**Bộ câu hỏi Android**

**A. Nhóm các câu hỏi để xác định kiến thức, kỹ năng và kinh nghiệm lập trình Android:**

**1. Lập trình hướng đối tượng là gì?**

*Là phương pháp lập trình mà trong đó lập trình viên sẽ giải quyết các bài toán = cách quan sát và tưởng tượng những hành động, đặc điểm của thực thể thật ngoài đời sống và đem vào lập trình như một đối tượng ảo (Object) mà hành động là các hàm(method) còn đặc điểm chính các biến(variable)*

***2. Các tính chất của lập trình hướng đối tượng?***

*4 tính chất: bao đóng (đóng gói), kế thừa, trìu tượng, đa hình*

***3. Tính bao đóng là gì, nó được thể hiện thế nào trong lập trình android?***

*là tính chất mà lớp ẩn đi các thuộc tính cũng như phương thức của nó, hạn chế việc truy cập từ bên ngoài. Tính bao đóng được thể hiện qua các access modifier (private, public, protected, default)*

***4. Có bao nhiêu loại access modifier, phạm vi truy cập của chúng?***

*có 4 loại:*

* *Public: truy cập không hạn chế từ bên ngoài lớp*
* *Protected: trong package và lớp con*
* *Default: trong package*
* *Private: chỉ trong phạm vi lớp*

***5. Có thể truy cập vào private method từ bên ngoài class không?***

*Có thể nếu sử dụng java reflection*

***6. Tính trìu tượng là gì, nó được thể hiện thế nào trong lập trình android?***

*Tính trìu tượng là tính chất cho phép tạo ra các class chứa các method chưa được implement (không có phần thân của method), các methods này sẽ được override lại ở các lớp con hoặc lớp implementation. Tính chất này giúp chúng ta tập trung vào những cốt lõi cần thiết của đối tượng thay vì quan tâm đến cách nó thực hiện. Trong lập trình android tính trìu tượng được thể hiện qua Interface và Abstract class*

***7. So sánh Interface và Abstract class?***

*Interface hỗ trợ đa kế thừa, chỉ chứa các abstract method, các methods này đều có access modifier là public, Abstract class không hỗ trợ đa kế thừa, có thể chứa các method và variable thường với các access modifier khác nhau, abstract method có thể không cần override ở class con nếu class con cũng là abstract class*

***8. Tính đa hình là gì, ghi đè (override) và nạp chồng (overloading) là gì?***

*Tính đa hình hiểu là trong từng trường hợp, hoàn cảnh khác nhau thì đối tượng có hình thái khác nhau. Ghi đè là một phương thức được viết lại hoặc triển khai ở lớp con hoặc lớp implementation, nạp chồng là một phương thức là trong một class tạo ra nhiều phương thức có cùng tên nhưng khác nhau (một phần) các tham số truyền vào và kiểu dữ liệu trả về.*

***9. So sánh, nêu các điểm mạnh của Kotlin so với Java trong lập trình Android?***

*Kotlin là một ngôn ngữ lập trình bậc cao được google đưa vào thay thế cho Java trong lập trình Android, Kotlin có gần như đầy đủ sức mạnh của Java công thêm những ưu điểm nổi trội giúp hỗ trợ cho lập trình viên:*

*- Cú pháp dễ học, dễ đọc, dễ hiểu, ngắn gọn với việc sử dụng when, let, apply, lambda, data class, biến lazy, extensions, smart cast, no more findViewById…*

*- Null-safety (optional variable) của Kotlin: giúp chúng ta thoát khỏi NullPointerExceptions, tránh những exception kiểu con trỏ null*

*- Functional programming (like Closure in swift): đây mà một điều tuyệt vời của kotlin so với Java giúp ta có thể truyền hàm số như 1 tham số của hàm số khác, tránh việc phải tạo quá nhiều Interface cho việc callback*

*- Function trong Kotlin hỗ trợ optional parameters hỗ trợ mạnh mẽ cho việc mở rộng, thêm parameters cho function*

***10. Các components chính của 1 ứng dụng Android?***

*Có 4 kiểu thành phần trong ứng dụng của Android bao gồm: Activities, Services, Content providers, Broadcast receivers.*

***11. Application là gì?***

*Lớp Application trong Android là lớp cơ sở trong ứng dụng Android chứa tất cả các component khác như activity và service. Lớp Application hoặc bất kỳ lớp con nào của lớp nó sẽ được khởi tạo trước bất kỳ lớp nào khác khi process cho ứng dụng của bạn được khởi tạo.*

