# Text Description automatically generated with medium confidenceA picture containing logo Description automatically generated

**Al-Imam Mohammad ibn Saud Islamic University**

College of Computer and Information Sciences

Computer Science Department

**CS392: Software Engineering 2**

**1st Semester 1443H / Fall 2021**

**Quality assessment Report of Alinma Bank App**

Workshop 1

|  |  |  |
| --- | --- | --- |
| **Name** | **ID** | **Email** |
| Sadeem Faisal Alqahtani | 440021429 | sfaalqahtani29@sm.imamu.edu.sa |
| Sarah Khalid Alaradi | 440023365 | skmaloridi@sm.imamu.edu.sa |
| Shoug Ali Alsuhaibani | 440022732 | ssuhaibani@sm.imamu.edu.sa |
| Sarah Abdullah Alsarami | 440020811 | samalsarami@sm.imamu.edu.sa |

**Section:** 371

**Supervised by**: Dr.Lamees Alhazaa

# Abstract

In this report, we were able to achieve the goal of a workshop focused on quality assessment for a local banking application, which was to provide relevant insights and improve the application's quality by performing a static analysis with the MobSF tool. This report looked at the overall quality of the Alinma Bank application in terms of clean code, code documentation, and security problems including application permissions, network security, and manifest analysis.

Table of Contents

[Abstract 2](#_Toc86281144)

[Introduction 4](#_Toc86281145)

[Alinma Static Analysis Results 5](#_Toc86281146)

[Security Issues 6](#_Toc86281147)

[1. Application Permissions 7](#_Toc86281148)

[2. APKID Analysis 9](#_Toc86281149)

[3. Browsable Activities 9](#_Toc86281150)

[4. Network Security 9](#_Toc86281151)

[5. Manifest Analysis 10](#_Toc86281152)

[6. Code Analysis 11](#_Toc86281153)

[7. Shared library Binary Analysis 12](#_Toc86281154)

[8. NIAP Analysis v1.3 12](#_Toc86281155)

[9. Domain Malware Check 13](#_Toc86281156)

[10. URLs 13](#_Toc86281157)

[11. Emails 14](#_Toc86281158)

[12. Trackers 14](#_Toc86281159)

[13. HardCoded Secrets 14](#_Toc86281160)

[Clean Code Analysis 15](#_Toc86281161)

[Meaningful Names 15](#_Toc86281162)

[Functions 17](#_Toc86281163)

[Error Handling 22](#_Toc86281164)

[Formatting 24](#_Toc86281165)

[Code Documentation Analysis 25](#_Toc86281166)

[Conclusion 25](#_Toc86281167)

[References 26](#_Toc86281168)

# Introduction

In March 2006, three Saudi government funds established Alinma Bank as an Islamic retail and corporate bank. Its initial owners were the Saudi Arabian General Investment Fund (GIF), the Saudi Arabian General Retirement Fund (GRF), and the Saudi Arabian General Organization for Social Insurance (GOSI). The company’s headquarters is located in Riyadh, Saudi Arabia.

To maintain delivering the best services to their customers with the technological revolution pace that’s happening in Saudi Arabia, Alinma provided an online banking service that simplifies their customers' financial lives and gives them access to a wide range of transactions and services at their convenience.

This report will outline the studies and the quality assessments of Alinma Bank app in order to provide profound insights and improve the overall quality of the app. This well be achieved by performing a static analysis on the Android version of the app by using the MobSF tool, and automating some of the source code quality analysis through the use of Codacy.

Going through this workshop taught us that even critical apps that were developed by big companies can have flaws. Maintenance is a very important part in the life cycle of the app development process because as we saw these minor problems that we found could cause the app a big failure that could have been avoided with maintenance and proper documentation.

# Alinma Static Analysis Results

Starting with a walkthrough of how we retrieved application's data (codes, docs, designs, etc.).

By using the MobSF tool, we download a docker and uploaded an APK file of the Alinma banking app. The results given by the tool were: Security score, tracker detection, size of the file, latest app version and the different analysis options. Such analysis options were network, manifest, code, binary and security analysis which was not available due to security measures. After using the tool, many analysis titles were found. We have assessed each one separately.

Going into detail of how we downloaded the tool, A docker was downloaded and some command lines were used in the terminal to help run the tool.

