

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



**BÁO CÁO GIỮA KỲ
PHÁT TRIỂN ỨNG DỤNG DI ĐỘNG**

**TÌM HIỂU FIREBASE
FIRESTORE VÀ ỨNG DỤNG
QUẢN LÝ THÔNG TIN HỌC SINH**

Người hướng dẫn: **GV VÕ VĂN THÀNH**

Người thực hiện: **TRẦN TẤN THÀNH – 52100110**

Khoá: 25

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO GIỮA KỲ
PHÁT TRIỂN ỨNG DỤNG DI ĐỘNG

TÌM HIỂU FIREBASE
FIRESTORE VÀ ỨNG DỤNG
QUẢN LÝ THÔNG TIN HỌC SINH

Người hướng dẫn: **GV VÕ VĂN THÀNH**

Người thực hiện: **TRẦN TẤN THÀNH – 52100110**

Khoá: 25

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Em chân thành cảm ơn Thầy Võ Văn Thành, người thầy tận tâm và kiến thức sâu rộng trong môn học Phát Triển Ứng Dụng Di Động.

Sự nhiệt huyết và tận tụy của Thầy không chỉ truyền đạt kiến thức mà còn truyền cả niềm đam mê với lĩnh vực này. Những hướng dẫn chi tiết, sự kiên nhẫn và sự cống hiến không ngừng nghỉ của Thầy đã đồng hành cùng em qua từng bước tiến trên con đường học tập.

Nhận được sự hỗ trợ và định hướng từ Thầy, em đã có cơ hội học hỏi và trải nghiệm những kiến thức thực tiễn, đồng thời phát triển khả năng sáng tạo và xây dựng ứng dụng di động một cách thành thạo. Em xin gửi lời cảm ơn sâu sắc đến Thầy Võ Văn Thành vì sự dẫn dắt và tạo điều kiện cho chúng em có một hành trình học tập ý nghĩa và đầy ý tưởng mới.

TP. Hồ Chí Minh, ngày 30 tháng 11 năm 2023

Tác giả

(Ký tên và ghi rõ họ tên)

Thành

Trần Tấn Thành

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Chúng tôi xin cam đoan đây là công trình nghiên cứu của riêng chúng tôi và được sự hướng dẫn khoa học của GV. Võ Văn Thành. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Dự án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung Dự án của mình. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 30 tháng 11 năm 2023

Tác giả

(Ký tên và ghi rõ họ tên)

Thành

Trần Tấn Thành

MỤC LỤC

DANH MỤC HÌNH VẼ	vi
CHƯƠNG 1. GIỚI THIỆU VỀ FIREBASE FIRESTORE	1
1.1 Firebase	1
1.2 Realtime Database.....	1
1.3 Cloud Firestore.....	2
1.4 Authentication	2
1.5 Cloud Storage.....	3
CHƯƠNG 2. QUẢN LÝ THÔNG TIN HỌC SINH	4
2.1 Đặc tả yêu cầu	4
2.2 Sơ đồ thực thể - quan hệ (ERD) mức vật lý.....	4
2.3 Hiện thực ứng dụng.....	6
2.3.1 Chức năng đăng nhập.....	6
2.3.2 Chức năng đổi ảnh đại diện và cập nhật thông tin người dùng	7
2.3.3 Chức năng xem danh sách người dùng trên hệ thống	9
2.3.4 Chức năng thêm một người dùng mới.....	11
2.3.5 Chức năng xóa người dùng.....	12
2.3.6 Chức năng xem lịch sử đăng nhập của người dùng	12
2.3.7 Chức năng xem danh sách học sinh.....	13
2.3.8 Chức năng thêm một học sinh mới.....	14
2.3.9 Chức năng xóa một học sinh.....	14
2.3.10 Chức năng cập nhật thông tin học sinh	15
2.3.11 Chức năng sắp xếp danh sách học sinh.....	16

2.3.12 Chức năng tìm kiếm học sinh.....	18
2.3.13 Chức năng xem danh sách chứng chỉ	19
2.3.14 Chức năng thêm chứng chỉ mới cho học sinh.....	19
2.3.15 Chức năng xóa chứng chỉ	20
2.3.16 Chức năng thêm danh sách học sinh từ File.....	20
2.3.17 Chức năng xuất danh sách học sinh ra File	23
2.3.18 Chức năng thêm danh sách chứng chỉ từ File	24
2.3.19 Chức năng xuất danh sách chứng chỉ ra File.....	25
CHƯƠNG 3. DEMO	26
3.1 Chức năng đăng nhập.....	26
3.2 Chức năng đổi ảnh đại diện và cập nhật thông tin người dùng.....	27
3.3 Chức năng xem danh sách người dùng trên hệ thống	28
3.4 Chức năng thêm một người dùng mới	29
3.5 Chức năng xóa người dùng	30
3.6 Chức năng xem lịch sử đăng nhập của người dùng	31
3.7 Demo chức năng xem danh sách học sinh	32
3.8 Demo chức năng thêm một học sinh mới	33
3.9 Demo chức năng xóa một học sinh.....	34
3.10 Demo chức năng cập nhật thông tin học sinh	35
3.11 Demo chức năng sắp xếp danh sách học sinh.....	36
3.12 Demo chức năng tìm kiếm học sinh.....	37
3.13 Demo chức năng xem danh sách chứng chỉ.....	38
3.14 Demo chức năng thêm chứng chỉ.....	39

3.15 Demo chức năng xóa chứng chỉ.....	39
--	----

TÀI LIỆU THAM KHẢO	41
---------------------------------	-----------

DANH MỤC HÌNH VẼ

Hình 1 Sơ đồ thực thể - quan hệ (ERD) mức vật lý.....	5
Hình 2 Đoạn code thực hiện chức năng đăng nhập (1).....	7
Hình 3 Đoạn code thực hiện chức năng đăng nhập (2).....	7
Hình 4 Đoạn code cập nhật thông tin người dùng (1).....	8
Hình 5 Đoạn code cập nhật thông tin người dùng (2).....	9
Hình 6 Đoạn code lấy danh sách người dùng trong hệ thống.....	10
Hình 7 Đoạn code bắt sự kiện thay đổi trên collection.....	10
Hình 8 Đoạn code thêm người dùng mới (1).....	11
Hình 9 Đoạn code thêm người dùng mới (2).....	11
Hình 10 Đoạn code xóa người dùng.....	12
Hình 11 Đoạn code xem lịch sử đăng nhập (1).....	13
Hình 12 Đoạn code xem lịch sử đăng nhập (2).....	13
Hình 13 Đoạn code lấy danh sách học sinh.....	14
Hình 14 Đoạn code thêm học sinh mới.....	14
Hình 15 Đoạn code xóa một học sinh.....	15
Hình 16 Đoạn code cập nhật thông tin học sinh.....	15
Hình 17 Đoạn code hỗ trợ chức năng sắp xếp.....	16
Hình 18 Đoạn code sắp xếp theo tên.....	17
Hình 19 Đoạn code sắp xếp theo ngày sinh.....	17
Hình 20 Đoạn code sắp xếp theo lớp.....	18
Hình 21 Đoạn code hỗ trợ tìm kiếm học sinh.....	18
Hình 22 Đoạn code tìm kiếm học sinh.....	19

Hình 23 Đoạn code lấy danh sách chứng chỉ	19
Hình 24 Đoạn code thêm chứng chỉ mới cho học sinh	20
Hình 25 Đoạn code xóa chứng chỉ	20
Hình 26 Đoạn code cấp quyền truy cập bộ nhớ	20
Hình 27 Đoạn code gọi Intent để chọn File (1).....	21
Hình 28 Đoạn code gọi Intent để chọn File (2).....	21
Hình 29 Đoạn code thêm học sinh từ File (1)	22
Hình 30 Đoạn code thêm học sinh từ File (2)	22
Hình 31 Đoạn code xuất danh sách học sinh ra File (1)	23
Hình 32 Đoạn code xuất danh sách học sinh ra File (2)	23
Hình 33 Đoạn code thêm danh sách chứng chỉ từ File (1).....	24
Hình 34 Đoạn code thêm danh sách chứng chỉ từ File (2).....	25
Hình 35 Đoạn code xử lý xuất danh sách chứng chỉ ra File (1)	26
Hình 36 Đoạn code xử lý xuất danh sách chứng chỉ ra File (2)	26
Hình 37 Demo chức năng đăng nhập	27
Hình 38 Demo chức năng đổi ảnh đại diện và cập nhật thông tin	28
Hình 39 Demo chức năng xem danh sách người dùng	28
Hình 40 Demo chức năng thêm người dùng mới.....	29
Hình 41 Demo chức năng xóa người dùng	30
Hình 42 Demo chức năng xem lịch sử đăng nhập	31
Hình 43 Demo chức năng xem danh sách học sinh	32
Hình 44 Demo chức năng thêm một học sinh mới	33
Hình 45 Demo chức năng xóa học sinh	34

Hình 46 Demo chức năng cập nhật thông tin học sinh	35
Hình 47 Demo chức năng sắp xếp học sinh.....	36
Hình 48 Demo chức năng tìm kiếm học sinh.....	37
Hình 49 Demo chức năng xem danh sách chứng chỉ.....	38
Hình 50 Demo chức năng thêm chứng chỉ.....	39
Hình 51 Demo chức năng xóa chứng chỉ.....	40

CHƯƠNG 1. GIỚI THIỆU VỀ FIREBASE FIRESTORE

1.1 Firebase

Firebase là một nền tảng phát triển ứng dụng di động và web của Google, cung cấp các dịch vụ và công cụ giúp nhà phát triển xây dựng ứng dụng một cách nhanh chóng và hiệu quả. Nền tảng này được thiết kế để giúp giảm thiểu khả năng phức tạp trong việc quản lý cơ sở dữ liệu, xác thực người dùng, phân tích dữ liệu, và triển khai ứng dụng.

Firebase cung cấp các API thuận tiện, mạnh mẽ và tương thích trên nhiều nền tảng, giúp quản lý và sử dụng cơ sở dữ liệu trở nên đơn giản. Với Firebase, việc tương tác với cơ sở dữ liệu không còn phức tạp như trước; chúng ta có thể gọi các API này một cách dễ dàng và phần phức tạp của việc quản lý máy chủ đã được Firebase xử lý.

Một số dịch vụ phổ biến của Firebase như: Cloud Firestore, Authentication, Hosting, Cloud Storage, Realtime Database, ...

1.2 Realtime Database

Cơ sở dữ liệu thời gian thực (Realtime Database) của Firebase là một cơ sở dữ liệu NoSQL dựa trên cấu trúc JSON, được thiết kế để lưu trữ và đồng bộ dữ liệu thời gian thực giữa các thiết bị và máy chủ. Đây là một phần quan trọng trong hệ thống Firebase, cho phép các ứng dụng cập nhật dữ liệu và nhận thông báo ngay lập tức khi có sự thay đổi.

Cấu trúc dữ liệu: Realtime Database lưu trữ dữ liệu theo cấu trúc JSON, cho phép tổ chức dữ liệu thành các cặp key-value và các cấu trúc phức tạp hơn.

Sự đồng bộ thời gian thực: Dữ liệu được đồng bộ hóa ngay lập tức trên tất cả các thiết bị kết nối khi có thay đổi, giúp các ứng dụng hiển thị thông tin mới mà không cần phải làm mới trang.

Event Listener: Firebase cung cấp các event listener cho phép ứng dụng theo dõi và phản ứng khi dữ liệu thay đổi. Điều này cho phép các ứng dụng thực hiện các hành động phụ thuộc vào dữ liệu mà không cần phải kiểm tra thường xuyên.

