1. **Mô tả các chức năng bằng ngôn ngữ tự nhiên**

* Chọn nhân vật: ở màn hình đầu tiên người dùng sẽ thấy một hoạt ảnh nhỏ của nhân vật mặc định ở góc trên bên phải màn hình, khi nhấn vào hoạt ảnh sẽ hiển thị ra màn hình chọn nhân vật. Màn hình nhân vật sẽ hiển thị nhân tên và hoạt ảnh lớn của nhân vật đang được chọn, khi bấm nút chuyển nhân vật sẽ chuyển tên và hoạt ảnh đang hiển thị sang nhân vật khác (có hai nút chuyển phía trước và phía sau). Bấm nút chọn nhân vật, sẽ cập nhật nhân vật được chọn và chuyển về màn hình chính, màn hình chính sẽ hiển thị hoạt ảnh nhỏ của nhân vật được chọn ở góc trên bên phải màn hình.
* Lưu lịch sử chơi, hiển thị kết quả chơi: sau khi người chơi kết thúc trò chơi (bị hết mạng hoặc vượt qua tất cả các màn chơi), kết quả chơi sẽ được lưu lại vào thiết bị, màn hình sẽ chuyển hướng đến màn hình kết quả và hiển thị tên nhân vật đã chọn, điểm số đạt được, thời gian chơi (tính bằng giây) và thời gian kết thúc trò chơi.
* Cấu hình âm thanh game: khi ở màn hình chính, người chơi sẽ nhìn thấy biểu tượng loa ở góc trên bên phải màn hình, mặc định là âm thanh được bật, khi chạm vào biểu tượng âm thanh sẽ tắt/bật âm thanh. Khi âm thanh được tắt/bật, người dùng chuyển màn hình khác cũng đều nghe được âm thanh đang tắt/bật, ngay cả trong thế giới game. Trong thế giới game cũng có thể bật/tắt âm thanh bằng cách chạm vào biểu tượng loa.
* Xem lịch sử chơi: ở màn hình chính, người dùng bấm chọn nút lịch sử chơi, màn hình lịch sử chơi sẽ hiển thị lịch sử chơi, lịch sử chơi sẽ được sắp xếp từ mới nhất đến cũ nhất, mặc định lịch sử chơi sẽ được đọc từ cơ sở dữ liệu trên thiết bị, để tránh việc khi không có kết nối mạng vẫn có thể xem được lịch sử chơi, lịch sử chơi được đồng bộ 2 chiều giữa server với thiết bị khi đã đăng nhập tài khoản và có kết nối mạng.
* Đăng nhập: ở màn hình chính người dùng sẽ thấy ảnh đại diện mặc định và tên guest ở góc trên bên tay trái màn hình, khi bấm vào đây màn hình hiển thị 2 tuỳ chọn đăng nhập hoặc đăng ký. Người dùng chọn đăng nhập, màn hình đăng nhập sẽ hiện lên yêu cầu người dùng nhập email và mật khẩu của tài khoản đã đăng ký để đăng nhập, khi đăng nhập thành công màn hình đăng nhập sẽ chuyển hướng về màn hình chính đồng thời hiển thị tên, ảnh đại diện, id của tài khoản và tải lịch sử chơi của tài khoản đó về thiết bị.
* Đăng ký: ở màn hình chính người dùng sẽ thấy ảnh đại diện mặc định và tên guest ở góc trên bên tay trái màn hình, khi bấm vào đây màn hình hiển thị 2 tuỳ chọn đăng nhập hoặc đăng ký. Người dùng chọn đăng ký, màn hình đăng ký hiện lên yêu cầu người dùng nhập các thông tin cần thiết để đăng ký tài khoản (email, tên, mật khẩu, mật khẩu nhập lại) và người dùng có thể chọn ảnh đại diện cho tài khoản của mình hoặc không. Khi đăng ký thành công màn hình sẽ chuyển hướng đến màn hình chính đồng thời hiển thị id, tên, ảnh đại diện của tài khoản và xoá toàn bộ lịch sử chơi hiện tại trên thiết bị.
* Xem thông tin tài khoản: ở màn hình chính, người dùng sẽ thấy ảnh đại diện, id và tên của tài khoản, khi bấm bấm vào sẽ chuyển hướng đến màn hình hiển thị thông tin tài khoản bao gồm: ảnh đại diện, id tài khoản, tên tài khoản, email tài khoản và ngày tạo tài khoản.
* Cập nhật thông tin tài khoản: ở màn hình xem thông tin cá nhân, người dùng sẽ thấy biểu tượng hình bút chì ngay bên phải id của tài khoản, khi bấm vào màn hình thay đổi thông tin sẽ hiện ra cho phép người dùng cập nhật tên và ảnh đại diện cho tài khoản của họ. Mặc định màn hình sẽ hiển thị ảnh đại diện hiện tại và tên hiện tại của tài khoản. Lúc này, người dùng bấm vào ảnh đại diện của mình sẽ được chọn ảnh trên thiết bị để thay đổi, người dùng bấm vào textfield chứa tên tài khoản của mình để đổi tên. Khi hoàn tất việc thay đổi thông tin, bấm lưu thay đổi để cập nhật thay đổi, cập nhật thay đổi thành công màn hình sẽ chuyển hướng đến màn hình xem thông tin tài khoản và hiển thị thông tin đã được cập nhật.
* Đăng xuất: ở màn hình xem thông tin tài khoản người dùng sẽ thấy nút đăng xuất ở góc trên bên phải màn hình, khi bấm vào sẽ đồng bộ lịch sử chơi ở thiết bị lên server sau đó xoá toàn bộ lịch sử chơi khỏi thiết bị và đăng xuất tài khoản khỏi thiết bị.
* Đồng bộ dữ liệu lịch sử chơi: Chức năng này là một dịch vụ chạy nền, khi ứng dụng được mở sẽ lắng nghe trạng thái mạng, khi có kết nối sẽ đồng bộ lịch sử chơi từ thiết bị lên server và từ server về thiết bị. Chức năng này hoạt động kể cả khi mở ứng dụng mà không có kết nối mạng, dịch vụ vẫn sẽ lắng nghe trạng thái mạng, khi đột ngột có kết nối mạng sẽ thực hiện đồng bộ như trên.
* Xem bảng xếp hạng: ở màn hình chính, người dùng sẽ nhìn thấy biểu tượng xếp hạng ở góc trên bên trái màn hình, khi bấm vào biểu tượng, sẽ thực hiện tải dữ liệu sau đó thực hiện sắp xếp và xếp hạng người chơi, cuối cùng sẽ hiển thị 5 người chơi có kết quả cao nhất hiển thị từ cao đến nhấp. Xếp hạng được tính theo số lần chơi, trung bình điểm, trung bình thời gian. Nếu ai có số lần chơi nhiều hơn sẽ xếp cao hơn, nếu số lần chơi của hai người là bằng nhau thì người có trung bình điểm cao hơn sẽ xếp cao hơn, nếu số lần chơi và trung bình điểm của hai người bằng nhau thì người có trung bình thời gian ít hơn sẽ được xếp cao hơn.

