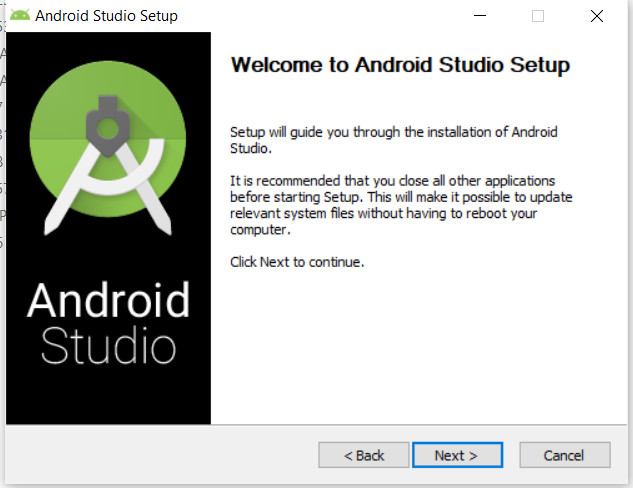
Họ và Tên: Bùi Chí Hải

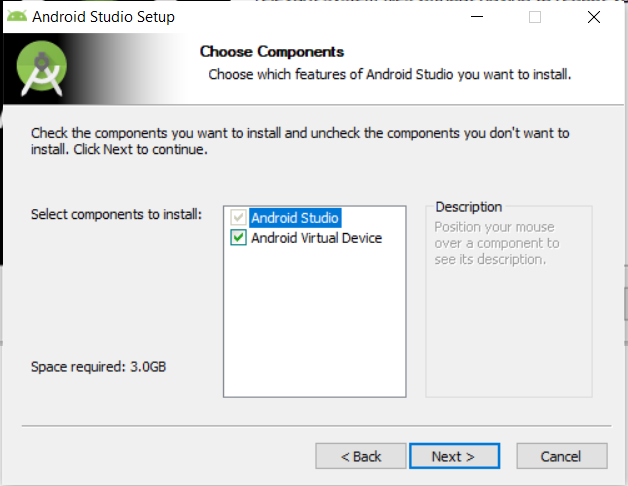
MSSV: B1906314

**BÀI TẬP TUẦN 4**

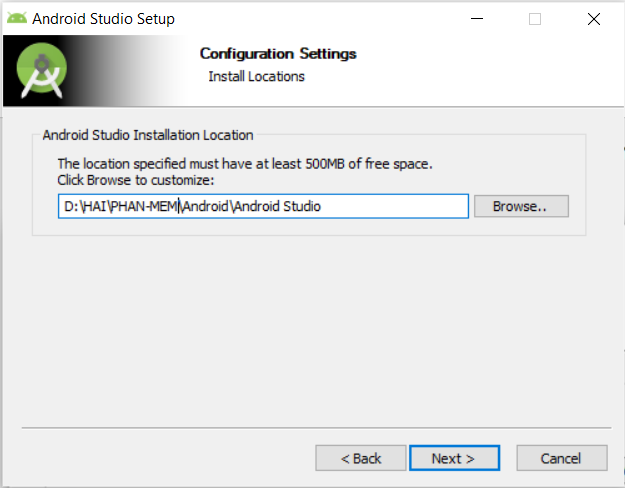
1. **Cài đặt Android studio**



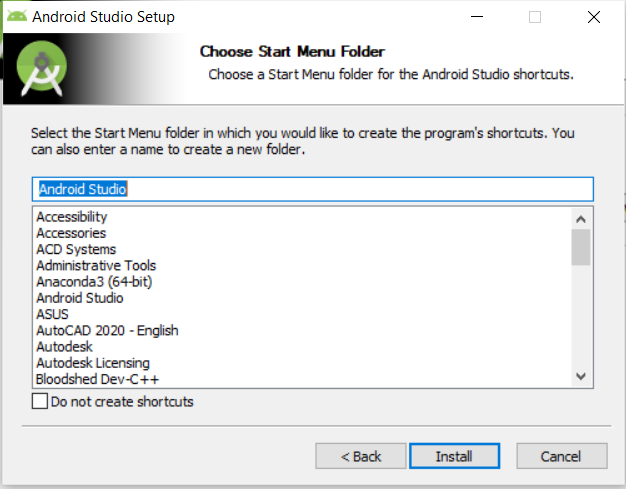
Hình 1: khởi chạy Anndroid studio bằng "Run as administrator" và bấm Next



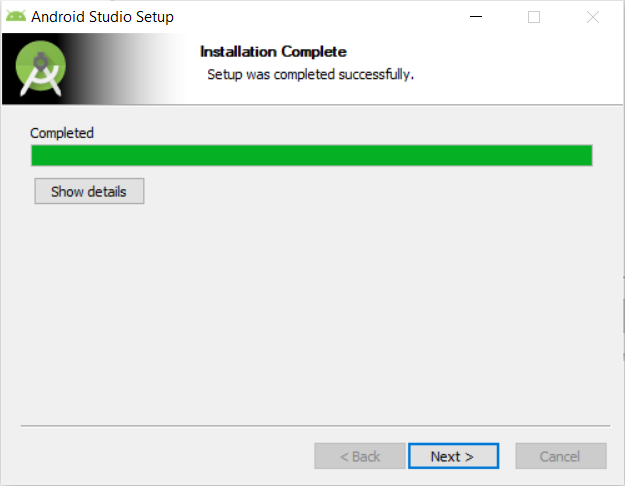
Hình 2: Ô "Android Virtual Decvice" đã được tích và tiếp tục bấm Next



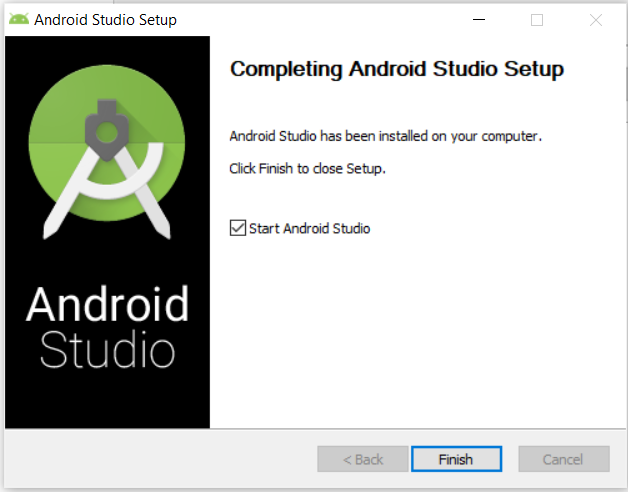
Hình 3: Chọn đường dẫn để lưu "Android studio" vào ổ đĩa và tiếp tục bấm Next



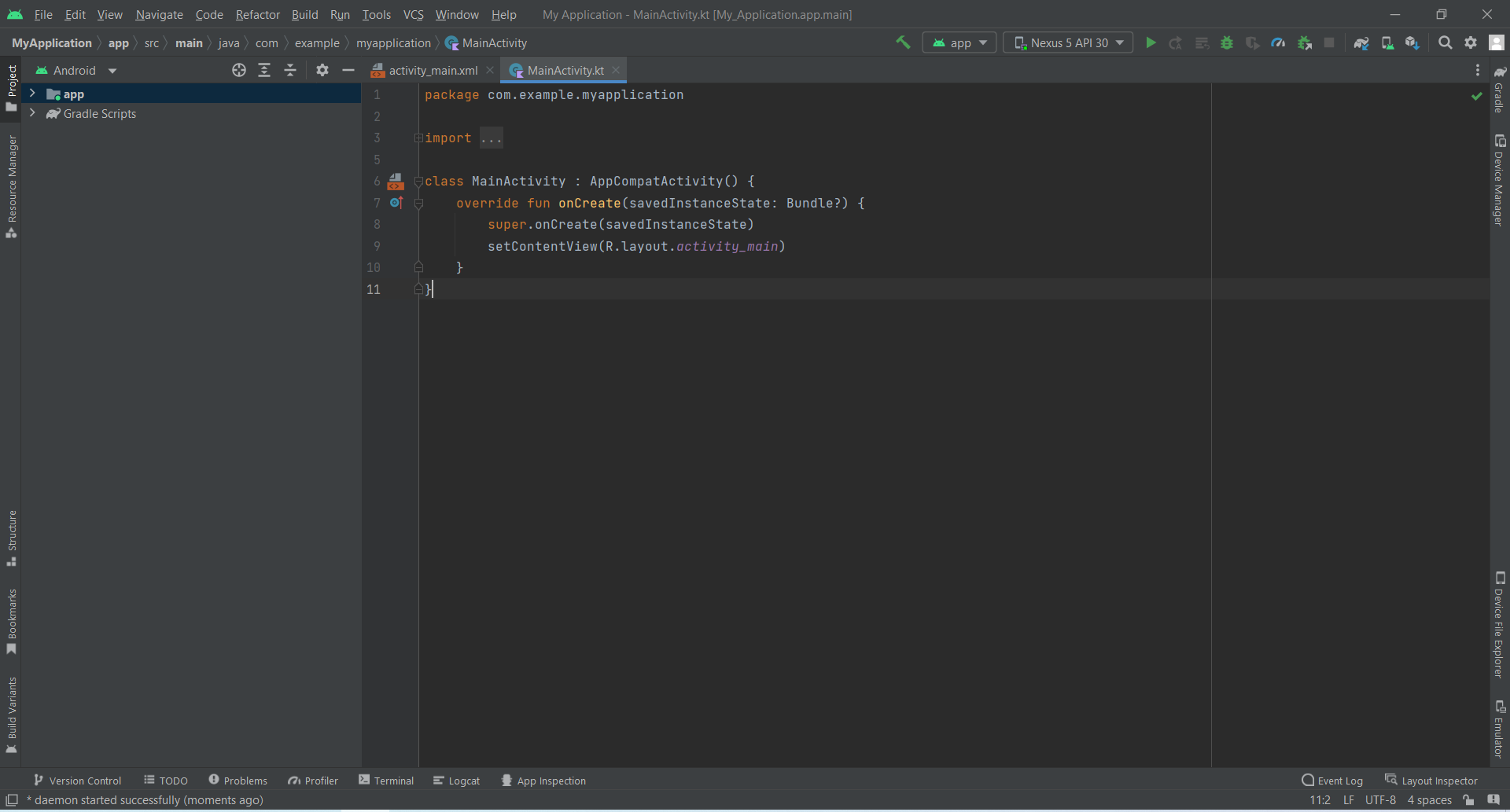
Hình 4: Ta bấm nút "Install" để cài Anroiid studio vào ổ đĩa