The two commands below were used to run the tool:

1.docker pull opensecurity/mobile-security-framework-mobsf

2.docker run -it --rm -p 8000:8000 opensecurity/mobile-security-framework-mobsf:latest

After the commands were run, an IP address was given that was associated for the MobSF tool. An APK file was download via google play and uploaded unto the tool. After a while of processing the results page was finally loaded.

Graphical user interface, application, website

Description automatically generated

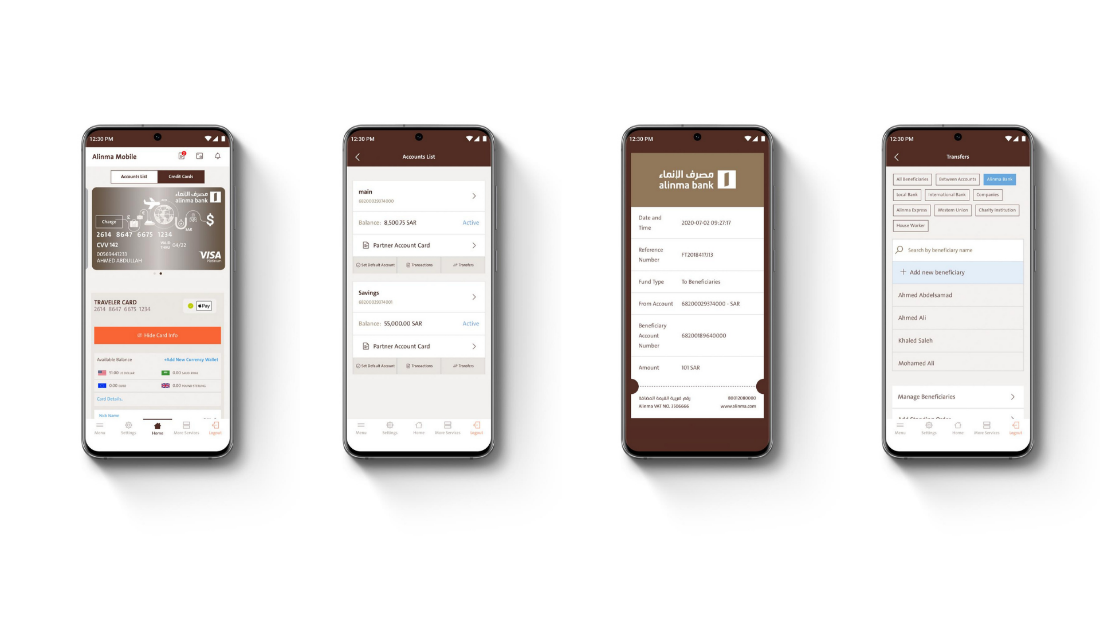
For the clean code analysis, we used Codacy to run the code analysis with the help of GitHub. A repository of the Alinma java code was uploaded to GitHub and linked to Codacy. After running the tool, issues of the code were available for preview.

Graphical user interface, text, application

Description automatically generated

# 

Before going in details about the issues we found, we would like to do some design analysis for the app pointing out some of the design issues in it.



Alinma bank application provides a set of features that allow users to securely conduct financial transactions and manage their accounts – anytime, anywhere.

The interface of the application is modern and reflects their entity, the colors, fonts, buttons, layout structure and photograph style are consistent with a compatible resolution size to android devices. A simple GUI that makes it easy to use, but we should point out that there is no guidance as a critical system like bank must be clear how to use it. Account cards are visible so clients can immediately see all their accounts. Massive functionalities in the homepage are unnecessary.

The app has some issues that makes it not efficient enough to users to perform their tasks quickly through the easiest process, such as when logging in or purchases received the OTP SMS is delayed too long, every weekend they undergo maintenance without a previous announcement. Also, it takes a long time to log in to the app beside the slow navigation between the interfaces.

# Security Issues

The following part is regarding the different analysis results given by the MobSF tool which were regarding the security part of the app. We have taken examined each segment thoroughly. We stated the problems within each title and some solutions that could be done to enhance the app. The Segments include: Analysis of the Application permissions, network security, manifest analysis and so on.

## Application Permissions

Via android static analysis report for Alinma bank we will discuss permissions as it is:

**Dangerous:**

**android.permission.WRITE\_EXTERNAL\_STORAGE:** This permission is allowing an application to write and modify, read on external memory, it is dangerous permission, but at first the user is asked if he allows access or not. Since it is rated dangerous it means that the user will be asked permission the best practices used to request dangerous permissions.