1. **Thiết kế cơ sở dữ liệu**

* Thiết kế:

A diagram of a couple of letters

Description automatically generated with medium confidence

* Mô tả bảng Users:
* Mục đích: lưu trữ thông tin tài khoản sau khi đăng ký thành công.
* Cột id: làm khoá chính của bảng, có kiểu varchar, được tạo bằng uuid.
* Cột email: đại diện cho email tài khoản của người dùng, có kiểu là varchar.
* Cột name: đại diện cho tên của người dùng, có kiểu là varchar.
* Cột photo: đại diện cho ảnh đại diện của người dùng đã được mã hoá dưới dạng base64, có kiểu là varchar.
* Cột created\_at: đại diện cho ngày tạo tài khoản của người dùng, có kiểu là varchar.
* Mô tả bảng Scores:
* Mục đích: lưu lại điểm và các thông tin liên quan sau mỗi lần chơi của người dùng.
* Cột id: làm khoá chính của bảng, có kiểu varchar, được tạo bằng uuid.
* Cột character: đại diện cho nhân vật mà người chơi đã chọn trong lần chơi đó, có kiểu varchar.
* Cột created\_at: đại diện cho thời gian mà bản ghi được tạo ra trên thiết bị chơi, có kiểu varchar.
* Cột points: đại diện cho số điểm mà người dùng đạt được trong lần chơi đó, có kiểu bigint.
* Cột time: đại diện cho thời gian mà người dùng chơi trong lần chơi đó (tính bằng giây), có kiểu bigint.
* Cột user\_id: đại diện cho id của tài khoản người dùng, có kiểu varchar.
* Mối quan hệ: Users – Scores là mối quan hệ một – nhiều. Một tài khoản có lịch chơi của tài khoản đó với mỗi bản ghi được gọi là score.
* Sử dụng Firetore để triển khai cơ sở dữ liệu phía server dưới dạng collection có các bản ghi là document.
* Collection users:

Trường created\_at có kiểu dữ liệu là string

Trường email có kiểu dữ liệu là string

Trường name có kiểu dữ liệu là string

Trường photo có kiểu dữ liệu là string

Bản ghi mẫu

A screenshot of a computer

Description automatically generated

* Collection scores:

Trường character có kiểu dữ liệu là string

Trường created\_at có kiểu dữ liệu là string

Trường points có kiểu dữ liệu là number

Trường time có kiểu dữ liệu là number

Trường user\_id: có kiểu dữ liệu là string

Bản ghi mẫu

A screenshot of a computer

Description automatically generated

* Tạo luật cho firebase để bảo vệ và thiết lập quyền truy cập cho các collections:
* Với collection users sẽ cho phép đọc, không cho phép xoá, cho phép tạo nếu id tài khoản đã đăng ký là id của bản ghi, cho phép sửa nếu đã đăng nhập và id của tài khoản giống với id của bản ghi.
* Với collection scores sẽ cho phép đọc, không cho phép xoá và sửa, cho phép tạo nếu người dùng đã đăng nhập và user\_id của bản ghi giống với id của tài khoản.
* Ngoài ra, không cho phép tạo mới hay đọc các collection khác.
* Sử dụng sqflite để lưu trữ dữ liệu scores trên thiết bị:

A screenshot of a computer program

Description automatically generated

* Cột character có kiểu dữ liệu là TEXT, đại diện cho nhân vật đã sử dụng để chơi trong lần chơi đó.
* Cột created\_at có kiểu dữ liệu là TEXT, đại diện cho thời gian mà bản ghi được tạo.
* Cột id có kiểu dữ liệu là TEXT, đại diện cho id của bản ghi.
* Cột points có kiểu dữ liệu là INTEGER, đại diện cho điểm số đã đại được trong lần chơi đó.
* Cột time có kiểu dữ liệu là INTEGER, đại diện cho thời gian đã chơi trong lần chơi đó, tính bằng giây.
* Cột user\_id có kiểu dữ liệu là TEXT, đại diện cho id của tài khoản đã chơi trong lần chơi đó, nếu chơi dưới dạng khách, sẽ được lưu giá trị chuỗi là “null”.

