**MỤC LỤC**

[**CHƯƠNG I : TỔNG QUAN** 2](#_Toc12195581)

[**1.1 Giới thiệu tổng quan và lý do chọn đề tài.** 3](#_Toc12195582)

[**1.2 Mục tiêu của đề tài** 3](#_Toc12195583)

[**1.2.1 Lý thuyết** 3](#_Toc12195584)

[**1.2.2 Thực tiễn** 3](#_Toc12195585)

[**1.3 Các bước nghiên cứu** 3](#_Toc12195586)

[**1.4 bố cục đề tài** 3](#_Toc12195587)

[**CHƯƠNG II: CƠ SỞ LÝ THUYẾT** 4](#_Toc12195588)

[**2.1.Hệ điều hành android.** 4](#_Toc12195589)

[**2.1.1 Đặc Điểm** 6](#_Toc12195590)

[**2.1.2. So Sánh với các hệ điều hành còn lại** 6](#_Toc12195591)

[**2.2.Kiến trúc và các thành phần** 7](#_Toc12195592)

[**2.2.1 Kiến trúc tổng quoát.** 8](#_Toc12195593)

[**2.2.2 Hệ điều hành** 8](#_Toc12195594)

[**2.2.3 Các thành phần của một ứng dụng android.** 10](#_Toc12195595)

[**2.3 Công nghệ Pattern** 16](#_Toc12195596)

[**2.3.1 Style Convention** 16](#_Toc12195597)

[**2.3.2 Design pattern** 17](#_Toc12195598)

[**2.3.2 Animation** 18](#_Toc12195599)

[**CHƯƠNG III : PHÂN TÍCH THIẾT KẾ** 19](#_Toc12195600)

[**3.1. Xác định yêu cầu** 19](#_Toc12195601)

[**3.1.1 Chức năng chính** 20](#_Toc12195602)

[**3.1.2 Mô hình hoạt động MVP pattern** 20](#_Toc12195603)

[**3.1.3 Giới Thiệu Về Firebase** 25](#_Toc12195604)

[**CHƯƠNG IV : GIỚI THIỆU ỨNG DỤNG** 29](#_Toc12195605)

[**4.1 Giới thiệu app** 29](#_Toc12195606)

[**4.1.1 Danh Sách Màn Hình** 29](#_Toc12195607)

[**4.1.2 Sign In Screen** 29](#_Toc12195608)

[**4.1.3 Vocabulary Screen** 31](#_Toc12195609)

[**4.1.4 Full Quiz Screen** 32](#_Toc12195610)

[**4.1.5 About Screen** 35](#_Toc12195611)

[**CHƯƠNG V : KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN** 36](#_Toc12195612)

[**5.1. Kết Luận** 36](#_Toc12195613)

[**5.2. Hướng Phát Triển** 37](#_Toc12195614)

**CHƯƠNG I : TỔNG QUAN**

**1.1 Giới thiệu tổng quan và lý do chọn đề tài.**

Android đang là hệ điều hành khá phổ biến, để hiểu hơn về lập trình android thì cần xây dựng ứng dụng để hiểu nó. Phần mềm ghép tranh là app khá hay để hiểu thêm về lập trình android.

**1.2 Mục tiêu của đề tài**

Các chức năng của phần mềm:

Hiển thị màn hình ghép tranh, cho phép người dùng thay đổi tranh, xem ảnh gốc, reset tranh, vào có thể chọn ảnh từ thư viện.

**1.2.1 Lý thuyết**

* Nghiên cứu lập trình trên hệ điều hành android
* Nghiên cứu ngôn ngữ lập trình java.
* Nghiên cứu xây dựng giao diện với xml trong android.
* Nghiên cứu mô hình MVP Framework.
* Nghiên cứu Design Pattern.
* Nghiên cứu Style Convention.

**1.2.2 Thực tiễn**

* Xây dựng ứng dụng ghép tranh trên hệ điều hành Android

**1.3 Các bước nghiên cứu**

* Phân tích yêu cầu.
* Lựa chọn công nghệ.
* Nghiên cứu cơ sở lý thuyết của công nghệ đã chọn.
* Áp dụng lý thuyết vào xây dựng ứng dụng thực tiễn.
* Kiểm tra, tham khảo các ứng dụng khác để tối ưu hóa ứng dụng.

**1.4 bố cục đề tài**

* Cơ sở lý thuyết.
* Phân tích thiết kế.
* Giải pháp xây dựng.
* Triển khai ứng dụng.

**CHƯƠNG II: CƠ SỞ LÝ THUYẾT**

**2.1.Hệ điều hành android.**

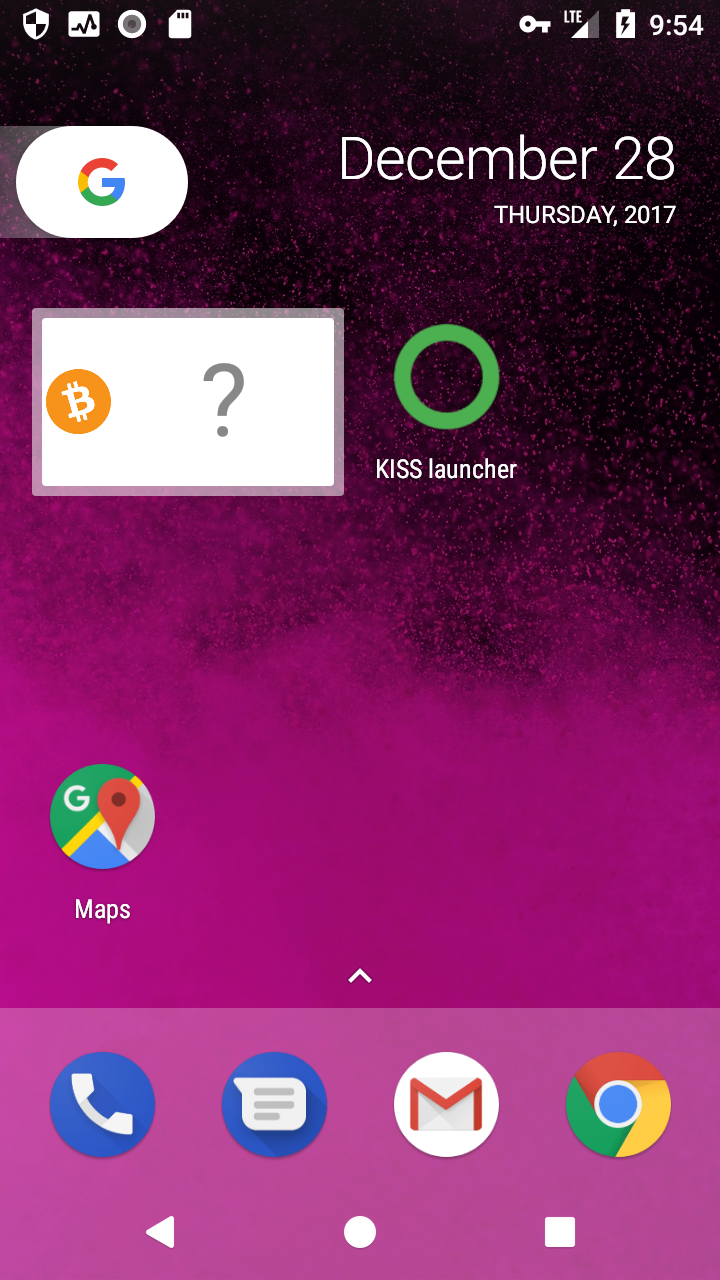
****

Figure 1Hệ điều hành Android

Android là một hệ điều hành dự trên nền tảng Linux được thiết kế dành cho các thiết bị di động có màn hình cảm ứng như điện thoại thông mình và máy tính bảng. Ban đầu, Android được phát triển bởi Tổng công ty Android, với sự hỗ trợ tài chính từ Google và sau này được chính Google mua lại vào năm 2005. Android ra mắt vào năm 2007 cùng với tuyên bố thành lập Liên minh thiết bị cầm tay mở: một hiệp hội gồm các công ty phần cứng, phần mềm, và viễn thông với mục tiêu đẩy mạnh các tiêu chuẩn mở cho các thiết bị di động. Chiếc điện thoại đầu tiên chạy Android được bán vào tháng 10 năm 2008.

Android có mã nguồn mở và Google phát hành mã nguồn theo Giấy phép Apache. Chính mã nguồn mở cùng với một giấy phép không có nhiều rang buộc đã cho phép các nhà phát triển thiết bị, mạng di động và các lập trình viên nhiệt huyết được điều chỉnh và phân phối Android một cách tự do. Ngoài ra, Android còn có một cộng đồng lập trình viên đông đảo chuyên viết các ứng dụng để mở rộng chức năng của thiết bị, bằng một loại ngôn ngữ lập trình Java có sửa đổi. Vào tháng 10 năm 2012, có khoảng 700.000 ứng dụng trên Android, và số lượt tải ứng dụng từ Google Play, của hang ứng dụng của Android, ước tính khoảng 25 tỷ lượt.

Những yếu tố này đã giúp Android trở thành nền tảng điện thoại thông mình phổ biến nhất thế giới, vượt qua Symbian vào quý 4 năm 2010, và được các công ty công nghệ lựa chọn khi họ cần một hệ điều hành không nặng nề, có khả năng tinh chỉnh, và giá rẻ chạy trên các thiết bị công nghệ cao thay vì tạo dựng từ đầu. Bản chất mở của Android cũng khích lệ một đội ngũ đông đảo lập trình viên và những người đam mê sử dụng mã nguồn mở để tạo ra những dự án do cộng đồng quản lý. Những dự án này bổ sung các tính năng cao cấp cho những người dùng thích tìm tòi hoặc đưa Android vào các thiết bị ban đầu chạy hệ điều hành khác.

Android chiếm 87,7% thị phần điện thoại thông minh trên toàn thế giới vào thời điểm quý 2 năm 2017, với tổng cộng 2 tỷ thiết bị đã được kích hoạt và 1,3 triệu lượt kích hoạt mỗi ngày. Sự thành công của hệ điều hành cũng khiến nó trở thành mục tiêu trong các vụ kiện liên quan đến bằng phát minh, góp mặt trong cái gọi là "cuộc chiến điện thoại thông minh" giữa các công ty công nghệ.

Được xây dựng trên một nền tảng mở, và một bộ thư viện đa năng, mạnh mẽ với nguyên lý mở, Android đã nhanh chóng được cộng đồng lập trình viên di động hưởng ứng mạnh mẽ. Nền tảng Android tích hợp nhiều tính năng nổi bật:

* Android là một hệ điều hành nhân Linux, đảm bảo sự tương tác với các phần cứng, quản lý bộ nhớ, điều khiển các tiến trình tối ưu cho các thiết bị di động.
* Bộ ứng dụng khung cho phép sử dụng lại và thay thế các thành phần riêng lẻ.
* Máy ảo Dalvik được tối ưu cho các thiết bị di động, chạy các ứng dụng lập trình trên ngôn ngữ Java.
* Các thư viện cho phát triển ứng dụng mã nguồn mở bao gồm SQLite, WebKit, OpenGL và trình quản lý đa phương tiện.
* Hỗ trợ các chuẩn đa phương tiện phổ biết, trên nền GSM, Bluetooth EDGE, 3G và Wifi.
* Hỗ trợ Camera, GPS, la bàn, máy đo gia tốc…
* Bộ phát triển ứng dụng SDK đầy đủ gồm thiết bị giả lập, công cụ sửa lỗi, tích hợp với Android Studio.