Hình 5: Khi cài xong ta tiếp tục bấm Next

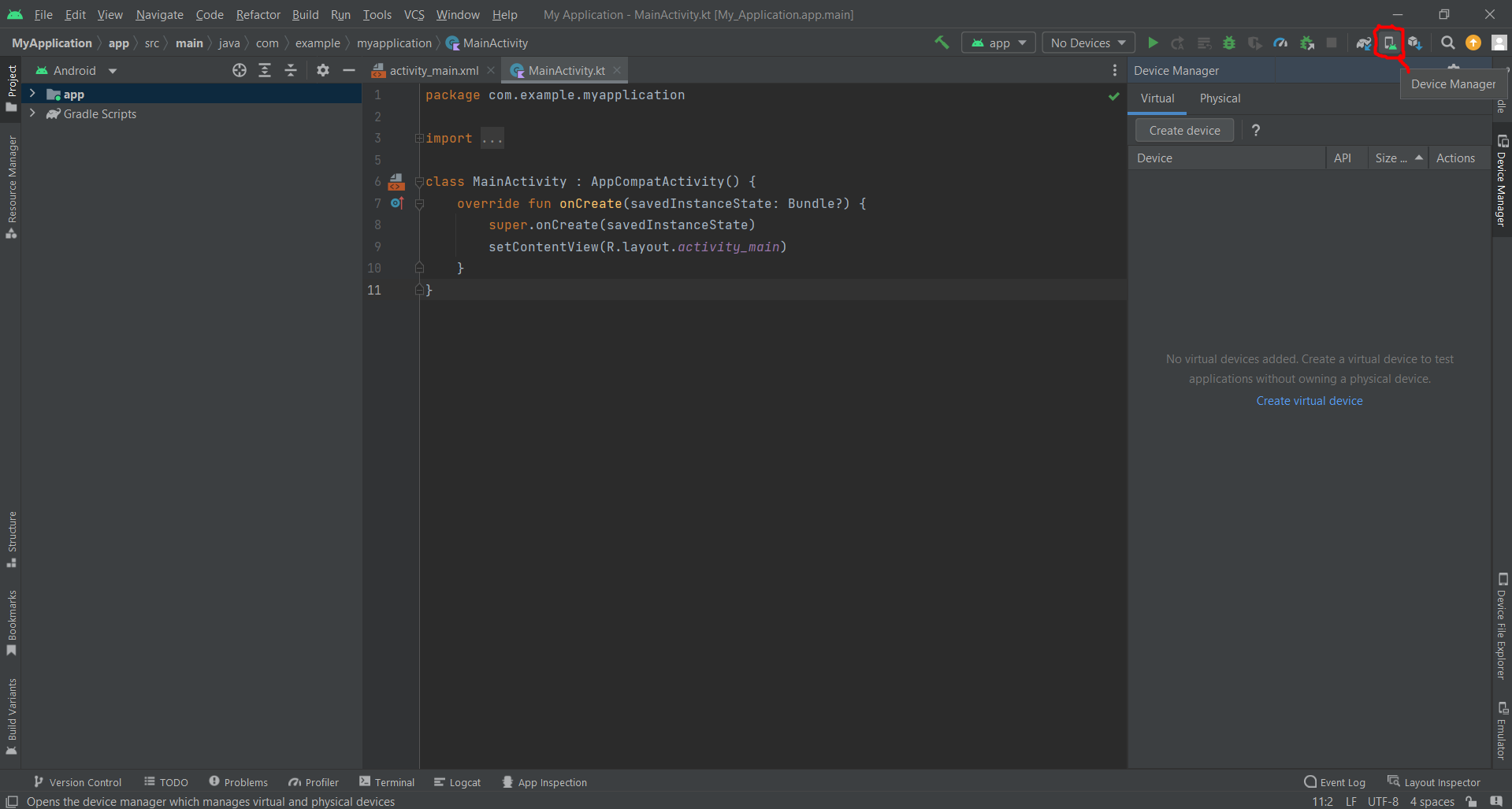


Hình 6: Ta bấm vào nút "Finish" để hoàn thành việc cài đặt Android studio

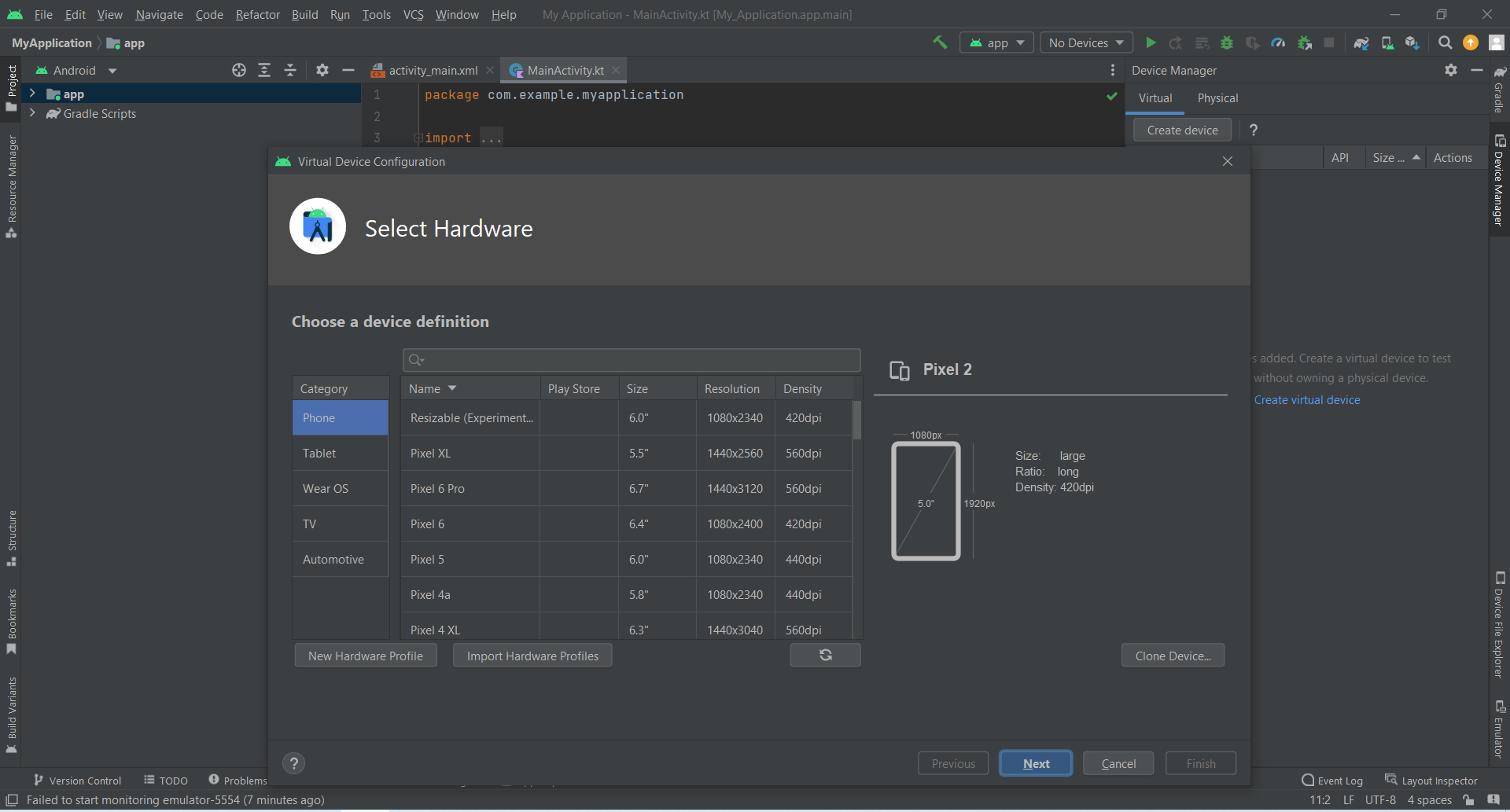


Hình 7: Và đây là giao diện khi khởi chạy "Android studio" lần đầu tiên

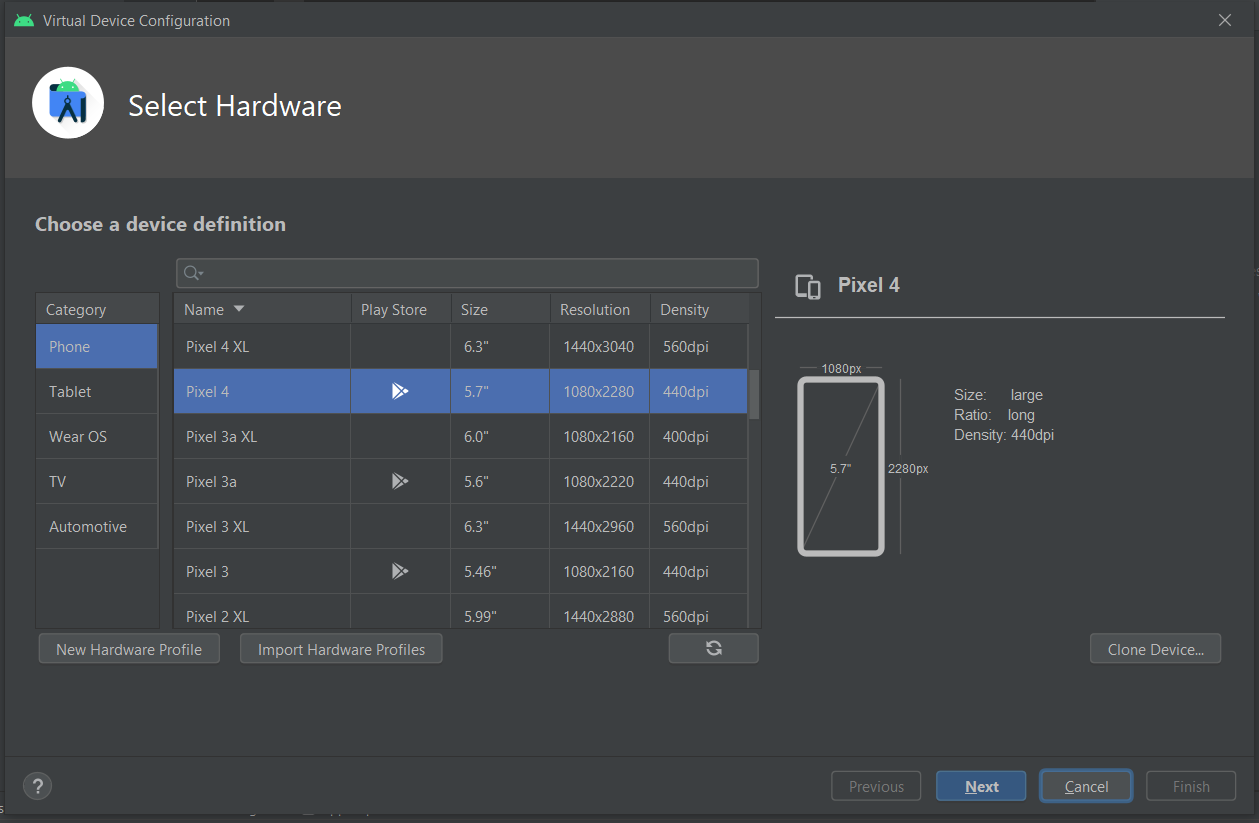
