

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1**

o0o



BÀI TẬP LỚN

**Đề tài: Ứng dụng ChatApp tích hợp chuyển đổi ảnh
Selfie2Anime**

Môn học: Chuyên đề Hệ Thống Thông tin

Lớp: 03

Số thứ tự nhóm: 12

Lưu Trung Kiên MSSV: B21DCCN071

Phạm Văn Tiến MSSV: B21DCCN708

Nguyễn Minh Thắng MSSV: B21DCCN668

Nguyễn Đức Quỳnh MSSV: B21DCCN646

Giảng viên hướng dẫn: Nguyễn Trọng Khánh

HÀ NỘI, 2025

MỤC LỤC

CHƯƠNG 1. TỔNG QUAN VỀ BÀI TẬP LỚN	1
1.1 Giới thiệu đề tài.....	1
1.2 Lý do chọn đề tài	1
1.3 Mục tiêu dự án	2
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT VÀ CÔNG NGHỆ SỬ DỤNG	3
2.1 Mô hình Selfie2Anime	3
2.1.1 Giới thiệu mô hình U-GAT-IT	3
2.1.2 Mạng tạo sinh không giám sát với sự chuẩn hóa lớp thích ứng (Unsupervised Generative Attentional Networks with Adaptive Layer- Instance Normalization).....	3
2.2 LiteRT	7
2.2.1 Tổng quan	7
2.2.2 Tính năng chính	7
2.2.3 Quy trình phát triển và tích hợp	7
2.3 Các công nghệ hỗ trợ phát triển ChatApp.....	8
2.3.1 Android Studio với ngôn ngữ Java	8
2.3.2 Firebase	8
2.3.3 Cloudinary	9
2.3.4 Thư viện và công nghệ khác.....	9
CHƯƠNG 3. KIẾN TRÚC ỨNG DỤNG	11
3.1 Kiến trúc chung hệ thống	11
3.2 Giao diện người dùng (Frontend)	11
3.3 Phần lõi ứng dụng (Backend).....	14
3.3.1 Đăng nhập và đăng ký tài khoản.....	14
3.3.2 Xử lý và hiển thị giao diện chính	14

3.3.3 Quản lý thông tin cá nhân	14
3.3.4 Quản lý danh sách người dùng và tìm kiếm.....	15
3.3.5 Xử lý hội thoại	15
3.3.6 Tích hợp tính năng chuyển đổi ảnh Selfie2Anime.....	15
CHƯƠNG 4. CÀI ĐẶT ỨNG DỤNG.....	17
4.1 Cài đặt mô hình Selfie2Anime	17
4.2 Ứng dụng ChatApp	19
4.2.1 Hướng dẫn cài đặt	19
CHƯƠNG 5. DEMO.....	22
TÀI LIỆU THAM KHẢO.....	29

DANH MỤC HÌNH VẼ

Hình 2.1	Kiến trúc của U-GAT-IT	4
Hình 2.2	Trực quan hóa các bản đồ sự tập trung và ảnh hưởng của các bức ảnh được thể hiện trong các thí nghiệm bị loại bỏ	6
Hình 3.1	Kiến trúc chung hệ thống	11
Hình 3.2	Giao diện ứng dụng	12
Hình 3.3	Sơ đồ hoạt động	13
Hình 4.1	Cài đặt mô hình	19
Hình 4.2	Cài đặt ứng dụng android	20
Hình 4.3	Cài đặt ứng dụng android	21

CHƯƠNG 1. TỔNG QUAN VỀ BÀI TẬP LỚN

1.1 Giới thiệu đề tài

Trong thời đại số, ứng dụng di động ngày càng trở nên thiết yếu, không chỉ phục vụ nhu cầu giao tiếp mà còn đáp ứng mong muốn về giải trí và cá nhân hóa trải nghiệm. Các ứng dụng nhắn tin hiện nay cạnh tranh khốc liệt, đòi hỏi phải có những tính năng nổi bật để thu hút và giữ chân người dùng.

Dự án “Ứng dụng ChatApp tích hợp chuyển đổi ảnh Selfie2Anime” được hình thành từ ý tưởng kết hợp hai lĩnh vực:

- Chức năng ChatApp: Ứng dụng cho phép người dùng gửi tin nhắn, chia sẻ ảnh, quản lý thông tin cá nhân (bao gồm tạo mã QR, thay đổi mật khẩu, tìm kiếm bạn bè, nhận thông báo, ...) với giao diện thân thiện, hiện đại.
- Tính năng chuyển đổi ảnh Selfie2Anime: Công nghệ chuyển đổi ảnh sử dụng mô hình U-GAT-IT cùng với TensorFlow Lite giúp biến đổi ảnh chân dung của người dùng sang phong cách hoạt hình (anime) độc đáo.

Sự kết hợp này không chỉ làm phong phú thêm trải nghiệm của ứng dụng mà còn tạo ra điểm nhấn về tính sáng tạo và ứng dụng công nghệ AI tiên tiến trong lĩnh vực di động.

1.2 Lý do chọn đề tài

Việc lựa chọn đề tài “Ứng dụng ChatApp tích hợp chuyển đổi ảnh Selfie2Anime” dựa trên các yếu tố sau:

- Nhu cầu giải trí và cá nhân hóa: Người dùng ngày nay không chỉ tìm kiếm công cụ giao tiếp mà còn mong muốn những trải nghiệm sáng tạo và giải trí độc đáo. Tính năng chuyển đổi ảnh giúp người dùng thể hiện cá tính và tạo dấu ấn riêng qua hình ảnh của mình.
- Ứng dụng công nghệ tiên tiến: Mô hình U-GAT-IT và việc tối ưu hóa qua TensorFlow Lite đã được chứng minh là hiệu quả trong việc chuyển đổi ảnh. Khi tích hợp vào ChatApp, công nghệ này giúp nâng cao giá trị trải nghiệm của người dùng bằng cách kết hợp giao tiếp và sáng tạo hình ảnh.
- Khả năng tích hợp liền mạch: Thay vì xây dựng hai dự án riêng biệt, việc tích hợp tính năng chuyển đổi ảnh Selfie2Anime vào ChatApp cho phép tạo ra một hệ thống đồng bộ. Người dùng có thể vừa trò chuyện, vừa sử dụng tính năng chuyển đổi ảnh mà không cần chuyển qua lại giữa các ứng dụng khác nhau.
- Tiềm năng phát triển và mở rộng: Dự án hướng tới việc tạo ra một nền tảng

giao tiếp đa năng, có khả năng mở rộng các tính năng mới trong tương lai, đáp ứng nhu cầu ngày càng đa dạng của người dùng, từ nhắn tin cơ bản đến tích hợp trí tuệ nhân tạo trong xử lý ảnh.

1.3 Mục tiêu dự án

Mục tiêu chính

Mục tiêu của đề tài này nhằm xây dựng một ứng dụng nhắn tin với giao diện thân thiện dễ sử dụng và bổ sung tính năng chuyển đổi ảnh Selfie2Anime, nhằm mang lại trải nghiệm giao tiếp và giải trí độc đáo cho người dùng.

Các mục tiêu cụ thể:

- Chức năng nhắn tin và quản lý người dùng:
 - Hỗ trợ đăng ký, đăng nhập và quản lý tài khoản.
 - Gửi và nhận tin nhắn văn bản, hình ảnh.
 - Tìm kiếm, kết nối bạn bè và chia sẻ thông tin cá nhân qua các công cụ như mã QR.
- Tính năng chuyển đổi ảnh Selfie2Anime:
 - Áp dụng mô hình U-GAT-IT kết hợp với TensorFlow Lite để chuyển đổi ảnh selfie sang phong cách anime.
 - Tích hợp liền mạch vào quy trình gửi ảnh và chia sẻ thông tin trong ứng dụng, giúp người dùng có thể áp dụng hiệu ứng nghệ thuật ngay trong khi giao tiếp.
- Thiết kế giao diện hiện đại và thân thiện:
 - Xây dựng giao diện người dùng trực quan, dễ sử dụng, đảm bảo trải nghiệm mượt mà trên nền tảng Android.
 - Tối ưu hóa bố cục và tương tác nhằm nâng cao tính thẩm mỹ và hiệu quả của ứng dụng.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT VÀ CÔNG NGHỆ SỬ DỤNG

2.1 Mô hình Selfie2Anime

2.1.1 Giới thiệu mô hình U-GAT-IT

Mạng GAN là mô hình được đề xuất trong bài báo khoa học với tiêu đề “Mạng tạo sinh không giám sát với sự chuẩn hóa lớp thích ứng cho việc chuyển đổi giữa các ảnh với nhau”. Việc chuyển đổi ảnh – ảnh nhằm tìm hiểu ánh xạ giữa hai miền khác nhau.

Trong các nghiên cứu ban đầu, khi dữ liệu ghép cặp được sử dụng, mô hình ánh xạ thường được huấn luyện có giám sát (ví dụ: sử dụng mô hình tạo sinh có điều kiện hoặc mô hình hồi quy). Còn trong bối cảnh không có dữ liệu ghép cặp, nhiều công trình đã chuyển sang sử dụng shared latent space để xử lý tính đa phương thức của bài toán.

Tuy nhiên, hiệu suất của các phương pháp trước đó phụ thuộc mạnh vào mức độ thay đổi về hình dạng và kết cấu giữa các miền, do đó việc tiền xử lý như cắt xén, căn chỉnh ảnh trở nên cần thiết.

