4	SDK	Software Development Kit
5	FCM	Firebase Cloud Messaging
6	CSDL	Cơ Sở Dữ Liệu

Chương 1. TỔNG QUAN VỀ ĐỀ TÀI

1.1. Giới thiệu về đề tài

Instagram là một trong những mạng xã hội chia sẻ hình ảnh phổ biến nhất hiện nay, với hàng trăm triệu người dùng trên toàn thế giới. Ứng dụng này cho phép người dùng đăng tải hình ảnh, video, tương tác với nhau thông qua lượt thích, bình luận và theo dõi. Với giao diện trực quan và trải nghiệm người dùng mượt mà, Instagram đã trở thành hình mẫu lý tưởng cho các ứng dụng mạng xã hội hiện đại.

Đề tài "**Instagram Clone**" là một phiên bản mô phỏng các tính năng cơ bản của Instagram, được triển khai dưới dạng một ứng dụng mobile. Trong khuôn khổ bài tập lớn này, nhóm chúng em tập trung xây dựng một hệ thống có thể:

- Đăng ký và đăng nhập người dùng
- Tải lên và hiển thị hình ảnh, video
- Tương tác với bài viết (like, comment, chỉnh sửa, share)
- Follow và tìm kiếm người dùng
- Chỉnh sửa hồ sơ người dùng
- Lưu trữ dữ liệu và hình ảnh thông qua backend và dịch vụ đám mây

Dự án được phát triển bằng ngôn ngữ **Kotlin** trên nền tảng **Android**, sử dụng **Firebase** để xử lý các chức năng như xác thực người dùng, lưu trữ hình ảnh, và quản lý cơ sở dữ liệu thời gian thực. Bên cạnh đó, nhóm cũng chú trọng đến giao diện người dùng (UI/UX) và tính bảo mật trong quá trình phát triển ứng dụng.

Thông qua đề tài này, sinh viên có thể hình dung rõ hơn về kiến trúc tổng thể của một ứng dụng mạng xã hội hiện đại, đồng thời áp dụng được nhiều kiến thức liên môn một cách thực tế và hiệu quả.

1.2. Mục tiêu của đề tài

- Hiểu và áp dụng các công nghệ phát triển ứng dụng di động.
- Nắm vững cách thiết kế và triển khai backend phục vụ ứng dụng di động.
- Xây dựng các chức năng chính của một mạng xã hội.
- Tăng cường kỹ năng làm việc nhóm, quản lý dự án và lập trình.

1.3. Phạm vi của đề tài

- Ứng dụng này chỉ được phát triển để phục vụ nhóm thực hiện dự án và không nhằm mục đích công khai ra toàn cầu.
- Người dùng chính là các thành viên trong nhóm, sử dụng ứng dụng để thực hành và kiểm thử các tính năng của một nền tảng mạng xã hội.
- Hệ thống không có cơ chế quản trị viên mà chỉ có người dùng bình thường.

1.4 Phân chia nhiệm vụ

<<Bảng phân chia nhiệm vụ>>

Thành viên	Nhiệm vụ
Lê Mạnh Hùng Anh	<ul style="list-style-type: none"> - Phân tích thiết kế giao diện Figma - Phân tích thiết kế CSDL - Phân tích thiết kế use case - Thực hiện chức năng đăng bài, đăng reels, hiển thị bài đăng, hiển thị reels, chỉnh sửa bài đăng, tương tác bài đăng
Phan Quang Huy	<ul style="list-style-type: none"> - Phân tích thiết kế giao diện Figma - Phân tích thiết kế use case - Thực hiện chức năng đăng nhập, đăng ký, quên mật khẩu, tìm kiếm người dùng, follow người dùng, chỉnh sửa hồ sơ

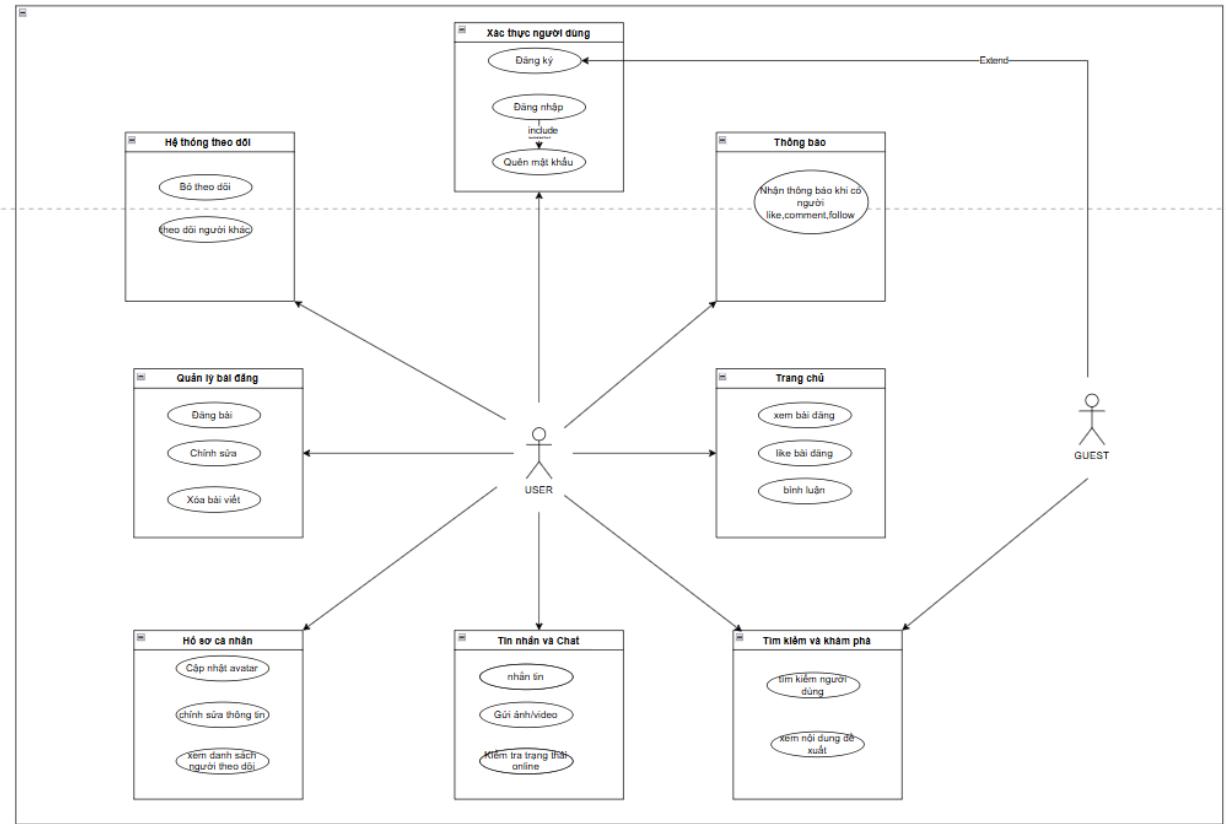
Chương 2. KIẾN TRÚC VÀ CÔNG NGHỆ

2.1. Kiến trúc hệ thống

Ứng dụng Instagram Clone được xây dựng theo kiến trúc phân lớp gồm 3 tầng chính:

- **Lớp giao diện người dùng (UI):**
Gồm các **Activity** và **Fragment** như **LoginActivity**, **SignUpActivity**, **HomeFragment**, **ProfileFragment**,... giúp hiển thị thông tin và tương tác với người dùng.
- **Lớp xử lý nghiệp vụ (Logic):**
Gồm các lớp **Adapter** (ví dụ: **PostAdapter**, **ReelAdapter**) dùng để kết nối dữ liệu với giao diện, và xử lý các thao tác người dùng như like, comment, theo dõi,...
- **Lớp dữ liệu (Data):**
Sử dụng Firebase để lưu trữ dữ liệu người dùng, bài viết, bình luận,... Bao gồm:
 - **Firebase Authentication:** xác thực người dùng
 - **Firebase Firestore:** lưu dữ liệu dạng collection-document
 - **Firebase Storage:** lưu ảnh và video

2.1.1 Use Case Diagram



Hình 2.1.1.1: Biểu đồ Use Case

2.1.2 Use Case Description – Đăng ký

Use Case Name: Đăng ký	ID: UC01	Importance level: Medium		
Primary Actor: App User	Use Case Type: Detail, essential			
Stakeholders & Interests:				
User – Muốn tạo tài khoản để sử dụng các tính năng của hệ thống.				
Brief Description: Use case này mô tả cách người dùng tạo tài khoản mới trên hệ thống Instagram Clone bằng cách cung cấp thông tin cá nhân như tên, email, mật khẩu.				

Trigger: User chọn chức năng “Đăng ký” trên trang chính của ứng dụng.

Relationship:

- **Association:** User
- **Include:** (none)
- **Extend:** (none)
- **Generalization:** (none)

Action Step

1. Người dùng nhập các thông tin đăng ký gồm: tên, email, mật khẩu.

2. Hệ thống kiểm tra thông tin đăng ký :

- Nếu thông tin hợp lệ, hệ thống tạo tài khoản mới.
- Nếu thông tin không hợp lệ, hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại.

3. Nếu đăng ký thành công hệ thống sẽ :

- Lưu tài khoản vào cơ sở dữ liệu.
- Gửi email xác nhận cho người dùng.
- Chuyển hướng người dùng đến trang chủ Instagram Clone.

Exception:

▪

Extension:

Pre-Condition:

Post-Condition:

Bảng 2.1.2.1: Use Case Đăng ký

2.1.3 Use Case Description – Đăng nhập

Use Case Name: Đăng nhập	ID: UC02	Importance level: Medium		
Primary Actor: App User	Use Case Type: Detail, essential			
Stackholders & Interests:				
User – Muốn truy cập vào tài khoản của mình để sử dụng có tính năng của hệ thống.				
Brief Description: Use case này mô tả cách người dùng đăng nhập vào hệ thống Instagram Clone bằng tên đăng nhập và mật khẩu				
Trigger: User chọn chức năng “Đăng nhập” trên trang chính của ứng dụng.				
Relationship:				
<ul style="list-style-type: none"> ▪ Association: User 				

- **Include:** (none)
- **Extend:** (none)
- **Generalization:** (none)

Action Step:

1. Người dùng đăng nhập tên đăng nhập và mật khẩu.

2. Hệ thống kiểm tra thông tin đăng nhập :

- Nếu thông tin hợp lệ, hệ thống cho phép người dùng truy cập vào trang chủ.
- Nếu thông tin không hợp lệ, hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại.

3. Nếu đăng nhập thành công hệ thống sẽ :

- Lưu phiên đăng nhập của người dùng.
- Chuyển hướng người dùng đến trang chủ Instagram Clone.
- Hiển thị thông báo chào mừng.

Exception:

▪

Extension:

Pre-Condition:

Post-Condition:

Bảng 2.1.2.2: Use Case Đăng nhập

2.1.4 Use Case Description – Quên mật khẩu

Use Case Name: Quên mật khẩu	ID: UC03	Importance level: Medium
Primary Actor: App User	Use Case Type: Detail, essential	
Stackholders & Interests:		
User – Muốn khôi phục mật khẩu khi quên.		
Brief Description: Use case này mô tả cách người dùng khôi phục mật khẩu bằng cách nhận email đặt lại mật khẩu.		
Trigger: User chọn chức năng “Quên mật khẩu” trên trang chính của ứng dụng.		
Relationship:		
<ul style="list-style-type: none"> ▪ Association: User ▪ Include: (none) ▪ Extend: (none) ▪ Generalization: (none) 		

Action Step:

1.Người dùng nhập địa chỉ email đã đăng ký.

2.Hệ thống kiểm tra email :

- Nếu email hợp lệ, hệ thống gửi email đặt lại mật khẩu.
- Nếu thông tin không hợp lệ, hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại.

3.Người dùng truy cập vào email và nhấp vào liên kết đặt lại mật khẩu.

4.Người dùng nhập mật khẩu mới và xác nhận.

5.Hệ thống cập nhật mật khẩu mới và hiển thị thông báo thành công.