Android cung cấp một tập hợp đầy đủ các phần mềm cho thiết bị di động bao gồm: hệ điều hành, các khung ứng dụng và các ứng dụng cơ bản.

**2.1.1 Đặc Điểm**

* Tính mở Android được xây dựng từ dưới đi lên cho phép người phát triển tạo các ứng dụng di động hấp dẫn với đầy đủ các điểm mạnh của các thiết bị cầm tay hiện có. Android hoàn toàn mở, một ứng dụng có thể gọi tới bất kể một chức năng lõi của điện thoại như tạo cuộc gọi, gửi tin nhắn hay sử dụng máy ảnh, cho phép người phát triển tạo phong phú hơn, liên kết hơn các tính năng cho người dùng. Android được xây dựng trên nhân Linux mở. Thêm nữa, nó sử dụng một máy ảo mà đã được tối ưu hóa bộ nhớ và phần cứng với môi trường di động. Android là một mã nguồn mở, nó có thể được mở rộng để kết hợp tự do giữa các công nghệ nổi trội. Nền tảng này sẽ tiếp tục phát triển bởi cộng đồng phát triển để tạo ra các ứng dụng di động hoàn hảo.
* Tính ngang hàng của các ứng dụng Với Android, không có sự khác nhau giữa các ứng dụng điện thoại cơ bản với ứng dụng của bên thứ ba. Chúng được xây dựng để truy cập như nhau tới một loạt các ứng dụng và dịch vụ của điện thoại. Với các thiết bị được xây dựng trên nền tảng Android, người dùng có thể đáp ứng đầy đủ các nhu cầu mà họ thích. Chúng ta có thể đổi màn hình nền, kiểu gọi điện thoại, hay bất kể ứng dụng nào. Chúng ta thậm chí có thể hướng dẫn điện thoại chỉ xem những ảnh mình thích.
* Dễ xây dựng ứng dụng Android cung cấp bộ thư viện giao diện lập trình ứng dụng đồ sộ và các công cụ để viết các ứng dụng phức tạp. Ví dụ, Android có thể cho phép người phát triển biết được vị trí của thiết bị và cho phép các thiết bị giao tiếp với nhau để có thể tạo nên mạng xã hội chia sẻ ngang hàng rộng khắp. Thêm nữa, Android còn bao gồm một bộ công cụ đầy đủ giúp cho việc phát triển trở nên dễ dàng.

**2.1.2. So Sánh với các hệ điều hành còn lại**

*Bảng 2.1 So Sánh Các Hệ Điều Hành Điện Thoại*

|  |  |  |
| --- | --- | --- |
| **Tên** | **Ưu điểm** | **Nhược điểm** |
| **Android** | Google Android là nền tảng mở, cho phép người dùng có thể tùy biến nền tảng theo ý thích, hơn nữa lại có một Liên minh thiết bị cầm tay mở hậu thuẫn, Google Android đang là đối thủ xứng tầm của iPhone của Apple. Google đang tích cực mở rộng cộng đồng phát triển các ứng dụng cho Android. Bộ công cụ phát triển phần mềm (SDK) đầy đủ, hỗ trợ đa nền (Linux, Windows hay Mac OS) do chạy trên máy ảo Java. Thư viện ngày càng hoàn thiện, dễ dàng cho người lập trình. | Hệ điều hành phân mảnh, không thống nhất trên các thiết bị, giới hạn về độ 'mở' và nhiều lỗ hổng bảo mật là những yếu điểm của Android OS |
| **Windows Phone** | Có thư viện API khá giống với API trên Win32, các công cụ hỗ trợ lập trình đầy đủ với Visual Studio, điều này làm cho những người phát triển trên Win32 không mất công tìm hiểu lại các API và các công cụ lập trình | Sự có mặt của iPhone và Android là hai trở ngại lớn với Windows Mobile. Hai nền tảng này đang hoàn thiện và được người dùng rất ưa chuộng. |
| **iPhone** | Màn hình cảm ứng đa điểm: iPhone sử dụng hoàn toàn bằng cảm ứng và không sử dụng các nút. Với iPhone ta có thể điều khiển trên màn hình kể cả việc trượt của các ngón tay. Ta có thể phóng to ảnh bằng cách trượt hai ngón tay ra xa và thu nhỏ bằng cách ngược lại.  Bộ cảm nhận gia tốc: Những phản ứng nhanh chóng của bộ cảm nhận gia tốc thay đổi độ phân giải màn hình từ dọc sang ngang tự động khi ta đặt điện thoại nằm ngang. Điều này làm sinh động thêm cho các trò chơi. Âm thanh, hình ảnh hoàn hảo. | Việc lập trình trên cho iPhone phải thực hiện trên hệ điều hành Mac, do đó không phải ai cũng có thể lập trình cho iPhone. Hơn thế, nếu muốn đưa chương trình ra máy thật người lập trình phải trả một khoản phí lập trình, điều này làm giảm tính cạnh tranh so với các đối thủ khác. |

**2.2.Kiến trúc và các thành phần**

**2.2.1 Kiến trúc tổng quoát.**

Android bao gồm bốn thành phần sau :

* Hệ điều hành.
* Thư Viện và các giao diện lập trình ứng dụng.
* Khung ứng dụng.
* Ứng dụng.

**2.2.2 Hệ điều hành**

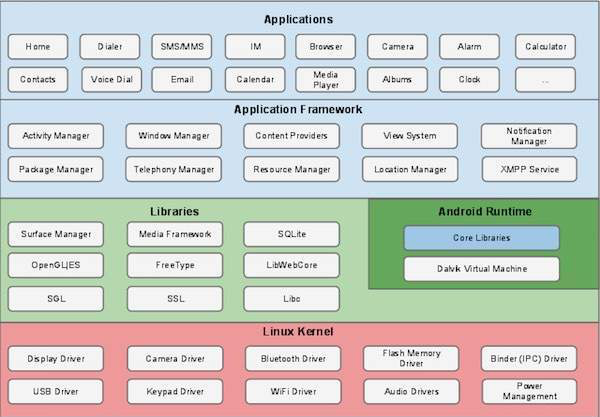
****

Figure 2 Kiến trúc tổng quát

Android sử dụng nhân Linux 2.6 làm nhân cho các dịch vụ hệ thống như bảo mật, quản lý bộ nhớ, quản lý tiến trình (xử lý tiến trình, đa luồng), ngăn xếp mạng và trình điều khiển thiết bị (giao tiếp USB, giao tiếp hồng ngoại, không đây, v.v…). Nhân Linux này cũng có vai trò như một lớp trừu tượng giữa phần cứng và phần mềm.

Tuy được phát triển dựa vào nhân linux nhưng thực ra nhân linux đã được nâng cấp và sửa đổi rất nhiều để phù hợp với tính chất của những thiết bị cầm tay như hạn chế về bộ vi xử lý, dung lượng bộ nhớ, kích thước màn hình, nhu cầu kết nối mạng không dây.

2.2.2.1 Các thành phần của nhân linux:



* Display Driver : Điều khiển việc hiển thị lên màn hình cũng như thu nhận những điều khiển của người dùng lên màn hình ( di chuyển, cảm ứng…).
* Camera Driver : điều khiển hoạt động của camera, nhận luồng dữ liệu từ camera về.
* Bluetooth Driver : Điều khiển thiết bị phát và thu sóng Bluetooth.
* USB Driver : Điều khiển bàn phím.
* Wifi Driver : Chịu trách nhiệm về việc thu phát sóng wifi.
* Audio Driver : Điều khiển các bộ thu phí phát âm thanh, giải mã các tính hiệu dạng audio thành tín hiệu số và ngược lại.
* Power Management : Giám sát việc tiêu thụ điện năng.
* M-system Driver : Quản lý việc đọc ghi… lên các thiết bị nhớ như thẻ SD, flash.
* Binder IPC Driver : Chịu trách nhiệm về việc kết nối và liên lạc với mạng vô tuyến như CDMA, GSM, 3G, 4G, E để đảm bảo những chức năng truyền thông được thực hiện.

2.2.2.2 Library



Android cung cấp một số các APIs cho phát triển ứng dụng. Danh sách

các API cơ bản sau được cung cấp bởi tất cả các thiết bị trên nền Android:

* *android.util*: Gói tiện ích cơ bản bao gồm nhiều lớp mức thấp như là các lớp quản lý (List, Stack…) lớp xử lý chuỗi, lớp xử lý XML.
* android.graphics: Cung cấp các lớp đồ họa mức thấp thực hiện các chức năng đồ họa, màu, vẽ cơ bản.
* android.database: Cung cấp các lớp mức thấp bắt buộc cho việc điều khiển cursor khi làm việc với các cơ sở dữ liệu.
* android.content: Các giao tiếp lập trình nội dung được dùng để quản lý truy cập dữ liệu và xuất bản bằng cách cung cấp các dịch vụ thao tác với tài nguyên, Content Provider, và các gói.
* android.view: View là lớp giao diện người dùng cơ bản nhất. Tất cả giao diện người dùng được tạo ra đều phải sử dụng một tập các View để cung cấp cho các thành phần tương tác người dùng.
* android.widget: Xây dựng dựa trên gói View. Những lớp widget những thành phần giao diện được tạo sẵn được sử dụng để tạo nên giao diện người dùng. Các widget bao gồm danh sách, nút bấm, hộp nhập, các kiểu trình bày (layout).
* com.google.android.maps: Bộ API mức cao cung cấp truy cập đến điềukhiển bản đồ sẵn trong Android từ ứng dụng được xây dựng. Bao gồm cả 17 lớp MapView cũng như Overlay và MapController để tương tác với bản đồ bên trong ứng dụng.
* android.provider: Để tạo thuận lợi cho người phát triển truy cập đến các Content Provider tiêu chuẩn (như là dữ liệu danh bạ), gói cung cấp (Provider) bao gồm các lớp cho phép truy cập đến cơ sở dữ liệu chuẩn trong tất cả các bản phân phối Android.
* android.telephony: Các API điện đàm cung cấp khả năng tương tác trực tiếp với tầng điện thoại trong các thiết bị, cho phép tạo, nhận, theo dõi các cuộc gọi, tình trạng các cuộc gọi và tin nhắn SMS.
* android.webkit: Gói WebKit cung cấp các API để làm việc với các nội dung Web-based bao gồm một lơp WebView để tạo ra giao diện web,nhúng trong ứng dụng và một trình quản lý cookie.

Cùng với các API của Android, còn có một tập các thư viện C/C++ như:

* OpenGL: Thư viện dùng để tạo ra các đồ họa 3D dựa vào chuẩn OpenGLES 1.0 API.
* FreeType: Hỗ trợ xử lý bitmap và font vector.
* GGL: Thư viện cơ bản, dùng để cung cấp các engine đồ họa 2D.
* Libc: Thư viện C chuẩn, được tối ưu cho các thiết bị Linux-based.
* SQLite Engine: Cơ sở dữ liệu quan hệ gọn nhẹ, dùng để lưu trữ dữ liệu của ứng dụng.
* SSL: Hỗ trợ sử dụng giao thức mã hóa Secure Sockets Layer trong bảo mật truyền thông Internet

**2.2.3 Các thành phần của một ứng dụng android.**

Một ứng dụng trên Android được cấu thành từ bốn thành phần cơ bản sau:

* Activites.
* Serivces.
* Broadcast Receivers.
* Content Provider.

