A Novel Mobile Transactional Payment Banking Scheme

Mahmoud Saleh Mahmoud Obaid

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree
of

Doctor of Philosophy
in
Computer Engineering

Eastern Mediterranean University
January 2020
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Doctor of Philosophy in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy in Computer Engineering.

Assoc. Prof. Dr. Zeki Bayram
Supervisor

Examining Committee

1. Prof. Dr. Albert Levi
2. Prof. Dr. Nesrin Özataç
3. Prof. Dr. Serra Sarıoğlu
4. Assoc. Prof. Dr. Zeki Bayram
5. Assoc. Prof. Dr. Alexander Chefranov

ABSTRACT

A novel real-time mobile payment (MOP) system that utilizes mobile phones together with interactive voice response technology (IVR) is proposed. The system consists of a communication network built upon existing banking infrastructures, thus minimizing the cost of adoption by banks. A sender or receiver communicates only with his/her bank, using a regular mobile phone, and banks talk to each other through a central authority, which acts as a broker. The communication of each component with other components in the system is clearly specified. Provision is made for the security of the whole system, making use of existing security arrangements between each participating bank and its customers, as well as established security practices of the banking industry. Rollbacks of transactions are also available.

Our scheme aims to achieve a high level of security without sacrificing efficiency, have a reliable infrastructure, minimize the overheads on all stakeholders and result in time savings in implementation and marketing. Through our scheme, all bank services involving transfer of funds between participants will become readily available anytime and anywhere, regardless of the availability of Internet or smart phones. For example, a user of the system will be able to pay for purchases, transfer money, track payments, and receive money in real time, alleviating the need to carry cash or credit cards.

Keywords: Communication network, real-time mobile payment, M-commerce, Interactive Voice Response, protocol, transaction, security

ÖZ

İnteraktif Ses Teknolojisi (IVR) ile birlikte mobil telefonları kullanan, yeni ve gerçek zamanlı bir mobil ödeme sistemi (MOP) öneriyoruz. Sistem, varolan bankacılık altyapısı üstüne kurulu olmasından dolayı bankaların sisteme dahil olmasının maliyetini azaltan bir iletişim ağından oluşur. Gönderici veya alıcı sadece yalın bir mobil telefon kullanarak kendi bankası ile iletişim kurar, ve bankalar da bir aracı görevi gören merkezi otorite vasıtası ile haberleşirler. Sistemdeki tüm parçaların diğer parçalarla haberleşmesi açıkça belirlenmiştir. Tüm sistemin güvenliği, müşteriler ile bankaları arasında hali hazırda var olan güvenlik önlemleri, ve bankacılık sektöründe kullanımda olan güvenlik uygulamaları üzerine inşa edilmiştir. Hareketlerin geridönüşü de sağlanmıştır.

Yöntemimiz, verimlilikten ödün vermeden yüksek düzeyde güvenlik sağlamayı, güvenli bir altyapıya sahip olmayı, tüm paydaşların mali yükünü en aza indirmeyi, ve hem implementasyonda, hem de pazarlamada zaman kazandırmayı hedefler. Yöntemimiz aracılığı ile, fon aktarımı içeren tüm bankacılık hizmetleri, İnternet ve akıllı telefonlar olmadan, her zaman ve her yerde kolayca gerçekleştirilebilecektir. Örneğin, bir sistem kullanıcısı, para veya kredi kartı taşımasına gerek kalmadan, gerçek zamanlı olarak alış-verişleri için ödeme yapabilecek, para aktarımı yapabilecek, ödemelerini takip edebilecek, ve ödeme alabilecektir.

Anahtar Kelimeler: İletişim ağı, gerçek zamanlı mobil ödeme, M-ticaret, interaktif ses teknolojisi, protokol, hareket.

ACKNOWLEDGMENT

First, I would like to thank almighty ALLAH for I have finally finished the writing of this thesis for my PhD degree in Computer Engineering despite all the difficulties and the unstable situation in Palestine as a result of the Israeli occupation and travel restrictions.

I would like to express my deepest gratitude to my supervisor Assoc. Prof. Dr. Zeki Bayram for guiding me well throughout the research work, from the title's selection to finding the results. His immense knowledge, motivation and patience have given me more power and spirit to excel in the research writing. Conducting the academic study regarding such a difficult topic couldn't be as simple as he made this for me. He is my mentor and a better advisor for my doctorate study beyond the imagination.

I would like also to thank all the department staff who gave me the support and help I needed during the writing of my thesis.

I would like to thank my wife Amena, my children Anhar and Rayan for their endless support and understanding during this thesis process, and my brother Murad for his support and encouragement.

In the end, I am grateful to my parents, siblings, friends and acquaintances who remembered me in their prayers for the ultimate success. I consider myself nothing without them. They gave me enough moral support, encouragement and motivation to accomplish my personal goals.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF LISTINGS

LIST OF ABBREVIATIONS

API	Application Programming Interface
B2B	Business to Business
B2P	Business to Person
C2B	Consumer to Business
GDP	Gross Domestic Product
HSM	Host Security Module
IVR	Interactive Voice Response
MNOs	Mobile Network Operators
MOP	Mobile payment proposed system
M-Payment	Mobile Payment
M-PESA	Mobile Swahili for money
MSISDN	Mobile Subscriber Integrated Services Digital Network
NFC	Near Field Communication
P2B	Person to Business
P2P	Person to person
PDA	Personal Digital Assistant
PIN	Personal Identification Number
RTGS	Real-Time Gross Settlement
SIM	Subscriber Identity Module
SMS	Short Message Service
TIPS	Target Instant Payment Settlement
USSD	Unstructured Supplementary Service Data

Chapter 1

INTRODUCTION

The adoption of digital innovation initiatives has resulted in the development of disruptive technologies that have shaped traditional industries in many ways [1]. One such invention was the development of the mobile payment platform that has changed the dynamics of the financial sector. The platform that is also called M-payment became the main idea of disruptive technologies and a boon for organizations seeking a system for reimbursement in the 21st century [2]. Further, the technology has no geographical limits and can be used both domestically and internationally. Notably, it can be used to make payments to friends, institutions, and governments. Thus, mobile money is any service that enables the transfer of funds through devices such as smartphones, tablets, PDAs [3].

1.1 Mobile Payment Definition

Mobile payment (M-Payment) is an interesting field of work for the researchers in areas like telecommunication, mobile computing, security and wireless networking. It is defined as any type of payment for goods, services, bills, government fees, transfers, or any other money transaction using mobile device or cell phone to initiate, authorize and confirm the transaction. Another definition for mobile payment which is more specific is: a transaction handled using mobile phone and payment instruments such as banking i.e. (cash, bank account, credit card) or pooling account i.e. (PayPal, wallet, ..etc.) and not using carrier billing or telebanking which is not integrated with banking infrastructure [4].

Mobile payment forum defines the mobile payment to be the transactions of a monetary with value via mobile telecommunication, mobile network, mobile users, cellular telephones, smart phones or PDA's, and mobile terminals. This process manages all kinds of payments, service payment and purchases for a confirming payment is preferable than the traditional payment card-based in terms of getting the financial process anywhere for both the recipient and the bank [5].

1.2 Reasons for the Development of Mobile Payment

Firstly, the development of mobile payment platforms has mostly been promoted by governmental regulations, technological changes, and industry competition. Notably, governments across the world have passed legislation aimed at developing innovations in the mobile phone sector [6].

Further, by allowing the sending and receiving of money through mobile phones, governments opened up the development and adoption of mobile payments. Also, technological changes have led to the mass development and production of smartphones. The availability of these devices enabled developers to disrupt the payments sector by developing systems that would make such services accessible to consumers. Industry-wide competition among financial players and technology companies like Apple, Google, and Samsung led to the development of mobile payment applications such as Apple Pay and Google Wallet that led to the further adoption of mobile payment. Thus, the technology developed out of the need for organizations to improve service delivery and create a competitive advantage.

Mobile commerce or M-commerce is a new kind of E-commerce conducted using wired and wireless technologies. It's beneficial for both a customers and governments. Mobile payments can be considered as one of the most significant tools of mobile commerce for its ability to replace other bank tools. Developments in mobile commerce have naturally increased the need for mobile payments and fueled their development.

1.3 Benefits of Mobile Payment

M-payment has had tremendous effects on businesses. It has helped fuel competition as companies seek unique ways to use the technology to edge out rivals and improve the quality of service delivery. Besides, mobile phone manufacturers such as Apple, Google, and Samsung have also ventured into the mobile payments business with the goal of diversifying their products. The diversification strategy has resulted in the development of products such as Google Wallet, Apple Pay, and Samsung Pay alongside well-known platforms such as PayPal and Stripe. Banks have also joined the race to get a share of the mobile payments market and even remain relevant in an industry that is fast going mobile [7]. For example, today many financial institutions have a revenue-sharing agreement with mobile payment firms that allow them to profit from every cash transaction made on mobile phones [8]. In so doing, they have become a vital part of the mobile payments infrastructure. Thus, M-payments have permeated the socio-economic aspects of society and changed the mode of business in the 21st century.

1.4 Outlook for Mobile Payments

It was predicted that by 2019, the world would have about three billion smartphones in active use [9]. The ownership of these devices on such a large scale would result in the broad adoption and use of mobile payments. In fact, Gartner Group predicted that the global M-payment market would exceed 450 million users coupled with monetary transactions of over $930 billion [10]. Furthermore, significant parts of the world still do not have an internet connection. As such, mobile phones remain the device of choice for making payments as people adopt technologies such as Interactive Voice Response (IVR), Short Message Service (SMS), and Unstructured Supplementary Service Data (USSD) [11].

Mobile payment has, in our opinion, a bright future, since it finishes the need for any other instruments of payment. It will replace all types of traditional and famous tools of payment such as cash, cheques, debit card and credits. Also, the SIM card has a great level of security, a very important consideration in any monetary transaction.

1.5 Problems with Current Mobile Payment Systems

Mobile Payment Systems have enabled the provision of a convenient, automated means to transfer money without the need to handle cash, use credit cards or other non-mobile based payment methods. However, the current implementations of mobile payment systems have several limitations that prevent them from reaching their full potential. Firstly, the system is dependent on external third parties (entities or organizations outside of the financial system), which necessitates that customer opens separate accounts that are different from the regular bank accounts. For example, in some instances, accounts are dependent on one’s telephone number. In this respect, customers have to operate accounts that are different from ones managed by their bank, and they also must replenish them by either cash deposits or credit cards. Furthermore, the process of managing mobile money customer accounts entails getting bank guarantees to cover the transaction’s monetary value. Another limitation is that any new updates to the system often require that customers and agents replace their handsets or SIM chips. Besides, users must have accounts in one bank only to conduct Real Time Person-to-Person (RT P2P) transactions. Lastly, the nature of M-payments does not promote real-time payment because of the lengthy process involved in transferring money from one bank to another. Thus, these factors limit the full exploitation of mobile money payment technologies.

1.6 A Novel Solution to the Current Problems of Mobile Payment Systems

In light of the problems specified above, we propose a unique communication network and infrastructure that will overcome the hurdles that limit M-payment transactions. Our proposed system MOP will connect banks to each other and to customers to allow all mobile phone-based transactions and payments seamlessly and securely. It will satisfy the needs of all stakeholders by creating an efficient and secure system. It will allow users to pay, track, manage, and receive money as long as they are using the system. Mobile financial services, in this case, will include P2P, C2B, and B2B transactions, microfinance services, and mobile banking. Thus, it will ensure an efficient and effective integration with other systems [11].

1.7 Structure of this Thesis

In this chapter, we gave the motivation for mobile payment, presented limitations of current mobile payment systems, and proposed a novel mobile payment system that overcomes these limitations. In Chapter 2 we present published research on mobile payment systems. In chapter 3 we discuss existing mobile payment methods and systems that are currently in use. Chapter 4 we give a detailed, yet high level, description of our proposed mobile payment system. In chapter 5, we give a specification of our implementation in the form of abstract classes and methods, which define how components of the system communicate with each other. In chapter 6, we describe a prototype implementation for the MOP system. We have discussion on the benefits and limitations of the proposed system, as well as a qualitative comparison with other payment systems in chapter 7. Finally, in chapter 8, we have the conclusion and future research directions. In appendices A, B and C we present the critical sections of code used in the prototype implementation for our proposed system. In appendix D we present a simulator for our proposed system. Appendix E contains the source code of the simulator. In appendix F we have raw timing data for transactions performed on the prototype implementation. Full code of the prototype implementation and simulator is available both in the CD accompanying this thesis, and online [47].

Chapter 2

RELATED WORKS

The disruptive nature of M-payment has prompted numerous studies on the dynamics of the technology, more so, its ecosystem and adoption [2]. Some studies have documented the impact of the technology on the banking and financial sector including the resources and assets of these institutions [12]. Some scholars studied the effect of the technology in Netherlands, more so, the relationship between banks and mobile network providers (MNOs) [13]. Another research in Denmark revealed how the digitalization of payments, as a technology innovation, affects the competition and collaboration among traditional and new stakeholders in the payment ecosystem at three levels of analysis [1]. A study by [14] showed that timing has and significant effect on the impact on the success of the adoption of mobile payment. Other scholars have studied the success of mobile payments in developing and developed countries.

One of the most studied mobile payment systems is in Kenya where the development of a service called M-PESA revolutionized the financial sector. The rapid adoption of mobile phones in the country led to the introduction of a mobile payment system to improve the circulation of money. The project was a success because, currently, M-PESA payments make up a quarter of the country's GDP [15]. The adoption of mobile payments in rising exponentially. In countries like China, approximately 80 percent of internet users make payments using their smartphones [16]. A study by

[17] revealed that users would soon adopt contactless credit card payments with smartphones that use radio frequency identification or near-field communication technologies. These advancements will become the most used platforms for mobile payments.

A person’s technological frame is essential in making them adopt NFC enabled smartphones. In this respect, consumers need to have a good understanding of the technology including how it will ensure the prevention of risks and threats associated with money transfers [18, 19]. Besides, stakeholders need to provide adequate information to help correct the negative perceptions of consumers towards the technologies [17, 20].

Countries like Canada already use the NFC enabled phones in making payments [21]. Thus, contactless payments are increasingly being adopted because they are believed to be convenient, safe, and open to further development [22].

Research on the security of mobile payment systems revealed that there are risks such as user masquerading, malware, man-in-the-middle attacks, and traffic interception. The lack of a two-factor authentication increases the risks of sending money to the wrong account or someone stealing a password [10]. Financial institutions have responded to these threats by providing consumers and dealers with trusted service managers that approve and verify any mobile-based transactions [23]. The process is geared towards making customers, and third parties trust the system and ensuring the safety of all partners. The delicate nature of the transactions between banks and mobile payment service providers also necessitates that any conflicting issues be addressed efficiently to achieve set objectives [24].

Mobile payment technologies have been designed to appeal to users. For example, they allow one to personalize the applications, use geo-targeting features, and even integrate social networks. Besides, the data is user generated. As such, the future of mobile payments looks promising [9]. Studies on the technology have reinforced the need to integrate the system into everyday use without destabilizing traditional payment methods [25]. The attainment of those goals requires the collaboration of all stakeholders in the payment industry. On the other end, some researchers assert that there is a need to overcome country-level institutional constraints including those that govern access to resources, to ensure the successful implementation of m-payment systems [26].

The success of mobile payment technologies is based on factors such as trust, user-friendliness, perceived usefulness, cost, and mobility. Several studies have been conducted to investigate the effects of these factors on mobile money adoption [2]. Other reviews focus on the cultural implications of the technology [27], while others have studied how its adoption is influenced by the effect of mental accounting theory [28]. A study by [29] which was based on learning theories, centered on the technology usage habits of users influenced mobile payment adoption. Research by [30] investigated payment habits of m-payment users, while [31] examined the effect of trust and technological environment on use of mobile money services.

[25] investigates the relation between technology for m-payment and its influence on the ecosystems, and how this payment can integrate with ecosystem without unstable influence on other traditional payment and keep the ecosystem stable with collaboration between these different stakeholders.

[26] claims that m-payment depends on country level institutional constraints. [12] investigates the main impacts of mobile payment on banks and financial institutions and requirements for m-payment from resources and different assets.

[13] investigates the collaboration between banks and mobile network operators (MNO's) in Netherlands.

[1] uses a new multi-level framework to study cooperation between different ecosystems, business, technology and theory. Using Denmark as a case study this article also differentiates between technology from both defensive and offensive strategies in m-payment.

[14] studies the timing factor and its impact on the optimal decision for success and completive advantage in mobile payment.

[24] analyses an in-depth case on collaboration between three major Dutch banks and three Dutch telecom operators who jointly developed a trusted service manager for mobile payment.

[27] examines the impact of espoused national cultural values on consumer's intention to use mobile payment devices using the UTAUT and Hofstede's cultural dimensions as the basis for the research framework.

[28] examines high speed rail (HSR) passengers' acceptance of mobile ticketing services, as indicated by their mobile access for ticketing information inquiries and use of quick response codes (QR codes) for payment and gate entrance.

[32] presents a qualitative study based on experience of six NFC pilots implemented in Finland, France, Italy, the Netherlands, Norway, and the UK. The research findings confirm that a number of demand and supply barriers negatively affect the rate of the penetration of the NFC payment. [33] tries to provide a clear account of the knowledge that exists on mobile payments. [34] tries to unveil factors explaining the failure of past mobile payment platforms.

The trust and risk are studied with the relation to the cost, social influence and convenience as main factors to investigate in any m-payment services in addition to the factors that have to be investigated in wide scientific scenarios that impact the consumer acceptance of m-payment services like ease of use, added value (useful & usefulness), compatibility, mobility, privacy, habit, quality, experience, income, satisfaction, complexity and uncertainty avoidance. Many articles e.g. [27-29, 35, 36] discuss one or more of such issues in each article or research studied using m-payment in parking ,fare tickets and money transfer addressed the adoption factors for different technologies [30] addressed the payment habit [31] investigated trust in many stakeholders and issuers in mobile service provider, vendor, environment and technology.

Overall, mobile payment technologies are developing at a blistering pace. Besides, users keep increasing, which paints the picture of a promising future. Phone manufacturers are also fast venturing into this market and keep adding new features in smartphones that seek to advance the adoption of m-payment. However, the successful adoption of the technology is pegged on factors such as trust, its user-friendliness, governmental regulations, cost, and mobility. Several studies have been conducted on the use of the technology and most of the reveal a rise in its use and its positive disruptive nature in the banking and finance sector. In this respect, there is a need to develop a system that would ensure seamless and secure transactions to promote its continued use.

Chapter 3

EXISTING MOBILE PAYMENT METHODS

Presently, the modern digital world has various payment methods, each with its unique operational techniques depending on the needs of the business models. Below is a discussion of the popular M-payment systems, and how these services function.

3.1 PayPal

PayPal, a proprietary system owned by PayPal Holdings, Inc., functions by posing as a third-party between the merchants and the customers. As [37] explains, all transactions are performed via the merchant. Besides, PayPal does not charge the sender for any transaction. Rather, all charges are incurred by the payment receiver. PayPal users have the option of using three distinct payment methods: (i) payments can be made through the web browsers, (ii) short messages (SMS), as well as (iii) through phone calls. To start operating, PayPal only needs a mobile phone and a PayPal account. However, the account should be activated by linking it to a registered financial system, such as a bank account or a credit card.

3.2 PayBox

PayBox business model employs text messages, which are sent from the client's phone number to the merchant. Subsequently, the merchant is required to send all the client's information, including the expected cost to PayBox [38]. Next, PayBox verifies the order placed by the client through a voice message. After the authentication, the client proceeds with the payment by sending a PIN code to PayBox. To complete the transaction, PayBox requests the bank to perform the transaction and use an SMS report to notify the merchant that the process is complete.

3.3 PayForIt

PayForIt is another payment method that allows users to purchase digital content and services like videos, applications, as well as games. This mobile payment service is very popular in the United Kingdom and has been adopted by a majority of the mobile phone operators. The system can be used for micropayments, where the users only need a mobile device to make payments [39]. The main users involved in the operations of the application include the clients, merchants, operators, as well as personnel tasked with authorizing the payment intermediary. The client is required to sign a contract using their mobile device through an authorized payment intermediary. A screen displaying the information regarding certain goods, costs, and the respective merchant is presented to the consumer. To complete the process, the operator is required to authorize the payment, upon which the cost is transferred to the merchant’s mobile account.

3.4 Square

Square is a point-of-sale (POS) system that provides enterprises with fast and reliable transactions. Square, which can either be used via the counter (cashier) or on the go, allows its users to receive payments using their Android and IOS devices [40]. Square comprises a magnetic stripe reader that supports payments made through credit and debit cards. Other add-on features include free downloadable software, item management platform, as well as a real-time sales and inventory tracker. Square uses the transaction-based business model, where users are charged a certain percentage depending on the nature of the transaction.

3.5 TIPS

TARGET Instant Payment Settlement (TIPS) is a relatively new market infrastructure service that was launched by the European Central Bank (ECB) to facilitate real-time monetary transactions for all its service providers [41]. Instant payment transactions verified via the TIPS platform are irreversible. Financial institutions using this system must satisfy the liquidity threshold that is pre-defined by the ECB to ensure payments are made in real time. Consequently, this fast and instant payment (in any currency) assists users to avoid credit risks. Ultimately, the TIPS system is valuable in enabling users to access financial services while ensuring secure transactions.

TIPS achieves real-time settlement of payment transactions across all platforms through the development of a middleware application that allows all payment systems to complete successfully [41]. To promote this, ECB allows third-party developers to develop and integrate their payment solutions to the TIPS middleware. Consequently, ECB is responsible for ensuring that the different parties settle their payments. The middleware categorizes clients into three distinct groups, with all the parties required to have deposited funds at the central bank. End users can register with their preferred parties.

Chapter 4

PROPOSED SOLUTION (MOP SYSTEM)

In this chapter, we describe our proposed mobile payment scheme in some detail.

4.1 Overview of the Proposed System MOP

Indeed, the invention of the mobile phone changed the banking industry in tremendous ways. Today, most financial institutions have already adopted mobile telephony in streamlining their services and connecting with their customers. These changes have been mandated by the vast use of smartphones by people even in the remotest parts of the world. The integration of mobile-based monetary transaction in banking has permeated sectors such as P2P, B2B, and P2B and made it easy and secure for people to access financial services through mobile phones.

In our proposal, there is no specific type of mobile phone required for the service. Rather, it works on building a whole and unified service for all involved parties. The main idea the exclusion of any external third party and relying on the current system and its credibility. In general, a *third party* is an authorized online service provider that has been introduced as part of Open Banking. It exists outside of your relationship with your bank, but may be involved in the online transactions you carry out, which means all money transactions will transfer to the third party first. *Third party* in the banking and financial sector is any party who is outside the financial sector that can authorize transactions and have a role as the controller for such financial transactions (e.g. Mobile operators, wallets, etc.). This means that the third party can own customer, merchant and transaction information in addition to the sharing of fees. The connection between the old system and the new one saves time and trust for the beneficiaries.

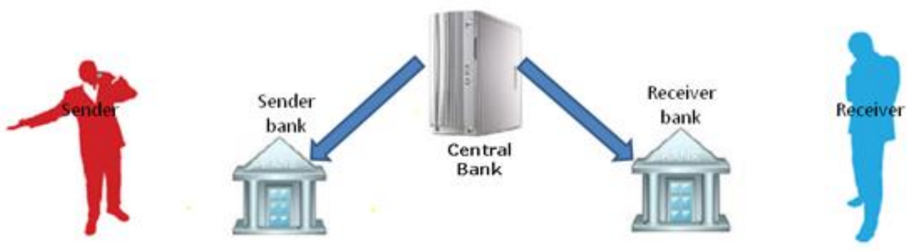


Figure 4.1: Solution Platform.

As shown in Figure 4.1, any recipient or sender should be connected to the bank account through his/her mobile phone. Depending on the transaction type, a bank may be an issuer (the party that sends money), or an acquirer (the party that receives money). In this mechanism, private information about users is kept in the banks only. Only the name, account number and mobile phone number are saved in the proposed system. The central bank is the correspondent between the banks and is used to help in making the financial movements of the clients of other banks in addition to its role in monitoring, legislation and evaluation.

The system works on the powerful payments system which becomes a source of trust and development for the users and beneficiaries. It excludes any external third party as defined above, especially mobile network operators (MNO's), which do not always enjoy the trust of users. So, all processes are controlled by banks, the users, beneficiaries, and monitory parties such as the central bank and financial institutions. Most of the service suppliers and consumers have bank accounts, hence the positive response to any proposed system involving banks that they trust. The central bank and the financial governmental authorities support this system which helps in monitoring, legislating and evaluation. This is the Real-time gross settlement system (RTGS), i.e. A specialist funds transfer system where the transfer of money or securities, takes place from one bank to another on a "real time" and on an economic basis.

There are two sub-systems in our proposal: Client and operator. These are explained below:

4.1.1 The Client Component

The client component will be used exclusively by the bank, connected to its core banking system. To ensure the security and efficient transfer of data, ISO 8583 will be used for data exchange. ISO 8583 provides a framework for creating protocols for the exchange of financial transaction messages. Typically, these are messages that involve transactions originating. It's important to realize that 8583 itself is not a protocol, just as XML isn't a file format. XML can be considered a description of how to specify file formats for structured data according to a set of rules. ISO 8583 is a metaprotocol providing a set of rules for the definition of financial transaction protocols. Also, the client’s data will be coded by the bank in the host security module (HSM), which will keep all data related to the client. No part of the data, except names, mobile phone numbers and account number of clients will be made available to other parties. Requests will be sent from MOP operator to client bank by TCP/8583 protocol. Firstly, the client will create a connection request by sign-ON. At this time the server will accept the connection. Then the client will request the transaction by message (8583 message), and server will send this request to the core bank system. After the transaction is done successfully or cancelled, the server will send response to this request. Finally, connection is terminated, as shown in Figure 4.2.

