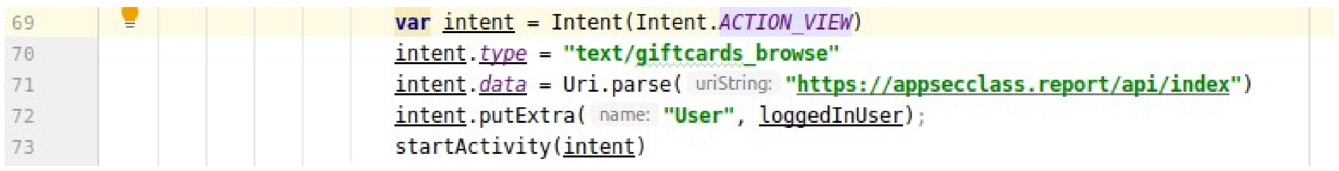
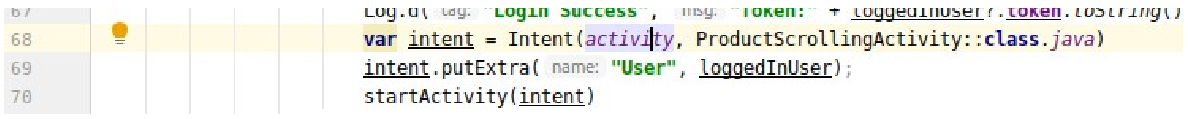
Part 2.1: What's the difference?

1. What are the two types of Intents?
    1. The two type of intents are explicit intent and implicit intent
2. Which of the two types of Intents are more secure?
    1. Explicit intent is more secure. Implicit its insecure because you dont know what service will respond to the intent
3. What type of Intent is shown on lines 69 to 73 of SecondFragment.kt?
    1. Below is a screenshot showing the code in lines 69 to 73 of the SecondFragment.kt. The type of intent is Implicit intent un the code below the component/context is not being specified



```
69                  var intent = Intent(Intent.ACTION_VIEW)
70                  intent.type = "text/giftcards_browse"
71                  intent.data = Uri.parse( uriString: "https://appsecclass.report/api/index")
72                  intent.putExtra( name: "User", loggedInUser);
73                  startActivity(intent)
```

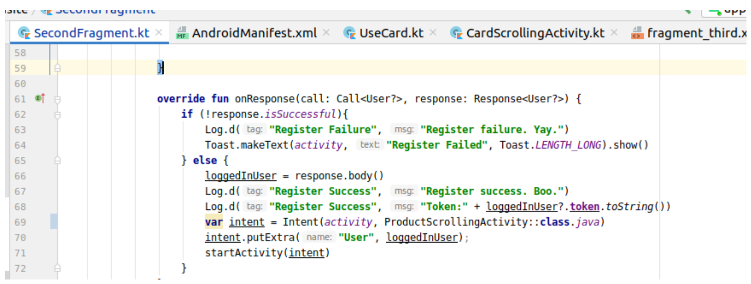
4. What type of Intent is shown on lines 68 to 70 of ThirdFragment.kt?
    1. Below is a screenshot showing the code in lines 68 to 70 of the ThirdFragment.kt. The type of intent for the code below is Explicit intent. In the code the component/context is being specified and the external class to be invoked is defined



```
67                  Log.d( tag: "Login Success", msg: "Token:" + loggedInUser?.token.toString()
68                  var intent = Intent(activity, ProductScrollingActivity::class.java)
69                  intent.putExtra( name: "User", loggedInUser);
70                  startActivity(intent)
```

5. Which of these two Intents is the proper way to do an Intent?

    1. The proper way is to use explicit intent because explicit intents are used in the application itself to switch between activities. I modified the code SecondFragment.kt to make it similar to the code in ThirdFragment.kt



```
SecondFragment.kt  AndroidManifest.xml  UseCard.kt  CardScrollingActivity.kt  fragment_third.x
58
59          }
60
61          override fun onResponse(call: Call<User?>, response: Response<User?>) {
62              if (!response.isSuccessful){
63                  Log.d( tag: "Register Failure",  msg: "Register failure. Yay.")
64                  Toast.makeText(activity,  text: "Register Failed", Toast.LENGTH_LONG).show()
65              } else {
66                  loggedInUser = response.body()
67                  Log.d( tag: "Register Success",  msg: "Register success. Boo.")
68                  Log.d( tag: "Register Success",  msg: "Token:" + loggedInUser?.token.toString())
69                  var intent = Intent(activity, ProductScrollingActivity::class.java)
70                  intent.putExtra( name: "User", loggedInUser);
71                  startActivity(intent)
72              }
```

Part 2.2: Shutting out the world

In order to remove the possibility of other applications using intents to launch activities of the application I first removed the launcher category inside the intent filter in the AndroidManifest file. I noticed that Android Studio generated an error. Then I removed all the lines with <intent-filter> and </intent-filter> from the file but again another error was generated when I compiled the project.

After more research about intents in the Android Manifest file I noticed the mention of the action "VIEW" can be used to open other apps. Therefore, I removed the line of code: `<action android:name="android.intent.action.VIEW" />` from every activity in the android manifest file

Part 3: Can you read me out there?