1. **Thiết kế lớp thực thể**

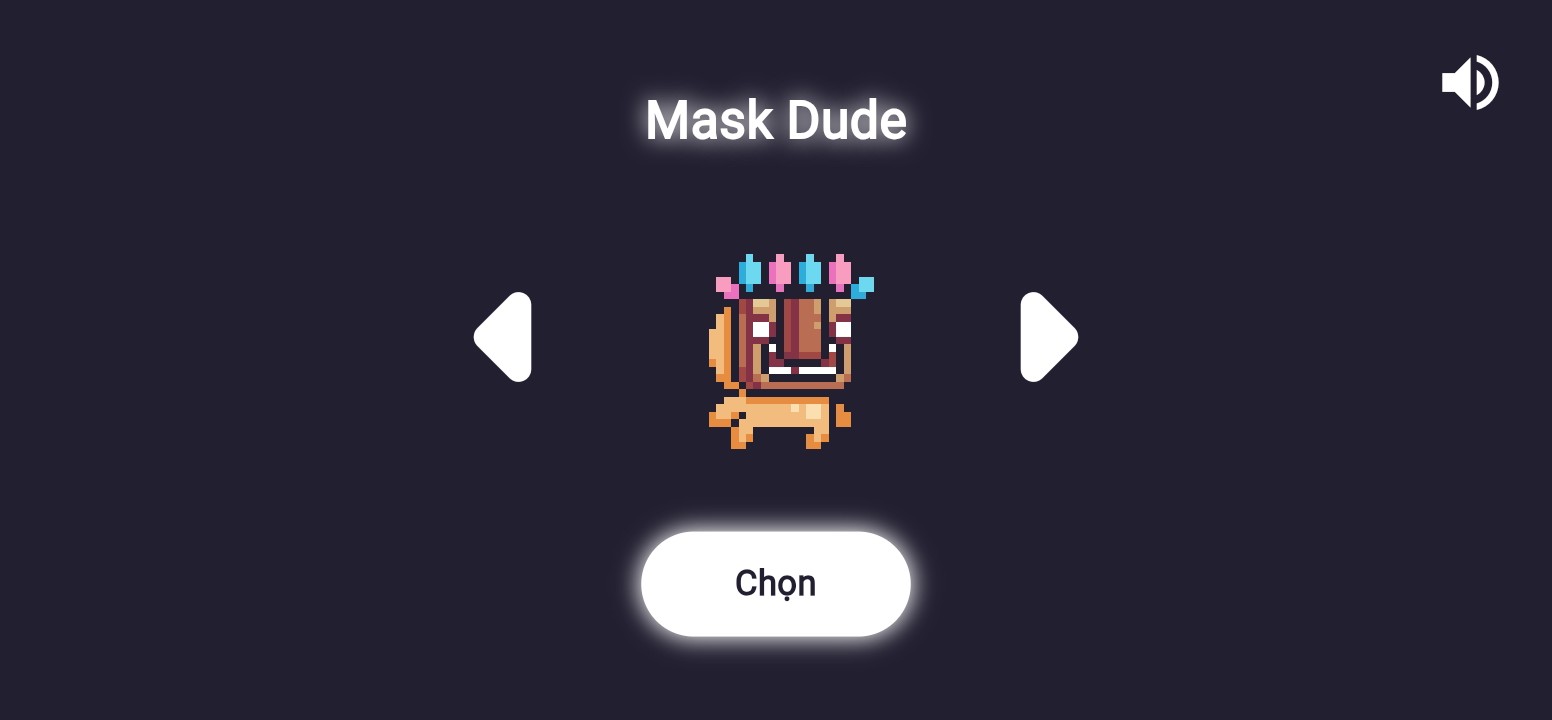
A blue screen with black text

Description automatically generated

* Thực thể UserModel có các trường:
* id: là id của tài khoản và bản ghi thông tin người dùng, có kiểu dữ liệu là String.
* name: là tên của người dùng, có kiểu dữ liệu là String.
* email: là email đã dùng để đăng ký tài khoản, có kiểu dữ liệu là String.
* photo: là ảnh đại diện của người dùng được mã hoá dưới dạng base64, có kiểu dữ liệu là String.
* createdAt: là thời gian mà người dùng đã đăng ký tài khoản, đã được chuyển dưới dạng DateTime.
* Thực thể ScoreModel có các trường:
* id: là id của bản ghi score, có kiểu dữ liệu là String, được tạo bằng uuid.
* points: là điểm đã đạt được trong lần chơi, có kiểu dữ liệu là int.
* time: là thời gian chơi trong một lần chơi, được để dưới dạng Duration.
* userId: là id của tài khoản người dùng, có kiểu dữ liệu String, nếu chơi ở chế độ khách, thực thể tạo ra sẽ có userId là chuỗi “null”.
* character: là nhân vật mà người dùng để chọn trong lần chơi, có kiểu dữ liệu là String.
* createdAt: là thời gian mà thực thể được tạo (khi trò chơi kết thúc), được để dưới dạng DateTime.
* Một UserModel có thể không có hoặc có nhiều ScoreModel, Một ScoreModel có thể không thuộc về (khi người dùng chơi dưới dạng khách) hoặc thuộc về một UserModel.
* Thực thể RankingModel có các trường:
* userId: là id của tài khoản người dùng, có kiểu dữ liệu String.
* userName: là tên của người dùng, có kiểu dữ liệu là String.
* avgPoints: là điểm trung bình của người dùng, có kiểu dữ liệu là double.
* avgTime: là thời gian trung bình mà người đó chơi, có kiểu dữ liệu là double.
* totalPlayed: là tổng số lần mà người đó chơi.

1. **Các chức năng**

* Chức năng chọn nhân vật:
* Giao diện chức năng:



* Biểu đồ lớp chi tiết:

A blue box with black text

Description automatically generated

StatefulWidget và State<T> là lớp thuộc thư viện material của Flutter framework.

Cubit<T> là lớp thuộc thư viện bên ngoài được thiết kế tối giản lại giúp hỗ trợ quản lý trạng thái dựa trên BLoC pattern.

SelectCharacterScreen là lớp kế thừa StatefulWidget chịu trách nhiệm khởi tạo màn hình chọn nhân vật và tạo ra instance của \_SelectCharacterScreenState để quản lý trạng thái.

\_SelectCharacterScreenState là lớp quản lý trạng thái kế thừa State<T> chịu trách nhiệm build widget, rebuild widget khi bấm chuyển nhân vật và gọi hàm xử lý logic từ PlayerCubit cập nhật nhân vật được chọn khi bấm chọn nhân vật.