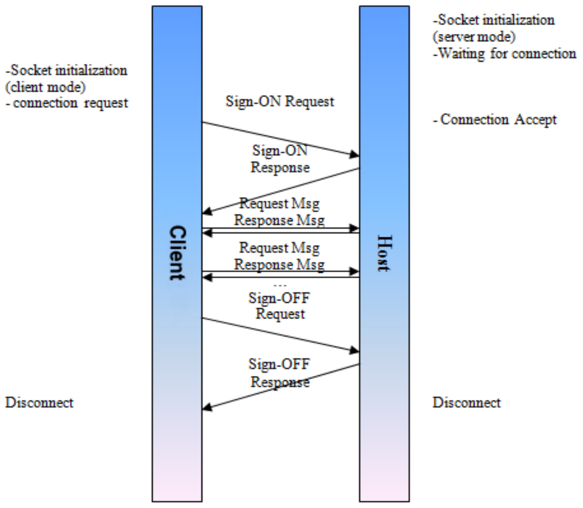


Figure 4.2: Messages Exchanges between Client and Core Bank System.

4.1.2 The Operator Component

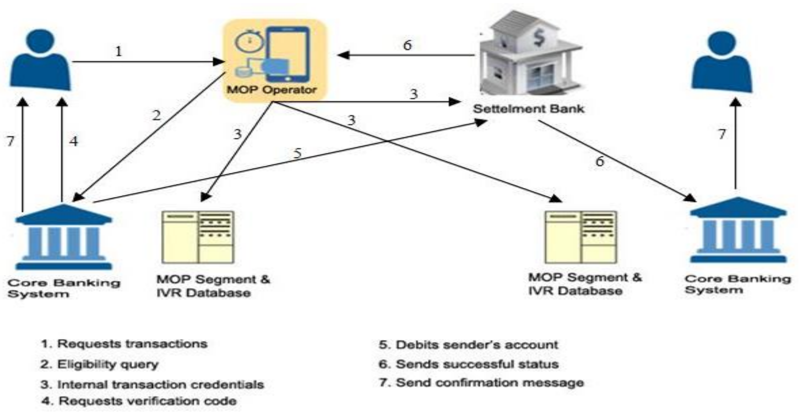
The operator component will be installed in the central bank or any organization appointed by the government to monitor the financial movements in the country. This part connects all banks together without need to connect each bank directly to other banks, thereby helping in preventing money laundering. The PIN and other confidential information of any client, such as historical data, account details and personal data will be authorized and used only by the customer’s bank. What will be available to the MOP system is the name, mobile number, money amount and the bank account number.

4.2 Operation of the MOP System

MOP will provide a fast, timely, and secure connection between old and modern systems. Furthermore, the MOP system is user-friendly and provides a secure medium that is based on modern technological tools. Its operation entails firstly taking a command from the MOP operator through input mechanisms such as the

IVR, SMS, USSD, or browser. The system prompts the user to provide details such as the acquirer MOP ID, amount to be transacted, and the currency. Secondly, the details issued by the client are sent to the issuer's bank for verification. Thirdly, the transaction details are forwarded to both the financial institutions and settlement bank once the request has been validated. The next step entails the issuer bank requesting for the verification code sent to the sender by the MOP system. The code helps in ensuring the integrity of the transaction to prevent any unauthorized transactions. Afterwards, the issuer bank debits the client's account and sends a confirmation message to the settlement bank that also sends a message to the bank to confirm the transactions. Lastly, the issuer bank and acquire bank send a notification to the issuer and the acquirer to confirm the success of the transactions. Thus, this mode of operation ensures the authenticity of the transactions and inspires trust in customers.

As a behavior of the system when any transaction is initiated and created in the operator, the transaction is directly replicated to the sender and receiver bank client, and when any error (regardless of the reason) occurs and any of the sites change the status of the transaction to "cancel", the status of all sites will be replicated to cancel the transaction. This property is called auto replication. Figure 4.3 depicts the sequence of actions that need to take place in our proposed method.



MOP Operator
Settlement Bank
Core Banking System
MOP Segment & IVR Database
MOP Segment & IVR Database
Core Banking System
1. Requests transactions
2. Eligibility query
3. Internal transaction credentials
4. Requests verification code
5. Debits sender's account
6. Sends successful status
7. Send confirmation message

Figure 4.3: New Mobile Payment Behavior.

The actions performed by each actor in the system are given in Listings 4.1 through 4.4 as pseudo-code. The function calls in the pseudo-code belong to our implementation of the MOP system as a simulator.

Listing 4.1: Actions Performed by the MOP Operator

MOP_operator

- Wait for transaction request from customer, then create transaction by ivrConnectionAction($url, $postdata) and autoaddcdrAction( )
- Get MOP ID of the receiver, amount, and currency, then collect information by getClient($MOP_id)
- Send eligibility query to sender_MOP bank by connectiontoserverAction($data, $ip,$return=null), connoperatorAction($data, $ip).
- Wait for eligibility response from Sender_MOP bank, by using excuteurlAction ($cdr_url, $encry_data)
- Send transaction details (MOP ID of the sender, reciver, amount, and currency) to sender_MOP bank, Receiver_MOP bank, and settlement bank by using sendtrxreciverbankAction( )
- If the connection failed, time is over 45 second, replay delay, or canceled, the transaction will be rolled back by changing status using changestatusAction( ) and autochangestatusAction()

Listing 4.2: Actions Performed by the Sender Bank

Sender_MOP

- Wait for request from MOP operator by using ivrConnectionAction($url, $postdata) and autoaddcdrAction( )
- Get MOP ID of the sender and receiver, amount, and currency using getClient($MOP_id)
- Check MOP ID of the sender and receiver, sender found and limit, and receiver black listed by using getBlacklistDao($MOP_id), getTransOnMinDate($limitation_date, $phone_id,$MOP_id)
- Send response to MOP_operator by using responseMOPAction()
- Create connection to sender by making new call by using callbackAction()
- Wait for PIN code, confirmation of transaction from sender by using athinticationlevelAction().
- Debit sender account by bank system and change status for transaction using changestatusAction( ) and autochangestatusAction()
- Send “successful” message to the settlement bank by using cbAction($data, $ip)
- Send successfully messages to sender using sendSMSAction($phone, $MessageBody)
- If the connection failed, time over 45 second, replay delay, or canceled, the transaction will be rolled back by change status to canceled using changestatusAction( ) and autochangestatusAction()

Listing 4.3: Actions Performed by the Receiver Bank

Receiver_MOP

- Wait for request from MOP_ operator, using getRecorde($MOP_id, $phone_id)
- Get MOP ID of the sender and receiver, amount, and currency by using getClient($MOP_id).
- Wait for response from settlement bank for success transaction if status is changed.
- Credit receiver account by bank system.
- Send successful messages to receiver by using sendSMSAction($phone, $MessageBody).

Listing 4.4: Actions Performed by the Settlement Bank

Settlement bank

- Wait for request from MOP_ operator.
- Get MOP ID of the sender and receiver, amount, and currency by using getClient($MOP_id).
- Wait for response from sender_MOP bank for success transaction if the status is changed.
- Debit sender_Mop bank by bank system.
- Credit receiver_MOP bank by bank system.
- Send successful transaction status to receiver_MOP bank by using changestatusAction( ) and autochangestatusAction()
- If the connection is failed, or time over 45 second, the transaction will be rolled back: Send cancelled transaction using changestatusAction( ) and autochangestatusAction()

4.3 Features of the Proposed System

This system will enable the user to:

1. Initiate a payment using his/her mobile or landline.

2. Easily track and manage his/her transactions.

3. Send/receive money for selling/buying products and services in an automated way.

4. MOP Solution will provide its users with a detailed history log of all performed transactions, whether it was successful, canceled or an aborted transaction. Users can view his transaction log either thorough the web interface of MOP system or by calling MOP Operator.

5. The archive storage of our MOP Solution, whether hosted in the MOP operator or at the bank, will store all the transaction details for the users.

6. MOP Number will be a short code that will be the same among all GSM operators and hopefully land line service provider.

4.4 How the System Works

Users must have a valid unique MOP account number and PIN to use the system. MOP system has two types of IVR: (i) a centralized inbound to receive requests from users, which is part of MOP operator, and (ii) decentralized outbound placed in each bank to contact the users and confirm their requests in order to proceed with the transactions, which is part of MOP bank. One centralized server (MOP operator) is responsible to manage, transfer, store, and update requests' status, and stores all MOP user accounts received from banks' servers, as shown in Figure 4.4.

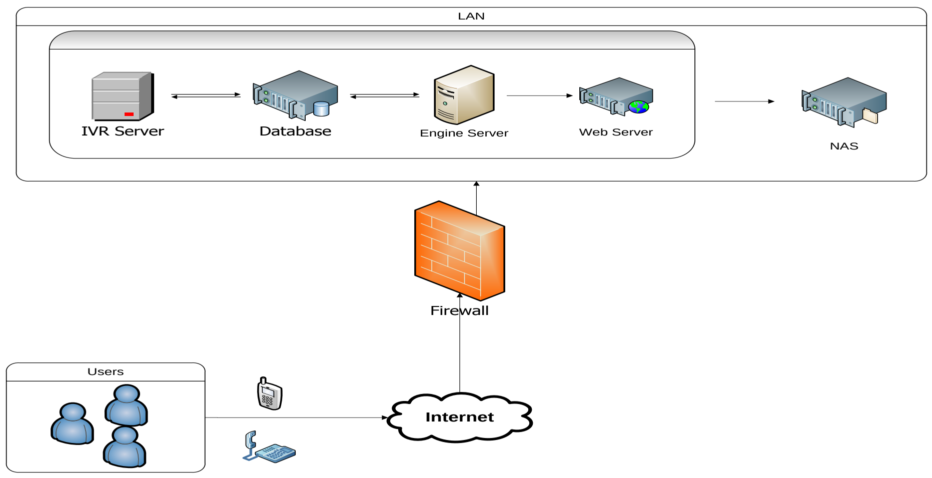


Figure 4.4: MOP Operator Component.

Many decentralized servers (MOP bank) placed in each bank are used to create, store and send bank user MOP accounts to the MOP operator. Also, they are responsible for receiving open requests, processes and updates. They inform the MOP operator about their status.

Each one consists of two sub systems, (i) MOP interface which is the part that is connected to MOP operator, and (ii) MOP bank segment that is connected to core bank system. MOP operator has one database used to store all MOP accounts and transactions, and two interfaces, one for bank admin to create MOP user accounts, and the other one for users to check and print out transactions status as shown in Figure 4.5.

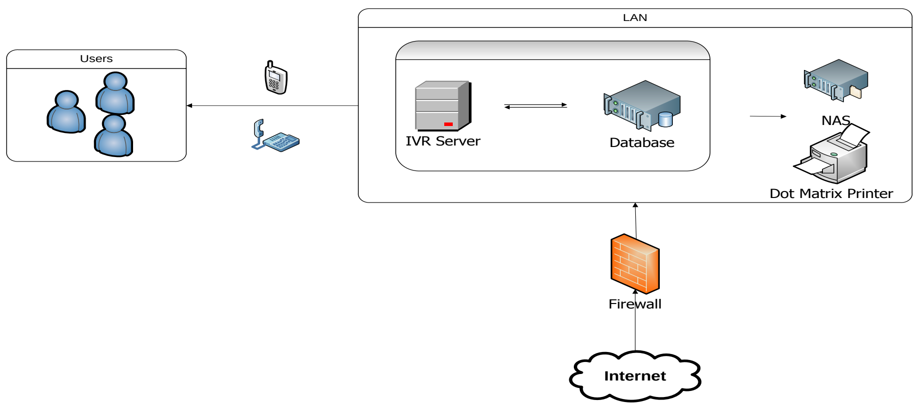


Figure 4.5: MOP Bank Component.

To operate, the MOP user will be required to dial the Inbound IVR number and either selects English or Arabic as the preferred language. After verifying the identity of the user, the IVR will display a welcome message and inform the user about all the information such as call cost, and prompt the visitor to select a payment service. The system will query the user's transaction limit and check if the user is blacklisted using the core bank system. Next, the user will be prompted to enter the receiver MOP account number, the MOP receiver ID and the amount to be paid. After the sender confirms the transaction, the IVR system then will display transaction details, while the system checks if there are enough funds in the MOP sender bank account. If so, the system will make a call to the user informing him/her that the transaction is being processed. As a security measure, the user is only allowed to make three trials for receiver number, after which the process is canceled. Figure 4.6 depicts the inbound IVR flowchart.

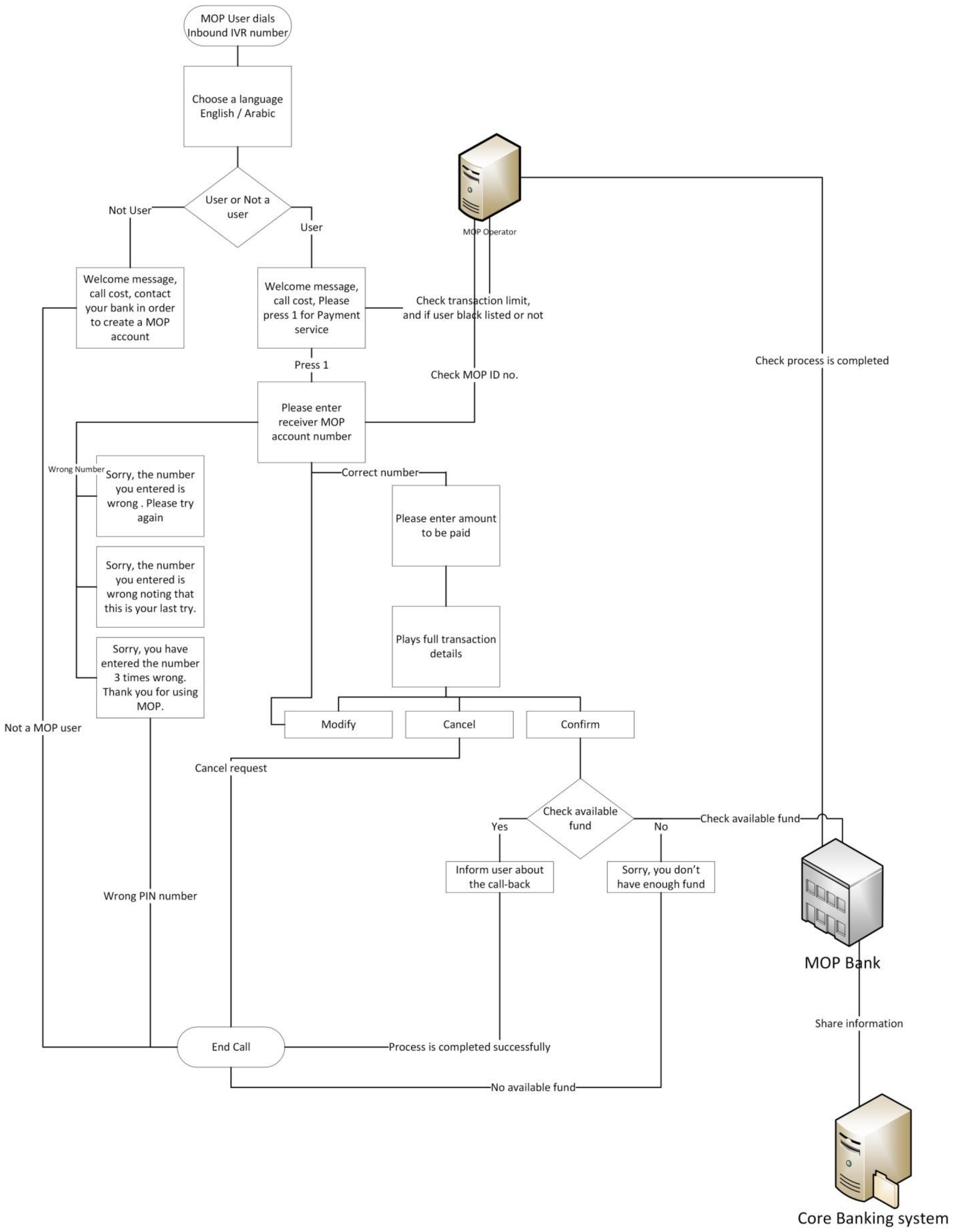

MOP User dials Inbound IVR number
Choose a language English / Arabic
User or Not a user
Not User
User
MOP Operator
Welcome message, call cost, contact your bank in order to create a MOP account
Welcome message, call cost, Please press 1 for Payment service
Check transaction limit, and if user black listed or not
Check process is completed
Press 1
Check MOP ID no.
Please enter receiver MOP account number
Wrong Number
Sorry, the number you entered is wrong . Please try again
Correct number
Please enter amount to be paid
Sorry, the number you entered is wrong noting that this is your last try.
Plays full transaction details
Sorry, you have entered the number 3 times wrong. Thank you for using MOP.
Not a MOP user
Modify
Cancel
Confirm
Cancel request
Check available fund
Yes
No
Check available fund
Inform user about the call-back
Sorry, you don't have enough fund
Wrong PIN number
MOP Bank
Share information
End Call
Process is completed successfully
No available fund
Core Banking system

Figure 4.6: Inbound IVR Flow Chart.

Next, the MOP operator makes a request to the MOP bank, prompting the bank to contact the sender as a security measure, describing the entire transaction details and requesting the sender to enter his/her PIN number. If the correct PIN is entered before the three trials, the IVR then confirms the message and the status of the process changes to *complete*. Eventually, the MOP bank transmits a confirmation message to the settlement bank and operator. The settlement bank then replies with a confirmation message to the receiver bank. Figure 4.7 describes the outbound IVR flow chart.

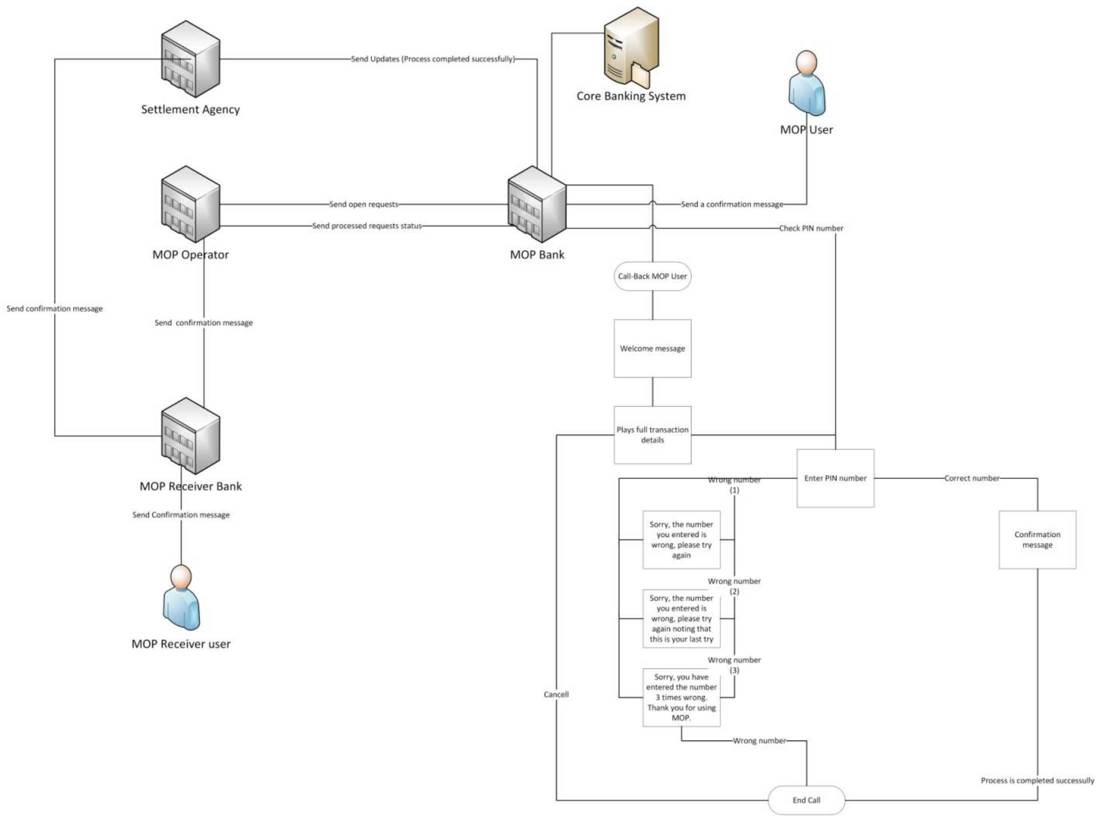

Settlement Agency
Send Updates (Process completed successfully)
Core Banking System
MOP User
MOP Operator
Send open requests
Send processed requests status
MOP Bank
Send a confirmation message
Check PIN number
Call-Back MOP User
Send confirmation message
Send confirmation message
Welcome message
Plays full transaction details
MOP Receiver Bank
Send Confirmation message
MOP Receiver user
Enter PIN number
Wrong number (1)
Correct number
Sorry, the number you entered is wrong, please try again
Confirmation message
Wrong number (2)
Sorry, the number you entered is wrong, please try again noting that this is your last try
Wrong number (3)
Sorry, you have entered the number 3 times wrong. Thank you for using MOP.
Cancell
Wrong number
Process is completed successully
End Call

Figure 4.7: Outbound IVR Flow Chart.

4.5 Use Cases of the MOP System

Below we describe typical use cases of the MOP system.

4.5.1 Bank Admin Use Case – Create MOP User Account

Bank admin accesses the web-based MOP admin interface in order to manage needed MOP user accounts (create, modify, delete, and re-set PIN numbers). As shown in Figure 4.8, which describes all the steps, the user requests an MOP user account from the bank customer service desk, then the bank customer service provides the user with MOP account application form. This form will vary based on the user requesting an MOP account, as the form for individuals is different from the form for corporate or service providers. After that, the user hands out the filled MOP form to bank customer service. Bank customer service desk opens the MOP admin interface in order to fill user information into the system based on the filled MOP account form and assigns an MOP ID from the available IDs generated by MOP operator. (MOP ID number will have 10 digits, 3 of them represent the country code). MOP bank Segment will generate MOP PIN number for the user. This PIN will be stored in the MOP bank database only and will not be transferred to the MOP operator. (MOP bank Segment will be responsible to generate MOP PIN; MOP user can have more than one PIN number for one MOP ID associated to his/her registered call-back number/s). Once bank service desk creates the new MOP user account, all information except sensitive ones will be sent to the MOP operator (Call-back numbers, any other information that is confidential by the bank). Then, a certified courier will send the PIN number to the user.

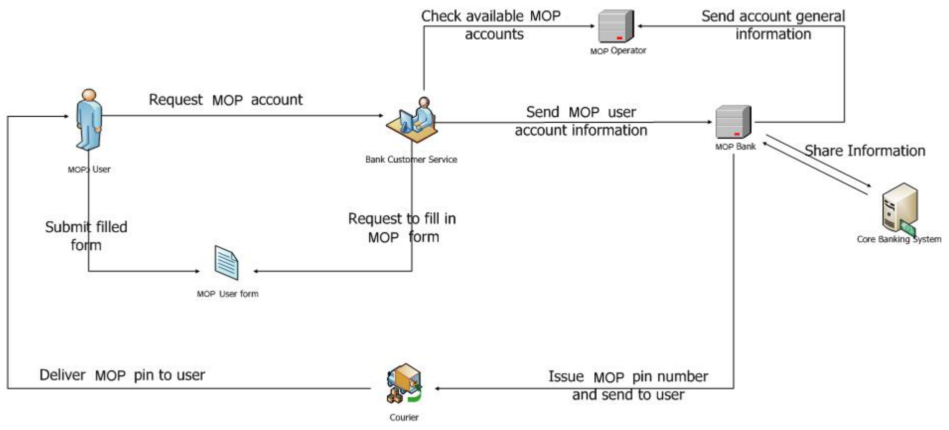


Figure 4.8: Create MOP User Account.

4.5.2 MOP User Use Case – Follow up on Transactions

After the bank admin creates MOP user account, the user can access the web-based interface in order to follow up and manage his/her transactions. We can describe these steps as shown in Figure 4.9 where we have a user accesses MOP user interface. Then, the user enters MOP account and password. MOP bank will generate the user interface password, which enables searching for needed transaction. Finally, we have "check status".

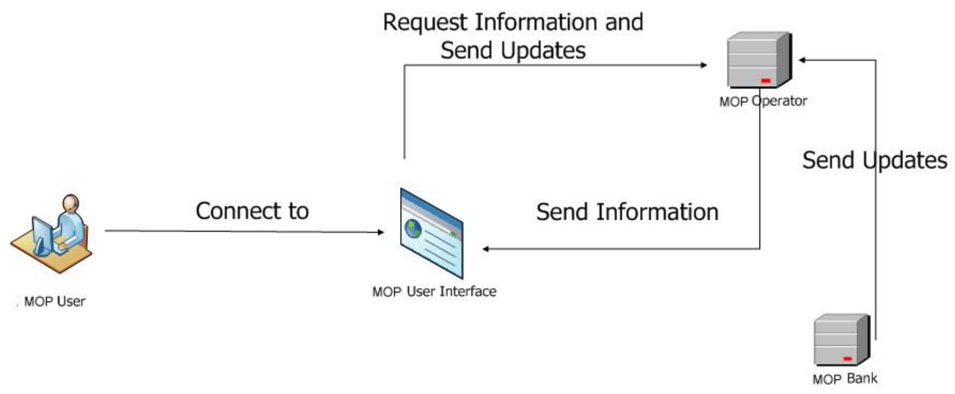


Figure 4.9: User Interface.