Các thành phần này không nhất thiết phải có mặt đầy đủ trong ứng dụng. Chúng ta có thể xem các thành phần nào được sử dụng trong ứng dụng bằng việc xem khai báo trong file *AndroidManifest.xml.*

2.2.3.1 Activity

* Khái niệm

Một hoạt động là một giao diện người dùng trực quan mà người dùng có thể thực hiện trên đó mỗi khi được kích hoạt. Một ứng dụng có thể có nhiều hoạt động và chúng có thể gọi qua lại lẫn nhau. Mỗi Activity là một dẫn xuất của lớp android.app.Activity.

Mỗi hoạt động có một cửa sổ để vẽ lên. Thông thường cửa sổ này phủ đầy màn hình, ngoài ra nó cũng có thể có thêm các cửa sổ con khác như là hộp thoại…Nội dung của cửa sổ của hoạt động được cung cấp bởi một hệ thống cấp bậc các View (là đối tượng của lớp Views).

* Vòng đời hoạt động(circle activities)

Các hoạt động trong hệ thống được quản lý bởi một cấu trúc dữ liệu ngăn xếp. Khi có một hoạt động được khởi tạo, nó được đẩy vào trong ngăn xếp, chuyển sang trạng thái thực thi và hoạt trộng trước đó sẽ chuyển sang trạng thái chờ. Hoạt động này chỉ trở lại trang thái kích hoạt khi mà hoạt động vừa khởi tạo kết thúc việc thực thi.

Một Activity có 3 trạng thái chính:

* Active hoặc running khi nó ở trên nhất màn hình và nhận tương tác người dùng.
* Paused khi Activity không còn là trọng tâm trên màn hình nhưng vẫn hiện thị trước người dùng.
* Stopped khi một Activity hoàn toàn bị che khuất, nó sẽ rơi vào trạng thái Stopped. Tuy nhiên, nó vẫn còn lưu trữ toàn bộ thông tin trạng thái. Và nó thường bị hệ thống đóng lại khi có tình trạng thiếu bộ nhớ. Khi chuyển giữa các trạng thái, ứng dụng sẽ gọi các hàm callback ứng

với các bước chuyển:

- void onCreate(Bundle savedInstanceState)

- void onStart()

- void onRestart()

- void onResume()

- void onPause()

- void onStop()

- void onDestroy()

Biểu đồ sau mô tả trạng thái trong vòng đời của một hoạt động. Hình chữ nhật thể hiện các phương thức Callback mà chúng ta có thể khai báo để gọi thực thi một số thao tác khi hoạt động chuyển sang trạng thái khác (phương thức Callback là phương thức được gọi lại bởi một phương thức khác khi có một sự kiện xảy ra). Các trạng thái chính của một hoạt động được thể hiện bởi các hình viên thuốc.

Vòng đời của một hoạt động có thể được thể hiện trong những quá trình

sau:

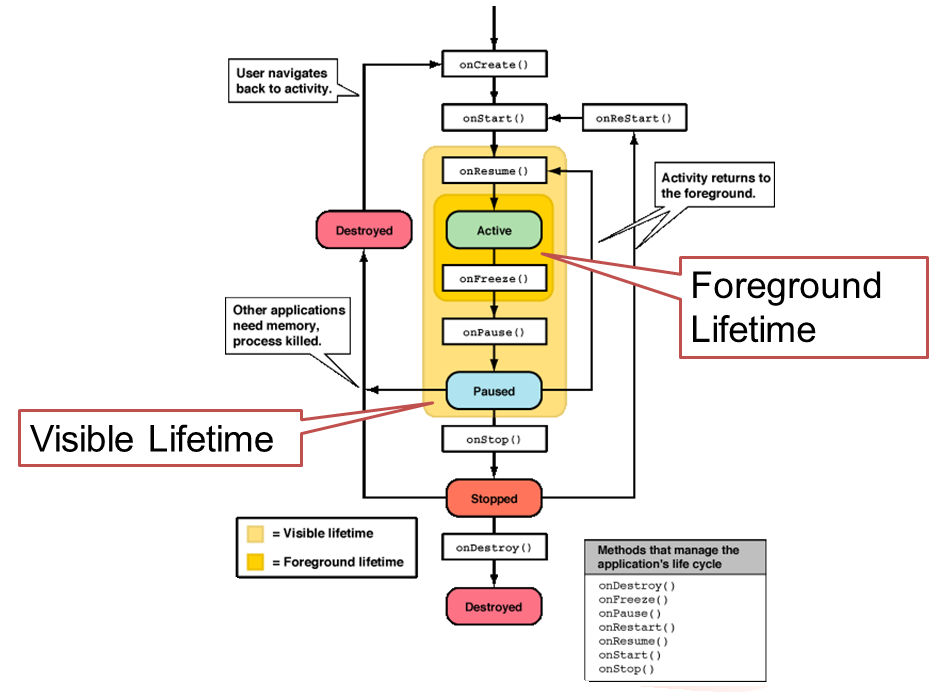


Figure 3 Vòng đời của một activity

Toàn bộ thời gian sống của một hoạt động bắt đầu từ lời gọi đầu tiên tới phương thức onCreate(Bundle) tới lời gọi phương thức onDestroy(). Trong quá trình này, một hoạt động sẽ khởi tạo lại tất cả các tài nguyên cần sử dụng trong phương thức onCreate() và giải phóng chúng khi phương thức onDestroy() được thực thi.

Thời gian sống có thể nhìn thấy của một hoạt động bắt đầu từ lời gọi tới phương thức onStart(), cho tới khi phương thức onStop() của nó được thực thi. Toàn bộ các tài nguyên đang được sử dụng bởi hoạt động vẫn tiếp tục được lưu giữ, người dùng có thể thấy giao diện nhưng không tương tác được với hoạt động do trong qua trình này hoạt động không ở trạng thái chạy tiền cảnh.

Thời gian sống tiền cảnh của một hoạt động là quá trình bắt dầu từ khi có lời gọi tới phương thức onResume() và kết thúc bằng lời gọi tới phương thức onPause(). Trong thời gian này, hoạt động chạy ở tiền cảnh và có thể tương tác với người dùng.

**2.2.3.2 Service**

Một dịch vụ (Service) là các đoạn mã được thực thi ngầm bởi hệ thống mà người sử dụng không thấy được. Mỗi service đều được mở rộng từ lớp cơ sở là service trong gói android.app. Có thể kết nối tới hoặc kích hoạt một Service thông qua interface mà Service đưa ra.Ví dụ như một chương trình chơi nhạc, sẽ có vài hoạt động cho phép người dùng duyệt danh sách các bài hát và lựa chọn bài nào để phát. Tuy nhiên, chức năng chơi nhạc không được thiết kế như một hoạt động bởi chúng ta sẽ muốn chuyển qua cửa sổ khác, như khi soạn tin nhắn thì bài nhạc vẫn tiếp tục được chơi. Trong trường hợp này, ứng dụng chơi nhạc sẽ khởi tạo một dịch vụ bằng cách sử dụng phương thức:

Context.startService().

Một ứng dụng có thể dễ dàng thực hiện liên kết tới một dịch vụ đang chạy (thậm chí khởi động nếu nó chưa thực thi) bằng phương thức Context.bindService(). Khi đó dịch vụ này sẽ cung cấp cho ứng dụng cơ chế để giao tiếp với chúng thông qua giao diện gọi là IBinder (đối với dịch vụ chơi nhạc có thể cho phép dừng hoặc chuyển qua bài nhạc kế tiếp).

* Vòng đời của một dịch vụ được hiểu là quá trình hoạt động từ khi nó được tạo ra cho tới khi bị loại khỏi hệ thống. Có hai cách thức để một dịch vụ có thể được chạy trong hệ thống.
* Khi hệ thống có lời gọi tới phương thức Context.startService(). Trong trường hợp này, dịch vụ sẽ được thực hiện liên tục cho tới khi hệ thống gọi phương thức Context.stopService().
* Khi các ứng dụng gọi phương thức Context.bindService() để tạo kết nối với dịch vụ (dịch vụ sẽ được khởi tạo nếu tại thời điểm đó nó đang không hoạt động). Ứng dụng sẽ nhận được một đối tượng IBinder do dịch vụ trả lại để có thể gọi các phương thức Callback phù hợp để truy cập tới các trạng thái của dịch vụ. Nếu do lời gọi Context.bindService() mà dịch vụ được khởi tạo thì nó sẽ được thực thi cho tới khi nào kết nối trên (tức là đối tượng IBinder) vẫn còn tồn tại.

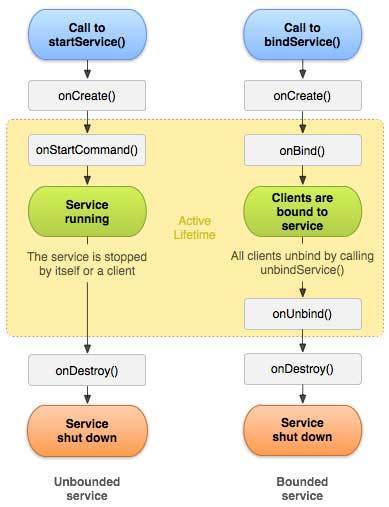


Figure 4 Sơ đồ chuyển trạng thái của service

**2.2.3.3 Bộ nhận quảng bá (Broadcast Receivers)**

Bộ nhận quảng bá là một thành phần không làm gì cả nhưng nó nhận và phản hồi lại các thông báo quảng bá. Nhiều quảng bá có nguồn gốc từ mã hệ thống, ví dụ thông báo thay đổi múi giờ, pin yếu, ảnh đã chụp hay thay đổi ngôn ngữ. Các ứng dụng có thể khởi động quảng bá, ví dụ để các ứng dụng khác biết rằng dữ liệu đã được tải về xong trên thiết bị và sẵn sàng sử dụng.

Một ứng dụng có thể có bất kỳ số lượng bộ nhận quảng bá nào để nhận những thông báo quan trọng với nó. Tất cả các bộ nhận quảng bá được kế thừa từ lớp BroadcastReceiver.

Bộ nhận quảng bá không có giao diện. Tuy nhiên, chúng có thể khởi động một hoạt động để đáp lại thông tin mà nó nhận được, hay chúng có thể sử dụng NotificationManager để thông báo người dùng biết. Các thông báo có thể được sự chú ý của người dùng theo các cách các nhau như là sáng màn hình, rung thiết bị, bật âm thanh nào đấy… Thông thường, chúng đặt thông báo trên thanh trạng thái, nơi người dùng có thể nhận được thông báo.

**2.2.3.4 Content Provider**

Các ứng dụng có thể lưu trữ dữ liệu của mình trong các tập tin hoặc sử dụng cơ sở dữ liệu SQLite sẵn có v.v… Content Provider có chức năng cung cấp một tập hợp các phương thức cho phép một ứng dụng có thể lưu trữ và lấy dữ liệu được quản lý bởi content provider đó.

Content Provider là một đặc trưng riêng của Android, nhờ đó mà các ứng dụng có thể chia sẻ dữ liệu với nhau một cách dễ dàng.

**2.2.3.5 Các Intent**

Content Provider được kích hoạt khi chúng được gọi từ một ContentResolver. Ba thành phần khác (hoạt động, dịch vụ và bộ nhận quảng bá) được kích hoạt bởi thông điệp không đồng bộ từ các Intent. Một Intent là một đối tượng có kiểu Intent chứa nội dung của thông điệp. Với các hoạt động và dịch vụ, nó gọi tên hành động được yêu cầu và xác định URI của dữ liệu tác động tới ở giữa. Ví dụ, nó có thể truyền tải một yêu cầu cho một hoạt động hiển thị một ảnh cho người dùng hay cho phép người dùng sửa văn bản. Với bộ nhận quảng bá, đối tượng Intent gọi tên của hành động được thông báo. Ví dụ, bộ nhận quảng bá có thể thông báo các phần nó quan tâm là nút chụp ảnh đã được bấm.