**android.permission.CAMERA:** it is dangerous because this permission allows application to take pictures and videos with camera, collect images, and seeing at any time, in some cases, user privacy may to break through this feature by malicious Applications, but at first the user is asked if he allows access or not Since it is rated dangerous it means that the user will be asked permission the best practices used to request dangerous permissions.

**android.permission.ACCESS\_COARSE\_LOCATION:** In this permission, when you access the coarse location sources such as the database of the mobile phone network, then phone location is determined approx., and malicious applications can use this to determine your Location approx.

**android.permission.ACCESS\_FINE\_LOCATION:** When this permission is available, the phone will be located globally via GPS, which may allow malicious applications to determine your location well, also consumes an additional battery power.

This feature is a unique feature of mobile applications and bringing it to an application is important and causes a lot of facilitation and contextual experience for users, but it can be exploited in the wrong way and is used maliciously to determine your location and saves and is exploited by malicious users and in order to preserve privacy and user location data, location permissions are requested ,Since it is rated dangerous it means that the user will be asked permission the best practices used to request dangerous permissions.

**Normal:**

**android.permission.USE\_FINGERPRINT:** It is a good feature and bring to application that a distinct experience for users, but this constant was deprecated in API level 28. Application in this case Applications should request USE\_BIOMETRIC instead.

**android.permission.RECEIVE\_BOOT\_COMPLETED**: this permission is allowing the app to have itself started as soon as the system has finished booting. This can make it take longer to start the phone and allow the app to slow down the overall phone by always running, which means that the problem of slowing down the device is created from allowing this permission in the application, Because the processor is under loading pressure apps.

**android.permission.INTERNET** (full Internet access) this is permission it normal it does not need to asked user if allows or not it just Allows an application to create network sockets, And the same with the rest of the normal permissions unless there is a problem.

**android.permission.ACCESS\_NETWORK\_STATE:** Allows an application to view the status of all networks if it ( (wifi ,3g,4g) also you can check if the network is connected or not.

**android.permission.FLASHLIGHT:** (control flashlight) allows the application to control the flashlight without activating the camera hardware.

**com.sec.android.provider.badge.permission.READ:**( Show notification count on app) Show notification count or badge on application launch icon for Samsung phones.

**com.sec.android.provider.badge.permission.WRITE:** (Show notification count on app) Show notification count or badge on application launch icon for Samsung phones.

**com.htc.launcher.permission.READ\_SETTINGS**: (Show notification count on app) Show notification count or badge on application launch icon for htc phones

**com.htc.launcher.permission.UPDATE\_SHORTCUT:** (Show notification count on app) Show notification count or badge on application launch icon for htc phones.

**com.sonyericsson.home.permission.BROADCAST\_BADGE** (Show notification count on app) Show notification count or badge on application launch icon for sony phones.

**com.sonymobile.home.permission.PROVIDER\_INSERT\_BADGE:**( Show notification count on app) Show notification count or badge on application launch icon for sony phones.

**com.anddoes.launcher.permission.UPDATE\_COUNT:**( Show notification count on app) Show notification count or badge on application launch icon for apex.

**com.majeur.launcher.permission.UPDATE\_BADGE :**( Show notification count on app) Show notification count or badge on application launch icon for solid.

**com.huawei.android.launcher.permission.CHANGE\_BADGE**:( Show notification count on app) Show notification count or badge on application launch icon for huawei phones

**com.huawei.android.launcher.permission.READ\_SETTINGS** :( Show notification count on app) Show notification count or badge on application launch icon for huawei phones

**com.huawei.android.launcher.permission.WRITE\_SETTINGS**:( Show notification count on app) Show notification count or badge on application launch icon for huawei phones

**android.permission.READ\_APP\_BADGE :**(show app notification) Allows an application to show app icon badges

**com.oppo.launcher.permission.READ\_SETTINGS**: (Show notification count on app) Show notification count or badge on application launch icon for oppo phones.

**com.oppo.launcher.permission.WRITE\_SETTINGS:** (Show notification count on app) Show notification count or badge on application launch icon for oppo phones.

**android.permission.WAKE\_LOCK:** (prevent phone from sleeping) Allows an application to prevent the phone from going to sleep.