4.5.3 MOP Operator Use Case- Caller is not a MOP User

Inbound IVR detects that the caller is not an MOP user, described as shown in Figure 4.10. Here, the caller dials MOP inbound IVR number using his/her mobile or landline number. Once the connection is started, inbound IVR will detect that caller is not a MOP user. Then, inbound IVR will play MOP welcome message followed by the call cost. After that, call will be handled by the non-MOP voice menu. One of the choices is that caller can choose to listen to MOP promotion audio and how to create an MOP account. List of banks registered with MOP can be presented to him/her if needed.

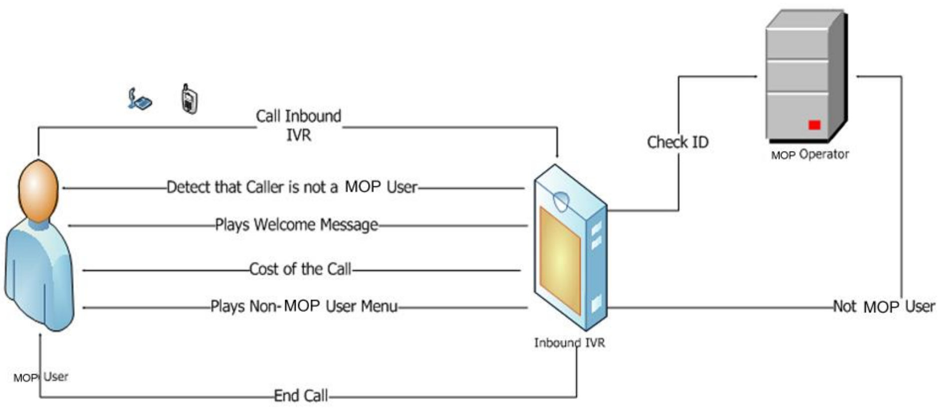


Figure 4.10: Caller is not a MOP User.

4.5.4 MOP Operator Use Case - Caller is a MOP User

After obtaining his/her MOP account and PIN, the user can initiate the process by calling the inbound IVR by following steps as shown in Figure 4.11. MOP user places a connection with the inbound IVR server using his/her mobile or landline number that is registered in the system. Once the connection started, inbound IVR will detect MOP user ID from his/her MSISDN, after a while, a pre-recorded welcome message will be played along with the call cost. A menu will be played for the user to choose the needed service. The user will select “Payment services” to perform the needed transaction. As an automatic mechanism, MOP operator will be aware of the limit per transaction/day/week requested for this user and will recognize if the client has been black listed by his bank or not. MOP operator will also be aware and able to identify if the client’s PIN status is frozen or not. After the MOP user enters MOP receiver number, it checks if MOP receiver account number is correct or not. User has only three trials to re-enter MOP receiver account number, besides, MOP user enters amount to be paid. After that, inbound IVR will play the full transaction details, waiting for the caller to choose one of the following options (Confirm, modify, or cancel). Once the user confirms the transaction, MOP operator will check with MOP bank if there are enough funds in caller bank account or not. If the user has enough funds, inbound IVR will ask him/her to wait for a call-back call in order to proceed with the transaction process. The MOP operator will perform many jobs after the call, which are: determining from MOP account number which bank is responsible to process this request, sending request to the exact MOP bank, updating request to "sent to MOP bank", requesting will be processed in MOP bank. We will discuss this in detail in the next use case. Outputting of the bank server process will be sent to MOP operator through MOP interface at MOP bank and MOP operator will update processed requests to the following status:

A. Success: If the request has been processed successfully.
B. Declined: For any reason if the request couldn't be processed (no fund, wrong MOP user receiver account).
C. Aborted: When the user decided to stop the process.
D. Error: If any error appeared in the system.

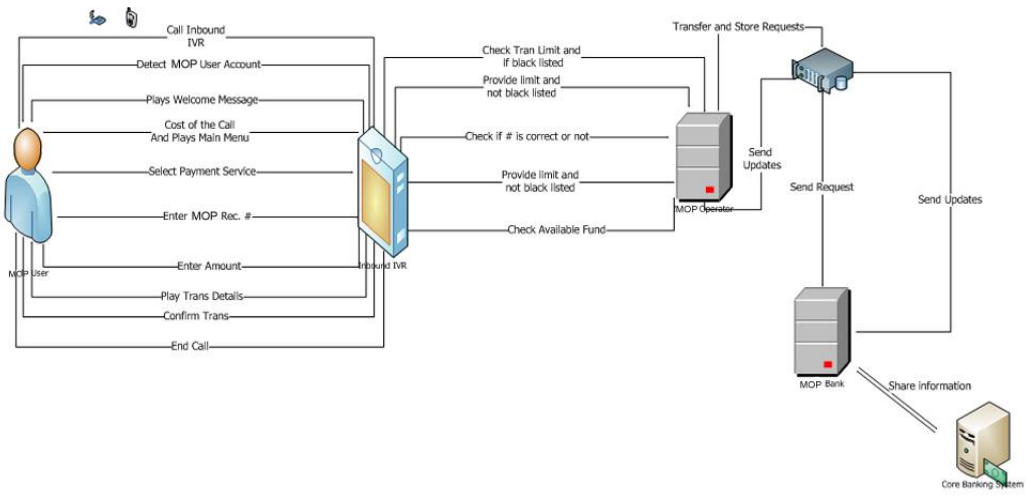


Figure 4.11: Caller is a MOP User.

4.5.5 MOP Bank Use Case

MOP bank segment will handle the processing of open requests received from MOP operator through MOP Bank Interface and forward it to MOP Bank segment. Also, it is responsible to provide MOP operator with request status. Each bank will receive and process only its requests as shown in Figure 4.12, in which MOP operator keeps sending all open requests to MOP banks interface. Once MOP bank segment receives an open unprocessed request, outbound IVR calls back MOP user on his/her mobile number or landline used to open this request. It also shows clearly that outbound IVR plays the pre-recorded welcome message. And a full transaction message will be played again waiting for MOP user to confirm or cancel the request. If MOP user chooses to cancel the request outbound IVR will end the call, cancel the request and archive it. At the time that MOP user confirms the transaction message, outbound IVR will ask the user to enter his/her PIN number. User has only three trials to re-enter MOP PIN number. If the PIN number entered three times incorrectly, the transaction will be cancelled, and the user will be blocked from using MOP. Here user will need to visit his/her bank and request to reactivate his/her PIN number without the need to issue a new one. A status update about the client's PIN will also be forwarded to MOP operator. Next system will inform MOP user that the process is completed successfully and end the call. On the other side, MOP bank will perform many duties like, notifying MOP operator and Guarantee bank about request status (completed successfully), sending SMS to MOP user. MOP operator and Guarantee bank will send request (completed successfully) to MOP receiver bank. In addition to that, MOP receiver bank will send SMS to MOP receiver user and can be followed by a confirmation call. At the end, this way MOP receiver bank will receive two confirmation messages, one from MOP operator and the other one from Guarantee bank or settlement agency. Failing to receive either or one of these messages will cancel the transaction and all parties will be informed.

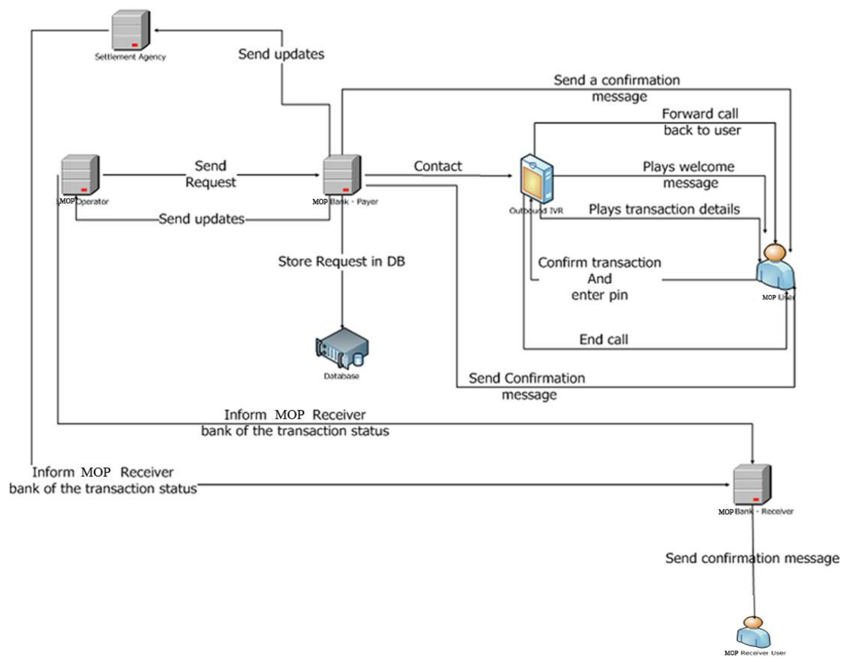

Settlement Agency
Send updates
Send a confirmation message
Forward call back to user
Send Request
Contact
Plays welcome message
MOP Operator
Send updates
MOP Bank - Payer
Outbound IVR
Plays transaction details
Store Request in DB
Confirm transaction And enter pin
MOP User
End call
Database
Send Confirmation message
Inform MOP Receiver bank of the transaction status
Inform MOP Receiver bank of the transaction status
MOP Bank - Receiver
Send confirmation message
MOP Receiver User

Figure 4.12: MOP Bank.

In our solution the life cycle of the transaction will be finished by the final authorization from the payee (sender) bank which will be the last step in the any payment and for any reason (i.e. timeout, connection lost, ...etc.) if the cycle cut anywhere during this cycle the transaction will marked as cancelled and any other effect will be rolled back.

In the banking sector and financially many transactions can be done in the same time and via different media, e.g. cheque withdrawal, loan settlement, mobile banking, ATM, card payment, Internet shopping, mobile payment . etc. and this can affect the balance inquiry for the customer. This is the well-known “concurrent access to shared resources” problem in computer science, in this case the shared resource being the customer account. In order to avoid any inconsistencies due to concurrent access, at the time any transaction is requested and before the final authorization the customer balance will be blocked by the amount of the transaction until the transaction is finished.

4.6 Security Features of MOP

In our novel conceptual framework, there are many layers of security that can increase both customer and merchant satisfaction by eliminating and decreasing foreseeable vulnerabilities, which can be harmful the process life cycle, such as:

1. Unauthorized access to clients' personal information, bank secrets: the critical information for any client are saved in the core banking system and no one can access this information via our system since it will be connected to the banking system through the bank infrastructure which is very highly secure and the connection is only through TCP/IP and 8583 message that request only authentication or general information.
2. Access to sensitive information and system configuration data: to access this setup information the attacker has to access at least two separate locations which are highly secure and need many layers of authentication in addition to many steps and multi users roles to change the settings in different locations; also changing settings doesn't mean in any way that he can do any transactions due to many different access and authentication requirements from different stake holders and different sites. Please see transaction life cycle.
3. Fraud and theft: to do any transaction the attacker needs customer phone device, customer PIN code, call back device, application password, and customer SIM chip to make transaction normally, but if he wants to do it internally from the system he needs to access three separate sites simultaneously and do the transaction in a specific flow and using specific information.

4. Gaining total control over the server, reading arbitrary files: the attacker needs to access the banking system to get customer information, but our system servers contain only general information about customers.
5. Denial of service of accounts: the attacker needs the customer mobile device and SIM card in it just to initiate transaction.
6. Attacks on LAN resources: the financial LAN is a closed private network and if he can access this network, he has to skip all banking security tools and has the server’s passwords and access roles which mean he has access only to the closed network and needs another journey to do any transaction through our system.

In summary, the proposed solution seeks to mitigate security vulnerabilities by adopting different measures. First, the system leverages on information only known to the customer; i.e., the client's PIN code which is initiated by the customer's bank HSM and only requested by the bank through its IVR system call back feature. The second measure involves something possessed by the customer, i.e., mobile phone and mobile SIM card. The third security feature is using several authentication methods and process flow, i.e. PIN code, call back, mobile number and SIM chip. Through this measure, the process is initiated by the operator. Afterward, the transaction is split into 3 sections (sender's bank, operator, and receiver's bank). Next, the sender bank requests for PIN code using the IVR call back to the sender's phone number, who has the SIM card and knows PIN code. Later, the PIN code is verified by the sender bank. Finally, the client's eligibility is determined by their internal bank system and all transaction information flow are highly encrypted and with no involvement of the other parts. The fourth safety measure is that a private network connects all parties with a secure VPN.

4.7 Other Features of MOP

4.7.1 Confidentiality

VPN connection will be established between MOP Operator and MOP bank segment. In addition, transaction requests will be encrypted using high-end encryption algorithms. (E.g. SSL or SHA).

MOP Bank Segment will be responsible to generate MOP PINs; banks will be responsible to provide users with their pins in order to use MOP system.

4.7.2 Availability

The system will be available to the intended audience 24 hours per day, 7 days a week. Since our system is built over the banking infrastructure and as known globally based on world reports the banks build their business continuity plan to be available 24/7, so our system will be as well as the core banking system because they use the same infrastructure.

Our system is not a third party system it will be owned by the bank exactly as any other system the bank acquired i.e. mobile banking, internet banking, so they will be aware off making it work 24/7.

Methods that will be used to ensure system availability:

1. System backup: MOP Solution will have backup procedure performed on daily bases; MOP System hosted at the banks is treated as part of the core banking solutions and is controlled 100% by the operating bank therefore the backup service for MOP solution will be handled by the bank. MOP

operator will be equipped with a backup solution that will be performing daily, weekly and monthly backups.

2. Disaster & Recovery Sites (D&R): as a pre-requisite by the Central Bank each bank should have a D&R site for its core application; and as per Central Bank mobile payment framework, each mobile payment application should have a D&R site. As mentioned before MOP solution that will be hosted at the bank will be part of the bank's core banking application and handled 100% by the operating bank, therefore an image of MOP solution will be hosted at the bank's D&R sites for high availability.

Chapter 5

HIGH-LEVEL SPECIFICATION OF MOP USING ABSTRACT CLASSES AND METHODS

In this chapter, we give an abstract, high level specification of the MOP system using class and method definitions for the main components of MOP. Since MOP components need to talk both to each other, and also to existing banking infrastructures (i.e. each bank's own computer system, and the central bank's computer system), we need APIs for each one of these. The method definitions, both those belonging to the MOP system, as well as those belonging to the participating banks and the central bank, taken as a whole, can be regarded as a protocol with which banks' existing systems can interact with the MOP system, and the MOP components talk to each other.

5.1 User Class

The user class represents the person who sends or receives money. Since the user is interacting with the system, we abstractly represent it as a class, with methods that stand for its actions.

5.1.1 Methods of the User Class

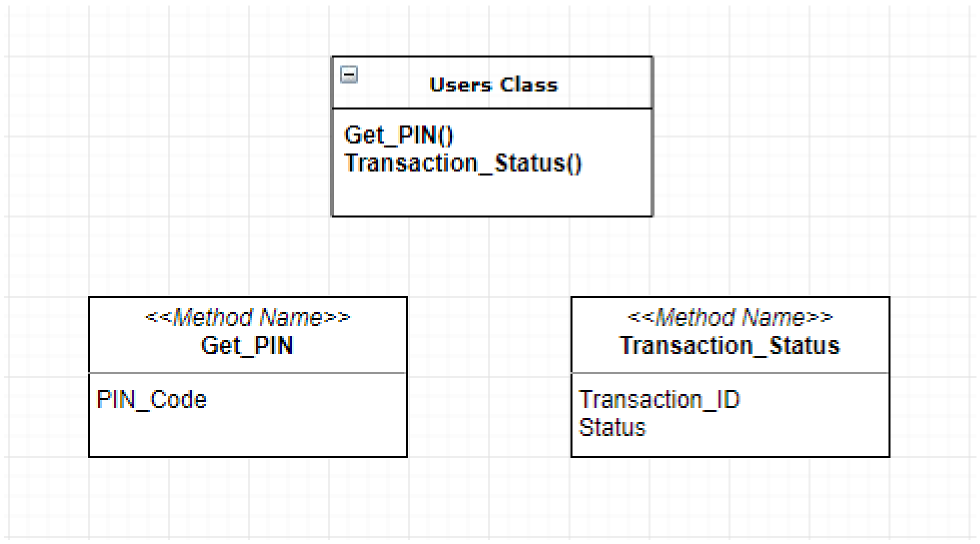


Figure 5. 1: Class and Methods for Users.

- `Get_PIN()`. The customer enters the PIN code when the bank requests it.
- `Transaction_status(Transaction_ID, Status)`. Method used by core bank system to inform the user about the transaction.

5.1.2 Typical Sequence of Actions Using the User Class

The customer requests to create a MOP account from the bank class using `Create_Account(Full_Name,Phone_Number,Bank_Account_ID,Credit, Limit_Transfer )`. This is done only one time, and the call returns a MOP_ID to the user. For every transfer, the user requests a transaction from the MOP operator by using `add_transaction(Sender_phonenumber, Receiver_phonenumber, Amount)` method. After that the core bank requests PIN code from the user by `Get_PIN()` method. The user receives information about the transaction from core bank through a call to its `Transaction_Status(Transaction_ID, Status)` method.

5.2 MOP Operator Class

The MOP operator class represents the entity with which users of the MOP system interact, for example by using a Web browser.

5.2.1 Methods of the MOP Operator Class

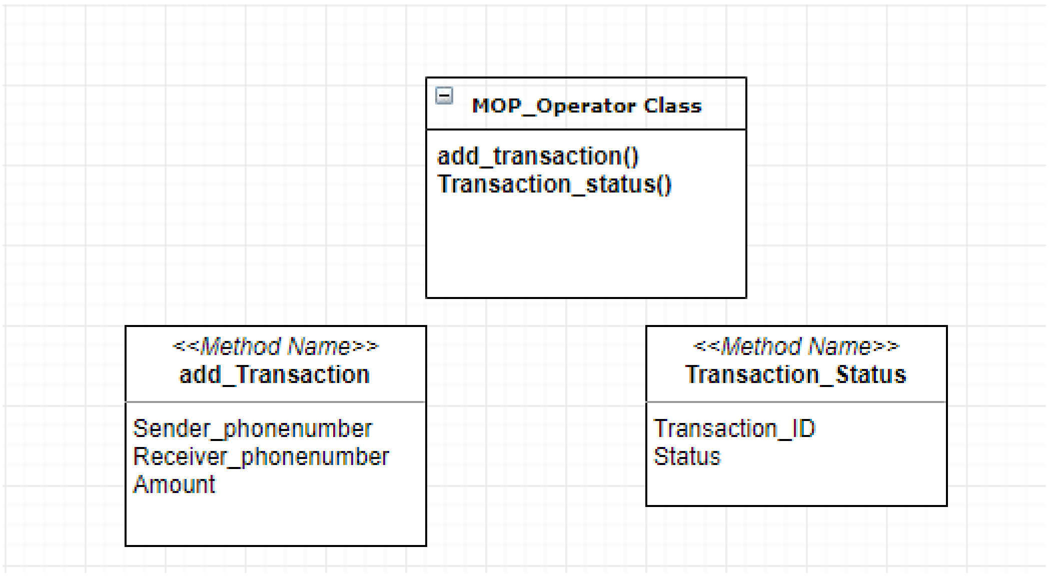


Figure 5.2: Class and Methods of the MOP Operator.

- `add_Transaction(Sender_phonenumber, Receiver_phonenumber, Amount).` Called by the user to initiate a transfer to a receiver. The receiver's phone number and amount are specified.
- `transaction_status(Transaction_ID, Status).` Used to receive information from the Central Bank MOP class about the current status of the transaction.

5.2.2 Typical Sequence of Actions Using the MOP Operator Class

The user makes a request by calling `add_Transaction(Sender_phonenumber, Receiver_phonenumber, Amount)`. The operator checks the phone number and decides it is a MOP client. The operator checks possible restrictions on the customer, such as if it is blacklisted, by calling the `Check_customer(Sender_phonenumber, Amount)` method of the user's bank's infrastructure class. If the user is eligible, then this transaction is sent to all parties, i.e. sender bank, receiver bank, and settlement bank, using their `add_Transaction(Transaction_ID, Sender_phonenumber, Receiver_phonenumber, Amount)` method. The MOP operator requests the account information from the user's bank by calling `getcustomer (Bank_account_ID, Phone_number)`.

5.3 Participating Bank MOP Class

This class represents the MOP component that resides with each participating bank.

5.3.1 Methods of the Participating Bank MOP Class

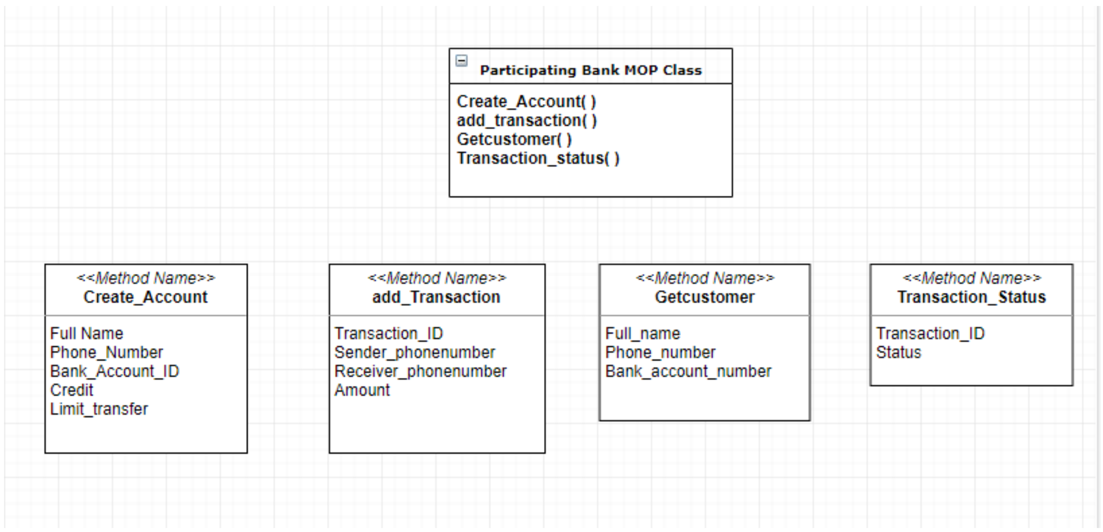


Figure 5. 3: Class and Methods for Participating Bank.

- `Create_Account(Full_Name, Phone_Number, Bank_Account_ID, Credit, Limit_transfer)`. Used to create a MOP account for a customer.
- `add_Transaction(Transaction_ID, Sender_phonenumber, Receiver_phonenumber, Amount)`. Used to send transaction details.

- `Getcustomer (Bank_account_ID, Phone_number)`. Gives all general information about the customer.
- `Transaction_status(Transaction_ID, Status)`. Used to receive information from the Central Bank MOP class about the current status of the transaction.

5.3.2 Typical Sequence of Actions Using the Participating Bank MOP Class

Upon receiving a call to its `add_Transaction(Transaction_ID, Sender_phonenumber, Receiver_phonenumber, Amount)` method, it sends the transaction to the core bank system and provides transaction status information when its `Transaction_Status(Transaction_ID, Status)` method is called.

5.4 Central (Settlement) Bank MOP Class

This class represents the MOP component that resides within the central bank.

5.4.1 Methods of the Central (Settlement) Bank MOP Class

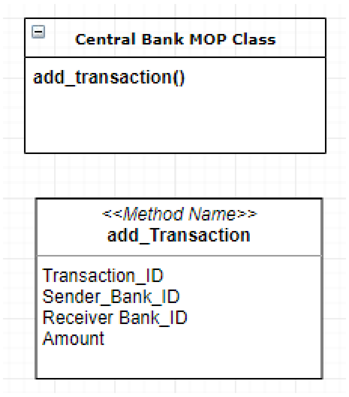


Figure 5. 4: Class and Methods for Central Bank.

- `add_Transaction (Transaction_ID, Sender Bank_ID, Receiver Bank_ID, Amount)`. Used to receive transaction details.

5.4.2 Typical Sequence of Actions Using the Central Bank MOP Class

When central bank MOP class receives the transaction information from MOP Bank using `add_Transaction(Transaction_ID, Sender Bank_ID, Receiver Bank_ID, Amount, currency)`, it sends the transaction status to MOP Operator, sender bank and receiver bank using their `transaction_status (Transaction_ID, Status)` method.

5.5 Participating Bank Infrastructure Class

This class represents the participating bank's own computer system. Its methods allow the MOP system to perform actions in the existing bank computer system.