Có vài phương thức cho việc kích hoạt mỗi thành phần:

Một hoạt động được khởi chạy thông qua một đối tượng Intent Context.startActivity() hay Activity.startActivityForResult(). Hoạt động đáp lại có thể theo dõi Intent được tạo ra đó bằng phương thức getIntent() và cập nhật thông qua phương thức setIntent(Intent). Android gọi phương thức onNewIntent() để bỏ qua các Intent đến trước nó.

Một hoạt động thường bắt đầu hoạt động khác. Nếu nó muốn trả lại kết quả hoạt động nó đã khởi chạy, nó sẽ gọi phương thức:

startActivityForResult() thay cho phương thức startActivity(). Ví dụ, nếu nó khởi chạy một hoạt động mà cho phép người dùng lấy một ảnh, nó có thể muốn lấy kết quả của ảnh được chọn. Kết quả được trả về trong một đối tượng Intent thông qua phương thức onActivityResult().

Một dịch vụ được bắt đầu thông qua một đối tượng Intent là Context.startService(). Android gọi phương thức onStart() của dịch vụ và thông qua đối tượng Intent của nó.

Tương tự, một Intent có thể thông qua Context.bindService() để thiết lập một kết nối liên tục giữa các thành phần và dịch vụ đích. Dịch vụ nhận đối tượng Intent qua lời gọi onBind() (nếu dịch vụ chưa được chạy, bindService() có thể chọn bắt đầu nó). Cho ví dụ, một hoạt động có thể thiết lập kết nối với dịch vụ chơi nhạc đề cập ở phần trước để nó có thể cung cấp cho người dùng giao diện sử dụng để điều khiển chơi lại. Hoạt động sẽ gọi bindService để thiết lập kết nối và sau đó gọi phương thức đã định nghĩa bởi dịch vụ để áp dụng chơi lại ca khúc.

Một ứng dụng có thể khởi tạo một quảng bá thông qua đối tượng Intent bằng phương thức như:

Context.setBroadcast(),Context.setOrderedBroadcast()và Context.sendStickyBroadcast(). Android chuyển những Intent tới tất cả các bộ nhận quảng bá nào quan tâm bằng việc gọi phương thức onReceive() của nó

**2.2.3.6 Menifest File**

Trước khi có thể khởi chạy một ứng dụng thành phần, nó phải xem ứng dụng bao gồm những thành phần nào. Thêm nữa, các ứng dụng khai báo các thành phần của nó trong một tập tin khai báo để đóng gói lại vào trong gói Android (tập tin .apk chứa các mã nguồn, tập tin và tài nguyên).

Tập tin này có cấu trúc của tập tin XML và luôn có tên là AndroidManifest.xml trong mọi ứng dụng. Tập tin này thực hiện một số chức năng như thêm và khai báo các thành phần của ứng dụng, tên các thư viện ứng dụng cần liên kết tới (ngoài thư viện chuẩn của Android) và xác định các quyền cho ứng dụng.

**2.2.3.7 Bộc lọc Intent**

Một đối tượng Intent có thể có tên rõ ràng trong thành phần đích. Nếu có, Android sẽ tìm thành phần đó (dựa trên khai báo trong tập tin khai báo) và kích hoạt nó. Nhưng nếu đích có tên không rõ ràng, Android phải xác định thành phần nào thích hợp nhất để đáp lại intent. Nó thực hiện so sánh đối tượng Intent với bộ lọc intent trong các đích có khả năng. Một thành phần của bộ lọc intent cho Android biết loại intent thành phần nào cần xử lý. Giống các thông tin cơ bản, chúng được khai trong tập tin khai báo. Đây là một ví dụ để thêm hai bộ lọc intent cho một hoạt động.

**2.3 Công nghệ Pattern**

**2.3.1 Style Convention**

Style Convention là tập hợp những nguyên tắc chung khi lập trình nhằm làm cho code dễ đọc, dễ hiểu, do đó dễ quản lý, bảo trì hơn.

Style Convention có những cái chung và cái riêng tuỳ ngôn ngữ, tuỳ cộng đồng, nhưng hầu hết được công nhận và đi theo bởi đa số các lập trình viên trên thế giới.

**2.3.2 Design pattern**

Design patterns là các giải pháp đã được tối ưu hóa, được tái sử dụng cho các vấn đề lập trình mà chúng ta gặp phải hàng ngày. Nó là một khuôn mẫu đã được suy nghĩ, giải quyết trong tình huống cụ thể rồi.

Các vấn đề mà bạn gặp phải có thể bạn sẽ tự nghĩ ra cách giải quyết nhưng có thể nó chưa phải là tối ưu. Design Pattern giúp bạn giải quyết vấn đề một cách tối ưu nhất, cung cấp cho bạn các giải pháp trong lập trình OOP.

Nó không phải là ngôn ngữ cụ thể nào cả. Design patterns có thể thực hiện được ở phần lớn các ngôn ngữ lập trình. Ta thường gặp nó nhất trong lập trình OOP.

2.3.2.1 Tại sao sử dụng design pattern

* Design Pattern giúp bạn tái sử dụng mã lệnh và dẽ dàng mở rộng.
* Nó là tập hơn những giải pháp đã được tối ưu hóa, đã được kiểm chứng để giải quyết các vấn đề trong software engineering. Vậy khi bạn gặp bất kỳ khó khăn gì, design patterns là kim chỉ nam giúp bạn giải quyết vấn đề thay vì tự tìm kiếm giải pháp cho một vấn đề đã được chứng minh.
* Design pattern cung cấp giải pháp ở dạng tổng quát, giúp tăng tốc độ phát triển phần mềm bằng cách đưa ra các mô hình test, mô hình phát triển đã qua kiểm nghiệm.
* Dùng lại các design pattern giúp tránh được các vấn đề tiềm ẩn có thể gây ra những lỗi lớn, dễ dàng nâng cấp, bảo trì về sau.
* Giúp cho các lập trình viên có thể hiểu code của người khác 1 cách nhanh chóng (có thể hiểu là tính communicate). Mọi thành viên trong team có thể dễ dàng trao đổi với nhau để cùng xây dựng dự án mà k mất quá nhiều thời gian.

2.3.2.2 phân loại design pattern

Có 3 nhóm chính sau:

* + Creational Pattern (nhóm khởi tạo) gồm: Abstract Factory, Factory Method, Singleton, Builder, Prototype. Nó sẽ giúp bạn trong việc khởi tạo đối tượng, như bạn biết để khởi tạo bạn phải sử dụng từ khóa new, nhóm Creational Pattern sẽ sử dụng một số thủ thuật để khởi tạo đối tượng mà bạn sẽ không nhìn thấy từ khóa này.
  + Structural Pattern (nhóm cấu trúc) gồm: Adapter, Bridge, Composite, Decorator, Facade, Proxy và Flyweight. Nó dùng để thiết lập, định nghĩa quan hệ giữa các đối tượng.
  + Behavioral Pattern gồm: Interpreter, Template Method, Chain of Responsibility, Command, Iterator, Mediator, Memento, Observer, State, Strategy và Visitor. Nhóm này dùng trong thực hiện các hành vi của đối tượng.

**2.3.2 Animation**

2.3.2.1 Giới Thiệu

Trong một ứng dụng web, desktop, hay ứng dụng trên mobile thực sự không thể thiếu đi những animation. Và animation là gì? Nó giúp được gì cho ứng dụng của chúng ta và sử dụng chúng khi nào? Animation là những chuyển động hay tập hợp những chuyển động trên các đối tượng giúp chúng chuyển động mượt mà hơn.

Ví dụ chúng ta có hai đối tượng là hình chữ nhật ở bên trái màn hình (vị trí A) và bây giờ tôi muốn di chuyển nó đến bên phải màn hình (vị trí B). Như vậy chúng ta có hai cách làm như sau:

Cách 1: Không sử dụng animation

Đơn giản chúng ta chỉ vẽ lại hình chữ nhật này ở phía bên trái màn hình. Cách này thì hình chữ nhật xuất hiện một cách đột ngột.

Cách 2: Sử dụng animation

Chúng ta sẽ trượt hình chữ nhật sang bên phải sau một khoảng thời gian để đến vị trí B. Theo cách này thì sự di chuyển của hình chữ nhật rất là mượt mà.

Trên đây là những giới thiệu cơ bản về Animation. Phần nội dung chính chúng ta sẽ nói kĩ hơn về Animation sử dụng ViewPropertyAnimator

2.3.2.2 Các cách thực hiện Animation trong Android

Trong Android có khá nhiều cách để các bạn có thể thực hiện Animation. Dưới đây tôi sẽ liệt kê một số class, method thường được dùng cho Animation.

* Những Animation có từ bản API 1 và thường được gọi với tên là System Animation. Gồm có những class: RotationAnimation, ScaleAnimtion, TranslateAnimation, AlphaAnimation, AnimationSet. Và tất cả những class này đều kế thừa từ class Animation nằm trong package
* Những Animation được giới thiệu ở bản Android 3.0 (API 11) và thường được gọi với tên là Animator. Những class chính là ValueAnimator, ObjectAnimator, AnimatorSet… Tất cả những class này đều kế thừa từ lớp Animator trong package android.animation.Animator
* Ngoài cách trên để chúng ta có thể thực hiên Animation còn có một class có tên là ViewPropertyAnimator giúp chúng ta có thể thực hiện Animate rất là dễ dàng.

2.3.2.3 Hoạt động của Animation

Thực chất thì Animation trong Android chỉ là sự thay đổi những thuộc tính của View. Và khi có những sự thay đổi thì Android sẽ tiến hành vẽ lại View (gọi lại phương thức onDraw của class View).

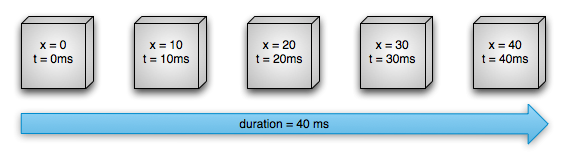


Figure 5 Hinh anh minh hoa animation

2.3.2.4 Sử dụng ViewPropertyAnimator để Animation View trong Android

Việc sử dụng ViewPropertyAnimator hoàn toàn rất dễ dàng. Và thông thường có những bước sau:

Bước 1: Lấy về đối tượng ViewPropertyAnimator

Bước 2: Xác định thuộc tính của view cần animate và sử dụng các phương thức tương ứng

Bước 3. Set các event cho animation

Bước 4: Set thời gian chạy (duration) cho animation, set phương tính toán nội suy cho animation, và chạy animation.

**CHƯƠNG III : PHÂN TÍCH THIẾT KẾ**

**3.1. Xác định yêu cầu**

**3.1.1 Chức năng chính**

* Ôn Luyện Thi Toeic.
* Trắc Nghiệm Toeic.
* Cho Phép Học Thuộc Từ Toeic.
* Các Chức Năng Bổ Sung Theo App.

