



Quality Assessment for Local Banking Apps

(Alinma Bank)

Supervised by :
Dr. Lamees Alhazzaa

Group 3

Prepared by :

Name	Student ID	Email
Asia Alrajeh	440020948	aonalrajeh@sm.imamu.edu.sa
Sara Ibrahim Almashharawi	440028560	siralmashharawi@sm.imamu.edu.sa
Hanin Alanazi	440021299	haralanazi99@sm.imamu.edu.sa

Section : 372

Table of Contents

CONTENT	PAGE NO.
Introduction	2
Mobile App analysis tool	3
Application permissions	5
Manifest analysis	6
Code quality analysis	7
Design	10
Cohesion and Coupling	12
GUI	13
Conclusion	14

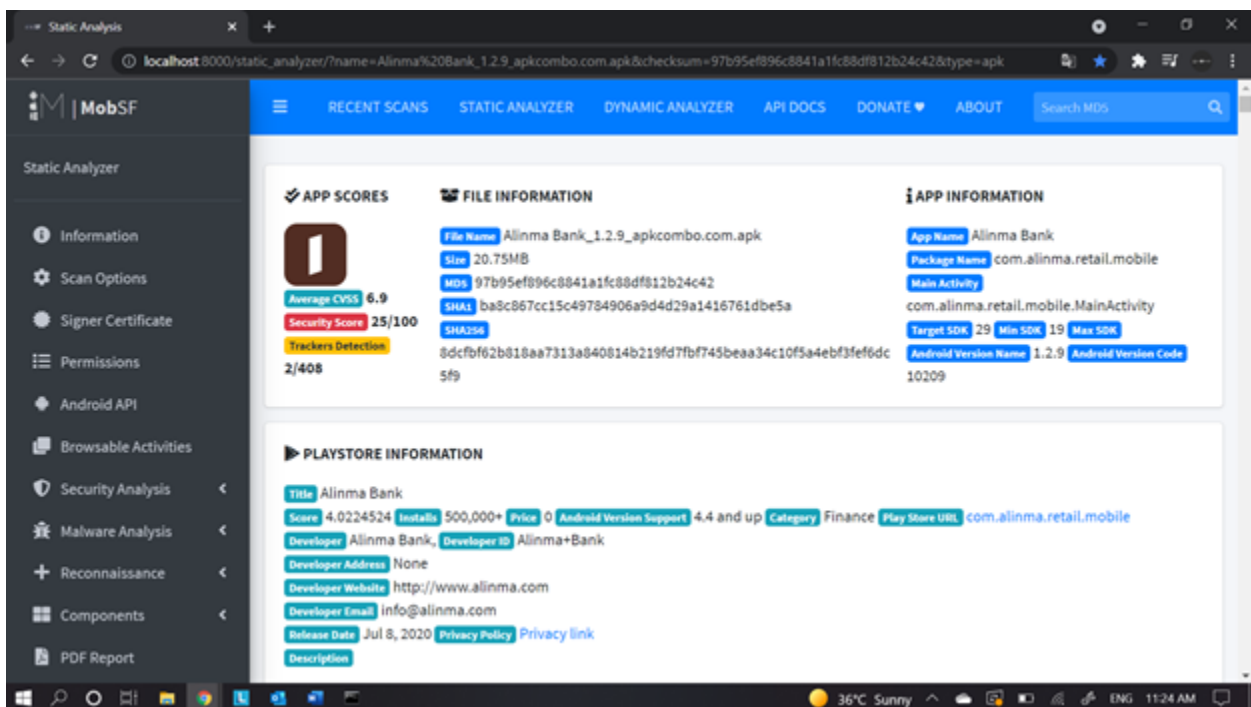
INTRODUCTION

Banking apps are becoming the most widely used applications in the world. As a result, making it strong and secure is a serious challenge that we must handle. The purpose of this report is to ensure the quality of the Alinma android application by applying static analysis using the Mobile Security Framework tool. Mobile Security Framework is a static and dynamic analysis tool for mobile security. It provides a percentage of an app's security that can be used to improve the app's security. After analyzing the app, we discovered several vulnerabilities, which we will describe in this report along with recommended mitigations based on our knowledge to make it useful, strong, secure, and accessible.

MOBILE APP ANALYSIS TOOL

To start the analysis , first we have to specify the requirement to install the MobSF tool. Then extract the APK of the Alinma bank application . We have to write “ run.bat “ in the command line and open localhost:8000 , then the tool will start running .Finally, upload the APK onto MobSF then it will start the analysis.