5.5.1 Methods of the Participating Bank Infrastructure Class

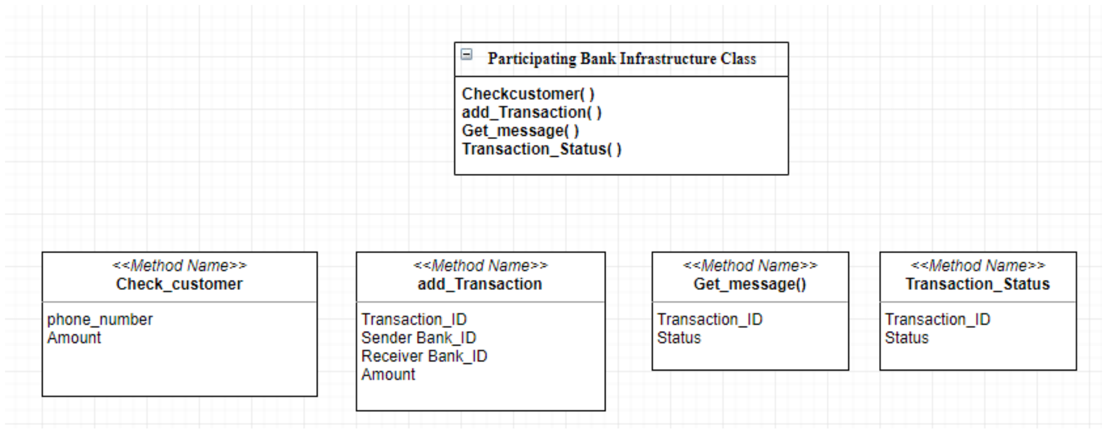


Figure 5.5: Class and Methods for Participating Bank Infrastructure.

- `Checkcustomer (phone_number, Amount)`. Used to check if the user is black listed or not and any other restrictions for this user.
- `add_Transaction (Transaction_ID, Sender Bank_ID, Receiver Bank_ID, Amount)`. Used to send transaction details.
- `Get_message (Transaction_ID, Status)`. Used to provide the transaction status to the customer.

- `Transaction_Status (Transaction_ID, Status)`. Used to change the transaction status.

5.5.2 Typical Sequence of Actions Using the Participating Bank Infrastructure Class

When the core bank system receives the transaction from MOP bank using `add_Transaction (Transaction_ID, Sender_Bank_Account, Receiver_Bank_Account, Amount)`, it creates new call to the customer and requests PIN code using `Get_PIN()`. When the transaction is completed and the transaction status is changed, the bank informs the user about the transaction by calling its `Transaction_Status (Transaction_ID, Status)` method (simulating an SMS message).

5.6 Central Bank Infrastructure Class

This class represents the central bank's own computer system. Its methods allow the MOP system to perform actions in the existing central bank computer system.

5.6.1 Methods of the Central Bank Infrastructure Class

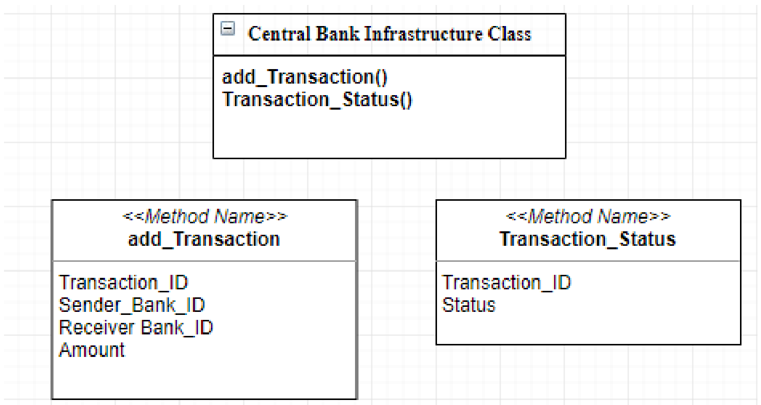


Figure 5.6: Class and Methods for Central Bank Infrastructure.

- `add_Transaction (Transaction_ID, Sender Bank_ID, Receiver Bank_ID, Amount)`. Used to receive transaction details.

- `transaction_status(Transaction_ID, Status)`. Changes the transaction status and sends the updated status to all involved parties, i.e. the MOP operator, sender MOP bank, and receiver MOP bank.

5.6.2 Typical Sequence of Actions Using the Central Bank Infrastructure Class

The central core bank requests transaction details from central MOP bank using `add_Transaction (Transaction_ID, Sender Bank_ID, Receiver Bank_ID, Amount)`, and then changes the transaction status using `Transaction_Status (Transaction_ID, Status)`.

5.7 Message Traffic among Components of MOP

Figure 5.7 depicts the sequence diagram of a typical transaction, and Figure 5.8 gives us the communication pathways among classes and methods of the MOP System. We can verbally describe what happens in the lifecycle of a transaction as follows: The user requests an MOP account from MOP bank, and then a transaction from the MOP operator. The MOP operator sends an eligibility request to the core bank to check any restrictions on the user and if it is blacklisted or not. Then the MOP bank requests transaction details from the MOP operator, the core bank requests financial transaction details from the MOP bank, and the core bank requests verification code from the user by PIN code. The central MOP bank requests financial transaction details from MOP bank, and the central core bank requests financial transactions from the central MOP bank. The last process is requesting the transaction status from the central core bank, and the core bank sends a message to the user to inform him/her of the transaction done.

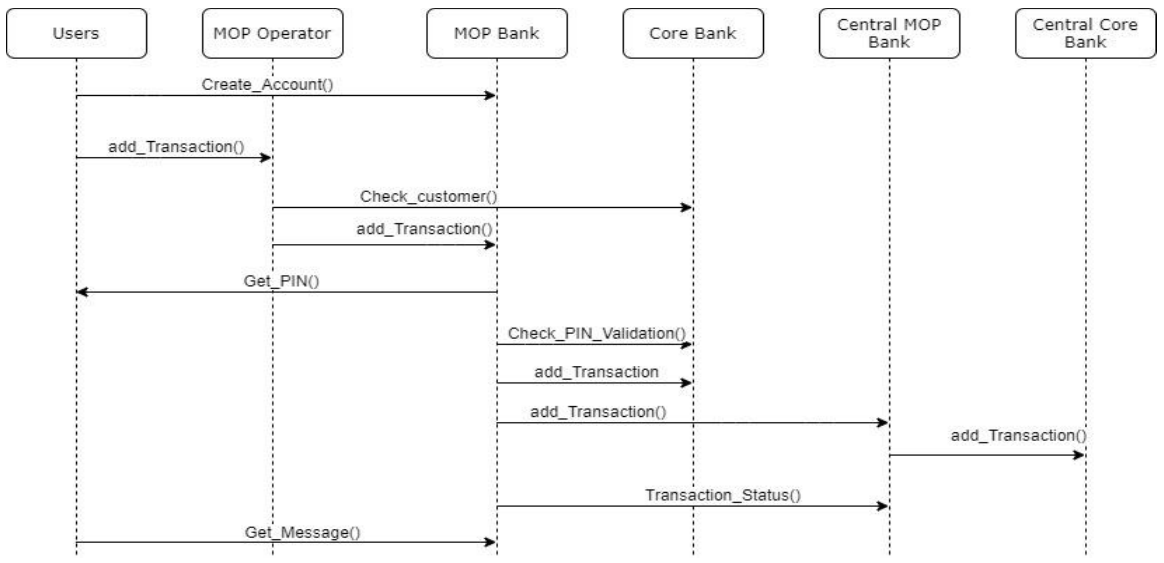


Figure 5.7: Sequence Diagram.

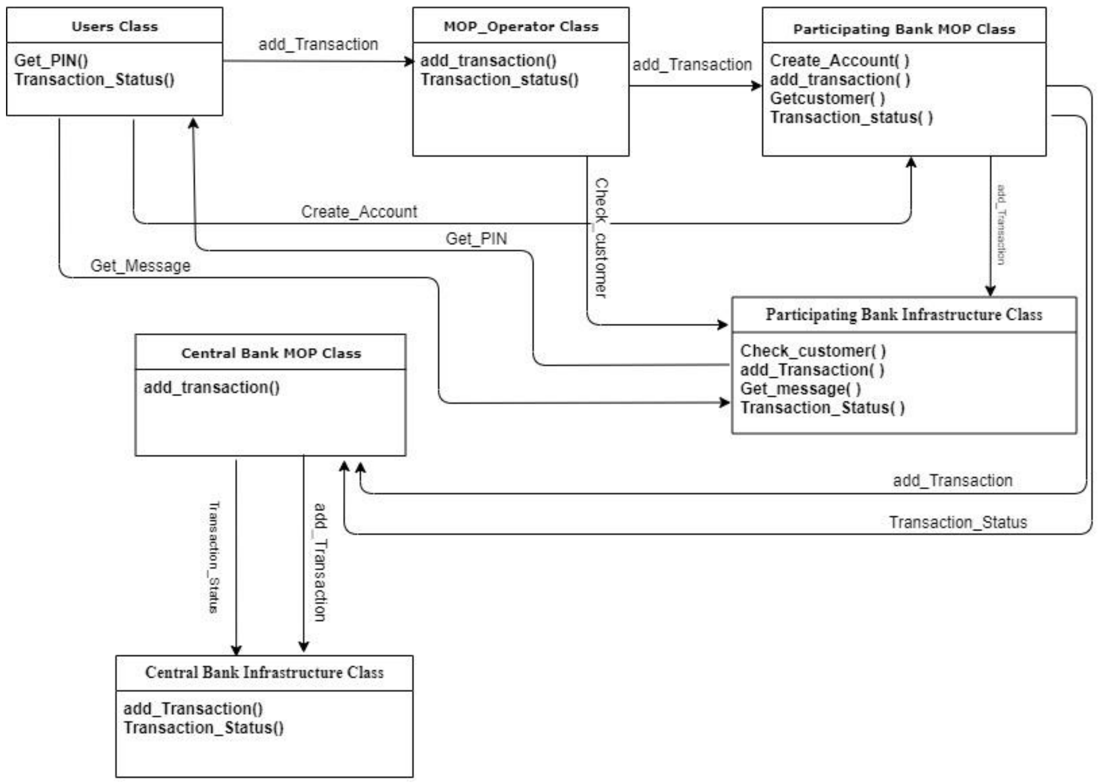


Figure 5.8: Communication Pathways among Classes and Methods of the MOP System.

Chapter 6

IMPLEMENTATION OF MOP

In this chapter we give implementation details of our MOP system prototype, which has both hardware and software components.

6.1 Hardware and Software Setup

Our proposed system was implemented on a virtual machine. Given the diversity of user devices, the system has web and mobile versions (iOS), thus allowing users to make payments through calls, SMSs, USSD, Web browsers, or mobile applications. We first built the network architecture as described by Figure 6.1, which was later divided into three subsystems. The first entailed a MOP operator that contained a data center (storage) SAN switch, two login servers, two IVR servers, two authentication servers to load balance, and two database servers. The second subsystem entailed MOP banks (client), for which every client included a database and IVR server for a callback, which is connected using the service gateway SRX 550 (the third subsystem). Nations were connected to each other through the service gateway international private leased circuit that includes a point to point private line to enable communication among corporations.

The MOP System runs on the Linux platform, which provides high availability and enhanced security. It can be installed on any certified platform of Linux such as Debian, Ubuntu, Red Hat, among others.

Several software packages will be required for the installation and operation of the MOP System. First, the latest version of the Debian Linux can be downloaded from [42]. Secondly, Asterisk, an open free platform for building communication systems that acts as the base of MOP System can be obtained from [43]. Apart from the Asterisk setup, an additional two downloads are needed from Asterisk website; that is, the DAHDI Library for communication interfaces and LIBPRI Library to encapsulate the protocols used to communicate over ISDN Primary Rate Interfaces. Thirdly, the MYSQL database will serve as the project's engine. Finally, users should download the Apache Web server that will be used to manage the system web server. Table 6.1 depicts the technologies used in the implementation of the MOP system.

Table 6.1: Technologies used in the implementation of the MOP system.

Technology	**Tool/Product Used**
Open system	UNIX or Linux
Platform	Java Platform (J2EE Framework)
Application Server	IBM WebSphere
Web Services	J2EE and SOAP
Database	Oracle (latest version)
Interface (TCP/IP based):	
1. Web Services	HTTPS
2. XML	eXtensible Markup Language
3. Standard API	Web API
4. Queue	MSMQ, MQ-Series, JMS Compliant
Clustering	HA Cluster, MySQL Cluster

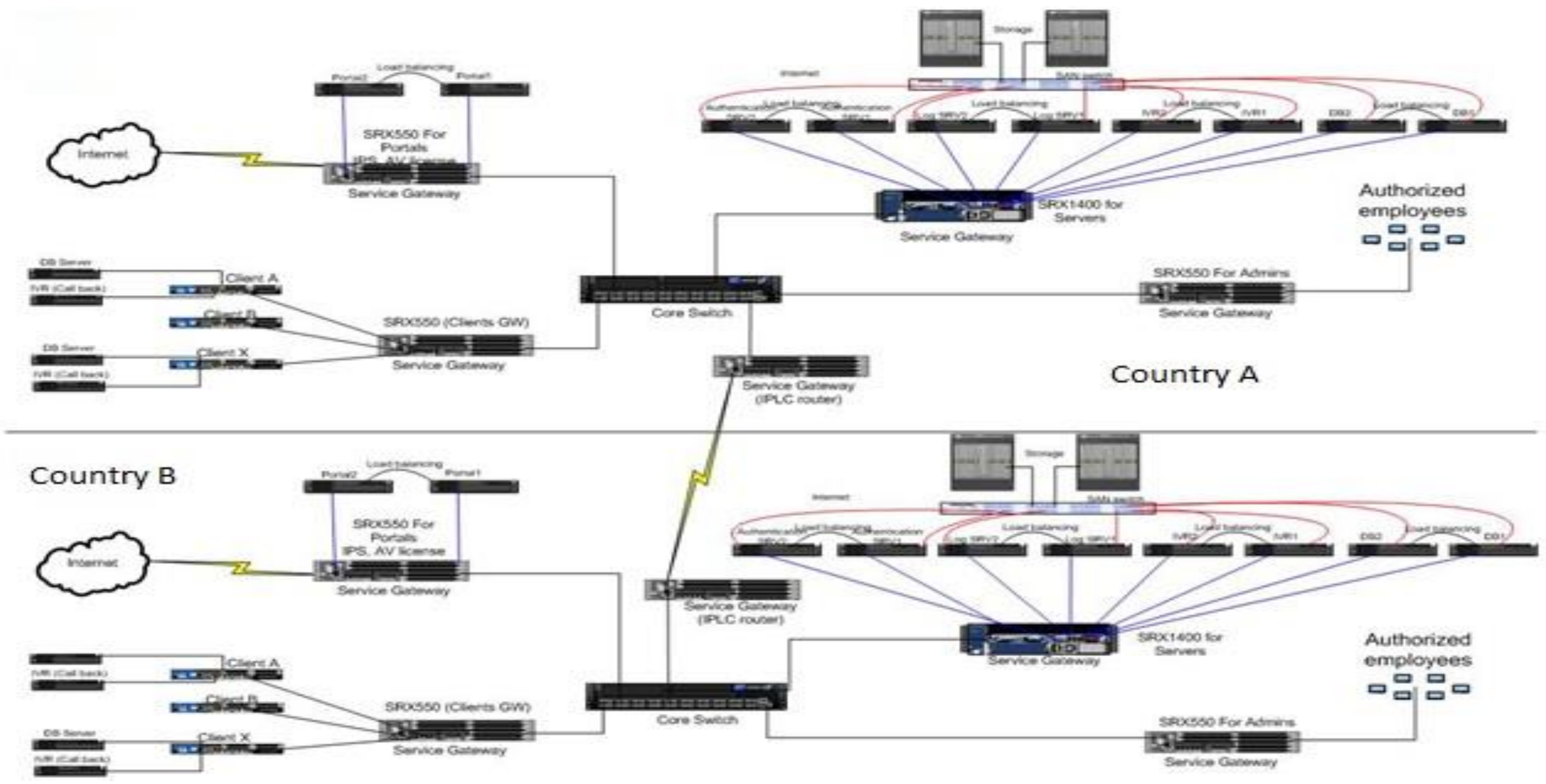


Figure 6.1: Network Architecture.

6.2 High-Level Components of Our Implementation

In our proposed solution we have 2 parts: (i) the operator (SW, HW) that can be installed and customized at the central bank or any other bank that can perform the functions of a central bank; this part will be important for banks' reconciliation and connection to all other banks that are involved in this service, and (ii) the client (SW, HW) which will be installed and customized at the banks (service members) by the bank team. The client will be integrated with the banking core system using special APIs.

6.3 MOP Operator

6.3.1 MOP Operator Web Pages

Administration access page is the page that allows the administrator to create and manage other system administrators with fewer privileges. It also allows them to access all modules in the system as; Agents, Manage banks, Services, Manage clients, Services Provider, Service Type, Manage Country, Manage Currency, Manage Prompts, Manage Setting and Transaction Type. All web pages in the MOP operator are explained in Table 6.2.

Table 6.2: Web pages in MOP Operator.

Web Page Name	**Description**
Agent.php	This module relates to MOP Agents as; Electricity Company, Telecom Company, Water Company and any other company which want to be a customer for MOP.
Bank.php	This module is used when you want to add new banks and choose a correspondent bank between them, it is also used to edit a certain bank or if you want to list existing banks.

Blacklist.php	This module is used when you want to add blacklisted customers.
Client.php	This allows showing all MOP Clients in different banks, it also gives you the ability to search Clients using some options and you also can filter clients from a specified bank.
Country.php	The first thing to do when you create a new operator in a new country is to Create Country for that country for every country which has a MOP Operator, then you have to choose which one of them is your local operator to detect remittance cases and how to deal with these cases.
Currency.php	List and edit currencies in this option.
Prompt.php	This module is responsible for adding or listing sound files for the user action on the Interaction Voice Response (IVR) system.
Setting.php	This module allows system administrators to list and change settings for the system.
Transactiontype.php	This module for commission and pricing for every transaction type and you also can add new transaction types.
Clientcontroller.php	Is the most important page because each function is called from this page. The list of functions is explained in Table 6.3.

6.3.2 Functions Called from the Web Pages

Functions listed in Table 6.3 are called from the pages are given in Table 6.2 with parameters that depend on the page.

Table 6.3: Functions used by PHP pages (MOP Operator)

Function Name	Description
function getById($id)	Get agent by given Id.
function getSubAgents($ Id)	Get sub-agents of given agent.
function getSubCategories($bankId)	Get sub-banks of given bank.
function add($)	Add new agent.
function update($)	Update agent.
function delete($)	Delete agent.
function getTree( )	Build agent tree with depth for each item.
function getParents($ Id)	Get parent agents (From root to parent one).
function getTranslatable($lang)	Get translate table items which haven't been translated of the default language.
function getSource($)	Get translation item which was translated to given agent.

Table 6.4: Functions used by Clientcontroller.php

Function name	Description
function init( )	Connection between bank to server to add the client details in the server.
function checkcallAction( )	Check if this phone number is MOP user or not.

function getRecordeByPhone($phone)	Get all user information by phone number.
function check($where)	Check if the user is blacklisted or not.
function addingtoinsertdbcAction()	Add the client in the MOP server (data coming from MOP bank).
function addphonesAction($data, $RecordId = null, $action = null)	Add phones numbers to phone table.
function addaccountsnumAction($data, $RecordId = null, $action = null)	Insert client account information to client_account table.
function transactionAction($pdf=null)	Display report for all transaction as pdf file.
function transactiondetailsAction()	Display the transaction details.
function addtooperatorAction()	Add the record in the operator after a successful transaction.
function connresponsebankAction($ip, $mymo, $client_id)	Respond to the bank and send last used MOP.
function array2encryptAction ($array)	Encrypt array of data.
function getTempTable()	When transaction is created then the information of this transaction is inserted in temp table.
function	Select client information and issuer_bank_ip from client,

getClient($MOP_id)	bank table.
function addcdrclientwaiting($phone,$MOPid",4,$type,$method)	Insert call info to cdr table.
function trxComm($trx_type, $issuer_bank)	Get commission info to transaction_type, pricing table.
function changeStauts($value)	Change status for cdr row depending on value that retrieve from bank.
function insertCdr($cdr, $cdr_comm, $cdr_comm_updated)	Insert call information to cdr, cdr_commission, cdr_commission_updated table.
function smsTransaction()	Get all rows from received_sms table where status ='0'.
function excuteurlAction($cdr_url, $encry_data)	Send post request to autoaddcdr, waitingcallback function on operator.
function checkTransactionStatus($status,$id)	Check cdr where status = 2 to insert in the cdr on the bank.
function changestatusAction()	Get value from bank to check if transaction need to make any change.
function selectcdrclientwaiting(null, null, "src_MOPid = '$MOPid' AND id = '$id' AND (status = '2' OR status = '3') ")	Get transaction from cdr table depend on (src_MOPid, status=2 OR 3).

6.3.3 Detailed Description of MOP Operator Functions and Methods

In this section, we explain the functions that are used by the MOP operator. The notation we used in the figures are as follows: each circle means function name; the arrows indicate that the function calls other functions from possibly different pages and rectangle gives us where this function or database table exists, i.e. the path for this function or table.

6.3.3.1 `checkcallAction` Function

In the first step we need to check if this phone number is MOP user or not, so we use `checkcallAction` function. To get all user information by phone number, we use `getRecordeByPhone($phone)`. `settingAction($setting_id, $field_id)` function calls `getSetting` function from Dao->pdo->mysql->Client.php. We check if the user is blacklisted or not by calling `check($where)` from blacklist modules with path blacklist->dao->pdo->Blacklist.php.

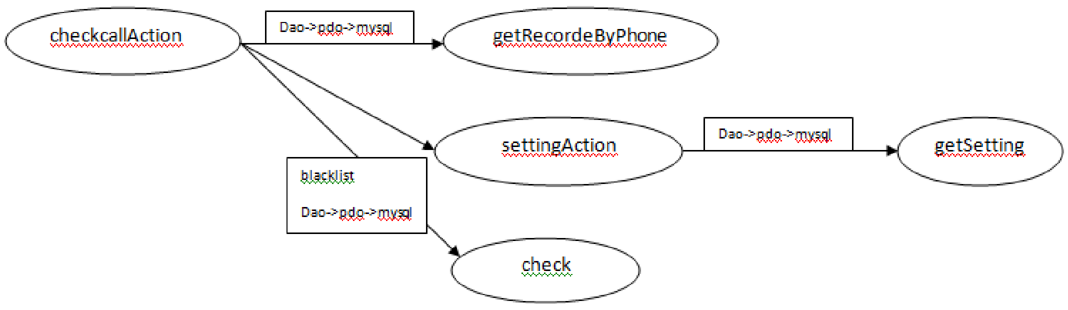


Figure 6.2: CheckcallAction Function.

6.3.3.2 `addingtoinsertdbcAction` Function

To add the client into the MOP server data from MOP bank we use `addingtoinsertdbcAction` function. `keyAction($id)` function calls `getKey($id)` function from Dao->pdo->mysql->Client.php to encrypt or decrypt from validation_key table. We decrypt data that comes from MOP bank depending on the key by using `decryptAction($key,$crypttext)` function. `getBankId($value)` is used to select bank ID depending on bank IP which is written in `bank config` from the bank table. To generate random PIN number for a phone number, we use `generateNumberAction($length)` function. To insert client account information to `client_account` table, we use `addaccountsnumAction($data, $RecordId = null, $action = null)` function. We call `add($answer)` function from core modules Dao->pdo->mysql-> Translation.php to insert information to `core_translation` table.

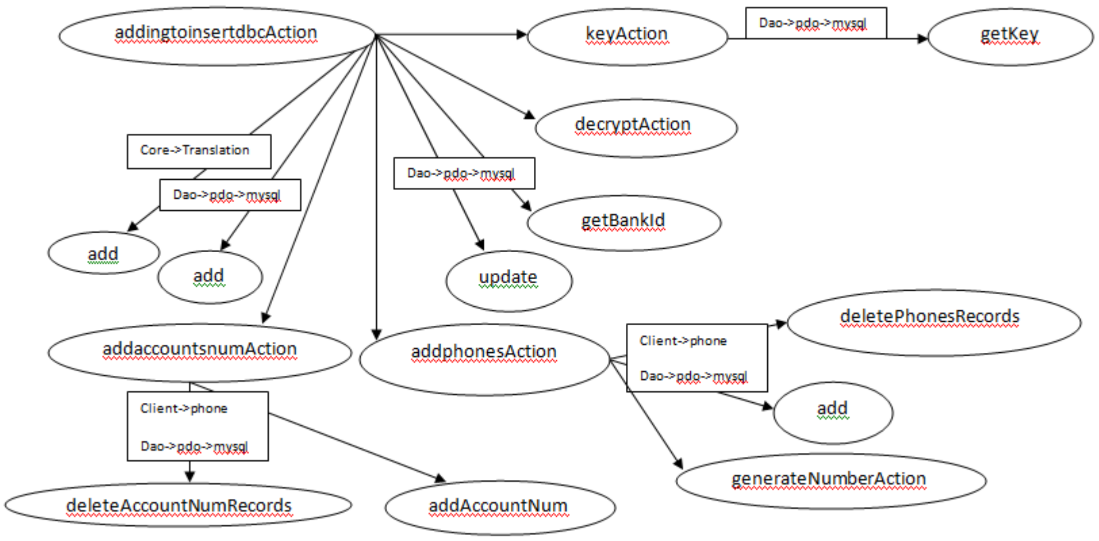


Figure 6.3: AddingtoinsertdbcAction Method

6.3.3.3 `addtooperatorAction` Function

To add the record in the operator after a successful transaction, we use `addtooperatorAction` function. It calls banks function from Dao->pdo->mysql->Client.php to get bank_id from bank where correspondent_bank='1' by using `banks($data)` function. It sends data to bank (index.php/add-client-waiting) to make a call-back for the client using `connectiontoserverAction($data, $ip,$return=null)` function. It calls `changeStauts($value)` function from Dao->pdo->mysql->Client.php to change status for cdr row depending on the value that is retrieved from the bank. After that it calls `getClient($MOP_id)` function from Dao->pdo->mysql->Client.php to get client information. Then, `insertCdr($cdr, $cdr_comm, $cdr_comm_updated)` function is called from Dao->pdo->mysql-> Log.php to insert call information to cdr, cdr_commission, cdr_commission_updated table.