PlayerCubit là lớp kế thừa từ Cubit<T> chịu trách nhiệm cho việc xử lý logic khi nhân vật được chọn, quản lý việc nhân vật nào đã được chọn ở nhiều màn khác nhau.

* Biểu đồ tuần tự chức năng:

A diagram of a diagram

Description automatically generated

1. Người dùng bấm chọn chuyển nhân vật.
2. Màn hình chọn nhân vật tạo một trạng thái.
3. Trạng thái được khởi tạo và widget được dựng.
4. Hiển thị widget lên giao diện người dùng.
5. Người dùng bấm chuyển nhân vật.
6. \_SelectCharacterScreen cập nhật trạng thái.
7. Trạng thái được cập nhật khiến widget được build lại.
8. Hiển thị nhân vật hiện tại được chuyển lên màn hình.
9. Người dùng bấm chọn nhân vật.
10. \_SelectCharacterScreen đọc PlayerCubit để gọi hàm xử lý sự kiện chọn nhân vật.
11. Phát ra trạng thái là tên nhân vật đã được chọn.
12. Màn hình được xoá khỏi stack.
13. Chuyển hướng đến màn hình chính.

* Chức năng lưu lịch sử chơi, hiển thị kết quả chơi:
* Giao diện chức năng:



* Biểu đồ lớp chi tiết:

A diagram of a computer

Description automatically generated

SharedPreferences là lớp thuộc thư viện bên ngoài hỗ trợ cho việc lưu dữ liệu dưới dạng key-value, lớp này được đăng ký trong GetIt dưới dạng singleton.

Database là lớp thuộc thư viện bên ngoài của sqflite hỗ trợ cho việc tạo và tương tác với cơ sở dữ liệu sqflite lớp này được đăng ký trong GetIt dưới dạng singleton.

GetIt là lớp thuộc thư viện bên ngoài hỗ trợ việc quản lý các instance bên trong ứng dụng, lớp này được sử dụng ở đây với mục đích lấy ra instance FirebaseAuth và Database đã được đăng ký bên trong GetIt.

Equatable là lớp thuộc thư viện bên ngoài hỗ trợ cho việc phân biệt sự khác nhau giữa các trạng thái được phát ra, từ đó Bloc dựa vào các trạng thái đó để build lại giao diện hay thực thi một số tác vụ.

GameResultState là lớp abstract kế thừa Equatable, là trạng thái chung cung cấp cho Cubit.

GameResultInitial là lớp kế thừa GameResultState, đóng vai trò là trạng thái khởi tạo.

GameResultSuccess là lớp kế thừa GameResultState, đóng vai trò là trạng thái lưu điểm thành công lên thiết bị, trạng thái này có đi kèm một message để cung cấp dữ liệu cho Bloc thực thi tác vụ.

GameResultFailure là lớp kế thừa GameResultState, đóng vai trò là trạng thái lưu điểm thất bại lên thiết bị.

Cubit<T> là lớp thuộc thư viện bên ngoài được thiết kế tối giản lại giúp hỗ trợ quản lý trạng thái dựa trên BLoC pattern, lớp này sử dụng GameResultState làm trạng thái trìu tượng.

GameResultCubit là lớp kế thừa Cubit<T> chịu trách nhiệm xử lý logic cho việc lưu lại điểm số vào cơ sở dữ liệu trên thiết bị và phát ra trạng thái dựa trên kết quả xử lý logic đó.

StatelessWidget là lớp được cung cấp bởi Flutter framework, lớp này cho phép kế thừa để tạo ra một widget không có trạng thái.

GameOverScreen kế thừa StatelessWidget, tạo ra widget không có trạng thái, hiển thị thông tin kết quả sau khi chơi.

* Biểu đồ tuần tự:

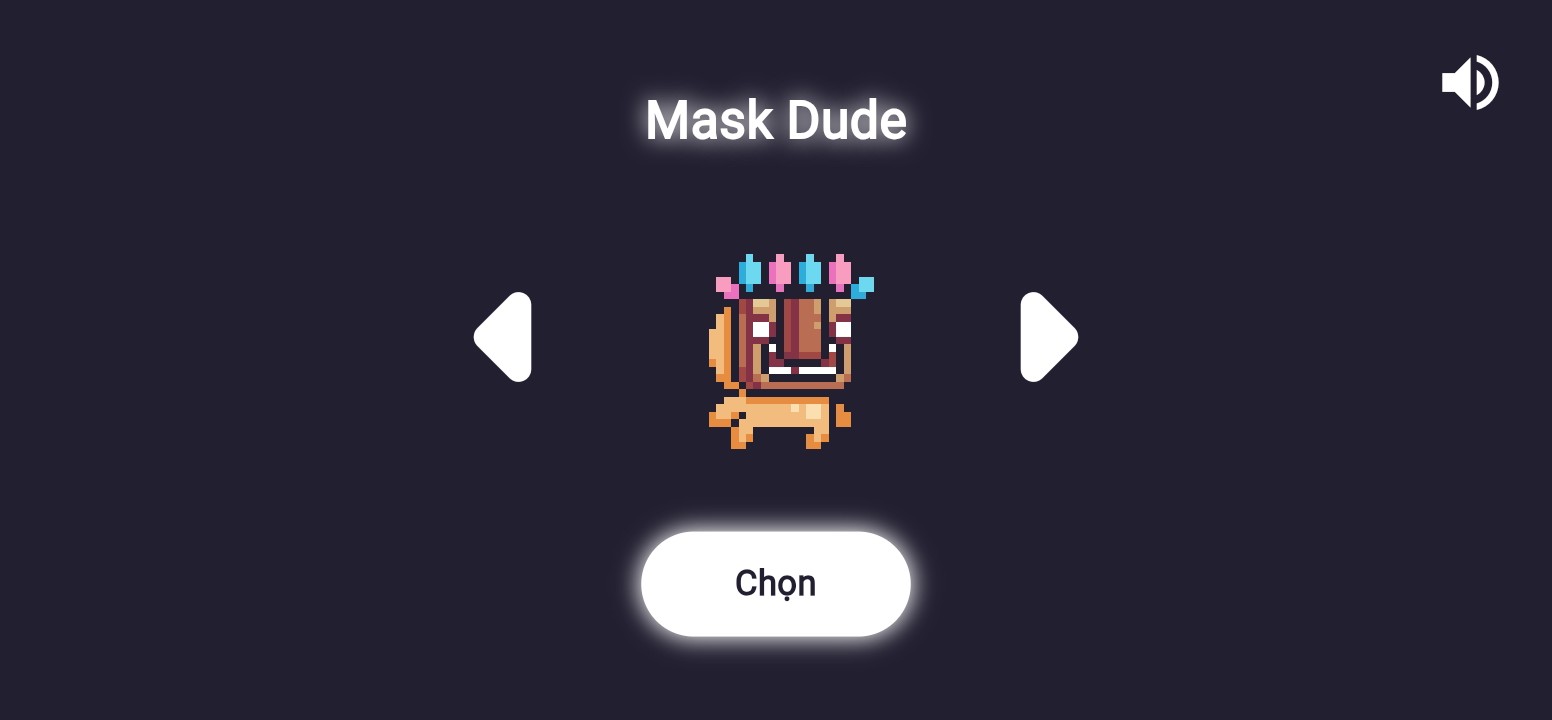
A diagram of a diagram

Description automatically generated

1. Người dùng kết thúc trò chơi (khi nhân vật hết mạng hoặc vượt qua toàn bộ màn chơi).
2. Sử dụng GameResultCubit để lưu kết quả chơi vào thiết bị.
3. GameResultCubit sử dụng SharedPreferences để kiểm tra có người dùng hiện tại hay không.
4. Trả về giá trị id người dùng hiện tại.
5. Nếu id tồn tại, lưu lại điểm cho người dùng hiện tại, nếu id là null lưu điểm cho khách.
6. Database thực hiện lưu điểm.
7. Trả về kết quả sau khi thực hiện insert.
8. GameResultCubit dựa vào kết quả phát ra trạng thái.
9. GameOverScreen dựng widget dựa theo trạng thái được phát ra.
10. Hiển thị giao diện lên màn hình.
11. Người dùng bấm chọn về trang chủ.
12. Gọi hàm xử lý sự kiện điều hướng về trang chủ.
13. Chuyển hướng màn hình về trang chủ và pop screen khỏi stack.

* Chức năng cấu hình âm thanh game:
* Giao diện chức năng: có thể bật/tắt âm thanh trên phạm vi toàn cục ứng dụng ở màn hình chính, màn chọn nhân vật và trong thế giới game.





* Biểu đồ lớp chi tiết:

A diagram of a diagram

Description automatically generated

Cubit<T> là lớp thuộc thư viện bên ngoài được thiết kế tối giản lại giúp hỗ trợ quản lý trạng thái dựa trên BLoC pattern.

SoundCubit là lớp kế thừa Cubit<T> chịu trách nhiệm xử lý logic cho việc bật/tắt âm thanh trên phạm vi toàn cục của ứng dụng, cubit này chỉ có thể phát ra hai giá trị là true hoặc false thể hiện sự bật hoặc tắt âm thanh tương ứng.

StatefulWidget và State là lớp thuộc thư viện material của Flutter framework, lớp này cho phép kế thừa để tạo ra một widget có trạng thái.

SelectCharacterScreen và StartScreen là hai lớp kế thừa từ StatefulWidget chịu trách nhiệm khởi tạo màn hình từ instance \_SelectCharacterScreenState và instance \_StartScreenState tương ứng, hai màn hình này đều chứa nút tắt bật âm thanh.

State<T> là lớp thuộc thư viện material của Flutter framework, kế thừa lớp này sẽ tạo ra được một lớp State của một widget T tương ứng.

\_SelectCharacterScreenState và \_StartScreenState là hai lớp quản lý trạng thái kế thừa State<T> chịu trách nhiệm build widget, rebuild widget, khi có sự thay đổi tắt/bật âm thanh tương ứng.

* Biểu đồ tuần tự chức năng:

A screen shot of a computer

Description automatically generated

1. Người dùng bấm vào biểu tượng âm thanh.
2. Đọc SoundCubit từ context và gọi hàm toggleSound của SoundCubit.
3. Thực hiện tắt/bật âm thanh trên phạm vi toàn cục của ứng dụng.
4. Phát ra trạng thái tương ứng với âm thanh hiện tại đang được bật/tắt.
5. Xây lại biểu tượng âm thanh trên giao diện tương ứng với trạng thái âm thanh hiện tại đang được bật/tắt.

* Chức năng xem lịch sử chơi:
* Giao diện chức năng:



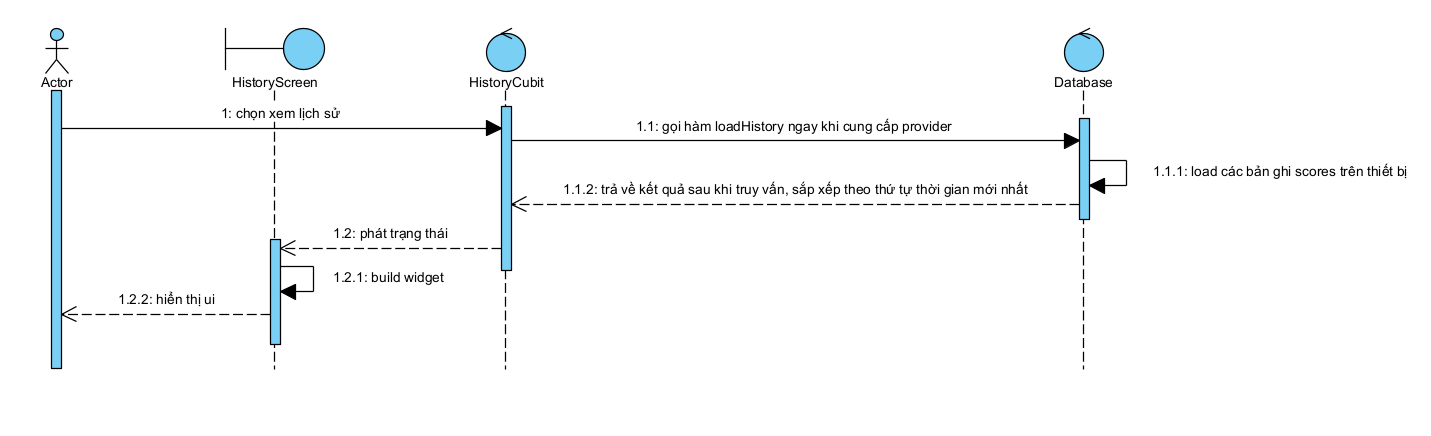
* Biểu đồ lớp chi tiết:

A diagram of a computer

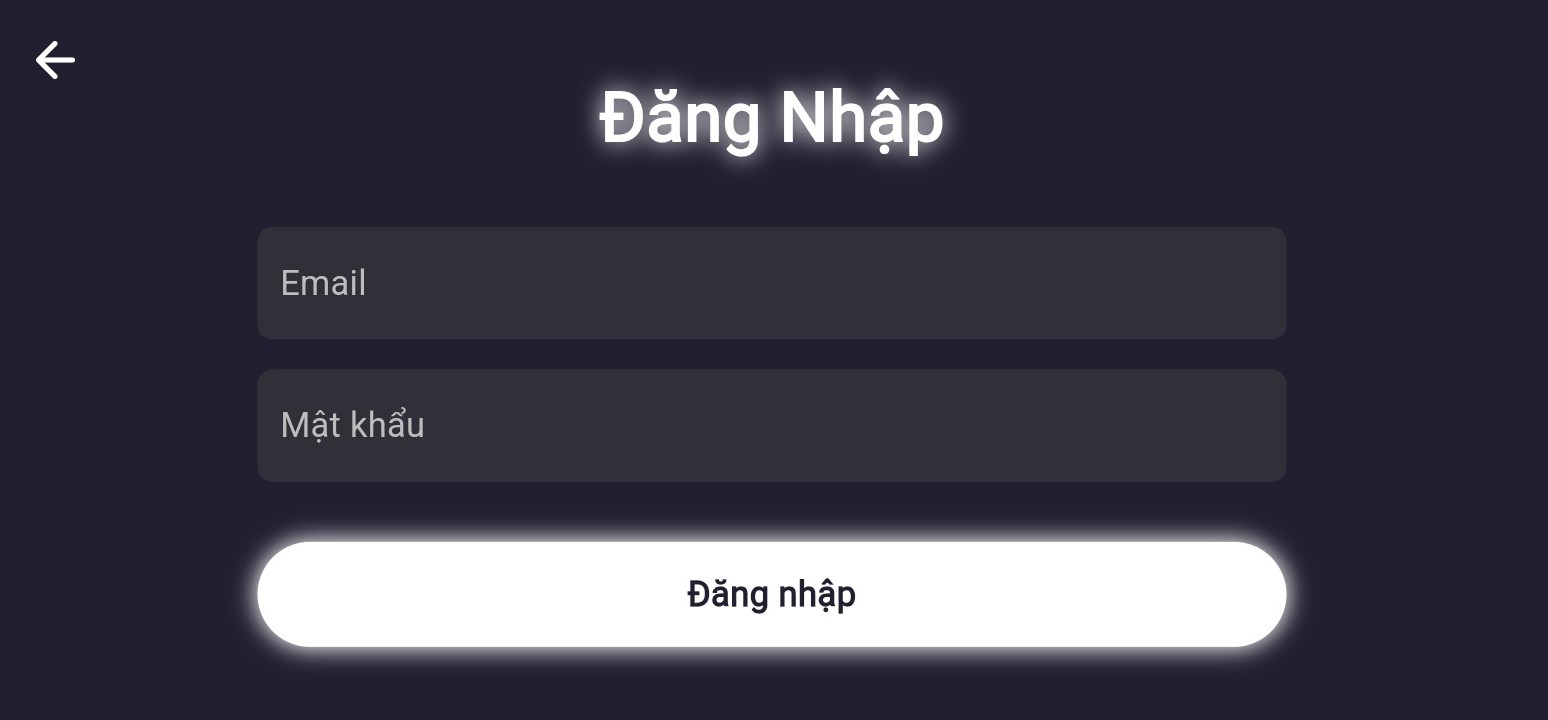
Description automatically generated

Equatable là lớp thuộc thư viện bên ngoài hỗ trợ cho việc phân biệt sự khác nhau giữa các trạng thái được phát ra, từ đó Bloc dựa vào các trạng thái đó để build lại giao diện hay thực thi một số tác vụ.