Realtime database chấp nhận nhiều kiểu dữ liệu: String, Long, Double, Boolean, Map<String, Object>, List<Object> để lưu trữ. Chúng ta cũng có thể sử dụng custom java objects để lưu trữ dữ liệu.

1.3 Cloud Firestore

Cloud Firestore là một cơ sở dữ liệu NoSQL linh hoạt và mạnh mẽ của Firebase. Nó cung cấp một cách tiếp cận tài liệu (document) và bộ sưu tập (collection) cho phép lưu trữ dữ liệu dễ dàng và phản hồi nhanh chóng. Đây là một bộ phận quan trọng trong hệ thống Firebase, được thiết kế để cung cấp tính linh hoạt cao và khả năng mở rộng tốt cho mobile, web và server.

Cũng giống như Firebase realtime database Cloud Firestore giúp cho việc đồng bộ dữ liệu giữa các ứng dụng phía client một cách nhanh chóng (Realtime) và hỗ trợ lưu offline data trong ứng dụng của bạn.

Cấu trúc dữ liệu linh hoạt: Firestore lưu trữ dữ liệu theo cấu trúc tài liệu và bộ sưu tập (collections-documents), cho phép tổ chức dữ liệu một cách hiệu quả hơn, tương tự như các bảng và hàng trong cơ sở dữ liệu quan hệ.

Đồng bộ thời gian thực: Dữ liệu được đồng bộ hóa ngay lập tức trên tất cả các thiết bị kết nối khi có thay đổi, cho phép các ứng dụng hiển thị thông tin mới mà không cần làm mới trang.

Đồng bộ hoá ngoại tuyến: Tính năng này của Firestore có nhiệm vụ thực hiện bộ nhớ đệm dữ liệu đang được dùng và cho phép ứng dụng có thể đọc, ghi, truy vấn và lắng nghe dữ liệu ngay cả khi thiết bị đang ngoại tuyến.

Truy vấn mạnh mẽ: Firestore hỗ trợ các truy vấn linh hoạt để truy xuất dữ liệu, cho phép lọc, sắp xếp và giới hạn dữ liệu trả về.

1.4 Authentication

Nền tảng đa dạng: Firebase Authentication hỗ trợ xác thực trên nhiều nền tảng, bao gồm email/password, số điện thoại, mạng xã hội như Google, Facebook, Twitter, GitHub và nhiều hơn nữa.

Xác thực mạnh mẽ và an toàn: Cung cấp các phương thức xác thực an toàn để ngăn chặn truy cập trái phép hoặc không được ủy quyền vào ứng dụng.

Quản lý người dùng linh hoạt: Cho phép quản lý thông tin người dùng, thiết lập quyền truy cập và tùy chỉnh trải nghiệm người dùng.

Xác thực đa cấp độ: Firebase cung cấp các cấp độ xác thực khác nhau để phù hợp với yêu cầu cụ thể của ứng dụng.

1.5 Cloud Storage

Cloud Storage trong Firebase cung cấp một dịch vụ lưu trữ đám mây mạnh mẽ, cho phép bạn lưu trữ và quản lý các tệp như hình ảnh, video, tệp âm thanh và các tài nguyên khác của ứng dụng của mình.

Lưu trữ tệp đa dạng: Cho phép lưu trữ nhiều loại tệp khác nhau với các định dạng và kích thước khác nhau.

An toàn và bảo mật: Dữ liệu được lưu trữ trên cơ sở hạ tầng đám mây an toàn và được bảo vệ bởi các biện pháp bảo mật nghiêm ngặt của Google.

Đa khu vực và sao lưu tự động: Dữ liệu được lưu trữ tại các trung tâm dữ liệu khắp thế giới, đồng thời có khả năng sao lưu tự động để đảm bảo an toàn và sẵn sàng.

Quản lý tệp từ Firebase Console và API: Firebase cung cấp giao diện người dùng trực quan qua Firebase Console và các API để quản lý, tải lên, tải xuống và xử lý tệp.

CHƯƠNG 2. QUẢN LÝ THÔNG TIN HỌC SINH

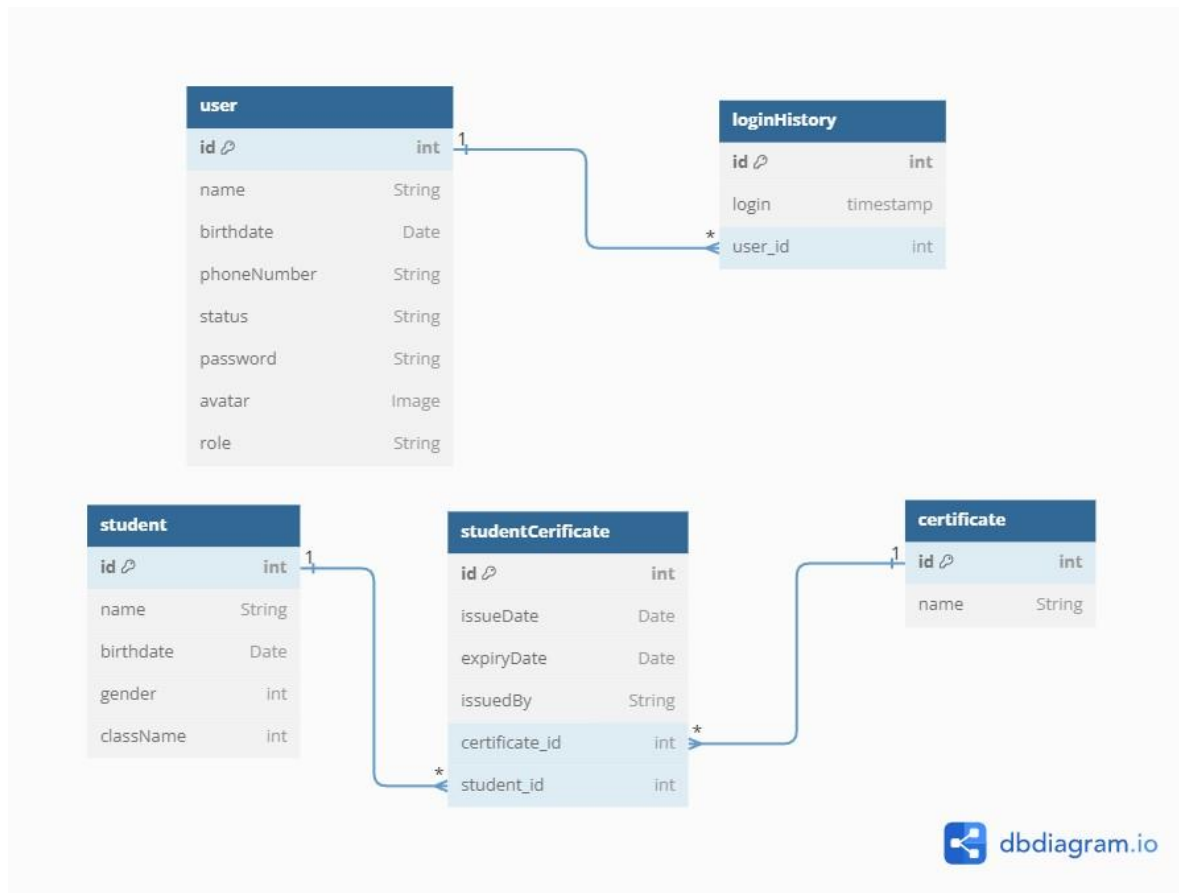
2.1 Đặc tả yêu cầu

Ứng dụng cần xây dựng là một nền tảng quản lý người dùng và sinh viên, với khả năng quản lý tài khoản người dùng, thông tin cá nhân, và danh sách sinh viên một cách linh hoạt và tiện lợi. Người dùng sẽ có khả năng đăng nhập, thay đổi hình đại diện, và quản lý thông tin cá nhân của họ. Chức năng quản lý người dùng bao gồm việc xem, thêm mới, xóa, và chỉnh sửa thông tin của người dùng, cũng như xem lịch sử đăng nhập của họ.

Đối với quản lý sinh viên, ứng dụng sẽ cung cấp khả năng xem, thêm mới, xóa, cập nhật thông tin sinh viên, và thực hiện các chức năng sắp xếp và tìm kiếm dựa trên các tiêu chí khác nhau. Đi kèm với việc quản lý sinh viên là khả năng truy cập vào trang chi tiết của mỗi sinh viên, hiển thị thông tin đầy đủ và quản lý danh sách chứng chỉ cá nhân của họ, bao gồm xem, thêm mới, xóa, và cập nhật thông tin chứng chỉ.

Ứng dụng cũng hỗ trợ nhập và xuất dữ liệu từ file, cho phép người dùng nhập danh sách sinh viên hoặc danh sách chứng chỉ từ file, cũng như xuất danh sách sinh viên và chứng chỉ sang định dạng Excel hoặc CSV. Để quản lý quyền hạn, hệ thống cung cấp ba vai trò người dùng: admin, quản lý, và nhân viên. Admin có quyền thực hiện tất cả các chức năng, quản lý có thể quản lý sinh viên và chứng chỉ, trong khi nhân viên chỉ có thể xem nội dung và cập nhật hình đại diện cá nhân của họ. Tài khoản người dùng bình thường không được phép tự tạo tài khoản, chỉ admin mới có thể tạo mới các tài khoản khác.

2.2 Sơ đồ thực thể - quan hệ (ERD) mức vật lý



Hình 1 Sơ đồ thực thể - quan hệ (ERD) mức vật lý

Bảng user sẽ lưu thông tin cơ bản của người dùng trong hệ thống như tên (name), ngày sinh (birthdate), số điện thoại (phoneNumber), trạng thái (status), mật khẩu (password), avatar và role (quyền hạn trong hệ thống).

Mỗi user sẽ có nhiều bản ghi lịch sử đăng nhập (login history), bảng lịch sử đăng nhập chứa thông tin về thời gian đăng nhập và user_id là khóa ngoại trỏ đến user.

Mỗi học sinh sẽ có các thông tin cơ bản như tên, ngày sinh, giới tính, lớp học hiện tại, mỗi học sinh sẽ có 1 hoặc nhiều chứng chỉ, thông tin về chứng chỉ bao gồm ngày thi, ngày hết hạn, đơn vị cấp.

2.3 Hiện thực ứng dụng

Em chọn sử dụng Cloud Firestore để làm backend lưu trữ dữ liệu thay vì Realtime Database bởi vì Cloud Firestore nhanh hơn, cấu trúc lưu trữ rõ ràng dễ đọc hơn so với Realtime Database.

2.3.1 Chức năng đăng nhập

Để thực hiện chức năng đăng nhập, em chọn cách làm thủ công thay vì ứng dụng Authentication của Firebase.

Logic để xác thực: Đầu tiên sẽ tìm trên firestore xem phone đã tồn tại hay chưa, nếu phone đã tồn tại thì trả về null để báo rằng đăng nhập không thành công, nếu tìm thấy phone thì tiếp tục thực hiện kiểm tra mật khẩu, mật khẩu nhập vào phải trùng khớp với mật khẩu đã được lưu trên firestore, mật khẩu này đã được mã hóa.