It will appear a summary of application’s information and the score of security :



The screenshot displays the MobSF Static Analyzer web interface. The browser address bar shows the URL: `localhost:8000/static_analyzer/?name=Alinma%20Bank_1.2.9_apkcombo.com.apk&checksum=97b95ef896c8841a1fc88df812b24c42&type=apk`. The interface is divided into a left sidebar with navigation options and a main content area displaying analysis results.

Static Analyzer

- Information
- Scan Options
- Signer Certificate
- Permissions
- Android API
- Browsable Activities
- Security Analysis
- Malware Analysis
- Reconnaissance
- Components
- PDF Report

APP SCORES

- Average CVSS: 6.9
- Security Score: 25/100
- Trackers Detection: 2/408

FILE INFORMATION

- File Name: Alinma Bank_1.2.9_apkcombo.com.apk
- Size: 20.75MB
- MD5: 97b95ef896c8841a1fc88df812b24c42
- SHA1: ba8c867cc15c49784906a9d4d29a1416761dbe5a
- SHA256: 8dcfbf62b818aa7313a840814b219fd7fbf745beaa34c10f5a4ebf3fef6dc5f9

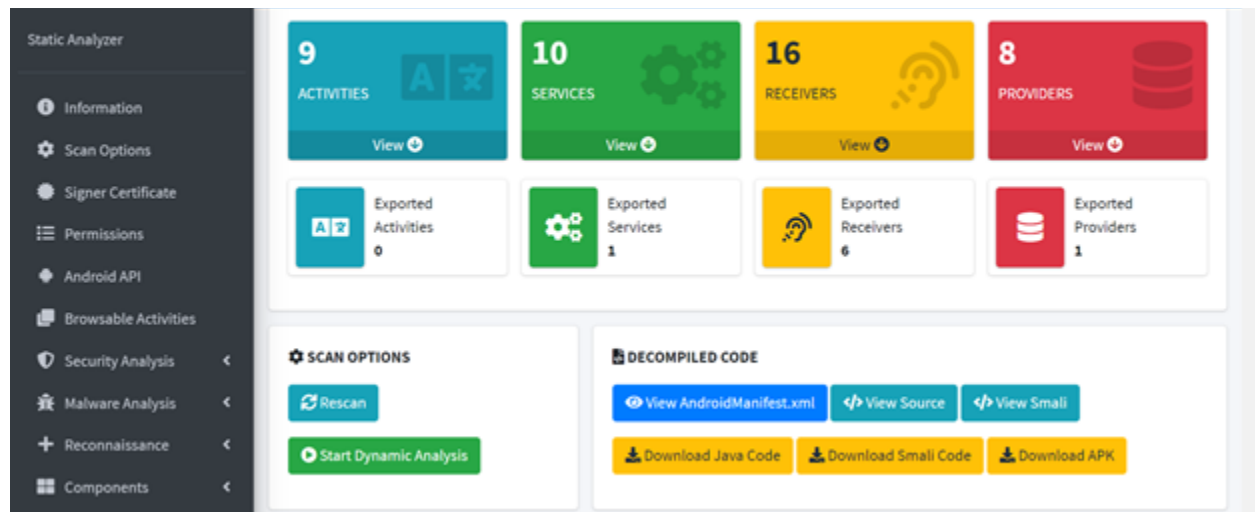
APP INFORMATION

- App Name: Alinma Bank
- Package Name: com.alinma.retail.mobile
- Main Activity: com.alinma.retail.mobile.MainActivity
- Target SDK: 29
- Min SDK: 19
- Max SDK: 29
- Android Version Name: 1.2.9
- Android Version Code: 10209

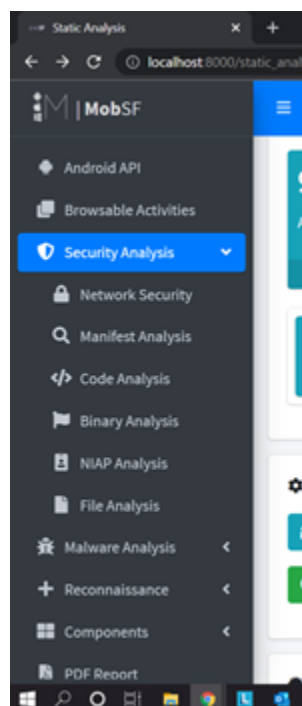
PLAYSTORE INFORMATION

- Title: Alinma Bank
- Score: 4.0224524
- Installs: 500,000+
- Price: 0
- Android Version Support: 4.4 and up
- Category: Finance
- Play Store URL: com.alinma.retail.mobile
- Developer: Alinma Bank
- Developer ID: Alinma+Bank
- Developer Address: None
- Developer Website: <http://www.alinma.com>
- Developer Email: info@alinma.com
- Release Date: Jul 8, 2020
- Privacy Policy: [Privacy link](#)
- Description:

When scroll down the page , we found an icon to download the source code in a zip file :



to determine the security issues, we click in security analysis in the left , then appears all the security issue :



APPLICATION PERMISSIONS

ISSUE	DESCRIPTION	MITIGATIONS
read/modify/delete external storage contents	Allows an application to write to external storage.	It should restrict these operations by preventing modification and writing ,only allowing reading.
Network based,GPS location	Access location, such as the mobile network database,and Global Positioning System to determine an approximate phone location, where available. Malicious applications can use this to determine approximately where you are and may consume additional.	Since it is Banking app , so it does not need to know information about locations , when needed to access it has to ask for permission
Take pictures and videos	This allows the application to collect images that the camera is seeing any time.	It should not access the camera, because it is a breach of privacy

MANIFEST ANALYSIS

ISSUE	DESCRIPTION	MITIGATIONS
Clear text traffic is Enabled For App	Cleartext is transmitted or stored text that has not been subjected to encryption and is not meant to be encrypted	Text must be encrypted because network attackers can eavesdrop on transmitted data and also modify it without being detected.
Application Data Can be Backed up	It allows anyone to backup your application data via adb. It allows users who have enabled USB debugging to copy application data off of the device.	Not everyone should do the backup, only technical staff for the bank have the authority to backup the data when needed
Broadcast Receiver	A Broadcast Receiver is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device.	This is out of our knowledge
Content Provider is not Protected.	A Content Provider is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device.	The content is sensitive it shouldn't be shared
Protection level permission of the services is not checked	A Service Is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device. It is protected by a permission which is not defined in the analysed application	the protection level of the permission should be checked where it is defined. only applications signed with the same certificate can obtain the permission.

CODE QUALITY ANALYSIS

Issue 1: Classes Name are variables

```
18.
19. /* access modifiers changed from: package-private */
20. public class a {
21.     private Chain<PushNotificationOpenHelper> a;
22.     @Nullable
23.     private final Context b = AndroidPlatformModule.getApplicationContext();
24.
25.     a(Chain<PushNotificationOpenHelper> chain) {
26.         this.a = chain;
27.     }
28.
```

solution: the class name should specify the reason for the class .

Issue 2 : variables name are meaningless :

```
37.
38. /* access modifiers changed from: package-private */
39. public void a(PushMessage pushMessage) {
40.     boolean z;
41.     Bundle bundle = new Bundle();
42.     bundle.putString(Pushwoosh.PUSH_RECEIVE_EVENT, pushMessage.toJson().toString());
43.     Intent intent = new Intent();
44.     String str = AndroidPlatformModule.getAppInfoProvider().a() + ".MESSAGE";
45.     intent.setAction(str);
46.     intent.setFlags(872415232);
47.     intent.putExtras(bundle);
48.     try {
49.         if (this.b != null) {
50.             this.b.startActivity(intent);
51.         }
52.         z = false;
53.     } catch (ActivityNotFoundException unused) {
54.         z = true;
55.         PWLog.warn("Can't launch activity. Are you sure you have an activity with '" + str + "' action in your manifest? Launching default activity.");
56.     }
57.     if (z) {
58.         try {
59.             String a2 = AndroidPlatformModule.getAppInfoProvider().a();
60.             PackageManager packageManager = AndroidPlatformModule.getManagerProvider().getPackageManager();
61.             Intent launchIntentForPackage = packageManager == null ? null : packageManager.getLaunchIntentForPackage(a2);
62.             if (launchIntentForPackage != null) {
63.                 launchIntentForPackage.addCategory("android.intent.category.LAUNCHER");
64.                 launchIntentForPackage.setFlags(872415232);
65.                 launchIntentForPackage.putExtras(bundle);
66.                 this.b.startActivity(launchIntentForPackage);
67.                 return;
68.             }
69.         }
70.     }
71. }
```

