



Implementation of Logging Feature in Android Payment SDK using Scrum Method

Ardian Dwi Rifai¹, Erika Ramadhani²

Informatics Department, Faculty of Industrial Technology, Universitas Islam Indonesia

¹17523093@students.uii.ac.id, ²115230409@uui.ac.id

Abstract

Electronic money or e-money is a technological innovation that allows payment transaction processing to be easier and more practical. PT Aino Indonesia is a payment gateway company that serves payment processes using this technology for various business segments. To satisfy the requirements of each business segment, Aino needs to develop a wide range of applications. These applications require a standard for the payment system to make it faster to integrate and maintain. The payment team at Aino has developed a module or SDK that provides a standard payment system for Android-based applications. Two new features will be added to this SDK, the logging feature and log file upload. The logging feature will record the data generated by each transaction process into a log file. Next, the SDK will upload the log files to the server to help retrieve transaction data uploaded from the device. Both features will help the process of tracking issues and problems that may arise during the transaction process. The development of the two features in this SDK uses the Scrum method to ensure it stays on track.

Keywords: logging, SDK, Scrum, Logback, library

1. Introduction

The rapid development of information technology can make it easier for people to carry out their daily activities. This has major impact on the development in people's economic life. We can see this in the form of payment innovations that no longer use cash or what is commonly referred to as non-cash payment system. One of the technologies that are being used by many people is electronic money or e-money. Electronic money is the value of money or prepaid products with the value or funds are stored in an electronic device [1]. Smartcards are one of the many technologies that are used to store electronic money. The amount of funds or the value of the money are stored in a microchip that have been embedded in the smartcard. Electronic money technology can be a payment solution with a relatively fast and inexpensive process because the value of money is stored in safe, inexpensive media that can be used offline [2]. PT Aino Indonesia is a company engaged in integrated and non-cash electronic payment services. Aino is a subsidiary company of PT Gama Multi Usaha Mandiri, a holding company of business units owned by Universitas Gadjah Mada. The business segments served by Aino include public transportation, toll roads, parking, tourism, and retail. Aino has a goal to become an integrated payment system solution provider company. With this goal, Aino wants to realize the vision of a less-cash city so that the use of cash payments can be reduced, and the use of integrated electronic payments can be maximized.

Aino has developed many contactless payment systems by integrating e-money products issued by several domestic banks. One of the systems developed is a mobile point of sales or mPOS device based on the Android operating system. This device can process chip-based transactions using smartcards and server-based transactions. These devices are equipped with a special application developed by Aino. Mobile Point of Sales devices are usually used in the retail segment such as food and beverage merchants,



vending machines, MSMEs, and parking booth. This mPOS device provides several advantages such as portability, applications that can be adapted to business needs, and the ability to accept multiple payment sources. With this system, service and transaction processes can be done easily and quickly to improve user experience and save costs previously required with traditional cashier tools [3].

To meet the requirements of diverse business segments, Aino develops different applications for each segment. The differences can be seen the form of interface design, features developed, how operate it and so on. Even though they are different, the payment features still must follow the process that has been determined by the Aino payment segment. This causes the same questions often arise regarding the integration of payment systems in the applications. These frequent questions can slow down the app development process and performance. A special module is needed to handle payment system integration across applications. The module developed is an SDK or software development kit for devices with the Android operating system. This SDK can help developers integrate Aino's payment systems into the applications. There are several features provided, including chip-based payments, server-based payments, and printing proof of transactions. Two new features will be added, log messages to file and upload the log files to the server. The addition of the logging feature to the SDK can be a solution to problems that often arise during operation in field applications. Some apps that have been developed do not have this feature. As a result, it is necessary to send personnel to the field to deal with the problems. It takes a lot of time and money to send personnel to check and fix the problems.

2. Research Methodology

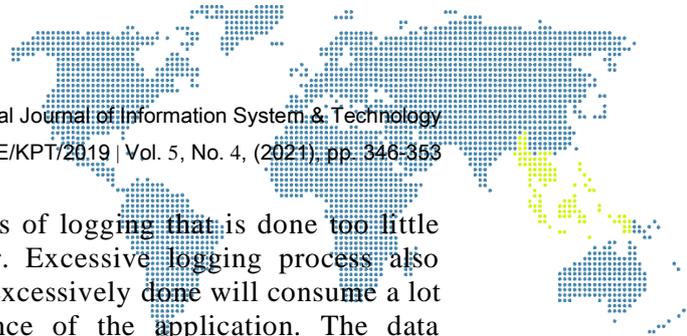
The development of the logging feature implements the Agile method with the Scrum framework. Scrum is a framework or framework in Agile software development that is iterative and incremental[9]. Scrum is a framework that gives a team ability to solve complex and adaptive challenges in the development and delivery of a quality product by leveraging collaboration, creativity, and productivity [10]. Scrum prioritizes the speed and flexibility of the software development process by ensuring the quality that has been tested[11].

In Scrum, the software development processes are done in several short and iterative cycles called sprints. One sprint usually lasts between two to four weeks. Each team member is committed to complete a job that has been selected and determined at the start of the sprint. This cycle will be repeated until the Scrum objectives have been met. Scrum consists of several roles called the Scrum Team, artifacts that represent the work or value performed by a team, and the activities performed (events). In a sprint, several meetings involve all members of the Scrum team, including Daily Scrum, Sprint Planning, and Weekly Meetings.

2.1. Logging

Logging in an application is a systematic and controlled way to represent the condition of an application so that it can be understood by humans[4]. A log can record various events that occur when the application is running. These log records can help app developers to analyze and identify problems that arise. The recorded information can provide clues as to which part of the application is causing the undesired behavior [5]. The logging process is quite beneficial to do on software that resides in a production environment. Message logs can increase the average time it takes to diagnose an error up to 2.2 times faster than an error diagnosed without a message log[6].

Given the importance of the logging process, it must be well designed and structured. A well-designed logging process should produce information relevant to



events and not create new problems. The process of logging that is done too little cannot give meaningful clues to the developer. Excessive logging process also cannot provide good benefits. Processes that are excessively done will consume a lot of resources and interfere with the performance of the application. The data generated is also not optimal due to redundant and useless data. To avoid these problems, developers need to pay attention to which part of the logging is executed and what information needs to be logged on the system being developed[7].

2.2. Logback

To simplify and speed up the development of logging, a third-party library is needed. Logback is a logging framework library for Java-based applications. Logback implements the SLF4J facade. By implementing this facade, the process of replacing the framework can be done easily. Logback is the successor of log4j which is also a library for the logging process that has several advantages when compared to its predecessor [8]. Even so, both frameworks were developed by the same developer so they both have almost the same concept and usage.

Logback architecture consists of several layers that have different functions. Three main components help application developers to develop the logging process and control how the process is executed. The first component is a logger that can perform the logging process. The second component is the appender that has the responsibility to redirect log messages to one or more destinations such as console, file, and socket. The third component is the layout, which determines how the logging information format is printed in a predefined style.

3. Results and Discussion

3.1. Discussion

Before the scrum begins, the scrum team will perform the inception step. The purpose of this step is to communicate the objectives, vision, and context of the project so that it can be the basis for decision-making and execution of the developed project [12]. Some of the activities in this step including making product backlogs, story mapping, defining the definition of done, planning poker, and determining team composition. The Product Owner leads the inception step. The user story method is applied in the story mapping activity to understand the needs and specifications of the feature. User Story is a method to describe the needs of a product from the perspective of the client or user [13]. The Scrum team performed followings activities in the sprint cycle:

a) Sprint Planning

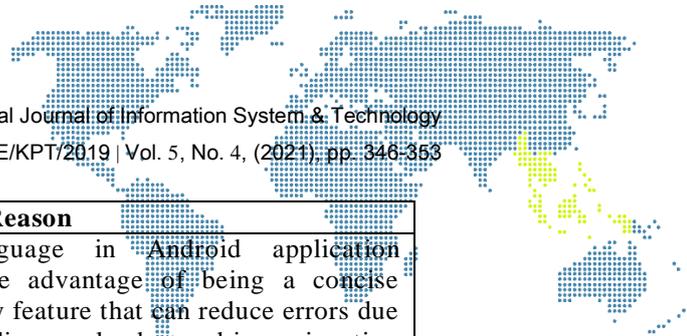
Sprint Planning is the initial meeting in a sprint that marks the start of the sprint cycle. The Product owner leads this meeting to plan and discuss the Product Backlog that has been created and how to achieve the objectives of the Sprint. The team usually delivers Sprint Planning in one day with a duration of approximately eight hours[9]. This logging feature development process will be developed for two sprints. Each Sprint will last two weeks. The Sprints are planned as shown in Table 1. This meeting also discusses the technology to be used as shown in Table 2.

Table 1. Sprint plans

Activity	Sprint amount
Develop logging and uploading the log file	1 Sprint
Integrate the project and create a documentation	1 Sprint

Table 2. Technologies used in the development process

Technology	Reason
Kotlin	Kotlin is an open-source programming language and has



Technology	Reason
	become a first-class language in Android application development. Kotlin has the advantage of being a concise language and has a null safety feature that can reduce errors due to NullPointerException. Kotlin can also be used in conjunction with Java-based code.
Logback Library	Logback has a variety of configuration options for logging using xml files.
FTPClient Library	FTPClient is easy to use and maintain

b) Daily Scrum

Daily Scrum is a meeting that is performed every day during the sprint iteration. This meeting lasted for approximately fifteen minutes and started at 09.30 am. At this meeting, each member of the team reports on what has been done the previous day, the work to be done, and the obstacles that arise. All the activities during the sprint are reported using the Jira application. This application makes it easy to monitor the project and to track issues, bugs, or problems that arise. The Daily Scrum is a form of the transparency principle of the Scrum method. With this, all the team members must be transparent about ideas, opinions, and problems encountered during the sprint.

c) Sprint Review

The team performed the Sprint Review on the last day of the sprint cycle. The duration of the meeting is approximately four hours. At this meeting, the Product Owner and Development Team review what has been achieved and what changes have occurred in the Sprint. Product review is carried out with demonstrations to observe how far the development process has been done. This demonstration will then be considered for backlog adjustments that will be carried out in the next sprint iteration. For this reason, the transparency of every member of the Scrum team is crucial to ensure that the project stays on the right track.

3.2. Result

Two features are developed in this iteration, logging to file and uploading the log files to server. The logging to file feature is developed using a third party Logback library. This library has many features, such as writing messages into files with specific configurations, file compression, and logging message formats. Logback has a rollover capability that allows writing log messages to a file and changes the target file when a specified condition is met. This library can speed up the feature development process and simplify integration with other features. The Logback configuration process can be done in two ways, directly in the program code or using an XML file. The configuration is done using an XML file because this way Logback will maintain the configuration even if the device is turned off. Figure 1 shows the Logback configuration in the logback.xml file.

```

1 <configuration>
2   <property name="DATA_DIR" value="/storage/emulated/0/FilesLogs" />
3   <appender name="FILE_APPENDER-${id}" class="ch.qos.logback.core.rolling.RollingFileAppender">
4     <file>${DATA_DIR}/currentlog.log</file>
5     <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
6       <fileNamePattern>${DATA_DIR}/log.%d{yyyy-MM-dd}-${id}(M,aux).%i.log.zip</fileNamePattern>
7       <maxHistory>1</maxHistory>
8       <maxFileSize>3MB</maxFileSize>
9       <totalSizeCap>1GB</totalSizeCap>
10    </rollingPolicy>
11    <encoder>
12      <pattern>%d %logger{55} - %msg%n</pattern>
13    </encoder>
14  </appender>
15  <root level="DEBUG">
16    <appender-ref ref="SIFT" />
17  </root>
18 </configuration>

```

Figure 1. Logback configuration in the xml file



The Logback configuration is performed using the `RollingFileAppender` component with the `SizeAndTimeBasedRollingPolicy` rule. This component supports the logger object to log messages into a specified file. The file naming is done using the `fileNamePattern` component. The `fileNamePattern` component will cause the rollover process to be executed daily with the file name according to the day and the order in which the file was created. To add a file compression process, the `fileNamePattern` file is added `.zip` element. With the specified rollover policy, Logback will limit the size of a file to 3 MB maximum, if it exceeds that size then the rollover process will be executed. Files will be preserved on the device with a limit of fourteen days. If the sum of all logfiles exceeds 1 GB, the oldest log files on the device will be deleted.

The use of the logging feature is done by accessing two methods that have been created. The first method is `kickstart` which is responsible for setting up the logger object. The `write` method is used to perform the logging process. Figure 2 shows the code for the `kickstart` and `write` program methods.

```

13 class TryLogging : CoroutineScope {
14     private val job = Job()
15     override val coroutineContext: CoroutineContext
16         get() = Dispatchers.IO + job
17
18     private lateinit var logger: ch.qos.logback.classic.Logger
19     private val LoggerContext = LoggerFactory.getLoggerFactory() as LoggerContext
20
21     fun write(tag: String, identifier: String, message: String) {
22         this.launch {
23             Log.e(tag, "msg: $identifier $message")
24             logger.debug("{} {} {}", tag, identifier, message)
25         }
26     }
27
28     fun kickStart() {
29         /* Starting the Logger */
30         logger = LoggerFactory.getLogger("name: TestLogger") as ch.qos.logback.classic.Logger
31         StatusPrinter.print(LoggerContext)
32     }
33 }
34
35 }
    
```

Figure 2. The source code for `kickstart` and `write` method

The logging process is tested with a simulation application. This application has a simple interface that only shows two buttons as shown in Figure 3. Both buttons are used to start the logging process and upload files to the server. When the log button is pressed, the logging process will be executed. Files created by the logging process can be found in the device directory as shown in Figure 4.

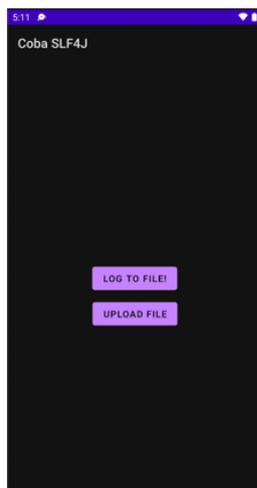
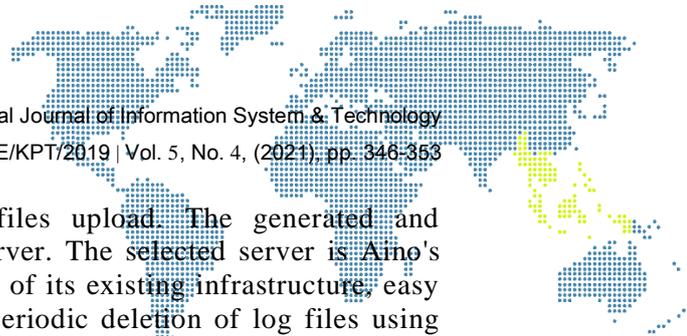


Figure 3. Logging feature simulation app interface



Figure 4. The files created by logging process in the device's directory



The next feature being developed is log files upload. The generated and compressed log files will be uploaded to the server. The selected server is Aino's own FTP server. This server was chosen because of its existing infrastructure, easy file management by merchant and device, and periodic deletion of log files using cron jobs.

The process of uploading files to an FTP server utilizes the third-party FTPclient library from Apache Commons. This library can handle the requirements to store and retrieve files from an FTP server. This library is implemented in the FTPClientHelper class which is later made into an object that can be used to perform the file upload process. This class executes the process of opening a connection as a communication between the server and the device, closing the connection, creating or changing directories on the server, and uploading files to the server.

The upload process can be used by accessing the methods in the upload feature implementation class. There are three methods in this class that can be called. The init method prepares the information about the server's storage directory, username, and password as well as opens a connection between the device and the server. The next method is uploadLog which is used to upload files. The first step of upload is to find the latest file that can be uploaded from the device's file directory. If there are files that can be uploaded, the file upload process is executed by calling the upload method of the FTPClientHelper instance. The last is the close method that closes the connection between the device and the server. This method is invoked when all file upload processes have been completed. Figure 5 shows the methods used in the implementation class of the upload process.

```

14 class LogUploadImpl(
15     private val client: FTPClientHelper
16 ) : LogUpload, CoroutineScope {
17     private val job = Job()
18     override val coroutineContext: CoroutineContext
19         get() = Dispatchers.IO + job
20
21     private lateinit var uploadDirectory: String
22     private lateinit var username: String
23     private lateinit var password: String
24
25
26     override fun init(uploadDirectory: String, username: String, password: String) {
27         this.uploadDirectory = "transaction_log/$uploadDirectory"
28         this.username = username
29         this.password = password
30         client.open(username, password)
31     }
32
33     override fun close() {
34         client.close()
35     }
36
37     override fun uploadLog(filePath: String) {
38         val logFile = getLatestFile(filePath)
39         if (logFile != null) {
40             launch { this: CoroutineScope
41                 val uploaded = client.uploadFile(logFile, uploadDirectory)
42                 if (!uploaded) {
43                     Log.e(tag: "Upload Process", msg: "Upload Failed")
44                 }
45             }
46         }
47     }

```

Figure 5. The upload implementation source code

The upload feature is tested by uploading a log file to an FTP server using the simulator app. The app uploads the log files to the DLPTTest FTP server. This server is public and is open to everyone. The servers provided are free of charge and the stored files will be deleted periodically. Testing is done by pressing the upload button on the app. It will upload a log file that was previously created during the logging process. Figure 6 shows the results of the upload process on the DLPTTest FTP server using FileZilla.