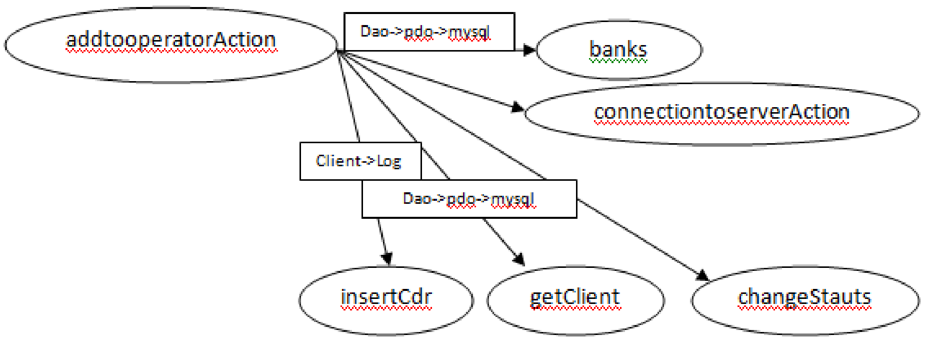


Figure 6.4: AddtooperatorAction Function.

6.3.3.4 `array2encryptAction` Function

To encrypt array of data we use `array2encryptAction ($array)`. Then we call `getKey ($id)` function from Dao->pdo->mysql->Client.php to get encrypt/decrypt key using `keyAction ($id)` from validation_key table.

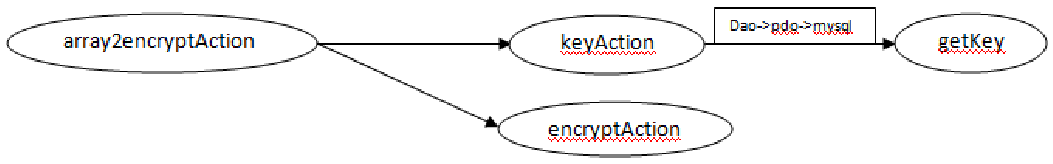


Figure 6.5: Array2encryptAction Function.

6.3.3.5 `autoaddcdraction` Function

When a transaction is created the information of this transaction is inserted into temp table. To do that, we use `getTempTable` function. Call `checkBankes($issuer_MOPid, null, 'issuer')` function from Dao->pdo->mysql->Client.php to get the sender and receiver banks. Call `getClient($MOP_id)` function from Dao->pdo->mysql->Client.php to select client information and issuer_bank_ip from client and bank tables. Call `addcdrclientwaiting($phone,$MOPid'',4,$type,$method)` function from Dao->pdo->mysql->Client.php to insert call information to cdr table. Call `trxComm($trx_type, $issuer_bank)` function from Dao->pdo->mysql->Client.php to get commission information for transaction_type from pricing table. Call `addTrxComm($data)` function from Dao->pdo->mysql->Client.php. Then inside this function call `getDefaultOperator` to insert commission information to cdr_commission table.

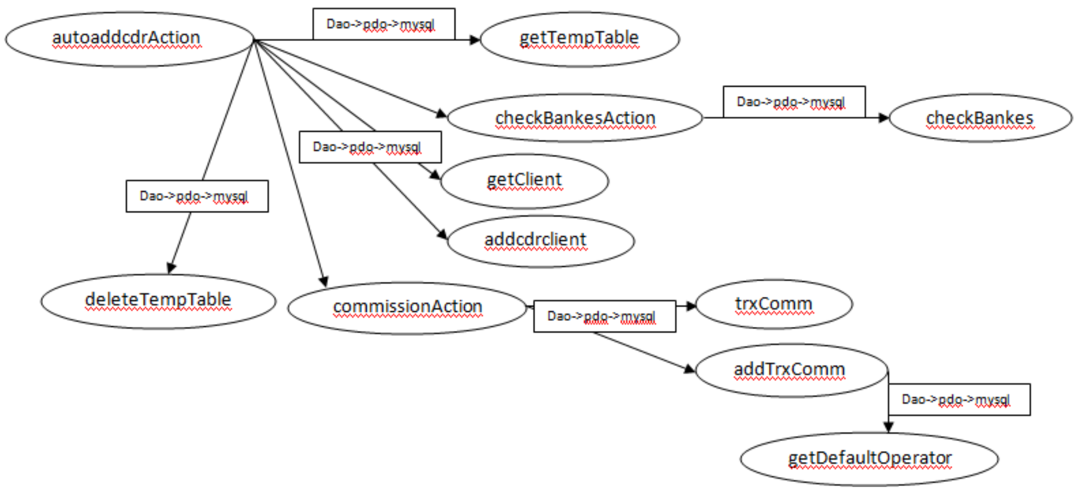


Figure 6.6: AutoaddcdrAction Function.

6.3.3.6 `callbackstatusAction` Function

Call `smsTransaction` function from Dao->pdo->mysql->Client.php to get all rows from received_sms table where status ='0'. Call `insertTemp` function from Dao->pdo->mysql->Client.php to insert SMS information to temp table. `excuteurlAction($cdr_url, $encry_data)` send post request autoaddcdr, waitingcallback function to operator. Call `checkTransactionStatus($status,$id)` function from Dao->pdo->mysql->Client.php from cdr table to check cdr where status = 2 to insert in the cdr on the bank.

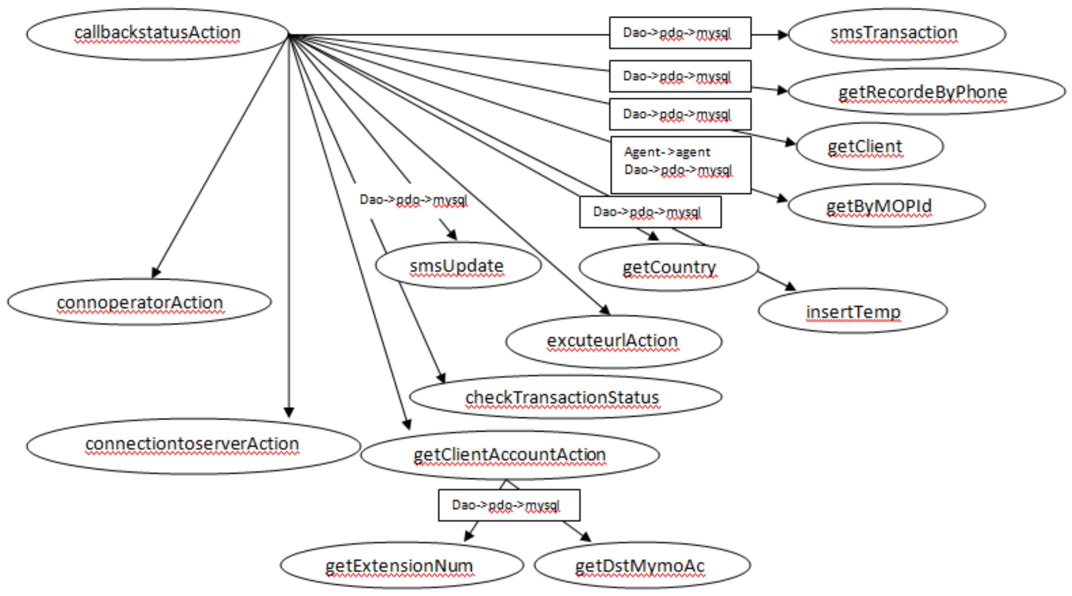


Figure 6.7: CallbackstatusAction Function.

6.3.3.7 `changestatusAction` Function

`changestatusAction` to get value from bank to check if transaction needs to make any changes. Call `selectcdrclientwaiting(null, null, "src_MOPid = '$MOPid' AND id = '$id' AND (status = '2' OR status = '3') ")` function from Dao->pdo->mysql->Client.php to get transaction from cdr table depending on (src_MOPid, status=2 OR 3). Call `getById` function from agent modules Dao->pdo->mysql-> Agent.php to get agent information depending on agent_id from agent table. Call `updateAgent` function from agent modules Dao->pdo->mysql-> Agent.php to update agent information depending on agent_id from agent_details table. Create connection with any IP address you selected and send POST method by calling `directconnectionAction($agentDetails->link,$data)` function. Call `changeStauts` function from Dao->pdo->mysql->Client.php to change status for cdr row depending on the value is retrieved from the bank.

connoperatorAction($data, $ip) send data to the bank using addtooperator function.

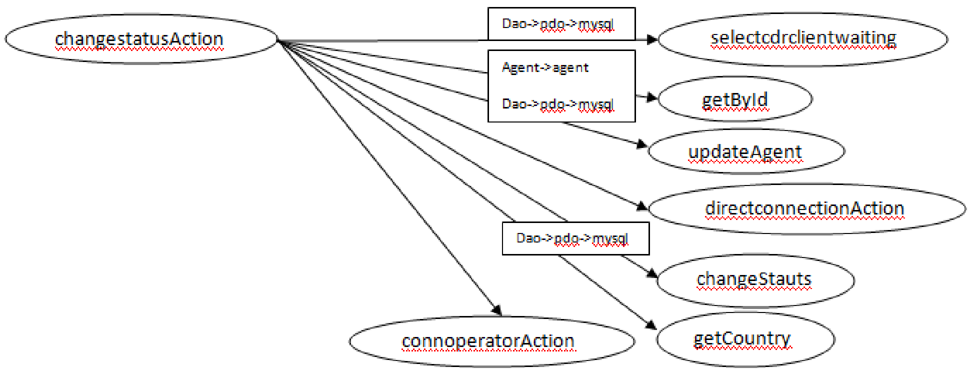


Figure 6.8: ChangestatusAction Function.

6.3.3.8 sendtrxreciverbankAction Function

Send transaction to the bank by using sendtrxreciverbankAction function. Call getClient($MOP_id) function from Dao->pdo->mysql->Client.php to get client information from client table JOIN bank table on bank_id. Call getRecorde($MOP_id, $phone_id) function from client modules Dao->pdo->mysql-> Log.php to get transaction information depending on id from cdr table JOIN cdr_commission, cdr_commission_updated table ON cdr_id. Call getCountry function from Dao->pdo->mysql->Client.php to get currency_code, currency_number from country table JOIN currency on currency_id. connoperatorAction to send data to bank by addtooperator function. connectiontoserverAction is used to send data (post method) to bank (index.php/add-client-waiting) to make call-back for client.

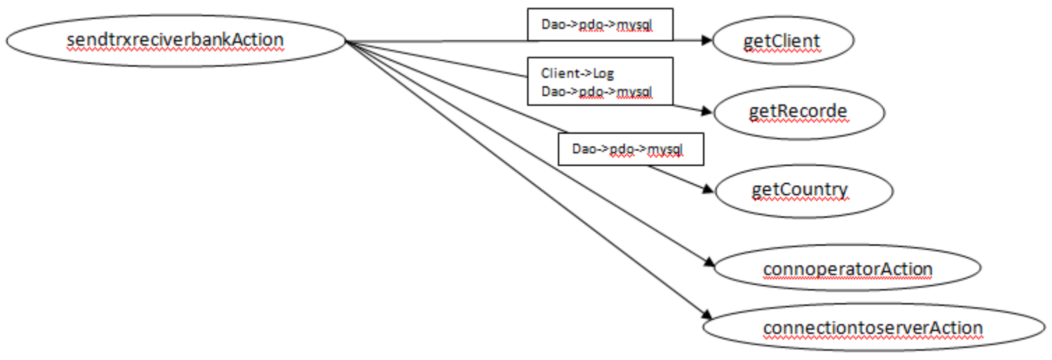


Figure 6.9: SendtrxreceiverbankAction Function.

6.3.3.9 `checktransactionAction` Function

Check transaction from IVR system by using `checktransactionAction( )`. Then call getRecorde function from Dao->pdo->mysql->Client.php to get client information from client table JOIN client_phone table depending on row MOP_id, phone using `getRecorde($MOP_id, $phone_id)`. Call getTransOnMinDate function from Dao->pdo->mysql->Client.php to get transaction information from cdr table JOIN cdr_commission table depending on row MOP_id, phone using `getTransOnMinDate ('ASC', $phone_id, $MOP_id)`.

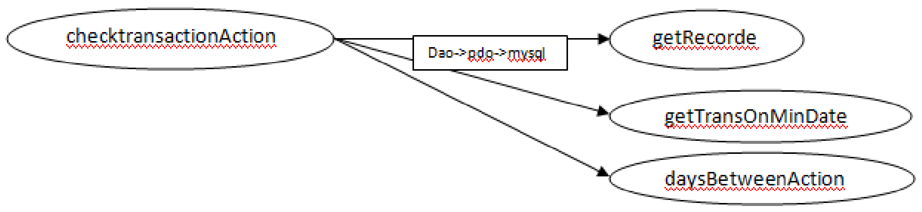


Figure 6.10: ChecktransactionAction Function.

6.4 MOP Bank

6.4.1 Mop Bank Web Pages

The administration access page is the page that allows the administrator to create and manage other system administrators with fewer privileges. It also allows them to access all modules in the system as Manage Clients, Manage Integrations, Manage ISO, Manage Prompts and Manage Settings as explained in Table 6.5.

Table 6.5: Web pages in MOP Bank.

Web Page Name	**Description**
Client.php	This module is used to manage users which allow you to make and list users and groups for MOP System. The important functions are explained in Table 6.6.
Integration.php	This module is responsible about system and database integrations, which is used to make connection between two different databases system "as between MYSQL and ORACLE".
Iso.php	This module is responsible for adding, editing and listing ISO which is used for connecting MOP System with Banking System.
Prompt.php	This module is responsible for adding or listing sound files for the user action on the Interaction Voice Response (IVR) system.
Setting.php	This module allows you to list available settings; you also can edit any setting.

6.4.2 Mop Bank Functions

In all the pages listed in Table 6.5, we use the functions that are listed in Table 6.6 with different parameters that depend on the page.

Table 6. 6: Functions used by PHP pages (MOP Bank)

Function Name	**Description**
function getById($id)	Get page by given Id.
function add($)	Add new page.
function update($)	Update page.
function updateOrder($)	Update page order.
function delete($)	Delete page.
function getTree( )	Build pages tree with depth for each item.
function getTranslatable($lang)	Get items from translate table which haven't been translated from the default language.
function getSource($product)	Get translation item which was translated to given page.
function saveGallery($tableName, $data)	Used to save all action done in the bank as a backup.
function getGalleryById($tableName,$className,$id)	Used to save all actions for each user as a backup.

Table 6. 7: Functions used by Clientcontroller.php

Function Name	Description
function addclientToAnotherBankAction( )	Add client transaction information to bank.
function addclientwailtingAction()	Used to insert transaction information to cdr, cdr_commission, cdr_commission_updated tables.
function createClientFileAction($cdr_id, $callbackNum, $method= null,$authentication=null)	Create file in the bank root that has call back number, file name of cdr ID for the client.
function correspondentBank()	Used to select, insert, delete bank information from correspondent_bank_ip table depending on operation type.
function getSetting ($setting_id, $field_id)	Used to get card information from setting table depending on setting_id, field_id.
function selectcdrclientwaiting(null,"status = '2' ", null,'id')	Used to get transaction information from cdr table.
function getCallBackNum (id)	Used to get callback number with call checkMOPidExistInTheClient.
function getClient (MOPid)	Used to get client information from client table depending on MOP_id.
function connectiontoserverAction ($MOPid, $status, $id, $bank_ip)	Connect to another bank and send transaction record with commission information.
function changeStauts($id, 6, $sender_bank_ip, $country_id)	Used to change status for cdr row depending on value that is retrieved from the bank.
function getRecorde(dst_MOPid)	Used to get client information from client table to JOIN client_phone table depending on row MOP_id.
function addnewlimitAction()	Used to add new request limit from portal.

function decryptAction ($key, $post[0])	Used to decrypt data that comes from MOP operator, portal depending on key.
function tempClient ($new, $decrypt[0]['client_id'], 'insert')	Used to get or insert or delete client request information from temp_client table depending on client_id.
function getAuthintication($cdrID)	Used to get authentication information from cdr_authentication_level table depending on cdr_id.
function ivrConnectionAction ($url, $postdata).	Send call back information to IVR system.
function autochangestatusAction()	Used to check if there are any transactions without a callback.
function checkMOPidExistInTheClient(MOP_id)	Used to check if MOP ID exists or not from client table depending on MOP_id.
function getCallBack($phone, client_id)	Used to get a call back information from callback_phone table depending on phone, client_id.
function cbAction ($data, $ip)	Send transaction information to correspondent bank.
function changestatusAction()	Change cdr row status.
function checkpinAction()	Check if PIN number is valid or not.
function getKey($id)	Used to get key information from validation_key table.
function hsmAction($data, 'check')	Send post method to HSM to check pin number.
function feesAction()	Used to insert fees information to bank.
function fees ($row,'insert')	Used to get or insert fees information from/to client_fees table depending on id.

6.4.3 MOP Bank Functions and Methods

6.4.3.1 `addclientToAnotherBankAction` Function

Add client transaction information to bank using `addclientToAnotherBankAction( )`. Call `addcdrclientwaiting` function from Dao->pdo->mysql->Client.php to insert transaction information to cdr, cdr_commission, cdr_commission_updated tables using `addcdrclientwaiting($dataForm)`. Call `correspondentBank` function from Dao->pdo->mysql->Client.php to select, insert, delete bank information from correspondent_bank_ip table depending on operation type.

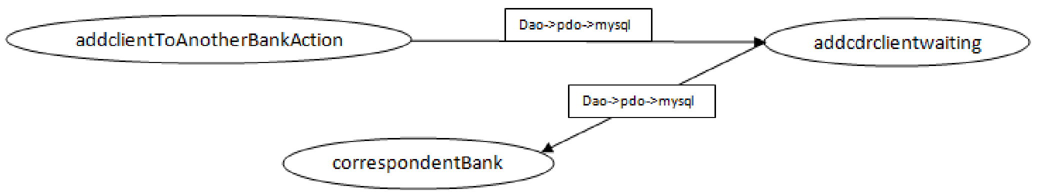


Figure 6.11: AddclienttoAnotherBankAction Function.

6.4.3.2 `addclientwaitingAction` Function

Check client information, then add client to cdr table by using `addclientwailtingAction()`. Call settingAction function `smsipAction()`. Call `settingAction` function from Dao->pdo->mysql->Client.php to get card information from setting table depending on setting_id, field_id using `getSetting ($setting_id, $field_id)`. Call `selectcdrclientwaiting(null,"status = '2' ", null,'id')` function from Dao->pdo->mysql->Client.php to get transaction information from cdr table. Call `addcdrclientwaiting($dataForm)` function from Dao->pdo->mysql->Client.php to insert transaction information to cdr, cdr_commission, cdr_commission_updated tables. Call getCallBackNum ($dataForm->id) function from Dao->pdo->mysql->Client.php to get callback number calls to checkMOPidExistInTheClient, and getCallBack functions. Call getClient ($dataForm->src_MOPid) function from Dao->pdo->mysql->Client.php to get client information from client table depending on MOP_id. Call getCallBackPhone ($client->client_id) function from Dao->pdo->mysql->phone.php to get callback information from callback_phone table depending on client_id. Send data to bank (index.php/add-client-waiting/) to make a call-back for client using connectiontoserverAction ($MOPid, $status, $id). Call changeStauts($dataForm->id, 6, $dataForm->sender_bank_ip, $dataForm->country_id) function from Dao->pdo->mysql->Client.php to change status for cdr row depending on value that is retrieved from the bank. Call getRecorde(dst_MOPid) function from Dao->pdo->mysql->Client.php to get client information from client table JOIN client_phone table depending on row MOP_id, phone.

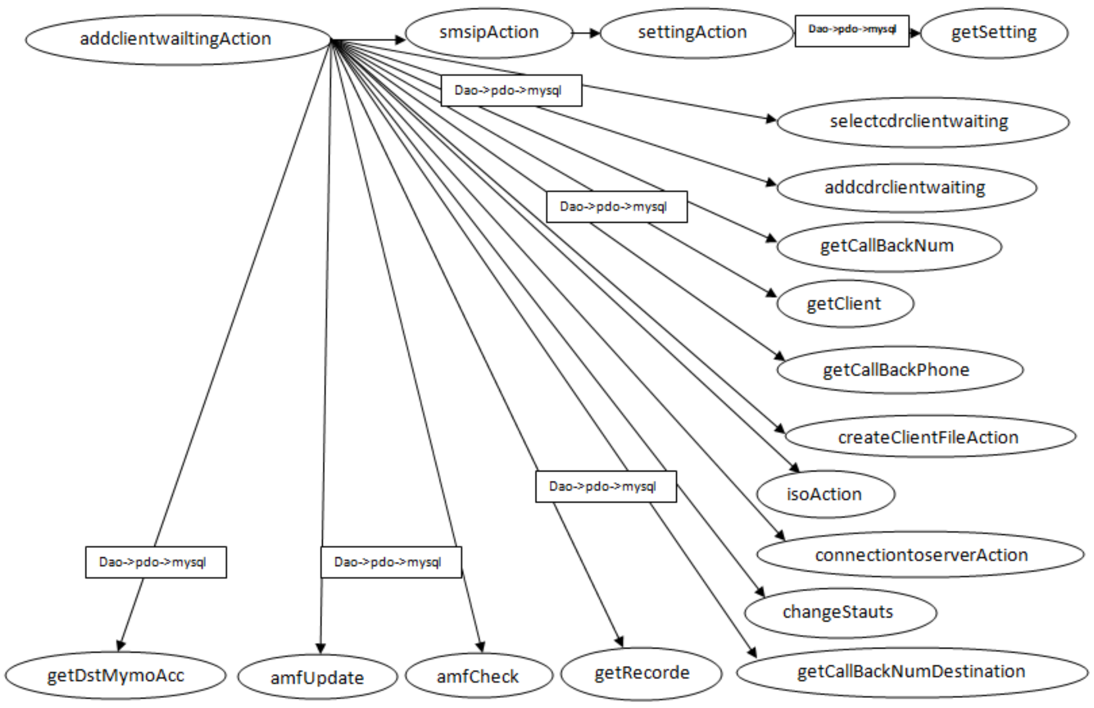


Figure 6.12: AddclientwaitingAction Function.

6.4.3.3 `addnewlimitAction` Function

To add new request limit from portal we use `addnewlimitAction()`. Decrypt data that comes from MOP operator, portal depending on key using `decryptAction ($key, $post[0])`. Call `getClientPhone($data['phone'])` function from Dao->pdo->mysql->Client.php to get client phone information from client_phone table depending on phone. Call `tempClient ($new, $decrypt[0]['client_id'], 'insert')` function from Dao->pdo->mysql->Client.php to get or insert or delete client request information from/to temp_client table depending on client_id.

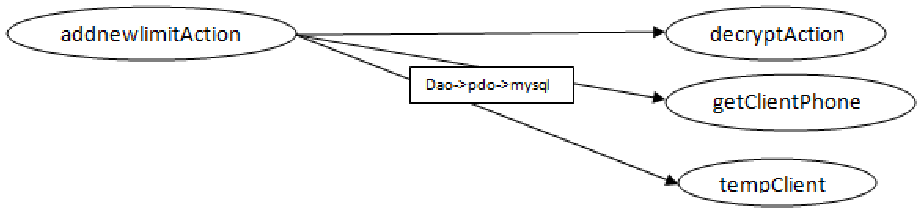


Figure 6.13: AddnewlimitAction Function.

6.4.3.4 `athinticationlevelAction` Function

Call `selectcdrclientwaiting (null, null, "p.id='$cdrID'")` function from Dao->pdo->mysql->Client.php to get transaction information from cdr table. Call `getAuthintication($cdrID)` function from Dao->pdo->mysql->Client.php to get authentication information from cdr_authentication_level table depending on cdr_id. Send call back information to IVR system using `ivrConnectionAction ($url, $postdata)`.

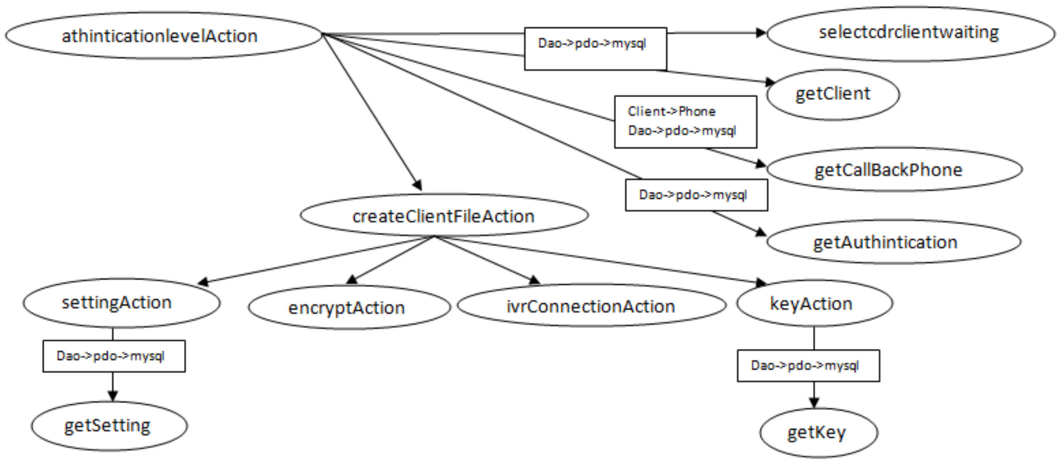


Figure 6.14: AthinticationlevelAction Function.