solution: meaningful identifiers

Issue 3 : the source code is poor of documentation :

```
17.
18. public class d extends a {
19.     static final BigInteger a = new BigInteger("1");
20.     private final b b;
21.
22.     public d(b bVar) {
23.         super(bVar);
24.         this.b = bVar;
25.     }
26.
27.     private Pair<BigInteger, BigInteger> a(String str) {
28.         return new Pair<>(a(str), b(str));
29.     }
30.
31.     private void a(String str, BigInteger bigInteger, BigInteger bigInteger2) {
32.         b bVar = this.b;
33.         bVar.a(str + "mod", bigInteger);
34.         b bVar2 = this.b;
35.         bVar2.a(str + "exp", bigInteger2);
36.     }
37.
38.     private BigInteger b(String str) {
39.         b bVar = this.b;
40.         return bVar.b(str + "exp", a);
41.     }
42.
43.
44.
45.
46.
47.
48.
49.
50.
51.
52.
53.
54.
55.
56.
57.
58.
59.
60.
61.
62.
63.
64.
65.
66.
67.
68.
69.
70.
71.
72.
73.
74.
75.
76.
77.
78.
79.
80.
81.
82.
83.
84.
85.
86.
87.
88.
89.
90.
91.
92.
93.
94.
95.
96.
97.
98.
99.
100.
101.
102.
103.
104.
105.
106.
107.
108.
109.
110.
111.
112.
113.
114.
115.
116.
117.
118.
119.
120.
121.
122.
123.
124.
125.
126.
127.
128.
129.
130.
131.
132.
133.
134.
135.
136.
137.
138.
139.
140.
141.
142.
143.
144.
145.
146.
147.
148.
149.
150.
151.
152.
153.
154.
155.
156.
157.
158.
159.
160.
161.
162.
163.
164.
165.
166.
167.
168.
169.
170.
171.
172.
173.
174.
175.
176.
177.
178.
179.
180.
181.
182.
183.
184.
185.
186.
187.
188.
189.
190.
191.
192.
193.
194.
195.
196.
197.
198.
199.
200.
201.
202.
203.
204.
205.
206.
207.
208.
209.
210.
211.
212.
213.
214.
215.
216.
217.
218.
219.
220.
221.
222.
223.
224.
225.
226.
227.
228.
229.
230.
231.
232.
233.
234.
235.
236.
237.
238.
239.
240.
241.
242.
243.
244.
245.
246.
247.
248.
249.
250.
251.
252.
253.
254.
255.
256.
257.
258.
259.
260.
261.
262.
263.
264.
265.
266.
267.
268.
269.
270.
271.
272.
273.
274.
275.
276.
277.
278.
279.
280.
281.
282.
283.
284.
285.
286.
287.
288.
289.
290.
291.
292.
293.
294.
295.
296.
297.
298.
299.
300.
301.
302.
303.
304.
305.
306.
307.
308.
309.
310.
311.
312.
313.
314.
315.
316.
317.
318.
319.
320.
321.
322.
323.
324.
325.
326.
327.
328.
329.
330.
331.
332.
333.
334.
335.
336.
337.
338.
339.
340.
341.
342.
343.
344.
345.
346.
347.
348.
349.
350.
351.
352.
353.
354.
355.
356.
357.
358.
359.
360.
361.
362.
363.
364.
365.
366.
367.
368.
369.
370.
371.
372.
373.
374.
375.
376.
377.
378.
379.
380.
381.
382.
383.
384.
385.
386.
387.
388.
389.
390.
391.
392.
393.
394.
395.
396.
397.
398.
399.
400.
401.
402.
403.
404.
405.
406.
407.
408.
409.
410.
411.
412.
413.
414.
415.
416.
417.
418.
419.
420.
421.
422.
423.
424.
425.
426.
427.
428.
429.
430.
431.
432.
433.
434.
435.
436.
437.
438.
439.
440.
441.
442.
443.
444.
445.
446.
447.
448.
449.
450.
451.
452.
453.
454.
455.
456.
457.
458.
459.
460.
461.
462.
463.
464.
465.
466.
467.
468.
469.
470.
471.
472.
473.
474.
475.
476.
477.
478.
479.
480.
481.
482.
483.
484.
485.
486.
487.
488.
489.
490.
491.
492.
493.
494.
495.
496.
497.
498.
499.
500.
501.
502.
503.
504.
505.
506.
507.
508.
509.
510.
511.
512.
513.
514.
515.
516.
517.
518.
519.
520.
521.
522.
523.
524.
525.
526.
527.
528.
529.
530.
531.
532.
533.
534.
535.
536.
537.
538.
539.
540.
541.
542.
543.
544.
545.
546.
547.
548.
549.
550.
551.
552.
553.
554.
555.
556.
557.
558.
559.
560.
561.
562.
563.
564.
565.
566.
567.
568.
569.
570.
571.
572.
573.
574.
575.
576.
577.
578.
579.
580.
581.
582.
583.
584.
585.
586.
587.
588.
589.
590.
591.
592.
593.
594.
595.
596.
597.
598.
599.
600.
601.
602.
603.
604.
605.
606.
607.
608.
609.
610.
611.
612.
613.
614.
615.
616.
617.
618.
619.
620.
621.
622.
623.
624.
625.
626.
627.
628.
629.
630.
631.
632.
633.
634.
635.
636.
637.
638.
639.
640.
641.
642.
643.
644.
645.
646.
647.
648.
649.
650.
651.
652.
653.
654.
655.
656.
657.
658.
659.
660.
661.
662.
663.
664.
665.
666.
667.
668.
669.
670.
671.
672.
673.
674.
675.
676.
677.
678.
679.
680.
681.
682.
683.
684.
685.
686.
687.
688.
689.
690.
691.
692.
693.
694.
695.
696.
697.
698.
699.
700.
701.
702.
703.
704.
705.
706.
707.
708.
709.
710.
711.
712.
713.
714.
715.
716.
717.
718.
719.
720.
721.
722.
723.
724.
725.
726.
727.
728.
729.
730.
731.
732.
733.
734.
735.
736.
737.
738.
739.
740.
741.
742.
743.
744.
745.
746.
747.
748.
749.
750.
751.
752.
753.
754.
755.
756.
757.
758.
759.
760.
761.
762.
763.
764.
765.
766.
767.
768.
769.
770.
771.
772.
773.
774.
775.
776.
777.
778.
779.
780.
781.
782.
783.
784.
785.
786.
787.
788.
789.
790.
791.
792.
793.
794.
795.
796.
797.
798.
799.
800.
801.
802.
803.
804.
805.
806.
807.
808.
809.
810.
811.
812.
813.
814.
815.
816.
817.
818.
819.
820.
821.
822.
823.
824.
825.
826.
827.
828.
829.
830.
831.
832.
833.
834.
835.
836.
837.
838.
839.
840.
841.
842.
843.
844.
845.
846.
847.
848.
849.
850.
851.
852.
853.
854.
855.
856.
857.
858.
859.
860.
861.
862.
863.
864.
865.
866.
867.
868.
869.
870.
871.
872.
873.
874.
875.
876.
877.
878.
879.
880.
881.
882.
883.
884.
885.
886.
887.
888.
889.
890.
891.
892.
893.
894.
895.
896.
897.
898.
899.
900.
901.
902.
903.
904.
905.
906.
907.
908.
909.
910.
911.
912.
913.
914.
915.
916.
917.
918.
919.
920.
921.
922.
923.
924.
925.
926.
927.
928.
929.
930.
931.
932.
933.
934.
935.
936.
937.
938.
939.
940.
941.
942.
943.
944.
945.
946.
947.
948.
949.
950.
951.
952.
953.
954.
955.
956.
957.
958.
959.
960.
961.
962.
963.
964.
965.
966.
967.
968.
969.
970.
971.
972.
973.
974.
975.
976.
977.
978.
979.
980.
981.
982.
983.
984.
985.
986.
987.
988.
989.
990.
991.
992.
993.
994.
995.
996.
997.
998.
999.
1000.
```