**android.permission.VIBRATE** : (control vibrator) Allows the application to control the vibrator.

**Unknown:**

There are unknown permissions and they were written in the report, but it is not clear what they are and what they do, such that **com.amazon.device.messaging.permission.RECEIVE com.alinma.retail.mobile.permission.RECEIVE\_ADM\_MESSAGE , me.everything.badger.permission.BADGE\_COUNT\_READ , me.everything.badger.permission.BADGE\_COUNT\_WRITE,**

**com.google.android.finsky.permission.BIND\_GET\_INSTALL\_REFERRER\_SERVIC.**

**Signature:**

**c2dm.permission. RECEIVE:** Permission for cloud to device messaging,it is received data from Internet allows apps to accept cloud to device messages sent by the app's service. (Using this service will incur data usage. Malicious apps could cause excess data usage.).

## APKID Analysis

Table

Description automatically generatedThis analysis was not familiar to any team member, so it won't be discussed. This is a sample of it.

## Browsable Activities

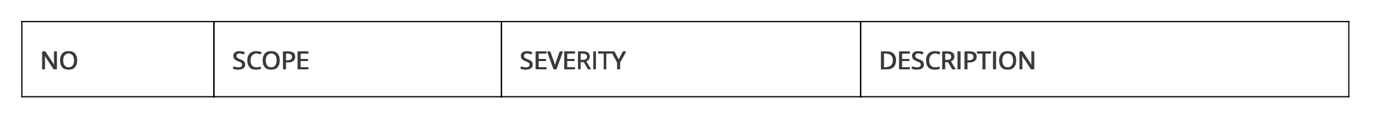
This title also was unfamiliar to any team member, so it won't be discussed. This is a sample of it.

Table

Description automatically generated

## Network Security

The network analysis was not provided, most likely for security purposes. This is a sample

of it.

## Manifest Analysis

In the manifest Analysis section of the report, several issues between medium to high severity appeared, some of it was understood by the team and there are some out of the team’s knowledge, but it will be discussed for the cause of explaining the reported issues.

Starting with **Clear text traffic,** which is a high severity issue were the app intends to use cleartext network traffic, such as cleartext HTTP, FTP stacks, DownloadManager, and MediaPlayer. The default value for apps that target API level 27 or lower is "true". Apps that target API level 28 or higher default to "false". The key reason for avoiding cleartext traffic is the lack of confidentiality, authenticity, and protections against tampering; a network attacker can eavesdrop on transmitted data and also modify it without being detected.

Followed by Medium severity issue **Application Data can be Backed up flag is missing**, the flag [android: allowBackup] should be set to false. By default, it is set to true and 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.

There was a number of issues (with a high severity rate) regarding a various Broadcast Receivers used in the app, they either were not Protected or Protected by a permission, but the protection level of the permission should be checked.

When we say **Broadcast Receiver is not Protected** it means the Broadcast Receiver is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device. The presence of intent-filter indicates that the Broadcast Receiver is explicitly exported.

And a **Broadcast Receiver is Protected by a permission, but the protection level of the permission should be checked** indicates that 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. It is protected by a permission which is not defined in the analyzed application. As a result, the protection level of the permission should be checked where it is defined. If it is set to normal or dangerous, a malicious application can request and obtain the permission and interact with the component. If it is set to signature, only applications signed with the same certificate can obtain the permission.

We also encountered high severity issues with **Services that were not Protected with an existent of intent-filter** meaning a Service is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device. The presence of intent-filter indicates that the Service is explicitly exported.

Finally, a **Content Provider is not Protected** is a high severity issue, the Content Provider is found to be shared with other apps.

## Code Analysis

We discover weaknesses in source code that it leads to vulnerabilities depending on the code analysis report, some of these issues just warning and the others are highly severe.

The App logs information for some reasons to create log files on a mobile device, such as keeping track of crashes, errors, and usage statistics. However, logging sensitive data may expose the data to attackers or malicious applications, and it may also violate user confidentiality.

When you write an application you should make sure that it does not have cryptographic algorithms and protocols that have significant known weaknesses or are otherwise insufficient for modern security requirements. Algorithms that were previously regarded as secure may become insecure over time; as a result, it's critical to review current best practices and update configurations as needed. This App uses the encryption mode CBC with PKCS5/PKCS7 padding and this configuration is vulnerable to padding oracle attacks and also the app has MD5 is a weak hash known to have hash collisions that occur when an attempt to find two input strings of a hash function that produce the same hash result. Algorithms with known weaknesses should be replaced and need to read about algorithm choice for more secure alternatives.