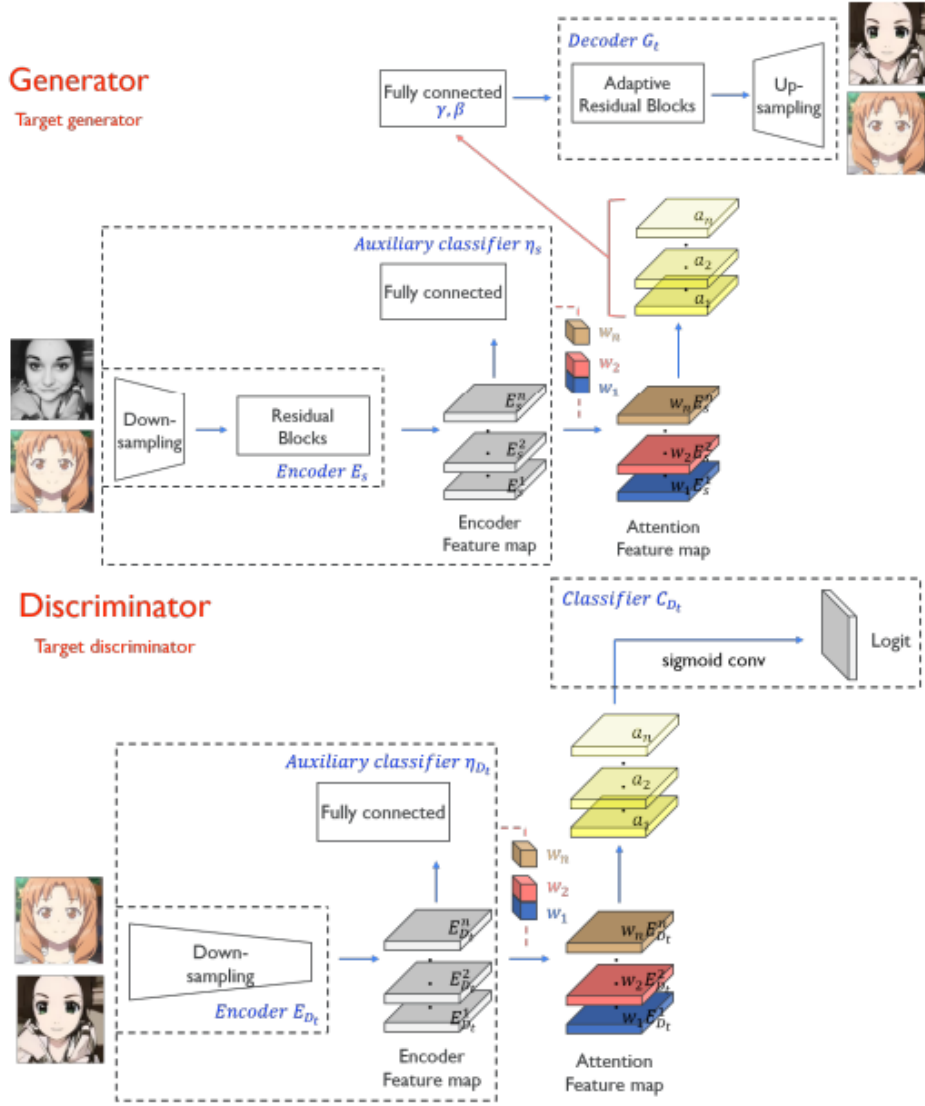
Để khắc phục những hạn chế này, một phương pháp mới đã được đề xuất với việc bổ sung attention module và một hàm chuẩn hóa mới gọi là AdaLIN. Attention module giúp mô hình biết nên dịch chuyển ảnh như thế nào bằng cách phân biệt miền nguồn và miền đích dựa trên các thông tin từ bộ phân loại phụ trợ.

2.1.2 Mạng tạo sinh không giám sát với sự chuẩn hóa lớp thích ứng (Unsupervised Generative Attentional Networks with Adaptive Layer-Instance Normalization)

Mục tiêu là huấn luyện một hàm $G_s \rightarrow t$ ánh xạ ảnh từ miền nguồn X_s tới miền đích X_t sử dụng các mẫu không được ghép cặp được lấy từ mỗi miền. Framework bao gồm hai generator $G_s \rightarrow t$ và $G_t \rightarrow s$ và hai discriminator D_s và D_t .

Generator

Coi $x \in X_s, X_t$ đại diện một mẫu từ miền nguồn và miền đích. Mô hình $G_s \rightarrow t$ bao gồm bộ mã hóa E_s , bộ giải mã G_t và bộ phân loại phụ trợ η_s , trong đó $\eta_s(x)$ đại diện cho khả năng x tới từ X_s . Coi $E_s^k(x)$ là bản đồ kích hoạt thứ k của bộ mã hóa và $E_s^{k_{ij}}(x)$ là giá trị tại điểm (i, j) . Bộ phân loại phụ trợ được huấn luyện để học trọng số của feature map thứ k cho miền nguồn, w_s^k , bằng việc sử dụng tổng hợp trung bình toàn cục (Global Average Pooling) và tổng hợp tối đa toàn cục (Global max pooling). Với việc tận dụng w_s^k , tập hợp bản đồ đặc trưng chú ý (attention feature map) có thể được tính toán: $a_s(x) = w_s * E_s(x) = \{w_s^k * E_s^k(x) \mid 1 \leq k \leq n\}$, trong



Hình 2.1: Kiến trúc của U-GAT-IT

đó n là số lượng các bản đồ đặc trưng được mã hóa. Sau đó mô hình $G_s \rightarrow t$ ngang bằng với $G_t(a_s(x))$. Sau đây là các khối với AdaLIN với các tham số γ và β được tính toán bởi một lớp được kết nối đầy đủ từ attention map:

$$\text{AdaLIN}(a, \gamma, \beta) = \gamma \cdot (\rho \cdot \hat{a}_I + (1 - \rho) \cdot \hat{a}_L) + \beta, \quad (2.1)$$

$$\hat{a}_I = \frac{a - \mu_I}{\sqrt{\sigma_I^2 + \epsilon}}, \quad \hat{a}_L = \frac{a - \mu_L}{\sqrt{\sigma_L^2 + \epsilon}}, \quad (2.2)$$

$$\rho \leftarrow \text{clip}_{[0,1]}(\rho - \tau \Delta \rho) \quad (2.3)$$

trong đó μ_I, μ_L và σ_I, σ_L lần lượt là trung bình theo từng kênh, từng lớp và độ lệch chuẩn, γ và β là các tham số được tạo ra bởi lớp được kết nối đầy đủ; τ là tốc độ học và $\Delta \rho$ biểu thị vector cập nhật tham số được xác định bởi trình tối ưu hóa (optimizer). Các giá trị của ρ giới hạn trong khoảng $[0, 1]$. Generator điều chỉnh

giá trị sao cho giá trị của ρ gần bằng 1 trong tác vụ mà việc chuẩn hóa theo từng mẫu là quan trọng và giá trị của ρ gần bằng 0 trong tác vụ mà LN là quan trọng. Giá trị của ρ được khởi tạo thành 1 trong các khối dư của bộ giải mã và 0 trong các khối lấy mẫu lên của bộ giải mã

Discriminator

Coi $x \in \{X_t, G_{s \rightarrow t}(X_s)\}$ đại diện cho một mẫu từ miền mục tiêu và miền nguồn. Giống như các mô hình chuyển đổi khác, discriminator D_t , một mô hình đa tỉ lệ (multi-scale model) bao gồm một bộ mã hóa E_D , một bộ phân loại C_D và một bộ phân loại phụ trợ η_{D_t} . Không giống như các mô hình chuyển đổi khác, $\eta_{D_t}(x)$ và $D_t(x)$ được huấn luyện để xác định x tới từ X_t hay $G_{s \rightarrow t}(X_s)$.

Cho mẫu x , $D_t(x)$ sử dụng bản đồ đặc trưng chú ý $a_{D_t}(x) = w_{D_t} * E_{D_t}(x)$ sử dụng w_{D_t} trên bản đồ đặc trưng được mã hóa E_{D_t} mà được huấn luyện bằng $\eta_{D_t}(x)$. Sau đó, discriminator $D_t(x)$ bằng với $C_{D_t}(a_{D_t}(x))$

Hàm mất mát

Để tối ưu hóa quá trình huấn luyện, các hàm mất mát sau được áp dụng:

- **Adversarial loss:** khớp phân phối của hình ảnh cần được thay đổi với phân phối hình ảnh mục tiêu:

$$L_{lsgan}^{s \rightarrow t} = (\mathbb{E}_{x \sim X_t}[(D_t(x))^2] + \mathbb{E}_{x \sim X_s}[(1 - D_t(G_{s \rightarrow t}(x)))^2]).$$

- **Cycle loss:** Để giảm bớt vấn đề mode collapse, ràng buộc nhất quán chu trình (cycle consistency constraint) cho bộ tạo (generator). Với một ảnh $x \in X_s$, sau khi thực hiện các phép biến đổi tuần tự từ X_s sang X_t và ngược lại, ảnh đó cần phải được chuyển đổi thành công về miền ban đầu:

$$L_{cycle}^{s \rightarrow t} = \mathbb{E}_{x \sim X_s}[|x - G_{t \rightarrow s}(G_{s \rightarrow t}(x))|_1].$$

- **Identity loss:** Để đảm bảo rằng phân phối màu sắc của ảnh đầu vào và ảnh đầu ra tương tự nhau, một ràng buộc nhất quán danh tính được áp dụng cho bộ tạo. Với ảnh $x \in X_t$, sau khi chuyển đổi x sử dụng $G_{s \rightarrow t}$, bức ảnh sẽ không thay đổi

$$L_{identity}^{s \rightarrow t} = \mathbb{E}_{x \sim X_s}[|x - G_{s \rightarrow t}(x)|_1].$$

- **CAM loss:** bằng việc tận dụng thông tin từ các bộ phân loại phụ trợ η_s và η_{D_t} , với ảnh $x \in \{X_s, X_t\}$, $G_{s \rightarrow t}$ và D_t biết được chỗ nào cần cải thiện hay cái gì

tạo ra nhiều khác biệt nhất giữa hai miền trong trạng thái hiện thời:

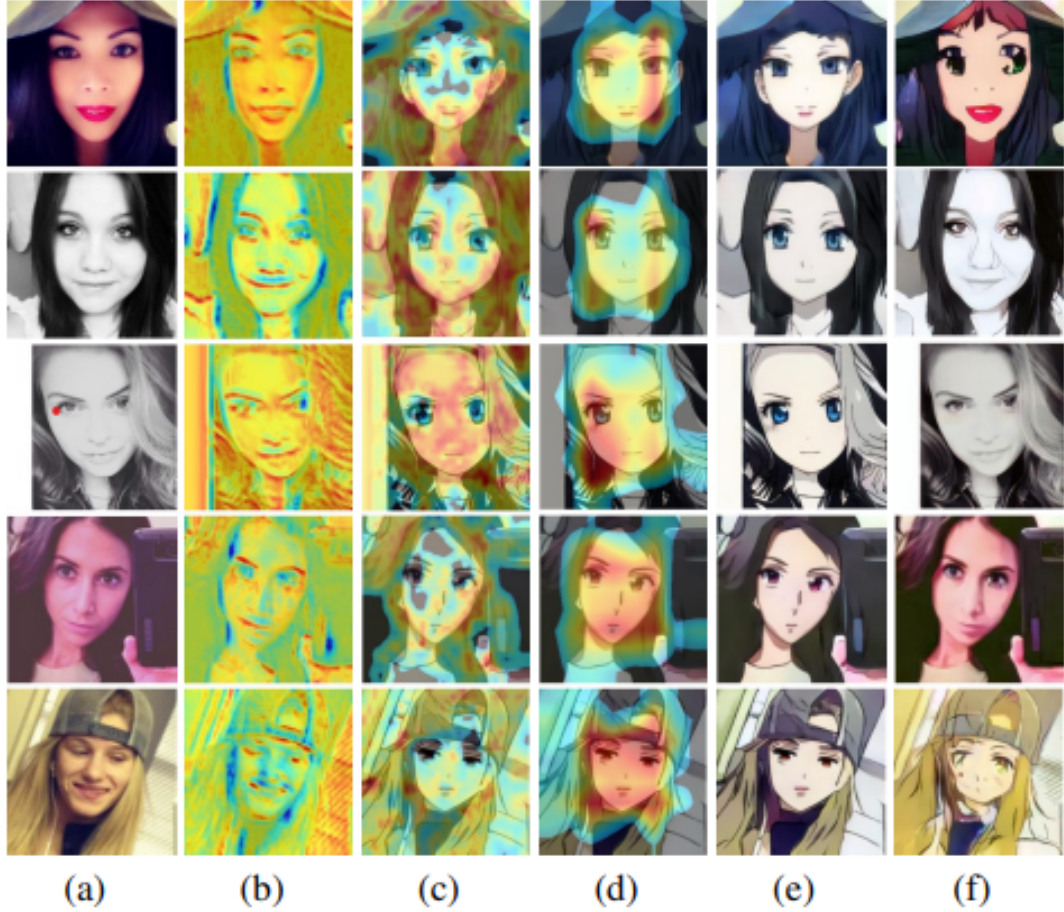
$$L_{cam}^{s \rightarrow t} = -(\mathbb{E}_{x \sim X_s}[\log(\eta_s(x))] + \mathbb{E}_{x \sim X_t}[\log(1 - \eta_s(x))]),$$

$$L_{cam}^{D_t} = \mathbb{E}_{x \sim X_t}[(\eta_{D_t}(x))^2] + \mathbb{E}_{x \sim X_s}[(1 - \eta_{D_t}(G_{s \rightarrow t}(x)))^2].$$

- **Full objective:** Cuối cùng huấn luyện các bộ mã hóa, giải mã, discriminators, và các bộ phân loại phụ trợ để tối ưu hóa mục tiêu cuối cùng:

$$\min_{G_{s \rightarrow t}, G_{t \rightarrow s}, \eta_s, \eta_t} \max_{D_s, D_t, \eta_{D_s}, \eta_{D_t}} \lambda_1 L_{lsgan} + \lambda_2 L_{cycle} + \lambda_3 L_{identity} + \lambda_4 L_{cam},$$

Trong đó $\lambda_1 = 1, \lambda_2 = 10, \lambda_3 = 10, \lambda_4 = 1000$. Tại đây. $L_{lsgan} = L_{lsgan}^{s \rightarrow t} + L_{lsgan}^{t \rightarrow s}$ và các mất mát khác được định nghĩa theo cách tương tự ($L_{cycle}, L_{identity}, L_{cam}$)



Hình 2.2: Trực quan hóa các bản đồ sự tập trung và ảnh hưởng của các bức ảnh được thể hiện trong các thí nghiệm bị loại bỏ

2.2 LiteRT

2.2.1 Tổng quan

LiteRT là môi trường thời gian chạy hiệu suất cao của Google dành cho các mô hình máy học trên thiết bị di động, được thiết kế đặc biệt để triển khai các giải pháp AI trong môi trường có tài nguyên hạn chế. Trước đây được biết đến với tên gọi TensorFlow Lite, LiteRT hiện đã được đổi tên nhằm nhấn mạnh vai trò của nó như một runtime nhẹ, tối ưu hóa cho việc suy luận trên các thiết bị di động và nhúng.

2.2.2 Tính năng chính

- **Tối ưu hoá cho công nghệ học máy trên thiết bị:** LiteRT giải quyết 5 quy tắc ràng buộc chính của ODML: độ trễ (không có lượt truy cập hai chiều đến máy chủ), quyền riêng tư (không có dữ liệu cá nhân nào rời khỏi thiết bị), khả năng kết nối (không bắt buộc phải có kết nối Internet), kích thước (giảm kích thước mô hình và tệp nhị phân) và mức tiêu thụ điện năng (suy luận hiệu quả và không có kết nối mạng)
- **Hỗ trợ nhiều nền tảng:** Tương thích với các thiết bị Android và iOS, Linux nhúng và bộ điều khiển vi mô.
- **Các tùy chọn mô hình nhiều khung:** AI Edge cung cấp các công cụ để chuyển đổi mô hình từ mô hình TensorFlow, PyTorch và JAX sang định dạng FlatBuffers (.tflite), cho phép bạn sử dụng nhiều mô hình hiện đại trên LiteRT. Bạn cũng có quyền sử dụng các công cụ tối ưu hoá mô hình có thể xử lý việc lượng tử hoá và siêu dữ liệu
- **Hỗ trợ nhiều ngôn ngữ:** Bao gồm SDK cho Java/Kotlin, Swift, Objective-C, C++ và Python.
- **Hiệu suất cao:** Tăng tốc phần cứng thông qua các trình uỷ quyền chuyên biệt như GPU và iOS Core ML.

2.2.3 Quy trình phát triển và tích hợp

Quy trình triển khai mô hình chuyển đổi ảnh sử dụng LiteRT gồm các bước chính sau:

- **Huấn luyện mô hình gốc:** Mô hình U-GAT-IT được huấn luyện trên tập dữ liệu chuyển đổi ảnh nhằm học được các đặc trưng cần thiết để chuyển đổi ảnh chân dung sang phong cách anime.
- **Chuyển đổi sang định dạng LiteRT:** Sau khi hoàn thành việc huấn luyện, mô hình được chuyển đổi sang định dạng tflite thông qua công cụ chuyển đổi của TensorFlow. Quá trình này giúp giảm kích thước mô hình và chuẩn bị cho việc

triển khai trên môi trường di động.

Nhờ vào LiteRT, ứng dụng có thể triển khai mô hình chuyển đổi ảnh hiệu quả, đảm bảo tốc độ xử lý nhanh chóng và tiêu thụ năng lượng thấp, từ đó nâng cao trải nghiệm người dùng khi áp dụng tính năng chuyển đổi ảnh trong ứng dụng ChatApp.

2.3 Các công nghệ hỗ trợ phát triển ChatApp

2.3.1 Android Studio với ngôn ngữ Java

Android Studio là Môi trường phát triển tích hợp (IDE) chính thức để phát triển ứng dụng Android. Nhờ có công cụ cho nhà phát triển và trình soạn thảo mã mạnh mẽ của IntelliJ IDEA, Android Studio cung cấp thêm nhiều tính năng giúp nâng cao năng suất khi xây dựng ứng dụng Android.

“Chat app” được xây dựng trên môi trường Android Studio với ngôn ngữ lập trình là Java. Nhờ những lợi thế của Android Studio mang lại giúp mô hình hóa, hiển thị giao diện, hỗ trợ máy ảo giúp cho công việc lập trình trở nên dễ dàng hơn. Nó còn hỗ trợ kết nối với Firebase.

2.3.2 Firebase

Firebase là một sản phẩm của Google nhằm giúp lập trình viên xây dựng, quản lý và phát triển ứng dụng một cách dễ dàng hơn và bảo mật hơn. Firebase cung cấp các dịch vụ về android, ios, web và unity và sử dụng NoSQL cho cơ sở dữ liệu để lưu trữ dữ liệu. Firebase hỗ trợ nhiều dịch vụ ở các mảng khác nhau như: Develop (Hosting, Test Lab, Cloud Messaging, Crash Reporting, Authentication), Grow (Invite, Ad - Word, Remote Config, Notifications, Dynamic Links). Những dịch vụ chính bao gồm dịch vụ back-end giúp lập trình viên như: Realtime Database, Cloud Firestore, Authentication, Remote Config Những dịch vụ mà sản phẩm Chat app sử dụng là:

Authentication

Hầu hết các ứng dụng hiện nay đều cần định danh người dùng, định danh người dùng cho phép ứng dụng bảo mật dữ liệu người dùng hơn. Và ứng dụng “Chat app” cũng không ngoại lệ.

Firebase Authentication cung cấp dịch vụ back-end, SDKs (Bộ phát triển phần mềm), và thư viện đồ họa để xác thực tài khoản người dùng trên ứng dụng. Nó hỗ trợ xác thực sử dụng mật khẩu, số điện thoại, gmail,... giống như phần lớn các ứng dụng hiện nay như Google, Facebook, Twitter,...

Việc sử dụng Authentication cho phép ứng dụng "Chat app" bảo mật tài khoản người dùng hơn. Nó còn hỗ trợ nhiều chức năng từ đăng nhập, đăng kí, cho đến lấy lại mật khẩu,... Qua đó người dùng có thể yên tâm sử dụng sản phẩm.

Firestore Database

Firestore sử dụng cơ sở dữ liệu đám mây NoSQL linh hoạt, có thể mở rộng, được xây dựng trên cơ sở hạ tầng Google Cloud, để lưu trữ và đồng bộ.

Ở “Chat app” Firestore được dùng để lưu trữ các đoạn hội thoại, và thông tin người dùng. Firestore sẽ giữ dữ liệu được đồng bộ trên ứng dụng khách thông qua trình nghe thời gian thực và cung cấp hỗ trợ ngoại tuyến cho thiết bị di động giúp “Chat app” phản hồi hoạt động bất kể độ trễ mạng hoặc kết nối Internet. Ngoài ra nó còn kết nối linh hoạt với các dịch vụ khác của Google.

2.3.3 Cloudinary

Cloudinary là một nền tảng đám mây chuyên về quản lý, xử lý và phân phối nội dung hình ảnh và video. Được thiết kế để phục vụ cả các nhà phát triển lẫn nhóm marketing, Cloudinary cung cấp một bộ công cụ toàn diện cho việc tải lên, lưu trữ, tối ưu hóa, chuyển đổi và phân phối nội dung media thông qua mạng phân phối nội dung (CDN).

Cloudinary được sử dụng rộng rãi bởi các công ty thương mại điện tử, nền tảng tin tức, mạng xã hội và các dịch vụ truyền thông để tăng hiệu suất hiển thị nội dung và giảm tải cho server backend.

Ở "Chat app", Cloudinary sẽ được sử dụng để lưu trữ ảnh trong hội thoại.

2.3.4 Thư viện và công nghệ khác

Gradle là một công cụ tự động hóa mà Android Studio sử dụng để tự động hóa và quản lý quá trình xây dựng. Gradle được biết đến với tính linh hoạt trong việc xây dựng phần mềm.