***12. Context là gì? phân biệt App context và activity context?***

*Một Context là một đối tượng để phục vụ các xử lý liên quan tới hệ thống hay cung cấp thông tin của môi trường hệ thống. Nó cung cấp các dịch vụ như để giải quyết resources, lấy quyền truy cập vào cơ sở dữ liệu và preferences, ... Context có thể được coi là thành phần xử lý về môi trường mà ứng dụng của bạn đang hoạt động trong đó.*

*- Application Context: là một context gắn liền với vòng đời của một ứng dụng. Application context có thể được sử dụng khi bạn cần một context có vòng đời riêng biệt với context hiện tại hoặc khi bạn truyền một context ra ngoài phạm vi hoạt động của một activity.*

*- Activity Context: context này là có sẵn ở trong một activity, nó gắn liền với vòng đời của activity này. Bạn nên sử dụng loại context này khi phạm vi hoạt động bạn cần chỉ ở trong activity này hay bạn cần một context có vòng đời gắn liền với context hiện tại.*

***13. Giải thích các khái niệm: build type, product flavor, build variant?***

*Build type xác định cách một module được xây dựng, ví dụ như có sử dụng ProGuard hay không, package name, có enable minify không.... Build type thường được chia thành 2 loại là debug và release*

*Product flavor xác định những gì sẽ được tạo ra, ví dụ như tài nguyên nào được bao gồm trong bản build (các thư mục assets chứa file ảnh png, file xml).*

*Build variant là các tổ hợp kết hợp bởi Build type và Product flavor*

*3 thuộc tính trên được sử dụng để tạo ra các target hay môi trường riêng biệt phục vụ cho việc build app với các config khác nhau*

***14. AsyncTask là gì, khi sử dụng AsyncTask phương thức nào bắt buộc phải override, phương thức nào được thực hiện trên main thread, phương thức nào thực hiện ở background thread?***

*AsyncTask là một đối tượng thích hợp dùng để giao tiếp giữa một Background Thread với UI Thread, nó rất dễ dùng. Nó cho phép bạn thực hiện công việc ở background và sau đó cập nhật giao diện ở UI Thread. AsyncTask chính là sự tổ hợp của một Thread và một Handler. AsyncTask có 3 tham số Generic Type truyền vào là Params, Progress, Result và 4 phương thức onPreExecute(), doInBackground(), onProgressUpdate() và onPostExecute(). 3 tham số đều phải là kiểu Object, nếu không có thì để là Void. doInBackground chạy ở background, còn 3 phương thức còn lại chạy trên UI Thread. Khi bạn cho một lớp kế thừa từ AsyncTask bắt buộc bản phải override phương thức doInBackground*

***15. Những lưu ý khi sử dụng AsyncTask trong Activity?***

*Vòng đời của AsyncTask không được gắn với vòng đời của activity start nó, khi activity bị huỷ các tiến trình trong AsyncTask vẫn tiếp tục chạy vì thế có thể dẫn tới hiện tượng leak memory (asynctask giữ reference đến activity khiến nó không thể bị huỷ và thu hổi bởi GC) và tệ hơn là crash app trong trường hợp AsyncTask cố gắng truy cập vào các đối tượng đã bị huỷ (view, context) của activity khi cập nhật giao diện trong hàm onPostExecute. Giải pháp đưa ra là hạn chế sử dụng AsyncTask cho các tác vụ background cần thời gian dài, check trước null khi truy cập đến các thành phần của activity trong onPostExecute, sử dụng Service, sử dụng các thư viện support cho việc xử lý background task khác như RX, Viewmodel + Livedata…*

***16. Kiểu tham chiếu và tham trị trong Java (Kotlin) là gì. Hãy phân biệt chúng?***

*Tham trị (value type): kiểu dữ liệu nguyên thuỷ như String, Int, Byte, Long.., với kiểu dữ liệu này khi bạn truyền nó như params của hàm hoặc gán nó cho 1 biến khác giá trị sẽ được truyền theo dạng copy, khi thay đổi giá trị mới giá trị cũ sẽ không thay đổi*

*Tham chiếu (reference type): kiểu dữ liệu của các class, với kiểu dữ liệu này khi bạn tạo ra 1 instance của class tức là bạn đã tạo 1 địa chỉ cho biến đó trên vùng nhớ tên biến chỉ đóng vai trò là 1 tham chiếu tới địa chị này, khi truyền vào hàm hoặc gán cho một tên biến khác thì địa chỉ được truyền vì thế nếu thay đổi giá trị của biến mới thì biến cũ cũng thay đổi theo (thực thế là thay đổi giá trị của biến trên vùng nhớ)*