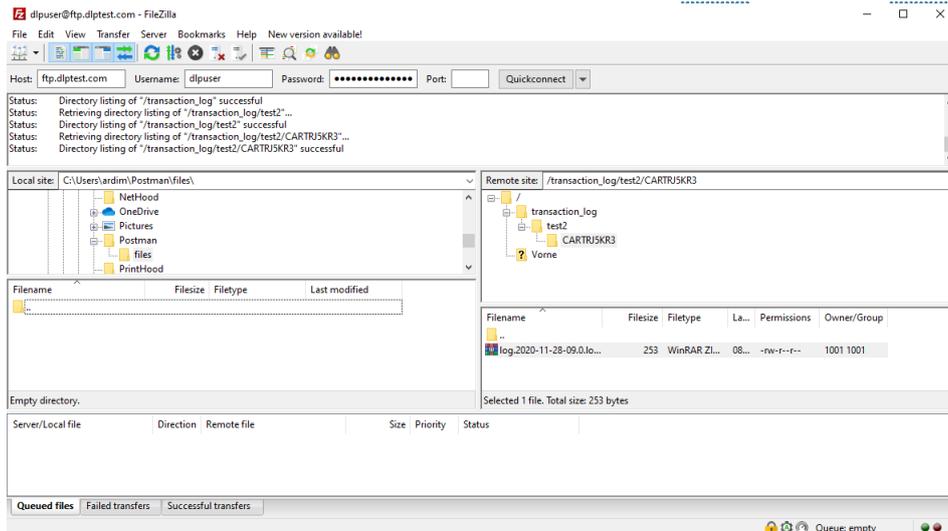


Figure 6. File uploaded by the simulation app on the DLPTTest server

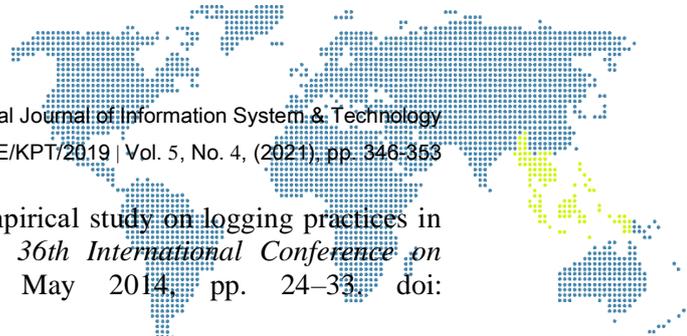
4. Conclusion

The logging feature in this SDK can record all information generated by the transaction process. This feature records the information generated by the process into a file stored on the device's storage media. The upload feature will upload the log file from the device to the provided FTP server. These two features will make it easier to track issues or problems that arise during the transaction process. The developers only need to look at the transaction data that has been recorded in the log files to find the source of the problem. With the upload feature, Aino does not need to send personnel to the field to retrieve data from the device. It can save costs and operational time. In addition, this SDK becomes a standard process that to various applications developed by Aino. So that the application development process can be done easier, faster, and does not waste a lot of resources.

The use of Scrum as a development method helps developers in carrying out the planned tasks. Scrum focuses developers to work on a particular task so that the development process remains focused and does not go out of a predetermined flow. Sudden changes can be implemented safely. It is possible because of transparency, adaptation, and inspection that are applied throughout the development process.

References

- [1] Bank für Internationalen Zahlungsausgleich, Ed., *Implications for central banks of the development of electronic money*. Basle, 1996.
- [2] A. Hidayat, "Working papers : Upaya meningkatkan penggunaan alat pembayaran non tunai melalui pengembangan E-Money," Jakarta, 2006.
- [3] Y. C. Lin, N.-H. Ha, and K.-S. Lin, "The Role of mPOS System in Process Change and Strategy Change: A Situated Change Perspective," p. 21, 2015.
- [4] S. Gupta, *Logging in Java with the JDK 1.4 logging API and Apache log4j*. Berkeley, Calif.: Apress, 2003. Accessed: Sep. 11, 2021. [Online]. Available: <http://www.books24x7.com/marc.asp?isbn=1590590996>
- [5] M. Davidekova and M. Gregu Ml, "Software Application Logging: Aspects to Consider by Implementing Knowledge Management," in *2016 2nd International Conference on Open and Big Data (OBD)*, Vienna, Aug. 2016, pp. 102–107. doi: 10.1109/OBD.2016.22.
- [6] D. Yuan *et al.*, "Be Conservative: Enhancing Failure Diagnosis with Proactive Logging," p. 14.



- [7] Q. Fu *et al.*, “Where do developers log? an empirical study on logging practices in industry,” in *Companion Proceedings of the 36th International Conference on Software Engineering*, Hyderabad India, May 2014, pp. 24–33. doi: 10.1145/2591062.2591175.
- [8] C. Gülcü, S. Pennec, and C. Harris, “Logback Project,” *Logback Project*. <http://logback.qos.ch/> (accessed Sep. 13, 2021).
- [9] J. Sutherland and K. Schwaber, “Nut, Bolts, and Origins of an Agile Framework,” p. 225.
- [10] K. Bhavsar, V. Shah, and S. Gopalan, “Scrum: An Agile Process Reengineering in Software Engineering,” *IJITEE*, vol. 9, no. 3, pp. 840–848, Jan. 2020, doi: 10.35940/ijitee.C8545.019320.
- [11] H. R. Suharno, N. Gunantara, and M. Sudarma, “Analisis Penerapan Metode Scrum Pada Sistem Informasi Manajemen Proyek Dalam Industri & Organisasi Digital,” *JTE*, vol. 19, no. 2, p. 203, Dec. 2020, doi: 10.24843/MITE.2020.v19i02.P12.
- [12] J. Rasmusson, *The Agile Samurai: How Agile Masters Deliver Great Software*, 1st ed. Pragmatic Bookshelf, 2010.
- [13] M. Cohn, *User Stories Applied: For Agile Software Development*. USA: Addison Wesley Longman Publishing Co., Inc., 2004.