1. **Cài đặt máy ảo**



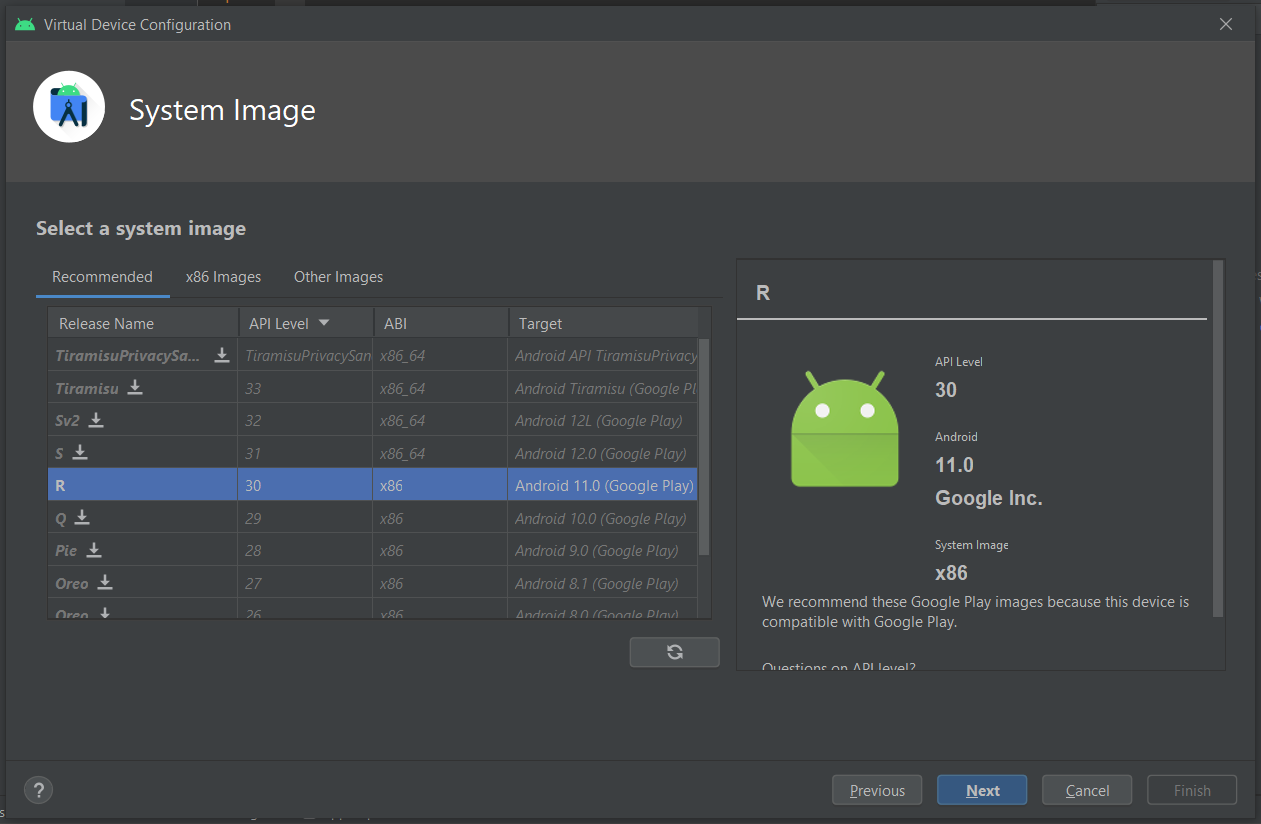
Hình 8: Bấm vào biểu tượng điện thoại "Device Manager" phía bên phải màn hình và chọn tiếp Create device



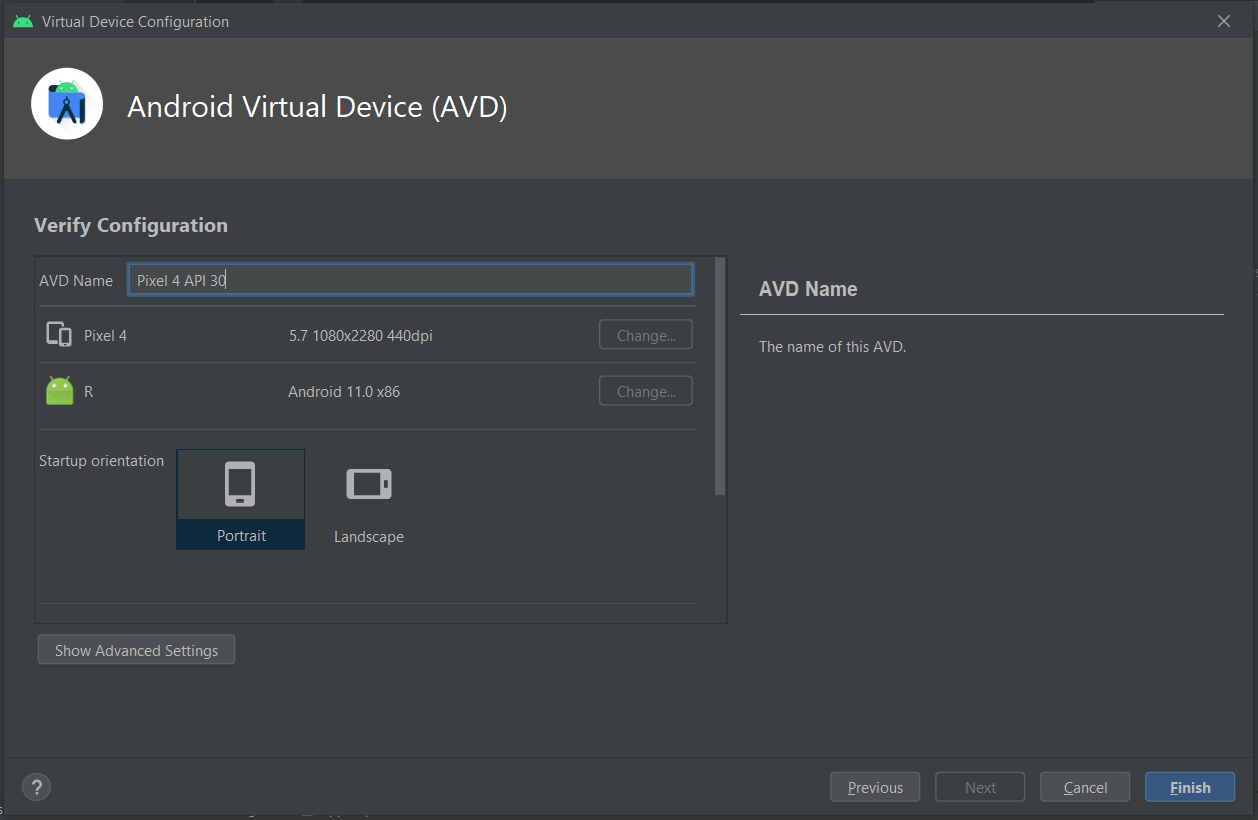
Hình 9: Bảng "Select Hardware" được hiển thị để chọn thiết bị máy ảo