Em sẽ dùng Completable để làm kiểu dữ liệu trả về cho hàm này, Completable là một lớp được thiết kế để xử lý các tác vụ bất đồng bộ. Đoạn code thực hiện chức năng đăng nhập như sau:


```

1 usage Thanh Tan Tran
public CompletableFuture<User> loginUser(String phone, String password) {

    CompletableFuture<User> result = new CompletableFuture<>();

    // Tìm người dùng với số điện thoại tương ứng
    userCollection.whereEqualTo( field: "phone", phone) Query
        .get() Task<QuerySnapshot>
        .addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                QuerySnapshot querySnapshot = task.getResult();
                if (querySnapshot != null && !querySnapshot.isEmpty()) {
                    // Tìm thấy người dùng với số điện thoại đã nhập
                    DocumentSnapshot userDocument = querySnapshot.getDocuments().get(0);
                    User user = userDocument.toObject(User.class);

                    // Lấy mật khẩu đã hash từ cơ sở dữ liệu
                    String hashedPasswordFromDB = user.getPassword();

                    // So sánh mật khẩu đã nhập với mật khẩu đã hash trong cơ sở dữ liệu
                    if (PasswordHasher.verifyPassword(password, hashedPasswordFromDB)) {
                        // Mật khẩu đúng, trả về thông tin người dùng
                        result.complete(user);
                    } else {
                        // Sai thông tin đăng nhập, không thể đăng nhập
                        result.complete( value: null);
                    }
                } else {
                    // Không tìm thấy người dùng với số điện thoại đã nhập

```

Hình 2 Đoạn code thực hiện chức năng đăng nhập (1)

```

        result.complete( value: null);
    }
} else {
    // Xử lý khi có lỗi xảy ra trong quá trình truy vấn Firestore
    Exception e = task.getException();
    result.complete( value: null);
}
});

return result;
}

```

Hình 3 Đoạn code thực hiện chức năng đăng nhập (2)

2.3.2 Chức năng đổi ảnh đại diện và cập nhật thông tin người dùng

Em thực hiện chức năng cập nhật thông tin người dùng và thay đổi ảnh đại diện chung vào một hàm, nếu dữ liệu gửi lên server gồm có Uri của ảnh mới, thì thực hiện lưu vào Firebase Storage sau đó Firebase Storage sẽ trả một callback về là một

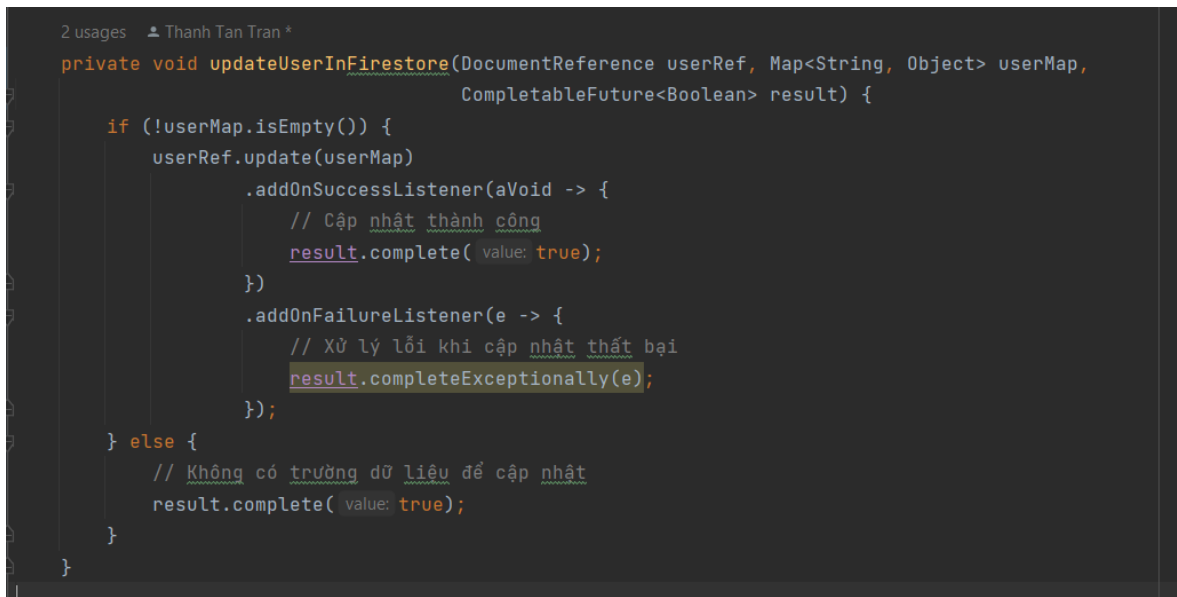
URI trỏ đến ảnh vừa thêm vào Firebase Storage, việc tiếp theo cần thực hiện đó là lưu URI này vào bản ghi User trên Firestore.

```

2 usages Thanh Tan Tran *
public CompletableFuture<Boolean> updateUser(User userUpdate) {
    CompletableFuture<Boolean> result = new CompletableFuture<>();
    DocumentReference userRef = userCollection.document(userUpdate.getId());
    StorageReference imageRef = storageReference.child(String.valueOf(System.currentTimeMillis()));
    if (userUpdate.getImgUri() != null) {
        // Check nếu avatar không null, thì upload ảnh lên storage
        imageRef.putFile(userUpdate.getImgUri()).addOnCompleteListener(task -> {
            if (task.isSuccessful() && task.isComplete()) {
                imageRef.getDownloadUrl().addOnSuccessListener(uri -> {
                    Map<String, Object> userMap = new HashMap<>();
                    userMap.put(k "avatar", uri.toString());
                    // Tiếp tục cập nhật dữ liệu người dùng với avatar
                    updateUserInFirestore(userRef, userMap, result);
                });
            } else {
                // Xử lý khi upload ảnh thất bại
                result.completeExceptionally(task.getException());
            }
        });
    } else {
        // Nếu avatar là null, chỉ cập nhật thông tin người dùng không bao gồm avatar
        Map<String, Object> userMap = new HashMap<>();
        userMap.put(k "name", userUpdate.getName());
        userMap.put(k "phone", userUpdate.getPhone());
        userMap.put(k "birthdate", userUpdate.getBirthdate());
        updateUserInFirestore(userRef, userMap, result);
    }
    return result;
}

```

Hình 4 Đoạn code cập nhật thông tin người dùng (1)



```

2 usages Thanh Tan Tran *
private void updateUserInFirestore(DocumentReference userRef, Map<String, Object> userMap,
                                   CompletableFuture<Boolean> result) {
    if (!userMap.isEmpty()) {
        userRef.update(userMap)
            .addOnSuccessListener(aVoid -> {
                // Cập nhật thành công
                result.complete(value: true);
            })
            .addOnFailureListener(e -> {
                // Xử lý lỗi khi cập nhật thất bại
                result.completeExceptionally(e);
            });
    } else {
        // Không có trường dữ liệu để cập nhật
        result.complete(value: true);
    }
}

```

Hình 5 Đoạn code cập nhật thông tin người dùng (2)

2.3.3 Chức năng xem danh sách người dùng trên hệ thống

Để hiển thị được danh sách người dùng trên hệ thống thì cần phải có một Adapter đảm nhiệm vai trò chuyển các Object User sang thành các View tương ứng. Em xin phép không chụp hình về code Adapter ở đây do khá dài, Adapter cho User có tên là UserAdapter (thầy có thể xem qua trong thư mục adapter).

Sau khi đã có adapter thì điều quan trọng kế tiếp đó là lấy dữ liệu để hiển thị. Em sẽ gọi API của Firestore để lấy toàn bộ bản ghi, sau đó chuyển các bản ghi này thành Java Object bằng cách gọi method toObject, sau khi chuyển thành Java Object, em sẽ thêm Object này vào List và trả về kết quả khi đã hoàn thành lấy dữ liệu. Hàm lấy dữ liệu sẽ có dạng sau:

```

1 usage Thanh Tan Tran
public CompletableFuture<List<User>> getAll() {
    CompletableFuture<List<User>> result = new CompletableFuture<>();
    userCollection.get()
        .addOnSuccessListener(queryDocumentSnapshots -> {
            List<User> userList = new ArrayList<>();
            // Dùng để tránh trường hợp chưa lấy dữ liệu hết đã trả về kết quả
            AtomicInteger tasksCount = new AtomicInteger(queryDocumentSnapshots.size());

            for (DocumentSnapshot doc : queryDocumentSnapshots) {
                Date loginTime = doc.getDate( field: "loginTime");
                String userId = doc.getString( field: "userId");
                User user = doc.toObject(User.class);
                userList.add(user);
                if (tasksCount.decrementAndGet() == 0) {
                    // Sắp xếp theo loại người dùng
                    Collections.sort(userList, (lu, ru) -> lu.getRole().compareTo(ru.getRole()));
                    result.complete(userList);
                }
            }
        })
        .addOnFailureListener(e -> {
            result.completeExceptionally(e);
        });
    return result;
}

```

Hình 6 Đoạn code lấy danh sách người dùng trong hệ thống

Firestore cũng cung cấp một API để theo dõi sự thay đổi từ Collection để cập nhật lại giao diện, do đó ở file Activity, em sẽ bắt sự kiện này để thực hiện render lại giao diện khi có thay đổi mới:

```

CollectionReference userRef = FirebaseFirestore.getInstance().collection( collectionPath: "user");
userRef.addSnapshotListener((querySnapshot, e) -> {
    if (e != null) {
        // Xử lý lỗi khi lắng nghe
        return;
    }
    if (querySnapshot != null) {
        List<User> updatedUserList = new ArrayList<>();
        for (QueryDocumentSnapshot document : querySnapshot) {
            User updatedUser = document.toObject(User.class);
            updatedUserList.add(updatedUser);
        }
        userAdapter.setUserList(updatedUserList);
        binding.rcvUsers.getAdapter().notifyDataSetChanged();
    }
});
}

```

Hình 7 Đoạn code bắt sự kiện thay đổi trên collection

2.3.4 Chức năng thêm một người dùng mới

Để thêm một người dùng mới thì sau khi lấy giá trị được nhập từ Admin, tất cả thông tin này sẽ được lưu vào Java Object là User mà em đã định nghĩa sẵn, sau đó gọi hàm để thêm. Nếu số điện thoại của người dùng chưa tồn tại thì hoàn toàn có thể thêm, nếu đã tồn tại không cho phép thêm.

```
1 usage Thanh Tan Tran
public CompletableFuture<Boolean> createNewUser(User user) {

    CompletableFuture<Boolean> result = new CompletableFuture<>();

    // Kiểm tra xem số điện thoại đã tồn tại trong Firestore chưa
    userCollection.whereEqualTo( field: "phone", user.getPhone()) Query
    .get() Task<QuerySnapshot>
    .addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            QuerySnapshot querySnapshot = task.getResult();
            if (querySnapshot != null && !querySnapshot.isEmpty()) {
                // Số điện thoại đã tồn tại, không cho tạo tài khoản mới
                // Xử lý thông báo hoặc hành động phù hợp ở đây
                result.complete( value: false);
            } else {
                // Số điện thoại chưa tồn tại, tạo tài khoản mới
                user.setPassword(PasswordHasher.hashPassword(user.getPassword()));
                userCollection.add(user)
                .addOnCompleteListener(task1 -> {
                    if (task1.isSuccessful()) {
                        // Tài khoản mới đã được tạo thành công
                        DocumentReference document = task1.getResult();
                        String userId = document.getId(); // Lấy ID của tài khoản vừa được tạo
                        // Thực hiện hành động sau khi tạo tài khoản thành công (nếu cần)
                        result.complete( value: true);
                    } else {
                        // Xử lý khi có lỗi xảy ra trong quá trình tạo tài khoản
                        Exception e = task1.getException();
                        result.complete( value: false);
                    }
                }
            }
        }
    })
}
```

Hình 8 Đoạn code thêm người dùng mới (1)

```
    }
    });
} else {
    // Xử lý khi có lỗi xảy ra trong quá trình truy vấn Firestore
    Exception e = task.getException();
    result.complete( value: false);
}
});
return result;
}
```

Hình 9 Đoạn code thêm người dùng mới (2)