Every Android-compatible device supports Files saved to external storage that are world-readable. Usernames, passwords, keys, and other sensitive information may be hardcoded in files. Data written to External Storage can be read by any app. So, the external storage can be used by an attacker to allow for arbitrary control of the application in some cases. Finally, when storing sensitive data—data that shouldn't be accessible from any other app—use internal storage, preferences, or a database. Internal storage provides the advantage of keeping data hidden from users.

Since a rooted device is much more at risk of being compromised, it's crucial to be aware of it. And this App may have root detection capabilities. Also, this App may request root (Super User) privileges that allow users to control the root settings.

The app uses SQLite Database and executes the raw SQL query. A SQL injection attack involves integrating SQL commands into input data, mimicking the syntax of a predefined SQL command. A successful SQL injection attack allows the attacker to read or write to the database and possibly execute administrative commands, depending on the permissions granted by the server. Untrusted user input in raw SQL queries can cause SQL Injection. Also, sensitive information should be encrypted and written to the database. To prevent this, make sure that input validation and parameterized queries including prepared statements.

This App copies data to clipboard. The clipboard is accessible system-wide and is therefore shared by apps. Malicious apps can take advantage of this sharing to access sensitive data stored in the clipboard. Be aware sensitive data should not be copied to clipboard.

## Shared library Binary Analysis

Table

Description automatically generatedThis title also was unfamiliar to any team member, so it won't be discussed. This is a sample of it.

## NIAP Analysis v1.3

Table

Description automatically generatedThis title also was unfamiliar to any team member, so it won't be discussed. This is a sample of it.

## Domain Malware Check

Table

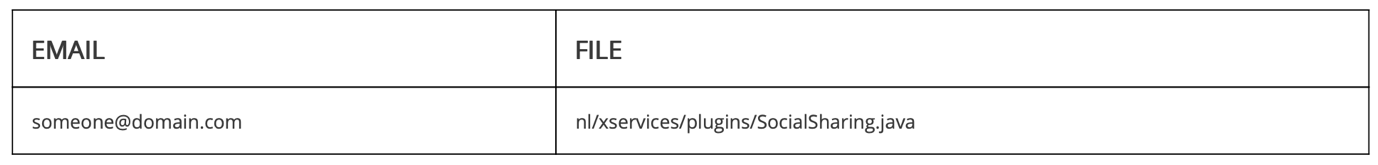
Description automatically generatedThis title also was unfamiliar to any team member, so it won't be discussed. This is a sample of it.

## URLs

Table

Description automatically generatedThis title also was unfamiliar to any team member, so it won't be discussed. This is a sample of it.

## Emails

This title also was unfamiliar to any team member, so it won't be discussed. This is a sample of it.

## Trackers

Table

Description automatically generatedThis title also was unfamiliar to any team member, so it won't be discussed. This is a sample of it.

## HardCoded Secrets

Table

Description automatically generated with medium confidenceThis title also was unfamiliar to any team member, so it won't be discussed. This is a sample of it.

# Clean Code Analysis

With the help of Codacy and some manual work, code issues were brought up. We have compared some code patterns related to the clean code by using the clean code handbook.

## Meaningful Names

Names are everywhere in the software, classes, functions and variables. So, they have a great importance in the continuity of the code and modification and development by other developers. Choosing good names takes time but saves more than it takes, one of the importance of a good name is that it be easy, clear, and understandable to Everyone who reads your code (including you).

Text

Description automatically generatedthere are some restrictions on names in many respects:

Graphical user interface, text

Description automatically generatedClass names should be nouns or noun phrases that denote something because it is impossible for these non-significant characters to be translated by other programmers and they do not have to.

The problem of a single letter causes confusion into what is the reason for choosing this letter over others why you don’t choose k or f? and what does it mean? and there is no logical reason.

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidenceThe method name must be verb and verb phrases in her error it just words you can be prefixed with is or set, according to the JavaBean standard.4

Text

Description automatically generatedThis method name is the same on enclosing class which it is not-constructor and should be avoided using the same word for two purposes, also the name is meaningless because its letters.

This name of method is very long. In contrast, the reader is a programmer and understands the technical terms and does not need to clarify the name pulling each name from the scope of the problem is so much it is better to abbreviate it and make it understandable.

Graphical user interface, text

Description automatically generated with medium confidence

These names are similar and differ slightly, and this can cause wasting a lot of time to discover the difference between them in the future, also It is possible that the developer selects an object by name without seeing your comments because there is no clear difference.