**3.1.2 Mô hình hoạt động MVP pattern**

3.1.2.1 Giới thiệu về mô hình MVP.

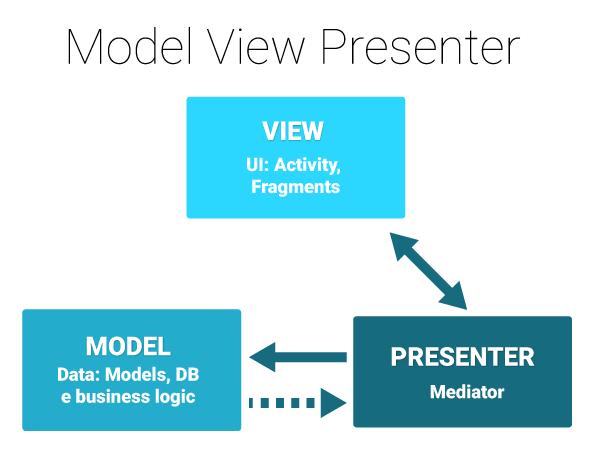
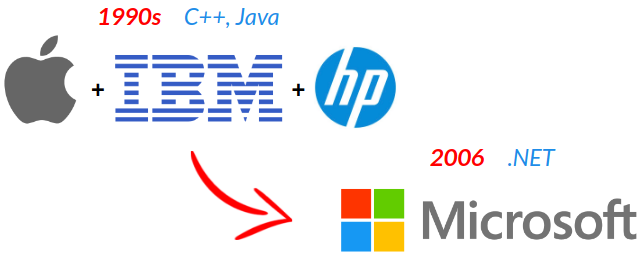


Figure 6mô hình MVP Android

MVP là viết tắt của model-view-presenter. Là một kiến trúc phần mềm ra đời từ những năm 1990 tại công ty Taligent ở mỹ. Taligent là một công ty bao gồm sự hợp tác của 3 đại gia công nghệ lớn là Apple, IBM và Hewlett-Packard. MVP ban đầu là kiến trúc cơ bản để phát triển ứng dụng trong môi trường CommonPoint dựa trên C++ và sau đó chuyển sang Java. Đến năm 1998 thì Taligent giải thể.

Năm 2006, Microsoft bắt đầu kết hợp vào documentation và examples cho lập giao diện người dùng (User Interface - UI) trong .NET framework



3.1.2.2 ***MVP là gì?***

MVP là một User Interface Architectural Pattern (kiến trúc giao diện người dùng) được thiết kế để tạo điều kiện cho Automated Unit Testing và cải tiến Separation of Concerns trong việc trình bày logic (presentation logic).

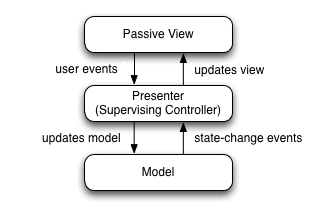
Thế thì Automated Unit Testing là gì? Hiểu một cách đơn giản là:

Thứ nhất Unit Testing là một quá trình phát triển phần mềm, trong đó có các phần nhỏ nhất có thể test, kiểm tra của ứng dụng gọi là Unit, được xem xét riêng rẽ, độc lập để hoạt động tốt.

Thứ hai: Unit Testing có thể được thực hiện thủ công nhưng thường là tự động nên được gọi là Automated Unit Testing.

Còn Separation of Concerns bạn có thể hiểu là nguyên tắc để tách một chương trình phần mềm thành các thành phần riêng biệt, phần này thay đổi thì không phục thuộc vào phần kia và ngược lại.

Vậy MVP cấu tạo bao gồm những gì để thỏa mãn 2 yếu tố trên:



**Mode**: là một interface xác định dữ liệu được hiển thị, hoặc dữ liệu này được thực hiện trong giao diện người dùng (UI)

**View**: là một interface thụ động dùng để hiện thị dữ liệu (là Model) và định hướng các lệnh người dùng (events) tới Presenter để Presenter hành động dựa trên các dữ liệu đó.

**Presenter**: hành động theo Model và View. Presenter lấy dữ liệu từ repositories (Model), sau đó định dạng dữ liệu và hiển thị lên View.

3.1.2.3 **Tại sao phải là MVP?**

Mọi công việc thật ra là đơn giản với dễ dàng hơn khi có một kế hoạch thích hợp đúng không nào! Nếu không lên kế hoạch thì sẽ mang lại nhiều khó khăn và rủi ro hơn, tốn công sức hơn. Trong lập trình phần mềm cũng vậy, cụ thể là trong Android. Để bắt đầu một dự án Android nào, thì việc đầu tiên là phải chọn được một kiến trúc. Nếu bạn không chọn đúng bạn có thể phải đối mặt với các vấn đề sau trong quá trình phát triển ứng dụng của mình:

1.Phức tạp để code cho Unit Testing. Trong khi Unit Testing khuyến khích Developers thay đổi mã nguồn.

2.Khó theo dõi logic trong dự án.

3.Khó để truy trì (maintain) và thêm các tính năng (feature) mới cho quá trình sử dụng và phát triển phần mềm.

Vậy một ứng dụng Android chất lượng cao thì đầu tiên nó phải có một kiến trúc lý tưởng đúng không nào? Vậy kiến trúc đó phải như thế nào:

1.**Simplicity**: Sự đơn giản! Kiến trúc phải tách ứng dụng Android thành các module nhỏ, và phải xác định một vai trò duy nhất và rõ ràng cho mỗi module đó.

2.**Focus on Business Logic**: Một kiến trúc lý tưởng phải phân tách mã để cho các Developers có thể tập trung vào business logic thay vì cải tiến các đoạn code khác.

3.**Low-cost Maintenance**: Chi phí bảo trì thấp! Một kiến trúc lý tưởng thì không mất nhiều thời gian để thay đổi logic. Nó cũng dễ dàng thêm (add) các tính năng mới hay gỡ bỏ (remove) những tính năng không còn phù hợp nữa.

Với MVP nó đã tách ứng dụng Android thành 3 tầng là model-view-presenter. Và trong mỗi tầng đều được chia thành module nhỏ và mỗi module lại có một vai trò riêng biệt. Vậy nó đã thỏa mãn Simplicity.

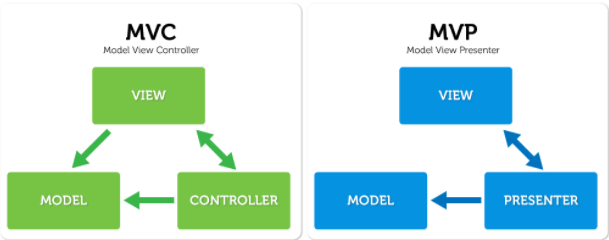
Tầng presenter là nơi tập trung xử lý logic, là cầu nối giữa model và view. Đây là nơi mà các Developers có thể focus để theo theo dõi logic của ứng dụng và thay đổi chúng. Thỏa mãn Focus on Business Logic.

Cũng vì tách thành 3 tầng nên MVP là cho ứng dụng Android dễ dàng Testing. Dễ dàng gỡ bỏ những tính năng không còn phù hợp và thêm các tính năng mới -> maintain ứng dụng dễ dàng, chi phí thấp

Vậy MVP đúng là một một pattern lý tưởng đúng chưa các bạn!!!

3.1.2.4 **So sánh MVC với MVP**

MVC (model-view-controller) có thể xem là một pattern được tiếp cận đầu tiên. Nó ra đời từ những năm 1970 và rất được thịnh hành trong lập trình Web. MVC cũng được áp dụng nhiều trong Android. Đó là trước đây, nhưng hiện tại thì không phù hợp nữa. Để hiểu vì sao thì ta có thể đem so sánh sự khác nhau với MVP.

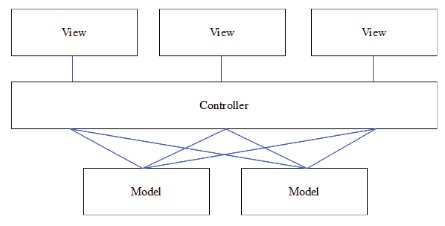


**MVC**

. View có thể gọi đến Model

. Unit Testing không tốt bằng MVP

. Tất cả các View cùng dùng chung một Controller

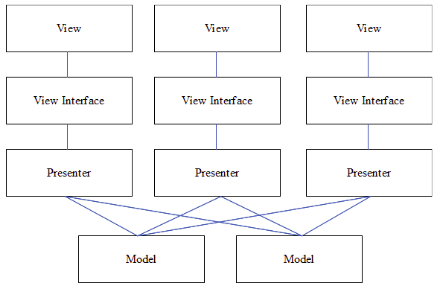


**MVP**

. View có thể xem là không có liên kết gì với Model, Presener có trách nhiệm liên kết Model tới View.

. Dễ dàng sử dụng Unit Testing hơn vì tương tác với View thông qua interface.

. Thường thì một View sẽ tương ứng với một Presenter. Những View phức tạp có thể có nhiều presenter cùng đảm nhiệm.



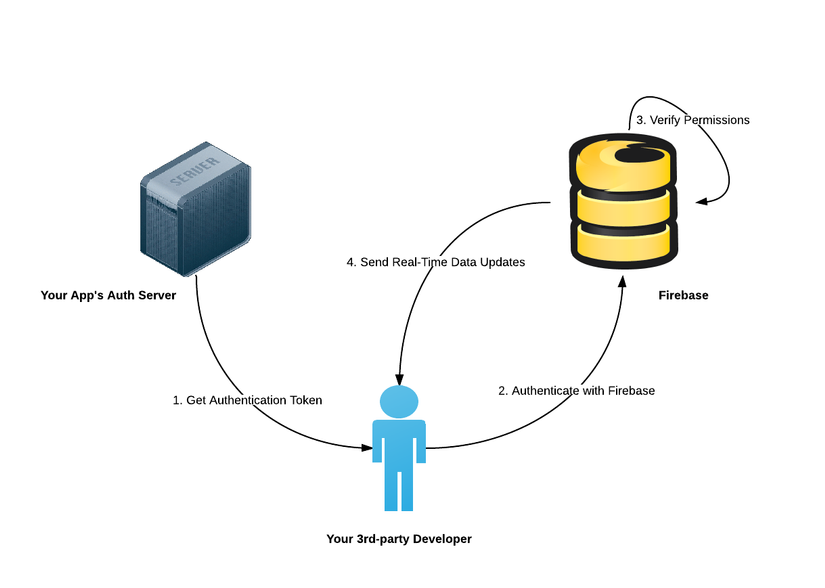
**3.1.3 Giới Thiệu Về Firebase**

Hiện Tại, Ứng dụng đang sử dụng FireBase dùng để lưu trữ dữ liệu server : User, data toeic. Image…

3.1.2.1 Giới thiệu FireBase.

Firebase là một dịch vụ hệ thống backend được Google cung cấp sẵn cho ứng dụng Mobile của bạn, với Firebase bạn có thể rút ngắn thời gian phát triển, triển khai và thời gian mở rộng quy mô của ứng dụng mobile mình đang phát triển. Hỗ trợ cả 2 nền tảng Android và IOS, Firebase mạnh mẽ, đa năng, bảo mật và là dịch vụ cần thiết đầu tiên để xây dưng ứng dụng với hàng triệu người sử dụng. Sử dụng Firebase bạn sẽ có được hưởng các lợi ích sau:

* Xây dựng ứng dụng nhanh chóng mà không tốn thời gian, nhân lực để quản lý hệ thống và cơ sơ sở hạ tầng phía sau: Firebase cung cấp cho bạn chức năng như phân tích, cơ sở dữ liệu, báo cáo hoạt động và báo cáo các sự cố lỗi để bạn có thể dễ dàng phát triển, định hướng ứng dụng của mình vào người sử dụng nhằm đem lại các trải nghiệm tốt nhất cho họ.
* Uy tín chất lượng đảm bảo từ Google: Firebase được google hỗ trợ và cung cấp trên nền tảng phần cứng với quy mô rộng khắp thế giới, được các tập đoàn lớn và các ưng dụng với triệu lượt sử dụng từ người dùng.
* Quản lý cấu hình và trải nghiệm các ứng dụng của Firebase tập trung trong một giao diện website đơn giản, các ứng dụng này hoạt động độc lập nhưng liên kết dữ liệu phân tích chặt chẽ.



3.1.2.2 **Nhóm công cụ Develop & test your app**

Realtime Database: Lưu trữ và đồng bộ dữ liệu người dùng thời gian thực, các ứng dụng hỗ trợ tính năng này có thể lưu trữ và lấy dữ liệu từ máy chủ trong tích tắc. Các dữ liệu được lưu trữ trong hệ thống cơ sở dữ liệu hỗ trợ NoSQL và được đặt trên nền tảng máy chủ Cloud, dữ liệu được ghi và đọc với thời gian thấp nhất tính bằng mili giây. Nền tảng này hỗ trợ đồng bộ hóa dữ liệu của người dùng kể cả khi không có kết nối mạng, tạo nên trải nghiệm xuyên suốt bất chấp tình trạng kết nối internet của người sử dụng. Reatime Database của Firebase hổ trợ: android, ios, web, c++, unity, và cả xamarin.

**Crashlytics**: Hệ thống theo dõi và lưu trữ thông tin lỗi của ứng dụng đang chạy trên máy người dùng. Các thông tin lỗi này được thu thập một các toàn diện và ngay tức thời. Cách trình bày hợp lý với từng chu trình hoạt động đến khi xảy ra lỗi, các báo cáo trực quan giúp người phát triển có thể nắm bắt và xử lý kịp thời các lỗi chính của ứng dụng.

**Cloud Firestore**: Lưu trữ và đồng bộ dữ liệu giữa người dùng và thiết bị – ở quy mô toàn cầu – sử dụng cơ sở dữ liệu noSQL được lưu trữ trên hạ tầng cloud. Cloud Firestore cung cấp cho bạn tính năng đồng bộ hóa trực tuyến và ngoại tuyến cùng với các truy vấn dữ liệu hiệu quả. Tích hợp với các sản phẩm Firebase khác cho phép bạn xây dựng các ứng dụng thực sự ngay cả khi kết nối internet bị gián đoạn.

**Authentication**: Quản lý người dùng một cách đơn giản và an toàn. Firebase Auth cung cấp nhiều phương pháp để xác thực, bao gồm email và mật khẩu, các nhà cung cấp bên thứ ba như Google hay Facebook, và sử dụng trực tiếp hệ thống tài khoản hiện tại của bạn. Xây dựng giao diện của riêng bạn hoặc tận dụng lợi thế của mã nguồn mở, giao diện người dùng tùy biến hoàn toàn.

**Cloud Functions**: Mở rộng ứng dụng của bạn bằng mã phụ trợ tùy chỉnh mà không cần quản lý và quy mô các máy chủ của riêng bạn. Các chức năng có thể được kích hoạt bởi các sự kiện, được phát sinh ra bởi các sản phẩm Firebase, dịch vụ Google Cloud hoặc các bên thứ ba có sử dụng webhooks.

**Cloud Storage**: Lưu trữ và chia sẻ nội dung do người dùng tạo ra như hình ảnh, âm thanh và video với bộ nhớ đối tượng mạnh mẽ, đơn giản và tiết kiệm chi phí được xây dựng cho quy mô của Google. Các Firebase SDK cho Cloud Storage thêm tính năng bảo mật của Google để tải lên và tải tệp cho các ứng dụng Firebase của bạn, bất kể chất lượng mạng.

**Hosting**: Đơn giản hóa lưu trữ web của bạn với các công cụ được thực hiện cụ thể cho các ứng dụng web hiện đại. Khi bạn tải lên nội dung web, chúng tôi sẽ tự động đẩy chúng đến CDN toàn cầu của chúng tôi và cung cấp cho họ chứng chỉ SSL miễn phí để người dùng của bạn có được trải nghiệm an toàn, đáng tin cậy, độ trễ thấp, dù họ ở đâu.

**Test Lab for Android**: Chạy thử nghiệm tự động và tùy chỉnh cho ứng dụng của bạn trên các thiết bị ảo và vật lý do Google cung cấp. Sử dụng Firebase Test Lab trong suốt vòng đời phát triển của bạn để khám phá lỗi và sự không nhất quán để bạn có thể cung cấp một trải nghiệm tuyệt vời trên nhiều thiết bị.

**Performance Monitoring**: Chẩn đoán các vấn đề về hiệu suất ứng dụng xảy ra trên thiết bị của người dùng của bạn. Sử dụng dấu vết để theo dõi hiệu suất của các phần cụ thể trong ứng dụng của bạn và xem chế độ xem tổng hợp trong bảng điều khiển Firebase. Luôn cập nhật thời gian khởi động của ứng dụng và theo dõi các yêu cầu HTTP mà không cần viết bất kỳ mã nào.

3.1.2.2 **Nhóm công cụ Grow & engage your audience**

**Google Analytics**: Phân tích thuộc tính và hành vi của người dùng trong một bảng điều khiển đơn để đưa ra các quyết định sáng suốt về lộ trình sản phẩm của bạn. Nhận thông tin chi tiết về thời gian thực từ báo cáo hoặc xuất dữ liệu sự kiện thô của bạn tới Google BigQuery để phân tích tùy chỉnh.

**Cloud Messaging**: Gửi tin nhắn và thông báo cho người dùng qua các nền tảng Android, iOS và web một cách miễn phí. Bạn có thể gửi tin nhắn đến các thiết bị, nhóm thiết bị hoặc các chủ đề hoặc phân đoạn người dùng cụ thể. Nhắn tin đám mây Firebase (FCM) thậm chí là các ứng dụng lớn nhất, cung cấp hàng trăm tỷ thư mỗi ngày.

**Predictions**: Firebase Predictions áp dụng deep learning máy học với dữ liệu phân tích của bạn để tạo các nhóm người dùng năng động dựa trên hành vi dự đoán. Các nhóm người dùng này có thể được sử dụng để nhắm mục tiêu trong các sản phẩm khác như thông báo, Cấu hình từ xa và nhiều hơn nữa.

**Dynamic Links**: Sử dụng Liên kết động để cung cấp trải nghiệm người dùng tùy chỉnh cho iOS, Android và web. Bạn có thể sử dụng chúng để hỗ trợ web di động để thúc đẩy chuyển đổi ứng dụng gốc, người dùng chia sẻ người dùng, các chiến dịch xã hội và tiếp thị và hơn thế nữa. Dynamic Links cung cấp cho bạn các thuộc tính bạn cần để hiểu rõ hơn về tăng trưởng di động của bạn. Đây là dịch vụ thay thế goo.gl của Google.

**Remote Config**:Tùy chỉnh cách ứng dụng của bạn hiển thị cho mỗi người dùng. Thay đổi giao diện, triển khai các tính năng dần dần, chạy thử nghiệm A / B, cung cấp nội dung tùy chỉnh cho người dùng nhất định hoặc thực hiện các cập nhật khác mà không cần triển khai một phiên bản mới-tất cả từ bảng điều khiển Firebase. Giám sát tác động của những thay đổi của bạn và thực hiện các điều chỉnh chỉ trong vài phút.

**Invites**: Cho phép người dùng chia sẻ tất cả các khía cạnh của ứng dụng của bạn, từ mã giới thiệu đến nội dung yêu thích, qua email hoặc SMS. Giải pháp out-of-the-box này hoạt động với Google Analytics for Firebase, để bạn biết khi người dùng mở hoặc cài đặt một ứng dụng qua lời mời.

**App Indexing**: Thu hút lại người dùng bằng các ứng dụng đã cài đặt của họ với tích hợp Google Tìm kiếm này. Nếu người dùng có ứng dụng của bạn và họ tìm kiếm nội dung có liên quan, họ có thể khởi chạy nó trực tiếp từ kết quả. Nếu người dùng chưa có ứng dụng của bạn, một thẻ cài đặt sẽ xuất hiện khi họ tìm kiếm các ứng dụng tương tự.

**AdMob**: Kiếm tiền bằng cách hiển thị quảng cáo hấp dẫn cho khán giả toàn cầu. AdMob có tất cả những gì bạn cần để thực hiện chiến lược kiếm tiền trên lớp bậc nhất và để tối đa hóa doanh thu do mỗi người dùng tạo ra. Nó có thể được điều chỉnh cho ứng dụng của bạn, và API của nó được xây dựng để tích hợp các định dạng quảng cáo phong phú một cách dễ dàng.

**AdWords**: Có được thông tin và níu kéo lại người dùng với khả năng của Google. Bạn có thể chạy quảng cáo trên Tìm kiếm, hiển thị và video cũng như nhắm mục tiêu phân khúc người dùng cụ thể mà bạn xác định trong Google Analytics for Firebase. Cải thiện nhắm mục tiêu quảng cáo và tối ưu hóa hiệu suất chiến dịch của bạn.

**CHƯƠNG IV : GIỚI THIỆU ỨNG DỤNG**

**4.1 Giới thiệu app**

* **Yêu Cầu Chính :**
* Cho phép người dung học toeic
* Làm bài quiz về toeic
* Hiển thị đáp án đúng sai của người dùng.
* Luyện từ vựng toeic bằng hình ảnh.

**4.1.1 Danh Sách Màn Hình**

|  |  |  |
| --- | --- | --- |
| STT | Tên | Loại |
| 1 | Sign In Screen | Hiển thị màn hình đăng nhập bằng Google |
| 2 | Vocabulary Screen | Hiện thị màn hình cho người dùng học từ vựng toeic. |
| 3 | Full Quiz Screen | Hiển thị tất cả các bài quiz Toeic |
| 4 | About Screen | Hiển thị về about của ứng dụng |

**4.1.2 Sign In Screen**

Trang mở đầu ứng dụng khi cài đặt lần đầu tiên

* Giao diện.

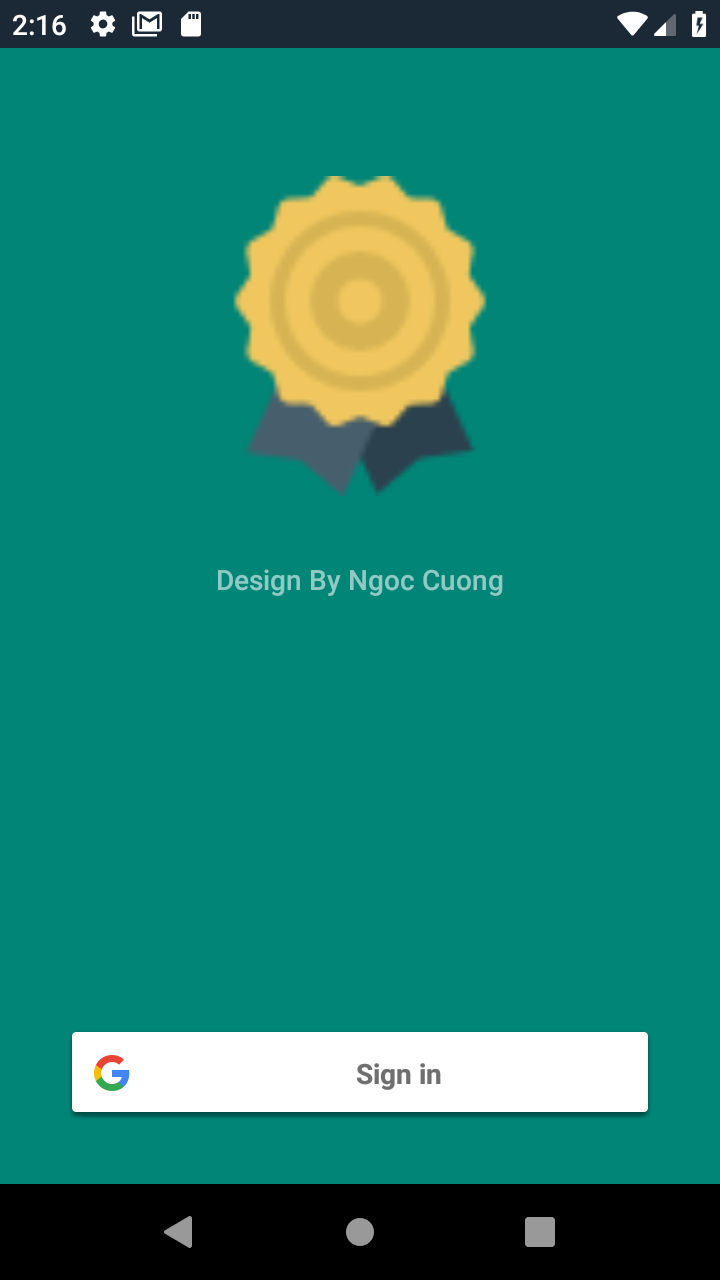


Figure 11màn hình đăng nhập bằng tài khoản google

* Mô tả chi tiết:
* Top : Hiển thị LogoAp và tên người code

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên | Loại | Ý Nghĩa |
| 1 | LogoApp | ImageView | Hiển Thị Logo App |
| 2 | Text about | TextView | Hiển thị tên người code(Desgin) |

* Bottom : Button của google.