2.3.5 Chức năng xóa người dùng

Để thực hiện chức năng này thì em sẽ tạo một hàm để nhận tham số đầu vào là id của user, sau đó thực hiện gọi API xóa của Firestore.

```
new *
public CompletableFuture<Boolean> deleteUser(String userId) {
    CompletableFuture<Boolean> result = new CompletableFuture<>();
    userCollection.document(userId).delete()
        .addOnSuccessListener(aVoid -> {
            result.complete(value: true);
        })
        .addOnFailureListener(e -> {
            result.complete(value: false);
        });
    return result;
}
```

Hình 10 Đoạn code xóa người dùng

2.3.6 Chức năng xem lịch sử đăng nhập của người dùng

Tương tự với xem danh sách người dùng, cần có một adapter để chuyển Object thành View, tên của adapter là LoginHistoryAdapter (thầy có thể xem trong thư mục adapter). Sau khi có adapter, việc tiếp theo là lấy danh sách các lịch sử.

```

1 usage  Thanh Tan Tran *
public CompletableFuture<List<LoginHistoryDTO>> getAll() {
    CompletableFuture<List<LoginHistoryDTO>> future = new CompletableFuture<>();
    historyCollection.get()
        .addOnSuccessListener(queryDocumentSnapshots -> {
            List<LoginHistoryDTO> loginHistories = new ArrayList<>();
            // Dùng để tránh trường hợp chưa lấy dữ liệu hết đã trả về kết quả
            AtomicInteger tasksCount = new AtomicInteger(queryDocumentSnapshots.size());
            for (DocumentSnapshot doc : queryDocumentSnapshots) {
                Date loginTime = doc.getDate( field: "loginTime");
                String userId = doc.getString( field: "userId");
                DocumentReference userRef = db.collection( collectionPath: "user").document(userId);
                userRef.get()
                    .addOnSuccessListener(userDocumentSnapshot -> {
                        User user = userDocumentSnapshot.toObject(User.class);
                        LoginHistoryDTO loginHistoryDTO = new LoginHistoryDTO(doc.getId(), loginTime, user);
                        loginHistories.add(loginHistoryDTO);
                        Log.d( tag: "HISFIRE", loginTime.toString());

                        // Giảm số lượng tác vụ cần hoàn thành và kiểm tra khi nào hoàn thành hết
                        if (tasksCount.decrementAndGet() == 0) {
                            // Sắp xếp theo thời gian đăng nhập
                            Collections.sort(loginHistories, (lhs, rhs) -> rhs.getLoginTime().compareTo(lhs.getLoginTime()));
                            future.complete(loginHistories);
                        }
                    })
            }
        })
        .addOnFailureListener(e -> {
            future.completeExceptionally(e); // Hoàn thành CompletableFuture với lỗi nếu có lỗi xảy ra
        });
}

```

Hình 11 Đoạn code xem lịch sử đăng nhập (1)

```

        });
    }
}
.addOnFailureListener(e -> {
    future.completeExceptionally(e);
});
return future;
}

```

Hình 12 Đoạn code xem lịch sử đăng nhập (2)

2.3.7 Chức năng xem danh sách học sinh

Tương tự với xem danh sách người dùng và lịch sử đăng nhập, tên adapter là StudentAdapter (thầy có thể xem trong thư mục adapter).

Thực hiện lấy danh sách học sinh từ Firestore:

```

1 usage Thanh Tan Tran
public CompletableFuture<List<Student>> getAll() {
    CompletableFuture<List<Student>> result = new CompletableFuture<>();
    studentCollection.get()
        .addOnSuccessListener(queryDocumentSnapshots -> {
            List<Student> studentList = new ArrayList<>();
            // Dùng để tránh trường hợp chưa lấy dữ liệu hết đã trả về kết quả
            AtomicInteger tasksCount = new AtomicInteger(queryDocumentSnapshots.size());

            for (DocumentSnapshot doc : queryDocumentSnapshots) {
                Student student = doc.toObject(Student.class);
                studentList.add(student);
                if (tasksCount.decrementAndGet() == 0) {
                    // Sắp xếp theo lớp
                    Collections.sort(studentList, Comparator.comparingInt(Student::getClassName));
                    result.complete(studentList);
                }
            }
        })
        .addOnFailureListener(e -> {
            result.completeExceptionally(e);
        });
    return result;
}

```

Hình 13 Đoạn code lấy danh sách học sinh

2.3.8 Chức năng thêm một học sinh mới

Để thêm một học sinh mới thì sau khi nhận được dữ liệu nhập vào từ người dùng thì gọi hàm thêm học sinh mới:

```

1 usage new *
public CompletableFuture<Boolean> insert(Student student) {
    CompletableFuture<Boolean> result = new CompletableFuture<>();

    studentCollection.add(student)
        .addOnSuccessListener(documentReference -> {
            // Xử lý khi việc thêm thành công -> tạo subcollection
            CollectionReference certificateCollection = documentReference.collection(collectionPath: "certificateList");
            result.complete(value: true);
        })
        .addOnFailureListener(e -> {
            // Xử lý khi việc thêm thất bại
            result.completeExceptionally(e);
        });
    return result;
}

```

Hình 14 Đoạn code thêm học sinh mới

2.3.9 Chức năng xóa một học sinh

Để xóa một học sinh thì cần phải truyền tham số là studentId, sau đó gọi API của Firestore để xóa học sinh.


```

new *
public CompletableFuture<Boolean> deleteStudent(String studentId) {
    CompletableFuture<Boolean> result = new CompletableFuture<>();
    studentCollection.document(studentId).delete()
        .addOnSuccessListener(aVoid -> {
            result.complete( value: true);
        })
        .addOnFailureListener(e -> {
            result.complete( value: false);
        });
    return result;
}

```

Hình 15 Đoạn code xóa một học sinh

2.3.10 Chức năng cập nhật thông tin học sinh

Để thực hiện chức năng này, sau khi người dùng nhập đầy đủ các thông tin cần cập nhật thì gọi đến hàm update này, hàm này sẽ gọi API để cập nhật lại thông tin của học sinh.

```

Thanh Tan Tran
public CompletableFuture<Boolean> update(Student student) {
    CompletableFuture<Boolean> result = new CompletableFuture<>();
    DocumentReference studentRef = studentCollection.document(student.getId());
    if(student != null) {
        Map<String, Object> updates = new HashMap<>();
        updates.put( k: "name", student.getName());
        updates.put( k: "birthdate", student.getBirthdate());
        updates.put( k: "gender", student.getGender());
        updates.put( k: "className", student.getClassName());
        studentRef.update(updates)
            .addOnSuccessListener(aVoid -> {
                // Cập nhật thành công
                result.complete( value: true);
            })
            .addOnFailureListener(e -> {
                // Xử lý lỗi khi cập nhật thất bại
                result.completeExceptionally(e);
            });
    }
    return result;
}

```

Hình 16 Đoạn code cập nhật thông tin học sinh

2.3.11 Chức năng sắp xếp danh sách học sinh

Ở chức năng này, em thực hiện sắp xếp dựa trên danh sách đã có, em cung cấp xếp theo tên, theo lớp, và theo ngày sinh do đó để tiện em sẽ định nghĩa ra một lớp để hỗ trợ cho việc này:

```

3 usages
public class StudentSorters {
    7 usages
    public List<Comparator<Student>> comparators = new ArrayList<>();
    2 usages
    public Comparator<Student> compareByName() { return Comparator.comparing(Student::getName); }
    2 usages
    public Comparator<Student> compareByClassName() {
        return Comparator.comparing(Student::getClassName);
    }
    2 usages
    public static Comparator<Student> compareByBirthdate() {
        return Comparator.comparing(student -> DateConvert.parseDate(student.getBirthdate()));
    }
    3 usages
    public void sortBy(Comparator<Student> comparator) { comparators.add(comparator); }
    3 usages
    public void unsortBy(Comparator<Student> comparator) { comparators.remove(comparator); }

    public void clearSort() { comparators.clear(); }

    6 usages
    public void applySort(List<Student> studentList) {
        for(Comparator<Student> comparator : comparators) {
            Collections.sort(studentList, comparator);
        }
    }
}

```

Hình 17 Đoạn code hỗ trợ chức năng sắp xếp

Sau khi sắp xếp, cập nhật lại giao diện, dưới đây là đoạn code sắp xếp và cập nhật lại giao diện:

```

StudentSorters studentSorters = new StudentSorters();
List<Student> originalList = new ArrayList<>();
originalList.addAll(studentAdapter.getStudentList());
imageName.setOnClickListener(v -> {
    if(checkArr[0]) {
        Drawable drawable = ContextCompat.getDrawable(context: this, R.drawable.ic_sort_name);
        imageName.setImageDrawable(drawable);

        studentSorters.unsortBy(studentSorters.compareByName());
        studentSorters.applySort(studentAdapter.getStudentList());
        // Nếu không còn so sánh nào nữa -> rollback về mảng cũ
        if(studentSorters.comparators.size() == 0) {
            studentAdapter.setStudentList(originalList);
            studentAdapter.notifyDataSetChanged();
        }
    } else {
        Drawable drawable = ContextCompat.getDrawable(context: this, R.drawable.ic_sort_name_active);
        imageName.setImageDrawable(drawable);

        studentSorters.sortBy(studentSorters.compareByName());
        studentSorters.applySort(studentAdapter.getStudentList());
    }
    studentAdapter.notifyDataSetChanged();
    checkArr[0] = !checkArr[0];
});

```

Hình 18 Đoạn code sắp xếp theo tên

```

imgBirthdate.setOnClickListener(v -> {
    if(checkArr[1] == true) {
        Drawable drawable = ContextCompat.getDrawable(context: this, R.drawable.ic_sort_birthdate);
        imgBirthdate.setImageDrawable(drawable);

        studentSorters.unsortBy(studentSorters.compareByBirthdate());
        studentSorters.applySort(studentAdapter.getStudentList());

        // Nếu không còn so sánh nào nữa -> rollback về mảng cũ
        if(studentSorters.comparators.size() == 0) {
            studentAdapter.setStudentList(originalList);
            studentAdapter.notifyDataSetChanged();
        }
    } else {
        Drawable drawable = ContextCompat.getDrawable(context: this, R.drawable.ic_sort_birthdate_active);
        imgBirthdate.setImageDrawable(drawable);

        studentSorters.sortBy(studentSorters.compareByBirthdate());
        studentSorters.applySort(studentAdapter.getStudentList());
    }
    studentAdapter.notifyDataSetChanged();
    checkArr[1] = !checkArr[1];
});

```

Hình 19 Đoạn code sắp xếp theo ngày sinh

```

imgClass.setOnClickListener(v -> {
    if(checkArr[2] == true) {
        Drawable drawable = ContextCompat.getDrawable(context, this, R.drawable.ic_sort_class);
        imgClass.setImageDrawable(drawable);

        studentSorters.unsortBy(studentSorters.compareByClassName());
        studentSorters.applySort(studentAdapter.getStudentList());
        // Nếu không còn so sánh nào nữa -> rollback về mảng cũ
        if(studentSorters.comparators.size() == 0) {
            studentAdapter.setStudentList(originalList);
            studentAdapter.notifyDataSetChanged();
        }
    } else {
        Drawable drawable = ContextCompat.getDrawable(context, this, R.drawable.ic_sort_class_active);
        imgClass.setImageDrawable(drawable);

        studentSorters.sortBy(studentSorters.compareByClassName());
        studentSorters.applySort(studentAdapter.getStudentList());
    }

    studentAdapter.notifyDataSetChanged();
    checkArr[2] = !checkArr[2];
});

```

Hình 20 Đoạn code sắp xếp theo lớp

2.3.12 Chức năng tìm kiếm học sinh

Ở chức năng này học sinh sẽ được tìm kiếm dựa trên tên, em cũng tìm dựa trên danh sách đã được lấy trước đó, kết quả trả về sẽ là danh sách các học sinh có tên gần giống với từ khóa nhập vào. Em định nghĩa ra một lớp riêng để hỗ trợ việc tìm kiếm này:

```

2 usages
public class StudentSearch {
    1 usage
    public static List<Student> search(String name, List<Student> studentList) {
        return studentList.stream()
            .filter(student -> student.getName().toLowerCase().contains(name.toLowerCase()))
            .collect(Collectors.toList());
    }
}

```

Hình 21 Đoạn code hỗ trợ tìm kiếm học sinh

```

        btnSearch.setOnClickListener(v -> {
            String name = edtSearch.getText().toString();
            List<Student> searchList = StudentSearch.search(name, studentAdapter.getStudentList());
            studentAdapter.setStudentList(searchList);
            studentAdapter.notifyDataSetChanged();
        });

        bottomSheetDialog.show();
    }
}

```

Hình 22 Đoạn code tìm kiếm học sinh

2.3.13 Chức năng xem danh sách chứng chỉ

Tương tự với danh sách học sinh, cần có một adapter, adapter này có tên là CertificateAdapter. Sau đó gọi hàm để lấy danh sách các chứng chỉ theo studentId.

```

1 usage  Thanh Tan Tran *
public CompletableFuture<List<Certificate>> getAllCertificate(String idStudent) {
    CompletableFuture<List<Certificate>> result = new CompletableFuture<>();
    CollectionReference certificateCollection = studentCollection.document(idStudent)
        .collection(collectionPath: "certificateList");
    certificateCollection.get()
        .addOnSuccessListener(queryDocumentSnapshots -> {
            List<Certificate> certificateList = new ArrayList<>();
            // Dùng để tránh trường hợp chưa lấy dữ liệu hết đã trả về kết quả
            AtomicInteger tasksCount = new AtomicInteger(queryDocumentSnapshots.size());
            for (DocumentSnapshot doc : queryDocumentSnapshots) {
                Certificate cert = doc.toObject(Certificate.class);
                certificateList.add(cert);
                if (tasksCount.decrementAndGet() == 0) {
                    result.complete(certificateList);
                }
            }
        })
        .addOnFailureListener(e -> {
            result.completeExceptionally(e);
        });

    return result;
}

```

Hình 23 Đoạn code lấy danh sách chứng chỉ

2.3.14 Chức năng thêm chứng chỉ mới cho học sinh

Để thực hiện chức năng này sau khi người dùng nhập các thông tin của chứng chỉ thì gửi thông tin này cùng với studentId thông qua hàm phía dưới.

```

1 usage new *
public CompletableFuture<Boolean> insertCertificate(Certificate certificate, String studentId) {
    CompletableFuture<Boolean> result = new CompletableFuture<>();
    CollectionReference certificateRef = studentCollection.document(studentId).collection( collectionPath: "certificateList");
    certificateRef.add(certificate)
        .addOnSuccessListener(documentReference -> {
            result.complete( value: true);
        })
        .addOnFailureListener(e -> {
            // Xử lý khi việc thêm thất bại
            result.completeExceptionally(e);
        });
    return result;
}

```

Hình 24 Đoạn code thêm chứng chỉ mới cho học sinh

2.3.15 Chức năng xóa chứng chỉ

Hàm thực hiện xóa chứng chỉ, tham số truyền vào là id của chứng chỉ và studentId vì chứng chỉ thuộc về một học sinh nên phải truyền studentId để xóa.

```

1 usage Thanh Tan Tran *
public CompletableFuture<Boolean> deleteCertificate(String id, String studentId) {
    CompletableFuture<Boolean> result = new CompletableFuture<>();
    studentCollection.document(studentId).collection( collectionPath: "certificateList").document(id).delete()
        .addOnSuccessListener(aVoid -> {
            result.complete( value: true);
        })
        .addOnFailureListener(e -> {
            result.complete( value: false);
        });
    return result;
}

```

Hình 25 Đoạn code xóa chứng chỉ

2.3.16 Chức năng thêm danh sách học sinh từ File

Để thực hiện thêm từ File, đầu tiên phải thực hiện nhiệm vụ cấp quyền truy cập vào bộ nhớ để có thể đọc và ghi File:

```

if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
    if (!Environment.isExternalStorageManager()) {
        Log.i( tag: "=== IS EXTERNALSTORAGEMANAGER ===", msg: "TRUE");
        // Request the MANAGE_EXTERNAL_STORAGE permission
        Intent intent = new Intent(Settings.ACTION_MANAGE_ALL_FILES_ACCESS_PERMISSION);
        startActivity(intent);
    }
}

```

Hình 26 Đoạn code cấp quyền truy cập bộ nhớ

Hàm để mở một Intent cho phép người dùng chọn File Excel:

Hình 27 Đoạn code gọi Intent để chọn File (1)

```
usage
private ActivityResultLauncher<Intent> pickFileLauncher = registerForActivityResult(
    new ActivityResultContracts.StartActivityForResult(),
    result -> {
        if (result.getResultCode() == Activity.RESULT_OK) {
            Intent data = result.getData();
            if (data != null) {
                Uri uri = data.getData();
                if (uri != null) {
                    // Sử dụng URI để lấy đường dẫn thực tế của tệp
                    String filePath = RealPath.getRealPath( context: StudentActivity.this, uri);
                    CompletableFuture<Boolean> check = studentFirestore.insertStudentByFile(filePath);
                    LoadingDialog loadingDialog = new LoadingDialog( activity: this);
                    loadingDialog.show();
                    // Xử lý kết quả khi hoàn thành
                    check.thenAccept(isSuccess -> {
                        if(isSuccess) {
                            loadingDialog.dismiss();
                            Toast.makeText( context: StudentActivity.this, text: "Thêm thành công", Toast.LENGTH_SHORT).show();
                        } else {
                            loadingDialog.dismiss();
                            Toast.makeText( context: StudentActivity.this, text: "Thêm thất bại", Toast.LENGTH_SHORT).show();
                        }
                    });
                }
            }
        }
    }
);
```

Hình 28 Đoạn code gọi Intent để chọn File (2)

Khi đã chọn File thành công, một callback sẽ được gọi lại, kết quả trả về sẽ nhận một URI, sau đó em lấy URI này để đổi thành một đường dẫn tuyệt đối trỏ đến File vừa chọn, nhiệm vụ cuối cùng là đọc file này và insert từng dòng dữ liệu bên trong nó. Để xử lý nhiệm vụ này, em có hàm sau:

```

1 usage new *
public CompletableFuture<Boolean> insertStudentByFile(String filePath) {
    CompletableFuture<Boolean> result = new CompletableFuture<>();
    try {
        FileInputStream fileInputStream = new FileInputStream(filePath);
        if (fileInputStream != null) {
            Workbook workbook = new XSSFWorkbook(fileInputStream);
            // Đọc sheet đầu tiên từ workbook
            Sheet sheet = workbook.getSheetAt(index: 0);

            // Đọc dữ liệu từ từng dòng trong sheet và thêm học sinh vào Firestore
            for (Row row : sheet) {
                if (row.getRowNum() == 0) {
                    // Bỏ qua dòng tiêu đề (nếu cần)
                    continue;
                }
                Log.d(tag: "Excel", row.getCell(cellnum: 0).getStringCellValue());
                String name = row.getCell(cellnum: 0).getStringCellValue();
                String birthdate = row.getCell(cellnum: 1).getStringCellValue();
                int gender = row.getCell(cellnum: 2).getStringCellValue().equalsIgnoreCase(anotherString: "Nam") ? 1 : 0;
                int className = (int) row.getCell(cellnum: 3).getNumericCellValue();
                // Tạo đối tượng Student từ dữ liệu trong dòng của Excel
                Student student = new Student();
                student.setName(name);
                student.setBirthdate(birthdate);
                student.setGender(gender);
                student.setClassName(className);

                // Thêm đối tượng Student vào Firestore
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
        result.completeExceptionally(e);
    }
    return result;
}

```

Hình 29 Đoạn code thêm học sinh từ File (1)

```

        studentCollection.add(student)
            .addOnSuccessListener(documentReference -> {
                // Xử lý khi việc thêm thành công
                // Có thể log hoặc xử lý tiếp theo ở đây
            })
            .addOnFailureListener(e -> {
                // Xử lý khi việc thêm thất bại
                result.completeExceptionally(e);
            });

        // Đóng workbook và file sau khi hoàn thành việc đọc
        workbook.close();
        fileInputStream.close();
        result.complete(value: true); // Đánh dấu hoàn thành khi đã đọc và thêm học sinh thành công
    } else {
        result.complete(value: false);
    }
}

} catch (IOException e) {
    e.printStackTrace();
    result.completeExceptionally(e);
}

return result;
}

```

Hình 30 Đoạn code thêm học sinh từ File (2)

2.3.17 Chức năng xuất danh sách học sinh ra File

Chức năng xuất danh sách học sinh ra File không phức tạp như thêm từ File, ở xuất File em chỉ tạo ra một File Excel sau đó với mỗi học sinh trong danh sách học sinh truyền vào từ hàm em sẽ tạo một hàng trong excel.

```

1 usage: new *
public boolean exportStudentToFile(List<Student> students) {
    String fileName = "students.xlsx";
    String filePath = Environment.getExternalStorageDirectory() + "/" + fileName;
    try (Workbook workbook = new XSSFWorkbook()) {
        Sheet sheet = workbook.createSheet( sheetname: "Students");
        // Create header row
        Row headerRow = sheet.createRow( rownum: 0);
        headerRow.createCell( column: 0).setCellValue("Tên");
        headerRow.createCell( column: 1).setCellValue("Ngày sinh");
        headerRow.createCell( column: 2).setCellValue("Giới tính");
        headerRow.createCell( column: 3).setCellValue("Lớp");
        // Populate data rows
        int rowNum = 1;
        for (Student student : students) {
            Row row = sheet.createRow(rowNum++);
            row.createCell( column: 0).setCellValue(student.getName());
            row.createCell( column: 1).setCellValue(student.getBirthdate());
            String gender = student.getGender() == 1 ? "Nam" : "Nữ";
            row.createCell( column: 2).setCellValue(gender);
            row.createCell( column: 3).setCellValue(student.getClassName() + "");
        }
        // Write to file
        File file = new File(filePath);
        FileOutputStream fileOut = new FileOutputStream(file);
        workbook.write(fileOut);
        fileOut.close();
        return true;
        // Show a message or handle the success event
    } catch (IOException e) {

```

Hình 31 Đoạn code xuất danh sách học sinh ra File (1)

```

        e.printStackTrace();
        // Handle error
        return false;
    }
}

```

Hình 32 Đoạn code xuất danh sách học sinh ra File (2)

2.3.18 Chức năng thêm danh sách chứng chỉ từ File

Tương tự với thêm danh sách học sinh, thêm danh sách chứng chỉ cũng tương tự ở phần chọn File Excel, chỉ khác ở phần xử lý insert.

```
1 usage new ^
public CompletableFuture<Boolean> insertCertificateByFile(String filePath, String studentId) {
    CompletableFuture<Boolean> result = new CompletableFuture<>();
    try {
        FileInputStream fileInputStream = new FileInputStream(filePath);
        if (fileInputStream != null) {
            Workbook workbook = new XSSFWorkbook(fileInputStream);
            // Đọc sheet đầu tiên từ workbook
            Sheet sheet = workbook.getSheetAt(index: 0);
            // Đọc dữ liệu từ từng dòng trong sheet và thêm certificate vào Firestore
            for (Row row : sheet) {
                if (row.getRowNum() == 0) {
                    // Bỏ qua dòng tiêu đề (nếu cần)
                    continue;
                }
                String name = row.getCell(cellnum: 0).getStringCellValue();
                String issueDate = row.getCell(cellnum: 1).getStringCellValue();
                String expiryDate = row.getCell(cellnum: 2).getStringCellValue();
                String issuedBy = row.getCell(cellnum: 3).getStringCellValue();
                // Tạo đối tượng Certificate từ dữ liệu trong dòng của Excel
                Certificate certificate = new Certificate();
                certificate.setName(name);
                certificate.setIssueDate(issueDate);
                certificate.setExpiryDate(expiryDate);
                certificate.setIssuedBy(issuedBy);

                // Thêm đối tượng vào Firestore
                studentCollection.document(studentId).collection(collectionPath: "certificateList").add(certificate)
                    .addOnSuccessListener(documentReference -> {
                        // Xử lý khi việc thêm thành công
                    })
            }
        }
    } catch (Exception e) {
        result.complete(false);
    }
    return result;
}
```

Hình 33 Đoạn code thêm danh sách chứng chỉ từ File (1)

```

        // Xử lý khi việc thêm thành công
        // Có thể log hoặc xử lý tiếp theo ở đây
    })
    .addOnFailureListener(e -> {
        // Xử lý khi việc thêm thất bại
        result.completeExceptionally(e);
    });
}

// Đóng workbook và file sau khi hoàn thành việc đọc
workbook.close();
fileInputStream.close();
result.complete( value: true); // Đánh dấu hoàn thành khi đã đọc và thêm thành công
} else {
    result.complete( value: false);
}

} catch (IOException e) {
    e.printStackTrace();
    result.completeExceptionally(e);
}

return result;
}

```

Hình 34 Đoạn code thêm danh sách chứng chỉ từ File (2)

2.3.19 Chức năng xuất danh sách chứng chỉ ra File

Cũng tương tự như với chức năng xuất danh sách học sinh ra File, chức năng này cũng tương tự.