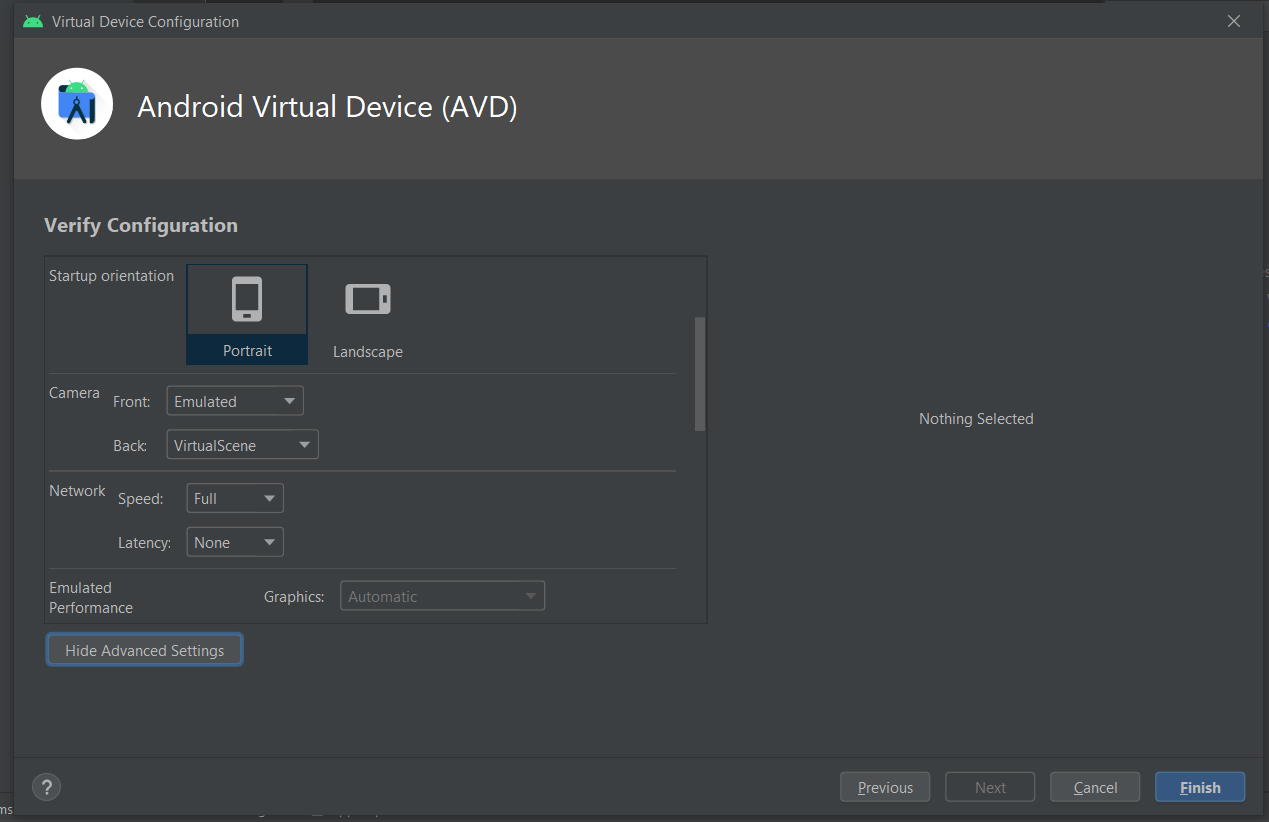
Hình 10: Chọn máy ảo mình mong muốn nhưng phải có hỗ trợ Chplay từ thiết bị và bấm Next



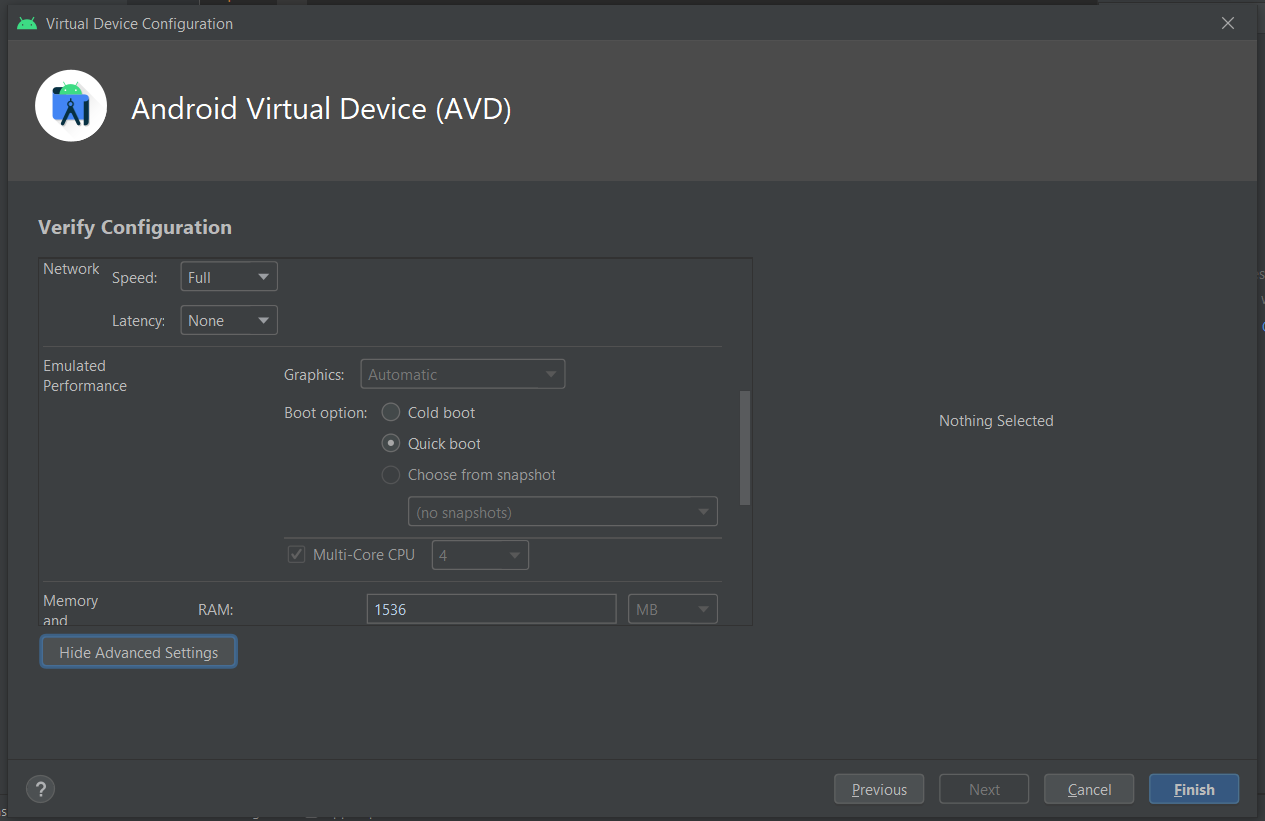
Hình 11: Chọn phần mềm cài cho máy ảo Và bấm Next



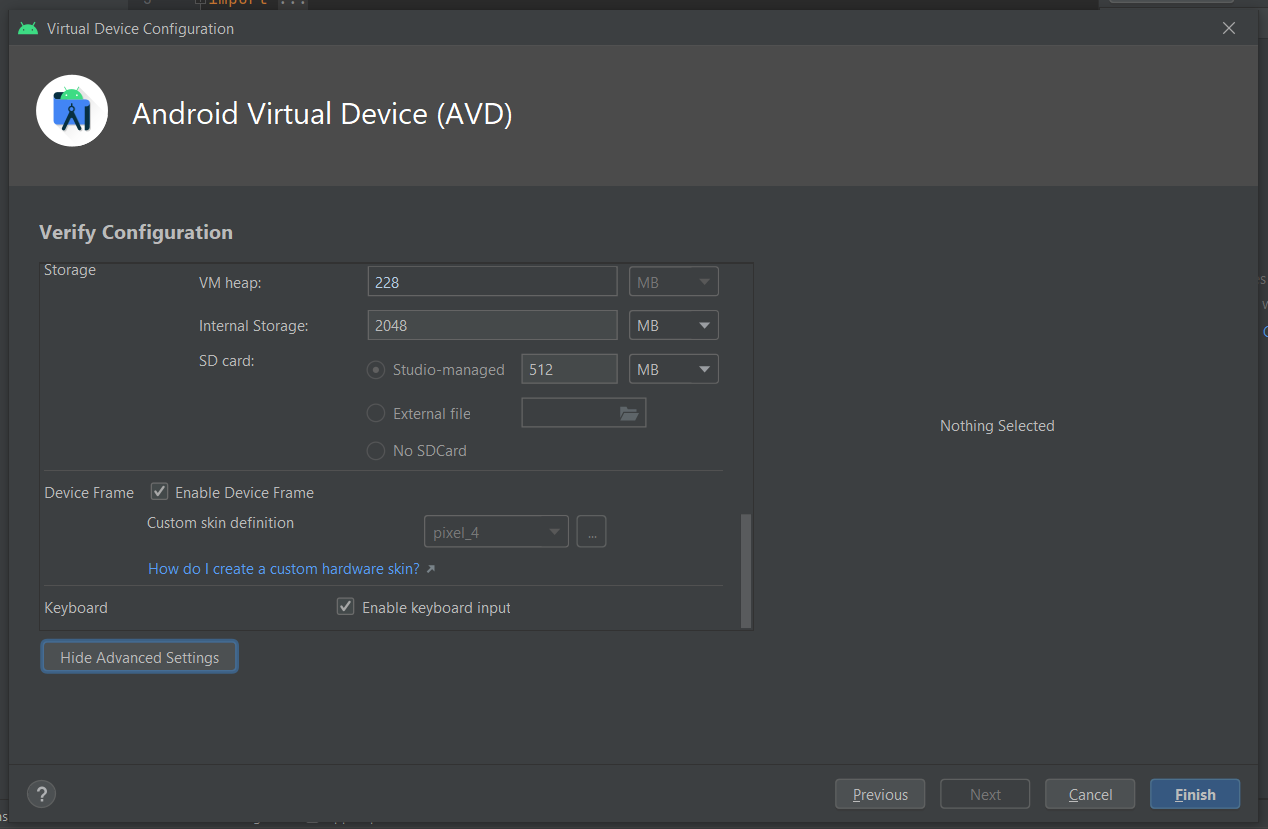
Hình 12: Bảng "Android Virtual Device (AVD)" ta để các giá trị mặt định



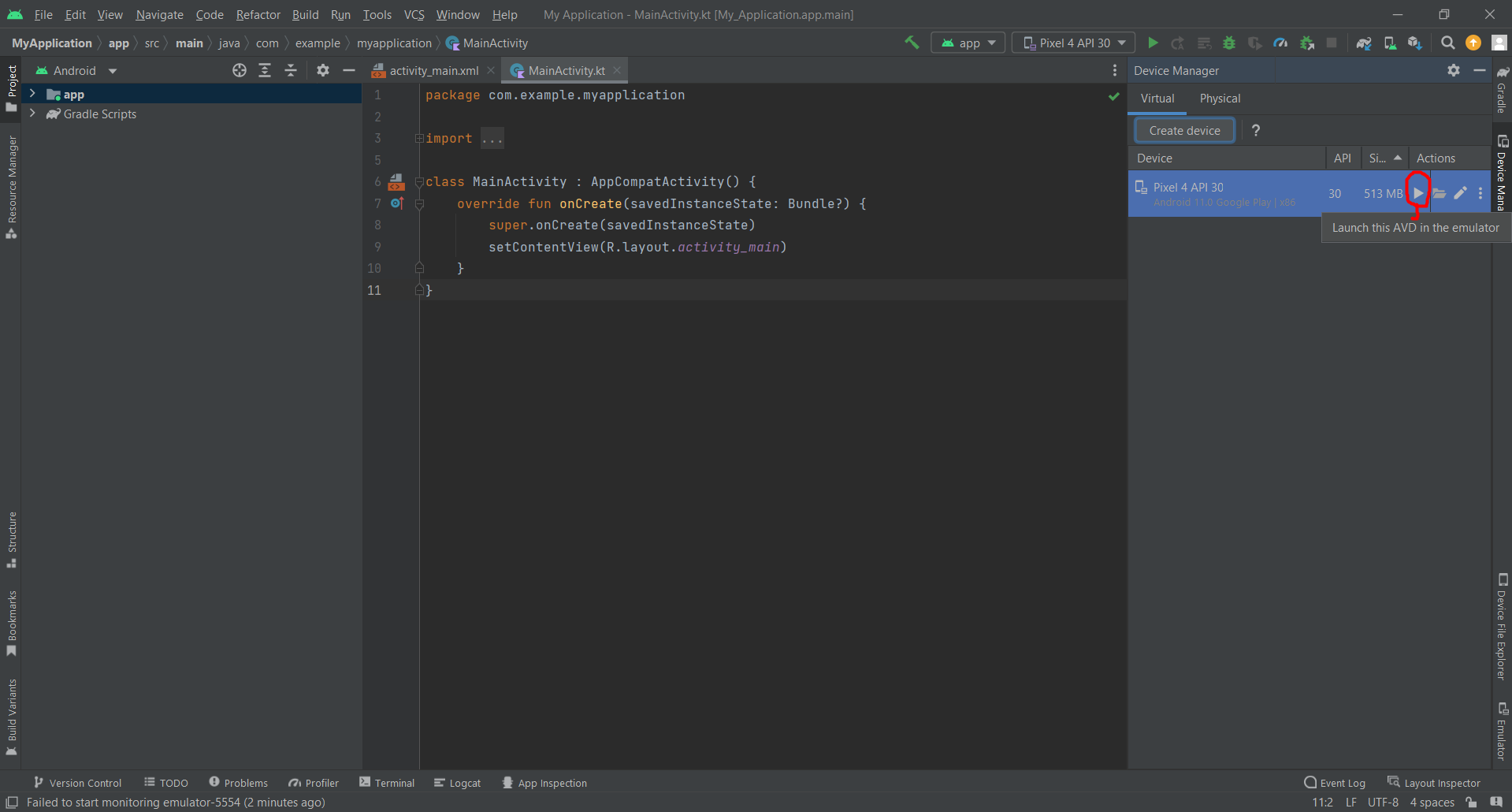
Hình 13: Bảng "Android Virtual Device (AVD)" ta để các giá trị mặt định



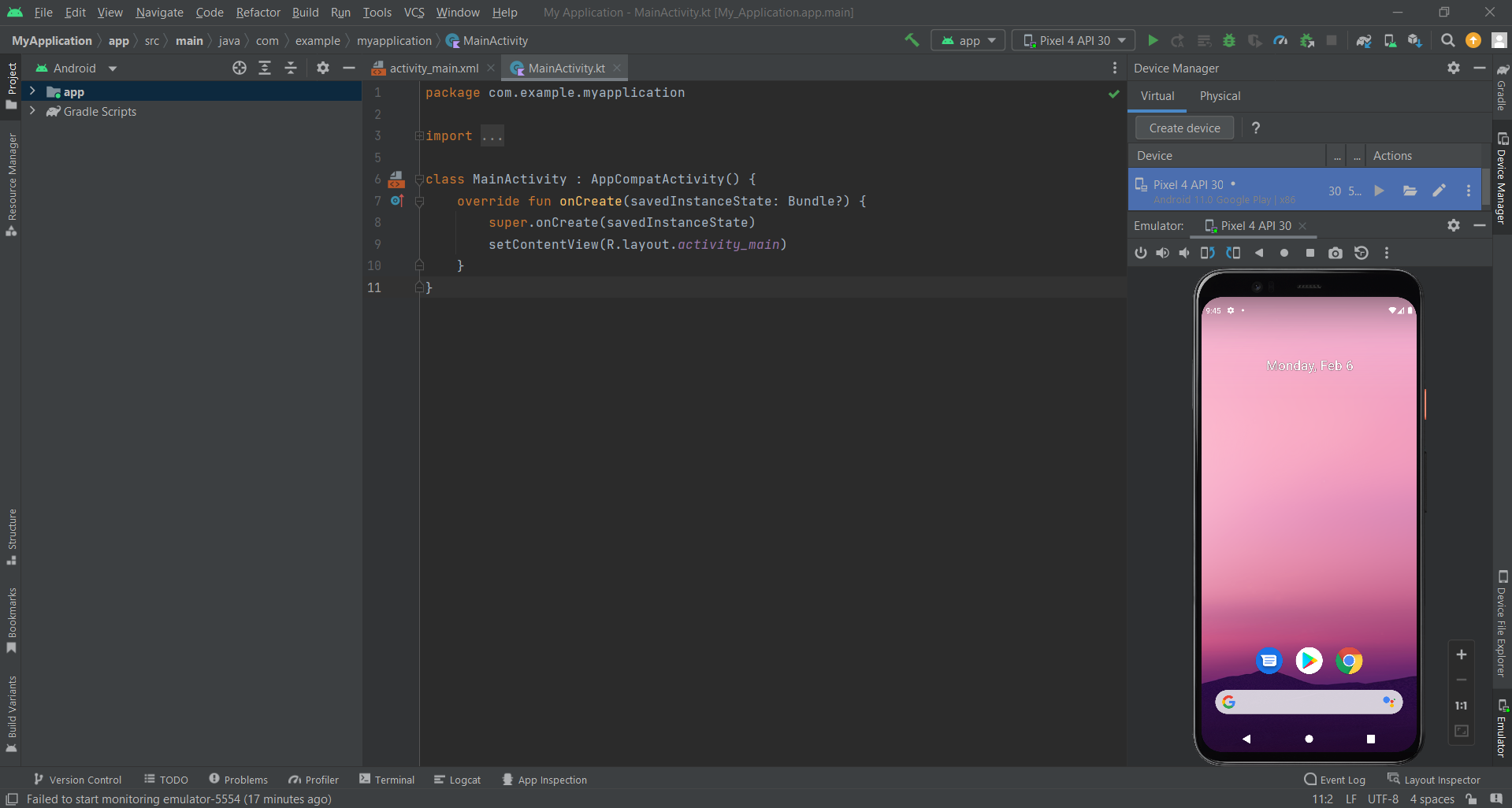
Hình 14: Bảng "Android Virtual Device (AVD)" ta để các giá trị mặt định



Hình 15: Bảng "Android Virtual Device (AVD)" ta để các giá trị mặt định và bấm vào nút "Finish" để cài máy ảo