Thư mục Gradle trong dự án Android chứa các tệp cấu hình bản dựng cho dự án. Các tệp này chỉ định cấu hình cho dự án, bao gồm các phần phụ thuộc, công cụ và các cài đặt cần thiết khác và chạy ứng dụng. Với Gradle, có thể dễ dàng thêm các plugin của bên thứ 3 vào ứng dụng đang phát triển ở file "build.gradle".

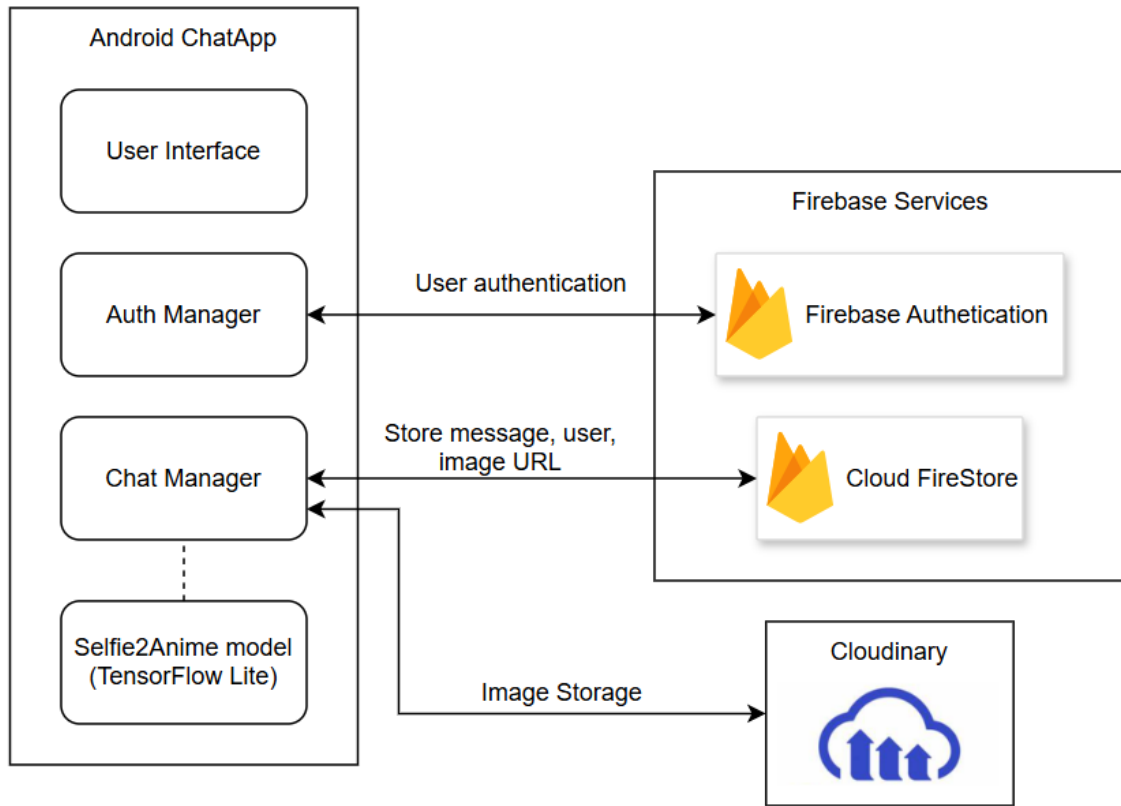
Đây là một số plugin, thư viện ngoài đã được sử dụng trong "Chat app":

- **Firestore:** Như đã trình bày, Firestore là một công cụ hỗ trợ mạnh mẽ cả về cơ sở dữ liệu cũng như có đầy đủ các dịch vụ tiện ích khác.
- **ZXing Android Embedded:** Thư viện quét mã cho Android, sử dụng ZXing để giải mã.
- **Retrofit:** Retrofit là ứng dụng khách REST dành cho Android và Java có thể được sử dụng cho các giao dịch mạng. Đó là một ứng dụng khách HTTP an toàn được phát triển bởi Square.

- Glide: Glide là mã nguồn mở hỗ trợ quản lý phương tiện (media management) nhanh chóng và hiệu quả dành cho Android, bao gồm giải mã, bộ nhớ đệm cũng như tổng hợp tài nguyên vào một giao diện đơn giản và dễ sử dụng.
- Picasso: Thư viện downloading và lưu trữ hình ảnh mạnh mẽ cho Android
- A scalable size unit: Thư viện hỗ trợ tỉ lệ khung hình cho các điện thoại khác nhau

CHƯƠNG 3. KIẾN TRÚC ỨNG DỤNG

3.1 Kiến trúc chung hệ thống



Hình 3.1: Kiến trúc chung hệ thống

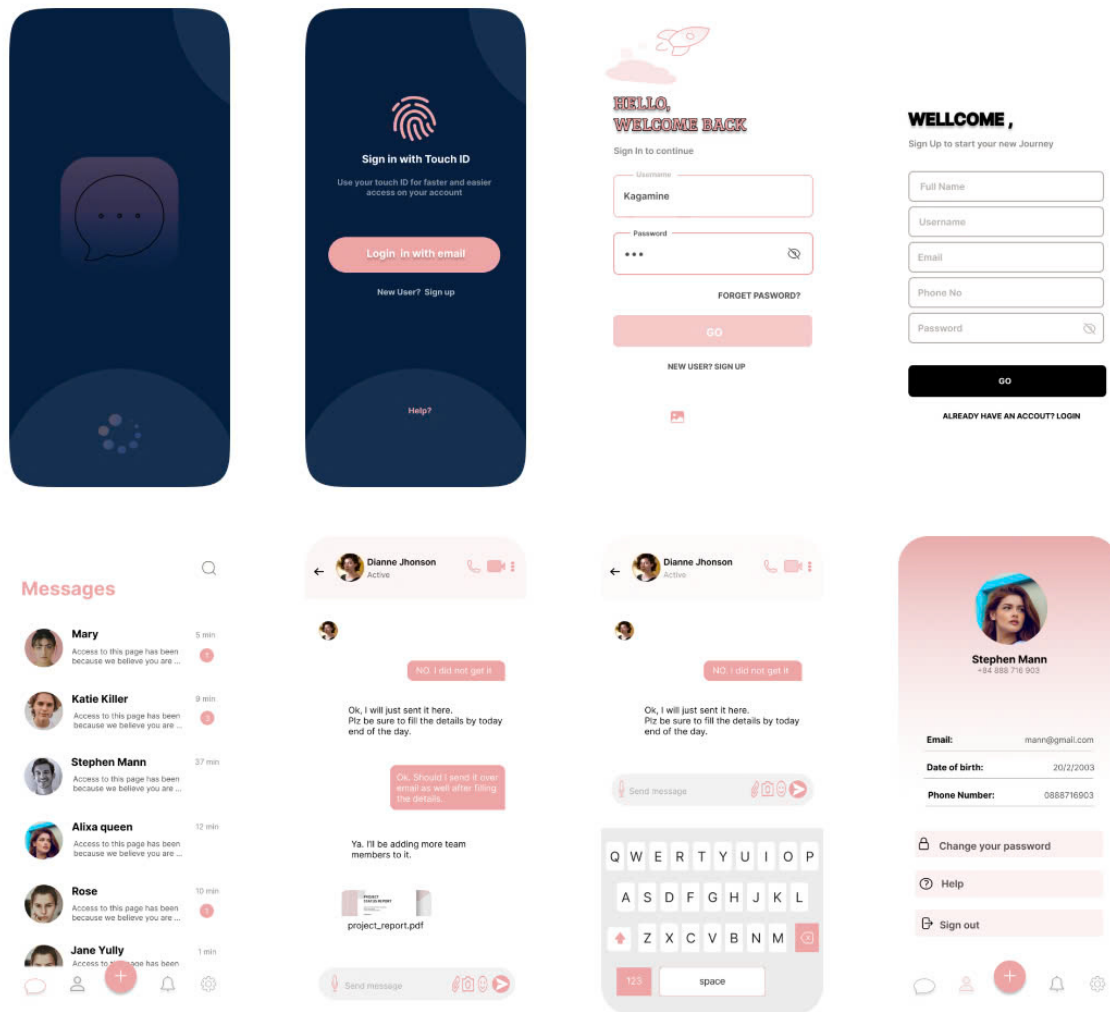
3.2 Giao diện người dùng (Frontend)

Ứng dụng Chat App gồm có các frame với các tính năng chính sau:

- Mở đầu (Intro: frame khi khởi động ứng dụng Chat app)
- Đăng kí, đăng nhập: frame cho phép người dùng đăng nhập hoặc tạo tài khoản mới
- Danh sách bạn bè: Hiển thị danh sách các người dùng đã nhắn tin, trạng thái on/off của người dùng và cho phép lựa chọn người dùng để nhắn tin
- Đoạn chat: frame cho phép tương tác giữa hai người dùng
- Thông tin cá nhân: Hiển thị thông tin cá nhân của người dùng

Bên cạnh đó, tính năng chuyển đổi ảnh được tích hợp liền mạch vào quy trình gửi và hiển thị hình ảnh của ứng dụng. Quy trình gửi như sau:

1. Chọn ảnh selfie: Người dùng chụp hoặc chọn ảnh từ thư viện.
2. Gửi yêu cầu chuyển đổi: Ảnh được gửi từ giao diện chat tới Module Selfie2Anime.



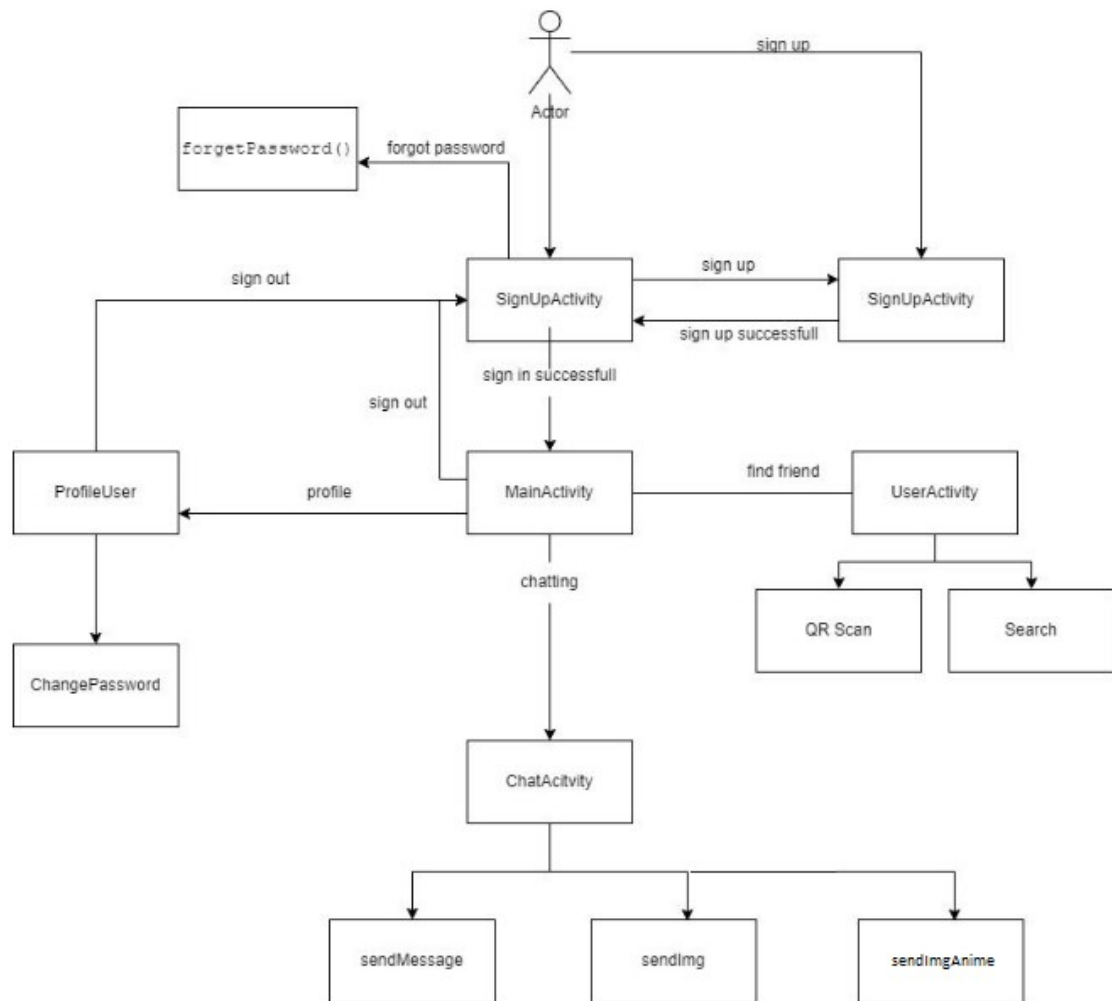
Hình 3.2: Giao diện ứng dụng

3. Xử lý chuyển đổi: Ảnh được chuyển qua LiteRT, nơi mô hình U-GAT-IT xử lý và chuyển đổi sang phong cách anime.
4. Hiển thị ảnh chuyển đổi: Ảnh chuyển đổi được trả về và hiển thị trong khung chat, giúp người dùng so sánh và chia sẻ ngay lập tức.

Trong ngữ cảnh phát triển ứng dụng Android, tất cả các tài nguyên liên quan đến giao diện của ứng dụng được tổ chức và lưu trữ trong thư mục "res". Các tài nguyên này sử dụng ngôn ngữ lập trình chính là XML (Extensible Markup Language) để mô tả dữ liệu cần hiển thị. XML là một ngôn ngữ đánh dấu linh hoạt và có khả năng mở rộng, được sử dụng để mô tả cấu trúc dữ liệu trong tài liệu dựa trên văn bản. Trong Android Studio, quá trình này được thực hiện thông qua class R, nơi tất cả các tài nguyên được ánh xạ.

Thư mục "res" của ứng dụng Chat app được chia thành các thư mục chính như layout, animation, drawable, và value. Trong đó:

- Thư mục layout: chứa layout của các activity



Hình 3.3: Sơ đồ hoạt động

- Thư mục drawable:
 - Chứa các tài nguyên hình ảnh và đồ họa để sử dụng trong quá trình xây dựng layout activity.
 - Cho phép tạo ra các hình ảnh, icon dưới dạng shape, vector, ... kèm theo cách hiệu ứng như bóng mờ, bo tròn, tạo viền,
 - Hỗ trợ chứa nhiều loại hình ảnh khác nhau, bao gồm PNG, JPEG, GIF, và cả các tệp vector drawable
- Thư mục animation: Chứa các tệp XML mô tả các hiệu ứng chuyển cảnh
- Thư mục value: Chứa các định nghĩa về màu, các từ viết tắt của chuỗi và xây dựng theme mà ứng dụng chat app sử dụng.

3.3 Phần lõi ứng dụng (Backend)

3.3.1 Đăng nhập và đăng ký tài khoản

Khởi động ứng dụng và kiểm tra phiên làm việc: Khi ứng dụng được khởi động, hệ thống sẽ kiểm tra trạng thái đăng nhập của người dùng thông qua một lớp quản lý phiên (session manager). Nếu người dùng chưa có phiên làm việc (chưa đăng nhập), ứng dụng sẽ chuyển hướng sang giao diện đăng nhập/đăng ký (SignInActivity/SignUpActivity).

Đăng ký và đăng nhập:

- Đăng ký (SignUpActivity): Người dùng nhập các thông tin cá nhân cần thiết như tên, email, mật khẩu, sau đó hệ thống sử dụng Firebase Authentication để tạo tài khoản mới với id tương ứng. Song song đó, các thông tin bổ sung (ví dụ: ảnh đại diện, số điện thoại) sẽ được lưu vào cơ sở dữ liệu Firestore để phục vụ việc hiển thị thông tin cá nhân sau này.
- Đăng nhập (SignInActivity): Khi người dùng đăng nhập, hệ thống xác thực thông tin bằng cách so sánh với dữ liệu đã lưu trên Firebase. Nếu thông tin hợp lệ, hệ thống khởi tạo phiên làm việc và tải dữ liệu người dùng cần thiết như danh sách người đã nhắn tin, lịch sử tin nhắn,...
- Quên mật khẩu: Hỗ trợ tính năng đặt lại mật khẩu thông qua ForgotPasswordDialog. Khi người dùng nhập địa chỉ email, hệ thống sẽ gửi một email khôi phục mật khẩu thông qua Firebase Authentication. Sau khi hoàn tất xác thực, ứng dụng chuyển hướng đến MainActivity, nơi người dùng có thể truy cập vào các chức năng chính.

3.3.2 Xử lý và hiển thị giao diện chính

- Khởi tạo giao diện chính: Sau khi xác thực, ứng dụng chuyển sang màn hình chính (Main Activity) nơi hiển thị danh sách các cuộc hội thoại, trạng thái hoạt động của người dùng và các thông báo liên quan.
- Kết nối và đồng bộ dữ liệu: Main Activity lắng nghe các sự thay đổi của dữ liệu từ Firestore thông qua các listener. Mỗi khi có cập nhật tin nhắn mới hoặc thay đổi trạng thái của bạn bè, giao diện sẽ được cập nhật tự động để hiển thị thông tin mới nhất.

3.3.3 Quản lý thông tin cá nhân

- Hiển thị và cập nhật thông tin người dùng: Sau khi đăng nhập, dữ liệu cá nhân của người dùng (họ tên, email, ảnh đại diện) được truy xuất từ Firestore và hiển thị trên giao diện cá nhân.

- Tạo và hiển thị mã QR: Ứng dụng cung cấp chức năng tạo mã QR để người dùng khác quét vào có thể tạo hội thoại với người dùng này.
- Thay đổi mật khẩu: Giao diện cho phép người dùng nhập mật khẩu cũ, mật khẩu mới và xác nhận mật khẩu. Hệ thống thực hiện kiểm tra hợp lệ (ví dụ: độ dài mật khẩu, khớp mật khẩu) trước khi cập nhật thông tin mới trên Firebase.
- Đăng xuất: Khi người dùng yêu cầu đăng xuất, phiên làm việc được kết thúc và người dùng được chuyển về màn hình đăng nhập.

3.3.4 Quản lý danh sách người dùng và tìm kiếm

- Hiển thị danh sách người dùng: Ứng dụng truy vấn dữ liệu từ Firestore để lấy danh sách bạn bè. Danh sách này được sắp xếp và cập nhật theo thời gian thực, hiển thị trạng thái online/offline của người dùng.
- Tìm kiếm người dùng: Giao diện tìm kiếm cho phép người dùng nhập tên để lọc và hiển thị danh sách người dùng phù hợp. Quá trình này được thực hiện bằng cách sử dụng các listener trên Firestore nhằm đảm bảo kết quả luôn được cập nhật theo thời gian thực.
- Kết nối qua mã QR: Ngoài tìm kiếm bằng văn bản, người dùng có thể sử dụng chức năng quét mã QR để kết nối nhanh, từ đó mở màn hình trò chuyện tương ứng.

3.3.5 Xử lý hội thoại

- Gửi và nhận tin nhắn: Khi người dùng soạn tin nhắn (văn bản hoặc hình ảnh), tin nhắn đó sẽ được đẩy lên Firestore theo định dạng của một cuộc hội thoại. Hệ thống phân biệt các cuộc trò chuyện dựa trên ID người gửi và người nhận.
- Cập nhật hội thoại thời gian thực: Các listener được đăng ký để lắng nghe sự thay đổi của bảng hội thoại, qua đó tự động cập nhật giao diện chat ngay khi có tin nhắn mới. Điều này giúp đảm bảo tính liên tục và mượt mà của cuộc trò chuyện.
- Gửi ảnh: Nếu tin nhắn chứa ảnh, hệ thống sẽ tải lên Cloudinary và lưu lại URL của ảnh trong cuộc hội thoại. Khi nhận tin nhắn, ứng dụng sẽ lấy URL này để tải và hiển thị ảnh tương ứng.

3.3.6 Tích hợp tính năng chuyển đổi ảnh Selfie2Anime

Quy trình chuyển đổi ảnh:

- Tải lên ảnh gốc: Khi người dùng muốn chuyển đổi ảnh, họ có thể chọn hoặc chụp một ảnh selfie từ giao diện chat. Ảnh gốc sẽ được gửi tới module chuyển đổi ảnh.

- Xử lý qua môi trường LiteRT: Ảnh được đưa vào quy trình xử lý qua LiteRT, nơi mô hình U-GAT-IT (được tối ưu hóa và chuyển đổi sang định dạng nhẹ tflite) thực hiện suy luận để chuyển đổi ảnh sang phong cách anime.
- Trả kết quả chuyển đổi: Ảnh đã chuyển đổi được trả về cho module chat, hiển thị trực tiếp trong giao diện tin nhắn.
- Lưu trữ ảnh chuyển đổi: Ảnh chuyển đổi cũng có thể được lưu trữ trên Firebase Storage để phục vụ việc truy xuất và chia sẻ sau này.

CHƯƠNG 4. CÀI ĐẶT ỨNG DỤNG

4.1 Cài đặt mô hình Selfie2Anime

Tạo SavedModel từ checkpoint

Phần này sẽ sử dụng file Selfie2Anime_Model.ipynb

Mục tiêu: Chuyển đổi các file checkpoint đã được huấn luyện sẵn của mô hình U-GAT-IT (phiên bản nhẹ từ Kaggle) sang định dạng TensorFlow SavedModel.

Hướng dẫn chạy:

1. Mở tệp Selfie2Anime_Model.ipynb.
2. Thiết lập môi trường: Đảm bảo bạn đang sử dụng môi trường Python 3.6.* và đã cài đặt TensorFlow phiên bản 1.14 (`%pip install tensorflow==1.14`). Cài đặt thư viện kaggle (`%pip install kaggle`).
3. Cấu hình Kaggle API: Đặt tệp kaggle.json (tải từ tài khoản Kaggle của bạn) vào thư mục gốc của dự án (Selfie2Anime-with-TFLite). Chạy cell để đọc thông tin xác thực (`with open('./kaggle.json') as f:...`).
4. Tải và giải nén Checkpoint: Chạy cell để tải dataset từ Kaggle (`!kaggle datasets download -d t04glovern/ugatit-selfie2anime-pretrained`). Sau đó, giải nén tệp `ugatit-selfie2anime-pretrained.zip` và đảm bảo thư mục chứa checkpoint có tên là `pretrained_model` nằm trong thư mục gốc của dự án. Cấu trúc sẽ là `Selfie2Anime-with-TFLite/pretrained_model/checkpoint/...`
5. Chạy các cell còn lại: Thực thi tuần tự các cell để khởi tạo tham số, lớp UGATIT, tải checkpoint đã giải nén, và cuối cùng là xuất mô hình ra định dạng SavedModel (`saved_model_dir = './saved_model' ... tf.saved_model.simple_save(...)`).

Kết quả đầu ra mong đợi:

- Một thư mục mới có tên `saved_model` sẽ được tạo trong thư mục Selfie2Anime-with-TFLite.
- Bên trong `saved_model`, sẽ có tệp `saved_model.pb` và các thư mục con chứa biến của mô hình. Đây chính là mô hình ở định dạng SavedModel.

Chuyển đổi SavedModel sang TFLite và kiểm thử

Phần này sẽ sử dụng TensorFlowLite_Conversion.ipynb

Hướng dẫn chạy:

1. Mở tệp TensorFlowLite_Conversion.ipynb.

2. Thiết lập môi trường: Đảm bảo bạn đang sử dụng môi trường có TensorFlow phiên bản 2.x (ví dụ: 2.2.0 như trong notebook).
3. Chuyển đổi sang TFLite: Chạy các cell định nghĩa hàm `convert_to_tflite` và cell gọi hàm này
4. Kiểm thử suy luận: Đảm bảo tệp ảnh `test300x300.png` có sẵn trong thư mục `Selfie2Anime-with-TFLite`. Chạy các cell để tải ảnh, tiền xử lý ảnh (thay đổi kích thước), và thực hiện suy luận bằng mô hình TFLite vừa tạo.

Kết quả đầu ra mong đợi:

- Một tệp mới có tên `selfie2anime.tflite` sẽ được tạo trong thư mục `Selfie2Anime-with-TFLite`. Đây là mô hình TensorFlow Lite chưa có metadata.
- Khuyến nghị: Sau khi tạo, bạn nên di chuyển tệp `selfie2anime.tflite` này vào thư mục `model_without_metadata` để chuẩn bị cho bước tiếp theo.
- Notebook sẽ hiển thị ảnh gốc (`test300x300.png`) và ảnh kết quả sau khi được xử lý bởi mô hình TFLite (ảnh theo phong cách anime).

Thêm Metadata vào mô hình TFLite

Phần này sẽ sử dụng file `Add_Metadata.ipynb`

Mục tiêu: Gắn thêm thông tin metadata (như mô tả mô hình, thông tin đầu vào/đầu ra, cách chuẩn hóa dữ liệu) vào tệp `.tflite` đã tạo ở Bước 2. Điều này giúp việc tích hợp mô hình vào ứng dụng di động trở nên dễ dàng hơn.

Hướng dẫn chạy:

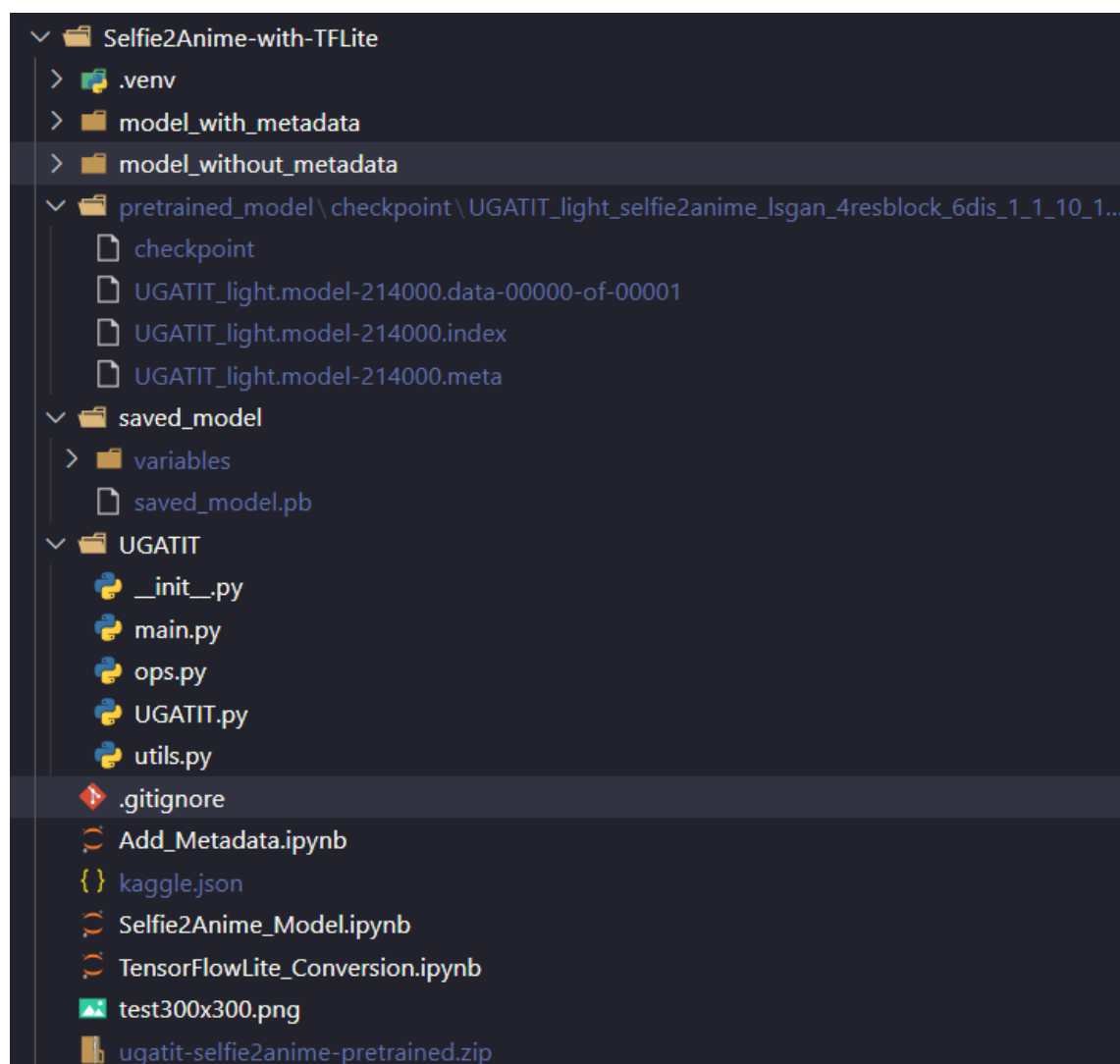
1. Mở tệp `Add_Metadata.ipynb`.
2. Chuẩn bị tệp và thư mục:
Tạo hai thư mục `model_without_metadata` và `model_with_metadata` trong `Selfie2Anime-with-TFLite` nếu chúng chưa tồn tại.
3. Di chuyển tệp `selfie2anime.tflite` (đã tạo ở Bước 2) vào thư mục `model_without_metadata`.
4. Cài đặt thư viện: Chạy cell để cài đặt `tflite-support`.
5. Chạy các cell còn lại: Thực thi tuần tự các cell để định nghĩa đường dẫn, lớp `MetadataPopulatorForGANModel`, hàm `populate_metadata`, và cuối cùng là cell gọi hàm `populate_metadata(MODEL_FILE)`.

Kết quả đầu ra mong đợi:

- Trong thư mục `model_with_metadata`: Một tệp `selfie2anime.tflite` mới. Đây là mô hình TFLite đã được gắn metadata.

- Một tệp selfie2anime.json chứa thông tin metadata dưới dạng JSON.

Sau khi hoàn thành cả ba bước, bạn sẽ có tệp selfie2anime.tflite trong thư mục model_with_metadata/, sẵn sàng để tích hợp vào ứng dụng android.



Hình 4.1: Cài đặt mô hình

4.2 Ứng dụng ChatApp

4.2.1 Hướng dẫn cài đặt

Chuẩn bị môi trường

- Android Studio (khuyến nghị bản mới nhất)
- JDK 8 trở lên
- Kết nối internet để tải dependencies

Cài đặt các file cấu hình

Thêm file google-services.json

1. Truy cập <https://console.firebase.google.com/>, tạo project mới hoặc chọn project

có sẵn.

2. Thêm Android app với package name giống trong file AndroidManifest.xml.
3. Tải file google-services.json về và đặt vào thư mục ChatAppFirebase/app/.

Thêm file clouddinary.properties

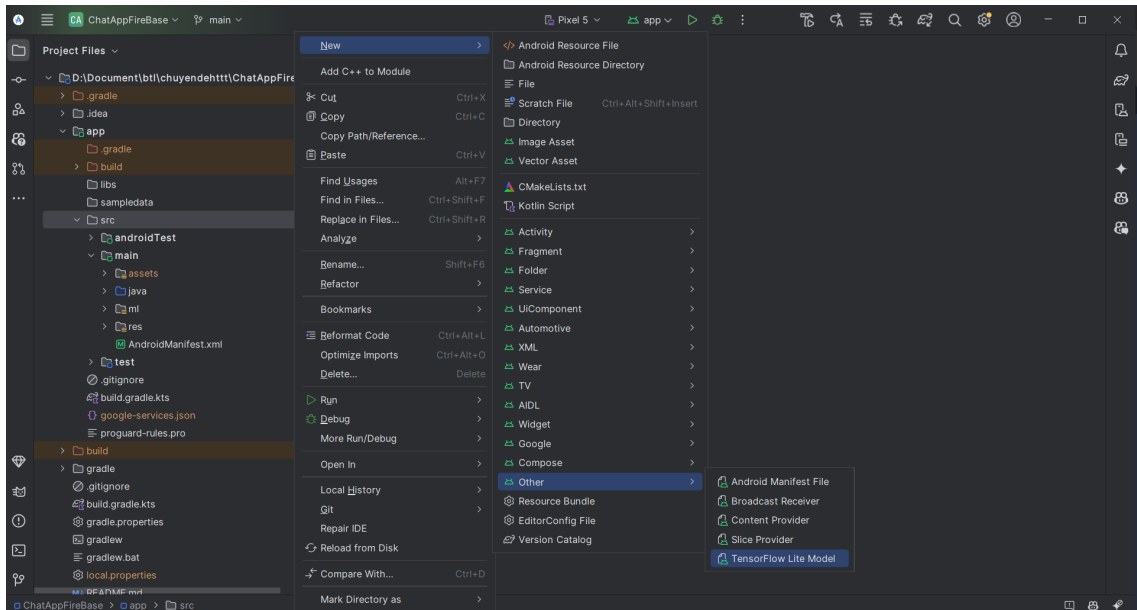
1. Đăng ký tài khoản tại <https://cloudinary.com/>.
2. Vào Setting/Upload và tạo 1 Upload Presets mới với name tùy chọn và mode Unsigned.
3. Lấy thông tin cloud_name và upload preset name từ dashboard.
4. Tạo file clouddinary.properties trong ChatAppFirebase/app/src/main/assets với nội dung:

```
cloud_name=YOUR_CLOUD_NAME
```

```
upload_preset=YOUR_UPLOAD_PRESET
```

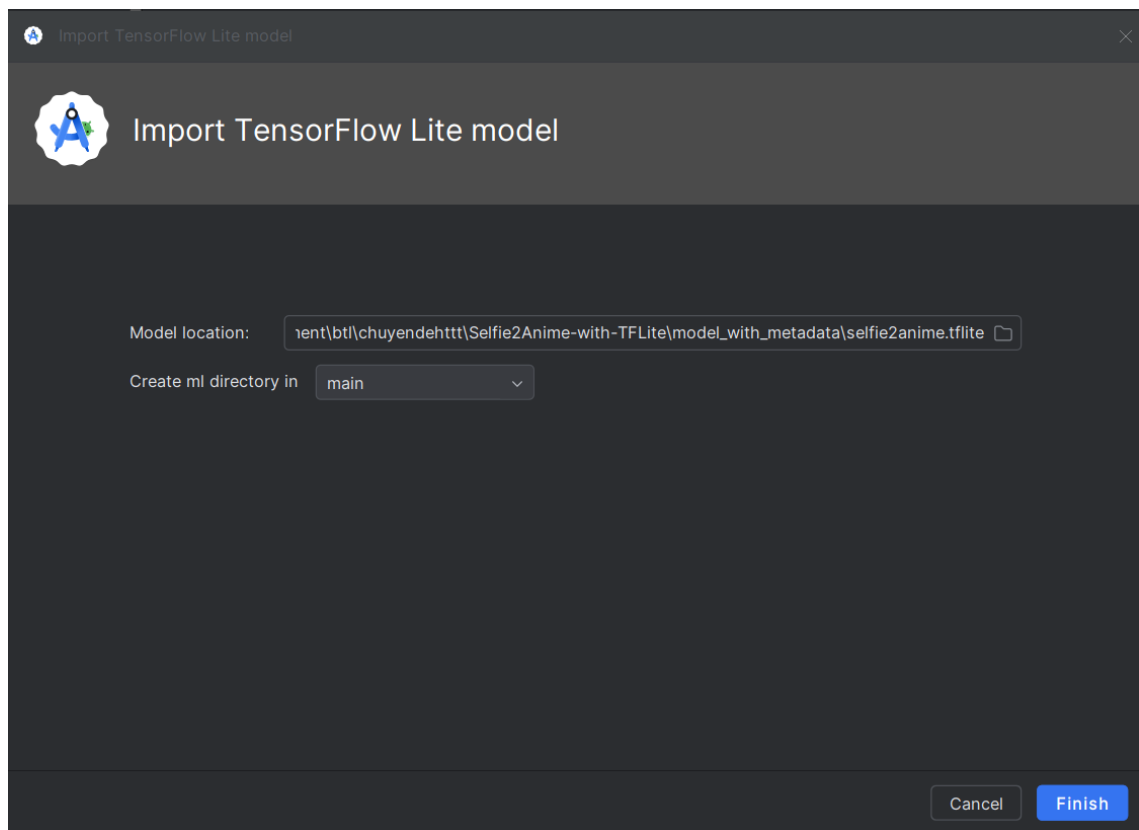
```
secure=true
```

Thêm model ML vào ứng dụng Trong Android Studio, có thể nhấn chọn New file trong folder src để thêm TensorFlow Lite Model:



Hình 4.2: Cài đặt ứng dụng android

Chọn model trong folder model_with_metadata:

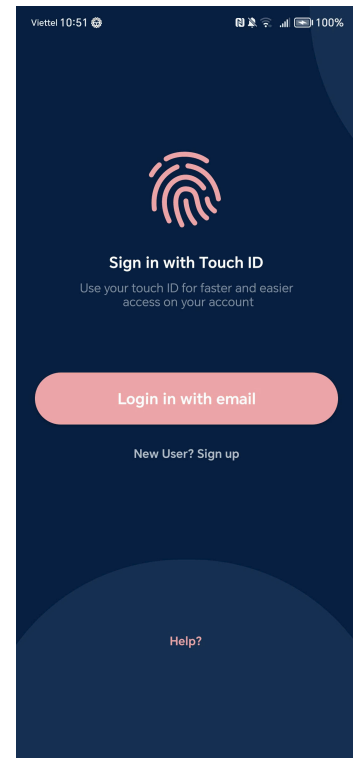


Hình 4.3: Cài đặt ứng dụng android

Hoàn thiện các bước trên thì tiến hành build và chạy ứng dụng, kết nối thiết bị hoặc dùng trình giả lập.

CHƯƠNG 5. DEMO

Màn hình cho phép người dùng chọn đăng nhập hoặc đăng ký nếu chưa có tài khoản.



Viettel 10:55

WELCOME,

Sign Up to start your new Journey



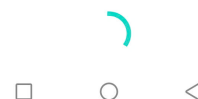
Name
Nguyễn Thắng

Email
thang@gmail.com

Password
.....

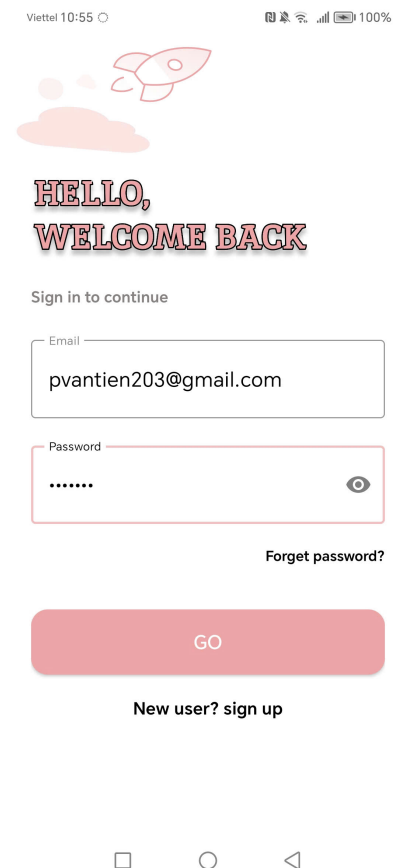
Confirm
.....

Already have an account? Login



Màn hình đăng ký:

Màn hình đăng nhập:



Viettel 10:55

HELLO,
WELCOME BACK

Sign in to continue

Email
pvantien203@gmail.com

Password

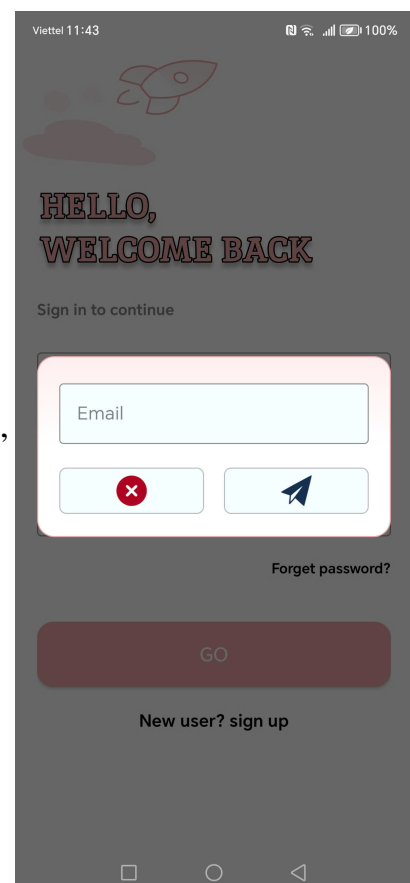
Forget password?

GO

New user? sign up

□ ○ ◀

Nếu quên mật khẩu, ấn quên trên màn hình đăng nhập, nhập email đã đăng ký:



Viettel 11:43

HELLO,
WELCOME BACK

Sign in to continue

Email

✕ ↗

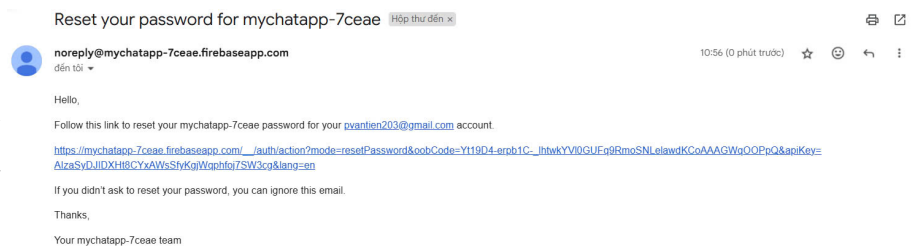
Forget password?