Đoạn code xử lý như sau:

```

1 usage new *
public boolean exportCertificate(List<Certificate> certificateList) {
    String fileName = "certificates.xlsx";
    String filePath = Environment.getExternalStorageDirectory() + "/" + fileName;

    try (Workbook workbook = new XSSFWorkbook()) {
        Sheet sheet = workbook.createSheet("Certificates");

        // Create header row
        Row headerRow = sheet.createRow(0);
        headerRow.createCell(0).setCellValue("Tên chứng chỉ");
        headerRow.createCell(1).setCellValue("Ngày thi");
        headerRow.createCell(2).setCellValue("Ngày hết hạn");
        headerRow.createCell(3).setCellValue("Đơn vị cấp");

        // Populate data rows
        int rowNum = 1;
        for (Certificate certificate : certificateList) {
            Row row = sheet.createRow(rowNum++);
            row.createCell(0).setCellValue(certificate.getName());
            row.createCell(1).setCellValue(certificate.getIssueDate());
            row.createCell(2).setCellValue(certificate.getExpiryDate());
            row.createCell(3).setCellValue(certificate.getIssuedBy());
        }

        // Write to file
        File file = new File(filePath);
        FileOutputStream fileOut = new FileOutputStream(file);
        workbook.write(fileOut);
    }
}

```

Hình 35 Đoạn code xử lý xuất danh sách chứng chỉ ra File (1)

```

        fileOut.close();
        return true;
        // Show a message or handle the success event
    } catch (IOException e) {
        e.printStackTrace();
        // Handle error
        return false;
    }
}

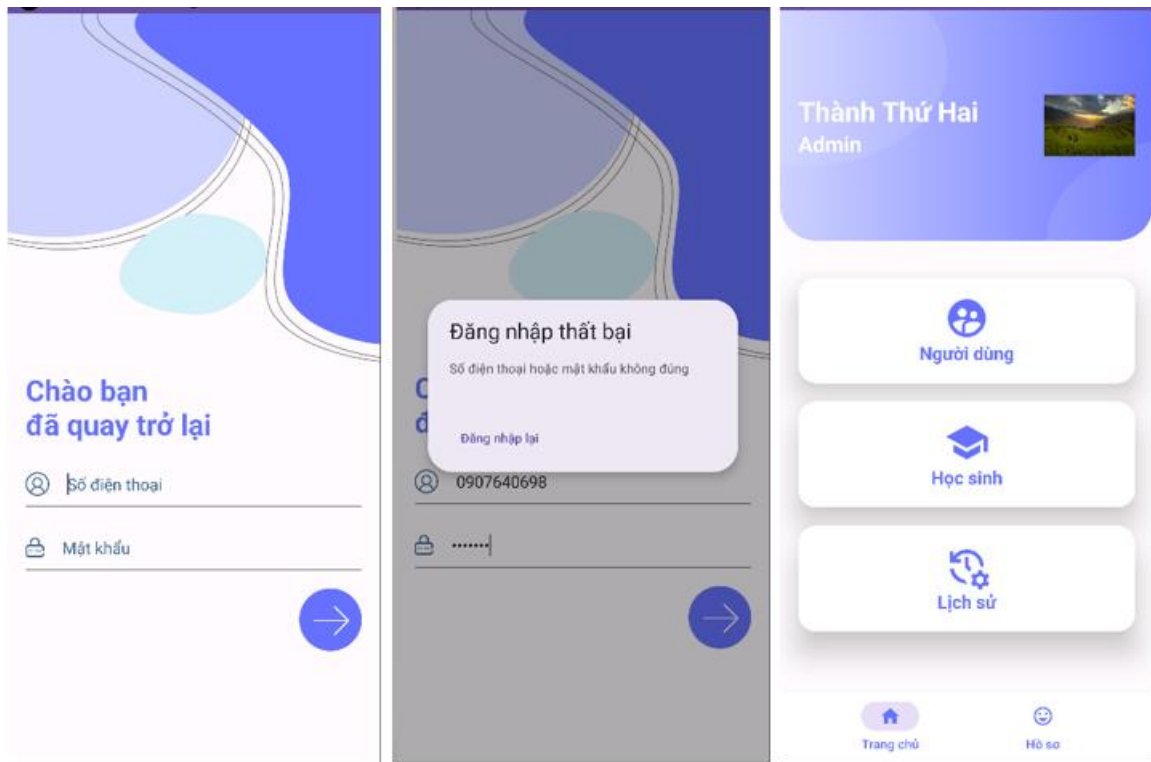
```

Hình 36 Đoạn code xử lý xuất danh sách chứng chỉ ra File (2)

CHƯƠNG 3. DEMO

3.1 Chức năng đăng nhập

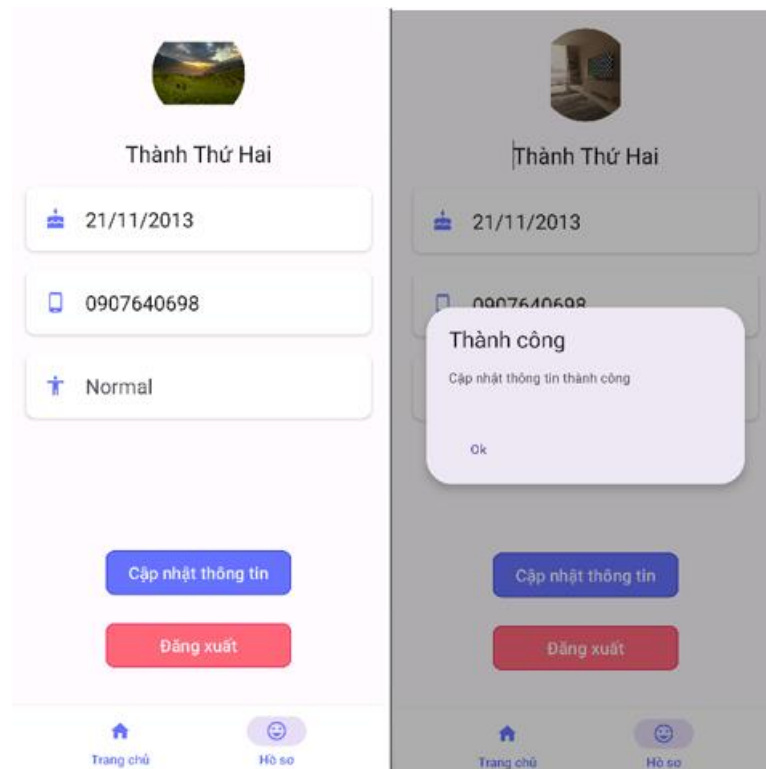
Giao diện đăng nhập, em sẽ tiến hành đăng nhập thất bại và thành công theo thứ tự để kiểm thử.



Hình 37 Demo chức năng đăng nhập

3.2 Chức năng đổi ảnh đại diện và cập nhật thông tin người dùng

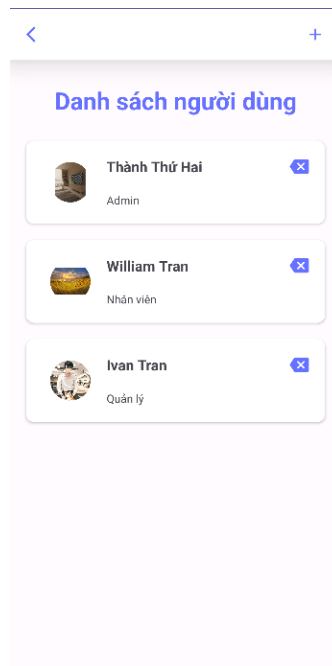
Giao diện trang thông tin cá nhân, em sẽ thực hiện thay đổi ảnh đại diện và tên để kiểm thử:



Hình 38 Demo chức năng đổi ảnh đại diện và cập nhật thông tin

3.3 Chức năng xem danh sách người dùng trên hệ thống

Nếu là Admin thì sẽ truy cập được quyền này, giao diện như sau:



Hình 39 Demo chức năng xem danh sách người dùng

3.4 Chức năng thêm một người dùng mới

The image displays two side-by-side screenshots of a mobile application interface. The left screenshot shows a form titled 'Thêm người dùng mới' (Add New User) with the following fields: 'Tên' (Name) with the value 'Demo', 'Ngày sinh' (Date of birth) with the value '10/02/2000', and 'Số điện thoại' (Phone number) with the value '0987654322'. Below these fields, there are radio buttons for 'Quyền hạn' (Permissions): 'Quản lý' (Manager) is selected, and 'Nhân viên' (Employee) is unselected. A blue 'Thêm' (Add) button is at the bottom. The right screenshot shows the 'Danh sách người dùng' (User List) screen. It contains a list of four users: 'Thành Thử Hai' (Admin), 'William Tran' (Nhân viên), 'Demo' (Quản lý), and 'Ivan Tran' (Quản lý). Each user entry has a profile picture, a name, a role, and a delete icon. At the bottom of the list, there is a green button with a checkmark and the text 'Thêm thành công' (Added successfully).