Exception:

-

Extension:**Pre-Condition:****Post-Condition:**

Bảng 2.1.2.3: Use Case Quên mật khẩu

2.1.5 Use Case Description – Đăng bài

Use Case Name: Đăng bài	ID: UC04	Importance level: Medium		
Primary Actor: App User	Use Case Type: Detail, essential			
Stackholders & Interests:				
User – Muốn đăng bài, hình ảnh lên Instagram Clone.				
Brief Description: Use case này mô tả cách người dùng tạo bài đăng mới trên Instagram Clone.				
Trigger: User chọn chức năng “Đăng bài” trên giao diện của ứng dụng.				
Relationship:				
<ul style="list-style-type: none"> ▪ Association: User ▪ Include: (none) ▪ Extend: (none) ▪ Generalization: (none) 				
Action Step:				

- 1.Người dùng chọn ảnh/video hoặc nội dung bài viết.**
- 2.Người dùng có thể thêm mô tả, gắn thẻ bạn bè.**
- 3.Người dùng nhấn nút “Đăng bài”.**
- 4.Hệ thống kiểm tra nội dung bài viết :**
 - Nếu hợp lệ, hệ thống lưu bài viết vào cơ sở dữ liệu và hiển thị trên trang chủ.
 - Nếu không hợp lệ, hệ thống hiển thị thông báo lỗi.
- 5.Nếu bài đăng thành công, hệ thống hiển thị thông báo xác nhận.**

Exception:

-

Extension:

Pre-Condition:

Post-Condition:

Bảng 2.1.2.4: Use Case Đăng bài

2.1.6 Use Case Description – Chính sửa bài đăng

Use Case Name: Chính sửa bài đăng	ID: UC09	Importance level: Medium		
Primary Actor: App User	Use Case Type: Detail, essential			
Stackholders & Interests:				
User – Muốn thay đổi nội dung bài đăng sau khi đăng tải.				
Brief Description: Use case này mô tả cách người dùng chỉnh sửa bài đăng của họ.				
Trigger: Người dùng chọn chỉnh sửa một bài đã đăng trước đó.				
Relationship: <ul style="list-style-type: none"> ▪ Association: User ▪ Include: (none) ▪ Extend: (none) ▪ Generalization: (none) 				
Action Step:				
<ol style="list-style-type: none"> 1.Người dùng vào hồ sơ cá nhân hoặc trang chủ. 2.Người dùng chọn một bài đang đã đăng trước đó. 				

- 3.Người dùng nhấn vào tùy chọn “Chỉnh sửa bài đăng”.
- 4.Hệ thống hiển thị giao diện chỉnh sửa.
- 5.Người dùng thay đổi nội dung, thêm hoặc xóa hình ảnh/video.
- 6.Người dùng nhấn “Lưu thay đổi”.
- 7.Hệ thống cập nhật bài đăng và hiển thị thông báo thành công.
- 2.Hệ thống tạo thông báo cho người dùng liên quan.
- 2.Hệ thống hiển thị thông báo trong mục “Thông báo”.
- 3.Người dùng có thể nhấn vào thông báo để xem chi tiết.

Exception:

- Nếu bài đăng không thuộc về người dùng, hệ thống không cho phép chỉnh sửa.
- Nếu có lỗi mạng trong quá trình lưu, hệ thống hiển thị thông báo lỗi.

Extension:

Pre-Condition:

Post-Condition:

Bảng 2.1.2.5: Use Case Chính sửa bài đăng

2.1.7 Use Case Description – Xóa bài đăng

Use Case Name: Xóa bài đăng	ID: UC09	Importance level: Medium		
Primary Actor: App User	Use Case Type: Detail, essential			
Stakeholders & Interests:				
User – Muốn xóa bài đăng không còn cần thiết.				
Brief Description: Use case này mô tả cách người dùng xóa một bài đăng khỏi trang cá nhân.				
Trigger: Người dùng chọn “Xóa bài viết” trong menu quản lý bài đăng.				
Relationship:				
<ul style="list-style-type: none"> ▪ Association: User ▪ Include: (none) ▪ Extend: (none) ▪ Generalization: (none) 				

Action Step:

- 1.Người dùng truy cập bài đăng của mình.
 - 2.Người dùng chọn “Xóa bài viết”.
 - 3.Hệ thống hiển thị cảnh báo “Xác nhận”.
 - 4.Nếu người dùng xác nhận, hệ thống xóa bài đăng khỏi cơ sở dữ liệu.
 - 5.Hệ thống hiển thị thông báo xác nhận xóa thành công.
- 2.Hệ thống tạo thông báo cho người dùng liên quan.
 - 2.Hệ thống hiển thị thông báo trong mục “Thông báo”.
 - 3.Người dùng có thể nhấn vào thông báo để xem chi tiết.

Exception:

▪

Extension:**Pre-Condition:****Post-Condition:****Bảng 2.1.2.6: Use Case Xóa bài đăng**

2.1.8 Use Case Description – Bình luận bài đăng

Use Case Name: Bình luận bài đăng	ID: UC05	Importance level: Medium
Primary Actor: App User	Use Case Type: Detail, essential	
Stackholders & Interests:		
User – Muốn bình luận bài đăng của mình hoặc người khác.		
Brief Description: Use case này mô tả cách người dùng bình luận một bài đăng.		
Trigger: User nhấn vào một bài đăng và chọn chức năng “Bình luận”.		
Relationship:		
<ul style="list-style-type: none">▪ Association: User▪ Include: (none)▪ Extend: (none)▪ Generalization: (none)		

Action Step:

1.Người dùng nhập nội dung bình luận.

2.Người dùng nhấn nút “Gửi bình luận”

3.Hệ thống kiểm tra nội dung bình luận:

- Nếu hợp lệ, bình luận được đăng lên bài viết.
- Nếu không hợp lệ, hệ thống hiển thị thông báo lỗi.

5.Nếu bình luận thành công, hệ thống hiển thị bình luận trên bài viết.

Exception:

▪

Extension:**Pre-Condition:****Post-Condition:**

Bảng 2.1.2.7: Use Case Bình luận bài đăng

2.1.9 Use Case Description – Thích bài đăng

Use Case Name: Like bài đăng	ID: UC06	Importance level: Medium		
Primary Actor: App User	Use Case Type: Detail, essential			
Stackholders & Interests:				
User – Muốn thể hiện sự yêu thích với đối với bài đăng.				
Brief Description: Use case này mô tả cách người “like” một bài đăng.				
Trigger: User nhấn vào biểu hiện “Thích” dưới bài đăng.				
Relationship:				
<ul style="list-style-type: none"> ▪ Association: User ▪ Include: (none) ▪ Extend: (none) ▪ Generalization: (none) 				
Action Step:				
1.Người dùng nhấn vào biểu tượng “Thích”.				
2.Hệ thống kiểm tra trạng thái like của người dùng trên bài đăng đó:				

- Nếu chưa like, hệ thống thêm lượt like vào bài đăng.
- Nếu đã like, hệ thống hủy like.

3.Hệ thống cập nhật số lượt like và hiển thị thông tin mới.

4.Nếu người dùng like bài đăng của người khác, hệ thống gửi thông báo đến chủ bài đăng.

Exception:

-

Extension:

Pre-Condition:

Post-Condition:

Bảng 2.1.2.8: Use Case Thích bài đăng

2.1.10 Use Case Description – Theo dõi người dùng

Use Case Name: Theo dõi người dùng	ID: UC07	Importance level: Medium
Primary Actor: App User	Use Case Type: Detail, essential	
Stackholders & Interests:		
User – Muốn theo dõi bài đăng của người khác.		
Brief Description: Use case này mô tả cách người dùng theo dõi người dùng khác trên Instagram Clone.		
Trigger: User nhấp vào nút “Theo dõi” trên hồ sơ của người khác.		
Relationship: <ul style="list-style-type: none"> ▪ Association: User ▪ Include: (none) ▪ Extend: (none) ▪ Generalization: (none) 		
Action Step: <ol style="list-style-type: none"> 1.Người dùng vào trang cá nhân của người khác. 2.Người dùng nhấp vào nút “Theo dõi”. 		

2.Hệ thống kiểm tra trạng thái theo dõi:

- Nếu chưa theo dõi, hệ thống thêm người dùng vào danh sách theo dõi.
- Nếu đã theo dõi, hệ thống hủy theo dõi.

3.Hệ thống cập nhật số lượt like và hiển thị thông tin mới.

4.Nếu người dùng bắt đầu theo dõi, hệ thống gửi thông báo đến người được theo dõi.

Exception:

▪

Extension:

Pre-Condition:

Post-Condition:

Bảng 2.1.2.9: Use Case Theo dõi người dùng

2.1..11 Use Case Description – Nhắn tin trực tiếp

Use Case Name: Nhắn tin trực tiếp	ID: UC08	Importance level: Medium		
Primary Actor: App User	Use Case Type: Detail, essential			
Stackholders & Interests:				
User – Muốn trò chuyện riêng với người khác.				
Brief Description: Use case này mô tả cách người dùng gửi tin nhắn trực tiếp trên Instagram Clone.				
Trigger: User nhấn vào hộp thư và chọn người để nhắn tin.				
Relationship: <ul style="list-style-type: none">▪ Association: User▪ Include: (none)▪ Extend: (none)▪ Generalization: (none)				
Action Step:				
1.Người dùng vào phần tin nhắn và chọn một cuộc trò chuyện.				
2.Người dùng nhập nội dung tin nhắn và nhấn “Gửi”.				
2.Hệ thống kiểm tra tin nhắn: <ul style="list-style-type: none">• Nếu hợp lệ, tin nhắn được gửi thành công.				

- Nếu không hợp lệ, hệ thống hiển thị thông báo lỗi.
- 3.Hệ thống hiển thị thông báo trong cuộc trò chuyện.**

4.Hệ thống gửi thông báo đến người nhận tin nhắn.

Exception:

-

Extension:

-

Pre-Condition:

Post-Condition:

Bảng 2.1.2.10: Use Case Nhắn tin trực tiếp

2.1.12 Use Case Description – Nhận thông báo

Use Case Name: Nhận thông báo	ID: UC09	Importance level: Medium		
Primary Actor: App User	Use Case Type: Detail, essential			
Stackholders & Interests:				
User – Muốn nhận thông báo khi có tương tác với bài đăng hoặc tài khoản của mình.				
Brief Description: Use case này mô tả cách người dùng nhận thông báo từ hệ thống.				
Trigger: Hệ thống phát hiện có sự kiện liên quan đến người dùng(like,comment, follow, tin nhắn).				
Relationship:				
<ul style="list-style-type: none"> ▪ Association: User ▪ Include: (none) ▪ Extend: (none) ▪ Generalization: (none) 				
Action Step:				
<ol style="list-style-type: none"> 1.Một sự kiện xảy ra (ví dụ: ai đó like bài viết, comment, gửi tin nhắn). 2.Hệ thống tạo thông báo cho người dùng liên quan. 2.Hệ thống hiển thị thông báo trong mục “Thông báo”. 3.Người dùng có thể nhấn vào thông báo để xem chi tiết. 				
Exception:				

▪
Extension:
Pre-Condition:
Post-Condition:

Bảng 2.1.2.11: Use Case Nhận thông báo

2.1.13 Use Case Description – Tìm kiếm người dùng

Use Case Name: Tìm kiếm người dùng	ID: UC010	Importance level: High		
Primary Actor: App User, Guest	Use Case Type: Detail, essential			
Stackholders & Interests:				
User, Guest – Muốn tìm kiếm tài khoản khác để theo dõi hoặc tương tác				
Brief Description: Use case này mô tả cách người dùng hoặc khách tìm kiếm tài khoản Instagram khác				
Trigger: Người dùng nhập tên hoặc từ khóa vào thanh tìm kiếm.				
Relationship:				
<ul style="list-style-type: none"> ▪ Association: User, Guest ▪ Include: (none) ▪ Extend: (none) ▪ Generalization: (none) 				
Action Step:				
<ol style="list-style-type: none"> 1. Người dùng truy cập trang “Tìm kiếm”. 2. Người dùng nhập từ khóa tìm kiếm. 3. Hệ thống hiển thị danh sách kết quả phù hợp. 4. Người dùng chọn một tài khoản từ danh sách. 5. Hệ thống chuyển hướng đến trang hồ sơ của tài khoản đó. 				
Exception:				
<ul style="list-style-type: none"> ▪ 				

Extension:
Pre-Condition:
Post-Condition:

Bảng 2.1.2.12: Use Case Tìm kiếm người dùng

2.1.14 Use Case Description – Xem nội dung đề xuất

Use Case Name: Xem nội dung đề xuất	ID: UC11	Importance level: Medium
Primary Actor: App User,Guest	Use Case Type: Detail, essential	
Stackholders & Interests:		
User,Guest – Muốn khám phá nội dung mới từ hệ thống.		
Brief Description: Use case này mô tả cách người dùng xem danh sách các bài đăng được đề xuất dựa trên sở thích và tương tác.		
Trigger: Người dùng truy cập tab “Khám phá”.		
Relationship:		
<ul style="list-style-type: none">▪ Association: User,Guest▪ Include: (none)▪ Extend: (none)▪ Generalization: (none)		
Action Step:		
<ol style="list-style-type: none">1.Người dùng truy cập trang “Khám phá”.2.Hệ thống lấy danh sách bài đăng dựa trên sở thích và xu hướng.3.Hệ thống hiển thị danh sách bài đăng được đề xuất.4.Người dùng có thể tương tác với bài đăng.		
Exception:		
<ul style="list-style-type: none">▪		
Extension:		

Pre-Condition:

Post-Condition:

Bảng 2.1.2.13: Use Case Xem nội dung đề xuất

2.1.15 Use Case Description – Cập nhật Avatar

Use Case Name: Cập nhật Avatar	ID: UC12	Importance level: Medium		
Primary Actor: App User	Use Case Type: Detail, essential			
Stackholders & Interests:				
User – Muốn cập nhật ảnh đại diện cá nhân.				
Brief Description: Use case này mô tả cách người dùng thay đổi ảnh đại diện trên Instagram Clone.				
Trigger: Người dùng truy cập hồ sơ cá nhân và chọn “Cập nhật ảnh đại diện”.				
Relationship: <ul style="list-style-type: none">▪ Association: User▪ Include: (none)▪ Extend: (none)▪ Generalization: (none)				
Action Step: <ol style="list-style-type: none">1.Người dùng vào trang hồ sơ cá nhân.2.Người dùng chọn “Cập nhật ảnh đại diện”.3.Người dùng tải lên ảnh mới hoặc chọn từ thư viện có sẵn.4.Hệ thống kiểm tra và cập nhật ảnh đại diện.5.Hệ thống hiển thị ảnh đại diện mới trên hồ sơ.				
Exception: <ul style="list-style-type: none">▪				

Extension:

Pre-Condition:

Post-Condition:

Bảng 2.1.2.14: Use Case Cập nhật Avatar

2.1.16 Use Case Description – Chính sửa thông tin cá nhân

Use Case Name: Chính sửa thông tin cá nhân	ID: UC13	Importance level: Medium		
Primary Actor: App User	Use Case Type: Detail, essential			
Stackholders & Interests:				
User – Muốn chỉnh sửa thông tin cá nhân như tên, bio, liên kết.				
Brief Description: Use case này mô tả cách người dùng chỉnh sửa thông tin cá nhân trên trang hồ sơ.				
Trigger: Người dùng truy cập trang hồ sơ và chọn “Chỉnh sửa thông tin”.				
Relationship: <ul style="list-style-type: none"> ▪ Association: User ▪ Include: (none) ▪ Extend: (none) ▪ Generalization: (none) 				
Action Step: <ol style="list-style-type: none"> 1.Người dùng vào trang hồ sơ. 2.Người dùng chọn “Chỉnh sửa thông tin”. 3. Người dùng nhập thông tin mới(tên, bio, liên kết). 4.Người dùng nhấn “Lưu thay đổi”. 5.Hệ thống cập nhật thông tin và hiển thị trên trang cá nhân. 6.Hệ thống tạo thông báo cho người dùng liên quan. 7.Hệ thống hiển thị thông báo trong mục “Thông báo”. 8.Người dùng có thể nhấn vào thông báo để xem chi tiết. 				

Exception:

▪

Extension:

Pre-Condition:

Post-Condition:

Bảng 2.1.2.15: Use Case Chính sửa thông tin cá nhân

2.1.17 Use Case Description – Xem danh sách người theo dõi

Use Case Name: Xem danh sách người theo dõi	ID: UC09	Importance level: Medium		
Primary Actor: App User	Use Case Type: Detail, essential			
Stackholders & Interests:				
User – Muốn xem danh sách những người đang theo dõi mình hoặc danh sách những người mình đang theo dõi.				
Brief Description: Use case này mô tả cách người dùng truy cập danh sách người theo dõi trên hồ sơ cá nhân				
Trigger: Người dùng truy cập hồ sơ cá nhân và chọn danh sách người theo dõi.				
Relationship:				
<ul style="list-style-type: none">▪ Association: User▪ Include: (none)▪ Extend: (none)▪ Generalization: (none)				
Action Step:				
1.Người dùng vào trang hồ sơ cá nhân. 2.Người dùng nhấn vào mục “Người theo dõi” hoặc “Người đang theo dõi”. 3.Hệ thống hiển thị danh sách người theo dõi hoặc danh sách người dùng mà họ đang theo dõi.				

4.Người dùng có thể nhấn vào từng tài khoản để xem hồ sơ chi tiết.

Exception:

▪

Extension:

Pre-Condition:

Post-Condition:

Bảng 2.1.2.16: Use Case Xem danh sách người theo dõi

2.2. Giới thiệu về Công nghệ phát triển

- **Kotlin:** Ngôn ngữ lập trình chính cho ứng dụng Android.
- **Firebase Authentication:** Quản lý người dùng đăng ký, đăng nhập.
- **Firebase Firestore:** Lưu trữ bài viết, bình luận, thông tin cá nhân,...
- **Firebase Storage:** Lưu trữ hình ảnh, video khi người dùng đăng bài.
- **Glide:** Thư viện hỗ trợ tải và hiển thị ảnh nhanh chóng từ URL.
- **RecyclerView + ViewPager2:** Quản lý danh sách bài viết và điều hướng giữa các tab.
- **Material Design:** Thiết kế giao diện đẹp mắt, hiện đại.

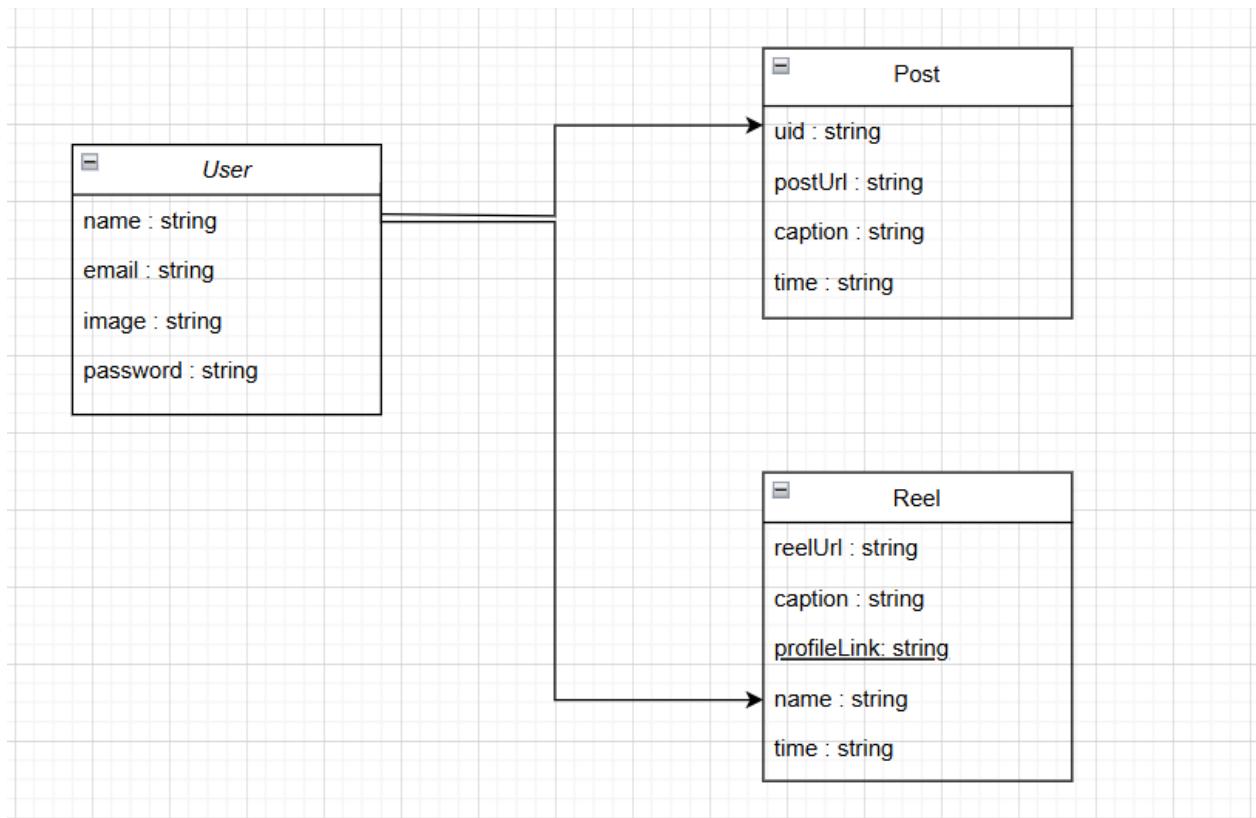
Chương 3. XÂY DỰNG ỨNG DỤNG

3.1. Thiết kế Figma

<https://www.figma.com/design/SMls3Pu9cBe5zfGrpu6vHQ/Instagram-UI-Screens?node-id=0-2&p=f&t=F21YZ17yJ1yEvzgo-0>

3.2. Thiết kế CSDL

<<Lược đồ>>



Hình 3.2.1 Lược đồ CSDL

(default)	User		nwpqXri5bGWPfs13uFYgsBrApW33	
+ Start collection	+ Add document		+ Start collection	
9So9MKKpdMooKSA0vYezJ7kyxJ2	9So9MKKpdMooKSA0vYezJ7kyxJ2		+ Add field	
9So9MKKpdMooKSA0vYezJ7kyxJ2...	j3ZWw5Mu6kXgcwTwS6UpSWvepm53		email: "hugah@gmail.com"	
9So9MKKpdMooKSA0vYezJ7kyxJ2...	nwpqXri5bGWPfs13uFYgsBrApW33 >		image: "https://firebasestorage.googleapis.com/v0/b/instagram-clone-2408.firebaseio.storage.app/o/Profile%2F8b7b97de-82a6-4958-ab4d-5f428e45aec6?alt=media&token=ed72bc02-f714-401a-aae9-ffa209650e99"	
Post	wFeGi5E6s7gTJDekPtLep9ribYD3		name: "hunganh"	
Reel			password: "1234567"	
User >				
j3ZWw5Mu6kXgcwTwS6UpSWvepm53				
j3ZWw5Mu6kXgcwTwS6UpSWvepm53...				
nwpqXri5bGWPfs13uFYgsBrApW33...				

Hình 3.2.2 Minh họa CSDL của User

(default)	Post		FH17250zvzrRD0ZLUDeE	
+ Start collection	+ Add document		+ Start collection	
9So9MKKpdMooKSA0vYezJ7kyxJ2	573SRbh3ws2b7GYsZe04		+ Add field	
9So9MKKpdMooKSA0vYezJ7kyxJ2...	FH17250zvzrRD0ZLUDeE >		caption: "1 buc tranh"	
9So9MKKpdMooKSA0vYezJ7kyxJ2...	HJGF2sR9mQanDNkk0mu7		postUrl: "https://firebasestorage.googleapis.com/v0/b/instagram-clone-2408.firebaseio.storage.app/o/PostImages%2F9d0a7c99-fefc-45fc-9ea0-3ebcbef9c437d?alt=media&token=83e12582-f363-48a6-8ced-34b857b31464"	
Post >	YwKqs4vk5x3s0VxAuZzu		time: "1744513703024"	
Reel	owk5ZDEQ1rSXU2rJi8rf		uid: "9So9MKKpdMooKSA0vYezJ7kyxJ2"	
User	rJBR8h1Fc36SWq7wJG6L			
j3ZWw5Mu6kXgcwTwS6UpSWvepm53	satwPTINo4eGGGwNe0wP			
j3ZWw5Mu6kXgcwTwS6UpSWvepm53...	sjntT2IwyfhAj0tYoWMf			
nwpqXri5bGWPfs13uFYgsBrApW33...				

Hình 3.2.3 Minh họa CSDL của Post

The screenshot shows the Google Cloud Firestore interface. On the left, there's a sidebar with collections: (default), Reel, Post, and Reel (selected). Under Reel, there are several document IDs: 818rriVMNTB1c1lCsxem, UBihfvoCKSy8hqDf2rbF (selected), b1ZY1Kjk73j2o5coAT18, ertB00MdreiD80Jt0cfK, 1Q4HRkRvAgBeVi2JRET7, and vEJ9XFrQTJ2Bq8f05AFM. The selected document 'UBihfvoCKSy8hqDf2rbF' has its details expanded, showing fields: caption ("chao ngay moi"), name ("hunganh3"), profileLink ("https://firebasestorage.googleapis.com/v0/b/instagram-clone-2408.firebaseioStorage.app/o/Profile%2F805ed4c6-97ce-447c-9c09-b35cadbe10a2?alt=media&token=7edc58e5-ea2c-4cff-b210-65245f21d6ad"), reelUrl ("https://firebasestorage.googleapis.com/v0/b/instagram-clone-2408.firebaseioStorage.app/o/ReelVideos%2F4b216d92-862f-45c1-ad33-858588799cb4?alt=media&token=a6b3829a-430c-4128-aeba-8425f86099b59"), and time ("1744599485367").

Hình 3.2.4 Minh họa CSDL của Reel

3.3. Giao diện ứng dụng.

- **Sử dụng thành phần Android trong ứng dụng**

Trong quá trình phát triển ứng dụng Instagram Clone, nhóm đã sử dụng các thành phần chính trong Android như sau:

Sử dụng 8 Activity tương ứng với các chức năng chính của ứng dụng:

1. LoginActivity – Đăng nhập tài khoản
2. SignUpActivity – Đăng ký tài khoản
3. MainActivity – Điều hướng chính giữa các Fragment
4. HomeActivity – Màn hình chính sau đăng nhập
5. EditProfileActivity – Chỉnh sửa thông tin cá nhân
6. CommentActivity – Xem và gửi bình luận cho bài viết
7. ReelUploadActivity – Tải lên video ngắn (Reel)
8. SplashActivity – Màn hình giới thiệu ban đầu

Thành phần UI được sử dụng:

- **Button:**
Được dùng trong các Activity để bắt sự kiện như "Đăng nhập", "Đăng ký",...
- **TextView:**
Hiển thị nội dung như tên người dùng, mô tả bài viết, thông báo trạng thái,...
- **Menu & Bottom Navigation:**
Ứng dụng sử dụng BottomNavigationView kết hợp với ViewPager2 để điều hướng giữa các fragment: Home, Search, Add, Reels, Profile.
- **Explicit Intent (Intent tường minh):**
Dùng để chuyển giữa các Activity
- **ImageView:**
Hiển thị ảnh đại diện, ảnh bài viết, ảnh nền, reels...
- **RecyclerView (Custom Adapter):**
Dùng thay thế ListView, cho phép hiển thị danh sách bài viết, người theo dõi, danh sách tìm kiếm,... với Adapter tùy chỉnh như PostAdapter, SearchAdapter, ReelAdapter,...
- **SearchView:**
Được dùng trong SearchFragment để tìm người dùng theo username.
- **WebView:**
Nếu bạn có phần liên kết đến website (chưa có trong phiên bản hiện tại), có thể thêm WebView hiển thị thông tin website như điều khoản, giới thiệu,...
- **EditText:**
Dùng để nhập nội dung mô tả bài viết, bình luận, nhập email và mật khẩu khi đăng nhập/đăng ký.
- **Firebase Realtime Storage & Firestore (thay thế SQLite):**
Thay vì SQLite, ứng dụng sử dụng Firebase để:
 - Lưu trữ dữ liệu người dùng, bài viết
 - Tải lên ảnh/video
 - Truy xuất bình luận, số like,...
- **Tài nguyên trong thư mục res và assets:**
Các ảnh minh họa, icon, layout xml,... được lưu trữ trong res. Nếu có file HTML hướng dẫn (ví dụ trang "Giới thiệu"), bạn có thể đặt trong assets.

Các thành phần Android được sử dụng:

STT	Thành phần	Mục đích sử dụng
1	Button	Chuyển màn hình, xác nhận đăng nhập, đăng ký
2	TextView	Hiển thị tên, mô tả, tiêu đề
3	ImageView	Hiển thị ảnh đại diện, bài viết, reels
4	RecyclerView + Adapter	Hiển thị danh sách bài viết, người dùng

5	EditText	Nhập thông tin đăng ký, mô tả bài viết
6	SearchView	Tìm kiếm người dùng trong SearchFragment
7	BottomNavigationView	Chuyển giữa các màn hình chính
8	Firebase	Lưu trữ và truy xuất dữ liệu toàn ứng dụng

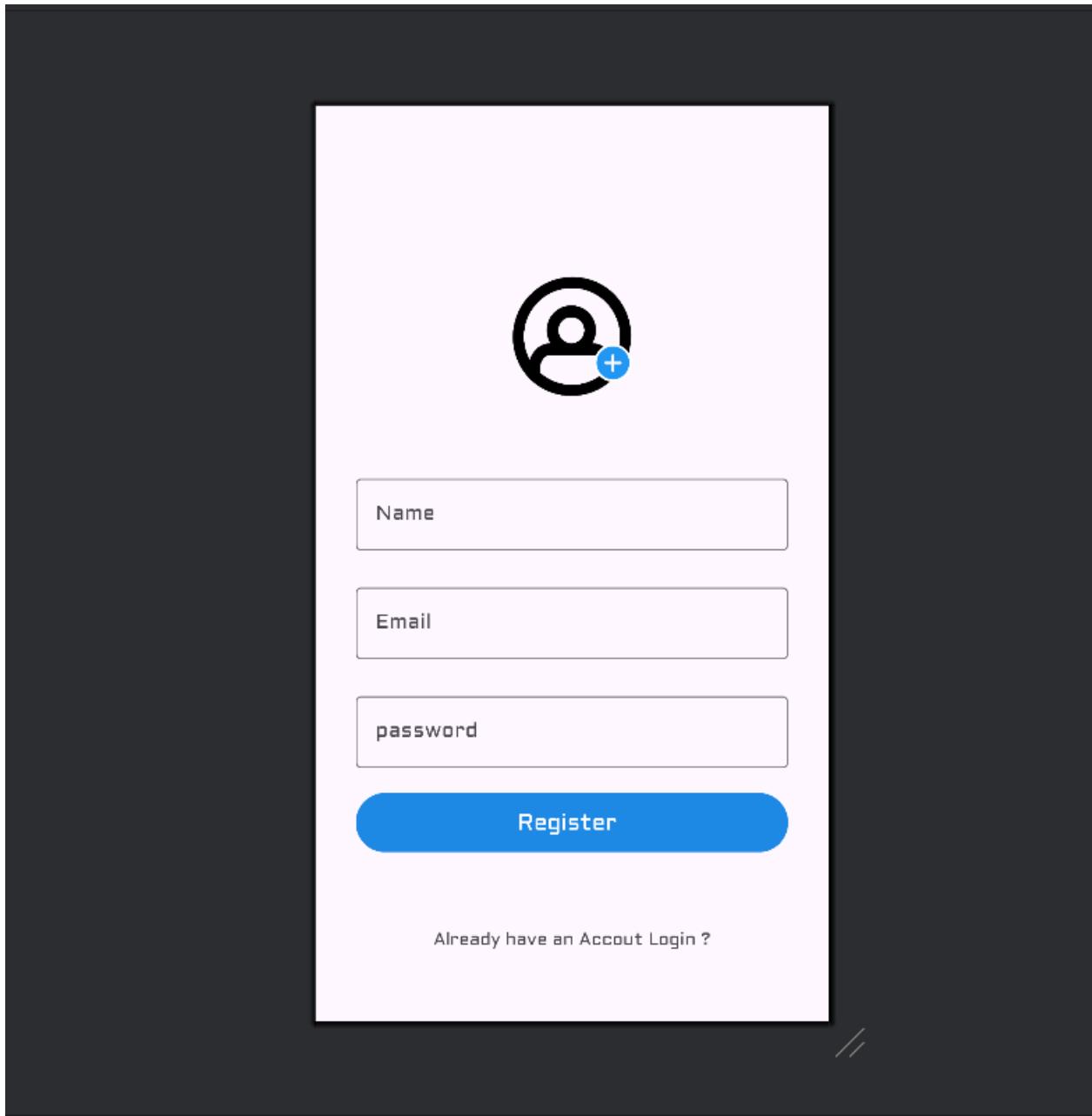
Bảng 3.3.1 Các thành phần Android được sử dụng

3.3.2 Màn hình truy cập



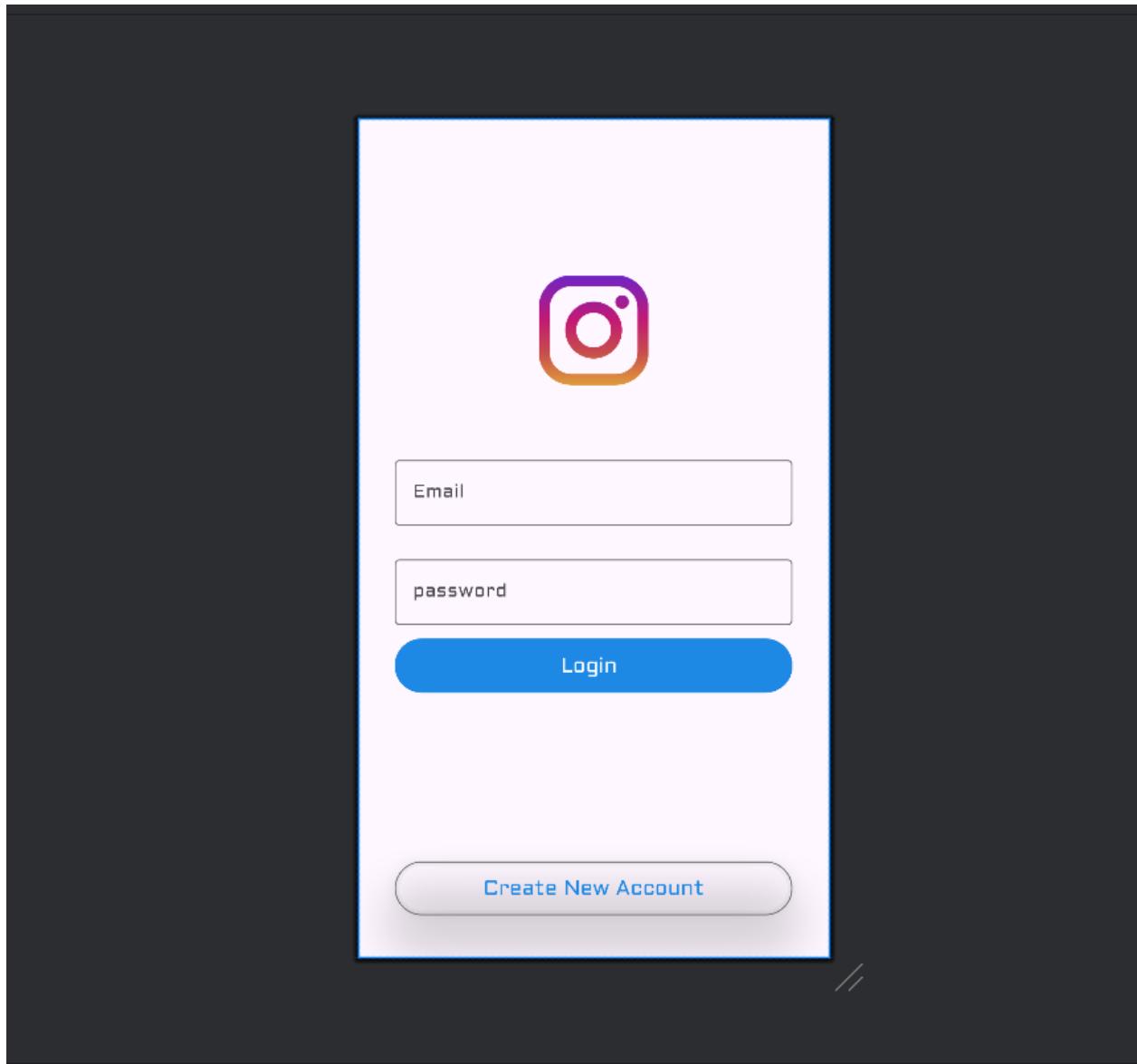
Hình 3.3.2 Giao diện truy cập

3.3.3 Màn hình đăng ký



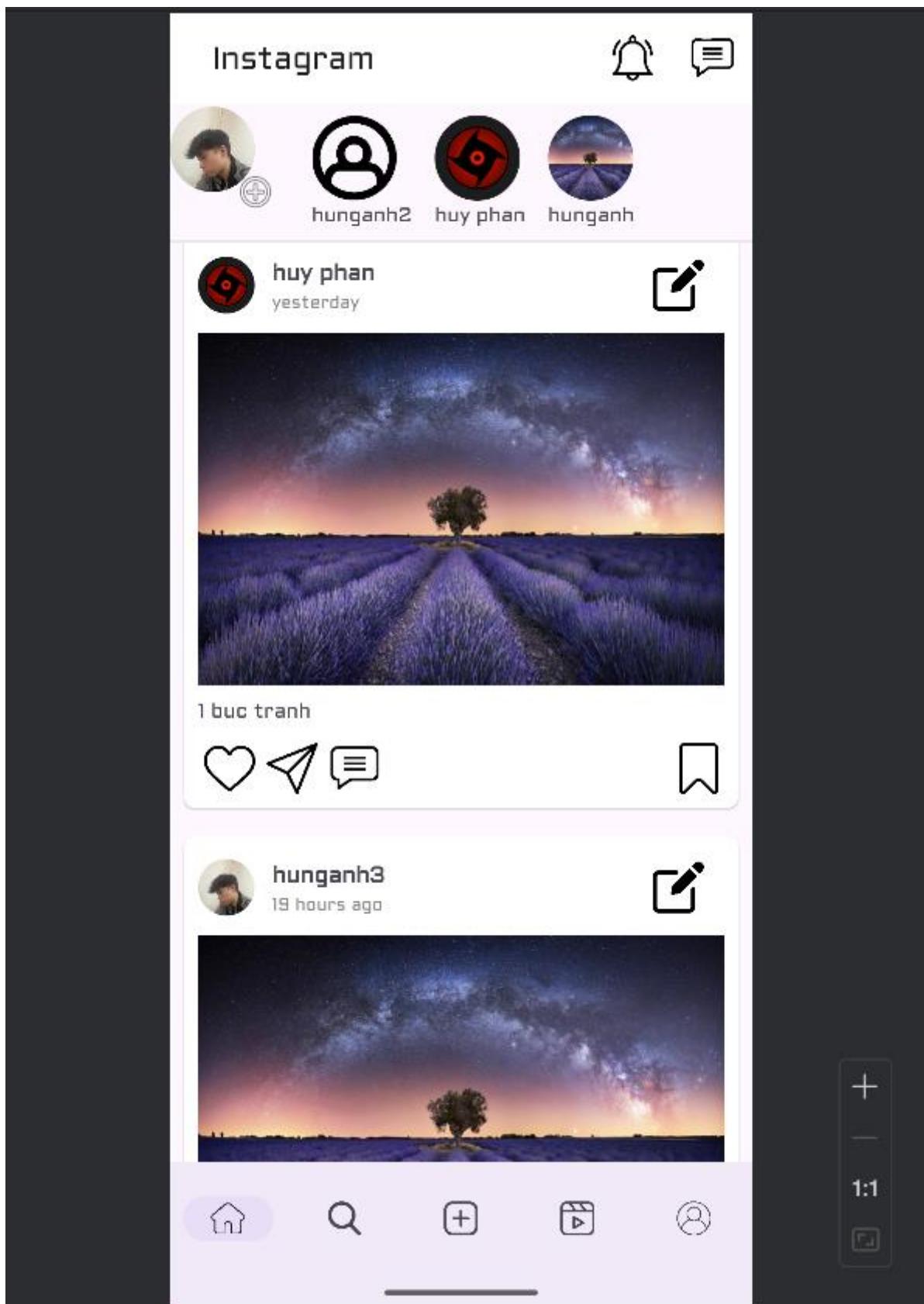
Hình 3.3.3 Giao diện đăng ký

3.3.4 Màn hình đăng nhập



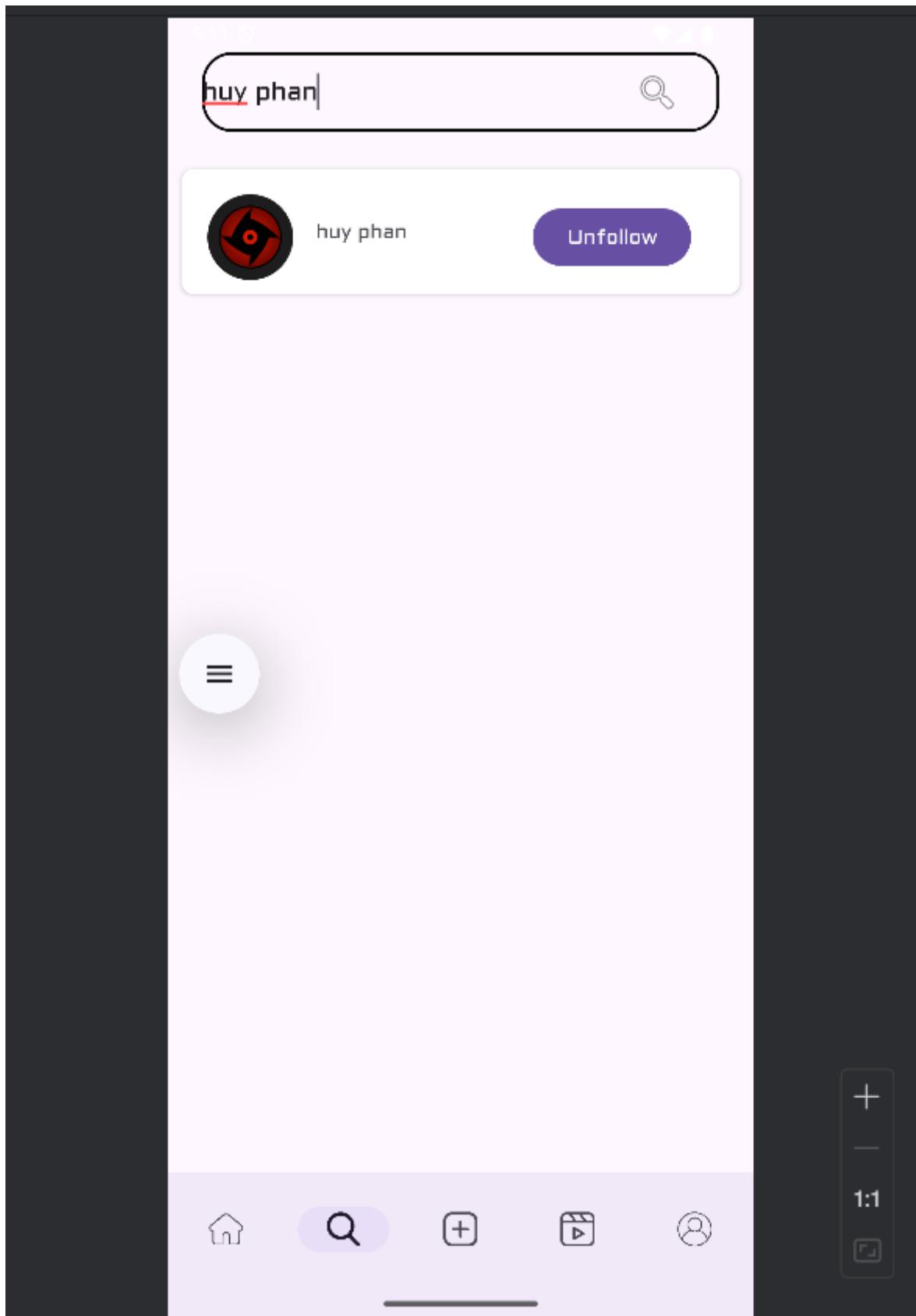
Hình 3.3.4 Giao diện đăng nhập

3.3.5 Màn hình chính



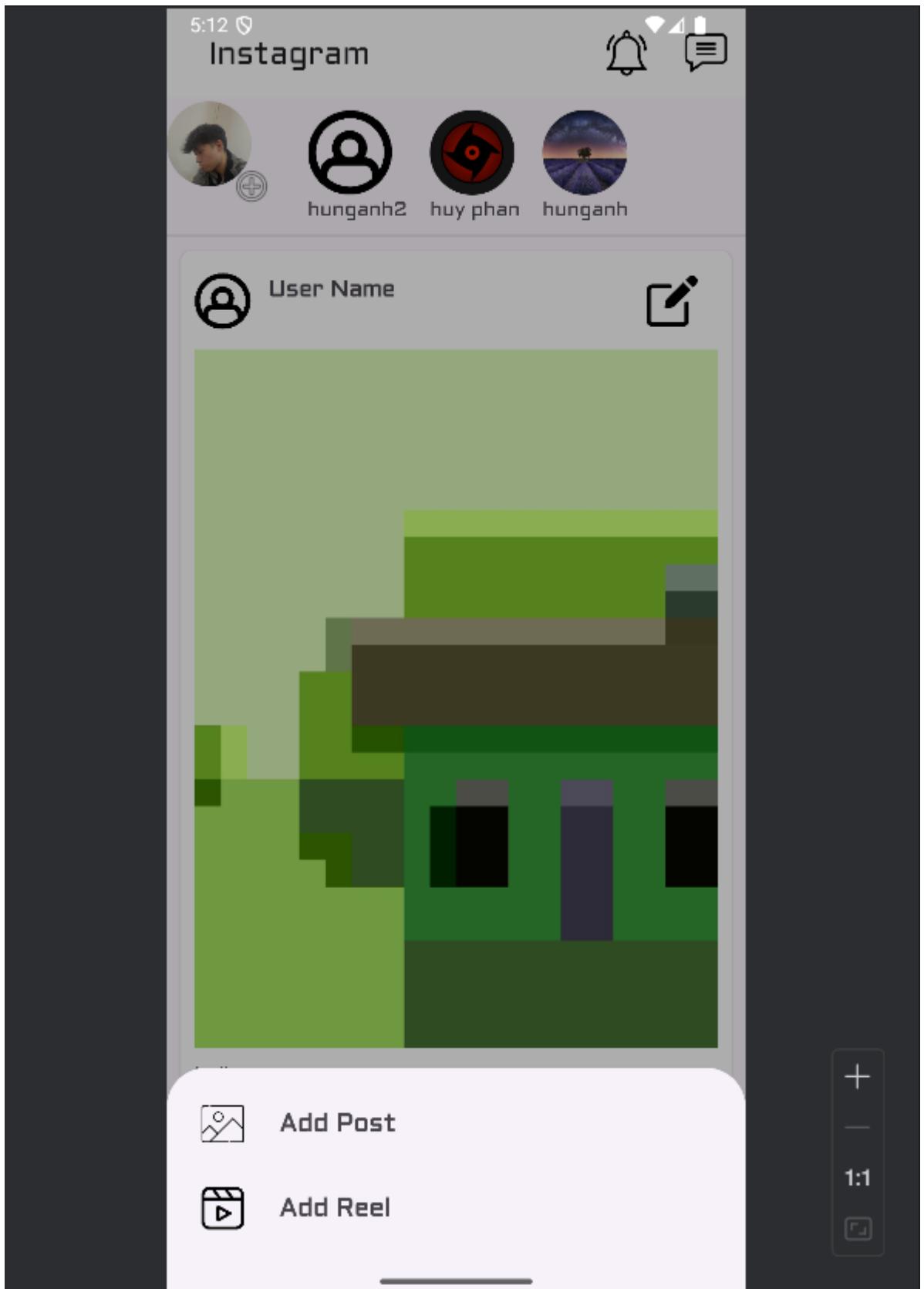
Hình 3.3.5 Giao diện màn hình chính

3.3.6 Màn hình tìm kiếm người dùng



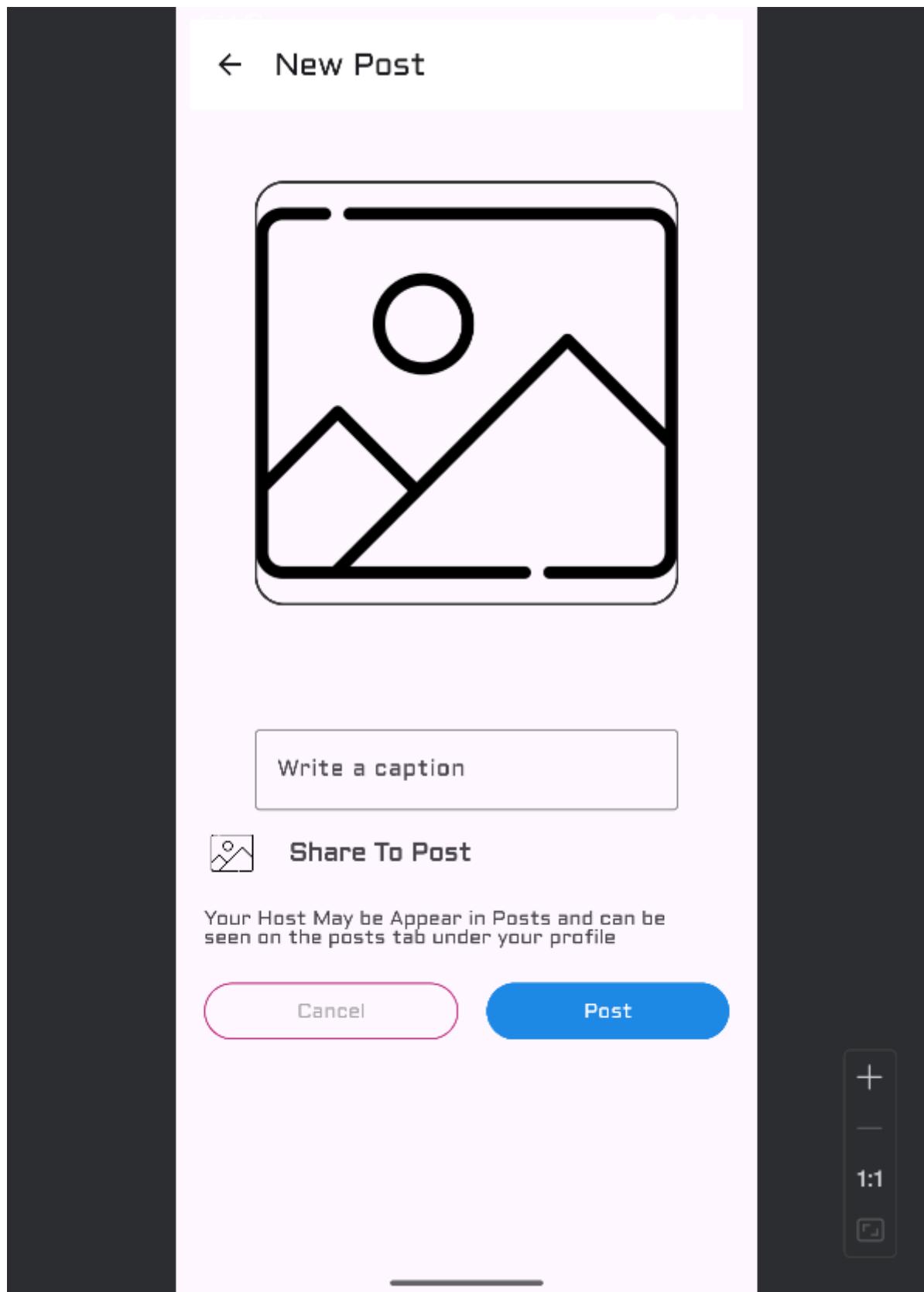
Hình 3.3.6 Giao diện tìm kiếm người dùng

3.3.7 Màn hình đăng bài



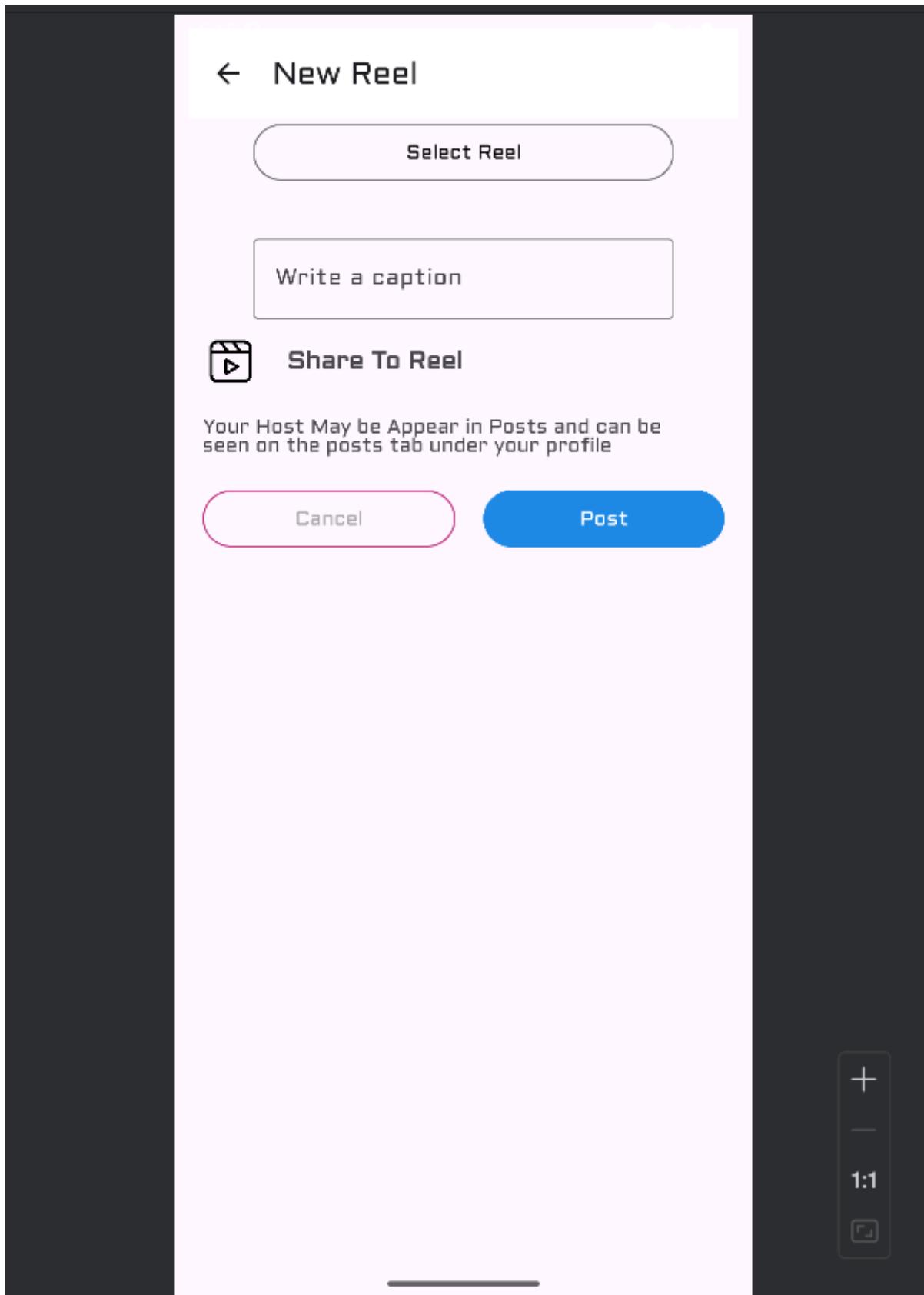
Hình 3.3.7 Giao diện đăng bài

3.3.8 Màn hình chức năng Add Post



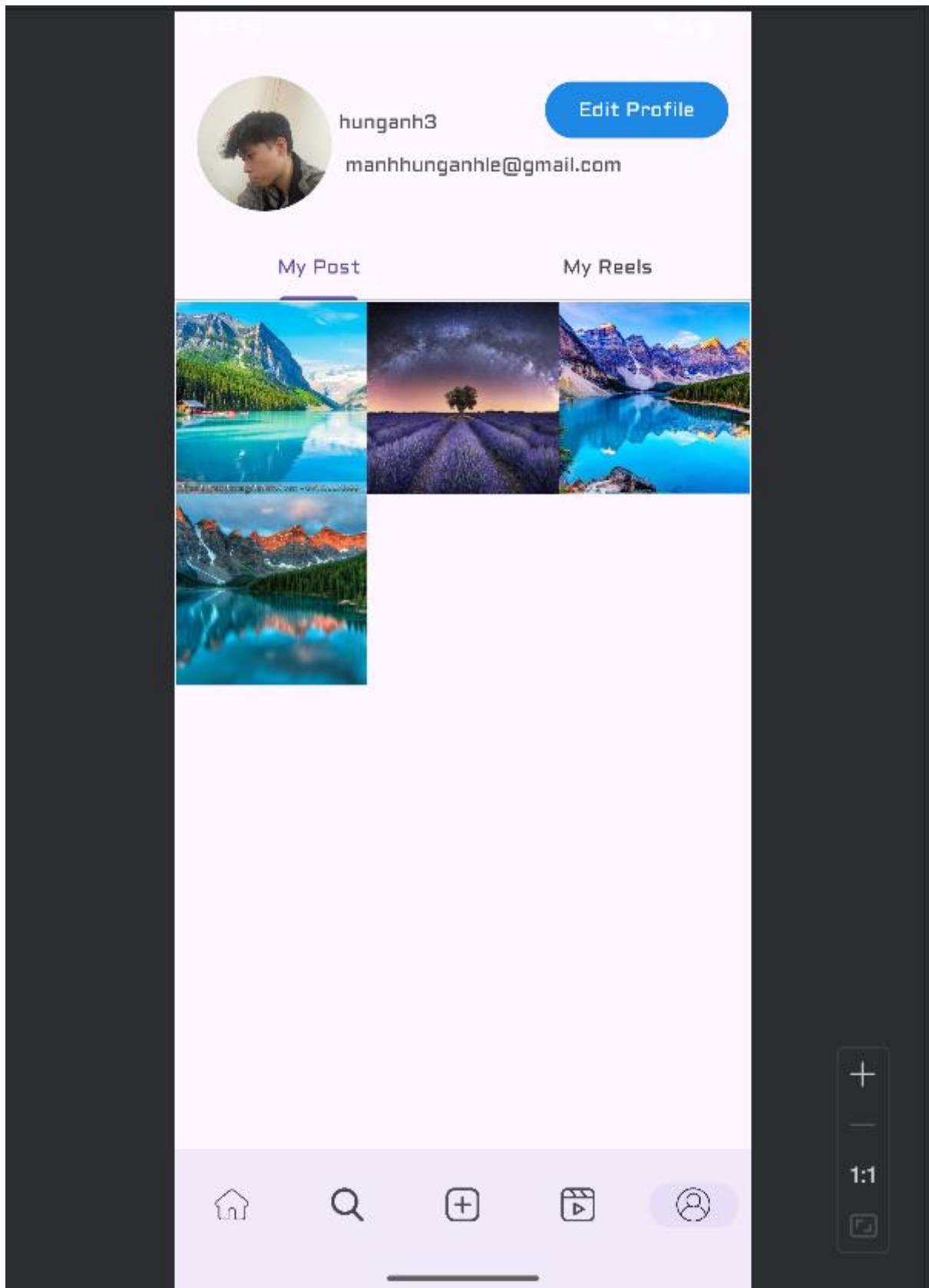
Hình 3.3.8 Giao diện chức năng Add Post

3.3.9 Màn hình chức năng Add Reels



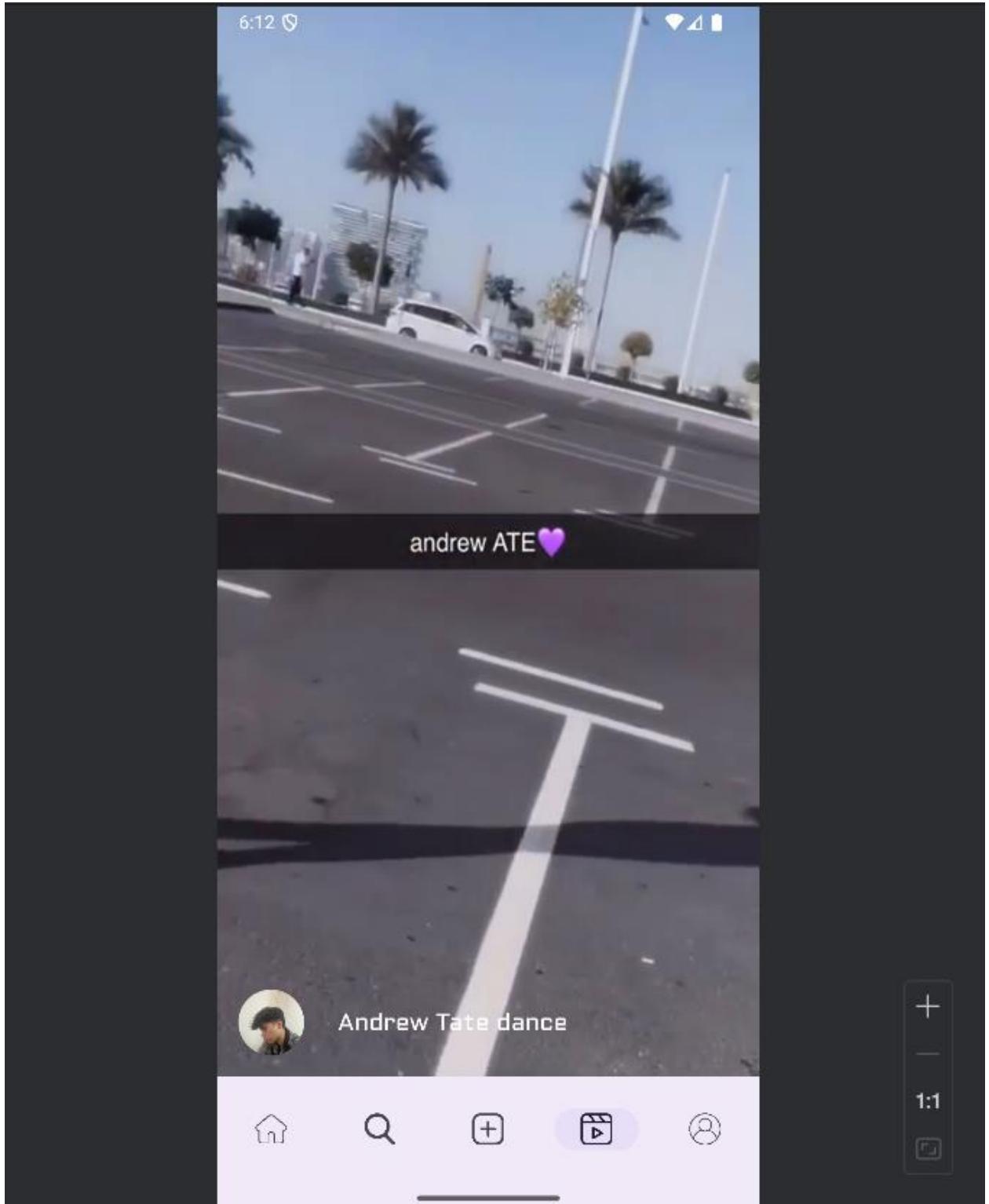
Hình 3.3.9 Giao diện chức năng Add Reels

3.3.10 Màn hình Profile



Hình 3.3.10 Giao diện Profile

3.3.11 Màn hình hiển thị Reels



3.3.11 Giao diện hiển thị Reels

3.4. Code minh họa các chức năng cốt lõi

3.4.1 Chức năng đăng ký SignUpActivity

```
package com.neatroots.instagramclone

import Models.User
import android.content.Intent
import android.os.Bundle
import android.text.Html
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.activity.result.contract.ActivityResultContracts
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import com.google.firebase.Firebase
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.auth
import com.google.firebaseio.firebaseio
import com.google.firebaseio.firebaseio.toObject
import com.neatroots.instagramclone.databinding.ActivitySignUpBinding
import com.neatroots.instagramclone.utils.USER_NODE
import com.neatroots.instagramclone.utils.USER_PROFILE_FOLDER
import com.neatroots.instagramclone.utils.uploadImage
import com.squareup.picasso.Picasso

class SignUpActivity : AppCompatActivity() {
    val binding by lazy {
        ActivitySignUpBinding.inflate(layoutInflater)
    }
    lateinit var user: User
    private val
    launcher=registerForActivityResult(ActivityResultContracts.GetContent()) {
        uri->
        uri?.let {
            uploadImage(uri, USER_PROFILE_FOLDER) {
                if(it!=null) {
                    user.image=it
                    binding.profileImage.setImageURI(uri)
                }
            }
        }
    }
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(binding.root)
        val text = "<font color=#FF000000>Already have an Account</font> <font color=#1E88E5>Login ?</font>"
        binding.login.setText(Html.fromHtml(text))
        user = User()
        if (intent.hasExtra("MODE")) {
            if (intent.getIntExtra("MODE", -1)==1) {
                binding.signUpBtn.text="Update Profile"
            }
        }
        Firebase.firebaseio.collection(USER_NODE).document(Firebase.auth.currentUser!?
        .uid).get().addOnSuccessListener {
            user=it.toObject<User>() !!
            if (!user.image.isNullOrEmpty()) {
                Picasso.get().load(user.image).into(binding.profileImage)
            }
        }
    }
}
```

```

        binding.name.editText?.setText(user.name)
        binding.email.editText?.setText(user.email)
        binding.password.editText?.setText(user.password)
    }
}
binding.signUpBtn.setOnClickListener {
    if (intent.hasExtra("MODE")){
        if (intent.getIntExtra("MODE",-1)==1) {

Firebase.firebaseio.collection(USER_NODE).document(Firebase.auth.currentUser!!
.uid).set(user)
        .addOnSuccessListener {
            startActivity(
                Intent(
                    this@SignUpActivity,
                    HomeActivity::class.java
                )
            )
            finish()
        }
    }

} else {

    if (binding.name.editText?.text.toString().equals("") or
        binding.email.editText?.text.toString().equals("") or
        binding.password.editText?.text.toString().equals(""))
    {

        Toast.makeText(
            this@SignUpActivity,
            "Please fill all the Information",
            Toast.LENGTH_SHORT
        ).show()
    } else {

FirebaseAuth.getInstance().createUserWithEmailAndPassword(
        binding.email.editText?.text.toString(),
        binding.password.editText?.text.toString()
).addOnCompleteListener { result ->

        if (result.isSuccessful) {
            user.name =
binding.name.editText?.text.toString()
            user.email =
binding.email.editText?.text.toString()
            user.password =
binding.password.editText?.text.toString()
            Firebase.firebaseio.collection(USER_NODE)

.document(Firebase.auth.currentUser!!.uid).set(user)
            .addOnSuccessListener {
                startActivity(
                    Intent(
                        this@SignUpActivity,
                        HomeActivity::class.java
                    )
                )
                finish()
            }
        } else {
            Toast.makeText(

```

3.4.2 Chức năng đăng nhập LoginActivity

```
package com.neatroots.instagramclone

import Models.User
import android.content.Intent
import android.os.Bundle
import android.widget.TextView
import android.widget.Toast
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import com.google.android.material.textfield.TextInputEditText
import com.google.firebase.auth.FirebaseAuth
import com.neatroots.instagramclone.databinding.ActivityLoginBinding

class LoginActivity : AppCompatActivity() {
    private lateinit var binding: ActivityLoginBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()

        binding = ActivityLoginBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.loginBtn.setOnClickListener {
            val email = binding.email.editText?.text.toString().trim()
            val password = binding.pass.editText?.text.toString().trim()

            if (email.isEmpty() || password.isEmpty()) {
                Toast.makeText(this, "Please fill all the details",
Toast.LENGTH_SHORT).show()
            } else {
                val user = User(email, password)
            }
        }

        FirebaseAuth.getInstance().signInWithEmailAndPassword(user.email!!,
user.password!!)
            .addOnCompleteListener { task ->
                if (task.isSuccessful) {
                    startActivity(Intent(this,
HomeActivity::class.java))
                    finish()
                }
            }
    }
}
```

```

        } else {
            Toast.makeText(this,
task.exception?.localizedMessage, Toast.LENGTH_SHORT).show()
        }
    }

    val tvForgotPassword = findViewById<TextView>(R.id.tvForgotPassword)

    tvForgotPassword.setOnClickListener {
        val edtEmail = findViewById<TextInputEditText>(R.id.edtEmail)
        val email = edtEmail.text.toString().trim()
        if (email.isNotEmpty()) {
            FirebaseAuth.getInstance().sendPasswordResetEmail(email)
                .addOnSuccessListener {
                    Toast.makeText(this, "Đã gửi email khôi phục!", Toast.LENGTH_SHORT).show()
                }
                .addOnFailureListener {
                    Toast.makeText(this, "Lỗi: ${it.message}", Toast.LENGTH_SHORT).show()
                }
        } else {
            Toast.makeText(this, "Vui lòng nhập email", Toast.LENGTH_SHORT).show()
        }
    }

    binding.createNewAccount.setOnClickListener {
        val intent = Intent(this, SignUpActivity::class.java)
        startActivity(intent)
    }
}
}

```

3.4.3 Chức năng hiển thị màn hình chính HomeFragment

```

package com.neatroots.instagramclone.fragments

import Models.Post
import Models.User
import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.Menu
import android.view.MenuInflater
import android.view.View
import android.view.ViewGroup
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import com.google.firebaseio.Firebase
import com.google.firebase.auth.auth
import com.google.firebaseio.firebaseio.firebaseio
import com.google.firebaseio.firebaseio.toObject
import com.neatroots.instagramclone.R
import com.neatroots.instagramclone.adapters.FollowAdapter
import com.neatroots.instagramclone.adapters.PostAdapter
import com.neatroots.instagramclone.databinding.FragmentHomeBinding
import com.neatroots.instagramclone.utils.FOLLOW
import com.neatroots.instagramclone.utils.POST
import com.neatroots.instagramclone.utils.USER_NODE

```

```

import com.squareup.picasso.Picasso

class HomeFragment : Fragment() {

    private lateinit var binding: FragmentHomeBinding
    private var postList = ArrayList<Post>()
    private lateinit var adapter: PostAdapter
    private var followList = ArrayList<User>()
    private lateinit var followAdapter: FollowAdapter

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentHomeBinding.inflate(inflater, container, false)

        // Setup toolbar
        setHasOptionsMenu(true)
        (requireActivity() as
        AppCompatActivity).setSupportActionBar(binding.materialToolbar3)

        // Setup Post RecyclerView
        adapter = PostAdapter(requireContext(), postList)
        binding.postRv.layoutManager = LinearLayoutManager(requireContext())
        binding.postRv.adapter = adapter

        // Setup Follow RecyclerView
        followAdapter = FollowAdapter(requireContext(), followList)
        binding.followRv.layoutManager =
        LinearLayoutManager(requireContext(), LinearLayoutManager.HORIZONTAL, false)
        binding.followRv.adapter = followAdapter

        // Load current user avatar
        Firebase.firestore.collection(USER_NODE)
            .document(Firebase.auth.currentUser!!.uid)
            .get()
            .addOnSuccessListener { document ->
                val user = document.toObject<User>()
                if (user != null && !user.image.isNullOrEmpty()) {
                    Picasso.get()
                        .load(user.image)
                        .placeholder(R.drawable.user_icon)
                        .into(binding.imageView3)
                }
            }
    }

    // Load follow list
    Firebase.firestore.collection(Firebase.auth.currentUser!!.uid +
FOLLOW)
        .get()
        .addOnSuccessListener {
            val tempList = ArrayList<User>()
            followList.clear()
            for (doc in it.documents) {
                val user = doc.toObject<User>()
                if (user != null) tempList.add(user)
            }
            followList.addAll(tempList)
            followAdapter.notifyDataSetChanged()
        }
}

```

```

    // Load posts
    Firebase.firestore.collection(POST)
        .get()
        .addOnSuccessListener {
            val tempList = ArrayList<Post>()
            postList.clear()
            for (doc in it.documents) {
                val post = doc.toObject<Post>()
                if (post != null) tempList.add(post)
            }
            postList.addAll(tempList)
            adapter.notifyDataSetChanged()
        }

        return binding.root
    }

    override fun onCreateOptionsMenu(menu: Menu, inflater: MenuInflater) {
        inflater.inflate(R.menu.option_menu, menu)
        super.onCreateOptionsMenu(menu, inflater)
    }
}

```

3.4.4 Chức năng hiển thị Profile của tôi ProfileFragment

```

package com.neatroots.instagramclone.fragments

import Models.User
import android.content.Intent
import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import com.google.firebase.Firebase
import com.google.firebase.auth.auth
import com.google.firebase.firestore.firestore
import com.google.firestore.toObject
import com.neatroots.instagramclone.SignUpActivity
import com.neatroots.instagramclone.adapters.ViewPagerAdapter
import com.neatroots.instagramclone.databinding.FragmentProfileBinding
import com.neatroots.instagramclone.utils.USER_NODE
import com.squareup.picasso.Picasso
import androidx.viewpager.widget.ViewPager

class ProfileFragment : Fragment() {
    private lateinit var binding: FragmentProfileBinding
    private lateinit var viewPagerAdapter: ViewPagerAdapter
    private lateinit var myPostFragment: MyPostFragment

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding = FragmentProfileBinding.inflate(inflater, container, false)

        // Nút sửa hồ sơ
        binding.editProfile.setOnClickListener {
            val intent = Intent(activity, SignUpActivity::class.java)
            intent.putExtra("MODE", 1)
            activity?.startActivity(intent)
            activity?.finish()
        }
    }
}

```

```

        }

        // Khởi tạo Fragment
        myPostFragment = MyPostFragment()

        // Setup ViewPager
        viewPagerAdapter =
ViewPagerAdapter(requireActivity().supportFragmentManager)
        viewPagerAdapter.addFragment(myPostFragment, "My Post")
        viewPagerAdapter.addFragment(MyReelsFragment(), "My Reels")
        binding.viewPager.adapter = viewPagerAdapter
        binding.tabLayout.setupWithViewPager(binding.viewPager)

        // Reload post mỗi khi chọn lại tab 0
        binding.viewPager.addOnPageChangeListener(object :
ViewPager.OnPageChangeListener {
            override fun onPageScrolled(position: Int, positionOffset: Float,
positionOffsetPixels: Int) {}
            override fun onPageSelected(position: Int) {
                if (position == 0) {
                    myPostFragment.reloadPosts()
                }
            }
            override fun onPageScrollStateChanged(state: Int) {}
        })

        return binding.root
    }

    override fun onStart() {
        super.onStart()
        // Load thông tin người dùng

        Firebase.firestore.collection(USER_NODE).document(Firebase.auth.currentUser!.
.uid)
            .get().addOnSuccessListener {
                val user: User = it.toObject<User>() !!
                binding.name.text = user.name
                binding.bio.text = user.email
                if (!user.image.isNullOrEmpty()) {
                    Picasso.get().load(user.image).into(binding.profileImage)
                }
            }
    }
}

```

3.4.5 Chức năng đăng bài PostAdapter

```

package com.neatroots.instagramclone.adapters

import Models.Post
import Models.User
import android.content.Context
import android.content.Intent
import android.view.LayoutInflater
import android.view.ViewGroup
import android.widget.PopupMenu
import android.widget.Toast
import androidx.recyclerview.widget.RecyclerView
import com.bumptech.glide.Glide
import com.github.marlonlom.utilities.timeago.TimeAgo
import com.google.firebase.firestore.ktx.firebaseio
import com.google.firebase.ktx.firebaseio

```

```

import com.google.firebaseio.firebaseio.toObject
import com.neatroots.instagramclone.R
import com.neatroots.instagramclone.databinding.PostRvBinding
import com.neatroots.instagramclone.utils.USER_NODE

class PostAdapter(
    private val context: Context,
    private val postList: ArrayList<Post>
) : RecyclerView.Adapter<PostAdapter.ViewHolder>() {

    private val likedPositions = HashSet<Int>() // ✅ Trạng thái like theo
    vị trí

    inner class ViewHolder(val binding: PostRvBinding) :
    RecyclerView.ViewHolder(binding.root)

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
        val binding = PostRvBinding.inflate(LayoutInflater.from(context),
        parent, false)
        return ViewHolder(binding)
    }

    override fun getItemCount(): Int = postList.size

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        val post = postList[position]

        // Load thông tin người dùng nếu có uid
        val uid = post.uid
        if (!uid.isNullOrEmpty()) {
            FirebaseFirestore.collection(USER_NODE)
                .document(uid)
                .get()
                .addOnSuccessListener { documentSnapshot ->
                    val user = documentSnapshot.toObject<User>()
                    user?.let {
                        Glide.with(context)
                            .load(it.image)
                            .placeholder(R.drawable.user_icon)
                            .into(holder.binding.profileImage)

                        holder.binding.name.text = it.name
                    }
                }
                .addOnFailureListener {
                    // Xử lý khi lỗi Firestore (nếu cần)
                }
        }

        // Load ảnh bài đăng
        Glide.with(context)
            .load(post.postUrl)
            .placeholder(R.drawable.loading)
            .into(holder.binding.postImage)

        // Set thời gian đăng bài
        try {
            val text = TimeAgo.using(post.time.toLong())
            holder.binding.time.text = text
        } catch (e: Exception) {
            holder.binding.time.text = ""
        }
    }
}

```

```

    }

    // Chia sẻ bài viết
    holder.binding.share.setOnClickListener {
        val intent = Intent(Intent.ACTION_SEND)
        intent.type = "text/plain"
        intent.putExtra(Intent.EXTRA_TEXT, post.postUrl)
        context.startActivity(intent)
    }

    // Caption bài viết
    holder.binding.csoption.text = post.caption

    //  Xử lý hiển thị trạng thái like
    val isLiked = likedPositions.contains(position)
    holder.binding.like.setImageResource(
        if (isLiked) R.drawable.heart_icon else R.drawable.heart
    )

    //  Toggle trạng thái khi nhấp
    holder.binding.like.setOnClickListener {
        if (likedPositions.contains(position)) {
            likedPositions.remove(position)
            holder.binding.like.setImageResource(R.drawable.heart)
        } else {
            likedPositions.add(position)
            holder.binding.like.setImageResource(R.drawable.heart_icon)
        }
    }

    holder.binding.menuButton.setOnClickListener {
        val popupMenu = PopupMenu(holder.itemView.context,
        holder.binding.menuButton)
        popupMenu.menuInflater.inflate(R.menu.post_menu, popupMenu.menu)

        popupMenu.setOnMenuItemClickListener {
            when (it.itemId) {
                R.id.deletePost -> {
                    Toast.makeText(holder.itemView.context, "Xóa bài
viết", Toast.LENGTH_SHORT).show()
                    true
                }
                R.id.editPost -> {
                    Toast.makeText(context, "Chỉnh sửa",
                    Toast.LENGTH_SHORT).show()
                    true
                }
                else -> false
            }
        }
        popupMenu.show()
    }
}
}

```

3.4.6 Chức năng tìm người dùng SearchAdapter

```

package com.neatroots.instagramclone.adapters

import Models.Post

```

```

import Models.User
import android.content.Context
import android.content.Intent
import android.view.LayoutInflater
import android.view.ViewGroup
import android.widget.PopupMenu
import android.widget.Toast
import androidx.recyclerview.widget.RecyclerView
import com.bumptech.glide.Glide
import com.github.marlonlom.utilities.timeago.TimeAgo
import com.google.firebase.firestore.ktx.firebaseio
import com.google.firebase.ktx.Firebase
import com.google.firebase.toObject
import com.neatroots.instagramclone.R
import com.neatroots.instagramclone.databinding.PostRvBinding
import com.neatroots.instagramclone.utils.USER_NODE

class PostAdapter(
    private val context: Context,
    private val postList: ArrayList<Post>
) : RecyclerView.Adapter<PostAdapter.ViewHolder>() {

    private val likedPositions = HashSet<Int>() // ☑ Trạng thái like theo
    vị trí

    inner class ViewHolder(val binding: PostRvBinding) :
    RecyclerView.ViewHolder(binding.root)

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
        val binding = PostRvBinding.inflate(LayoutInflater.from(context),
        parent, false)
        return ViewHolder(binding)
    }

    override fun getItemCount(): Int = postList.size

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        val post = postList[position]

        // Load thông tin người dùng nếu có uid
        val uid = post.uid
        if (!uid.isNullOrEmpty()) {
            Firebase.firebaseio.collection(USER_NODE)
                .document(uid)
                .get()
                .addOnSuccessListener { documentSnapshot ->
                    val user = documentSnapshot.toObject<User>()
                    user?.let {
                        Glide.with(context)
                            .load(it.image)
                            .placeholder(R.drawable.user_icon)
                            .into(holder.binding.profileImage)

                        holder.binding.name.text = it.name
                    }
                }
                .addOnFailureListener {
                    // Xử lý khi lỗi Firestore (nếu cần)
                }
        }
    }
}

```

```

// Load ảnh bài đăng
Glide.with(context)
    .load(post.imageUrl)
    .placeholder(R.drawable.loading)
    .into(holder.binding.postImage)

// Set thời gian đăng bài
try {
    val text = TimeAgo.using(post.time.toLong())
    holder.binding.time.text = text
} catch (e: Exception) {
    holder.binding.time.text = ""
}

// Chia sẻ bài viết
holder.binding.share.setOnClickListener {
    val intent = Intent(Intent.ACTION_SEND)
    intent.type = "text/plain"
    intent.putExtra(Intent.EXTRA_TEXT, post.imageUrl)
    context.startActivity(intent)
}

// Caption bài viết
holder.binding.csoption.text = post.caption

//  Xử lý hiển thị trạng thái like
val isLiked = likedPositions.contains(position)
holder.binding.like.setImageResource(
    if (isLiked) R.drawable.heart_icon else R.drawable.heart
)

//  Toggle trạng thái khi nhấn
holder.binding.like.setOnClickListener {
    if (likedPositions.contains(position)) {
        likedPositions.remove(position)
        holder.binding.like.setImageResource(R.drawable.heart)
    } else {
        likedPositions.add(position)
        holder.binding.like.setImageResource(R.drawable.heart_icon)
    }
}

holder.binding.menuButton.setOnClickListener {
    val popupMenu = PopupMenu(holder.itemView.context,
    holder.binding.menuButton)
    popupMenu.menuInflater.inflate(R.menu.post_menu, popupMenu.menu)

    popupMenu.setOnMenuItemClickListener {
        when (it.itemId) {
            R.id.deletePost -> {
                Toast.makeText(holder.itemView.context, "Xóa bài
viết", Toast.LENGTH_SHORT).show()
                true
            }
            R.id.editPost -> {
                Toast.makeText(context, "Chỉnh sửa",
                Toast.LENGTH_SHORT).show()
                true
            }
            else -> false
        }
    }
}

```

```
    popupMenu.show()  
}  
}  
}
```

3.4.7 Chức năng đăng tải Reels ReelAdapter

```
package com.neatroots.instagramclone.adapters

import Models.Reel
import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.neatroots.instagramclone.R
import com.neatroots.instagramclone.databinding.ReelDgBinding
import com.squareup.picasso.Picasso

class ReelAdapter(var context: Context, var reelList: ArrayList<Reel>)
: RecyclerView.Adapter<ReelAdapter.ViewHolder>() {
    inner class ViewHolder(var binding: ReelDgBinding) :
        RecyclerView.ViewHolder(binding.root)

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
        var binding = ReelDgBinding.inflate(LayoutInflater.from(context), parent, false)
        return ViewHolder(binding)
    }

    override fun getItemCount(): Int {
        return reelList.size
    }

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        Picasso.get().load(reelList.get(position).profileLink).placeholder(R.drawable.user_icon).into(holder.binding.profileImage)
        holder.binding.caption.setText(reelList.get(position).caption)
        holder.binding.videoView.setVideoPath(reelList.get(position).reelUrl)
        holder.binding.videoView.setOnPreparedListener {
            holder.binding.progressBar.visibility = View.GONE
            holder.binding.videoView.start()
        }
    }
}
```

3.4.8 Chức năng follow người dùng FollowAdapter

```
package com.neatroots.instagramclone.adapters

import Models.Reel
import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.neatroots.instagramclone.R
import com.neatroots.instagramclone.databinding.ReelDgBinding
```

```
import com.squareup.picasso.Picasso

class ReelAdapter(var context: Context, var reelList: ArrayList<Reel>)
: RecyclerView.Adapter<ReelAdapter.ViewHolder>() {
    inner class ViewHolder(var binding: ReelDgBinding) :
        RecyclerView.ViewHolder(binding.root)

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
        var binding = ReelDgBinding.inflate(LayoutInflater.from(context),
parent, false)
        return ViewHolder(binding)
    }

    override fun getItemCount(): Int {
        return reelList.size
    }

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {

Picasso.get().load(reelList.get(position).profileLink).placeholder(R.drawable
.user_icon).into(holder.binding.profileImage)
        holder.binding.caption.setText(reelList.get(position).caption)
        holder.binding.videoView.setVideoPath(reelList.get(position).reelUrl)
        holder.binding.videoView.setOnPreparedListener {
            holder.binding.progressBar.visibility = View.GONE
            holder.binding.videoView.start()

        }
    }
}
```

KẾT LUẬN

1. Kết quả đạt được

Qua quá trình nghiên cứu và triển khai, nhóm chúng em đã hoàn thành phiên bản đầu tiên của ứng dụng **Instagram Clone** với các chức năng cơ bản nhất của một nền tảng mạng xã hội chia sẻ hình ảnh. Cụ thể:

- Xây dựng thành công giao diện người dùng bằng **Kotlin và Android XML**, thân thiện và dễ sử dụng.
- Tích hợp hệ thống **đăng ký, đăng nhập và chỉnh sửa hồ sơ** sử dụng Firebase Authentication.
- Cho phép người dùng **đăng bài viết (ảnh, mô tả)** và hiển thị chúng dưới dạng danh sách và dạng lưới.
- Thực hiện tính năng **bình luận và tương tác với bài viết** (comment).
- Hiển thị danh sách người dùng đang theo dõi, người theo dõi (follow), hỗ trợ tìm kiếm người dùng.
- Quản lý dữ liệu qua **Firebase Firestore và Firebase Storage**, đảm bảo đồng bộ và thời gian thực.
- Áp dụng kiến thức về Fragment, Adapter, RecyclerView, Intent, Firebase, giúp nhóm nâng cao kỹ năng phát triển ứng dụng thực tế.

2. Nhược điểm

Dù đã đạt được nhiều kết quả tích cực, ứng dụng vẫn còn một số hạn chế nhất định:

- Giao diện chưa được tối ưu hoàn toàn theo chuẩn UI/UX chuyên nghiệp.
- Các chức năng như xem story, bình luận bài viết, chỉnh sửa bài viết, nhắn tin, thông báo và điều hướng đề xuất vẫn chưa được hoàn thiện
- CSDL còn sơ xài chưa tối ưu
- Chưa xử lý triệt để các trường hợp lỗi như mất kết nối mạng, đăng ảnh thất bại,...
- Chưa có cơ chế xác minh tài khoản hoặc bảo mật hai lớp (2FA).
- Hiệu năng có thể bị ảnh hưởng nếu dữ liệu người dùng tăng cao do chưa tối ưu truy vấn Firebase.

3. Hướng phát triển

Để nâng cấp ứng dụng trở nên hoàn chỉnh hơn và gần với Instagram thực tế, nhóm dự kiến phát triển thêm một số tính năng trong tương lai:

- Hoàn thiện những tính năng tương tác trên bài viết.
- Cải thiện UI/UX, sử dụng Jetpack Compose hoặc thư viện thiết kế hiện đại.
- Tích hợp chức năng đăng story và xem story như Instagram.
- Tăng cường bảo mật, thêm xác minh tài khoản và giới hạn đăng nhập sai.
- Tối ưu hóa truy vấn và lưu trữ dữ liệu, tránh tải toàn bộ dữ liệu không cần thiết.

- Thêm hệ thống thông báo real-time (Firebase Cloud Messaging).
- Triển khai trên nhiều thiết bị và hỗ trợ Dark Mode.
- Tích hợp các thuật toán phù hợp để điều hướng để xuất nội dung cho từng đối tượng người dùng khác nhau

TÀI LIỆU THAM KHẢO

1. CSE441.202502. Thư mục bài giảng Lập trình di động k55 / ThS Kiều Tuấn Dũng
2. Instagram Clone - Build an Instagram Clone App in Android - Complete Android Studio Project / Neetroots Youtube