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên | Loại | Ý Nghĩa |
| 1 | Sign In | Button Google | Button này là một dạng framework của google. Cho phép người dùng login vào các ứng dụng bằng chính tài khoản google của mình. |

**4.1.3 Vocabulary Screen**

Màn hình hiển thị 600 từ vựng toeic giúp cho người dùng có thể học các từ vựng.

* Ảnh chi tiết:

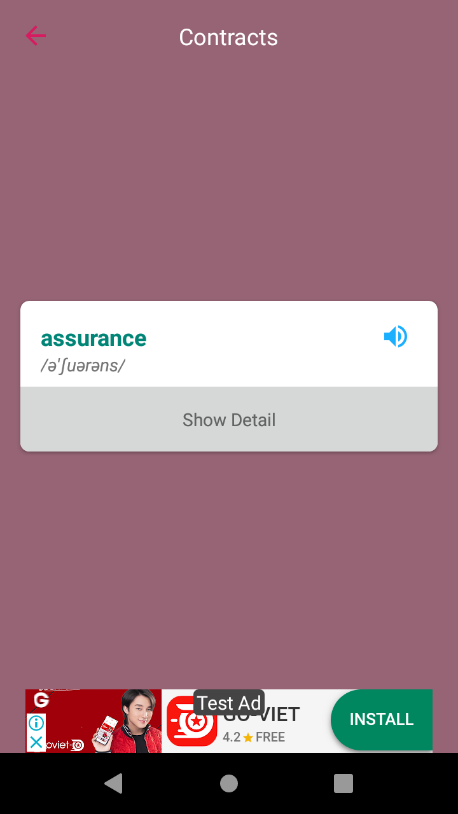
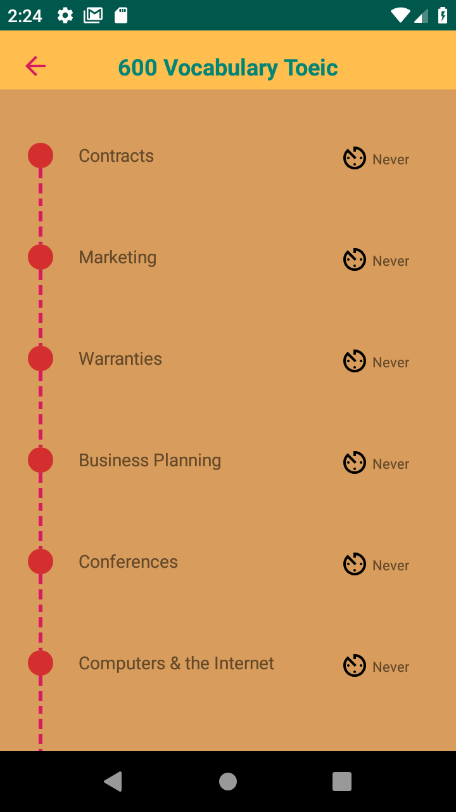


Figure 12hình ảnh về vocabularuy

* Mô tả chi tiết.

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên | Loại | Ý Nghĩa |
| 1 | Timeline | RecyclerView | Hiển thị các từ loại toeic theo chủ đề. Có 600 từ toeic và khoản 32 chủ đề. |
| 2 | Text to speech | Extension Google | Đâylà 1 dạng extension của google. Khi người dùng không biết phát âm các từ trong toeic. Nó sẽ giúp người dùng phát âm nó bằng cách click hình cái loa trên màn hình thứ 2. |
| 3 | Support Detail | Expandle View | Khi người dùng không biết từ vựng này, hoặc muốn ghi nhớ nó bằng hình ảnh. Người dùng ấn vào detail sẽ click vào button Detail, Nó sẽ hiển thị hình ảnh giải thích từ vựng đó. |

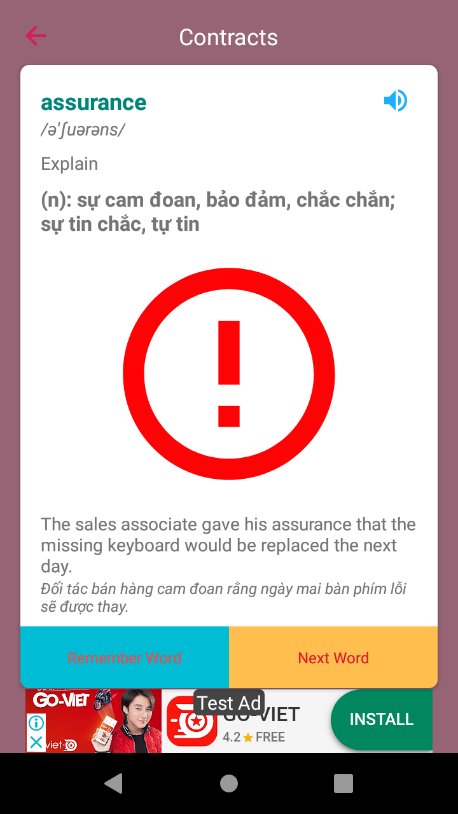
* Ảnh chi tiết:
* 

Figure 13hình ảnh về chọn hình ảnh

**4.1.4 Full Quiz Screen**

Hiển thị tất cả các câu hỏi toeic theo kiểu quiz example.

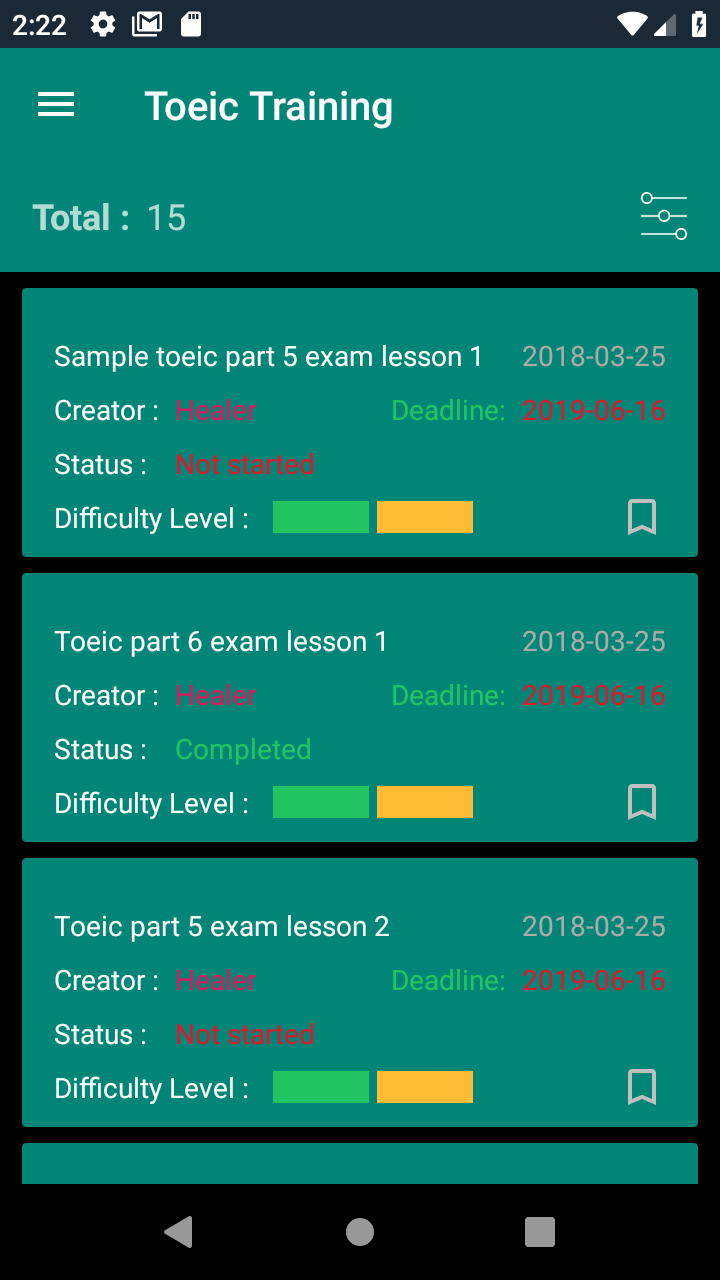


Figure 14hình ảnh về chọn Full Quiz

Tính năng :

+ Hiển thị ngày cập đăng, ngày cập nhập.

+ Hiển thị người tạo câu hỏi.

+ Hiển thị mức độ khó.