## Functions

Functions are the first line of organization in any program. Writing them well saves you a lot of trouble.

When methods are excessively long this usually indicates that the method is doing more than its name/signature might suggest which also means a violation of the high cohesion design principle. As will as They become challenging for others to digest since excessive scrolling causes readers to lose focus. with just a few simple method extractions, some renaming, and a little restructuring by removing any copy/pasted code lead us to closer to a small and one level of abstraction function.

This is very long method. thus, we could not contain it in one picture, it consists of 130 lines!

Text

Description automatically generated

Duplication is a problem because it bloats the code and will require four-fold modification should the algorithm ever have to change. It is also a four-fold opportunity for an error of omission. And as we saw in the code, there were a lot of duplicated blocks and some of them could have been avoided.

Duplicated blocks from line 605-619 and from 620-634.

Text

Description automatically generated

Using a branching statement as the last part of a loop may be a bug, and/or is confusing. Ensure that the usage is not a bug, or consider using another approach. And an Empty if statement blocks were encountered a lot what is the point where a condition is checked but nothing is done about it. This an example combining the two problems, empty if statement block shown at line 100 and 102 and the branching statement is shown in line 108.

Text

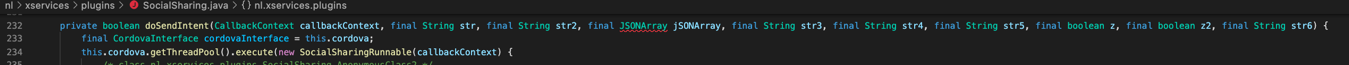
Description automatically generated

The NPath complexity was encountered a lot in the code, which indicates the number of acyclic execution paths through a method. Meaning that the NPath counts the number of full paths from the beginning to the end of the block of the method. That metric grows exponentially, as it multiplies the complexity of statements in the same block. Below you can see an example method of it.

Text

Description automatically generated

Methods with numerous parameters are a challenge to maintain, especially if most of them share the same datatype, and they are even harder from a testing point of view. Imagine the difficulty of writing all the test cases to ensure that all the various combinations of arguments work properly. If there are no arguments, this is trivial. If there’s one argument, it’s not too hard. With two arguments the problem gets a bit more challenging. With more than two arguments, testing every combination of appropriate values can be daunting.



Text

Description automatically generatedAn Empty switch statements serve no purpose and should be removed. See below for the example.

Methods should have a descriptive unambiguous name that describes what the function does. A long descriptive name is better than a short enigmatic name. A long descriptive name is better than a long descriptive comment. The picture below contains several methods with inscrutable names.

Text

Description automatically generated

Text

Description automatically generatedAs we mentioned earlier Functions should do one thing and one thig only. Error handling is one thing. Thus, a function that handles errors should do nothing else. This implies that if the keyword try exists in a function, it should be the very first word in the function and that there should be nothing after the catch/finally blocks. But in the example here we see that the function is doing other thing beside Error handling.

After seeking looking for issues and rules violations, we should highlight the programmers' good practices that been followed when writing the code. Per to our observations, the whole code's structure satisfied the low coupling design principle.

Always keep in mind when writing functions, that they will come out long and complicated. They will have lots of indenting and nested loops. They will have long argument lists. The names will be arbitrary, and there will be a duplicated code. But we also should have a suite of unit tests that cover every one of those clumsy lines of code.

So then we should refactored refine that code, splitting out functions, changing names, eliminating duplication. And sometimes shrinking the methods and reordering them. Even Sometimes break out whole classes, all the while keeping the tests passing.

In the end, we will wind up with functions that follows the rules we discussed. And there is more of it, but always keep in mind we do not write them that way to start. we do not think anyone could.

## Error Handling

Error handling is just one of those things that we all must do when we program. Devices can fail and input can be unexpected. In short, things can go wrong, and it is our responsibility as programmers to ensure that our code does what it is supposed to do when they do. Error handling is necessary, but it is incorrect if it obscures reasoning.

Text

Description automatically generatedText

Description automatically generatedIn some techniques for handling and reporting errors were limited. You either set an error flag or returned an error code that the caller could check. The caller must check for errors immediately after the call, this is easy to forget. For this reason, it is better to throw an exception when you encounter an error. The calling code is cleaner.

Notice how much cleaner it is. This isn’t just a matter of aesthetics. The code is better because two concerns that were tangled, the algorithm for device shutdown and error handling, are now separated.

To satisfy your tests, try writing tests that force exceptions and then adding behavior to your handler. When you run code in the try component of a try-catch-finally statement, you are stating that execution can stop at any time and continue at the catch. Whatever happens in the try, your catch must leave your program in a consistent condition, perform finally block that have important code in our program that needs to be executed irrespective of whether the exception is thrown. As a result, when writing code that may throw exceptions, it is a good way to start with a try-catch-finally statement.

Graphical user interface

Description automatically generatedGraphical user interface

Description automatically generatedExamples of try-catch-finally exception handling:

Throwing a specific error/exception rather than a generic/general exception is a recommended practice in Java. However, to make your code cleaner Avoid throwing certain exception types. Rather than throw a raw RuntimeException, Throwable, Exception, or Error, use a subclassed exception or error instead.

Graphical user interface

Description automatically generated with low confidenceGraphical user interface

Description automatically generated with low confidenceThese some examples in Alinma application source code:

Graphical user interface, text, application, email

Description automatically generatedYou should provide enough context for each exception you throw to determine the source and location of an error. Create informative error messages and pass them along with your exceptions. Mention the operation that failed and the type of failure. If you are logging in your application, pass along enough information to be able to log the error in your catch.

**Avoid** throwing NullPointerExceptions manually. To avoid a method being called with a null parameter, you may consider using an IllegalArgumentException instead, making it clearly seen as a programmer-initiated exception. However, there are better ways to handle this:

other exceptions are standardly used for certain kinds of illegal arguments and states. If a caller passes null in some parameter for which null values are prohibited, convention dictates that NullPointerException be thrown rather than IllegalArgumentException.

A screenshot of a computer

Description automatically generated with medium confidenceTo implement that, you are encouraged to use java.util.Objects.requireNonNull() This method is designed primarily for doing parameter validation in methods and constructors with multiple parameters.

## Formatting

Following a specific formatting method can really help the programmers when reading the code after sharing it with others, the code is then easier on the eyes.

Using proper indentation also falls under formatting but indentation should be taken into consideration what programming language we used. Indentation could also show were the code falls in the hierarchy of the code.

In this section of code, vertical alignment could’ve been followed where all the values would be in one vertical column.

Text

Description automatically generated

# Code Documentation Analysis

The purpose of code documentation is to compensate for the times we need to express ourselves in the code. Good comments are beneficiary in times where we do not understand the code written which would save us time. It is good to include informative comments, such as the parameter types and what is the return value type for easy understanding. Developers should have included these so that other team members have an easier time understating.

Below are some screenshots of some segments of the java code for the Alinma app. They show the lack of documentation that the app developers should have included. The lack of documentation could be a security precaution.

Text

Description automatically generated

They should’ve explained the purpose of this class, and the purpose of the assigned variables as comments.

Text

Description automatically generated

The purpose of the method was not explained as a comment.

Overall, the java codes showed very poor documentation, there were rarely any comments explaining the purposes of the classes nor the methods. It did not follow the Javadoc style guide as there were not headers before the classes nor sufficient comments in the classes.

# Conclusion

To conclude this analysis report of our chosen banking app, we learned that a lot of factors are included in the inspection and the evaluation of the app from the security analysis all the way to the documentation analysis of the code. Using the MobSF tool helped sum the different segments in the app, we were able to look at each segment individually to know which flaws were detected. The purpose of this is to show that even big branded and known apps and systems will have flaws, and we need to analyze in order to reduce the flaws and bring up the overall score of the system or app. Maintenance needs to be a big part of the app throughout the years to reduce these minor problems. This workshop assessed the application only, must work on the maintenance of the issues that have been discussed in our report to improve the overall quality and avoid failure to the application.

# References

1. Arabian Business

[www.arabianbusiness.com](http://www.arabianbusiness.com)

1. How to install and Use MobSF on macOS | Android Penetration Testing Suite

[www.youtube.com](https://www.youtube.com/watch?v=LzHSwa_Ef_o)

1. How to install Docker on Mac OS in 3 minutes

[www.youtube.com](https://www.youtube.com/watch?v=LtooWDUL1Js)

4. How to use MobSF

[www.youtube.com](https://www.youtube.com/watch?v=rFRuzxGMeM8)

5. Clean Code by Robert C. Martin