solution: use standard documentation like java has javaDoc.

Issue 4 : classes are not coherent

```
12.
13. public final class AESCrypt {
14.     private static final String AES_MODE = "AES/CBC/PKCS7Padding";
15.     private static final String CHARSET = "UTF-8";
16.     public static boolean DEBUG_LOG_ENABLED = false;
17.     private static final String HASH_ALGORITHM = "SHA-256";
18.     private static final String TAG = "AESCrypt";
19.     private static final byte[] ivBytes = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
20.
21.     private static SecretKeySpec generateKey(String str) throws NoSuchAlgorithmException, UnsupportedEncodingException {
22.         MessageDigest instance = MessageDigest.getInstance(HASH_ALGORITHM);
23.         byte[] bytes = str.getBytes(CHARSET);
24.         instance.update(bytes, 0, bytes.length);
25.         byte[] digest = instance.digest();
26.         log("SHA-256 key ", digest);
27.         return new SecretKeySpec(digest, "AES");
28.     }
29.
30.     public static String encrypt(String str, String str2) throws GeneralSecurityException {
31.         try {
32.             SecretKeySpec generateKey = generateKey(str);
33.             log("message", str2);
34.             String encodeToString = Base64.encodeToString(encrypt(generateKey, ivBytes, str2.getBytes(CHARSET), 2);
```

solution: the identifiers are meaningful names and use underscore identifiers but other classes use characters, should use one code structure and stick to it.