***17. Phân biệt equal và == ?***

*== và equal chỉ khác nhau đối với các biến kiểu tham chiếu****. ==*** *đơn giản giúp ta so sánh xem* ***2 biến đó có trỏ tới cùng 1 vùng nhớ hay không, equal là 1 method*** *của class dùng để* ***so sánh nội dung của 2 biến****. Ta có thể override method equal của 1 class để custom giá trị trả về*

*Kiểu tham trị(value type): Lưu trực tiếp giá trị trong bộ nhớ stack*

*Kiểu tham chiếu(Reference type): lưu địa chỉ của ô nhớ trong stack còn giá trị thực tế nằm trong heap*

***18. Intent là gì? có mấy loại Intent? Intent có thể chứa kiểu dữ liệu tham chiếu không?***

*Intent có thể hiểu đơn giản là một lớp được tạo ra để start và truyền dữ liệu tới Activity và Service hoặc sử dụng trong Broadcast Receiver. Android hỗ trợ 2 loại Intent là Intent tường minh (explicit) chỉ rõ mục tiêu mà intent được gửi đến và Intent không tường minh (implicit) hệ thống sẽ dựa vào action để chọn một đích đến phù hợp. Không thể truyền trực tiếp dữ liệu kiểu tham chiếu bằng intent mà phải sử dụng Serializable hoặc Parcelable.*

***19. So sánh Serializable và Parcelable?***

*+ Giống nhau: là quá trình chuyển đổi một đối tượng thành một luồng (stream) byte để lưu trữ vào bộ nhớ, có thể được tái tạo sau này khi cần, trong khi vẫn giữ trạng thái và dữ liệu ban đầu của đối tượng.*

*+ Khác nhau: Serializable là một standard Java interface trong khi Parcelable là một interface cụ thể của Android, bạn phải tự triển khai việc serialization. Parcelable nhanh hơn nhiều so với Serializable ( Serializable sử dụng cơ chế reflection và đó là một quá trình tương đối chậm. Cơ chế này cũng có xu hướng tạo ra rất nhiều đối tượng tạm thời và gây tiêu tốn tài nguyên khi bộ Garbage collection phải thu thập tất cả chúng).*

***20. Proguard là gì? Nên hay không nên sử dụng Proguard khi build ứng dụng?***

*ProGuard là một công cụ rút gọn (shrink), tối ưu hoá (optimize) và làm mờ (obfuscate) code. ProGuard là một phần của Android Gradle build process và được tích hợp trong Android SDK. Nên sử dụng Proguard vì:*

*- Giảm dung lượng của ứng dụng (shrink)*

*- Loại bỏ những lớp và phương thức dư thừa, các tài nguyên không được sử dụng (optimize)*

*- Làm cho ứng dụng khó đảo ngược bởi làm xáo trộn code (obfuscate)*

***21. Main Thread (UI Thread) là gì?******điểm cần lưu ý khi làm việc với Main thread?***

*Main Thread là thread đầu tiên được khởi chạy khi start ứng dụng, được dùng để xử lý giao diện cho ứng dụng. Vì vậy để tránh hiện tượng ANR (application not responding) hay block UI thread không nên thực hiện các tác vụ nặng, tốn nhiều thời gian trên Main thread. Ngoài ra request network trên main thread sẽ dẫn tới ngoại lệ NetworkOnMainThreadExeption gây crash app.*

***22. Garbage collector là gì? Nó hoạt động như thế nào?***

*Garbage collector (trình thu gom rác) là tiến trình quản lý bộ nhớ và thu gom rác của hđh Android. Khi một đối tượng không còn được tham chiếu và do đó không thể truy cập được bằng code trong ứng dụng Garbage collector sẽ loại bỏ nó và lấy lại bộ nhớ không sử dụng.*

***23. Phân biệt vùng nhớ Stack và Heap?***

*Stack là một vùng nhớ được sử dụng để lưu trữ các tham số và các biến local của phương thức mỗi khi một phương thức được gọi ra*

*Heap là một vùng nhớ trong bộ nhớ được sử dụng để lưu trữ các đối tượng khi từ khóa new được gọi ra, các biến static và các biến toàn cục (biến instance).*

*Giống nhau: đều là vùng nhớ được hđh cấp phát cho ứng dụng và tồn tại trong RAM*

*Khác nhau: ngoài việc khác nhau về định nghĩa và cách sử dụng như trên thì điểm khác nhau cơ bản nữa là vùng nhớ Heap là cố định trong khi Stack có thể co giãn (phình ra khi method được gọi và co lại khi kết thúc). Việc truy cập vào vùng nhớ Stack là nhanh hơn vùng nhớ Heap, Khi vùng nhớ Heap bị đầy sẽ bắn ra ngoại lệ OutOfMemoryException, khi Stack bị đầy là StackOverFlowException*

***24. Phạm vi sử dụng của các loại biến trong Android (static, toàn cục, local)? cần chú ý gì khi tạo biến liên quan tới phạm vi của biến?***

*biến static: toàn bộ chương trình, biến toàn cục được sử dụng trong class, hoặc thông qua instance, method của class đó, biến local được sử dụng trong hàm*

*Lưu ý quan trọng khi tạo và sử dụng biến là*

*- giữ cho phạm vi sử dụng của biến nhỏ nhất có thể: nếu có thể dùng local thì ko nên tạo biến toàn cục, nếu có thể dùng biến toàn cục thì ko nên tạo biến static*

*- giữ cho thời gian sống của biến ngắn nhất có thể*

***25. Java là ngôn ngữ đơn kế thừa hay đa kế thừa?***

*Java là ngôn ngữ đơn kế thừa (một lớp con chỉ có thể extend 1 lớp cha duy nhất) nhưng ta có thể sử dụng Interface để bổ xung có khiếm khuyết trên (1 lớp có thể implement vô số Interface)*

***26. Tại sao lại phải đặt String trong file String.xml hoặc Constant mà không “Hard-code” trực tiếp trong code? Những String nào nên để trong file xml, string nào nên define trong file Constant?***

*Việc tránh hard-code là một phần quan trọng của coding convention của Android hay bất kỳ ngôn ngữ nào nhằm mục đích tái sử dụng cũng như dễ dàng sửa đổi về sau, trong Android việc lưu String trong file String.xml còn để tiện lợi cho việc dịch (localize) sang các ngôn ngữ khác sau này. Các string cần được dịch nên được để trong file String.xml, các string không cần dịch, là các key (preference key, intent key), url… nên được để trong Constant*

***27. Nên sử dụng vector image hay png trong lập trình android?***

*luôn sử dụng vector image nếu có thể với các lý do sau:  
- Nhẹ hơn khá nhiều so với file png*

*- Dễ dàng chỉnh sửa, update*

*- Rất tốt cho việc sử dụng với các kích thước màn hình khác nhau*

*Tuy nhiên có 1 số trường hợp vẫn bắt buộc phải sử dụng ảnh png: nếu ứng dụng support android version cũ, icon cho notification, icon cho app…*

***28. Dagger là gì, tại sao phải (nên) dùng dagger?***

*Dagger 2 - là một dependency injection framework, được sử dụng để implement nguyên lý Dependency Inversion*

*Vậy Dependency injection, Dependency Inversion là gì*

*Dependency Inversion: Đây là một nguyên lý để thiết kế và viết code. (chữ D trong S.O.L.I.D)*

*Dependency Injection: Đây là một cách để hiện thực Dependency Inversion. Các module phụ thuộc (dependency) sẽ được inject vào module cấp cao.*

*Dagger (DI) được dùng để làm giảm sự phụ thuộc giữa các module, dễ dàng hơn trong việc thay đổi module, bảo trì code và testing.*

***29. EventBus vs BroadcastReceiver?***

*Đều là công cụ giúp cho việc gửi và nhận dữ liệu trong ứng dụng android, nhưng EventBus mạnh mẽ, tiện lợi và linh hoạt hơn khá nhiều trong việc sử dụng, cụ thể các ưu điểm của EventBus so với BroadcastReceiver như sau:*

*- Eventbus có thể gửi được gửi được mọi kiểu dữ liệu (classes) trong khi BR chỉ gửi và nhận thông qua intent nên chỉ support kiểu nguyên thuỷ hoặc serializeable hoặc parcelable class*

*- Việc triển khai Eventbus đơn giản hơn khá nhiều so với BR*

*- Eventbus thuận lợi hơn trong việc gỡ lỗi, đọc hiểu và bảo trì code*

***30. Phân biệt list và set?***

*List là một chuỗi các phần tử được sắp xếp trong khi Set là một danh sách các phần tử riêng biệt không có thứ tự. List cho phép các phần tử giống nhau còn set thì không, list có thể truy cập theo position còn set thì không.*

***31. Có phải Notification luôn được xử lý trong hàm onMessageReceived của class FirebaseMessagingService?***

*Hầu như nhưng không phải tất cả Notification có 2 loại là data và non-data, trong trường hợp app in background non-data notification sẽ được xử lý bởi system tray. Khi user click vào notification loại này launch Activity sẽ được gọi.*

***32. So sánh các local database: Room, Realm, Sqlite?***

*Đây đều là các local database được sử dụng rộng rãi trong các ứng dụng Android, chúng đều có ưu và nhược điểm riêng*

*Sqlite: có lẽ là một trong những database đầu tiên được dùng trong android, hiện tại ngày càng ít được sử dụng do tính chất phức tạp của nó, Để làm việc với SQLite, ta phải thực hiện rất nhiều thao tác: xây dựng khung chương trình, mô hình, helpers, truy vấn, blah blah. Thực sự việc này tốn rất nhiều thời gian trong tiến trình phát triển phần mềm. Những người mới tiếp cận với SQLite sẽ phải bỏ ra nhiều giờ loay hoay trong khởi tạo, nhập xuất, xây dựng câu truy vấn và đặc biệt là code sẽ rất khó quản lý.*

*Realm: Một thư viện tương đối nhanh và thuận tiện, việc tạo và truy cập database đơn giản. Tài liệu nhiều, cộng đồng sử dụng lớn. Có lẽ, một trong những lựa chọn tốt nhất để lưu trữ dữ liệu trên thiết bị di động tại thời điểm này, điểm trừ chỉ có thể là tăng kích thước của file apk lên 2,5 MB.:*

*Room: được đưa vào Google I / O 2017 và hiện tại nằm trong bộ Android Jackpack được tối ưu để làm việc với cơ sở dữ liệu trên HĐH Android cùng với các components khác như Livedata, Viewmodel. Room cung cấp một lớp trừu tượng trên SQLite cho phép truy cập cơ sở dữ liệu không cần sử dụng truy vấn SQL trong khi khai thác toàn bộ sức mạnh của SQLite. Vì giải pháp này, được cung cấp và hỗ trợ bởi Google, nó sẽ nhanh chóng trở nên phổ biến và là đối thủ mạnh của Realm trong tương lai*

*33. So sánh* ***SharedPreferences với database (Sqlite, Room, Realm)?***

*SharedPreferences: Dùng để lưu các giá trị đơn giản ở dạng key/value. Muốn truy cập vào giá trị trong SharedPreferences bạn chỉ cần biết key, điều này giúp cho việc truy cập dữ liệu nhanh chóng và dễ dàng, nhưng bạn ko thể lưu dữ liệu dạng class, cấu trúc và không thể thực hiện tìm kiếm*

*Database: Dùng để lưu trữ một lượng lớn dữ liệu có cấu trúc rõ ràng, điều này giúp cho việc query database trở lên dễ dàng, tất nhiên việc đọc dữ liệu sẽ chậm hơn so với sử dụng SharedPreferences*

***34. Có an toàn không nếu bạn lưu các thông tin nhạy cảm ví dụ như thông tin thẻ tín dụng, private key vào database hoặc SharedPreferences của app?***

*Không an toàn vì toàn bộ các dữ liệu này đều được lưu lại trong điện thoại, khi điện thoại bị tấn công (with root level access) các thông tin này hoàn toàn có thể bị đánh cắp. Giải pháp đưa ra là có thể lưu các thông tin này trên server (nếu được phép) hoặc sử dụng Encode khi lưu và decode khi lấy ra sử dụng bằng keystore của app*

*35. Khi xoá ứng dụng SharedPreferences và database có được lưu lại không? giải pháp là gì khi bạn muốn sử dụng tiếp dữ liệu khi user cài lại app? Toàn bộ database và dữ liệu lưu trong SharedPreferences sẽ bị xoá, nếu muốn tiếp tục truy cập vào dữ liệu khi cài lại có thể chọn phương pháp lưu vào files*

***36. Nêu những hiểu biết của bạn về Material design?***

*Material Design là một phong cách thiết kế mới được Google giới thiệu cùng lúc với phiên bản Android 5.0 Lollipop*

*Phong cách thiết kế Material Design nhắm đến những đường nét đơn giản, sử dụng nhiều mảng màu đậm nổi bật, các đối tượng đồ họa trong giao diện dường như: “trôi nổi” lên. Ngoài ra, nó còn bao gồm cả những hiệu ứng chuyển động tự nhiên khi các nút, menu hiện diện trên màn hình. Tất cả đều nhằm mang lại cho người dùng trải nghiệm mới mẻ hơn, thú vị hơn và gần giống đời thực hơn.*

***37. Sự khác nhau cơ bản của các method GET, PUT, POST?***

***38. Reactive programming (RX) là gì, các thành phần chính của RX, tại sao nên sử dụng RX trong ứng dụng Android?***

***39. Service trong android là gì, có mấy loại service, service chạy trên MainThread hay Background Thread?***

***40. Khi tạo Singleton class, làm thế nào để tránh việc tạo ra nhiều hơn 1 instance?***

***41. Sự khác nhau cơ bản giữa ListView và RecyclerView, tại sao nên sử dụng RecyclerView để hiện thị list item***

***42. Số lượng ViewHolder được tạo ra trong 1 Adapter của RecyclerView là bao nhiêu, khi scroll RecyclerView thì method nào được gọi.***

***43. Theo bạn hiểu Coding Convention là gì? bạn đã áp dụng nó thế nào trong các dự án đã tham gia?***

***44. Trong các dự án bạn đã tham gia bạn đã sử dụng những công nghệ gì, gặp phải khó khăn gì và bạn đã giải quyết nó như thế nào?***

**B. Nhóm câu hỏi về cấu trúc dự án**

***45. Bạn đã từng làm việc với các Design Pattern (DP) nào? So sánh ưu, nhược điểm của mô hình MVC, MVP, MVVM?***

***46. Khi bắt đầu một dự án bạn sẽ lựa chọn DP nào? Tại sao?***

***47. Theo bạn phần base của một project có tầm quan trọng thế như thế nào? Bạn có kinh nghiệm gì trong việc dựng base của dự án?***

***48. Những điểm cần lưu ý khi lựa chọn sử dụng một thư viện trong project?***

***49. Nguyên tắc S.O.L.I.D trong lập trình là gì? Khi xây dựng project bạn có ý thức như thế nào trong việc áp dụng các nguyên tắc này?***

***50. Theo bạn tại sao trong những năm gần đây Google đưa ra một loạt các khái niệm cũng như thư viện mới như: Room, LiveData, Jetpack, ViewModel, Databinding?***

**C. Nhóm câu hỏi về Teamwork - flow làm việc:**

***51. Bạn có kinh nghiệm sử dụng các thư viện quản lý source code nào? Khi nhận 1 task mới bạn sẽ code trên branch nào? Code xong bạn sẽ làm gì?***

***52. Bạn đã bao giờ gặp vấn đề conflict khi merge code chưa, bạn xử lý thế nào trong trường hợp này?***

***53. Bạn xử lý thế nào trong trường hợp có xung đột ý kiến với thành viên khác trong team, nếu bạn là team leader và các thành viên trong team có xung đột với nhau?***

***54. Bạn đã làm việc ở team mobile nhiều nhất là bao nhiêu thành viên? Ít nhất là bao nhiêu thành viên? Công ty có size bao nhiêu người?***

***55. Nếu là team lead thì bạn quản lý nhiều nhất bao nhiêu người? Các công việc bạn là quản lý những gì? Có định hướng phát triển team như thế nào?***

***56. Bạn code ở những công ty cũ thì có theo flow không? Theo bạn flow chuẩn để thực hiện code 1 chức năng là như thế nào?***

**D. Câu hỏi chung**

***56. Tại sao bạn lại chuyển việc? Bạn đã chuyển bao nhiêu lần kể từ khi đi làm rồi?***

***57. Chỗ ở hiện tại ở đâu? Nếu xa công ty thì có tiện đi lại không?***

***58. Tại sao bạn biết đến Ominext? Bạn đã biết những gì về công ty?***

***59. Cảm nhận của bạn khi đến công ty là gì? Bạn có kỳ vọng gì khi làm việc ở Ominext hay không?***

***59. Bạn có sở trường, sở đoản gì? có chơi game, chơi môn thể thao gì không?***

***60. Bạn có câu hỏi gì về công ty hay không?***

***61. Mức lương offer của bạn là bao nhiêu? Bao giờ thì bạn có thể làm việc được?***

***62. Định hướng tương lai gần, tương lai xa của bạn là như thế nào?***