6.4.3.5 `autochangestatusAction` Function

Used to check if there are transactions without callback. Call `autoChangeStatus()` function from Dao->pdo->mysql->Client.php, then call getCallBackNum ($dataForm->id), createClientFile ($cdr_id, $callbackNum, $method=null,$authentication=null). Call checkMOPidExistInTheClient(MOP_id) function from Dao->pdo->mysql->Client.php to check if MOP ID exists or not from client table depending on MOP_id. Call getCallBack($phone, client_id) function from Dao->pdo->mysql->Client.php to get call back information from callback_phone table depending on phone, client_id.

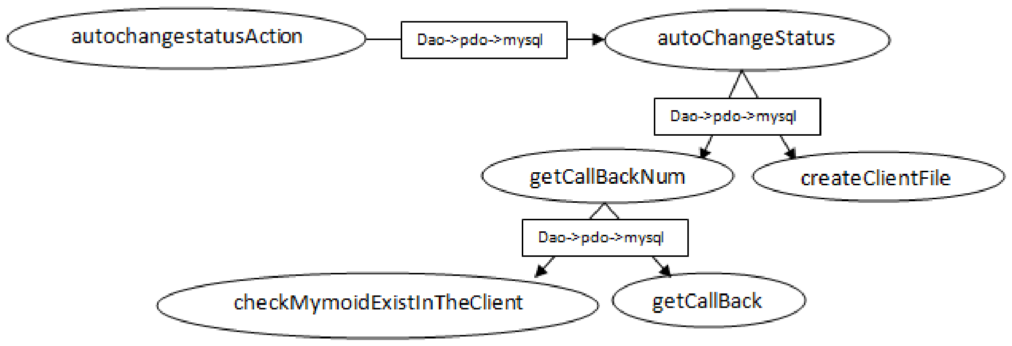


Figure 6.15: AutochangestatusAction Function.

6.4.3.6 `callbackAction` Function

Get transaction information to create callback using `callbackAction()`. Call `selectcdrclientwaiting (null,"status = '2' ", null,'id')` function from Dao->pdo->mysql->Client.php to get transaction information from cdr table.

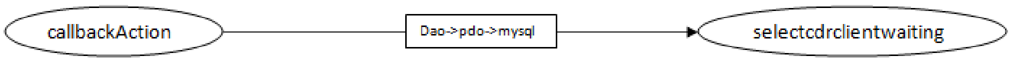


Figure 6.16: CallbackAction Function.

6.4.3.7 `changestatusAction` Function

Send transaction information to correspondent bank using `cbAction ($data, $ip)`. Change cdr row status using `changestatusAction()`.Call `settingAction()` function using `smsipAction()`. Call `changeStauts` function from Dao->pdo->mysql->Client.php to change status for cdr row depending on value that is retrieved from bank. Call `getDstMOPAcc()` function from Dao->pdo->mysql->Client.php to get client information from client table JOIN client_account table depending on MOP_id. Call `inActiveClient ($getDstMymoAcc->client_id)` function from Dao->pdo->mysql->Client.php to get client information from client table depending on is_active = '1'.

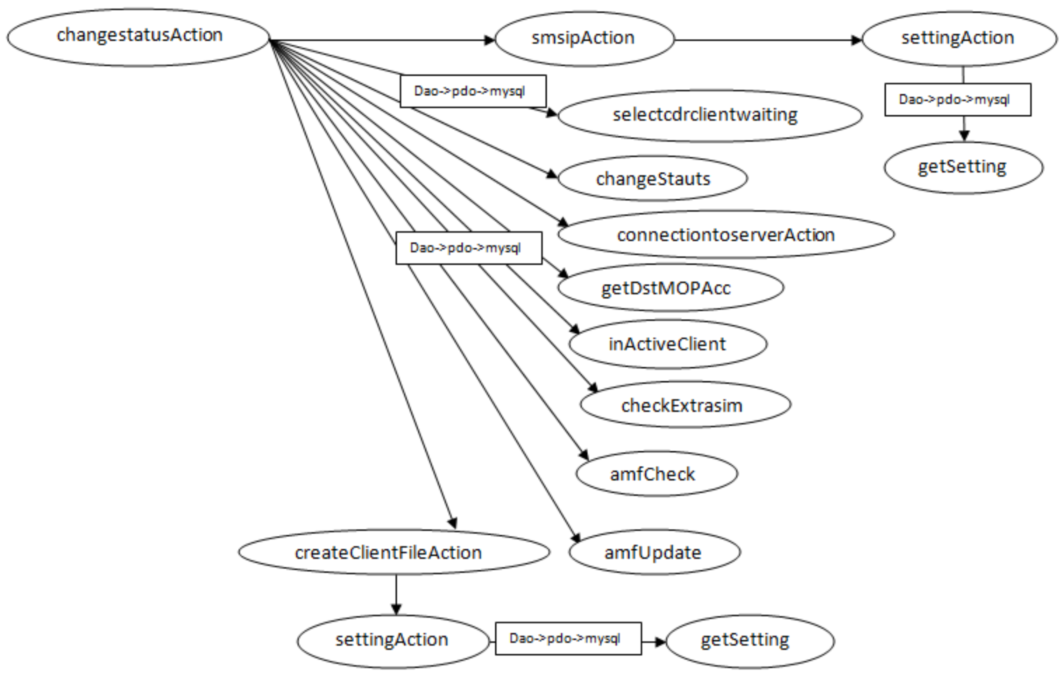


Figure 6.17: ChangestatusAction Function.

6.4.3.8 `checkpinAction` Function

Check if PIN number is valid or not using `checkpinAction().`Call `getClient(MOP_id)` function from Dao->pdo->mysql->Client.php to get client information from client table depending on MOP_id. Call `getKey($id)` function from Dao->pdo->mysql->Client.php to get key information from validation_key table. Send post method to HSM to check pin number using `hsmAction($data, 'check').`

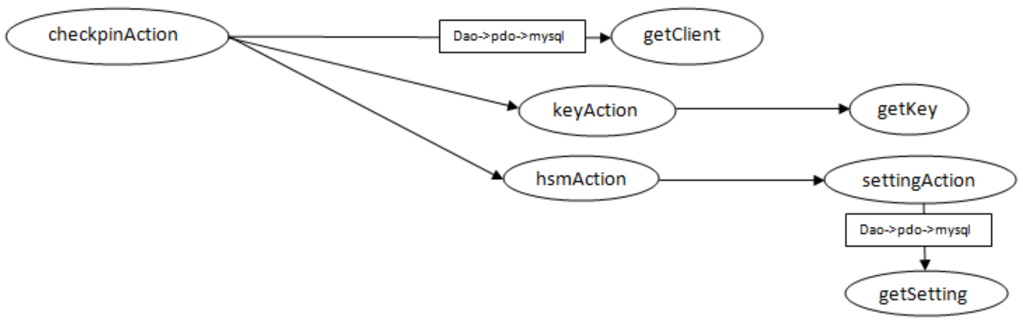


Figure 6.18: CheckpinAction Function.

6.4.3.9 `feesAction` Function

Used to insert fees information to bank. Call `fees ($row,'insert')` function from Dao->pdo->mysql->Client.php to get or insert fees information from client_fees table depending on id.

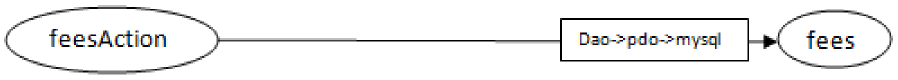


Figure 6.19: FeesAction Function.

6.4.3.10 `hsmAction` Function

Send post request to HSM system to check or change client information using `hsmAction($data, $op)`. Call `settingAction ($setting_id, $field_id)` function from Dao->pdo->mysql->Client.php to get setting information from setting table depending on setting_id, field_id.

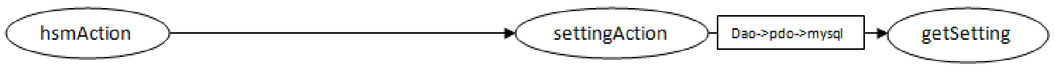


Figure 6.20: HSMAction Function.

6.5 Security Implementation

In our implementation we used the Hardware Security Module (HSM), also called cryptographic accelerators, to enable secure cryptographic-key management and fast cryptographic operations. They do the former by keeping cryptographic-key material in special nonvolatile memory designed to erase its content when it’s tampered with. They do the latter with circuits that facilitate arithmetic with long integers, on which much public-key cryptography is based. Consequently, HSMs are used to generate keys and certificates, store keys, and sign documents [44]. The HSM support Public-key Cryptography Standard #11, Microsoft CryptoAPI, CryptoAPI Next Generation, and Java Cryptography Architecture, Java Cryptography Extension. HSM establishes secure channels by various methods such as[45] :

1. Asymmetric RSA (1024 bit), DSA, Diffie-Hellman and Elliptic Curve Cryptography (ECDSA, ECDH, Ed25519, ECIES)
2. Symmetric AES, AES-GCM, DES, Triple DES, ARIA, SEED, RC2, RC4, RC5, CAST, and more.
3. Hash/Message Digest/HMAC: SHA-1, SHA-2, SM3.
4. Random Number Generation: designed to comply with AIS 20/31 to DRG.4 using HW based true noise source alongside NIST 800-90A compliant CTR-DRBG.

Chapter 7

DISCUSSION

In this chapter we highlight the advantages of MOP and discuss various aspects of MOP in relation to other payment systems (mobile or otherwise).

7.1 Applicability

Our system is applicable at all times and places, for personal and electronic purchases, transferring money (B2P, P2B, P2P), invoices and payments for all services (e.g. water, electricity and taxes), governmental payments (passport fees, licenses and other documents), school and university tuitions, speed cash, and international transfers.

7.2 Innovative Features of MOP

Despite widespread adoption of mobile-based payment systems, an audit of the same reveals several limitations, which MOP seeks to overcome [37]. To this end, MOP will provide a platform for mobile-based transactions without the need for a new SIM card. The process will include real-time person-to-person transactions between users regardless of the banks they use, and without the need for a third party to facilitate the transaction. MOP will be integrated into existing banking systems and reliable financial institutions which will confirm and authorize all transactions. In this respect, this proposal provides a mobile phone-based solution that will help financial companies achieve seamless mobile integration of their systems and services. Besides, it is worth noting that the technology does not require one to have a specific type of mobile phone, thereby enabling the provision of unified services to all stakeholders. Furthermore, the technology eliminates intermediaries and other third parties with questionable credentials.

7.3 How MOP Deals with the Security Issue

The proposed solution seeks to mitigate security vulnerabilities by adopting different measures. First, the system leverages on information only known to the customer; i.e., the client's PIN code which is initiated by the customer's bank HSM and only requested by the bank through its IVR system call back feature. The second measure involves something possessed by the customer, i.e., mobile phone and mobile SIM card. The third security feature is using several authentication methods and process flow. Through this measure, the process is initiated by the operator. Afterward, the transaction is split into 3 sections (sender's bank, operator, and receiver's bank). Next, the sender bank requests for PIN code using the IVR call back to the sender's phone number, who has the SIM card and knows PIN code. Later, the PIN code is verified by the sender bank. Finally, the client's eligibility is determined by their internal bank system and all transaction information flow are highly encrypted and with no involvement of the other parts. The fourth safety measure is that a private network connects all parties with a secure VPN. The trend these days for any payment solution is to cover the most important points: real time settlement, security, interoperability and user convention, and MOP addresses all of these points.

7.4 Comparison with Other Payment Systems

In table 7.1 we compare our proposed system (MOP) with other systems according to third party, monthly fees, per transaction fee, need to new account, multi currency support, country support and how much time is needed to make a transaction. Information on other payment methods are obtained from [37-39, 46].

Table 7.1: Payment methods compared.

Payment Method	**Square**	**VISA**	**PayPal**	**PayBox**	**Pay for IT**	**Authorize.Net**	**Proposed System (MOP)**
Third Party	Yes	Yes	Yes	Yes	Yes	Yes	No
Monthly Fee	0$	2$	0$	0$	0$	25$	0$
Per-transaction fee	2.9%+0.3$	2.9%+0.3$	2.9%+0.3$	Depends on reseller	Depends on MNO’s	2.9%+0.3$	0.1%
New Account needed	Yes	Yes	Yes	Yes	Yes	Yes	No
Multi Currency Support	Yes	USD	Yes	Yes	Yes	USD	Yes
Countries	5	All Countries	203	All Countries	UK	190	All Countries
Time	1-3 Days	1-3 Days	Min. 6 Hours	1-2 Days	1-2 Days	Min. 6 Hours	Average 70.10 seconds, with a standard deviation of 15.67.

We can see that all other payment methods use an external third party, which casts doubts about their security. The customer needs to open several accounts in each payment method and charge this account to transfer money. Most of the payment method are free, but get fees for each transaction, that means they are more expensive.

The most important point is the time needed to transfer money: all payment methods need more than 6 hours to transfer money, but in our system, after the user calls to the MOP operator and gives the transaction details, the system needs on average 70.10 seconds, with a standard deviation of 15.67 seconds, to transfer money (Appendix F contains the raw data for the transactions).

Chapter 8

CONCLUSION AND FUTURE WORK

In order to remedy the shortcomings of existing mobile payment systems, we proposed a new mobile payment methodology, using a novel communication network, while leveraging existing trusted banking infrastructure. The proposed methodology has the following features: (i) it jointly connects the banks together and allows the customers to process all kinds of transactions involving money transfer with the use of their cell phones and without the need for a new SIM card; (ii) real-time person to person transactions between users with similar or different bank accounts can be processed without the need for a third party mediator; (iii) there is no need for Internet or a smart phone to make a transfer; (iv) IVR is used for communication of banks with their customers; and (v) the proposed communication network is built over the already existing “banking system" infrastructures, with processes approved by reliable financial institutions, which reinforces and strengthens consumer confidence and reliability in the proposed methodology.

We also implemented a proof-of-concept prototype, as well as a web-based simulator, of our proposal and collected timing data for performing transactions on the prototype implementation. The collected timing data showed that our system compares very favorably with other money transfer schemes.

Blockchain technology has the potential for fraud reduction, secure, and fast transactions, lower cost, improved data quality, smart contracts, payments, and can provide a solid trading platform. For future work, we plan to investigate ways in which blockchain technology can be incorporated into our proposal in order to eliminate the banks' and financial authorities' concerns, answer the customers' (both merchants and retailers) needs for faster, safer, cheaper, real-time and secure payments that eliminate the need for intermediary parties to approve and reconcile the transaction, while reducing the operational risk, as all the transactions will be transparent and practically unalterable.

REFERENCES

[1] J. Hedman and S. Henningsson, "The new normal: Market cooperation in the mobile payments ecosystem," *Electronic Commerce Research and Applications*, vol. 14, pp. 305-318, 2015.

[2] T. Dahlberg, J. Guo, and J. Ondrus, "A critical review of mobile payment research," *Electronic Commerce Research and Applications*, vol. 14, pp. 265-284, 2015.

[3] I. O. Oyefolahan, S. a. A. Ahmed, and A. Abubakar, "An empirical study of customers' adoption of Mobile Money Transfer Services in Somaliland," in *Information and Communication Technology for The Muslim World (ICT4M), 2014 The 5th International Conference on*, 2014, pp. 1-5.

[4] Gartner. (2019, 20.3.2019). *Gartner*. Available: https://www.gartner.com/en.

[5] V. K. Raina, "Overview of mobile payment: technologies and security," in *Banking, Finance, and Accounting: Concepts, Methodologies, Tools, and Applications*, ed: IGI Global, 2015, pp. 180-217.

[6] J. Liu, R. J. Kauffman, and D. Ma, "Competition, cooperation, and regulation: Understanding the evolution of the mobile payments technology ecosystem," *Electronic Commerce Research and Applications*, vol. 14, pp. 372-391, 2015.

[7] D. Shrier, G. Canale, and A. Pentland, "Mobile money & payments: Technology trends," ed: MIT Connection Science’s series on financial technology, 2016.

[8] W. Jack and T. Suri, "Risk sharing and transactions costs: Evidence from Kenya's mobile money revolution," *The American Economic Review,* vol. 104, pp. 183-223, 2014.

[9] E. Seth, "Mobile Commerce: A Broader Perspective," *IT Professional,* vol. 16, pp. 61-65, 2014.

[10] J. T. Isaac and S. Zeadally, "Secure Mobile Payment Systems," *IT Professional,* vol. 16, pp. 36-43, 2014.

[11] B. Singh and K. Jasmine, "Comparative study on various methods and types of mobile payment system," in *Advances in Mobile Network, Communication and its Applications (MNCAPPS), 2012 International Conference on*, 2012, pp. 143-148.

[12] A. Gaur and J. Ondrus, "The role of banks in the mobile payment ecosystem: a strategic asset perspective," in *Proceedings of the 14th annual international conference on electronic commerce,* 2012, pp. 171-177.

[13] P. Ozcan and F. M. Santos, "The market that never was: Turf wars and failed alliances in mobile payments," *Strategic management journal,* vol. 36, pp. 1486-1512, 2015.

[14] K. S. Staykova and J. Damsgaard, "The race to dominate the mobile payments platform: Entry and expansion strategies," *Electronic Commerce Research and Applications,* vol. 14, pp. 319-330, 2015.

[15] A. Mugambi, C. Njunge, and S. C. Yang, "Mobile-Money Benefits and Usage: The Case of M-PESA," *IT Professional,* vol. 16, pp. 16-21, 2014.

[16] W.-M. To and L. S. Lai, "Mobile banking and payment in China," *IT Professional,* vol. 16, pp. 22-27, 2014.

[17] K.-Y. Chen and M.-L. Chang, "User acceptance of 'near field communication'mobile phone service: an investigation based on the 'unified theory of acceptance and use of technology'model," *The Service Industries Journal,* vol. 33, pp. 609-623, 2013.

[18] S. Chauhan, "Evaluating Acceptance of Mobile Money by Poor Citizens in India: An Empirical Study," 2014.

[19] M. Cocosila and H. Trabelsi, "An integrated value-risk investigation of contactless mobile payments adoption," *Electronic Commerce Research and Applications,* vol. 20, pp. 159-170, 2016.

[20] F. Bar, M. S. Weber, and F. Pisani, "Mobile technology appropriation in a distant mirror: Baroquization, creolization, and cannibalism," *new media & society,* p. 1461444816629474, 2016.

[21] V. Mishra and S. S. Bisht, "Mobile banking in a developing economy: A customer-centric model for policy formulation," *Telecommunications Policy,* vol. 37, pp. 503-514, 2013.

[22] J. C. Aker, R. Boumnijel, A. McClelland, and N. Tierney, "Payment Mechanisms and Antipoverty Programs: Evidence from a Mobile Money Cash Transfer Experiment in Niger," *Economic Development and Cultural Change,* vol. 65, pp. 1-37, 2016.

[23] J. Kaye, J. Vertesi, J. Ferreira, B. Brown, and M. Perry, "# CHIMoney: Financial interactions, digital cash, capital exchange and mobile money," in *CHI'14 Extended Abstracts on Human Factors in Computing Systems*, 2014, pp. 111-114.

[24] M. de Reuver, E. Verschuur, F. Nikayin, N. Cerpa, and H. Bouwman, "Collective action for mobile payment platforms: A case study on collaboration issues between banks and telecom operators," *Electronic Commerce Research and Applications,* vol. 14, pp. 331-344, 2015.

[25] J. Hedman and S. Henningsson, "Competition and collaboration shaping the digital payment infrastructure," in *Proceedings of the 14th Annual International Conference on Electronic Commerce*, 2012, pp. 178-185.

[26] R. Magnier-Watanabe, "An institutional perspective of mobile payment adoption: The case of Japan," in *2014 47th Hawaii International Conference on System Sciences*, 2014, pp. 1043-1052.

[27] K. Alshare and A. Mousa, "The moderating effect of espoused cultural dimensions on consumer's intention to use mobile payment devices," 2014.

[28] Y.-H. Cheng and T.-Y. Huang, "High speed rail passengers' mobile ticketing adoption," *Transportation Research Part C: Emerging Technologies*, vol. 30, pp. 143-160, 2013.

[29] L. Jia, D. Hall, and S. Sun, "The Effect of Technology Usage Habits on Consumers' Intention to Continue Use Mobile Payments," 2014.

[30] T. Dahlberg and A. Oorni, "Understanding changes in consumer payment habits-do mobile payments and electronic invoices attract consumers?," in *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, 2007, pp. 50-50.

[31] H. Xin, A. A. Techatassanasoontorn, and F. B. Tan, "Exploring the influence of trust on mobile payment adoption," 2013.

[32] T. Apanasevic, "Factors Influencing the Slow Rate of Penetration of NFC Mobile Payment in Western Europe," in *ICMB*, 2013, p. 8.

[33] J. P. de Albuquerque, E. H. Diniz, and A. K. Cernev, "Mobile payments: a scoping study of the literature and issues for future research," *Information Development*, p. 0266666914557338, 2014.

[34] A. Gannamaneni, J. Ondrus, and K. Lyytinen, "A post-failure analysis of mobile payment platforms," in *System Sciences (HICSS), 2015 48th Hawaii International Conference on*, 2015, pp. 1159-1168.

[35] T. Dahlberg, N. Mallat, J. Ondrus, and A. Zmijewska, "Past, present and future of mobile payments research: A literature review," *Electronic Commerce Research and Applications*, vol. 7, pp. 165-181, 2008.

[36] L. Goeke and K. Pousttchi, "A scenario-based analysis of mobile payment acceptance," in *Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR), 2010 Ninth International Conference on*, 2010, pp. 371-378.

[37] PayPal. (2019, 20.3.2019). *Paypal*. Available: https://www.paypal.com/us/home

[38] PayBox. (2019, 20.3.2019). *PayBox*. Available: https://www.pay-box.in/.

[39] PayforIT. (2019, 20.3.2019). *PayforIT*. Available: https://www.payforit.org/.

[40] Square. (2019, 20.3.2019). *Square*. Available: https://squareup.com/us/en.

[41] E. C. Bank. (2019, 12.5.2018). *What is TARGET Instant Payment Settlement (TIPS)?* Available: https://www.ecb.europa.eu/paym/target/tips/html/index.en.html.

[42] D. Linux. (2018). *Debian Linux*. Available: http://cdimage.debian.org/debiancd/7.1.0/amd64/iso-dvd/.

[43].Asterisk. (2018). *Asterisk*. Available: http://www.asterisk.org/downloads/asterisk/all-asterisk-versions.

[44] B. Köppel and S. Neuhaus, "Analysis of a hardware security module's high-availability setting," *IEEE Security & Privacy*, vol. 11, pp. 77-80, 2013.

[45] D. Kim, Y. Jeon, and J. Kim, "A secure channel establishment method on a hardware security module," in *2014 International Conference on Information and Communication Technology Convergence (ICTC)*, 2014, pp. 555-556.

[46] WUFFO. (2019, 30.07). Available: https://www.wufoo.com/payments/payment-gateway-comparison/.

[47] OneDrive (2020, 20.01). Available: https://aaujedu-my.sharepoint.com/:u:/g/personal/mahmoud_obaid_aaup_edu/ETp8J5oLfhRJjF2oKqgdXoAB0tfFjWDiaPJ44DtToZlQQQ?e=NndIjW.

APPENDICES

Appendix A: Critical Code for the “Client” module of the MOP Prototype

In this appendix, we give critical sections of the code in the “Client” module, with some explanation to explain what the code does.

Listing A.1: CheckcallAction Function in File Client.php

```
public function checkcallAction() {

      $request = $this->getRequest();
      $phone   = $request->getParam('phone');

      $getRecordeByPhone  = $this->ObjectDao->getRecordeByPhone($phone);

      //Zend_Debug::dump($getRecordeByPhone);die;

      if(!$getRecordeByPhone)
      {// no user under this phone number ...
         $value = number_format(0);
         print($value);die;
      }

      $setting = $this->settingAction(null,20);

      $checkList = array();
      foreach($setting as $row)
      {
         if($row->value == 'on')
         {
            $checkList[] = $row->key;
         }
      }

      $where = '';
      foreach($checkList as $row)
      {
         if(isset($row) && $row == 'international_id')
            $where.= "cif_nation_code = '$getRecordeByPhone->national_id'";

         if(isset($row) && $row == 'name')
         {
            if($where)
               $where.=' AND ';

            $firstname = $getRecordeByPhone->firstname;
            $where.= "cif_first_name_eng = '$firstname'";

            if($getRecordeByPhone->middlename)
            {
               $middlename =$getRecordeByPhone->middlename;
               $where.= " AND cif_sec_name_eng = '$middlename'";
            }

            $fathername = $getRecordeByPhone->fathername;
            $where.= " AND cif_third_name_eng = '$fathername'";

            $familyname = $getRecordeByPhone->familyname;
            $where.= " AND cif_last_name_eng = '$familyname'";


         }


      }
      //echo $where;die;
```

```
    $blacklist = MOP_Model_Dao_Factory::getInstance()->setModule('blacklist')->getBlacklistDao();
    $blacklist->setDbConnection($this->conn);
    $blacklist = $blacklist->check($where);

    if($blacklist)
    {// black list
       $value = number_format(2);
       print($value);die;
    }
    //Zend_Debug::dump($getRecordeByPhone->MOP_id);die;


    print($getRecordeByPhone->MOP_id);die;
 }
```

In the first step we need to check if the phone number is a registered as a MOP user or not, so we use the `checkcallAction` function. To get all user information by phone number we use the `getRecordeByPhone`. `settingAction` function calls `getSetting` function from Dao->pdo->mysql->Client.php. We check if the user is black listed or not by calling `check` from blacklist module with Path blacklist->dao->pdo->Blacklist.php.