Issue 5 : method names are character :

```
52.
53.     private void d(String str) {
54.         b bVar = this.b;
55.         bVar.b(ite + "exp");
56.         b bVar2 = this.b;
57.         bVar2.b(str + "mod");
58.     }
59.
60.     private PrivateKey e() throws InvalidKeySpecException, NoSuchAlgorithmException {
61.         Pair<BigInteger, BigInteger> a2 = a("private.key");
62.         return KeyFactory.getInstance("RSA").generatePrivate((RSAPrivateKeySpec) a2.first, a2.second);
63.     }
64.
65.     private void f() {
66.         d("public.key");
67.     }
68.
69.     private void g() {
70.         d("private.key");
71.     }
72.
```

solution: the method name should specify the reason for the method .

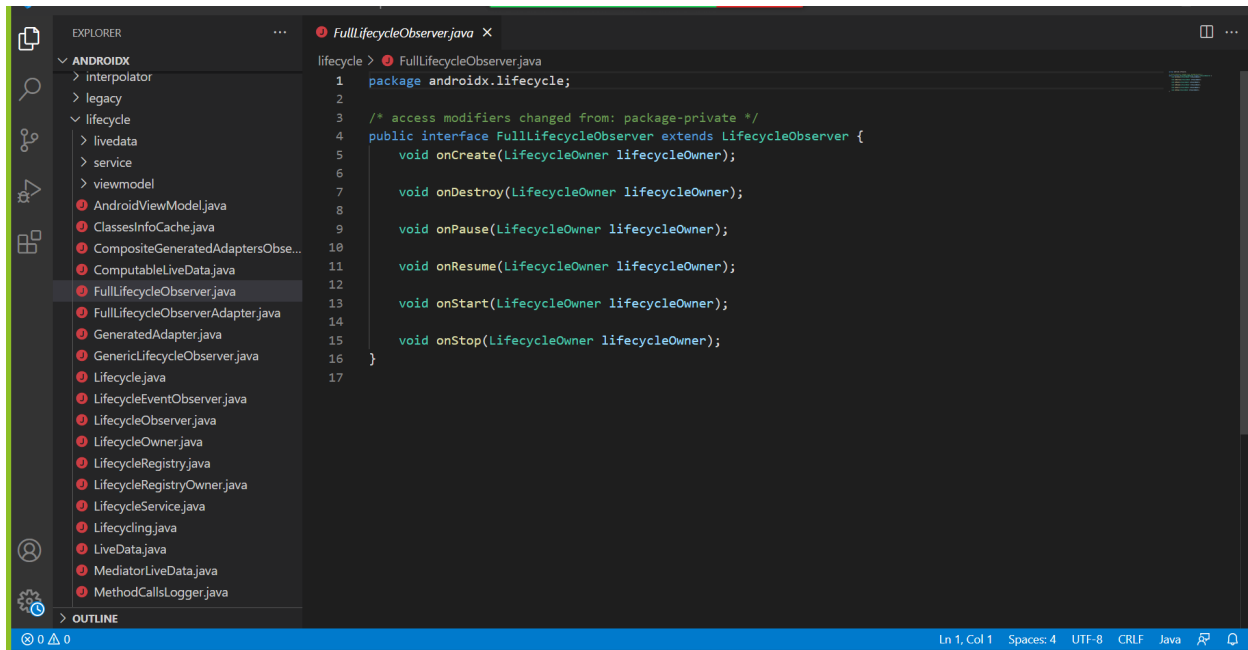
Issue 6 : alignment are not vertical :

```
494.
495.     public static float e() {
496.         try {
497.             Intent a2 = AndroidPlatformModule.getReceiverProvider().a((BroadcastReceiver) null, new IntentFilter("android.intent.action.BATTERY_CHANGED"));
498.             if (a2 == null) {
499.                 return -1.0f;
500.             }
501.             int intExtra = a2.getIntExtra(FirebaseAnalytics.Param.LEVEL, -1);
502.             int intExtra2 = a2.getIntExtra("scale", -1);
503.             if (intExtra == -1 || intExtra2 == -1) {
504.                 return -1.0f;
505.             }
506.             return (((float) intExtra) / ((float) intExtra2)) * 100.0f;
507.         } catch (Exception unused) {
508.             return -1.0f;
509.         }
510.     }
511.
```