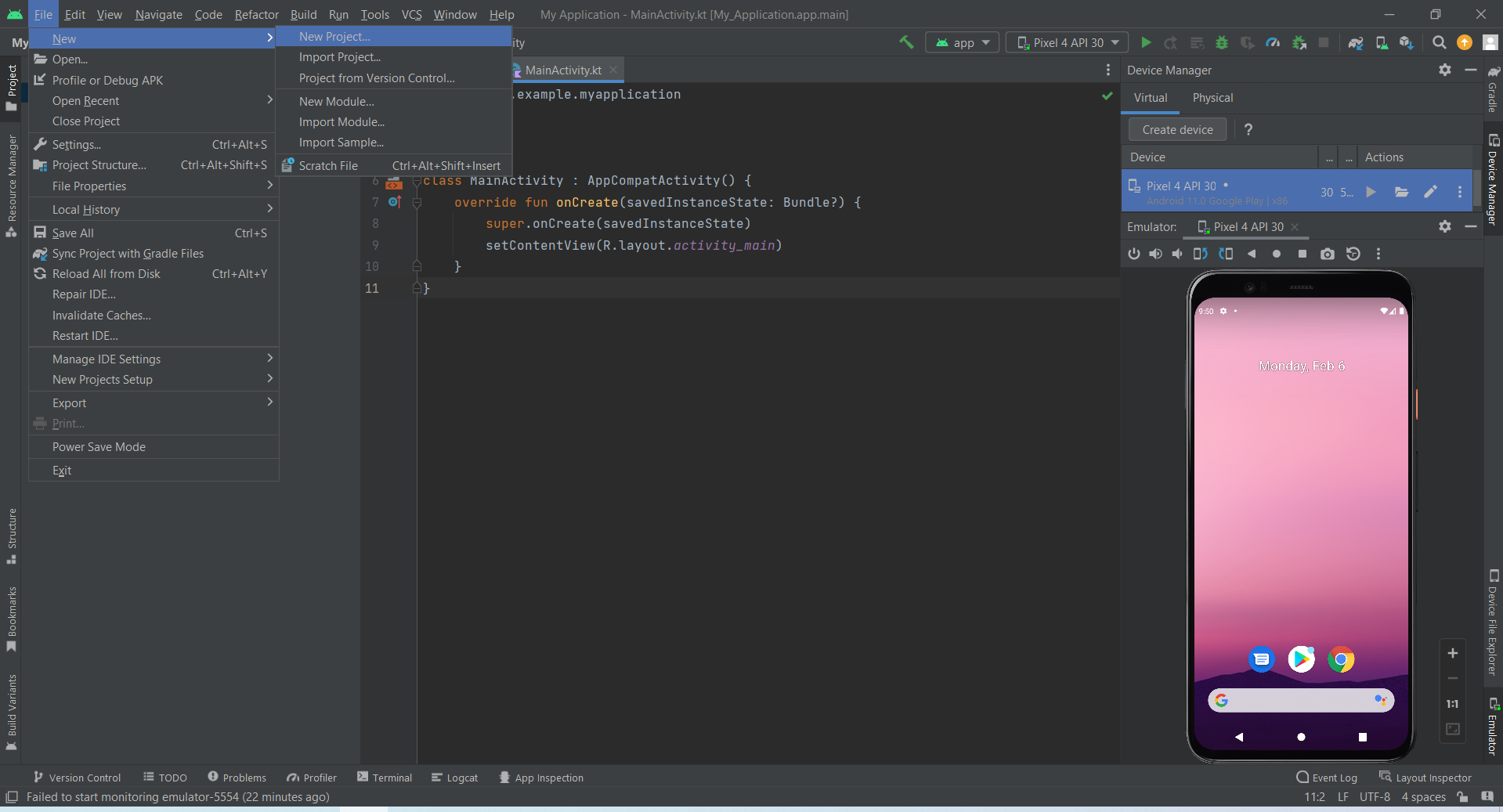


Hình 16: Máy ảo đã được cài đặt và bấm vào hình tam giác để khởi động máy ảo

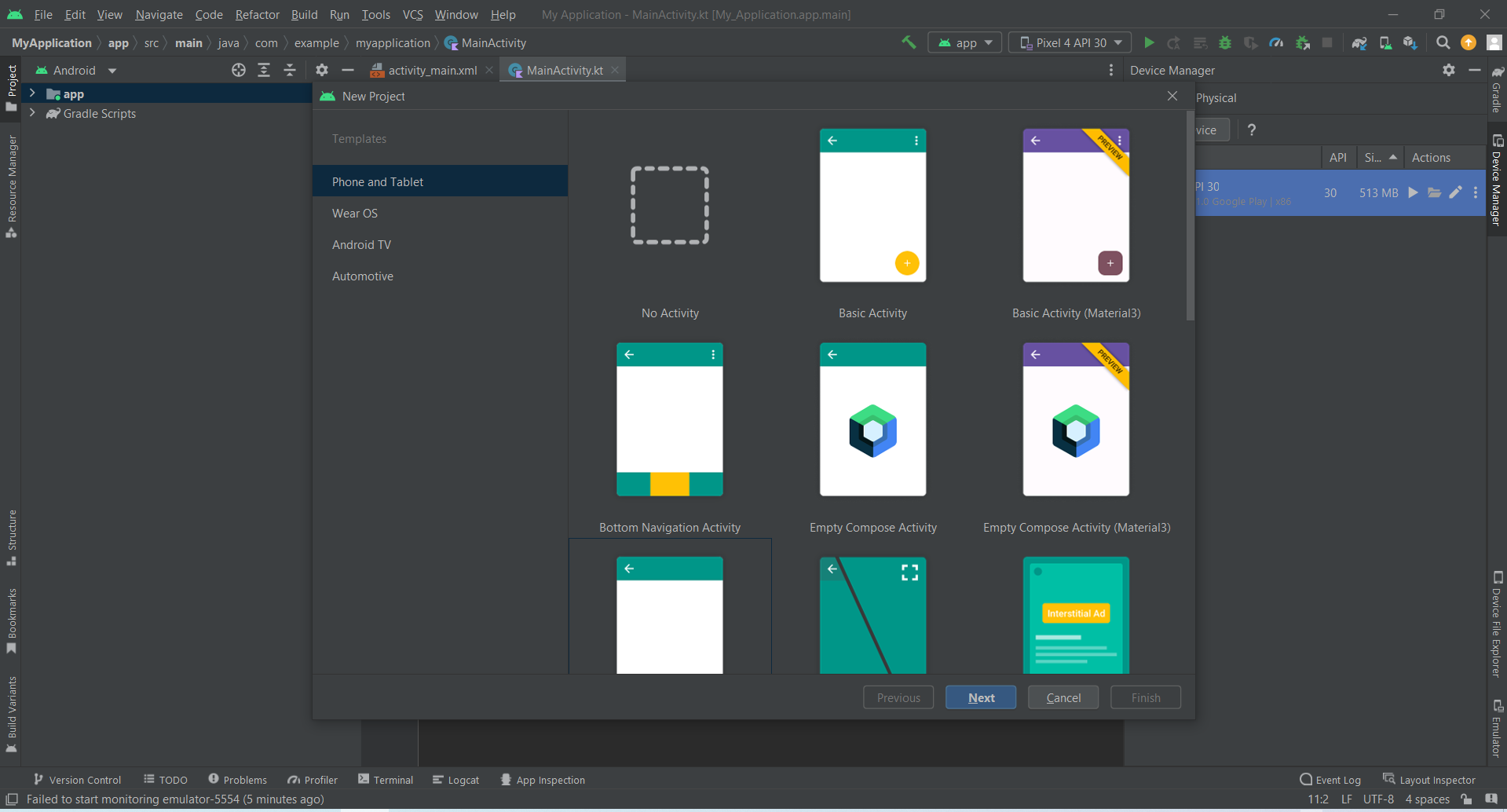


Hình 17: Máy ảo đã được khởi động thành công và được hiển thị bên phải màn hình

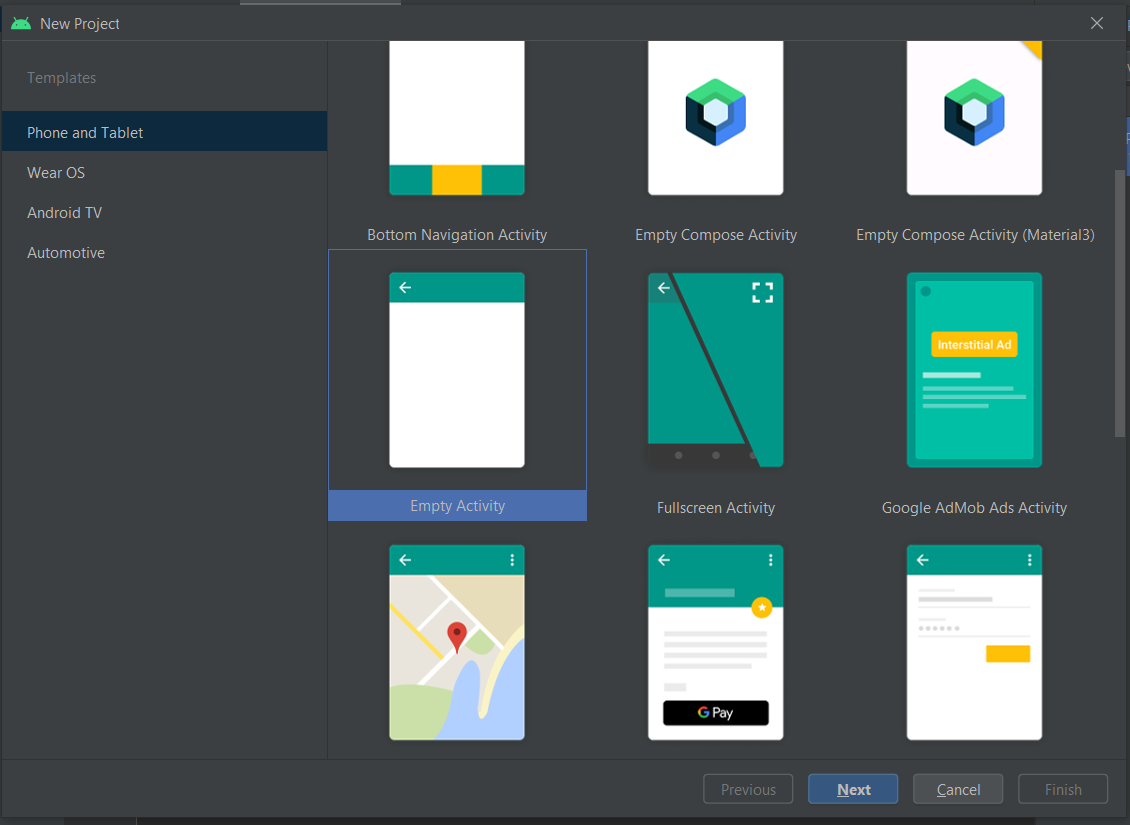
1. **Tạo project "Hello World" run trên máy ảo (Tạo tên project theo cấu pháp: MSSV\_HoTen\_HelloWorld)**