Listing A.2: AddingtoinsertdbcAction Function in File Client.php

```
public function addingtoinsertdbcAction()
   {
       /**
       *  add the client in the MOP server
       */
      $request = $this->getRequest();
      // post date from MOP bank depending on curl function
      $formData = $request->getPost();
      //Zend_Debug::dump($formData);die;
      /**
       *
       */
       if(isset($formData['main']) && $formData['main'])
       {
          $string = $formData['main'];
          $key = $this->keyAction(7);
          // decrypt
          $decry_data   = $this->decryptAction($key, $string);
          $decry_data   = rtrim($decry_data, "\0");
          $data_decrypt = json_decode($decry_data,true);
          //Zend_Debug::dump($data_decrypt);die;
       }

      /**
       *
       */
      $data_decrypt['validation'] = $formData['validation'];

      if(isset($formData['action']))
          $data_decrypt['action']     = $formData['action'];

      $formData = $data_decrypt;
      //Zend_Debug::dump($formData);die('dd');
      //echo $formData['validation'];die;
      //if($formData && isset($formData['validation']) && $formData['validation'] == MOP_Config::getConfig()->web-
>secretcode)
      //Zend_Debug::dump($formData);die;
      if($formData)
      {
          $moduleName     = $this->module;

          $formDataExist = array();

          foreach($formData as $key => $value)
          {
             if($key != 'action' && $key != 'validation' && $key != 'bank_ip')
                 $formDataExist[$key] = $value;

             // Select bank id depending on bank ip which we have written on bank config
             if($key == 'bank_ip')
             {
                 $bank_id = $this->ObjectDao->getBankId($value);
                 $formDataExist['bank_id'] = $bank_id;
             }

          }

          $className      = ucfirst($moduleName).'_Models_'.ucfirst($moduleName);

          /*$objectName = 'get'.ucfirst($moduleName).'Dao';
          $conn_2 = MOP_Db_Connection::factory()->getMasterConnection();
          $ObjectDao_2 = MOP_Model_Dao_Factory::getInstance()->setModule($moduleName)->$objectName();
          $ObjectDao_2->setDbConnection($conn_2);*/

          if(isset($formData['action']) && $formData['action'] == 'edit')
          {
             $MOPserver_dataForm = new $className($formDataExist);

             $RecordId = $this->ObjectDao->update($MOPserver_dataForm, 'MOP_server');

             /***************************/

             if(isset($formData['extraData']) && is_array($formData['extraData']))
             {
```

```
            $this->addphonesAction($formData['extraData'], $formData['client_id'], $formData['action']);
            $this->addaccountsnumAction($formData['extraData']['account_num'], $formData['client_id'], $formData['action']);
            $this->addaccountsnumAction($formData['extraData']['account_num'], $formData['client_id'],$formData['action']);

          }
          /****************************/
          die;
        }
        else
        {
            $formDataExist['language'] = 'en';

            /*$data = array(
                'name'                              => $formData['name'],
                'phone'        => $formData['phone'],
                'MOP_id'      => $formData['MOP_id'],
                'bank_ip'      => $formData['bank_ip'],
                'creation_date'=> $formData['creation_date'],
                'language'     => 'en',
                'is_active'    => $formData['is_active']
              );*/

            //Zend_Debug::dump($MOPserver_dataForm);die;

            $MOPserver_dataForm = new $className($formDataExist);
            $RecordId = $this->ObjectDao->add($MOPserver_dataForm, 'MOP_server');

            /****************************/

            if(is_array($formData['extraData']))
            {
              $this->addphonesAction($formData['extraData'], $RecordId);
            }

            if(is_array($formData['extraData']['account_num']))
            {
              $this->addaccountsnumAction($formData['extraData']['account_num'], $RecordId);
            }
            /****************************/

            $MOP_id = array();
            $MOP_id['MOP_id'] = $formData['MOP_id'];
            //$this->ObjectDao->MOPLastUsed('delete',$MOP_id['MOP_id']);

            /**
            *  old form to MOP id
            */
            //$this->ObjectDao->update_MOPid($MOP_id);

            /**
             *  adding record to the traslate table
             */

            $source = Zend_Json::decode($request->getPost('sourceItem', '{"id": "", "language": ""}'));
            //Zend_Debug::dump($source['language']);die;
                                $translationDao = MOP_Model_Dao_Factory::getInstance()->setModule('core')-
>getTranslationDao();
                                $translationDao->setDbConnection($this->conn);
                                $translationDao->add(new Core_Models_Translation(array(
                                        'item_id'          => $RecordId,
                                        'item_class'      => get_class($MOPserver_dataForm),
                                        'source_item_id'  => ('' == $source['id']) ? $RecordId : $source['id'],
                                        'language'        => 'en',
                                        'source_language' => ('' == $source['language']) ? null : $source['language'],
                                )));

            die(1);
        }

    }
    else
    {
        Zend_Layout::startMvc(array('layoutPath' => MOP_APP_DIR . DS . 'templates' . DS . 'default' . DS . 'layouts' . DS .
'en'));
```

```
        Zend_Layout::getMvcInstance()->setLayout('message');
    }
  }
```

In the bank, in order to add the client into the MOP server data from the MOP bank we use addingtoinsertdbcAction function. keyAction function calls getKey function from Dao->pdo->mysql->Client.php to encrypt or decrypt from validation_key table. Decrypted data is obtained from the MOP bank depending on the key by using decryptAction function. getBankId is used to Select bank ID. To generate a random PIN for a phone number we use the generateNumberAction function. To insert client account information to the client_account table we use addaccountsnumAction function. We call add function from core modules Dao->pdo->mysql-> Translation.php to insert information to the core_translation table.

Listing A.3: AddtooperatorAction Function in File Client.php.

```
function addtooperatorAction() {

    // adding the recorde in the operator in the other country (after successfully Trx)

    $request    = $this->getRequest();
    $postData   = $request->getPost();

    if(isset($postData['disposition']))
    {
        $disposition = $postData['disposition'];
    }
    else
    {
        $disposition = null;
    }

    if(isset($postData['extra_data']))
    {
        $extra_data = $postData['extra_data'];
    }
    else
    {
        $extra_data = null;
    }

      $cdr = array(
            'id'                        => $postData['cdr_id'],
            'duration'      => $postData['duration'],
            'dst_MOPid'    => $postData['dst_MOPid'],
            'src_MOPid'    => $postData['src_MOPid'],
            'issuer_bank'   => $postData['issuer_bank'],
            'receiver_bank' => $postData['receiver_bank'],
            'amount'        => $postData['amount'],
            'start'         => $postData['start'],
            'answer'        => $postData['answer'],
            'end'           => $postData['end'],
            'src'           => $postData['src'],
            'clid'          => $postData['clid'],
            'channel'       => $postData['channel'],
            'billsec'       => $postData['billsec'],
            'disposition'   => $disposition,
            'status'        => $postData['status'],
            'trx_method'    => $postData['trx_method'],
            'trx_type'      => $postData['trx_type'],
            'extra_data'    => $extra_data,
            'country_id'    => $postData['country_id']
        );

    if(isset($postData['trx_type_u']))
    {
        $updated = 1;
    }
    else
    {
        $updated = 0;
    }

    $cdr_comm = array(
                    'trx_type'        => $postData['trx_type'],
                    'client_id'       => $postData['client_id'],
                    'MOP_comm'       => $postData['MOP_comm'],
                    'bank_comm'       => $postData['bank_comm'],
                    'merchant_comm'   => $postData['merchant_comm'],
                    'amount_sender'   => $postData['amount_sender'],
                    'amount_receiver' => $postData['amount_receiver'],
                    'cdr_id'          => $postData['cdr_id'],
                    'updated'         => $updated,
                    'country_id'      => $postData['country_id']
                );

     $cdr_comm_updated = null;
     if(isset($postData['trx_type_u']))
     {
        $cdr_comm_updated = array(
                        'trx_type'          => $postData['trx_type_u'],
                        'client_id'         => $postData['client_id_u'],
                        'MOP_comm'          => $postData['MOP_comm_u'],
                        'bank_comm'         => $postData['bank_comm_u'],
                        'amount'            => $postData['amount_u'],
                        'amount_sender'     => $postData['amount_sender_u'],
                        'cdr_id'            => $postData['cdr_id'],
                        'updated_date'      => $postData['updated_date_u'],
                        'updated_by'        => $postData['updated_by_u'],
                        'country_id'        => $postData['country_id']
                      );
        }


        /**
        *  to Correspondent Bank ...
        */
        $postData['correspondent_bank'] = 1;

        $bank = $this->ObjectDao->banks(null, true);

        if(isset($bank->ip))
           $bankCoo = $this->connectiontoserverAction($postData, $bank->ip);


        if(isset($postData['remittance']) && $bankCoo)
           $changeStauts = $this->ObjectDao->changeStauts($callback->id, 1, $callback->country_id);
        elseif(isset($postData['remittance']) && $bankCoo == 6)
        {
           $changeStauts = $this->ObjectDao->changeStauts($callback->id, 6, $callback->country_id);

           $value = number_format(6);
           print($value);die;
        }
        elseif(isset($postData['remittance']) && !$bankCoo)
           $changeStauts = $this->ObjectDao->changeStauts($callback->id, 5, $callback->country_id);


        if(!isset($postData['remittance']))
        {
           /**
           *  to Receiver Bank ...
           */
           $getClient      = $this->ObjectDao->getClient($postData['dst_MOPid']);
           if($getClient->issuer_bank_ip)
           {
              $receiver = $postData;
              $receiver['receiver_bank'] = 1;

              $receiverBankIp   = $getClient->issuer_bank_ip;
              $bankCoo = $this->connectiontoserverAction($receiver, $receiverBankIp);
           }

           //Zend_Debug::dump($bank);die('dd 54 ALA');

           $phoneDao   = MOP_Model_Dao_Factory::getInstance()->setModule('client')->getLogDao();
                $phoneDao->setDbConnection($this->conn);
           $cdrRecord  = $phoneDao->insertCdr($cdr, $cdr_comm, $cdr_comm_updated);
        }


        die;
     }
```

Adding the record in the operator after a successful transaction we use `addtooperatorAction` function. We call `banks` function from Dao->pdo-

>mysql->Client.php to get bank_id from the bank. We send data to bank (index.php/add-client-waiting) to make call-back for client using `connectiontoserverAction` function. We call `changeStauts` function from Dao->pdo->mysql->Client.php to change status for cdr row depending on value that is retrieved from the bank. After that we call `getClient` function from Dao->pdo->mysql->Client.php to get client information. We call `insertCdr` function from Dao->pdo->mysql-> Log.php to insert call information to cdr, cdr_commission, cdr_commission_updated tables.

Listing A.4: AutoaddcdrAction Function in File Client.php.

```
function autoaddcdrAction() {

      $getTempTable = $this->ObjectDao->getTempTable();

      $reuest = $this->getRequest();
      if($reuest->isPost())
      {
         $post   = $reuest->getPost();
         //
         if(isset($post['respose']))
            $respose_false = $post['respose'];
      }

      foreach($getTempTable as $row)
      {
         $fields = explode('*',$row->data);
         //Zend_Debug::dump($fields);die;
         $phone_number       = $fields[0];
         $MOP_reciver_id    = $fields[1];
         $MOP_sender_id     = $fields[2];
         $amount            = $fields[3];
         $type              = $fields[4];
         $method            = $fields[5];
         $account           = $fields[6];
         $currency          = $fields[7];

         /**
          * Phone_Number*MOP_Reciver*MOP_Sender*Amount*Type*Method*Account*Currency*Extension*Trx_way
          */

         if (isset($fields[8]))
            $extension         = $fields[8];
         else
            $extension         = '';

         if (isset($fields[9]))
            $trxWay         = $fields[9];
         else
            $trxWay         = '';


         /**
           *
           */
         if($extension && is_numeric($extension))
            $extension     = "extension=$extension";
         else
            $extension     = '';


         if( !is_numeric($phone_number) ||
            !is_numeric($MOP_reciver_id) ||
            !is_numeric($MOP_sender_id) ||
            !is_numeric($amount)
           )
         {
            //$value = number_format(0);
            //print($value);
            //die;
         }
         else
         {
            $banks = $this->checkBankesAction($MOP_sender_id, $MOP_reciver_id);

            if(!$account)
               $account = 1;

            $clientReceiver = $this->ObjectDao->getClient($MOP_reciver_id);

            $MOP_receiver    = substr($MOP_reciver_id, 0, 3);
            $MOP_sender      = substr($MOP_sender_id, 0, 3);

            if($type == 3 && $method == 3)
            {//bill trx
            $type      = 3;
            $method    = 3;

            $extension = "$fields[8]";
          }
          if($type == 14 && $method == 14)
          {
            $type      = 14;
            $method    = 14;
          }
          elseif(isset($clientReceiver->user_type))
          {//if exist then the trancefer is local
            $type   = $clientReceiver->user_type;
            $method = $clientReceiver->user_type;
          }
          elseif($MOP_sender != $MOP_receiver)
          {//if not the trancefer is remetance
            $type   = 13;
            $method = 13;
          }

          $transactionLast  = $this->ObjectDao->addcdrclientwaiting($phone_number,
                                                    $MOP_reciver_id,
                                                    $MOP_sender_id,
                                                    $amount,
                                                    null,
                                                    $type,
                                                    $method,
                                                    $trxWay,
                                                    null,
                                                    $banks['issuer_bank'],
                                                    $banks['receiver_bank'],
                                                    $account,
                                                    $currency,
                                                    $extension);


          /**
          * commission
          */
          $this->commissionAction($amount, $type, $banks['client_id_issuer'], $banks['issuer_bank'], $transactionLast);
          /**
          *
          */

          if(!$transactionLast)
          {
            //$value = number_format(0);
            //print($value);
          }
          else
          {
            if(!isset($respose_false))
            {
              $value = number_format($transactionLast);
              print($value);
            }
          }
        }
        /**
          *
          */
        $getTempTable = $this->ObjectDao->deleteTempTable($row->id);
        //$value = number_format(1);
        //print($value);
        //die;
      }

      //$value = number_format(00);
      //print($value);
      die;
   }
```

When a transaction is created, the information of this transaction is inserted in the temp table to which we get using the `getTempTable` function. We call `checkBankes` function from Dao->pdo->mysql->Client.php to get the sender and receiver banks. We call the `getClient` function from Dao->pdo->mysql->Client.php to select client information and issuer_bank_ip from client bank table. We call the `addcdrclientwaiting` function from Dao->pdo->mysql->Client.php to insert call information into the cdr table. We call `trxComm` function from Dao->pdo->mysql->Client.php to get commission information. We call `addTrxComm` function from Dao->pdo->mysql->Client.php; inside this function we call `getDefaultOperator` to insert commission info to cdr_commission table.

Listing A.5: CallbackstatusAction Function in File Client.php.

```
function callbackstatusAction($status=null,$ip=null,$id=null)
   {

       $reuest = $this->getRequest();
       if($reuest->isPost())
       {
          $post   = $reuest->getPost();
          //Zend_Debug::dump($post);die;
          if(isset($post['respose']))
              $respose_false = $post['respose'];
       }

       $sms = $this->ObjectDao->smsTransaction();
       if($sms)
       {
          foreach($sms as $row)
          {
              $details = explode('*',$row->keyword);

              if(count($details) && $details[0] && $details[1] && $details[2])
              {
                  $senderPhone    = $row->phone;
                  $MOPSender = $this->ObjectDao->getRecordeByPhone($senderPhone);
                  $MOPSender = $MOPSender->MOP_id;
              }
              else
                  continue;

              $MOPReceiver   = $details[0];
              $type          = $details[1];
              $amount        = $details[2];
              if(count($details) == 4)
                  $billNumber      = $details[3];

              /******************/

              $client = $this->ObjectDao->getClient($MOPReceiver);

              if ($client && $type == 1)
              {
                  $type = $client->user_type;
              }
              elseif($client && $type == 2)
              {

                  $AgentDao = MOP_Model_Dao_Factory::getInstance()->setModule('agent')->getAgentDao();
                $AgentDao->setDbConnection($this->conn);
                  $agent = $AgentDao->getByMOPId($MOPReceiver);

                  if($agent->agent_id)
                      $extension = "agent=$agent->agent_id";

                  $type      = 3;
                  $method    = 3;

              }

              if(!$client)
              {
                  //$client = $this->checkMOPidremittanceAction($MOPReceiver);
                  $type = 13;
              }
              if(!isset($extension))
                  $extension = '';

              $data = "$senderPhone*$MOPReceiver*$MOPSender*$amount*$type*$type*1*1*$extension*";

              $data_array = array(
                              'id'           => null,
                              'section_id'    => '1',
                              'data'          => $data);
```

```
            $insert = $this->ObjectDao->insertTemp($data_array);


            $ip_sms = $this->ObjectDao->getCountry("status='1'");
            $cdr_url = 'http://'.$ip_sms->operator_ip . $this->view->url(array(),'client_client_autoaddcdr',true);

            $response = $this->excuteurlAction($cdr_url,null,false);

            //$this->excuteurlAction($urlCallBack);
            /******************/
            $smsUpdate = $this->ObjectDao->smsUpdate($row->id);
        }

    }

    if($id && $ip)
        $records = $this->ObjectDao->checkTransactionStatus($status,$id);
    else
        $records = $this->ObjectDao->checkTransactionStatus($status);

    //Zend_Debug::dump($records);die;
    $ip_status = false;
    if($ip)
    {
        $ip        = $ip;
        $ip_status = true;
    }

    foreach($records as $record)
    {
        if(!$ip_status)
            $ip = $record->issuer_bank_ip;

        $getClientAccount = $this->getClientAccountAction($record);
        //Zend_Debug::dump($getClientAccount);die('dd');

        $record->account_sender     = $getClientAccount['sender'];
        $record->account_receiver   = $getClientAccount['receiver'];

        /**
         *
         * [ IF (Transaction in the local country but sender bank deferent from receiver bank)
         *   NOTE: IF $record->receiver_ip == null then the trx is Remittance ]
         * ELSEIF (Trx is Remittance)
         * ELSE (sender bank the same receiver bank so i don't need to send the record to CB')
         *
         */
        if($record->receiver_ip && $record->issuer_bank != $record->receiver_bank) {
            // sender

            $bank = $this->connectiontoserverAction($record, $ip, true);

            // receiver
            $receiver = $record;
            $receiver->receiver_bank = 1;
            $bank = $this->connectiontoserverAction($receiver, $record->receiver_ip, true);

            // CB
            $cb = $record;
            $cb->correspondent_bank = 1;
            $bank = $this->connectiontoserverAction($cb, $record->c_bank, true);
        }
        elseif(!$record->receiver_ip) {
            // sender
            $bank = $this->connectiontoserverAction($record, $ip, true);
            // CB
            $cb = $record;
            $cb->correspondent_bank = 1;
            $bank = $this->connectiontoserverAction($cb, $record->c_bank, true);

            /**
             * send trx to another operator
             */
            $receiver = $record;
            $receiver->receiver_bank = 1;
            $country_code_receiver    = substr($receiver->dst_MOPid, 0, 3);
            $country_code_sender      = substr($MOP_sender, 0, 3);
            //Zend_Debug::dump($cdrRecord);die('d52828d');
            if($country_code_receiver != $country_code_sender)
            {
                $getCountry = $this->ObjectDao->getCountry("code = '$country_code_receiver'");

                if(isset($getCountry->operator_ip) && $getCountry->operator_ip)
                {
                    $operator_ip = $getCountry->operator_ip;
                    $cdrRecord  = $this->connoperatorAction($cdrRecord, $operator_ip);
                }
                //Zend_Debug::dump($getCountry);die('dd');
            }
            /**
             * end
             */

        }
        else
        {

            if(isset($ip->operator_ip))
                $ip = $ip->operator_ip;

            $bank = $this->connectiontoserverAction($record, $ip,true);
        }

    }

    if(!$status)
        die;
}
```

We call `smsTransaction` function from Dao->pdo->mysql->Client.php to get all rows from received_sms table where status ='0'. We call `insertTemp` function to insert SMS information to temp table. `excuteurlAction` is used to send post request to autoaddcdr, waitingcallback function in the operator. We call `checkTransactionStatus` function to insert in the cdr at the bank.

Listing A.6: ChangestatusAction Function in File Client.php.

```
function changestatusAction()
   {
       $request = $this->getRequest();

       $postData = $request->getPost();

       //$phone = $postData['phone'];

       $MOPid = $postData['MOPid'];

       $id = $postData['id'];

       $status = $postData['status'];

//       $MOPid = '1000056429';
//       $id = '72';
       $callback = $this->ObjectDao->selectcdrclientwaiting(null, null, "src_MOPid = '$MOPid' AND id = '$id' AND (status =
'2' OR status = '3') ");


       /**
        *  if request Credit send number to the client
        */
       if(isset($callback) && $callback->trx_method == 9)
       {
          $catID = $callback->extra_data;
          $catID = str_replace('category_id=','',$catID);


          //Zend_Debug::dump();die;

           $this->sendcreditsmsAction($catID,$callback->src);
       }

       /**
        * if trx methode = 3 -> the trx to bill bayment :)
        * here the extra data form = agent=[AGENT ID]
        */
       if(isset($callback) && $callback->trx_method == 3)
       {
          $billData = $callback->extra_data;
          $billData = explode('=',$billData);

          if(count($billData) == 2)
          {// if count = 2 then there only AgentID
             $conn = MOP_Db_Connection::factory()->getMasterConnection();
                                   $agentDao = MOP_Model_Dao_Factory::getInstance()->setModule('agent')->getAgentDao();
                                   $agentDao->setDbConnection($conn);

             $agentDetails = $agentDao->getById($billData[1]);

             $data = array('agent_id'=> $billData[1]);

             $agent = $agentDao->updateAgent($data);

             if($agentDetails->conn_type == 4)
                $send_data = $this->directconnectionAction($agentDetails->linklink,array('MOP_id'=> $callback->src_MOPid));

          }
          elseif(count($billData) == 4 && $billData[2] == 'bill_number')
          {
             $conn = MOP_Db_Connection::factory()->getMasterConnection();
                                   $agentDao = MOP_Model_Dao_Factory::getInstance()->setModule('agent')->getAgentDao();
                                   $agentDao->setDbConnection($conn);
                                   $agentDetails = $agentDao->getById($billData[1]);

             $data = array('agent_id'=> $billData[1],'bill_number' => $billData[3],'MOP_id' => $MOPid);


             $agent = $agentDao->updateAgent($data);

             if($agentDetails->conn_type == 4)
                $send_data = $this->directconnectionAction($agentDetails->link,$data);
             if($agentDetails->conn_type == 5)
```

```
            {
               $isoMessage = $billData[1].".$MOPid.".$billData[3];
               $data = array('iso'=>$isoMessage);
               $send_data = $this->directconnectionAction($agentDetails->link,$data);
            }

         }
         //Zend_Debug::dump();die;
      }

      if($callback)
      {
         $changeStauts = $this->ObjectDao->changeStauts($callback->id, $status, $callback->country_id);

         if(isset($postData['remittance']))
         {
               /**
                *
               */
               $country_code_receiver    = substr($callback->dst_MOPid, 0, 3);
               $getCountry = $this->ObjectDao->getCountry("code = '$country_code_receiver'");
               if(isset($getCountry->operator_ip))
               {
                  $operator_ip = $getCountry->operator_ip;
                  $callback->remittance = true;
                  $cdrRecord  = $this->connoperatorAction($callback, $operator_ip);

                  if($cdrRecord == 6)
                  {
                     $changeStauts = $this->ObjectDao->changeStauts($callback->id, 6, $callback->country_id);
                  }
               }
               /**
                *
               */

         }


         $value = number_format(1);
         print($value);die;
      }
      else
      {
         $value = number_format(0);
         print($value);die;
      }
   }
```

`changestatusAction` is used to get value from the bank to check if transaction needs to make any changes. We call `selectcdrclientwaiting` function to get transaction from cdr table. We call `getById` function from agent module Dao->pdo->mysql->Agent.php to get agent information depending on agent_id from agent table. We call `updateAgent` function to update agent information depending on agent_id from agent_details table. We create a connection and send information by using the `directconnectionAction` function. We call `changeStauts` function from Dao->pdo->mysql->Client.php to change the status for a cdr row depending on the value that is retrieved from the bank. `connoperatorAction` sends data to bank `addtooperator` function.