I modified the following the following code line below from http to https:

```
 var builder: Retrofit.Builder =
Retrofit.Builder().baseUrl("http://appsecclass.report").addConverterFactory(GsonConverterFac
tory.create()) by
```

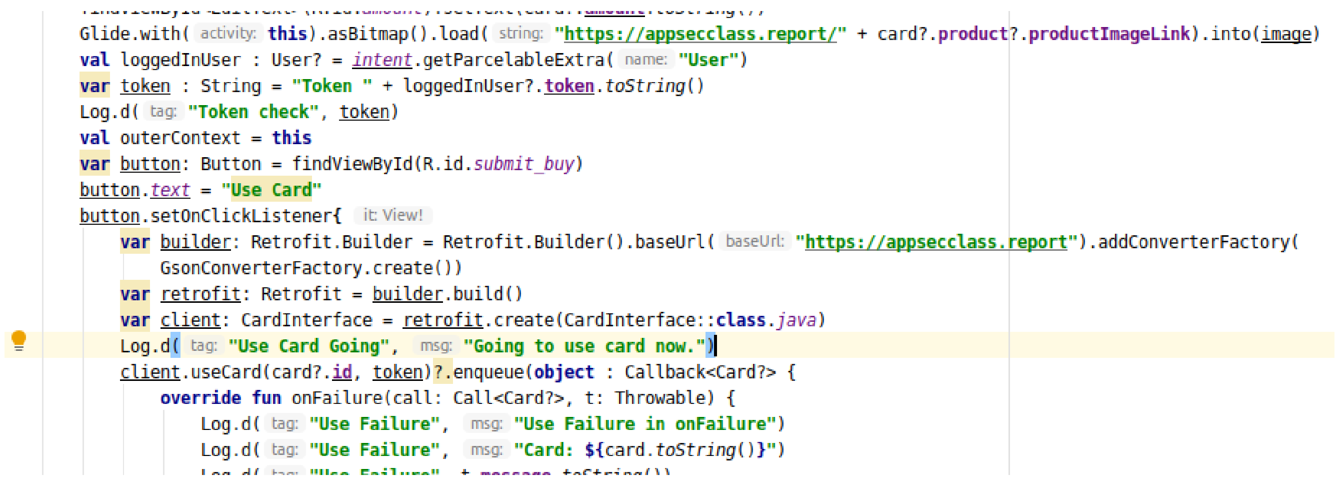
```
 var builder: Retrofit.Builder =
Retrofit.Builder().baseUrl("https://appsecclass.report").addConverterFactory(GsonConverterFa
ctory.create()) by
```

I modified all the code lines that reference the website in the following files:

1. ThirdFragment.kt
2. CardScrollingActivity.kt
3. ProductScrollingActivity.kt
4. UseCard.kt
5. GetCard.kt
6. CardRecyclerViewAdapter.kt
7. RecyclerViewAdapter.kt

Part 4: Oops, was that card yours?

I believe that manipulating the code line “card?” can be use to allow users to GiftCards that do not belong to them. The attacker can use that to select a card that belongs to another user.



```
Glide.with( activity: this).asBitmap().load( string: "https://appsecclass.report/" + card?.product?.productImageLink).into(image)
val loggedInUser : User? = intent.getParcelableExtra( name: "User")
var token : String = "Token " + loggedInUser?.token.toString()
Log.d( tag: "Token check", token)
val outerContext = this
var button: Button = findViewById(R.id.submit_buy)
button.text = "Use Card"
button.setOnClickListener{  it: View!
    var builder: Retrofit.Builder = Retrofit.Builder().baseUrl( baseUrl: "https://appsecclass.report").addConverterFactory(
        GsonConverterFactory.create())
    var retrofit: Retrofit = builder.build()
    var client: CardInterface = retrofit.create(CardInterface::class.java)
    Log.d( tag: "Use Card Going",  msg: "Going to use card now.")
    client.useCard(card?.id, token)?.enqueue(object : Callback<Card?> {
        override fun onFailure(call: Call<Card?>, t: Throwable) {
            Log.d( tag: "Use Failure",  msg: "Use Failure in onFailure")
            Log.d( tag: "Use Failure",  msg: "Card: ${card.toString()}")
```

Part 5: Privacy is Important

You should remove all necessary code in the following files:

1. AndroidManifest.xml

I removed the following code from the file:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

2. UserInfo.kt

I removed the following code from the file:

```
@POST("/api/metrics")
```

3. CardScrollingActivity.kt

I removed the following code from override fun OnCreate(savedInstanceState: Bundle?)

```
val locationPermissionCode = 2
var locationManager = getSystemService(Context.LOCATION_SERVICE) as LocationManager
if ((ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED)) {
    ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.ACCESS_FINE_LOCATION), locationPermissionCode)
}
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 5000, 5f, this)
```

```
sensorManager = getSystemService(Context.SENSOR_SERVICE) as SensorManager
mAccel = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)
```

I also removed the following functions:

```
override fun onResume() {
    super.onResume()
    mAccel?.also { accel ->
        sensorManager.registerListener(this, accel, SensorManager.SENSOR_DELAY_NORMAL)
    }
}

override fun onPause() {
    super.onPause()
    sensorManager.unregisterListener(this)
}
```

4. ProductScrollingActivity.kt

I removed the following code from override fun OnCreate(savedInstanceState: Bundle?)

```
val locationPermissionCode = 2
var locationManager = getSystemService(Context.LOCATION_SERVICE) as LocationManager
if ((ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED)) {
    ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.ACCESS_FINE_LOCATION), locationPermissionCode)
}
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 5000, 5f, this)
sensorManager = getSystemService(Context.SENSOR_SERVICE) as SensorManager
mAccel = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)
```

I also removed the following functions:

```
override fun onResume() {
    super.onResume()
    mAccel?.also { accel ->
        sensorManager.registerListener(this, accel, SensorManager.SENSOR_DELAY_NORMAL)
    }
}

override fun onPause() {
    super.onPause()
    sensorManager.unregisterListener(this)
}
```