Danh sách người dùng

Tên
Demo

Ngày sinh
10/02/2000

Số điện thoại
0987654322

Quyền hạn
☒ Quản lý ☐ Nhân viên

Thêm

Danh sách người dùng

Thành Thử Hai Admin

William Tran Nhân viên

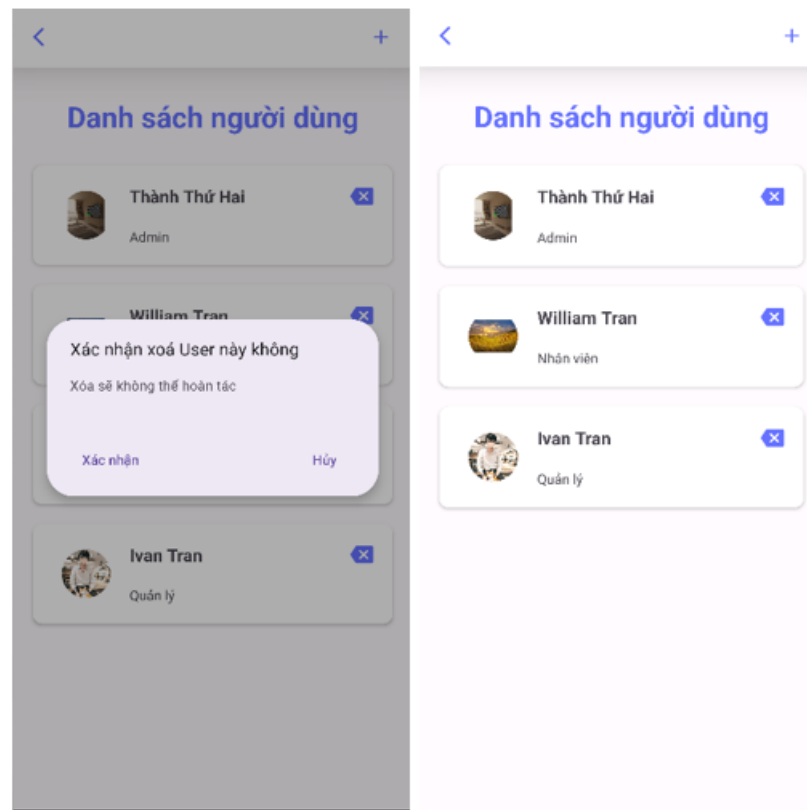
Demo Quản lý

Ivan Tran Quản lý

Thêm thành công

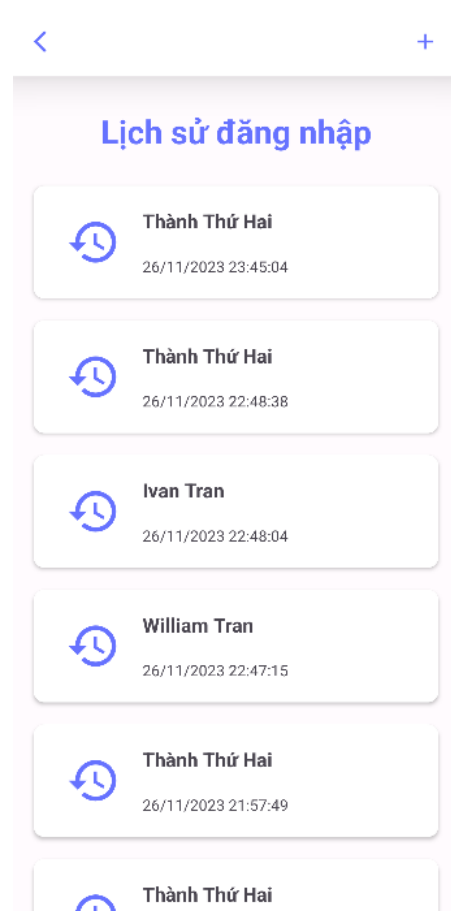
Hình 40 Demo chức năng thêm người dùng mới

3.5 Chức năng xóa người dùng



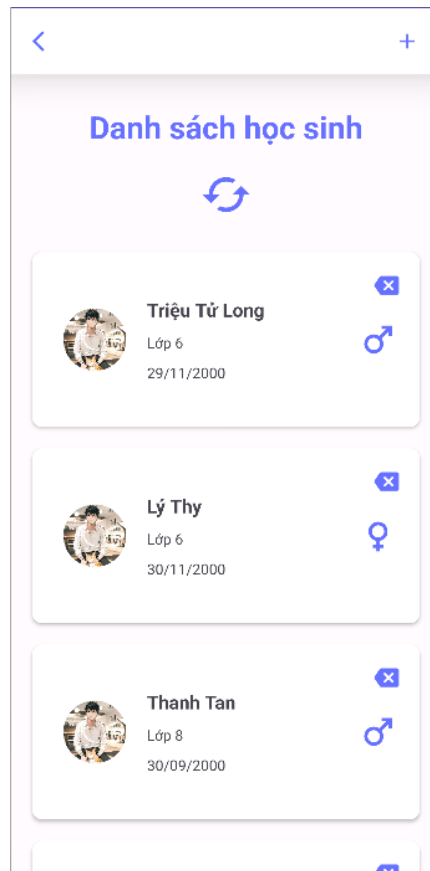
Hình 41 Demo chức năng xóa người dùng

3.6 Chức năng xem lịch sử đăng nhập của người dùng



Hình 42 Demo chức năng xem lịch sử đăng nhập

3.7 Demo chức năng xem danh sách học sinh



Hình 43 Demo chức năng xem danh sách học sinh

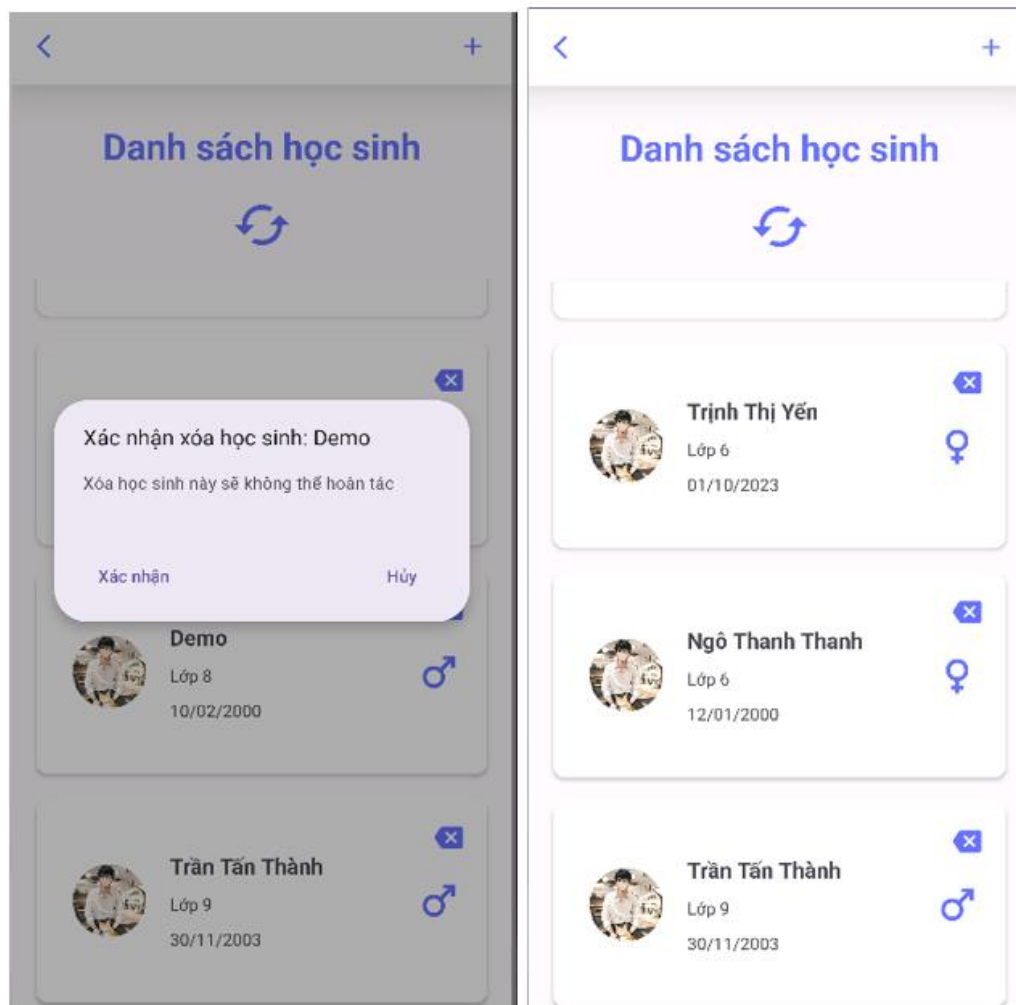
3.8 Demo chức năng thêm một học sinh mới

The image displays two side-by-side mobile application screens. The left screen is a form titled "Danh sách học sinh" (Student List) with a refresh icon. It contains input fields for "Tên" (Name) with the value "Demo", "Ngày sinh" (Date of Birth) with the value "10/02/2000", and "Giới tính" (Gender) with radio buttons for "Nam" (Male) and "Nữ" (Female), where "Nam" is selected. Below these is a "Lớp" (Class) section with radio buttons for "Lớp 6", "Lớp 7", "Lớp 8", and "Lớp 9", where "Lớp 8" is selected. A blue button at the bottom is labeled "Thêm học sinh" (Add student). The right screen also shows the "Danh sách học sinh" title with a refresh icon. It displays a list of three student entries, each with a profile picture, name, class, date of birth, and gender icon. The entries are: 1. Lý Thy, Lớp 6, 30/11/2000, Female. 2. Thanh Tan, Lớp 8, 30/09/2000, Male. 3. Demo, Lớp 8, 10/02/2000, Male. Each entry has a delete icon (a blue square with a white 'x') in the top right corner.

Tên	Lớp	Ngày sinh	Giới tính
Lý Thy	Lớp 6	30/11/2000	Nữ
Thanh Tan	Lớp 8	30/09/2000	Nam
Demo	Lớp 8	10/02/2000	Nam

Hình 44 Demo chức năng thêm một học sinh mới

3.9 Demo chức năng xóa một học sinh



Hình 45 Demo chức năng xóa học sinh

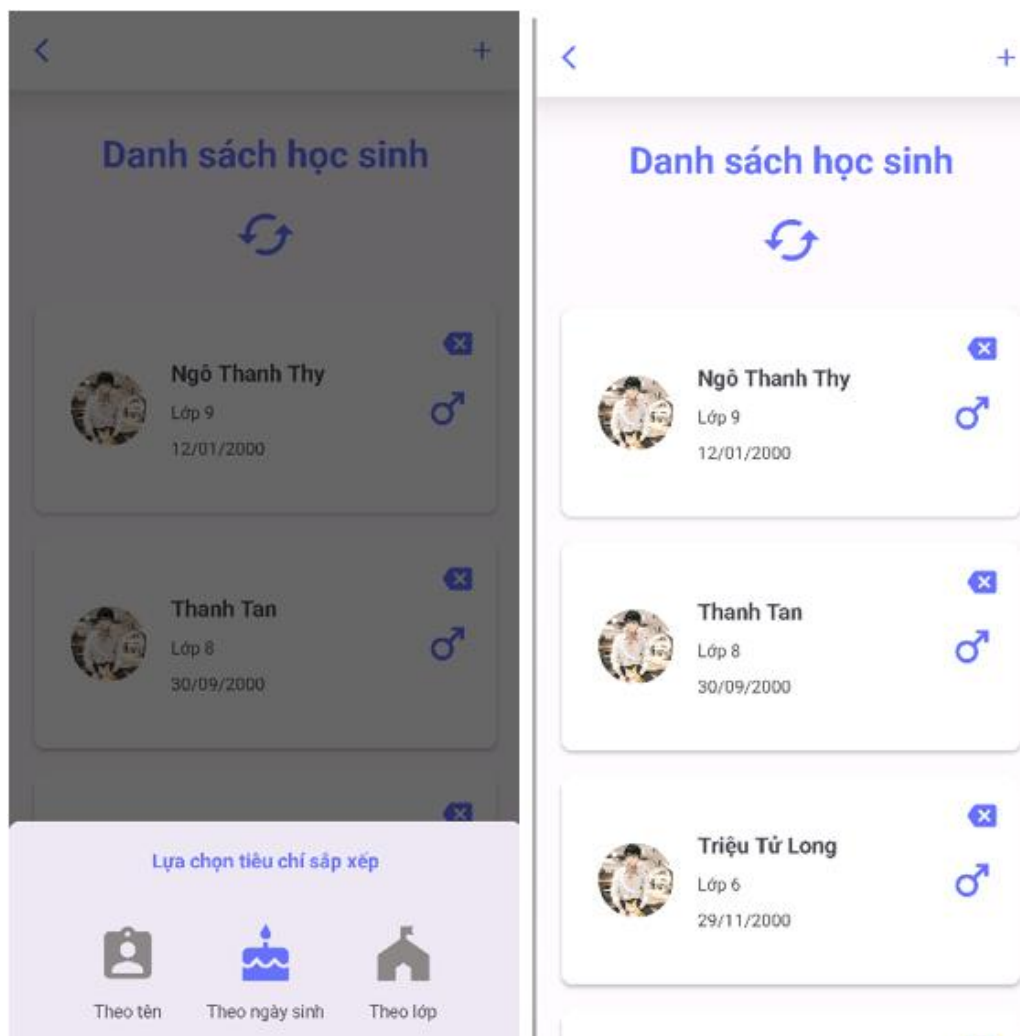
3.10 Demo chức năng cập nhật thông tin học sinh

The image displays three sequential mobile app screens for updating student information. Each screen has a header with a back arrow and a plus icon.