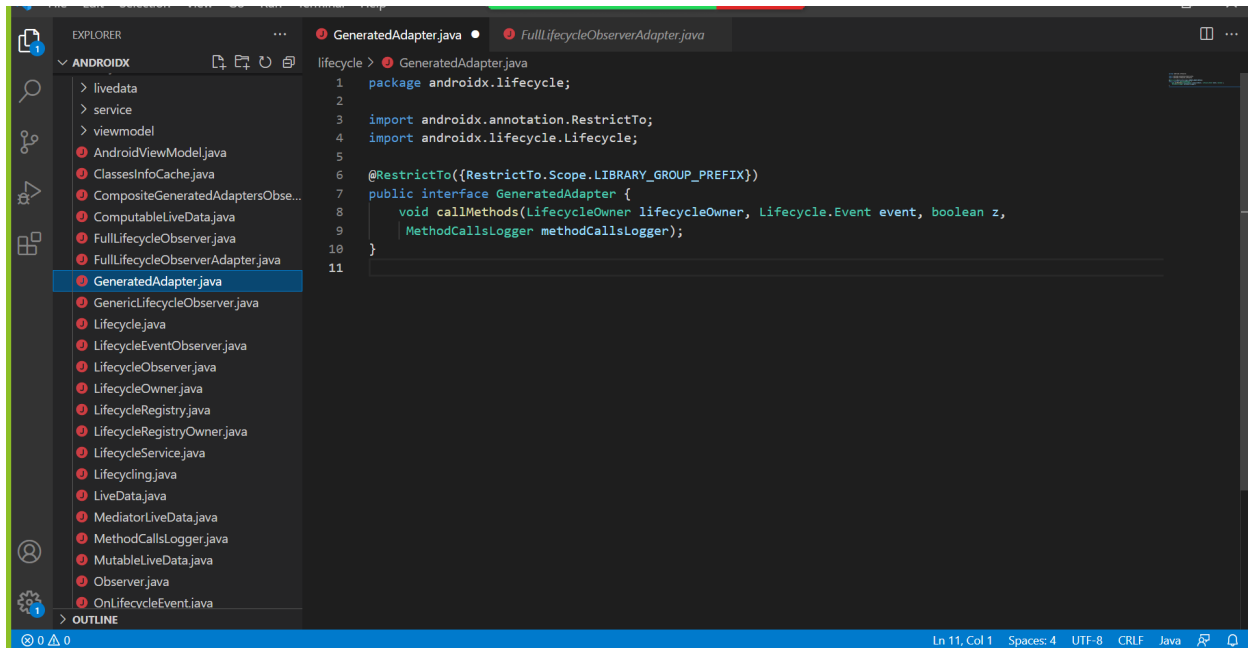
solution: use vertical alignment to be easy to read.

DESIGN

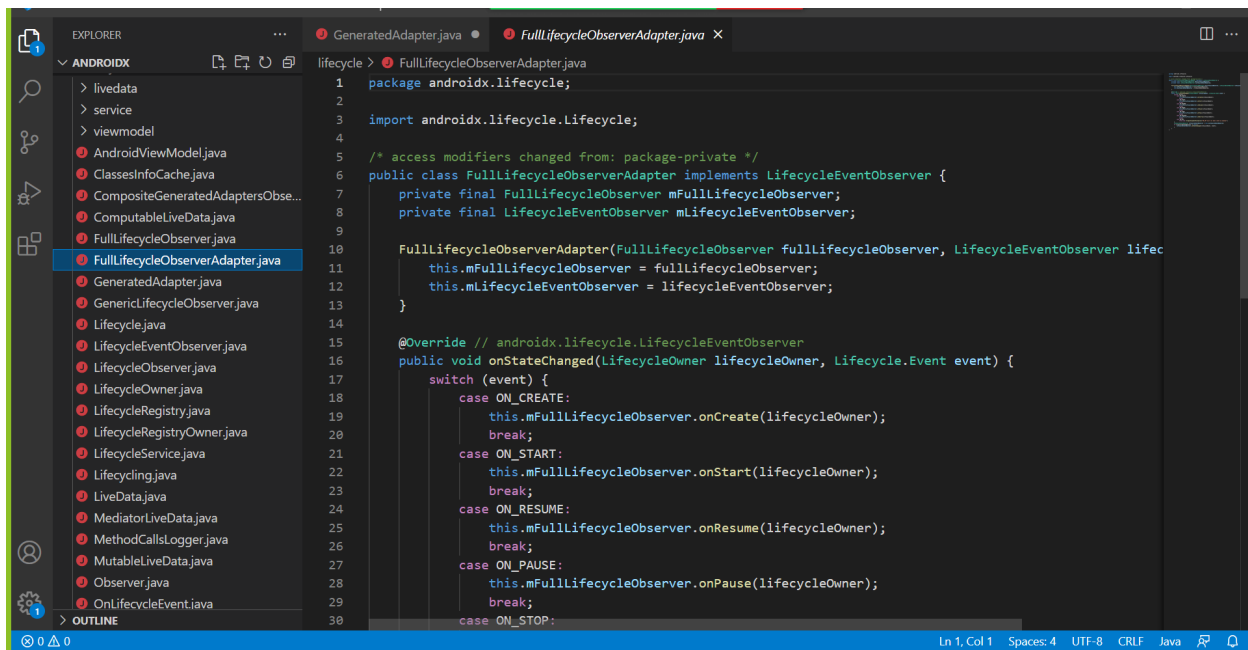
As we navigate through the code we have noticed the design patterns are observer and adapter which makes software less brittle , because it follows the standards .



Observer is a behavioral design pattern that allows you to establish a subscription mechanism for many objects to be notified of events that occur on the object they're observing. These systems can handle change because they minimize the interdependency between objects by supporting the principle of loose coupling between objects that interact with each other and can be added/removed at any point in time. Observers delegate the responsibility for monitoring for an event to a central object.



Adapter is convert the interface of a class into another interface clients expect. It follows the principle of clean code “DRY”, that avoid creating additional subclasses which lead to duplicate the code. The benefits are that you can change behaviour at runtime and you are not tied down to having many many different objects and the code will be reusable and flexible.



COHESION AND COUPLING

We notice that the modules are high cohesion because all methods in the class support central purpose . ex : here the class is SingleDocumentFile and all the methods inside the class support the document file such as createFile,lastModified,canRead , etc .

```
package androidx.documentfile.provider;

import android.content.Context;
import android.net.Uri;
import android.provider.DocumentsContract;
import androidx.annotation.Nullable;
import androidx.annotation.RequiresApi;

@RequiresApi(19)
class SingleDocumentFile extends DocumentFile {
    private Context mContext;
    private Uri mUri;

    SingleDocumentFile(@Nullable DocumentFile documentFile, Context context, Uri uri) {
        super(documentFile);
        this.mContext = context;
        this.mUri = uri;
    }

    @Override // androidx.documentfile.provider.DocumentFile
    public DocumentFile createFile(String str, String str2) {
        throw new UnsupportedOperationException();
    }

    @Override // androidx.documentfile.provider.DocumentFile
    public DocumentFile createDirectory(String str) {
        throw new UnsupportedOperationException();
    }

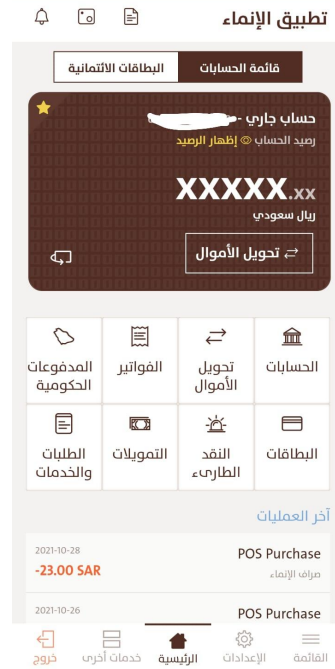
    @Override // androidx.documentfile.provider.DocumentFile
    public Uri getUri() {
        return this.mUri;
    }

    @Override // androidx.documentfile.provider.DocumentFile
    @Nullable
```

Also the application follows the principle of Low coupling since it uses design patterns that support of loose coupling such as observer pattern as shown in previous section .

GUI

GUI is very explicit ,the application opens with one click and you can easily find the buttons and scroll the app efficiency and it is familiar to find what we need, also window title bars and naming conventions are consistent which makes it usable.



CONCLUSION

In conclusion, we have offered several solutions to problems in our report that have been evaluated on quality from all sides, such as design, coding, standard, and security, using MobSF tools. The security score is 25/100, which is quite poor for banking apps, according to MobSF. To provide users with reliability, the security score must be raised. We've noticed that application permissions, such as accessing secondary storage, taking images, and monitoring locations, need to be further restricted. Another thing that manifest analysis provides a list of critical issues that must be handled, such as data decryption, content protection, and a number of different networking issues . Furthermore, we analyse the application design such as coupling and cohesion, and which design patterns that the application's use .However , the point of the application was the GUI which is very explicit and usable .In future , we expect that the issues will be solved as we suggested above and the code will follow the standard of clean code .