GO

New user? sign up

□ ○ ◀

Hệ thống sẽ gửi email về để tiến hành tạo lại mật khẩu:



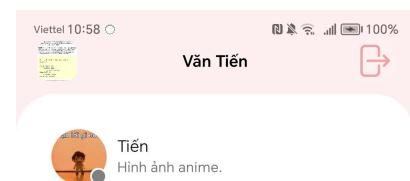
Màn hình đổi mật khẩu:

A form titled 'Reset your password for pvantien203@gmail.com'. It contains a text input field labeled 'New password' with an eye icon for toggling visibility. A blue 'SAVE' button is at the bottom right.

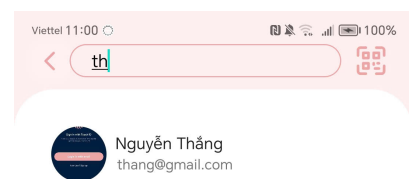
Màn hình đổi mật khẩu thành công:

A message box titled 'Password changed' with the text 'You can now sign in with your new password'.

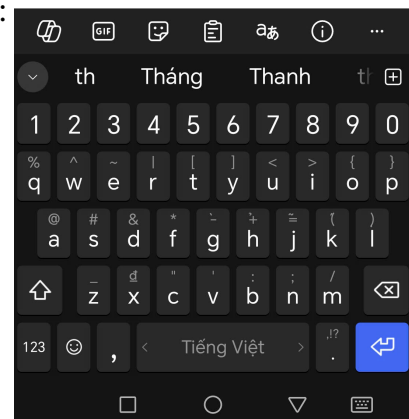
Màn hình trang chủ:



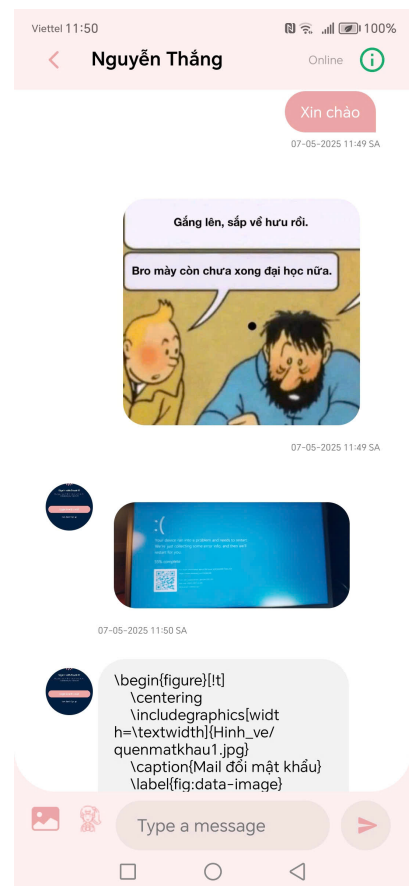
Màn hình thông tin cá nhân, bao gồm mã QR cá nhân (người dùng khác có thể quét QR để tạo hội thoại), đổi mật khẩu và đăng xuất:



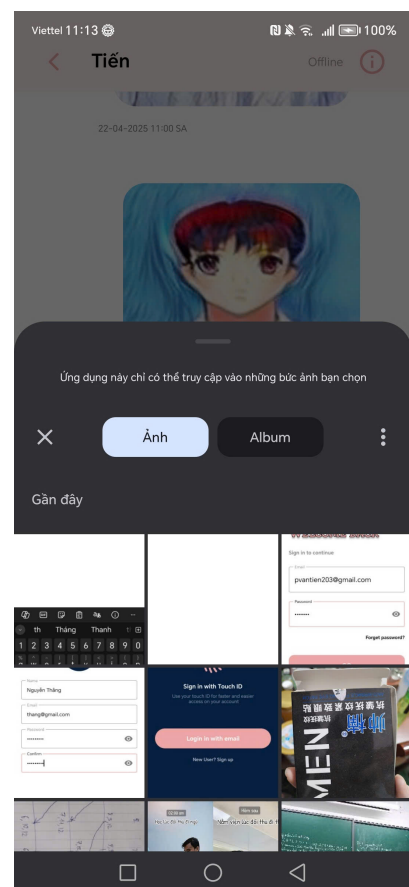
Màn hình tìm kiếm người dùng khác để tạo hội thoại, có thể tìm kiếm bằng văn bản hoặc bằng quét mã QR:



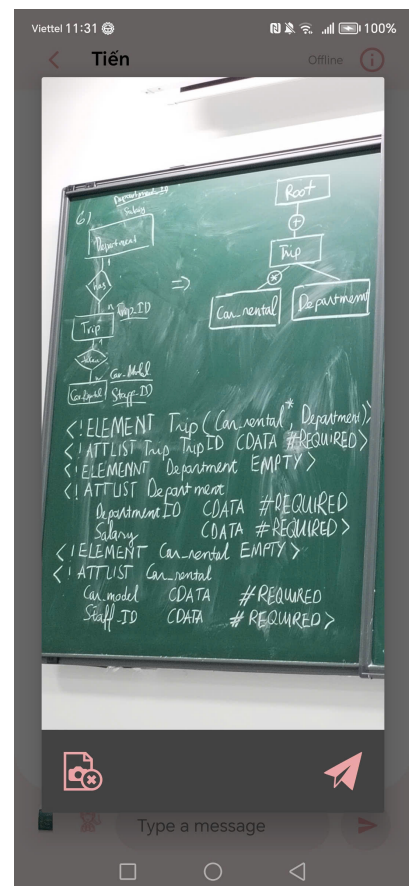
Màn hình hội thoại:



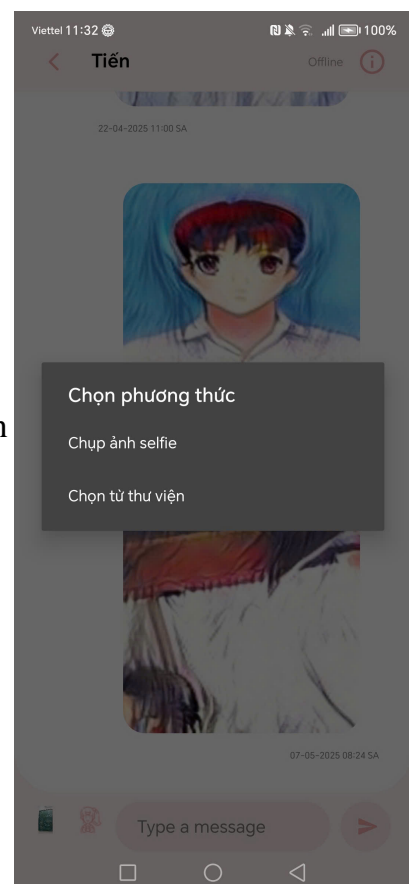
Màn hình chọn và gửi ảnh trong thư viện:



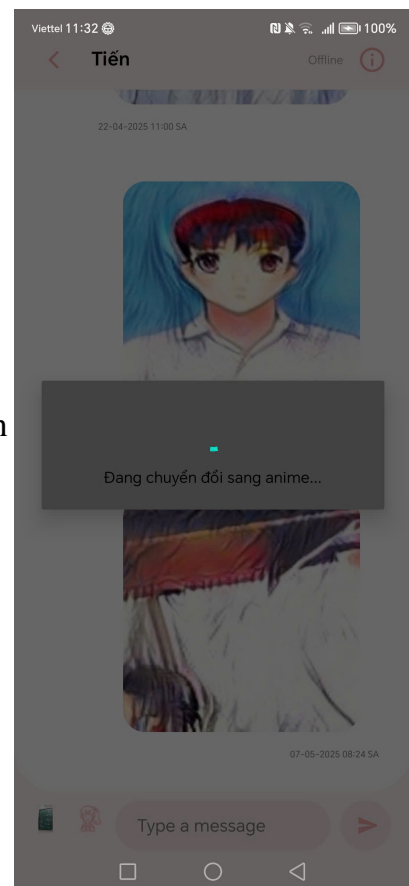
Ảnh sau khi chọn:



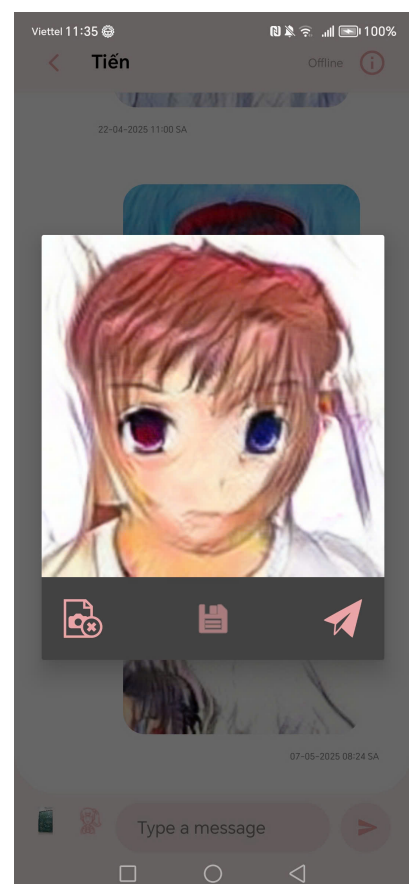
Màn hình cho chuyển đổi ảnh anime, đầu tiên là chọn phương thức:



Nếu chọn ảnh thư viện thì sau khi chọn xong sẽ tiến hành chuyển đổi sang anime và hiển thị ra kết quả:



Kết quả ảnh anime:



Nếu chọn chụp ảnh selfie, chụp xong thì ấn chuyển đổi rồi hiển thị ra kết quả tương tự như trên.

TÀI LIỆU THAM KHẢO

M. Maynard-Reid, "Selfie2Anime-with-TFLite," GitHub. [Online]. Available: <https://github.com/margaretmz/Selfie2Anime-with-TFLite/>. [Accessed: March 7, 2025].

J. Kim, "UGATIT," GitHub. [Online]. Available: <https://github.com/taki0112/UGATIT>. [Accessed: March 7, 2025].

N. Glover, "UGATIT," GitHub. [Online]. Available: <https://github.com/t04glovern/UGATIT>. [Accessed: March 7, 2025].

J. Kim, M. Kim, H. Kang, and K. Lee, "U-GAT-IT: Unsupervised Generative Attentional Networks with Adaptive Layer-Instance Normalization for Image-to-Image Translation," arXiv preprint arXiv:1907.10830, 2019. [Online]. Available: <https://arxiv.org/abs/1907.10830>

Source code: <https://github.com/phamT1042/ChatApp-Selfie2Anime>