Interconnection of Two Different Payment Systems

KEVIN AMMOURI

KANGYOU N CHO
Interconnection of Two Different Payment Systems

Kevin Ammouri and Kangyoun Cho

2019-06-25

Bachelor’s Thesis

Examiner
Gerald Q. Maguire Jr.

Academic adviser
Anders Västberg

Industrial adviser
Henrik Gradin, Centiglobe
Abstract

Mobile money, a means of transferring payments via mobile devices, has become increasingly popular. The demand for convenient financial products or services is a crucial factor in why innovative developers want to incorporate mobile money into existing financial products/services. The goal is to provide convenient financial services that enable customers to quickly send and receive money between two mobile payment platforms.

The Swedish blockchain company, Centiglobe, is searching for a system whereby payments can be made conveniently between two mobile payment platforms, specifically Alipay and M-PESA. This thesis sought to develop such a system by utilizing the application programming interfaces (APIs) (provided by Alipay and M-PESA) coupled with Centiglobe’s blockchain to facilitate payments between an Alipay user and an M-PESA user.

Solving this problem began with an initial literature study of previous work related to this topic and reading the extensive API documentation provided by Alipay and Daraja Safaricom (the developers of M-PESA). Next, a flowchart was created and used as a guide throughout the development of the system. Testing the system entailed integration testing. The performance of the system was determined by measuring the execution time to make a cross system payment.

A one-way transfer system was developed, as Alipay users can make a payment to M-PESA users but not the reverse. The results of the integration testing shows that the system is a feasible solution. The execution time of a payment shows that it is relatively quick (~9.1 seconds); thus the performance is adequate.

The conclusion is that this system is a viable solution for incorporating Alipay and M-PESA as mobile payment services. Moreover, the system partially facilitates person-to-person payments between them – subject to the limitations of the Alipay API. In addition, this system provides a foundation for other inter-platform mobile payment solutions.

Keywords
Alipay, M-PESA, API, Integration, Blockchain
Sammanfattning

Mobila pengar, ett sätt att överföra betalningar via mobila enheter, har blivit alltmer populära. Efterfrågan på praktiska finansiella produkter eller tjänster är en avgörande faktor för varför innovativa utvecklare vill integrera mobila pengar i befintliga finansiella produkter / tjänster. Målet är att tillhandahålla praktiska finansiella tjänster som gör det möjligt för kunder att snabbt skicka och ta emot pengar mellan två mobila betalningsplattformar.

Det svenska blockchainföretaget Centiglobe söker ett system där betalningar kan göras bekvämt mellan två mobila betalningsplattformar, särskilt Alipay och M-PESA. Denna avhandling försökte utveckla ett sådant system genom att använda applikationsprogrammeringsgränssnitt (API) (tillhandahållet av Alipay och M-PESA) i kombination med Centiglobe’s blockchain för att underlätta betalningar mellan en Alipay-användare och en M-PESA-användare.


Ett envägsöverföringssystem utvecklades, eftersom Alipay-användare kan göra en betalning till M-PESA-användare men inte tvärtom. Resultaten av integrationstestningen visar att systemet är en genomförbar lösning. Utbetalningstiden för en betalning visar att den är relativt snabb (~9.1 sekunder); därav en lagom prestanda.

Slutsatsen är att detta system är en lönsam lösning för att integrera Alipay och M-PESA som mobila betalningstjänster. Dessutom underlättar systemet delvis personliga betalningar mellan dem – med förbehåll för begränsningarna i Alipay API. Dessutom erbjuder detta system en grund för andra mobila betalningslösningar mellan plattformarna.

Nyckelord

Alipay, M-PESA, RESTful API, Integration, Blockchain
Acknowledgements

We would like to express our sincere gratitude to our examiner Prof. Gerald Q. Maguire Jr. for always being available and for replying instantly when we needed help with our thesis. His guidance, enthusiasm, knowledge, and his genuine support has helped us tremendously with the writing of this thesis. We could not have wished for any better examiner than Prof. Maguire.

Besides our examiner, we would also like to express our gratitude to Henrik Gradin, founder of Centiglobe, for providing us with the task of interconnecting Alipay and M-PESA that this thesis revolves around. Thanks to him, we have gained insight into what blockchain technology entails and we have learned APIs can be utilized to develop innovative products and services. In addition, we would like to thank him for investing his time in supporting and helping us throughout the project.

Stockholm, June 2019
Kevin Ammouri & Daniel Kangyoun Cho
# Table of contents

Abstract ......................................................................................................................... i
Sammanfattning ............................................................................................................. iii
Acknowledgements ....................................................................................................... v
Table of contents .......................................................................................................... vii
List of Figures .............................................................................................................. ix
List of Tables .............................................................................................................. xi
List of acronyms and abbreviations .......................................................................... xiii

1 Introduction .............................................................................................................. 1
   1.1 Background ......................................................................................................... 2
   1.2 Problem ............................................................................................................ 2
   1.3 Purpose ............................................................................................................. 3
   1.4 Goal .................................................................................................................. 3
   1.5 Methodology ................................................................................................... 3
   1.6 Delimitations ................................................................................................... 3
   1.7 Structure of the thesis ...................................................................................... 3

2 Background .............................................................................................................. 5
   2.1 RESTful APIs .................................................................................................. 5
   2.2 Alipay ............................................................................................................. 5
       2.2.1 Sandbox Environment ........................................................................... 5
       2.2.2 Alipay Wallet ......................................................................................... 6
       2.2.3 Cross-border WAP Payment .................................................................. 7
       2.2.3.1 Payment request (via HTTP) ................................................................. 8
       2.2.3.2 SDK & Payment response ................................................................. 9
       2.2.3.3 Digital Signature .............................................................................. 9
   2.3 M-PESA ......................................................................................................... 9
       2.3.1 M-PESA Sandbox ................................................................................... 10
       2.3.2 HTTP Header ......................................................................................... 10
       2.3.3 B2C API Call ......................................................................................... 11
       2.3.4 C2B Simulate Transaction API ............................................................ 13
   2.4 Spring Framework ........................................................................................... 14
   2.5 Retrofit .......................................................................................................... 14
   2.6 Blockchain Technology ..................................................................................... 14
   2.7 Related work .................................................................................................... 15
       2.7.1 Achieving Peer-to-Peer with Blockchain ............................................. 15
       2.7.2 Previous Research on Integrating Mobile Money with M-PESA .......... 15
       2.7.3 Adyen Integrations .................................................................................. 16
       2.7.4 Alipay Payments with Stripe .............................................................. 17
       2.7.5 Allowing Canadian Merchants to Accept Alipay Payments ............... 18
   2.8 Summary ........................................................................................................... 18

3 Methods .................................................................................................................... 19
   3.1 General Approach .......................................................................................... 19
   3.2 Literature Study .............................................................................................. 19
   3.3 Engineering Methodology .............................................................................. 20
   3.4 Testing .............................................................................................................. 20
       3.4.1 Test Bed ................................................................................................. 20
       3.4.2 Integration testing and Performance ...................................................... 21
# Table of contents

4 Integration of Alipay and M-PESA ........................................ 23
   4.1 Main Components .................................................. 23
   4.2 Data Objects ..................................................... 23
      4.2.1 Pay Out .................................................. 24
      4.2.2 Pay In .................................................. 24
      4.2.3 Status Code ............................................. 24
   4.3 Making a Payment with Alipay .................................... 24
      4.3.1 Gateway Interface ......................................... 25
      4.3.2 Payment Request ......................................... 25
      4.3.3 Verifying the Transaction ............................... 27
   4.4 Receiving a payment within M-PESA ............................... 27
      4.4.1 Gateway Interface ......................................... 27
      4.4.2 Payment Request ......................................... 29
         4.4.2.1 Authentication request ............................. 29
         4.4.2.2 B2C request ....................................... 29
      4.4.3 Verifying the transaction ............................... 30
   4.5 Registering Payments with the Blockchain ....................... 30
   4.6 Full Integration Process ........................................ 32
5 Results and Analysis .................................................. 35
   5.1 Major results .................................................... 35
   5.2 Performance Analysis ........................................... 35
   5.3 Validity Analysis ................................................ 35
   5.4 Discussion ....................................................... 36
6 Conclusions and Future Work .......................................... 39
   6.1 Conclusions ...................................................... 39
   6.2 Limitations ...................................................... 39
   6.3 Future work ...................................................... 40
   6.4 Reflections on sustainability and ethical aspects of this work . 40
References .................................................................. 43
Appendix A: Detailed results .......................................... 47
List of Figures

Figure 1-1: Estimated number of mobile phone users world-wide 2015-2020 (the source for this data was https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/) ................................................................. 2
Figure 2-1: Merchant account details from the Alipay Sandbox .............................................. 6
Figure 2-2: Buyer account for the Alipay Wallet ................................................................. 6
Figure 2-3: Payment flow of the Cross-border WAP Payment procedure with an application ........................................................................................................................................... 7
Figure 2-4: Creation of M-PESA Sandbox app. ....................................................................... 10
Figure 2-5: HTTP Header Parameters for calling M-PESA’s APIs ........................................ 10
Figure 2-6: A HTTP POST request to M-PESA B2C endpoint with M-PESA test credentials ........................................................................................................................................... 11
Figure 2-7: HTTP response from M-PESA B2C endpoint ...................................................... 12
Figure 2-8: Result parameters are shown in web callback when B2C API call was completed successfully ........................................................................................................................................... 12
Figure 2-9: A correct HTTP POST request to M-PESA C2B endpoint with M-PESA test credentials ........................................................................................................................................... 13
Figure 2-10: HTTP response from M-PESA C2B endpoint ................................................... 13
Figure 2-11: Result parameters are shown in web callback when C2B API call was completed successfully ........................................................................................................................................... 14
Figure 2-13: A screenshot of an example of a Source object [44] .............................................. 17
Figure 4-1: Alipay Spring Gateway Interface inside the Application using Retrofit ............... 25
Figure 4-2: Transaction begins, Application calls Start Pay In to the Alipay Spring Gateway ........................................................................................................................................... 25
Figure 4-3: Executing PayTask method from Alipay SDK within the application .................. 26
Figure 4-4: Check if the transaction is completed and registered ........................................ 27
Figure 4-5: M-PESA Gateway Interface inside the Application using Retrofit .................... 28
Figure 4-6: Start Pay Out and Complete Pay Out procedures ............................................. 28
Figure 4-7: M-PESA API interface for a B2C request call within the M-PESA gateway using Retrofit ........................................................................................................................................... 30
Figure 4-8: M-PESA gateway initiating Authentication request and B2C payment request to M-PESA (steps A-E) ......................................................................................................................... 30
Figure 4-9: Registering a transaction with the blockchain ...................................................... 31
Figure 4-10: Checking with blockchain if funds are available ............................................. 31
Figure 4-11: Complete One-way Integration with step indication, explanation of each step is covered in Table 4-5 ......................................................................................................................... 32
Figure 5-1: User interaction with the application, numbered from 1 to 6 ................................. 38
Figure A-1: Execution time results ........................................................................................ 48
## List of Tables

<table>
<thead>
<tr>
<th>Table No.</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2-1</td>
<td>Parameters used in the API Cross-border WAP Payment provided by Alipay</td>
<td>8</td>
</tr>
<tr>
<td>Table 3-1</td>
<td>Software tools</td>
<td>20</td>
</tr>
<tr>
<td>Table 4-1</td>
<td>The main components of the integration process</td>
<td>23</td>
</tr>
<tr>
<td>Table 4-2</td>
<td>Different possible status codes for a payment object</td>
<td>24</td>
</tr>
<tr>
<td>Table 4-3</td>
<td>Important key value pairs used to conduct a transaction with PayTask</td>
<td>26</td>
</tr>
<tr>
<td>Table 4-4</td>
<td>Important key value pairs used to conduct a B2C request with M-PESA</td>
<td>29</td>
</tr>
<tr>
<td>Table 4-5</td>
<td>Explanation of each step for the Complete One-way Integration shown in Figure 4-1</td>
<td>33</td>
</tr>
<tr>
<td>Table 5-1</td>
<td>Median time taken for different procedures within the system</td>
<td>35</td>
</tr>
<tr>
<td>Table A-1</td>
<td>Execution time in milliseconds (ms) for each procedure during testing</td>
<td>47</td>
</tr>
</tbody>
</table>
# List of acronyms and abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>B2B</td>
<td>Business-to-Business</td>
</tr>
<tr>
<td>B2C</td>
<td>Business-to-Customer</td>
</tr>
<tr>
<td>C2B</td>
<td>Customer-to-Business</td>
</tr>
<tr>
<td>CNY</td>
<td>Chinese Yuan</td>
</tr>
<tr>
<td>ICT</td>
<td>Information and Communication Technology</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>KES</td>
<td>Kenyan Shilling</td>
</tr>
<tr>
<td>MSISDN</td>
<td>Mobile phone number registered/subscribed in a GSM or UMTS mobile network, “Mobile Station International Subscriber Directory Number”</td>
</tr>
<tr>
<td>P2P</td>
<td>peer-to-peer</td>
</tr>
<tr>
<td>REST</td>
<td>for Representational State Transfer</td>
</tr>
<tr>
<td>WAP</td>
<td>Wireless Application Protocol</td>
</tr>
</tbody>
</table>
1 Introduction

The way society conducts businesses has changed drastically during the past decade. Information Technology (IT) is a major contributor to technological advancements and these advancements have made it possible for customers to communicate and interact with banks through mobile banking, thus quickly becoming the dominant payment method. This is correlated with mobile phone penetration (shown in Error! Reference source not found. Error! Reference source not found.). The figure shows that the number of mobile phone users is currently (in 2019) estimated to be 4.68 billion. The years preceding 2019 showed a continuous increase in mobile phone usage and this trend is expected to continue in the upcoming years. Therefore, third-party digital payment platforms, such as Alipay and Safaricom’s M-PESA, have an opportunity for use in e-commerce, thus explaining their immense popularity in their respective operating areas* and why they have quickly become dominant payment methods as they facilitate various valuable and convenient financial services to customers. More specifically, M-PESA facilitates services such as person-to-person transfers, receiving mobile phone credits, paying electricity bills, paying school fees, and even saving money when using M-PESA [1]. Alipay offers a full range of payment options including peer-to-peer (P2P) payments. Additionally, these digital platforms facilitate enhanced e-commerce functions and offer a smooth personalized experience [2].

In general, interactions between customers and banks through third-party digital platforms has revolutionized the banking sector and has resulted in increasing numbers of users utilizing payment services via a mobile application. This is described as “Open Banking”. Open Banking has had a significant impact on shaping the way we perform transactions, trade commodities, purchase services or goods, and more. As mentioned in [3] and [4], Open Banking is viewed as a model in which an application programming interface (API) allows you to share banking data with third parties. More specifically, Open APIs provide an interface for a developer [3] that enables the development of innovative applications and services and thus further enhances the capabilities of a financial service in order to create a smooth and tailored customer experience.

* Alipay’s operating area is China, and M-PESA’s operating areas are: Kenya, Tanzania, India, and elsewhere.
1.1 Background

Alipay and M-PESA provide various convenient banking services to customers. However, these digital platforms are limited when it comes to cross-platform services. According to Fumnanya Agbugah - Ezeana [5], M-PESA users will now be able to use M-PESA for payments on AliExpress, an online retail service owned by Alibaba [6]. Nevertheless, M-PESA users cannot engage in person-to-person transactions - or any other banking service with Alipay users and vice versa. Fortunately, Alipay and M-Pesa both provide open APIs, based on REST (more on this in Section 2.1) for developers to use freely. Furthermore, Alipay and Safaricom provide sandbox environments where you can use these APIs and safely run tests.

These APIs play a pivotal role in this thesis project as they enabled us to develop a service that enables Alipay users to directly interact with M-PESA users, thus facilitating payments/transactions between them. There is detailed documentation for these APIs and this will be of help when developing such a services. More generally, utilizing these APIs enables third-party digital payment platforms to enhance their services.

This thesis is based on a task given by Centiglobe, a Swedish blockchain company, that strives for global trading by interconnecting different financial systems [7]. Centiglobe gave us the task to utilize the APIs provided by M-PESA and Alipay in order to integrate these mobile payment platforms into a cross-platform mobile payment system. Centiglobe wanted us to achieve this by using their own blockchain to decentralize the system and by using M-PESA's and Alipay's APIs to integrate their services. The resulting system should facilitate transactions between Alipay and M-PESA users.

1.2 Problem

M-PESA and Alipay users are limited to their respective mobile system platform. However, since M-PESA and Alipay both provide several Open Banking APIs to enable further development of these platforms, there is potential to develop a solution that integrates these two different payment services.
With that being said, is it possible to develop a system that interconnects Alipay and M-PESA, thus facilitating payments between these two mobile payment services?

1.3 Purpose

The purpose of this thesis project was to find a solution that incorporates Alipay and M-PESA as mobile payment services, thus enabling person-to-person payments between Alipay and M-PESA users. In addition, this thesis project is meant to be a basis for reasearchers/developers who are working on similar topics.

1.4 Goal

The goal of this project is to develop an interconnecting system that enables person-to-person payments between Alipay and M-PESA users. If possible, then performance tests of the system will be conducted.

1.5 Methodology

An overview of the methodology used in this degree project is presented in this section. A more detailed explanation of the different methodologies used is given in Chapter 3.

To make accurate decisions on how to integrate the two systems, qualitative research was required on each of the payment services. This qualitative research included finding information about the two mobile payment services, their APIs, and in what context to apply each API of the two different systems. This research resulted in discovering what type of programming environment was required, which programming languages were needed, and which software tools would be most beneficial. Alipay and M-PESA both provide documentations, in which there is detailed information on how to utilize their APIs as well as presenting payment flows and user interactions. Understanding these documentations was essential for this thesis project.

1.6 Delimitations

This thesis will not focus on user interface design nor any other factors related to user experience*. Furthermore, it is important to note that when testing the different APIs and when conducting payments from one system to another, only fake virtual currency will be used. Therefore, we do not need to consider any legal aspects that could limit our research. In addition, our project is independent of other projects at Centiglobe that are developing or have developed an integration process between other payment services†, i.e. the outcome of our project will only be an interconnection of payment system between Alipay and M-PESA. Furthermore, this project will not focus on optimizing the system after it has been developed.

1.7 Structure of the thesis

Chapter 2 presents additional background information regarding Alipay, M-PESA, and relevant aspects of the integration process. Chapter 3 will present a more detailed explanation of the different methodologies and methods used in this thesis. Chapter 4 describes our implementation of the system, and Chapter 5 presents our results and discusses the results from an analytical point of view. Finally, Chapter 6 presents our conclusions and our suggestions for future work.

---

* This is addressed in other research such as “E-wallet – designed for usability” by Bercis Arslan and Blenda Fröjdih [8]
† Such as “Connecting Electronic Wallets to Achieve Cross-Platform P2P Micropayments” by Daniel Muresu and Ahmad Shabeeb [9]
2 Background

As both Alipay and M-PESA’s APIs are built on REST, this chapter provides basic background information about REST-based APIs. Additionally, crucial information and facts that need to be known to understand the integration of Alipay and M-PESA will be provided. This includes, but is not limited to, information regarding Centiglobe’s blockchain and Spring Framework. The chapter also describes related work.

2.1 RESTful APIs

Representational State Transfer (REST) is an architectural style that is built upon various constraints such as stateless communication, a layered system, client-server, … [10]. For instance, these various constraints make it possible for different systems on the Web to easily communicate with each other and provide the benefit of separating the client-side and server-sides. This ultimately means that the client and server can be modified and evolve independently without affecting each other [10]. Consequently, the client’s requests must contain all the data necessary for the server to properly process the request.

Safaricom describes in [11], that their M-PESA API endpoints regarding business-to-business (B2B), business-to-customer (B2C), and customer-to-business (C2B) are built on a HTTP-based REST architecture. Additionally, the API request parameters and responses are all formatted in JSON. A client invoking the API endpoints needs to ensure that all API requests contain the required parameters for the client-server interaction to succeed. Furthermore, when using these APIs to access data, HTTP verbs are used since data are represented as HTTP resources. The HTTP verb POST sends data to the server and is the most common verb when working with M-PESA’s API endpoints. Safaricom also states that REST clients (such as Postman [12]) and command line tools (such as Node.js [13]) can be used to invoke their API endpoints. Alipay’s APIs are also HTTP-based RESTful APIs.

2.2 Alipay

This section describes Alipay’s sandbox environment and the Alipay wallet features. Furthermore, this section describes the process of utilizing Alipay’s API.

2.2.1 Sandbox Environment

Alipay provides a full suite of tools and functions in order to assist external developers in how to use their product. One of these tools is their sandbox environment. By definition, a sandbox environment is “[…] a type of software testing environment that enables the isolated execution of software or programs for independent evaluation, monitoring or testing” [14]. This test environment was essential to gain an understanding of how the different APIs within Alipay work and how one should apply them in production.

The Alipay sandbox provides an account for each specific service depending on what type of API is used. The account that will be used is associated with Cross-border WAP Payment (further details are given in Section 2.2.3). This account will be the merchant account that will receive funds from the transactions. The account has an ID that starts with 2088 (indicating “Alipay standard”), an account name in the form of an email address, and other information as shown in Figure 2-1. In addition, the sandbox provides a buyer test account. Similar to the merchant account, the buyer account (shown in Figure 2-2) has an ID, email address, login password to Alipay Wallet Payment (further details in Section 2.2.2), and a payment password for making payments. It is important to note that all transactions within the sandbox environment utilize only fake virtual currency.
2.2.2 Alipay Wallet

There are three ways to use Alipay wallet to make payments. These alternatives are:

- **QR-code scanning**
  By scanning a QR-Code the buyer (the user) will be able to see the amount to be paid and other metadata about the transaction [15].

- **Built-in barcode**
  According to Alipay barcode documentation [16], the Alipay wallet assigns a randomly generated number and generates a barcode image of this number. This image could be scanned by the merchant or the merchant could enter the number manually. Once the barcode is processed by the merchant, the buyer to which this randomly generated barcode belongs will be prompted via an interface that displays the same information as QR code scanning provides.
Invoking Alipay Wallet

According to Alipay WAP documentation [17], the interaction between the merchant and the buyer is through an application and not a physical interaction. The application managed by the merchant company invokes the Alipay wallet. Subsequently, the buyer could agree to a given payment by entering their password for the Alipay Wallet.

The first two alternatives require some type of physical interaction between the merchant and the buyer; hence, the most appropriate option to look at is invoking the Alipay Wallet to conduct payments. Therefore, in the following sections, the focus is on invoking the Alipay Wallet. The API documentation regarding this alternative is in the Alipay Developer Guide [18]. This process will be explained further in the next section.

2.2.3 Cross-border WAP Payment

A Cross-border Wireless Application Protocol (WAP) Payment enables the user to make transactions either through an application or a mobile website provided by the merchant. The steps to perform a transaction are described in [19]. These steps are also presented below and illustrated in the form of a flowchart in Figure 2-3.

1. The customer issues a transaction and decides to pay with Alipay.
2. The merchant application sends a transaction request to Alipay.
3. Alipay responds with a payment link that either invokes the Alipay Wallet or the mobile website.
4. The application redirects the customer to the Alipay Wallet or the mobile website with the help of the received payment link.
5. The user will be prompted by the Alipay Wallet or the mobile website to enter their password to complete the transaction.

Figure 2-3: Payment flow of the Cross-border WAP Payment procedure with an application (the source for this image was https://gw.alipayobjects.com/os/skylark-tools/public/files/adb468781c83ae74f961d218852baec.jpg?26originHeight%3D1000%26originWidth%3D1000%26size%3D240051%26status%3Ddone%26width%3D500 from [19])
2.2.3.1 Payment request (via HTTP)

In order to invoke the Cross-border WAP Payment API, one must send a payment request which provides the required parameters to the Alipay gateway. The Cross-border WAP Payment’s parameters are presented in [20] and summarized in Table 2-1. The documentation notes that these parameters are key-value pairs [20]. In addition, the Alipay gateway for the test environment is https://mapi.alipaydev.com/gateway.do.

The base URL is the gateway and the parameters are appended to the gateway URL in the form: key1=value1&key2=value2&..&keyN=valueN. Therefore, a complete URL would be similar to the following: https://mapi.alipaydev.com/gateway.do?key1=value1&key2=value2&..&keyN=valueN.

Table 2-1 presents the most important parameters, explaining their purposes and the possible values that these parameters have. The full list of parameters for Cross-border WAP Payment can be found in [20].

Table 2-1: Parameters used in the API Cross-border WAP Payment provided by Alipay

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>service</td>
<td>To inform Alipay what type of service you are initiating so the gateway server understands how to handle the parameters and their values. For the Cross-border WAP Payment, the service is called create_forex_trade [20].</td>
<td>create_forex_trade</td>
</tr>
<tr>
<td>partner</td>
<td>The merchant account ID as mentioned in Section 2.2.1. This merchant account will be the merchant receiving the funds.</td>
<td>2088xxxxxxxxxxxxxxx</td>
</tr>
<tr>
<td>notify_url</td>
<td>The URL that will receive an asynchronous notification once the payment is complete.</td>
<td>Webservice for managing notification received, and verification of the payment</td>
</tr>
<tr>
<td>out_trade_no</td>
<td>A unique ID associated with the current transaction, this ID is generated/selected by the merchant.</td>
<td>Long Integer</td>
</tr>
<tr>
<td>total_fee</td>
<td>The total amount the buyer will pay.</td>
<td>Arbitrary number with two decimal places.</td>
</tr>
<tr>
<td>sign</td>
<td>To ensure a secure payment transaction, a signature has to be generated and the value has to be put together with the other parameters when initiating the request (more on this in Section 2.2.3.3).</td>
<td>Unique String</td>
</tr>
<tr>
<td>sign_type</td>
<td>The type of signing used (more on this in Section 2.2.3.3).</td>
<td>RSA or MD5</td>
</tr>
<tr>
<td>app_pay</td>
<td>If payment should be done through the merchant application or through a mobile website.</td>
<td>“Y” or “N” (String format)</td>
</tr>
</tbody>
</table>
2.2.3.2  **SDK & Payment response**

Alipay provides an SDK for a developer to use in order to integrate the Alipay services [21]. This SDK automatically opens the Alipay Wallet (if installed) or the mobile website for the user to see the amount, enter their password, and complete the transaction. This function in the SDK is called PayTask for Android and AlipaySDK for the iOS version [21].

According to the API’s document [21], the inputs for the function is the order information with the parameters defined in the following format: `key1=value1&key2=value2&...&keyN=valueN`. As mentioned previously in Section 2.2.3.1, the gateway is not included in the order information. The function sets the headers for the HTTP request and performs an HTTP POST request. Once the payment is executed, the function will return a string containing the result of the transaction. This result string contains the result code, a memo (often empty), and the order information.

2.2.3.3  **Digital Signature**

For Alipay to ensure security, each payment request needs a unique signature for the specific request. There are two different methods to choose from when signing a payment request, either MD5 or RSA [22].

If the type of signature is MD5, the developer can use the MD5 key provided by Alipay in the sandbox for a specific type of account. In this case, the account is the one associated with the Cross-border WAP Payment (see Section 2.2.3.1). In order to sign with MD5, the following steps need to be done [23, 24]:

1. Generate pre-string (pre-sign string), which has a format similar to the order information mentioned in Section 0. However, the parameters’ sign, sign_type, and other parameters with null or empty values are not included in the pre-string. The parameters must be ordered alphabetically inside the pre-string.
2. The merchant’s MD5 key is appended to the pre-string to create an assembled string.
3. The MD5 signature function takes in the assembled string and generates the signature.

In contrast, if the signature type is RSA, then the procedure becomes more complicated (but the security level is higher). The RSA private and public key pair can be generated using OpenSSL as explained in [22]. Once the necessary keys are generated, the signing process follows the steps below [23, 24]:

1. The same as the first step for MD5.
2. The RSA signature function takes in the pre-string and the private RSA key and generates the signature.
3. The signature is encoded using the Base64 scheme.

The final string generated in both the MD5’s step 3 and RSA’s step 3, are used as the value for the sign parameter seen in Table 2-1. If the sign is generated through MD5, the value of the sign_type parameter also seen in Table 2-1 should be set to MD5, otherwise if the sign is generated through RSA, the value of sign_type is set to RSA.

2.3  **M-PESA**

This section presents information regarding M-PESA’s sandbox environment and an overview of what it entails to make a B2C and C2B API call in the sandbox.
2.3.1 M-PESA Sandbox

M-PESA Sandbox provides a safe environment for the developer to access M-PESA services by the provided M-PESA API endpoints. The B2C and C2B API endpoints were crucial for the task given by Centiglobe because facilitating transactions between two different mobile money platforms requires B2C to C2B and vice versa. These API provide the system with the ability to transfer funds to and from M-PESA.

There are several steps that need to be done in order to invoke the API endpoints provided in the sandbox. As presented in [11], an app in the sandbox needs to be created and registered as an “MPesa Sandbox” product (see Figure 2-4). Doing so grants access to the B2C and C2B APIs. Furthermore, the app needs to be authenticated via the OAuth API that is provided by Safaricom. When successfully authenticated, an OAuth 2.0 access token [25] is returned from the API call. This token is subsequently used in the HTTP header of all API calls made in the sandbox environment. Once all initialization steps presented in [11] are completed and the access token is retrieved, one can begin invoking the APIs to access M-PESA services.

![Figure 2-4: Creation of M-PESA Sandbox app.](image)

2.3.2 HTTP Header

The M-PESA API endpoints require certain parameters to be included in the HTTP header when initiating an API call. As mentioned in the previous section, the OAuth 2.0 access token needs to be included in the HTTP header as a parameter and the content-type needs to be specified as “application/json” since M-PESA APIs only supports this specific content-type. A description of these parameters and their values is shown in Figure 2-5.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization</td>
<td>OAuth 2.0 Access Token. Bearer keyword followed by a space and the OAuth 2.0 Access Token. <strong>Bearer &lt;Access-Token&gt;</strong></td>
</tr>
<tr>
<td>Content-Type</td>
<td>Only <strong>application/json</strong> content type is supported.</td>
</tr>
</tbody>
</table>

![Figure 2-5: HTTP Header Parameters for calling M-PESA's APIs](image)
### 2.3.3 B2C API Call

As mentioned in Section 2.3.1, an HTTP header is needed when invoking an API but this alone does not suffice for a successful API call, as a client needs to include all of the parameters that the server requires for the client-server interaction to succeed. In this case, the server is the M-PESA service and the client is the developer invoking an API endpoint to access this specific M-PESA service; hence, the developer needs to provide a request body containing those specific request parameters that are required by the endpoint when using the HTTP POST method.

The request parameters for B2C in M-PESA's sandbox environment are generated by M-PESA's test credentials and can be used for testing when invoking the B2C API call (except for QueueTimeOutURL and ResultURL). In this API, the parameter “PartyA” is the node sending the money and PartyB is the node receiving the money. In this case PartyA is identified by the shortcode provided by M-PESA test credentials. “PartyB” is identified by the mobile subscriber integrated services digital network (MSISDN) number, i.e., a mobile subscriber's phone number. In this case the MSISDN is the Safaricom phone number provided in the M-PESA test credentials. In the request parameters “QueueTimeOutURL” and “ResultURL”, the developer needs to provide a webserver or callback server. These URL parameters are essential for receiving real-time information when a B2C API call completes. The data to be received from these servers contains result parameters (in JSON format) that were generated when the call to the API endpoint successfully completes. In addition, a response parameter is generated by M-PESA when an HTTP request has been made. Figure 2-6 illustrates the POST request with request parameters based on the test credentials. Figure 2-7 shows an example of the response parameters generated by M-PESA when the HTTP POST request has been made. Figure 2-8 is an example of the result parameters generated and shown in the server/URL provided in “ResultURL” parameter when the call to the API has successfully completed. A more detailed description of these request parameters, response parameters, and result parameters for the B2C API can be found in [26].

![Figure 2-6: A HTTP POST request to M-PESA B2C endpoint with M-PESA test credentials](image)

```
POST /mpesa/b2c/v1/paymentrequest HTTP/1.1
Accept: */*
Accept-Encoding: gzip
Accept-Language: en-GB
Authorization: Bearer kT0sz5z3FEKXAOtgTC3Diuizix
Content-Length: 676
Content-Type: application/json
Host: sandbox.safaricom.co.ke
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3798.102 Safari/537.36
X-Forwarded-For: 85.229.14.143
X-Forwarded-Port: 443
X-Forwarded-Proto: https

"InitiatorName": "testapi",
"SecurityCredential": "qGm5V5hFmaXlFf/+bx2fuOqTAWkh7Yq8hjrtMVCzEPFh6fd6UaVh2mNtT1PMcxWUWXKSEX1no2t",
"CommandID": "BusinessPayment",
"Amount": "100",
"PartyA": "669458",
"PartyB": "254708374149",
"Remarks": "id",
"QueueTimeOutURL": "http://id8c816d.ngrok.io/timeout",
"ResultURL": "http://id8c816d.ngrok.io/result",
"Occassion": "id"
```
HTTP/1.1 200 OK
Cache-Control: no-store
Connection: keep-alive
Content-Length: 254
Content-Type: application/json; charset=UTF-8
Date: Mon, 13 May 2019 12:40:43 GMT
Server: Apigee Router

{
  "ConversationID": "AG_20190513_00007b4f76961f562947",
  "OriginatorConversationID": "4076-1898169-2",
  "ResponseCode": "0",
  "ResponseDescription": "Accept the service request successfully."
}

**Figure 2-7:** HTTP response from M-PESA B2C endpoint

---

**Figure 2-8:** Result parameters are shown in web callback when B2C API call was completed successfully

---

---

---
2.3.4 C2B Simulate Transaction API

The C2B Simulate Transaction API lets a developer simulate a payment from client-to-business in the M-PESA sandbox. Before simulating a C2B transaction, the developer needs to register a validation URL and a confirmation URL (these are callback servers similar to those mentioned in Section 2.3.3 for M-PESA). These URLs are registered by invoking the C2B Register URL API; hence, two API calls are needed when working with the C2B Simulate API.

Once the callback servers are registered with M-PESA, one needs to provide the required request parameters in the HTTP POST request body when invoking the C2B Simulate Transaction API. This simulates a C2B transaction between a given MSISDN and a given shortcode. Unlike the B2C API, the shortcode is the node receiving the transaction and the MSISDN is the mobile phone number initiating the transaction. An HTTP POST request to the C2B API endpoint with parameters generated from the test credentials is shown in Figure 2-9. As was the case for the B2C API, the response parameters are returned when the POST request is made (see Figure 2-10) and the result parameters are shown in the web callback once the whole process has successfully completed (see Figure 2-11).

![Figure 2-9: A correct HTTP POST request to M-PESA C2B endpoint with M-PESA test credentials](image)

![Figure 2-10: HTTP response from M-PESA C2B endpoint.](image)
2.4 Spring Framework

The Spring Framework is defined as a "[...] Java Platform that provides comprehensive infrastructure support for developing Java applications" [27]. This framework enables the developer to focus on developing an application while avoiding any concerns about developing and managing the infrastructure. As stated in [27], the Java platform does not provide a developer with the functionality to organize basic building blocks to form a unified whole. Therefore, if a developer is working with several incompatible components and wants to create an application by composing these components, the developer must still find workarounds to fix this incompatibility. Fortunately, the Spring Framework has an implementation of Inversion of Control, also known as Dependency Injection, which provides a convenient way for a developer to compose these disparate components, thus achieving a fully working application.

2.5 Retrofit

In [28], retrofit is described as a type-safe HTTP client. Retrofit essentially takes REST endpoints and models them as Java interfaces. In other words, retrofit converts REST APIs to Java interfaces and these interfaces can be used to get and send JSON data via a RESTful web service. Retrofit is not a component of Spring Framework; although it can be used when working with REST endpoints. Retrofit is especially valuable when working with disparate components that consists of REST API calls.

2.6 Blockchain Technology

In the context of this thesis, a blockchain can be defined as: "[...] a decentralized transaction and data management technology [...]" [29]. Transaction systems are often characterized by having a centralized system in which transactions and data are controlled by third-party organizations (e.g. a bank) [29]. Blockchain technology enables a decentralized system; hence, no third parties control the transactions or the data.

Decentralization and persistency are two key characteristics of a blockchain. A combination of these two characteristics means that transactions can be validated quickly and invalid transactions are discovered immediately, thus preventing suspicious/corrupted transactions from occurring. Another key characteristic is anonymity. Blockchain technology enables anonymity by enabling users to communicate via the blockchain without revealing the identity of the user [30]. In addition, as stated in [29, 30], blockchain technology provides transparency to the users by providing publicly available information regarding completed transactions. This makes it easy for the users to verify and track their own transactions. This transparency is also another key characteristic of a blockchain.
2.7 Related work

This section presents various related work found as a part of the literature study that was conducted.

2.7.1 Achieving Peer-to-Peer with Blockchain

Chen and Xue [31] proposed a blockchain-based solution to enable a decentralized ecosystem for data exchange. According to them, current data exchange markets are run by third-party agencies, thus if there is some sort of a failure by the third-party, the whole system will fail. In addition, there are additional costs of third-party agencies and increased time costs in data exchange processes since all transactions need to be handled by the third-party. Therefore, they proposed a blockchain-based ecosystem to realize P2P data exchange, eliminating the need for third-parties. This means that extra costs for third-party agencies can be avoided and the corresponding time costs for exchanging data are potentially reduced. Furthermore, they state that using a blockchain-based ecosystem provides a convenient and reliable way for consumers and producers to audit their data since the blockchain automatically records transaction logs that are available for the data owners.

In their proposed ecosystem, all nodes in the ecosystem are meant to cooperate with each other to build a P2P network system or a market in which no third parties are involved. This is possible since a characteristic of blockchain stated in [31], is that it is Byzantine fault-tolerant, i.e. a blockchain tolerates a limited number of nodes that have arbitrary behavior (e.g. crashing, cheating, collusion, etc.). The main characteristic of their ecosystem is that their network is: “[…] operated by all participating nodes in an autonomous way” [31], similar to the Bitcoin network.

Their use of their blockchain, called the Data Exchange Network, is essentially used for the same reasons we are using blockchain technology in our system, i.e. to facilitate and provide a convenient transaction process without involving third-party agencies, thus avoiding additional costs (e.g., management fees, time costs, etc.) that comes with third-party agencies.

2.7.2 Previous Research on Integrating Mobile Money with M-PESA

Jake Kendall, et al. [32], a group of software developers and integrators specializing in the M-PESA platform, explain how the M-PESA platform can be harnessed as a platform to develop/produce new services. They state these services fall into two categories in which the first one is focused on enhancing M-PESA’s connections with financial institutes for delivering financial products, and the latter is focused on enhancing M-PESA’s ability to enable interconnection with other mobile and online payment systems.

According to them, in the first category there have been multiple technological companies striving to facilitate mobile money transfers between small financial institutions and M-PESA. However, they are struggling to integrate with M-PESA. For instance, Tangazoletu uses Spotcash, a mobile banking application, in which they are developing a set of tools to integrate M-PESA with financial institutes. This enables microfinance institutions to deposit and withdraw money to and from users’ savings accounts [32]. In addition, Kopo Kopo and CoreTEC Systems are providers that are developing/providing similar services to Tangazoletu. Moreover, Zege Technologies has developed an M-PESA integrated solution called M-PAYER. The co-founders of Zege Technologies, Kariuki Gathitu and Stella Njoki, are the brains behind M-PAYER, which is a mobile payment management system [34]. More specifically, M-PAYER is defined as: “a mobile and web application that enables

---

1 A Kenyan ICT firm that develops ICT products [33]
2 https://kokane.co.ke/
3 http://coretec.co.ke/index.html
4 An IT company consisting of developers and designers that is established in Nairobi [34]
businesses to manage income and expense and transactions of the business on both mobile money and cash” [35].

As they note, various mobile payment services such as iPay, Pesapal, Jambopay and more, have incorporated M-PESA in their integrated solutions. However, trying to integrate M-PESA with other mobile- and online payment systems comes with several challenges. One of the major challenges is the cost of M-PESA transactions which is quite high (more than US$1) and so are the M-PESA fees, especially the person-to-person fee which is over US$0.40. Therefore, it is challenging for the providers to integrate M-PESA with existing financial products/services even though an integrated system would lower costs (in both time and money) for the clients and benefit the service providers by offering low operational costs. Furthermore, it is stated that integrating with mobile money is a complex process due to M-PESA’s poorly functioning API, leading to high integration costs for financial institutes and resulting in poor performance systems.

They explain how M-PESA’s poorly designed API is a major barrier to achieving a good performance and effective integrated solution. M-PESA’s API is limited in a way that only certain functions are provided, hence, it is not a flexible API that a developer can work with. In addition, as mentioned previously in this section, the prices on M-PESA transactions are too high and thus another major barrier. Furthermore, the authors state that developing and maintaining customer relationships can be affected negatively since mobile money integration with financial institutes entails less face-to-face contact with clients.

2.7.3 Adyen Integrations

Adyen is a company that has developed an online payments solution with the intention of building a modern payment infrastructure directly connected to globally preferred payment methods and to Visa and Mastercard in order to achieve unified commerce [36]. Their solution dynamically provides local payment methods that are based on the shoppers’ transaction data”, hence facilitating global payments for shoppers through their Adyen platform. Adyen’s online payments documentation [37] presents extensive and detailed steps regarding how to integrate their solution for a website or a mobile app. Moreover, Adyen’s platform provides the possibility to integrate with Alipay in order to accept Alipay payments, i.e. when the shopper is selecting a payment method to pay with, Alipay can be selected as the payment method, thus prompting the Alipay application on the shopper’s phone. The Alipay API documentation [38] presents a thorough guide of how to integrate specifically with Alipay to accept Alipay payments. Alipay is one of the payment methods that has been incorporated in their solution. Moreover, it provides two delivery channels, ecommerce and point of sale [39]. In addition, by adding support for Alipay in the payments platform, Adyen has managed to attract Chinese shoppers as they can use payments methods they are familiar with and pay in their own currency (avoiding the need to pay exchange fees or being subject to exchange rates) [40].

Adyen have recently become partners with Cellulant” [42]. Cellulant handles 12% of Africa’s digital payments; thus enabling Adyen to gain access to: “ [...] 40 mobile money operators, over 600 local & international merchants and over 120 banks in Africa with just a single integration to Cellulant’s payment platform” [41]. As a result, Adyen can now offer its global payment solution to the African market. Not surprisingly, Safaricom’s M-PESA is one of the mobile money services that Adyen has access to, thus their solution enables merchants to access and use M-PESA as a local payment method to accept payments in Africa [41].

Adyen has managed to develop a successful interconnected payments system in which various mobile money operators have been incorporated thus facilitating global payments. Additionally,

---

* e.g. currency and geographic location (country code)

“Cellulant is a leading Pan-African technology company that organizes Africa’s marketplaces by connecting buyers, sellers, and other critical stakeholders with an underlying payments solution that enables them to make and receive payments.” [41]
Adyen provides well-defined and fully customizable API, thus facilitating integrations with Adyen’s functionalities.

2.7.4 Alipay Payments with Stripe

Stripe is another technology company that has built a payment infrastructure. They have developed a payments platform that provides the possibility to integrate with several mobile money operators, Alipay being one of them. Stripe’s API, also based on REST, provides a full suite of tools whereby Source objects is crucial for incorporating Alipay as a payment method. Source objects are defined as: “a single integration path for creating payments using any supported method” [43]. As stated by Stripe in [43], Stripe users will be able to use Sources in their back-end code to enable accepting payments from Chinese customers through Alipay. Figure 2-13 illustrates an example of a Source object that is used by the integration during a payment process. When a customer has selected Alipay as the payment method, the customer is redirected to Alipay for authorizing the payment. Once authorized, the integration uses the Source object to make a payment and complete it [43].

Interestingly, major businesses - such as Lyft, Slack, Wish, and more - are using Stripe for integrated solutions in which they have incorporated several payment methods for facilitating global payments and to provide convenient payment services to their customers [44].

![THE SOURCE OBJECT](image)

Figure 2-12: A screenshot of an example of a Source object [44].
2.7.5 Allowing Canadian Merchants to Accept Alipay Payments

SnapPay [45] is the official Canadian representative and Alibaba and WeChat’s official partner, thus offering several Alipay payment features when integrating their solution. The solution that they have developed is a plugin that enables Canadian merchants to accept payments from Chinese customers using Alipay and WeChat as payment methods. The settlement currencies are Canadian dollars or US dollars; hence Canadian merchants can choose to receive the payment in either of these currencies.

SnapPay also delivers an extensive and detailed API documentation [46] on how to integrate their solutions. Their API supports several payment scenarios: mobile commerce with Alipay, Alipay QR Code transactions, Alipay barcode payments, and more. Their API provides an adaptable and flexible API that can be utilized to enable Canadian merchants to accept Alipay payments [47].

2.8 Summary

This chapter presented information that is crucial for understanding how the M-PESA and Alipay APIs work. Additionally, information regarding Spring Framework, Retrofit, and blockchain technology has been presented to provide the reader with insights into how these tools can be utilized for developing our a system that interconnects Alipay and M-PESA.

This chapter introduced several related works that have developed an integrated payment system/infrastructure to facilitate payments between different mobile money operators/services or in order to enable certain payment methods to be used by the consumers when purchasing a service or a good. A common theme amongst these related works is that they provide fully adaptable and customizable APIs coupled with detailed API documentation for developers in order to enable developers to integrate these solutions. The related works that have been presented in this chapter are clear examples of successfully developed interconnected systems that facilitate global payments by incorporating local payment methods.
3 Methods

The purpose of this chapter is to provide an overview of the research method used in this thesis. Section 3.1 explains the general approach used to solving the problem. Section Error! Reference source not found. describes alternative methods that were considered. Section 3.2 presents the literature study that was conducted. Section 3.3 introduces the engineering methodology and Section 3.4 describes the tests that have been conducted in this thesis project.

3.1 General Approach

Throughout our studies, we have learned that modeling a problem and planning are two pivotal factors when solving a complex problem. Therefore, we created a detailed plan of how to effectively approach the problem. This plan made us realize that we needed to extensively model the problem by progressively building a flowchart (more on this in Chapter 4). In order to gather the facts and knowledge needed to create an initial flowchart, an extensive literature study was conducted. This literature study provided us with pertinent information regarding the various necessary components such as, but not limited to, REST APIs, building a RESTful web service, and how to utilize Spring Framework in our solution. This gave us insight into various corner cases that we might encounter and what issues we needed to address. In general, the literature study provided a solid foundation for this thesis.

We believe there are no alternative methods that are as effective as modeling the integration process with a flowchart as a flowchart provides a detailed description and highlights key aspects and core components of the process. Our integration process involved many steps, thus using a flowchart made it easy to follow the planned steps and generally made the work more intuitive. When working with APIs provided by services such as Alipay and M-PESA, the documentation often provides payment flows in the form of a flowchart. Therefore, the only logical and most plausible approach for our integration process was to model all the steps in a flowchart.

3.2 Literature Study

The literature study consisted mainly of reading documentations (specifically API documentations) provided by Alipay and M-PESA. These documents provided us with information regarding their payment flows, payment services, and more. In addition, the documents gave insight into the fundamentals of Alipay and M-PESA.

Alipay’s WAP documentation [19] and the documentations for the API [20] were essential for understanding how to conduct payments with Alipay through their merchant application.

The API documentation provided by Safaricom [11] was a crucial source for understanding the various M-PESA API endpoints and it provided with extensive information on how to invoke their APIs.

Square’s official documentation [28] was used to learn retrofit and to understand how we could use retrofit in our integration. In addition, Spring’s guide [48] was a pivotal source that contributed to our understanding of how the Spring Framework could be utilized to create payment gateways for Alipay and M-PESA.

Relevant articles were procured via databases such as IEEE Xplore Digital Library and Google Scholar. For instance, [29] and [30] provided general information about Blockchain technology. These articles were not directly related to the process of developing an interconnecting system between Alipay and M-PESA, but they provided us with information regarding the purpose of a blockchain and thus contributed to us understanding the purpose of Centiglobe’s blockchain.
3.3 Engineering Methodology

As mentioned in Section 3.1, we approached our problem by progressively building a flowchart. We knew that our final flowchart would involve several complex steps; therefore, we made sure to apply the divide-and-conquer paradigm, thus breaking up our problem into several subproblems and solving each of these subproblems. The resulting solutions are combined to provide a solution to the original problem.

One of these subproblems was to understand the payment flow of a transaction in Alipay and another was to understand the payment flow of a transaction in M-PESA. Another pivotal subproblem to solve was to figure out how the Spring Framework could be used to create payment gateways for both Alipay and M-PESA. By using the divide-and-conquer paradigm, we were able to easily analyze each step of the development process. Therefore, it became easy to solve each subproblem separately and eventually we combined the solutions and produced a detailed flowchart. Once we added all of the steps to the flowchart, it was easy to see the dependencies between various steps and how the integration process would work. The flowchart was created using draw.io*.

All of the code written during this project was written in Java. Three of the main software tools that aided us in working with the integration process are listed in Table 3-1.

<table>
<thead>
<tr>
<th>Table 3-1: Software tools</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Android Studio</strong></td>
</tr>
<tr>
<td><strong>IntelliJ with Spring Framework</strong></td>
</tr>
<tr>
<td><strong>Ngrok</strong></td>
</tr>
</tbody>
</table>

3.4 Testing

In order to ensure that the integrated system functioned properly, different tests had to be conducted. This section describes the specifications of the machine used to test the integrated system. This section also describes how the tests were conducted to ensure that the integration process was successfully implemented and that the integrated system was fully functional. The specific results of the tests and how they were validated are discussed and analyzed in Chapter 5.

3.4.1 Test Bed

All the tests were executed on the same machine with the software tools described in Table 3-1. The specifications of the machine are:

- **Operating System**: macOS Mojave version 10.14.5
- **Processor**: 2.3 GHz Intel Core i5 (2 cores)
- **Memory**: 8 GB 2133 MHz
- **Java Version**: OpenJDK 12.0.1 2019-04-16

* https://www.draw.io/
3.4.2 Integration testing and Performance

The testing consisted of integration testing and performance testing. The integration testing was to ensure that the money sent from one party was received on the party. There are several ways to ensure that a payment has been validated in both the Alipay service and the M-PESA service. These methods will be described and discussed in Chapter 5.

The idea of the performance testing was to ensure that the integrated system was effective and did what was expected to do within a reasonable time. We conducted 30 iterations of a payment and calculated a median time for each procedure within the integrated system to be executed. Hence, we can estimate the execution time for the application to complete the payment requested by the user of the application.
4 Integration of Alipay and M-PESA

In this chapter, we describe the implementation of the our system between Alipay and M-PESA. We found that it was possible to develop the system. However, only a one-way interconnecting system was possible due to Alipay not providing any B2C API. As a result, we were limited to using the C2B API. As one can only use the Alipay Wallet to pay for merchandise and services, there is no method for a business to pay an existing Alipay user using Alipay’s API. According to [50], it is stated that: “Alipay only allows Chinese citizens to send money through the app” [50]. Therefore, the implementation shown in this chapter only enables Alipay users to send money to an M-PESA user.

Section 4.1 introduces the main components in our implementation. Section 4.2 introduces the major data objects used and their purpose in the system flow. In Section 4.3, we will describe how we conduct a payment within Alipay. Section 4.4 describes the process of making a payment to an existing M-PESA user. Section 4.5 explains how the blockchain will be utilized. Finally, Section 4.6 will summarize the entire integration process.

4.1 Main Components

Table 4-1 describes the components used in the integration process.

<table>
<thead>
<tr>
<th>Type of component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>The client application that will initiate payment requests based on a client’s request.</td>
</tr>
<tr>
<td>Alipay Spring Server</td>
<td>A Spring server that executes different operations based on different requests from the application, managed by us.</td>
</tr>
<tr>
<td>Alipay</td>
<td>The default Alipay gateway, managed by Alipay.</td>
</tr>
<tr>
<td>M-PESA Spring Server</td>
<td>A Spring server that executes different operations based on different requests from the application, managed by us.</td>
</tr>
<tr>
<td>M-PESA</td>
<td>The default M-PESA gateway, managed by Safaricom.</td>
</tr>
<tr>
<td>The Blockchain</td>
<td>Centiglobe blockchain, managed by Centiglobe.</td>
</tr>
</tbody>
</table>

4.2 Data Objects

The major data objects is the Pay Out object, Pay In object, and the status codes that define the objects’ current status. The Pay Out object represents money (in our case, fake virtual money) going out of the system while the Pay In is money entering the system. In the implementation described in this thesis, the Pay Out object is for M-PESA and the Pay In object is for Alipay. This because we are transferring money from Alipay to M-PESA; hence, money is coming in to the system via the payment made by the user using the Alipay Wallet and money is going out of the system to be placed in the M-PESA account as requested by the user.
4.2.1 Pay Out

The Pay Out object contains the following information:

- The Pay Out ID, a unique ID generated by the M-PESA spring gateway to define the Pay Out object. The ID is generated through the Java library UUID*.
- The MSISDN, the phone number for the M-PESA user receiving the payment.
- The amount of money used in the transaction.
- The type of currency used, e.g. USD.
- The blockchain ID, a unique ID generated by the blockchain.
- The conversation ID, a unique ID generated by M-PESA to identify the transaction.
- The status of the transaction; could be either of the values described in Section 4.2.3.

4.2.2 Pay In

The Pay In object contains information such as:

- The Pay In ID, a unique ID generated by the Alipay spring gateway to define the Pay In object. The ID is generated through the java library UUID.
- The trade number, which is the “out_trade_no” mentioned in Table 2-1 is generated by the application itself, when a user decide to pay with the Alipay Wallet. The trade number is generated through the java library UUID.
- The amount of money used in the transaction.
- The type of currency used, e.g. USD.
- The blockchain ID, a unique ID generated by the blockchain.
- The status of the transaction; could be either of the values described in Section 4.2.3.

4.2.3 Status Code

The status code defines the status of a specific payment (Pay In or Pay Out) object. The different status codes are shown in Table 4-2.

<table>
<thead>
<tr>
<th>Status code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete</td>
<td>Transaction is completed.</td>
</tr>
<tr>
<td>Incomplete</td>
<td>Transaction is incomplete. (initial value)</td>
</tr>
<tr>
<td>Failed</td>
<td>Transaction failed.</td>
</tr>
<tr>
<td>Pending</td>
<td>Transaction is being processed.</td>
</tr>
</tbody>
</table>

4.3 Making a Payment with Alipay

In this section we describe the Gateway Interface used in the implementation of Alipay and how we keep track of transactions. We will also discuss how we used the SDK in order to perform a transaction. Finally, we will describe how we verify a transaction in order for it to be correctly registered with the blockchain.

* https://docs.oracle.com/javase/7/docs/api/java/util/UUID.html
4.3.1 Gateway Interface

When a user wishes to send money from an Alipay Wallet to a registered M-PESA user, the user has to first make a payment within Alipay. Once this request is initiated by the user, the application first notifies the Alipay Spring gateway that a transaction is about to occur. The Spring gateway expects an amount and the trade number and will then generate a Pay In object (see Figure 4-1). Subsequently, the Pay In object is stored in the gateway and then the gateway sends the Pay In object to the application. Both the Spring gateway and the application have access to the Pay In object. This procedure is called Start Pay In and is based upon a POST call to the Alipay Spring gateway (see Figure 4-1). It is important to note that the string defined after the POST annotation in Figure 4-1 is an extension to the base URL. The base URL is the URL for the Spring server (using ngrok). For an example of a base URL, see Table 4-3 under the parameter “notify_url”. A more detailed explanation of the Complete Pay In procedure defined in Figure 4-2 can be found in Section 0.

![Figure 4-1: Alipay Spring Gateway Interface inside the Application using Retrofit](image)

![Figure 4-2: Transaction begins, Application calls Start Pay In to the Alipay Spring Gateway](image)

4.3.2 Payment Request

We use the SDK provided by Alipay to assist the user to pay via their Alipay Wallet. We call the PayTask method from the Alipay SDK in the application; hence, the payment will occur via the application itself rather than through an external component. Furthermore, the PayTask method needs to be executed in a different thread and not in the UI thread (i.e., the main thread) due to executing longer operations and/or network operations within the Android Studio. Therefore, this must be in a separate thread so that the system (application) does not crash if something goes wrong with the call to PayTask.

Figure 4-3 shows the system flow of PayTask execution. Step A occurs when the application tells Alipay to initiate a payment with the requested parameters. The important parameters used in the
implementation and their respective values are shown in Table 4-3. Since the parameter “app_pay” is set to Y (meaning Yes), Alipay will invoke the Alipay Wallet, as happens in step B in Figure 4-3. Once Alipay Wallet generates a prompt, the user can enter his or her password to complete the transaction as shown in steps C and D. Once Alipay has processed the transaction, the results will be sent to the client application and to the Alipay gateway as an asynchronous notification - since our “notify_url” parameter was set to the Alipay spring gateway URL (as shown in Table 4-3). This is related to steps E and F in Figure 4-3.

The results sent back to the application are parsed and the result’s status parameter is extracted. If the status equals 9000, indicating that the transaction was successful [21], then we can finish the payment by calling the POST method Complete Pay In as shown in Figure 4-1 (more on this in Section 0). Otherwise we notify the user that the transaction was unsuccessful and abort the payment.

Figure 4-3: Executing PayTask method from Alipay SDK within the application

Table 4-3: Important key value pairs used to conduct a transaction with PayTask.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>partner</td>
<td>2088621924671008</td>
</tr>
<tr>
<td>notify_url</td>
<td>The Alipay Spring Gateway tunneled ngrok URL as the base URL followed with the extension /notify. Example: <a href="https://9ac07c78.ngrok.io/notify">https://9ac07c78.ngrok.io/notify</a> where the base URL is <a href="https://9ac07c78.ngrok.io">https://9ac07c78.ngrok.io</a></td>
</tr>
<tr>
<td>out_trade_no</td>
<td>The generated trade number within the application. Example: 3f08fc54-f323-45e3-ad8e-b4b20c8ea51c</td>
</tr>
<tr>
<td>total_fee</td>
<td>The amount the user requested to pay</td>
</tr>
<tr>
<td>currency</td>
<td>USD</td>
</tr>
<tr>
<td>app_pay</td>
<td>Y</td>
</tr>
</tbody>
</table>
4.3.3 Verifying the Transaction

Once the payment is made the application has to make a call to the Alipay Spring gateway via the procedure Complete Pay In (see Figure 4-1). The Complete Pay In procedure requires that the application sends the Pay In ID to the Alipay Spring gateway as shown in Figure 4-1 in order for the gateway to know which Pay In object needs to be verified and registered with the blockchain.

When the application sends the Pay In ID, the Alipay Spring gateway retrieves the Pay In object and checks if the status is complete, if not we wait (until we have a timeout) and try again, since the notification may not have been processed yet which will result in a failed transaction. However, if the status is complete, the Alipay spring gateway will register the transaction with the blockchain, more on this in Section 4.5. Once registered with the blockchain, the Alipay Spring gateway responds to the application’s request indicating the Complete Pay In procedure.

![Figure 4-4: Check if the transaction is completed and registered.](image)

4.4 Receiving a payment within M-PESA

In this section we describe the Gateway Interface used in the implementation for M-PESA and how we send and manage the Pay Out objects. We also explain how an M-PESA user to can receive the money sent from Alipay. Finally, we explain how we verify a transaction to ensure that what we are sending back to the application is accurate information. It is important to note that M-PESA’s part in this transaction will not be executed if the payment from Alipay was unsuccessful. Hence, payment will only occur once the Alipay payment is verified and registered with the blockchain.

4.4.1 Gateway Interface

The interface for the M-PESA Spring gateway is similar to that of the Alipay Spring gateway as shown in Figure 4-5 with a Start Pay Out procedure and a Complete Pay Out procedure. The Start Pay Out procedure is executed at the beginning of a payment request request by the user. The difference

---

1 The asynchronous notification sent from Alipay once payment is complete, seen in step F Figure 4-4, contains important results such as the trade status flag and the trade number flag. If the trade status flag shows that the trade is completed, we (Alipay spring gateway) retrieve the Pay In object that has the same trade number as the trade number flag and update its status to complete. It is important to note that this could happen before or after the application makes the Complete Pay In request.
between Start Pay In and Start Pay Out is that the Start Pay Out procedure needs a MSISDN rather than a trade number. The MSISDN is the destination for the M-PESA user who should receive the payment. The MSISDN is extracted by the application when a user of the application specifies which M-PESA user should receive the money. Similarly, the amount is extracted and since the input is USD we multiply the amount by the exchange rate (currently a factor of 100'). Once the application calls Start Pay Out, the M-PESA Spring gateway creates a Pay Out object, stores it inside the server (statically), and returns the Pay Out object - which is identical to what the Start Pay In procedure does. The Complete Pay Out method will be called once the application has confirmed that the Pay In object from the Complete Pay In call (shown in Figure 4-4) has its status flag set to Complete. Otherwise, the application will not proceed with the Complete Pay Out call – as the transaction has failed. The Complete Pay Out procedure makes it possible for the M-PESA Spring gateway to know when to execute a B2C request as will shown in Section 4.4.2. The call to Complete Pay Out requires a Pay Out ID in order for the gateway to understand the context of the transaction that is to be made, based upon extracting the Pay Out object and deciding on what status to set depending on the B2C request (more about this in Section 4.4.2). The Start Pay Out and Complete Pay Out procedures are shown in a flowchart in Figure 4-6.

```java
public interface MpesaGatewayAPI {
    @POST("/startpayout")
    @FormUrlEncoded
    Call<PayOut> startPayOut(@Field("amount") String amount, @Field("msisdn") String MSISDN);

    @POST("/completpayout")
    @FormUrlEncoded
    Call<PayOut> completePayOut(@Field("id") String payOutID);
}
```

Figure 4-5: M-PESA Gateway Interface inside the Application using Retrofit

Figure 4-6: Start Pay Out and Complete Pay Out procedures

*1 USD is approximately 100 KES (Kenyan shilling)*
4.4.2 Payment Request

The B2C payment request will be executed by the M-PESA Spring gateway rather than the application itself unlike Alipay’s PayTask. As previously mentioned, the M-PESA gateway will initiate the B2C request once the Complete Pay Out call has been made by the application. Therefore, this call can only occur if the payment for Alipay was successful. Once called, the M-PESA gateway extracts the Pay Out object with the help of the Pay Out ID sent by the application. The Pay Out object contains the blockchain ID for the transaction. The M-PESA gateway needs to ensure that the blockchain has registered the transaction before continuing with the authentication request from M-PESA (more on this in Section 4.4.3). When the blockchain has confirmed that a transaction can be made, then the M-PESA gateway will initiate the authentication request, shown as steps A and B in Figure 4-8, and then finish the B2C request shown in step C, D, and E in Figure 4-8. The base URL for the M-PESA API shown in Figure 4-7 is https://sandbox.safaricom.co.ke/ - as this is the M-PESA sandbox environment.

4.4.2.1 Authentication request

The authentication request correspond to the GET request shown in Figure 4-7. In order to receive the access token, we make a GET request to the M-PESA sandbox with a basic token in an authentication header. The basic token can be generated knowing a key and a secret and then encoding them with a Base64 encoder (provided by the Java library*). The key and the secret are received from M-PESA when an application in M-PESA is created, see Section 2.3.1. Once encoded and the basic token generated, we can initiate the GET request. When we make a call to the M-PESA sandbox we receive everything inside the AuthResult class (as specified in Figure 4-7). The AuthResult class contains an access token as well as an expiration time indicating when the access token is no longer valid (usually one hour). We then extract the access token from the AuthResult and send it as a header authorization in the B2C request to initiate a payment.

4.4.2.2 B2C request

The response parameters from M-PESA once the B2C request call has been initiated will be placed inside the PaymentResponse class which has the same entries as shown in Figure 2-7. The parameters used in conducting a B2C request can be found in Table 4-4. When a response has arrived we check the response code, if it is not equal to zero we know that the request was not successful and therefore we abort the payment and set the status of the Pay Out object to Failed and return it to the application. However, if it is zero we will map the Pay Out ID to the Conversation ID received and wait until the result parameters are handled (as will be discussed in Section 4.4.3). Once the result is processed and confirmed we send the object to the application; hence, completing the Complete Pay Out procedure.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>InitiatorName</td>
<td>testapi</td>
</tr>
<tr>
<td>Amount</td>
<td>The amount found inside the Pay Out object</td>
</tr>
<tr>
<td>PartyA</td>
<td>600458 (The shortcode for the account sending the money)</td>
</tr>
<tr>
<td>PartyB</td>
<td>MSISDN (The receiver)</td>
</tr>
<tr>
<td>ResultURL</td>
<td>M-PESA Spring Gateway tunneled ngrok URL as the base URL followed with the extension “/result”. Example: <a href="https://35a5a581.ngrok.io/result">https://35a5a581.ngrok.io/result</a></td>
</tr>
</tbody>
</table>

* https://docs.oracle.com/javase/8/docs/api/java/util/Base64.html
4.4.3 Verifying the transaction

Once the B2C request is finished, M-PESA sends a notification of the results of the transaction (step E in Figure 4-8). This notification has the same structure as shown in Figure 2-8. The destination of the notification is dependent on the ResultURL parameter (from Table 4-4), which is in our case is the M-PESA Spring gateway.

These results are parsed within the M-PESA Spring gateway, and the following keys & values are extracted: Result Type, Result Code, and the Conversation ID. The M-PESA Spring gateway first makes sure that the result code equals zero, if it does not, we will abort the payment. If it does equal zero, it continues by looking at result type that indicates if this notification was sent more than once. If the result type does not equal zero, the result notification is thrown away since it has already been processed. However, if this is the first time the notification is received, then the gateway extracts the Conversation ID and uses it to find the Pay Out ID that correlates to that Conversation ID. Finally, the Pay Out object is extracted and the status set to Complete.

4.5 Registering Payments with the Blockchain

Before the Alipay Spring gateway responds to the Complete Pay In called by the application it has to register the transaction via the blockchain (by invoking an API call) as shown in Figure 4-9. This procedure takes approximately 5000 milliseconds to execute [51]. Once the procedure is done the blockchain will return a blockchain ID which we will put into the Pay In object before responding to the Complete Pay In call.

When the M-PESA Spring gateway receives the Complete Pay Out call from the application it checks if the Pay Out object’s blockchain ID is confirmed as a successful registration within the blockchain. This is done through an API call to the blockchain, see Figure 4-10. This procedure is not as time consuming as the registration and takes approximately 100 milliseconds [51] to execute.
To simulate the time to execute these procedures, we emulate a Spring gateway that accepts two calls: Register and Check. The Register procedure sleeps for 5000 milliseconds and then respond with a unique ID which represents the blockchain ID. The Check procedure sleeps for 100 milliseconds and respond with either true or false. The reason why we simulated these procedures is because we do not have access to the actual blockchain API and the register procedure uses real money exchanges which is outside of the scope for this thesis.

Figure 4-9: Registering a transaction with the blockchain

Figure 4-10: Checking with blockchain if funds are available
4.6 Full Integration Process

The full integration process is illustrated in Figure 4-11 with different numbers of steps and is directly related to Table 4-5 that provides a description for each of these steps. For a more detail explanation of each step, see the previous subsections.
Table 4-5: Explanation of each step for the Complete One-way Integration shown in Figure 4-11

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Client wants to send money from Alipay to M-PESA</td>
<td>6a</td>
<td>Alipay gateway makes API call to blockchain to register transaction</td>
</tr>
<tr>
<td>1a</td>
<td>Start Pay Out (Section 4.4.1)</td>
<td>6b</td>
<td>Blockchain returns blockchain ID</td>
</tr>
<tr>
<td>1b</td>
<td>Returns Pay Out object</td>
<td>7</td>
<td>Alipay gateway responds to the Complete Pay In call with the updated payment object</td>
</tr>
<tr>
<td>2a</td>
<td>Start Pay In (Section 4.3.1)</td>
<td>8</td>
<td>Compete Pay Out (Section 4.4.1)</td>
</tr>
<tr>
<td>2b</td>
<td>Returns Pay In object</td>
<td>9a</td>
<td>M-PESA gateway checks with blockchain if funds are available for the sent blockchain ID</td>
</tr>
<tr>
<td>3a</td>
<td>Initiate PayTask (Section 4.3.2)</td>
<td>9b</td>
<td>Blockchain confirms that a payment can be made by the M-PESA gateway</td>
</tr>
<tr>
<td>3b</td>
<td>Alipay prompts Alipay Wallet</td>
<td>10a</td>
<td>M-PESA gateway requests Access Token from M-Pesa</td>
</tr>
<tr>
<td>3c</td>
<td>User enter password</td>
<td>10b</td>
<td>M-PESA respond with requested token</td>
</tr>
<tr>
<td>4a</td>
<td>Alipay Wallet signals transaction complete</td>
<td>11a</td>
<td>M-PESA gateway initiates a B2C request</td>
</tr>
<tr>
<td>4b</td>
<td>Alipay sends the result code</td>
<td>11b</td>
<td>M-PESA responds with transaction response status/details</td>
</tr>
<tr>
<td>4c</td>
<td>Asynchronous notification with more results regarding the transaction</td>
<td>11c</td>
<td>M-PESA sends a notification to the M-PESA gateway with the result status/details</td>
</tr>
<tr>
<td>5</td>
<td>Complete Pay In (Section 0)</td>
<td>12</td>
<td>M-PESA gateway responds to the Complete Pay In call with the updated payment object. (Transaction complete or failed depending on the status of that returned payment object)</td>
</tr>
</tbody>
</table>
5 Results and Analysis

In this chapter, our results are presented and analyzed. Furthermore, this section presents a discussion of our results and analysis.

5.1 Major results

We developed a one-way integration of Alipay and M-PESA. We are able to send fake virtual money from an Alipay user to an M-PESA user. The entire process of sending money from Alipay to M-PESA within the application is shown in Figure 5-1: User interaction with the application, numbered from 1 to 6.

The analysis of the performance and the validation of the system integration is given in the following sections.

5.2 Performance Analysis

Table 5-1 shows the median time the different procedures take when executing a transfer. The time to execute each procedure is calculated through various execution measurements. We tested our system, i.e. the procedures in the system, 30 times, sending US$10 (fake virtual money) each time. We measured how long each execution of each individual procedure took by using a Java system library to get the current time in milliseconds. We retrieved the time before executing the procedure and the time when the procedure finished. By subtracting these times, we calculated the time it took to execute the procedure in milliseconds. The individual test results can be found in Appendix A, Table A-1.

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Median time taken to execute the procedure (ms)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Pay In</td>
<td>372</td>
<td></td>
</tr>
<tr>
<td>Start Pay Out</td>
<td>522</td>
<td></td>
</tr>
<tr>
<td>PayTask</td>
<td>Cannot be calculated; depends on user’s interaction within Alipay Wallet and Alipay’s time to complete the payment made to Alipay.</td>
<td></td>
</tr>
<tr>
<td>Complete Pay In</td>
<td>5334.5</td>
<td>Register transaction with blockchain included</td>
</tr>
<tr>
<td>Complete Pay Out</td>
<td>3746.5</td>
<td>Check transaction with blockchain included</td>
</tr>
</tbody>
</table>

5.3 Validity Analysis

Before executing our tests, we logged all the Pay In objects and the Pay Out objects once each transaction was finished. Additionally, each trade number from the asynchronous notification sent by Alipay and each Conversation ID from the notification sent by M-PESA were also logged each time the Spring gateways received a notification from the respective service. When the testing was completed, we made sure that each trade number existed within a Pay In object as well as each Conversation ID existed within a Pay Out object. Since these numbers and IDs are unique, and because we succeeded in pairing all trade numbers and conversation IDs we know that all 30 executions were successful transactions. It is important to note that the logging was done in the background; hence, it did not interfere with the timing of the Spring gateways or the application.
To further validate our results we ensured that money sent from Alipay ended up with the M-PESA user. Since we did 30 executions and sent US $10 each time, the total amount sent was US$300.

For M-PESA, we checked the “B2CUtilityAccountAvailableFunds”, the business account within M-PESA that is paying the M-PESA user using the parameter shown in Figure 2-8 before and after the 30 executions. The value of the parameter had decreased by 30990. This is true since $300 \times 100 + 33 \times 30 = 30990$ KES. The factor of 100 is the exchange rate from USD to KES (see Section 4.4.1) and the extra 990 (33 x 30) KES is a per-transaction fee*.

For Alipay, we checked the balance of the buyer account within Alipay’s sandbox before and after the 30 executions. Before the executions, the opening balance was exactly 10000 CNY (see Error! Reference source not found.) and once all the executions were completed the balance was 8021.71 CNY. This is correct, because according to the conversion rate shown in picture 4 in Figure 5-1 we should have $300 \times 6.5943 = 1978.29$ CNY less after the 30 executions which is exactly what we ended up with since $10000 - 8021.71 = 1978.29$.

5.4 Discussion

As we noted in Table 5-1 the time to execute the PayTask function was not calculated. This is because, the time it takes to execute PayTask depends on how long the user interacts with the Alipay Wallet (see picture 4 in Figure 5-1: User interaction with the application, numbered from 1 to 6.

). This cannot be calculated and therefore is not included in the time it takes for the integrated system to complete a payment. This is also the case for the time it takes the Alipay Wallet to complete the transaction. We cannot compute how long Alipay takes to complete the payment within the Alipay Wallet since the entire execution time for the PayTask function consists of the user interaction time as well as the time for Alipay to complete the payment. Hence, we cannot divide the operations up and calculate Alipay’s time to complete each part of the payment process. Therefore, the total time for the integrated system to complete a payment is dependent on the time it takes for the sum of the different procedures to be executed.

Additionally, the Start Pay In and Start Pay Out procedures are executed in parallel when the user clicks on the “Pay with Alipay” button as shown in picture 3 in Figure 5-1: User interaction with the application, numbered from 1 to 6.

Once the payment with Alipay is done, we execute the rest of the procedures. Because of this, we can say with certainty that the time it takes for the application to finish the payment after the payment with Alipay is done takes \( \sim 9081 \) milliseconds (\( \sim 9.1 \) seconds), i.e., this is the time it takes for the Complete Pay In plus the time for the Complete Pay Out to finish.

Furthermore, when examining the different execution times in our testing results regarding the procedures Complete Pay In and Complete Pay Out, we noticed that the Complete Pay In procedure is more consistent in its execution time than the Complete Pay Out procedure. The reason behind this is because once the application calls the Complete Pay In procedure, the PayTask function has already been executed and therefore by the time the application has open up a new communication channel with the Alipay Spring gateway server, the asynchronous notification sent from Alipay has almost always already been processed by the gateway; hence, a quick response will be delivered. However, in the case of the Complete Pay Out procedure, since the M-PESA Spring gateway is conducting the B2C transaction it has to wait for the result notification to arrive in order to process it and then continue by responding to the application’s Complete Pay Out call. The time it takes for the result notification sent from M-PESA to arrive can vary depending on, but not limited to, the performance of M-PESA’s own servers.

* M-PESA charges additional 33 KES for every transaction in the sandbox environment.
The design choices we made when building the system may not have been optimal. For example, the Alipay Spring gateway is not doing a lot of work compared to the M-PESA Spring gateway. The Alipay gateway processes the asynchronous notification sent from Alipay once the payment through the PayTask function was completed. However, we do not have to process the notification since the PayTask itself (as was mentioned in Section 0) returns a result indicating the status of the transaction. If the result status equals 9000, the result code indicating that the transaction is completed, we can continue with registering the payment with the blockchain within the application as a background operation. This leads to fewer network operations from the application since it does not have to involve the Alipay gateway anymore and the Start Pay In and Complete Pay In procedures can be completely removed from the system.

Our two Spring gateways are highly dependent on the notifications received from the respective services. If we remove the Alipay gateway as mentioned previously we have solved half of the dependency problem; however, the M-PESA gateway has no option but to wait for the result notification. The reasoning for this is that the response code field (shown in Figure 2-7), can be 0 (successful), but the result code (received as a part of the result notification sent by M-PESA) does not have to be successful even though the response code was. The problem arises when the M-PESA Spring gateway is waiting for the notification, but the notification does not arrive - this will cause problems for the system and generate unexpected outcomes. To solve this, one could apply a timeout for the spring gateway, hence, once the result processing has taken too long to be executed we know that M-PESA failed to send the results and we issue a timeout for the transaction, which will result in a failed transaction.

A final point concerns storing data within the Alipay and M-PESA Spring gateways. Our current system store data statically which is ineffective in the long term usage of the integrated system. The data that needs to be stored are the transaction payment objects (Pay In and Pay Out). Each Pay In ID and Pay Out ID is mapped to its Pay In and Pay Out object respectively and each trade number and MSISDN is mapped to its Pay In and Pay Out ID respectively. These can be stored statically until the transaction within the integrated system has been completed but the objects, which describe the transactions, need to be stored in order to maintain a record of transfers made within the integrated system. The storage of this data needs to be effective because in the future this product may perform many transactions, hence the needs to be handled appropriately.

An important note regarding our system is that there is a distinct design flaw when making a payment. When an Alipay user initiates a payment, the user will enter an amount in USD. The same amount is then procured and used as a parameter when calling M-PESA's B2C API even though the currency is KES for M-PESA users. For example, in our system, if an Alipay user initiates a payment in the amount US$100, the M-PESA user will receive 100 KES instead of an amount equivalent to US$100. As mentioned in Section 5.3, we made a quick fix to resolve this issue, which entailed multiplying the amount entered by the Alipay user by the exchange rate from USD to KES. However, this should not be a fixed rate as exchange rates change over time. Consequently, the amounts sent from an Alipay user and the amount received by an M-PESA user in our system are not equivalent.
Figure 5-1: User interaction with the application, numbered from 1 to 6.
6 Conclusions and Future Work

This section presents the conclusions of our thesis and discusses what factors limited our efforts when conducting this thesis. Furthermore, this section presents our suggestions for future work. Finally, the chapter ends with some reflections on ethical and sustainability issues related to this thesis project.

6.1 Conclusions

From the performance tests, it is concluded that payments that are conducted in our system are relatively quick compared to bank wire transfers through financial institutes which can take minutes or within 1-3 banking days (depending on the financial institute). The same goes for international wire transfers. Although, the Centiglobe’s blockchain is a major contributor to the execution time as it takes five seconds to register a transaction (which is fairly slow). Nonetheless, the fact that we were able to conduct our performance tests coupled with the results implies that we managed to find a plausible solution to enable person-to-person payments between Alipay and M-PESA users. However, our system only enables one-way payments, i.e. only Alipay users can make a payment to an M-PESA user, not vice versa. As a consequence, the thesis has not fully achieved its goal, i.e. facilitating person-to-person payments for both Alipay users and M-PESA users. Therefore, there is much work to be done and there are several possible options for extending the capabilities of our system. We conclude that the most pivotal future work to be done is to enable M-PESA users to make payments to an Alipay user. Achieving this would make the system more viable for use when conducting person-to-person payments between these mobile payment services. The fact that we could not develop a two-way interconnecting system coupled with not being able to implement a refund functionality has resulted in our system being highly dependent on the M-PESA component. More specifically, if the M-PESA server would crash, the whole system would malfunction as the Alipay user has already made a payment and the blockchain has registered the transaction, while that the M-PESA user will not receive their payment due to the crash. This suggests that there needs to be a way to replay failed transactions when the M-PESA servers are again functional.

As one of the purposes of this thesis was to provide a basis for other reasearchers conducting similar research in this topic area, we conclude that the design of the system that has been developed in this project provides a foundation for other solutions.\footnote{E.g., using Cross-Border payments in Handelsbanken, the time to receive money from a payment varies depending on which Cross-Border payment you select. The time it takes to receive the money can be within the same day, within one banking day, or within 1-3 banking days \cite{52}.}

Lastly, we would like to conclude that as the solutions that has been covered in Section 2.7 provides great and flexible APIs (compared to Alipay’s and M-PESA’s APIs), we believe it will be more convenient and suitable to use the APIs provided by the microfinance services (e.g. Adyen or Stripe) rather than using the APIs provided directly by Alipay and M-PESA when developing a solution that incorporates various mobile payment platforms. We have encountered too many issues when working with Alipay’s and M-PESA’s APIs which is also why we conclude that one should avoid these APIs.

6.2 Limitations

Due to poorly designed and poorly functional M-PESA APIs, we were limited in what API calls we could use to develop the system. As previously mentioned in Section 2.7.2, M-PESA API is known to be a major barrier for achieving a good performing integrated system as it only provides certain limited functions. We experienced this personally as we needed to find workarounds to harness the inflexible API provided by M-PESA to achieve our goal. This made the development of a system interconnecting Alipay and M-PESA more complex.
Furthermore, when modelling the system, we noticed that Alipay did not provide any B2C-related API calls, thus only C2B could be used. In contrast, M-PESA provided both C2B and B2C APIs, hence we use B2C API on M-PESA’s side and C2B API on Alipay’s side to enable person-to-person payments. This limited our work because we could only develop a one-way system, i.e. only Alipay users can initiate a payment, hence M-PESA users can only accept payments from Alipay users.

Additionally, we struggled to test our system due to the PayTask function (within the Alipay SDK) not functioning properly. When calling this function, we kept receiving an error saying “Server is currently busy” without any error codes or error descriptions. Therefore, we could not determine what caused this issue. Although, we checked to see if this issue was caused by our implementation, by troubleshooting the issue through the demo app (that uses PayTask) provided by Alipay. The issue persisted even in their demo app; hence, we concluded that the issue is most likely caused by Alipay’s server. Consequently, when conducting tests of our system, we needed to repeatedly call the PayTask function until it eventually worked. Unfortunately, there were several occasions when we could not bypass the error message, thus halting our work related to Alipay.

Due to the limited duration of this project, we could not incorporate a refund functionality in our system. As M-PESA’s Reversal API does not work, it would have been difficult or perhaps even impossible to implement a refund feature on M-PESA’s side. In contrast, Alipay provides a refund API; therefore, in theory, it should be possible to implement a refund feature on Alipay’s side. However, we have not yet tested this API.

6.3 Future work

One obvious future work that can and should be done is to extend the capabilities of our system by enabling M-PESA users to make payments to Alipay users, thus enabling bi-directional transfers. The related works that were presented in Section 2.7 are great examples of systems in which the nodes in the system can both send and receive money, regardless of the mobile money operator. As presented previously in Section 2.3.4, M-PESA provides a C2B API, hence there exists an API for M-PESA users to initiate a payment. The problem lies within Alipay as mentioned in the previous section. Therefore, if Alipay would provide an API that enables B2C transactions, we believe it will be relatively easy to extend our one-way transfers to a realize bi-directional transfer.

As mentioned in the previous section, we did not have the time to incorporate a refund feature into our system. This would be an important feature to incorporate into our system, especially in a two-way transfer system as it would enable both nodes to make refunds; for instance, when an error occurred during mid-transaction. As stated in the previous section, M-PESA’s reversal API does not work but we do know that Alipay provides a refund API. Therefore, if someone wanted to implement a fully functional refund feature in our system, i.e. a refund feature in a two-way transfer system, one would most likely have to contact Safaricom support and ask them to fix their reversal API.

Another prominent future work is to optimize our system based on the design flaws that were mentioned in Section 5.4. As stated in Section 1.6, this thesis did not consider optimizing our system after it was developed - as the goal was to develop an system that supported person-to-person transfers between Alipay and M-PESA. Therefore, we prioritized establishing the basic functionality of the system without considering any optimizations.

6.4 Reflections on sustainability and ethical aspects of this work

Throughout the development of this system, we did not consider any ethical aspects due to our system being developed in a test environment (sandbox environment), meaning the payments that are conducted in our system does not use real money. Therefore, we did not need to consider any legal or ethical aspects that are crucial to consider when managing (real) money and sensitive user information (e.g. banking information). Additionally, we did not need to consider any ethical aspects
related to information/sensitive data regarding the users conducting the payments as these users are test users (fake users) created in the sandbox environments provided by Alipay and M-PESA.

Nonetheless, if this system would go live and be used by many users, not considering any ethical aspects related to privacy, user information, money, and more, when developing an inter-platform mobile payment solution would lead to detrimental effects. It is important to note that our task was to develop a system that would have the basic functionality to send money from A to B and vice versa. We were not given any responsibility to develop a system that is fully encrypted. Therefore, it is up to Centiglobe to make sure that crucial ethical aspects are handled when they are going live with our system. More specifically, we believe our system should be truthful, emphasize integrity, honest, and be fair when facilitating P2P payments. Centiglobe should consider these factors when further developing our system.

Regarding sustainability, we have not considered any aspects related to sustainability while we developed our system. We do not believe our system will have any negative impact on sustainability as we do not use any natural resources to develop our system. Although, if this system would go live, it would be energy consuming as it would require computers/servers to be up and running in order for the gateways to be active. Therefore, it would perhaps have a minor negative affect on sustainability.
References


Conclusions and Future Work


## Appendix A: Detailed results

Table A-1: Execution time in milliseconds (ms) for each procedure during testing

<table>
<thead>
<tr>
<th>Procedure Execution</th>
<th>Start Pay In</th>
<th>Start Pay Out</th>
<th>Complete Pay In</th>
<th>Complete Pay Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution 1</td>
<td>280</td>
<td>372</td>
<td>5331</td>
<td>3455</td>
</tr>
<tr>
<td>Execution 2</td>
<td>334</td>
<td>745</td>
<td>5281</td>
<td>3990</td>
</tr>
<tr>
<td>Execution 3</td>
<td>326</td>
<td>880</td>
<td>5276</td>
<td>4340</td>
</tr>
<tr>
<td>Execution 4</td>
<td>294</td>
<td>481</td>
<td>5300</td>
<td>4070</td>
</tr>
<tr>
<td>Execution 5</td>
<td>330</td>
<td>443</td>
<td>5280</td>
<td>3840</td>
</tr>
<tr>
<td>Execution 6</td>
<td>817</td>
<td>841</td>
<td>5541</td>
<td>3496</td>
</tr>
<tr>
<td>Execution 7</td>
<td>329</td>
<td>564</td>
<td>5297</td>
<td>3773</td>
</tr>
<tr>
<td>Execution 8</td>
<td>284</td>
<td>348</td>
<td>5315</td>
<td>4374</td>
</tr>
<tr>
<td>Execution 9</td>
<td>442</td>
<td>425</td>
<td>5327</td>
<td>3663</td>
</tr>
<tr>
<td>Execution 10</td>
<td>319</td>
<td>557</td>
<td>5280</td>
<td>4404</td>
</tr>
<tr>
<td>Execution 11</td>
<td>781</td>
<td>1177</td>
<td>5427</td>
<td>3606</td>
</tr>
<tr>
<td>Execution 12</td>
<td>498</td>
<td>693</td>
<td>5406</td>
<td>4110</td>
</tr>
<tr>
<td>Execution 13</td>
<td>340</td>
<td>591</td>
<td>5404</td>
<td>3720</td>
</tr>
<tr>
<td>Execution 14</td>
<td>378</td>
<td>524</td>
<td>5322</td>
<td>3144</td>
</tr>
<tr>
<td>Execution 15</td>
<td>506</td>
<td>788</td>
<td>5315</td>
<td>3405</td>
</tr>
<tr>
<td>Execution 16</td>
<td>452</td>
<td>591</td>
<td>5373</td>
<td>3228</td>
</tr>
<tr>
<td>Execution 17</td>
<td>382</td>
<td>546</td>
<td>5441</td>
<td>3445</td>
</tr>
<tr>
<td>Execution 18</td>
<td>383</td>
<td>520</td>
<td>5401</td>
<td>3936</td>
</tr>
<tr>
<td>Execution 19</td>
<td>336</td>
<td>1199</td>
<td>5313</td>
<td>6343</td>
</tr>
<tr>
<td>Execution 20</td>
<td>377</td>
<td>788</td>
<td>5337</td>
<td>3704</td>
</tr>
<tr>
<td>Execution 21</td>
<td>456</td>
<td>377</td>
<td>5353</td>
<td>3551</td>
</tr>
<tr>
<td>Execution 22</td>
<td>362</td>
<td>372</td>
<td>5398</td>
<td>3182</td>
</tr>
<tr>
<td>Execution 23</td>
<td>443</td>
<td>496</td>
<td>5359</td>
<td>3857</td>
</tr>
<tr>
<td>Execution 24</td>
<td>306</td>
<td>366</td>
<td>5330</td>
<td>3188</td>
</tr>
<tr>
<td>Execution 25</td>
<td>301</td>
<td>315</td>
<td>5332</td>
<td>7765</td>
</tr>
<tr>
<td>Execution 26</td>
<td>831</td>
<td>1473</td>
<td>5370</td>
<td>4120</td>
</tr>
<tr>
<td>Execution 27</td>
<td>402</td>
<td>409</td>
<td>5354</td>
<td>4584</td>
</tr>
<tr>
<td>Execution 28</td>
<td>367</td>
<td>468</td>
<td>5393</td>
<td>3940</td>
</tr>
<tr>
<td>Execution 29</td>
<td>320</td>
<td>353</td>
<td>5301</td>
<td>3403</td>
</tr>
<tr>
<td>Execution 30</td>
<td>385</td>
<td>460</td>
<td>6351</td>
<td>3356</td>
</tr>
</tbody>
</table>
Figure A-1: Execution time results