- Screen 1:** Shows the initial form for 'Ngô Thanh Thanh'. Fields include 'Tên' (Ngô Thanh Thanh), 'Ngày sinh' (12/01/2000), 'Giới tính' (Female selected), and 'Lớp' (Class 6 selected). Buttons at the bottom are 'Xem chi tiết các chứng chỉ' and 'Cập nhật thông tin'.
- Screen 2:** Shows the form for 'Ngô Thanh Thy'. Fields include 'Tên' (Ngô Thanh Thy), 'Ngày sinh' (12/01/2002), 'Giới tính' (Male selected), and 'Lớp' (Class 9 selected). Buttons at the bottom are 'Xem chi tiết các chứng chỉ' and 'Cập nhật thông tin'.
- Screen 3:** A confirmation screen showing a green checkmark and the message 'Cập nhật thông tin thành công' (Information updated successfully). Buttons at the bottom are 'Xem chi tiết các chứng chỉ' and 'Cập nhật thông tin'.

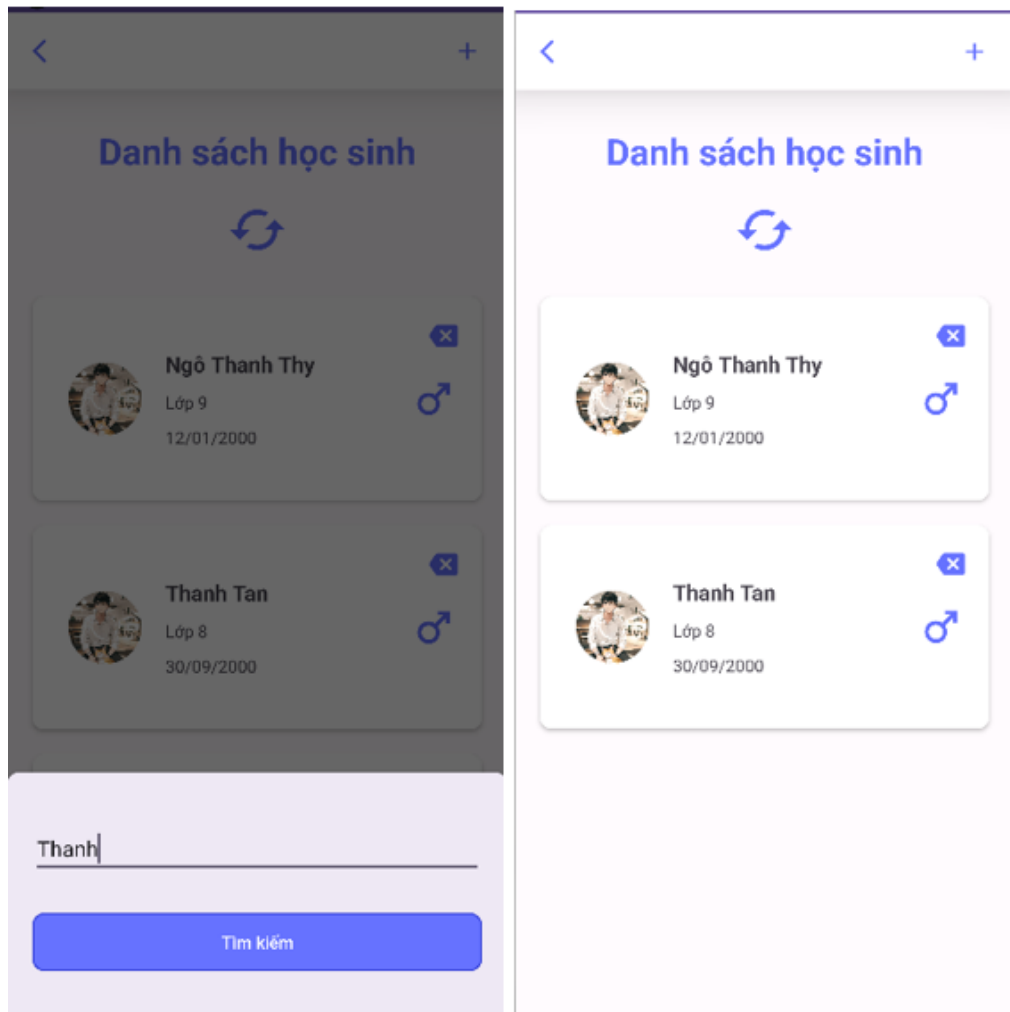
Hình 46 Demo chức năng cập nhật thông tin học sinh

3.11 Demo chức năng sắp xếp danh sách học sinh



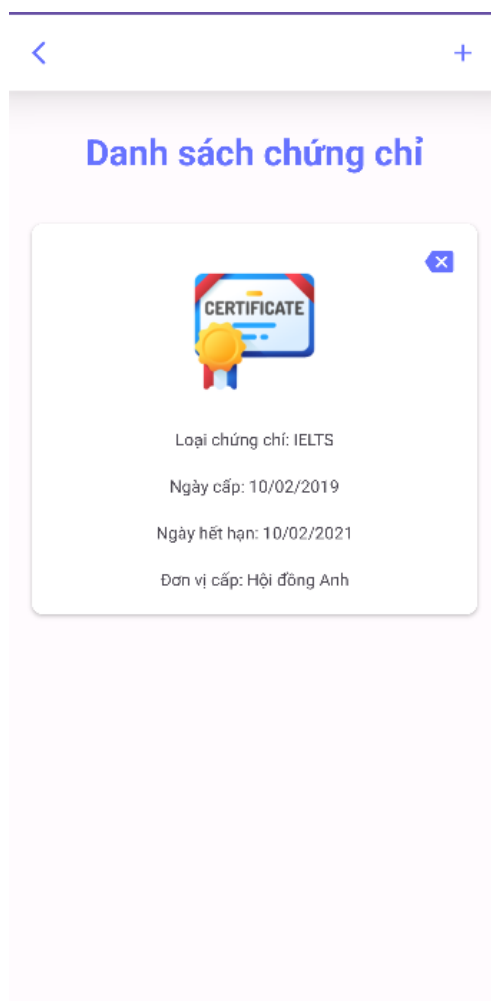
Hình 47 Demo chức năng sắp xếp học sinh

3.12 Demo chức năng tìm kiếm học sinh



Hình 48 Demo chức năng tìm kiếm học sinh

3.13 Demo chức năng xem danh sách chứng chỉ



Hình 49 Demo chức năng xem danh sách chứng chỉ

3.14 Demo chức năng thêm chứng chỉ

The image displays two mobile application screens for managing certificates.

Left Screen: Add Certificate Form

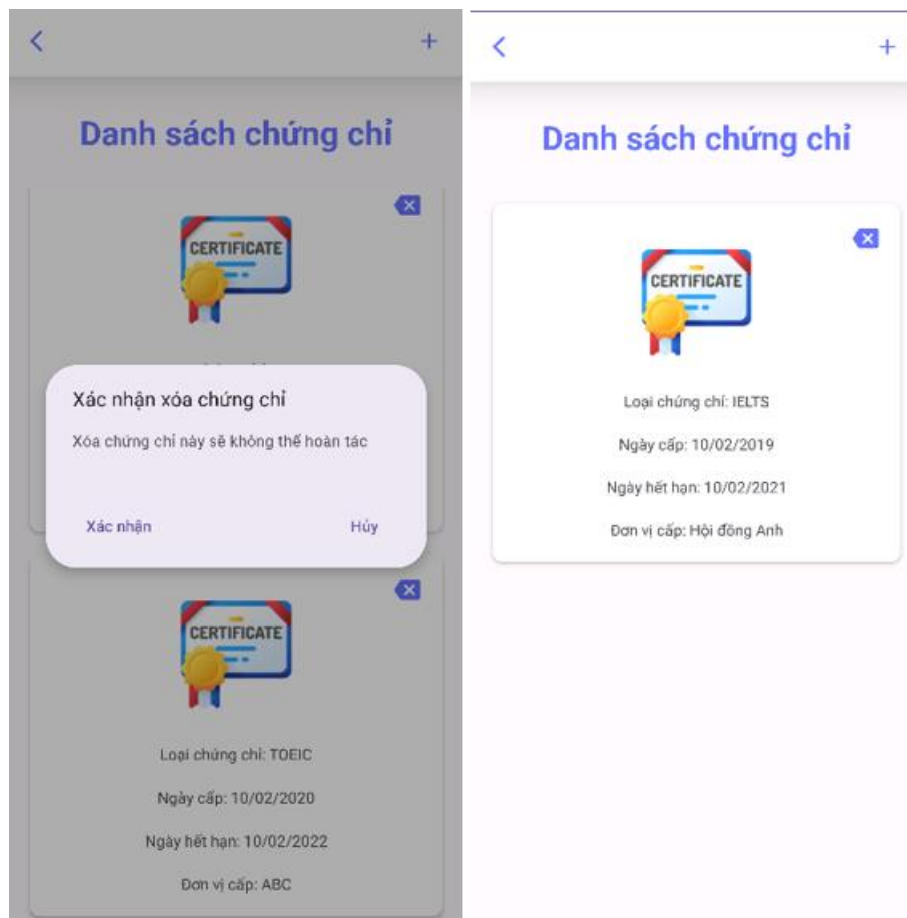
- Title:** Danh sách chứng chỉ
- Loại chứng chỉ:** Radio buttons for IELTS, TOEIC (selected), and MOS.
- Ngày thi:** Text input field containing 10/02/2020.
- Ngày hết hạn:** Text input field containing 10/02/2022.
- Đơn vị cấp bằng:** Text input field containing ABC.
- Action:** A blue button labeled "Thêm chứng chỉ".

Right Screen: Certificate List

- Title:** Danh sách chứng chỉ
- Item 1:**
 - Icon: Certificate with a gold medal.
 - Loại chứng chỉ: IELTS
 - Ngày cấp: 10/02/2019
 - Ngày hết hạn: 10/02/2021
 - Đơn vị cấp: Hội đồng Anh
- Item 2:**
 - Icon: Certificate with a gold medal.
 - Loại chứng chỉ: TOEIC
 - Ngày cấp: 10/02/2020
 - Ngày hết hạn: 10/02/2022
 - Đơn vị cấp: ABC

Hình 50 Demo chức năng thêm chứng chỉ

3.15 Demo chức năng xóa chứng chỉ



Hình 51 Demo chức năng xóa chứng chỉ

TÀI LIỆU THAM KHẢO

Tiếng Anh

Firebase Firestore (25/11/2023), Firestore Documentation, website:

<https://firebase.google.com/docs/firestore>

Firebase Storage (25/11/2023), Firebase Storage Documentation, website:

<https://firebase.google.com/docs/storage>