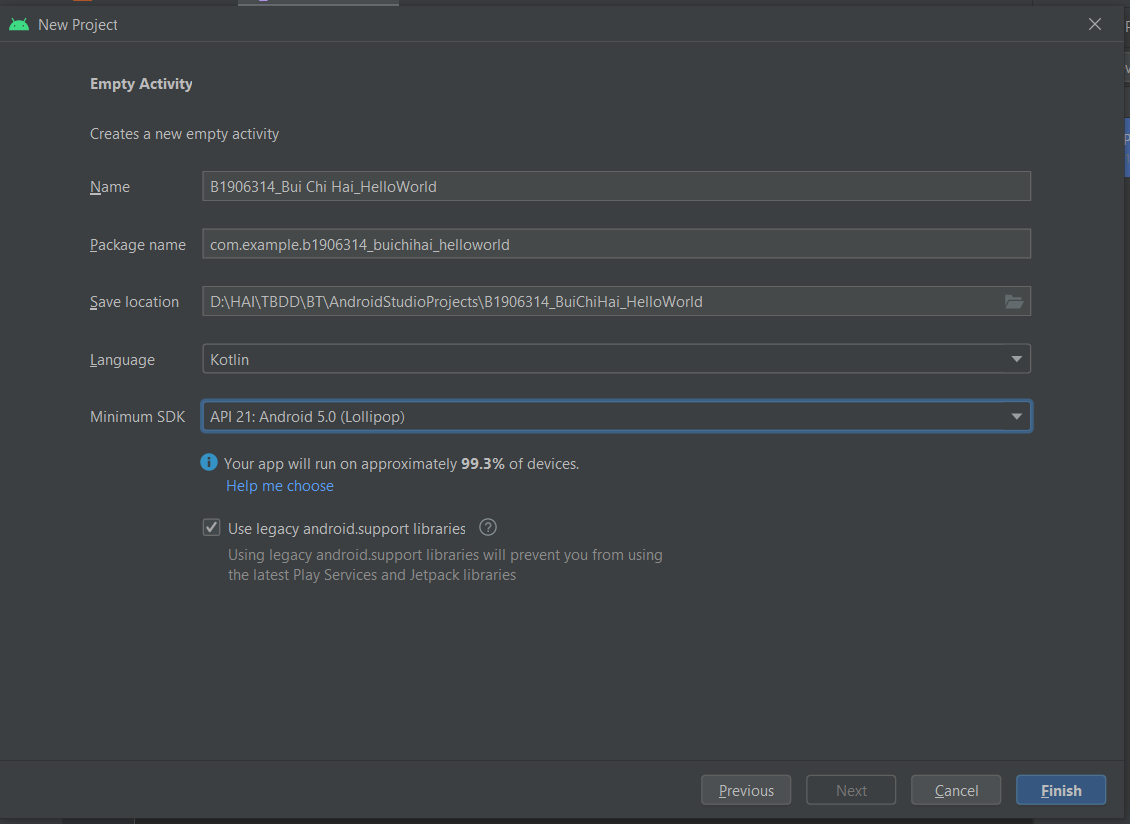
Hình 18: Click vào File chọn new và chọn new project để tạo file mới



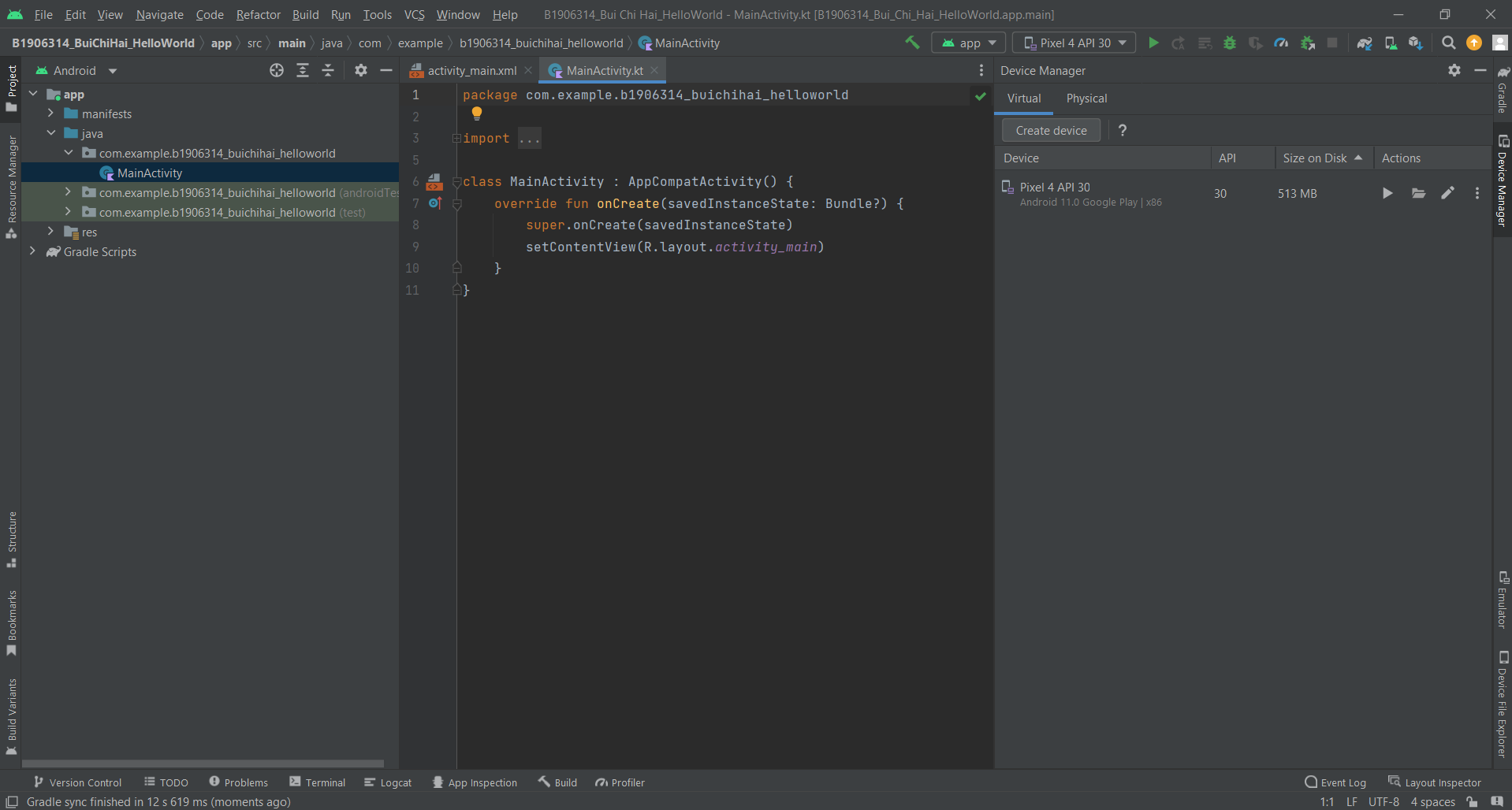
Hình 19: hiển thị giao diện new project ta chọn phần "Phone and Table"



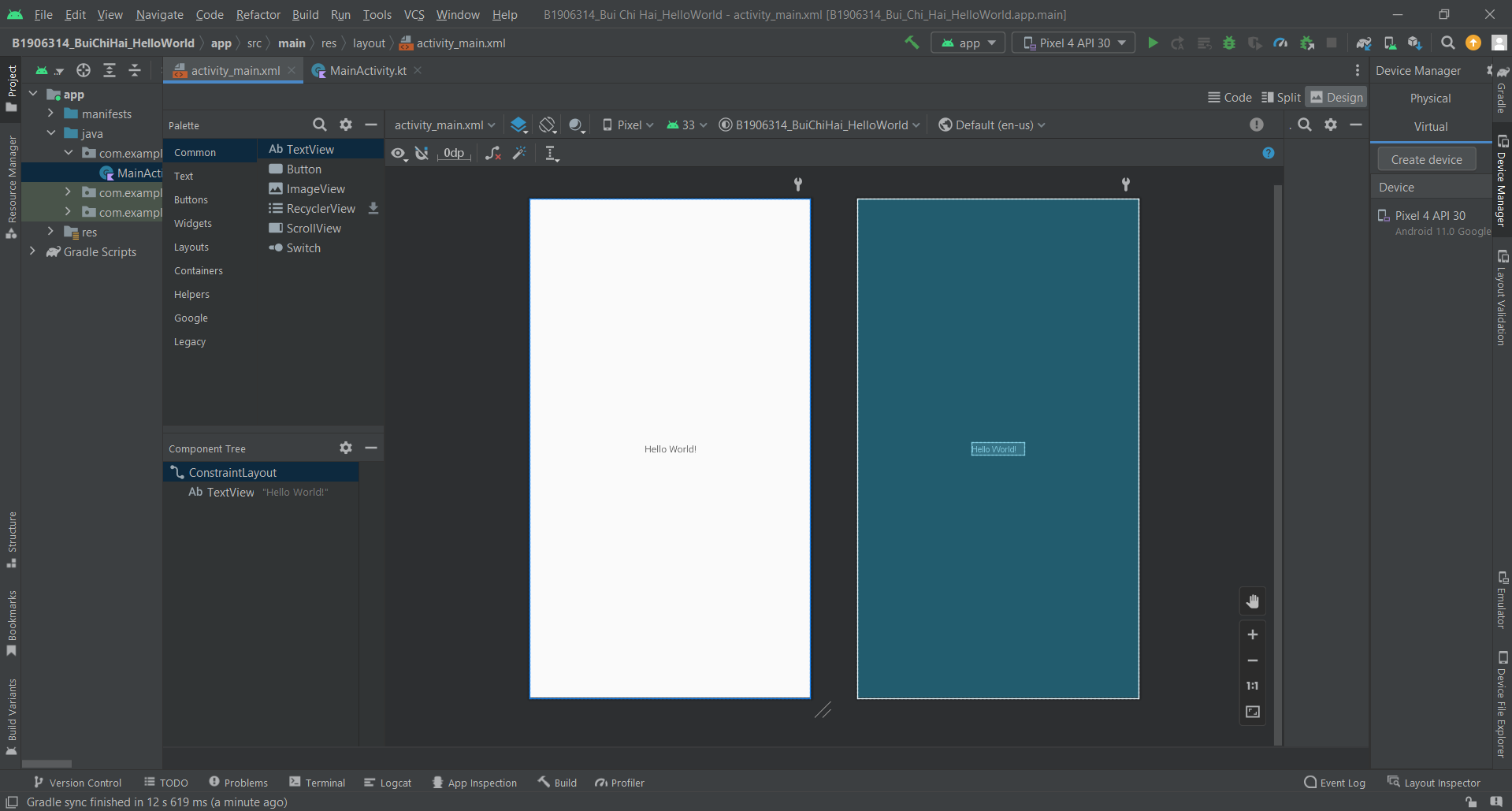
Hình 20: Chọn tiếp "Empty Activity" vầ bấm Next



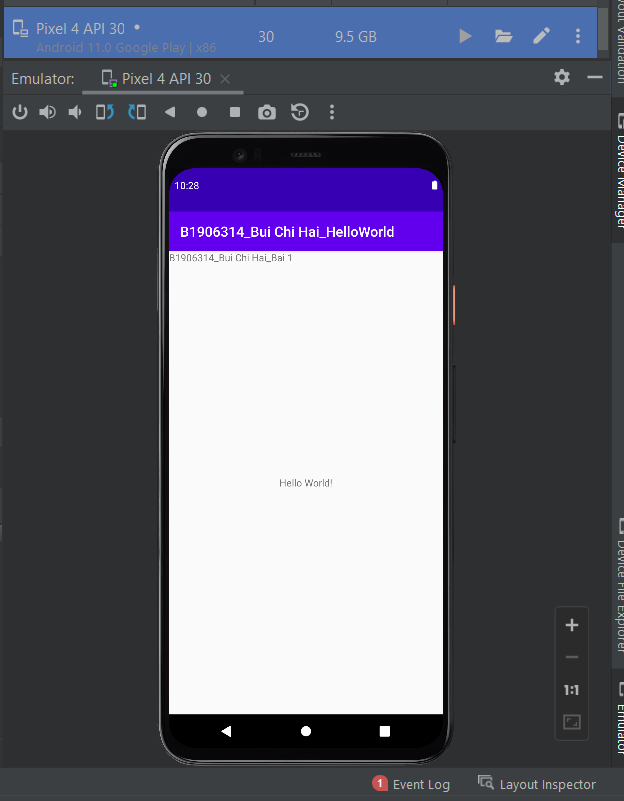
Hình 21: Nhập tên file và chọn nơi lưu project với ngôn ngữ "Kotlin" và chọn SDK tích hợp



Hình 22: giao diện khi hoàn thành việc tạo project



Hình 23: Giao diện ở "activity\_main.xml"



Hình 24: giao diện khi chạy dòng lệnh và hiển trên máy ảo

1. **Override các phương thức Activity để kiểm tra vòng đời hoạt động Activity (Sử dụng Logcat để kiểm tra).**

**Override các phương thức**

### Phương thức ghi đè

* Trong trường hợp lý tưởng nhất có thể, tất cả các ứng dụng Android sau khi chạy đều được lưu lại trong bộ nhớ (cụ thể là RAM), để sau này ứng dụng có thể khởi động lại nhanh hơn.
* Tuy nhiên trên thực tế, các thiết bị di động có tài nguyên phần cứng khá hạn chế, về cả RAM, tốc độ xử lý,… và vì vậy ứng dụng nào ít được sử dụng sẽ bị hủy tiến trình chạy nhằm thu hồi bộ nhớ.
* Như chúng ta đã đề cập trước đây, chúng ta luôn làm rõ mọi thứ trong Kotlin. Và khác với Java, Kotlin yêu cầu các annotation phải rõ ràng (chúng ta gọi là open) cho các overridable member và để có thể ghi đè.
* override annotation là yêu cầu bắt buộc đối với Derived.v(). Nếu nó bị thiếu, trình biên dịch sẽ báo lỗi. Nếu không có open annotation trên một hàm, giống như Base.nv(), khai báo một phương thức với cùng tên trong lớp con là bất hợp pháp dù là có ghi đè hoặc không ghi đè. Trong một final class (ví dụ một lớp không có open annotation), các thành viên open đều bị cấm.
* Một thành viên được đánh dấu là ghi đè bản thân nó là open, tức là nó có thể được ghi đè trong các lớp con. Nếu bạn muốn ngăn cấm việc ghi đè, hãy sử dụng final:

**Thuộc tính ghi đè**

* Ghi đè thuộc tính hoạt động tương tự như ghi đè phương thức; các thuộc tính được khai báo trên lớp cha và được khai báo lại trên một lớp con và có từ khóa override phía trước. Mỗi một thuộc tính được khai báo có thể được ghi đè bởi một thuộc tính cùng với một initializer hoặc bởi một thuộc tính cùng với phương thức getter.
* Bạn có thể ghi đè một thuộc tính val với một thuộc tính var, nhưng không thể làm ngược lại. Có thể làm được vậy bởi vì thuộc tính var cơ bản khai báo một phương thức getter và ghi đè nó bằng var giống như là khai báo bổ sung một phương thức setter trong lớp con. Lưu ý, bạn có thể sử dụng từ khóa override như là một phần của khai báo thuộc tính trong constructor chính

<https://viblo.asia/p/lop-va-ke-thua-trong-kotlin-bWrZnvOQZxw#_phuong-thuc-ghi-de-6>



### Các trạng thái chính trong vòng đời activity

* **Running**

Khi Activity được kích hoạt, và được hệ thống để vào BackStack, nó sẽ bước vào trạng thái active. Với trạng thái active, người dùng hoàn toàn có thể nhìn thấy và tương tác với Activity của ứng dụng.

* **Pause**

Trạng thái này khá đặc biệt. Trạng thái tạm dừng. Như bạn đã làm quen trên kia, trạng thái này xảy ra khi mà Activity của bạn vẫn đang chạy, người dùng vẫn nhìn thấy, nhưng Activity khi này lại bị che một phần bởi một thành phần nào đó. Chẳng hạn như khi bị một dialog đè lên. Cái sự che Activity này không phải hoàn toàn. Chính vì vậy mà Activity đó tuy được người dùng nhìn thấy nhưng không tương tác được.

* **Stop**

Trạng thái này khá giống với trạng thái tạm dừng trên kia. Nhưng khi này Activity bị che khuất hoàn toàn bởi một thành phần giao diện nào đó, hoặc bởi một ứng dụng khác. Và tất nhiên lúc này người dùng không thể nhìn thấy Activity của bạn được nữa.

Hành động mà khi người dùng nhấn nút Home ở System Bar để đưa ứng dụng của bạn về background, cũng khiến Activity đang hiển thị trong ứng dụng rơi vào trạng thái dừng này.

* **Dead**

Nếu Activity được lấy ra khỏi BackStack, chúng sẽ bị hủy và rơi vào trạng thái này. Trường hợp này xảy ra khi user nhấn nút Back ở System Bar để thoát một Activity. Hoặc lời gọi hàm finish() từ một Activity để “kill chính nó”. Cũng có khi ứng dụng ở trạng thái background quá lâu, hệ thống có thể sẽ thu hồi tài nguyên bằng cách dừng hẳn các Activity trong ứng dụng, làm cho tất cả các Activity đều vào trạng thái này.

Khi vào trạng thái dead, Activity sẽ kết thúc vòng đời của nó.  
Những ý trên giúp bạn nắm được tổng quan các trạng thái mà một Activity có thể trải qua.

[**https://viblo.asia/p/activity-va-vong-doi-cua-activity-bWrZngRwlxw**](https://viblo.asia/p/activity-va-vong-doi-cua-activity-bWrZngRwlxw)