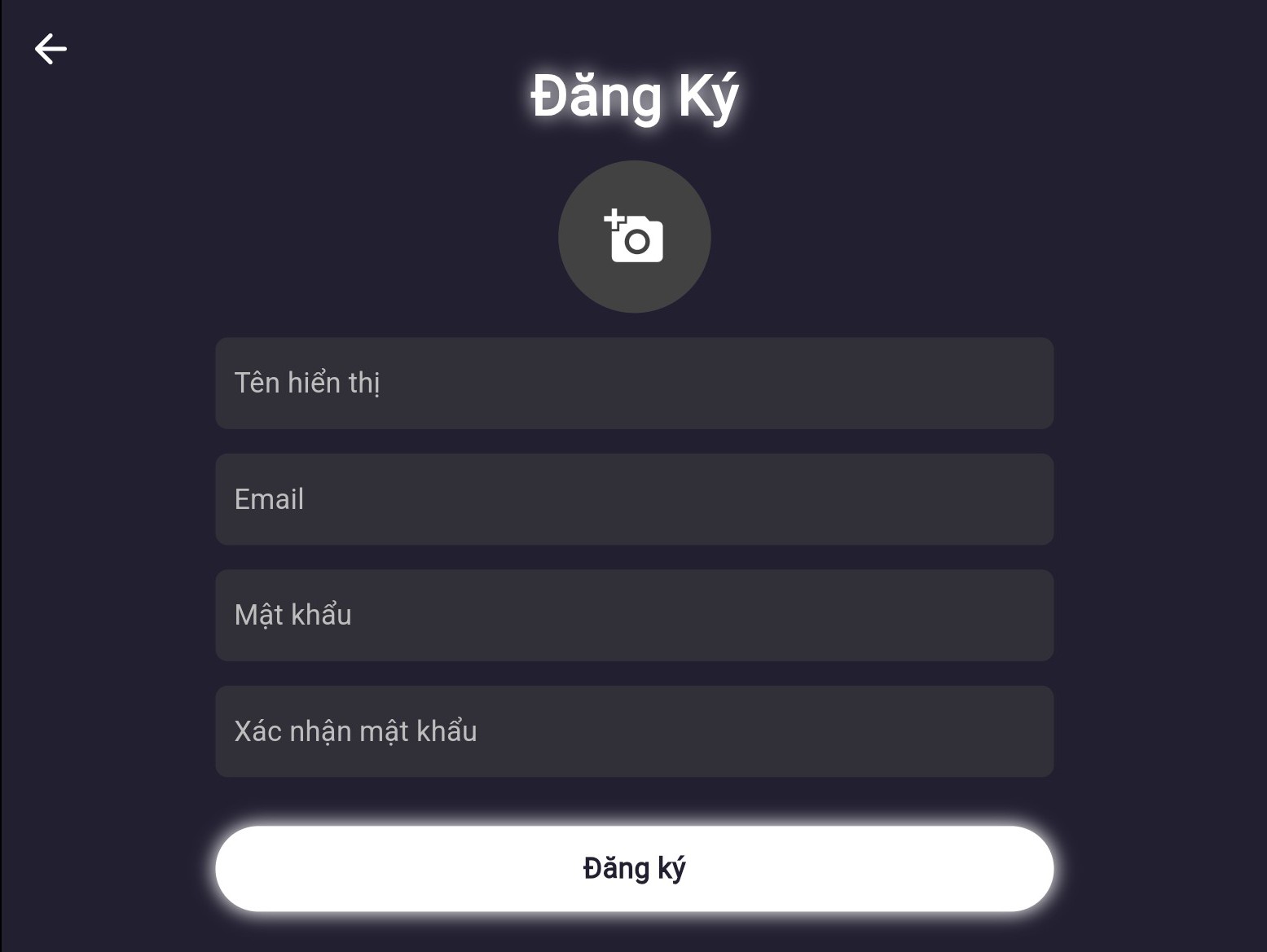
* Biểu đồ tuần tự chức năng:



* Chức năng đăng nhập:
* Giao diện chức năng:



* Biểu đồ lớp chi tiết:
* Biểu đồ tuần tự chức năng:
* Chức năng đăng ký:
* Giao diện chức năng:



* Biểu đồ lớp chi tiết:

A diagram of a computer

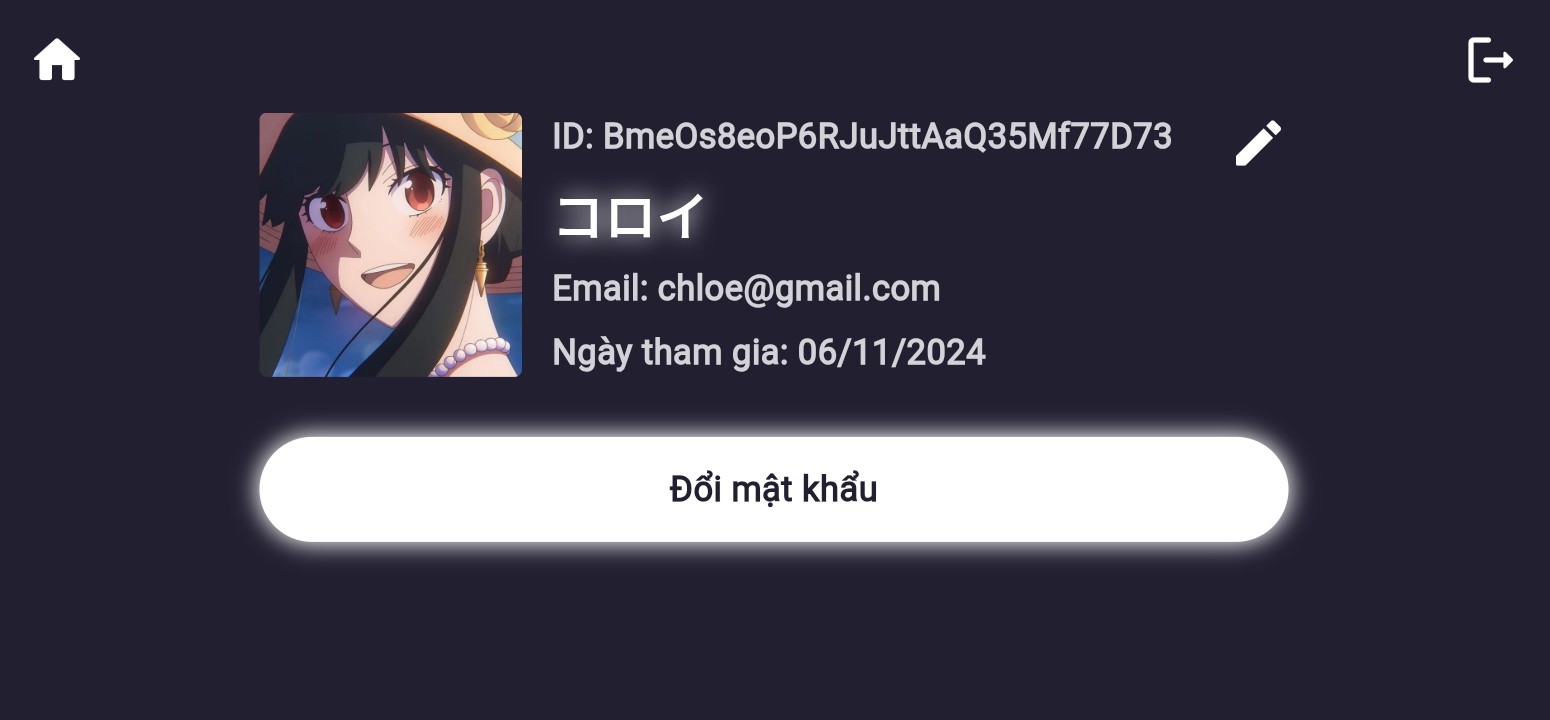
Description automatically generated

* Biểu đồ tuần tự chức năng:

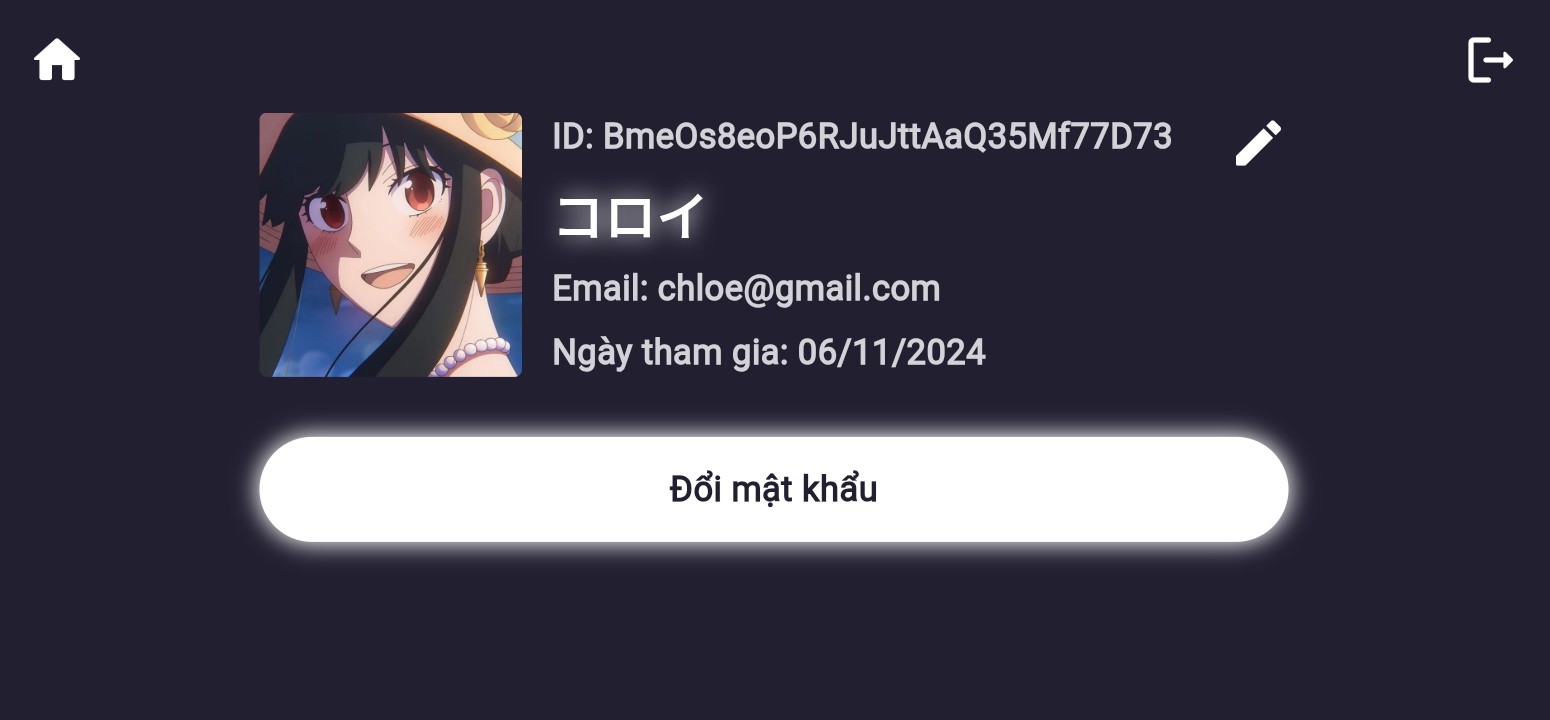
A diagram of a project

Description automatically generated

* Chức năng xem thông tin tài khoản:
* Giao diện chức năng:



* Biểu đồ lớp chi tiết:
* Biểu đồ tuần tự chức năng:
* Chức năng cập nhật thông tin tài khoản
* Giao diện chức năng:
* Biểu đồ lớp chi tiết:
* Biểu đồ tuần tự chức năng:
* Chức năng đăng xuất:
* Giao diện chức năng:



* Biểu đồ lớp chi tiết:
* Biểu đồ tuần tự chức năng:
* Chức năng đồng bộ lịch sử chơi:
* Giao diện chức năng: chức năng là một dịch vụ chạy ngầm, không có giao diện.
* Biểu đồ lớp chi tiết:
* Biểu đồ tuần tự chức năng:
* Chức năng xem bảng xếp hạng:
* Giao diện chức năng:





* Biểu đồ lớp chi tiết:
* Biểu đồ tuần tự chức năng: