Part 1: Straight forward, needed to download android studio and download the project file to import the files into android studio. Search for AVD by clicking CTRL + SHIFT + A and search for AVD. Then configured my emulator to run the app. Ran Successfully.

Part 2:

In android, every app or software you build must run in a platform. Code goes to a complier; complier compiles the code and creates a representation of the code which is translated into different machine language. Java code that complies and create representation for android can have different representation for windows/apple machine. The platform that android uses is Gradle which companies develop tools run on Gradle. Java would go to the Gradle build system, if its Java code, passes to Java Virtual Machine and create bytes code for it to be ready to be execute when UI elements such as buttons are clicked. Within the java folder, there is a file called SecondFragment, in line 42, the function login_button_submit is a type of UI element that is link to the Gradle script upon execution.

With operating systems, internet, games, security processes can communicate with each other even when they are in a different processing stack. The way for them to communicate with each other is using multiprocessing communicate. Android implement multiprocessing communicate to call one app to call another app. The app send data to the internet using an android basic OS process, if you are building an app you want to have functionalities with other apps that are available on your OS for example Android OS. This is where **intent** comes, Intent is unique way to call another application from your client applications. Each of your files that are name activity are micro applications that can run on one platform.

1) They are two types of intent. Explicit and Implicit. Explicit intent is when you specific an activity or micro application in the android application while implicit intent allows you to access all the micro application within the android system, giving option for more user to choose.
2) I believe the explicit intent is more secure because you're only accessing a particular activity.
3) In SecondFragment.kt, line 68-73 contains an implicit intent since it is calling Intent.Action_View which activates the intent class that able to communicate to all the activity within the android application. It looks like using the android net to parse the API request and sending user to the communication
4) In ThirdFragment.kt, 68-70 contains an explicit intent since it is calling a particular activity, ProductScrollingActivity and passing the variable User to the application.
5) We use the explicit intent that ThirdFragment.kt is using to specify which activity it is using.

2.2) To remove possible application using intent within the AndroidManifest.xml is following the process that shown within the stack overflow link, https://stackoverflow.com/questions/6460559/how-do-i-prevent-other-android-apps-from-accessing-my-activities. I thought adding android:exporeted = fales would block the application but got errors. Look into other ways fixing this issue and find out setting signature permissions and defining an activity internally did the job.

Part 3: Contained HTTP request that is not using TLS communication over HTTPS. Went over each files that needed to be change and add the https communication but after checking, it seems the website needs a certificate request that needs to be signed by the certificate authority to create a digital certificate on the server side.

SecondFragment.kt

```
48          var builder: Retrofit.Builder = Retrofit.Builder().baseUrl( baseUrl: "https://appsecclass.report")
```

ThirdFragment.kt

```
46          var builder: Retrofit.Builder = Retrofit.Builder().baseUrl( baseUrl: "http://appsecclass.report").addConverterFactory(
```

CardScrollingActivity.kt

```
59          var builder: Retrofit.Builder = Retrofit.Builder().baseUrl( baseUrl: "https://appsecclass.report").
var builder: Retrofit.Builder = Retrofit.Builder().baseUrl( baseUrl: "https://appsecclass.report").a
```

```
123    var builder: Retrofit.Builder = Retrofit.Builder().baseUrl( baseUrl: "https://appsecclass.report").
```

ProductScrollingActivity.kt

```
61     var builder: Retrofit.Builder = Retrofit.Builder().baseUrl( baseUrl: "https://appsecclass.report").addConverterFactory(
101    var builder: Retrofit.Builder = Retrofit.Builder().baseUrl( baseUrl: "https://appsecclass.report").addConverterFactory(
127    var builder: Retrofit.Builder = Retrofit.Builder().baseUrl( baseUrl: "https://appsecclass.report").addConverterFactory(
```

UseCard.kt

```
35     Glide.with( activity: this).asBitmap().load( string: "https://appsecclass.report/" + card?.product?.product
43     var builder: Retrofit.Builder = Retrofit.Builder().baseUrl( baseUrl: "https://appsecclass.report").
44         GsonConverterFactory.create())
```

GetCard.kt

```
Glide.with( activity: this).asBitmap().load( string: "https://appsecclass.report/"
40     var builder: Retrofit.Builder = Retrofit.Builder().baseUrl( baseUrl: "https://appsecclass.report").
```

CardRecyclerViewAdapter.kt

```
21         Glide.with(context).asBitmap().load( string: "https://appsecclass.report/"
23         Glide.with(context).asBitmap().load( string: "https://appsecclass.report/" +
```

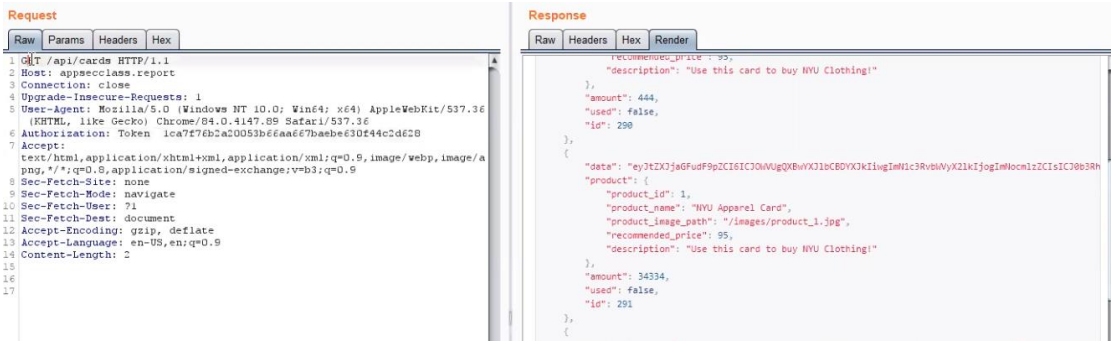
RecyvlerViewAdapter.lt

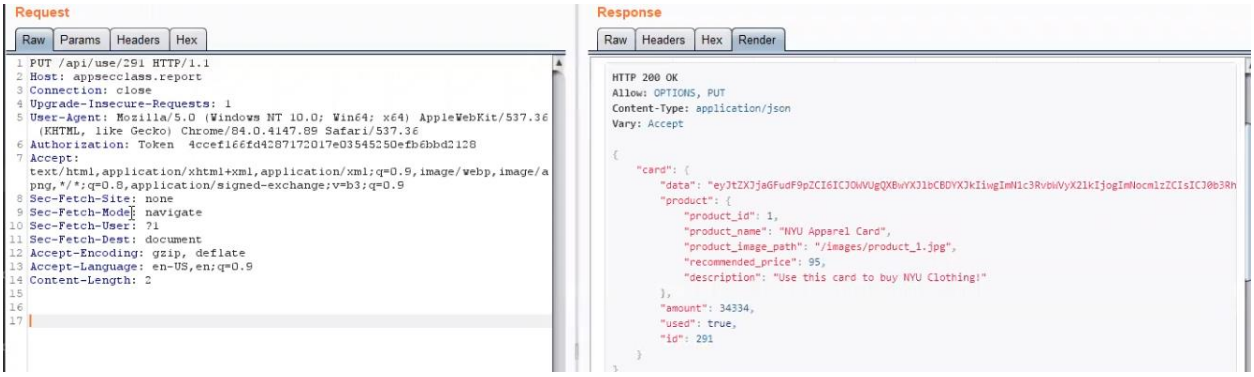
```
23     Glide.with(context).asBitmap().load( string: "https://appsecclass.report/" + product.productImageLink).into(image)
```

Part 4. While using burp suite to see each request and added my authorization token to log in to get the user response, I notice one interesting when testing the request listed in the CardInterface.kt. I was able to see my cards that contain a card I recently purchased by sending a get request, GET https://appsecclass.report/api/cards) and made it disappear when using PUT https://appsecclass.report/api/use/{card_number} since its making an update to the server

Making the request within my browser, capture it with brup suit and change the parameter worked successfully.

So, test this vulnerability, I created account and bought a lot of giftcards. Then I called a put request to a different user to use a card that that was purchase by the first user I created. After initiated the request I logout back to the user and saw the user card was gone.





Fixes: Their two fixies that could be implemented

1) whenever buy a card it creates a random number that small and increment. I believe we should hash it into a large number for attackers to have a hard number to iterate.

2) fix that work is if the server side validate that card the user uses it his card when the user buys since you see what available card that the user uses.

Part 5)

removing the privacy, I believe it was straight forward needed to remove and calls that related to sensors, metrics, and locations. I commented out to test if I made the program crash since it might have important dependency.
AndroidManifest.xml

UserInfo.kt

CardScrollingActivity.kt, removed argument calls.

ProductScrollingActivity.kt

Reporter.kt had some dependency and location activity so I commented out the whole code.