+ Hiển thị trạng thái hoàn thành

=> Tất cả mọi thứ đều được lưu vào Firebase

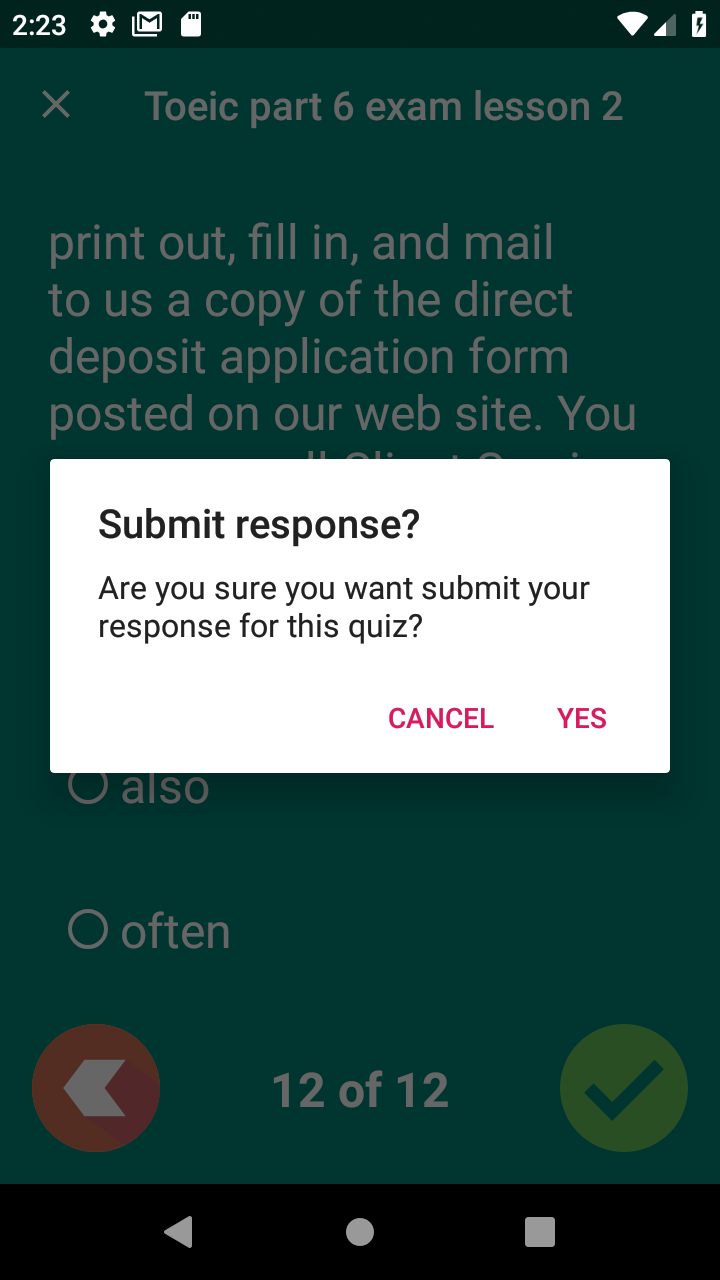
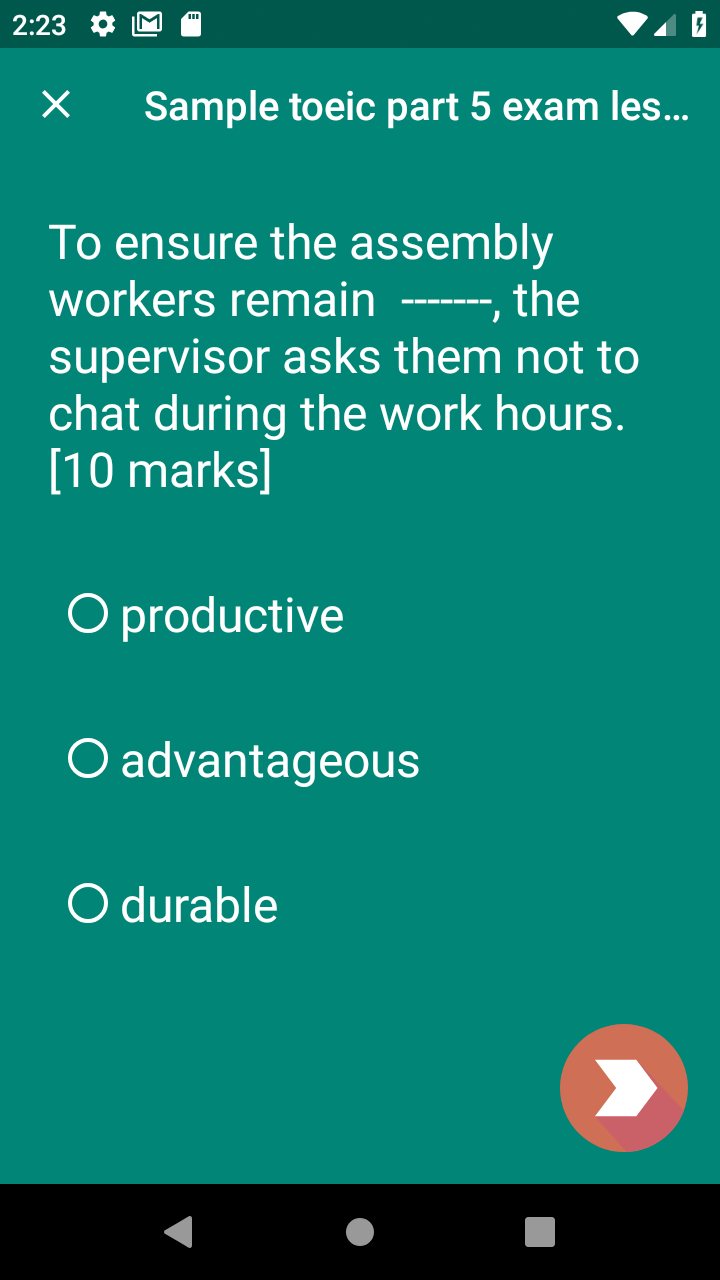


Figure 15hình ảnh về quiz example

Khi người dùng chọn câu trả lời xong, Ứng dụng sẽ request check câu trả lời từ server xem có đúng hay không. Nếu người dùng trả lời đúng, Ứng dụng sẽ tự tính điểm trên server.

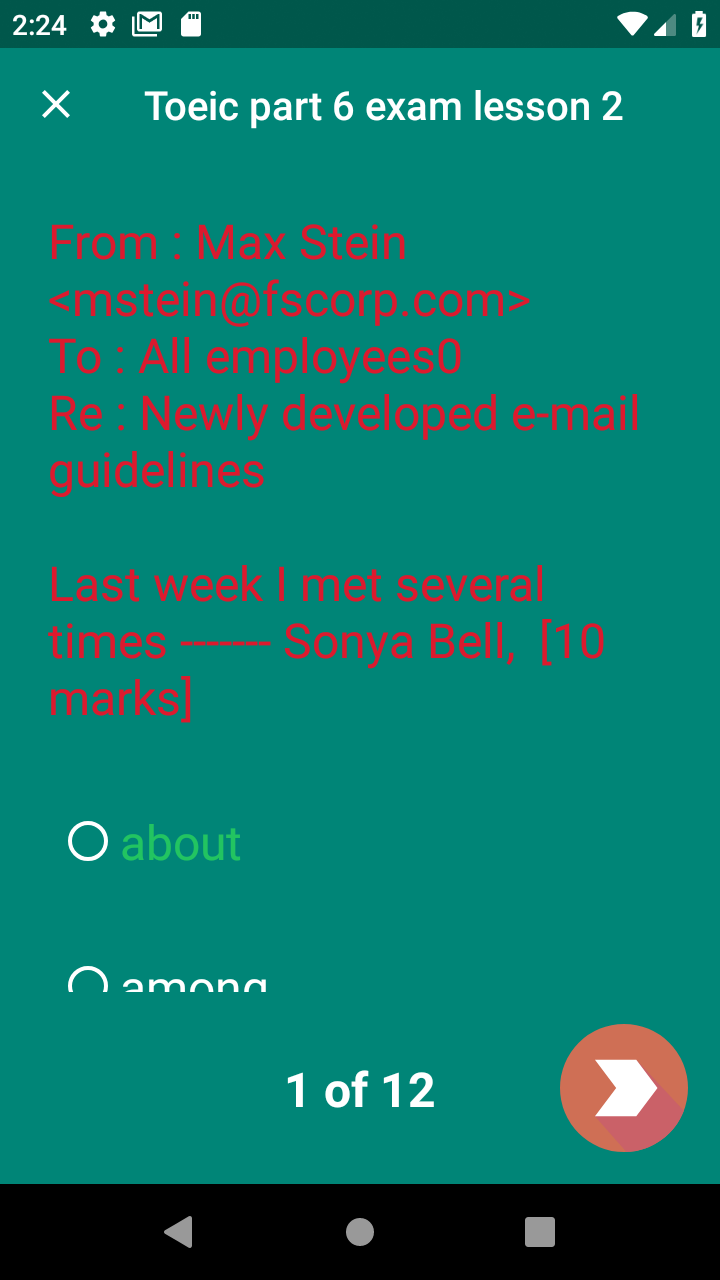
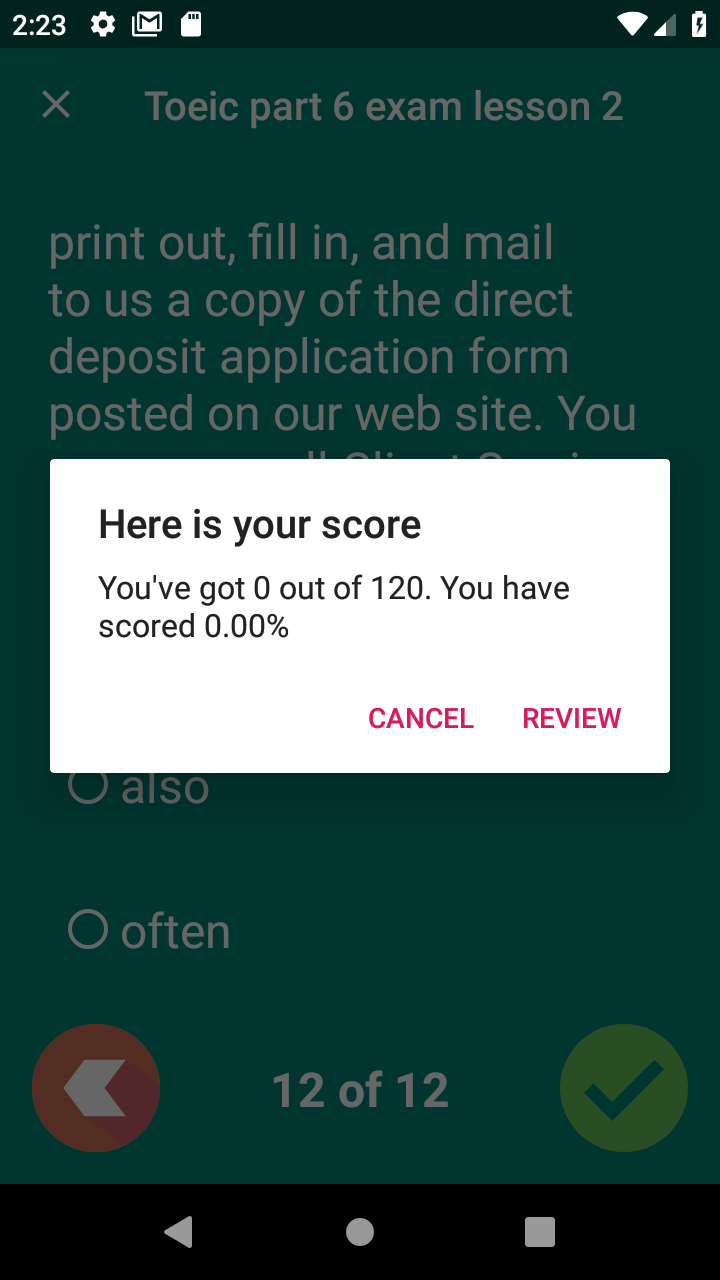


Figure 16hình ảnh về submit quiz

Chức năng review lại câu trả lời là một tính năng hay của App, Nó cho phép người dùng có thể review lại các câu trả lời của mình, sau này sẽ không phải gặp lại.

**4.1.5 About Screen**

****

Figure 17 hình ảnh về About Screen

Màn hình hiển thị thông tin về phiên bản hiện tại, vào version name and thông tin người tạo ra ứng dụng.

**CHƯƠNG V : KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

**5.1. Kết Luận**

1. Tìm hiểu được quá trình lập trình App Adroid.
2. Xây dựng được ứng dụng Android hoàn chỉnh.
3. Biết thêm về các công nghệ của Google : DesignPattern, MVP, MVC…
4. Năm được Framework mới như : Firebase...

**5.2. Hướng Phát Triển**

1. Tạo thêm nhiều loại quiz như Toefl hoặc Ielts.
2. Xây dựng lại design app.
3. Có thể update lên Google play cho người dùng sử dụng.