Listing A.7: SendtrxrecieverbankAction Function in File Client.php.

```
function sendtrxreciverbankAction() {

      $request    = $this->getRequest();
      $postData   = $request->getPost();

      if($postData['MOPid'])
      {
         $getClient = $this->ObjectDao->getClient($postData['MOPid']);

         $cdr_id     = $postData['id'];

         $phoneDao   = MOP_Model_Dao_Factory::getInstance()->setModule('client')->getLogDao();
              $phoneDao->setDbConnection($this->conn);
         //$cdrRecord  = $phoneDao->getById($cdr_id, true);
         $cdrRecord  = $phoneDao->getRecorde($cdr_id);

         $MOP_sender = $cdrRecord->src_MOPid;

         $country_code_receiver    = substr($postData['MOPid'], 0, 3);
         $country_code_sender      = substr($MOP_sender, 0, 3);

         //Zend_Debug::dump($cdrRecord);die('d52828d');

         if($country_code_receiver != $country_code_sender)
         //if(1)
         {
            $getCountry = $this->ObjectDao->getCountry("code = '$country_code_receiver'");

            if(isset($getCountry->operator_ip))
            {
               //$getRecord  = $phoneDao->getRecorde($cdr_id);

               $operator_ip = $getCountry->operator_ip;
               $cdrRecord  = $this->connoperatorAction($cdrRecord, $operator_ip);
            }
            //Zend_Debug::dump($getCountry);die('dd');
         }
         else
            $this->connectiontoserverAction($cdrRecord, $getClient->issuer_bank_ip);

         /**
         *  to Correspondent Bank ...
         */
         /*$cdrRecord->correspondent_bank = 1;

         $bank = $this->ObjectDao->banks(null, true);
         if(isset($bank->ip))
            $bankCoo = $this->connectiontoserverAction($cdrRecord, $bank->ip);
         */
         $value = number_format(1);
         print($value);die;
      }
      else
      {
         $value = number_format(0);
         print($value);die;
      }

      //print($ip);die;
   }
```

We send transaction to a bank by using the `sendtrxreciverbankAction` function. We call `getClient` function from Dao->pdo->mysql->Client.php to get client information from the join of client table and bank table on bank_id. We call the `getRecorde` function from the client module Dao->pdo->mysql-> Log.php to get transaction information depending. We call `getCountry` function from Dao->pdo->mysql->Client.php to get currency_code and currency_number. `connoperatorAction` is used to send data to the bank by using the `addtooperator` function. `connectiontoserverAction` is used to send data (post method) to the bank (index.php/add-client-waiting) to make call-back for the client.

Listing A.8: AddtooperatorAction Function in File Client.php.

```
function addtooperatorAction() {

     // adding the recorde in the operator in the other country (after successfully Trx)

     $request    = $this->getRequest();
     $postData   = $request->getPost();

     if(isset($postData['disposition']))
     {
         $disposition = $postData['disposition'];
     }
     else
     {
         $disposition = null;
     }

     if(isset($postData['extra_data']))
     {
         $extra_data = $postData['extra_data'];
     }
     else
     {
         $extra_data = null;
     }

      $cdr = array(
            'id'                     => $postData['cdr_id'],
            'duration'      => $postData['duration'],
            'dst_MOPid'    => $postData['dst_MOPid'],
            'src_MOPid'    => $postData['src_MOPid'],
            'issuer_bank'   => $postData['issuer_bank'],
            'receiver_bank' => $postData['receiver_bank'],
            'amount'        => $postData['amount'],
            'start'         => $postData['start'],
            'answer'        => $postData['answer'],
            'end'           => $postData['end'],
            'src'           => $postData['src'],
            'clid'          => $postData['clid'],
            'channel'       => $postData['channel'],
            'billsec'       => $postData['billsec'],
            'disposition'   => $disposition,
            'status'        => $postData['status'],
            'trx_method'    => $postData['trx_method'],
            'trx_type'      => $postData['trx_type'],
            'extra_data'    => $extra_data,
            'country_id'    => $postData['country_id']
         );

     if(isset($postData['trx_type_u']))
     {
         $updated = 1;
     }
     else
     {
         $updated = 0;
     }

     $cdr_comm = array(
                      'trx_type'        => $postData['trx_type'],
                      'client_id'       => $postData['client_id'],
                      'MOP_comm'        => $postData['MOP_comm'],
                      'bank_comm'       => $postData['bank_comm'],
                      'merchant_comm'   => $postData['merchant_comm'],
                      'amount_sender'   => $postData['amount_sender'],
                      'amount_receiver' => $postData['amount_receiver'],
                      'cdr_id'          => $postData['cdr_id'],
                      'updated'         => $updated,
                      'country_id'      => $postData['country_id']
                    );

      $cdr_comm_updated = null;
      if(isset($postData['trx_type_u']))
      {
         $cdr_comm_updated = array(
                    'trx_type'          => $postData['trx_type_u'],
                    'client_id'         => $postData['client_id_u'],
                    'MOP_comm'          => $postData['MOP_comm_u'],
                    'bank_comm'         => $postData['bank_comm_u'],
                    'amount'            => $postData['amount_u'],
                    'amount_sender'     => $postData['amount_sender_u'],
                    'cdr_id'            => $postData['cdr_id'],
                    'updated_date'      => $postData['updated_date_u'],
                    'updated_by'        => $postData['updated_by_u'],
                    'country_id'        => $postData['country_id']
                  );
    }


    /**
    *  to Correspondent Bank ...
    */
    $postData['correspondent_bank'] = 1;

    $bank = $this->ObjectDao->banks(null, true);

    if(isset($bank->ip))
       $bankCoo = $this->connectiontoserverAction($postData, $bank->ip);

    if(isset($postData['remittance']) && $bankCoo)
       $changeStauts = $this->ObjectDao->changeStauts($callback->id, 1, $callback->country_id);
    elseif(isset($postData['remittance']) && $bankCoo == 6)
    {
       $changeStauts = $this->ObjectDao->changeStauts($callback->id, 6, $callback->country_id);

       $value = number_format(6);
       print($value);die;
    }
    elseif(isset($postData['remittance']) && !$bankCoo)
       $changeStauts = $this->ObjectDao->changeStauts($callback->id, 5, $callback->country_id);


    if(!isset($postData['remittance']))
    {
       /**
       *  to Receiver Bank ...
       */
       $getClient      = $this->ObjectDao->getClient($postData['dst_MOPid']);
       if($getClient->issuer_bank_ip)
       {
          $receiver = $postData;
          $receiver['receiver_bank'] = 1;

          $receiverBankIp   = $getClient->issuer_bank_ip;
          $bankCoo = $this->connectiontoserverAction($receiver, $receiverBankIp);
       }

       //Zend_Debug::dump($bank);die('dd 54 ALA');

       $phoneDao   = MOP_Model_Dao_Factory::getInstance()->setModule('client')->getLogDao();
            $phoneDao->setDbConnection($this->conn);
       $cdrRecord  = $phoneDao->insertCdr($cdr, $cdr_comm, $cdr_comm_updated);
    } die;    }
```

Appendix B: Critical Code for the "Operator" module of the MOP Prototype

In this appendix, we give critical sections of the code in the "Operator" module.

Listing B.1: MOP Operator Helper.

```
namespace App\Helpers;
use App\Bank;
use App\Transaction;
use DateTime;
class Helper
{
   static public function callApi($method,$url,$data)
   {
    try {
      $client = new \GuzzleHttp\Client();
      if ($method == "POST") {
         $response = $client->post($url, ['form_params' => $data]);
         $reason = $response->getReasonPhrase();

         if ($reason == "OK")
            return json_decode($response->getBody()->getContents());


         return false;
      } elseif ($method == "GET") {
         $response = $client->request($method, $url);
         $reason = $response->getReasonPhrase();

         if ($reason == "OK") {
            $response = json_decode($response->getBody()->getContents());
            return $response;
         }
      }
    } catch (\Throwable $th) {
      return false ;
    }




   }

   static public function getUrlApi($bank,$operation)
   {
     $bank = Bank::find($bank);
        if($bank){
           return $bank->ip.$operation;
        }else {
           return false;
        }
   }

   static public function addTransaction($transaction)
   {
      $transactionM = Transaction::find($transaction->id);
      if(!$transactionM)
      {
         $transactionM = new Transaction;
         $transactionM->id = $transaction->id;
      }

      $transactionM->sender = $transaction->sender;
      $transactionM->is_bank = $transaction->is_bank;
      $transactionM->receiver = $transaction->receiver;
      $transactionM->status = $transaction->status;
      $transactionM->type =  $transaction->type;
```

```
       $transactionM->amount =  $transaction->amount;
       $transactionM->cancelation = $transaction->cancelation;
       $transactionM->bank =  $transaction->bank;
       $transactionM->start_time =  $transaction->start_time;
       $transactionM->end_time =  $transaction->end_time;
       $transactionM->save();

    }

    static public function getStatus($status) {
       switch($status){
          case 0 : return "Success" ;
          case 1 : return "Failed"  ;
          case 2 : return "pending"  ;
          case 3 : return "Check Customers"  ;
          case 4 : return "Ask Pin Code";
          case 5 : return "Check Pin Code";
       }
    }

    static public function sendSMS($mobileNumber,$text,$senderId = "TestApp" ){
             //API URL
             $message = urlencode($text);

$url="http://josmsservice.com/smsonline/msgservicejo.cfm?numbers=".$mobileNumber.",&senderid=FlexService&AccName=
flexserv&AccPass=flexserv123&msg=".$message."&requesttimeout=5000000";
             $response = self::callApi("GET",$url,null);
                if($response)
                 return $response;
                }
   static public   function format_interval($start_date,$end_date) {
                $first_date = new DateTime($start_date);
                $second_date = new DateTime($end_date);

          $interval = $first_date->diff($second_date);
                   $result = "";
                   if ($interval->y) { $result .= $interval->format("%y years "); }
                   if ($interval->m) { $result .= $interval->format("%m months "); }
                   if ($interval->d) { $result .= $interval->format("%d days "); }
                   if ($interval->h) { $result .= $interval->format("%h hours "); }
                   if ($interval->i) { $result .= $interval->format("%i mins "); }
                   if ($interval->s) { $result .= $interval->format("%s s "); }

                   return $result;
                }
}
```

Listing B. 2: MOP Operator API.

```
 namespace App\Http\Controllers\Api;

use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App\User;
use App\Bank;
use App\Transaction;
use App\Helpers\Helper;

use Illuminate\Support\Facades\Hash;
class ApiController extends Controller
{
    public function EditUser(Request $request){
        try {
            $user = User::find($request->id);
            $userMail = User::where('email',$request->email)->first();

            if($userMail)
              {
                  if($userMail->id != $request->id){
                   return response()->json([
                       'message' => 'Email already exsit',
                       'status' => false
                   ]);
                  }
              }
            if($user)
            {
              $user->phone_number = $request->phone_number;
              $user->national_id = $request->national_id;
              $user->name = $request->name;
              $user->save();
              return response()->json([
                'message' => 'User Edited !!',
                'status' => true
            ]);
            }else {
                return response()->json([
                    'message' => 'User Does\'nt exsit !!',
                    'status' => false
                ]);
            }
        } catch (\Throwable $th) {
            return response()->json([
                'message' => 'Error !',
                'status' => false,
                'object' => $th
            ]);
        }

    }

    public function AddUser(Request $request){
              $userc = User::where('email',$request->email)->first();
              if($userc)
              {
                return response()->json([
                    'message' => 'Email Exist !!',
                    'status' => false
                ]);
              }

              $pass = str_random(8);
              $codebank = $this->CodeBank($request->bank);
              $hashed_random_password = Hash::make($pass);


            $user =  User::create([
                'name' => $request->name,
                'email' => $request->email,
                'account_number' => $codebank,
                'password' => $hashed_random_password,
                'bank' => $request->bank,
                'phone_number' => $request->phone_number,
                'national_id' => $request->national_id,
            ]);

            $text = "Hello ".$request->name." Your Account on OperatorBank Created Successufully : your password is: ".
            $pass;
            if($request->pin_code != 0)
            $text .= "|| Your pin code is :".$request->pin_code;
            Helper::sendSMS($request->phone_number,$text);
            return response()->json([
                'message' => 'User Added !!',
                'status' => true,
                'user' => $user,
            ]);


    }

    private function CodeBank($bank) {
        return $bank."BB".uniqid();
        }


    public function deleteUser($id)
        {
          $user = User::find($id);
          if($user)
           {
             Transaction::where('sender', $id)
                   ->update(['sender' => $user->phone_number]);
             Transaction::where('receiver', $id)
                         ->update(['sender' => $user->phone_number]);
                         $user->delete();
              return response()->json([
                            'message' => 'User Deleted !!',
                            'status' => true,
                        ]);
           }else {
            return response()->json([
                'message' => 'User doesn\'t exist !!',
                'status' => false,
            ]);
           }
        }

}
```

Listing B.3: MOP operator HomeController.

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Customer;
use App\Helpers\Helper;
use Validator;
use MercurySeries\Flashy\Flashy;
use App\Bank;
use App\Transaction;
use App\User;
use Illuminate\Support\Facades\URL;
use Illuminate\Support\Facades\Hash;
use DateTime;
use Illuminate\Support\Facades\Session;
class HomeController extends Controller
{
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
      // $this->middleware('auth');
    }

    /**
     * Show the application dashboard.
     *
     * @return \Illuminate\Contracts\Support\Renderable
     */
    public function index()
    {
        $user = auth()->user();
        $data = null;

        $url = Helper::getUrlApi($user->bank,config("api.bank.getCustomer").$user->id);

        $data =   Helper::callApi("GET",$url,null);
         $status = $data->status;
          if($data->status) {

            $data =$data->customer;
            return view('home',compact("data","user","status"));
           }
          $data =$data->message;
        return view('home',compact("data","user",'status'));
    }

    public function gettransaction()
    {
        $user = auth()->user();
        $transactions = null;

        $url = Helper::getUrlApi($user->bank,config("api.bank.gettransactions").$user->id);

        $transactions =   Helper::callApi("GET",$url,null);

       $transactions = $transactions->transactions;
    //  dd($transactions);
        return view('getTransactions',compact("transactions","user"));
    }

    public function addTransaction()
    {
        $user = auth()->user();
        if(!Session::has('transaction'))
        {

            $transactionM = new Transaction;
            $transactionM->id = time();
            $transactionM->sender = $user->id;
            $transactionM->status = 2;
        $transactionM->bank = $user->bank;
        $transactionM->cancelation = "Not Started";
        $transactionM->type = "OUT";
        $transactionM->receiver = "-";
        $transactionM->amount = "-";
        $transactionM->is_bank = false;
        $transactionM->start_time = date("Y-m-d H:i:s");
        $transactionM->end_time =  date("Y-m-d H:i:s");
        $transactionM->save();
        session()->put('transaction', $transactionM->id);
     }

    $banks = Bank::get();

    return view('addTransaction',compact("user","banks"));
}

public function saveTransaction(Request $request)
{

    Validator::make($request->all(), [
        'receiver' => 'required',
        'amount' => 'required',
        'pin_code' => 'required',
    ])->validate();
        /// Check Pin Code ///
    $transaction = Transaction::find(Session::get('transaction'));
    $transaction->status = 5;
    $transaction->cancelation = "Check Pin Code";
    $transaction->save();
        /// Check Pin Code ///
    $response = null;
    $data = $request->all();
    $user = auth()->user();
    $data['sender'] = $user->id;
    $data['type'] = 'OUT';
    $data['start_time'] = date("Y-m-d H:i:s");
     $is_bank = 0 ;
    if($user->bank == $request->bank){
        $is_bank = 1;
    }

    $data['senderphone'] =  $user->phone_number;
    $data['is_bank'] = $is_bank;
    $data['idtrans'] = Session::get('transaction');
    $url = Helper::getUrlApi($user->bank,config("api.bank.addTransaction"));

        if($is_bank){
            $response = Helper::callApi("POST",$url,$data);
           if($response)
            {
                Helper::addTransaction($response->transactionOut);
                Helper::addTransaction($response->transactionIn);
            }else {
                Flashy::primary("There's a technical problem, please try again");
                  return redirect(URL::previous());
            }
          $message = $response->message;
        } else {
            //dd($data);
            /*$url = Helper::getUrlApi($req->bank,config("api.bank.checkCustomer"). $data['receiver']);
            $checkreceiver = Helper::callApi("GET",$url,null);
                if(!$checkreceiver->status)
                Flashy::primary("The Receiver doesn't Exist !");
                return redirect(URL::previous());*/
           // var_dump($data);

            $responseIn = Helper::callApi("POST",$url,$data);
            if(!$responseIn)
            {
                Flashy::primary("There's a technical problem, please try again");
                  return redirect(URL::previous());
            }
            $urlIn = Helper::getUrlApi($request->bank,config("api.bank.addTransactionin"));
            $data['sender'] = $user->phone_number;
            $data['type'] = 'IN';
            if($responseIn->status)
            {
               $responseOut = Helper::callApi("POST",$urlIn,$data);
                     if(!$responseOut)
                  {
                     $dataroll['id'] = $responseIn->transactionOut->id;
                     $UrlRoll = Helper::getUrlApi($user->bank,config("api.bank.rollbacktransaction"));
                     $rollbackCall = Helper::callApi("POST",$UrlRoll,$dataroll);

                     Flashy::primary($rollbackCall->message);
                     return redirect(URL::previous());
                  }else {
                     if($responseOut->status)
                     {
                        $Urlend = Helper::getUrlApi($user->bank,config("api.bank.endtransactionsender"));
                        $end = Helper::callApi("POST",$Urlend,$data);
                        if(!$end)
                         dd("there's a problem !!");
                     }else {
                        $dataroll['id'] = $responseIn->transactionOut->id;
                        $UrlRoll = Helper::getUrlApi($user->bank,config("api.bank.rollbacktransaction"));
                        $rollbackCall = Helper::callApi("POST",$UrlRoll,$dataroll);

                        Flashy::primary($rollbackCall->message);
                        return redirect(URL::previous());
                     }
                  }
               Helper::addTransaction($responseOut->transaction);
            }
            Helper::addTransaction($responseIn->transactionOut);
            $message = $responseIn->message;
          }


      session()->forget('transaction');

      Flashy::primary($message);
      return redirect(URL::previous());
    }

   public function profile(){
    $user = auth()->user();

    return view('profile',compact("user"));
   }

   public function editprofile(Request $request){
    $user = auth()->user();


    $obj_user = User::find($user->id);
    if($request['password'] != null)
    {
    $obj_user->password = Hash::make($request['password']);
    $obj_user->save();
    }
    Flashy::primary("Update information with success");
    return redirect(URL::previous());
   }


}
```

Listing B.4: Describe MOP Operator API Calls.

```
return [

    /*
    |--------------------------------------------------------------------------
    | Describe Api Calls
    |--------------------------------------------------------------------------
    |
    |
    |
    */

    'bank' => [
        'addCustomer' => "/api/savecustomerdetails",
        'getCustomer' => "/api/customer/",
        'gettransactions' => "/api/gettransaction/",
        'addTransaction' => "/api/addtransactionsender",
        'addTransactionin' => "/api/addtransactionreceiver",
        'checkCustomer' => "/api/checkcustomer/",
        'sendCustomerPin' => "/api/sendcustomerpin/",
        'endtransactionsender' => "/api/endtransactionsender/",
        'rollbacktransaction' => "/api/rollbacktransaction/",

    ],




];;
```

Listing B.5: Describe the MOP Operator Web route using API.

```
/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Auth::routes();

Route::post('/resetpassword', 'Auth\LoginController@resetpassword')->name('resetpassword');

Route::group(['middleware' => ['check.user','web']], function () {
    Route::get('/', 'HomeController@index')->name('home');
    Route::get('/addtransaction', 'HomeController@addTransaction')->name('transaction');
    Route::post('/addtransaction', 'HomeController@saveTransaction')->name('addtransaction');
    Route::get('/transactions', 'HomeController@gettransaction')->name('gettransaction');
    Route::post('/checkbeforesendpin', 'AjaxController@CheckBeforeSendPin')->name('checkbeforesendpin');
    Route::get('/profile', 'HomeController@profile')->name('profile');
    Route::post('/profile', 'HomeController@editprofile')->name('editprofile');

});

Route::group(['middleware' => ['check.admin','web']], function () {
    Route::get('/addbank', 'AdminController@bank')->name('bank');
    Route::post('/addbank', 'AdminController@addBank')->name('addbank');
});
```

Listing B.6: MOP Bank Helper.

```
namespace App\Helpers;
use App\Customer;

class Helper
{
   static public function callApi($method,$url,$data)
    {
        $client = new \GuzzleHttp\Client();
        if ($method == "POST") {
            $response = $client->post($url, ['form_params' => $data]);
            $reason = $response->getReasonPhrase();
            if ($reason == "OK")
                return json_decode($response->getBody()->getContents());


            return false;
        } elseif ($method == "GET") {
            $response = $client->request($method, $url);
            $reason = $response->getReasonPhrase();

            if ($reason == "OK") {

                $response = json_decode($response->getBody()->getContents());
                return $response;
            }
        }

        return true;
    }

     static public function sendSMS($mobileNumber,$text){
                $message = urlencode($text);
                //API URL

$url="http://josmsservice.com/smsonline/msgservicejo.cfm?numbers=".$mobileNumber.",&senderid=FlexService&AccName=
flexserv&AccPass=flexserv123&msg=".$message."&requesttimeout=5000000";
                $response = self::callApi("GET",$url,null);
                    if($response)
                     return $response;

                // init the resource
               // dd($response);
                //Print error if any
                    }
    static public function SmsText($req,$status,$message,$whome = "sender"){

                if($whome == "sender")
                {
                    if($status ==0)
                        return "the transaction Completed with Success, You transfer ".$req->amount. " To ".
                                 $req->receiver;
                    else {
                        return "The transaction failed Cause: ".$message;
                           }
                    }else {
                     $sender = Customer::find($req->sender);
                       return "You recieved  ".$req->amount. " USD From : ".$req->phone_number;
                }
            }
    static public function CheckPin($pin_code,$sender){
                 if($sender->pin_code != 0)
                 {
                    if($sender->pin_code == $pin_code)
                        return true;
                    else
                        return false;
                 }else{
                    if($sender->last_pin_code == $pin_code)
                        return true;
                    else
                        return false;
                 }
            }
}
```

Listing B.7: MOP Bank API.

```
namespace App\Http\Controllers\Api;

use App\Customer;
use App\Transaction;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App\Helpers\Helper;
use DateTime;
class BankController extends Controller
{

    public function add_customer(Request $request){

        $customer = new Customer;

        $customer->id =  $request->id;
        $customer->name =  $request->name;
        $customer->account_number = $request->account_number;
        $customer->phone_number = $request->phone_number;
        $customer->national_id = $request->national_id;
        $customer->credit =$request->credit;
        $customer->email = $request->email;
        $customer->pin_code = $request->pin_code;
        $customer->limit_transfer = $request->limit_transfer;
        $customer->save();

        return response()->json([
            'message' => 'Customer Added !!',
            'customer' => $customer,
            'status' => true
        ]);

    }

    public function get_customer($id){



        $customerEntity =  Customer::find($id);
            if($customerEntity){
                return response()->json([
                    'customer' => $customerEntity,
                    'status' => true
                ]);
            }else {
                return response()->json([
                    'message' => 'Customer Doesn\'t find !!',
                    'status' => false
                ]);
            }


    }

     public function check_customer($phone){

        $receiver = Customer::where('phone_number',$phone)->first();
            if($receiver){
                return response()->json([
                    'customer' => $receiver,
                    'status' => true
                ]);
            }else {
                return response()->json([
                    'message' => 'Customer Doesn\'t find !!',
                      'status' => false
                ]);
            }


    }

    public function get_transactions($id){
```

```
    $transactionEntity =  Transaction::with('senderC')->with('receiverC')->where('receiver',$id)->orWhere('sender',$id)->get();
        if($transactionEntity->first()){
            return response()->json([
                'status'  => true,
                'transactions' => $transactionEntity
            ]);
        }else {
            return response()->json([
                'status'  => true,
                'transactions' => [],
            ]);
        }


}

public function add_transactionSender(Request $request) {

    $customerM = new Customer;
    $transactionM = new Transaction;
    $message = "Transaction Added  Succesfully";
    $status = " ";
    $cstmr =  $customerM->find($request->sender);
    $receiver =  $request->receiver ;
    $isOnprocess = true;
    $transactionIn = null;
    $state = false;
    if(Helper::CheckPin($request->pin_code,$cstmr))
    {
        if($request->is_bank)
        {
            $receiver = $customerM->where('phone_number',$request->receiver)->first();
                if ($receiver) {
                    $receiver = $receiver->id;
                   // $this->addtransactionreceiver($request);
                } else {
                    $message = "The Receiver Account doesn't exist ";
                    $status  = 1;
                    $receiver = $request->receiver;
                    $isOnprocess = false ;
                }
        }
        if($isOnprocess){

            if( $cstmr->limit_transfer != 0 && $request->amount  >=  $cstmr->limit_transfer)
            {
                $message = "You across amount of limit transfer";
                $status  = 1;
            }else {
        if ($cstmr->credit >= $request->amount)
        {
            $cstmr->update(["credit" => $cstmr->credit - $request->amount ]);
            $status  = 0;
              $state = true;
            if($request->is_bank)
                    {
                        $transactionIn = $this->addtransactionreceiver($request,$cstmr,$receiver,true);
                    }
        }else {
            $message = "You don't have enough money to make this transaction";
            $status  = 1;
        }
    }
                    }
    }else {
        $message = "The pin code is wrong !";
        $status  = 1;
    }
    $transactionM->id = $request->idtrans;
    $transactionM->sender = $cstmr->id;
    $transactionM->is_bank = $request->is_bank;
    $transactionM->receiver = $receiver;
    $transactionM->status = $status;
    $transactionM->type = $request->type;
```

```
      $transactionM->amount = $request->amount;
      $transactionM->start_time = $request->start_time;
//    $transactionM->end_time =new DateTime();
      $transactionM->end_time = date("Y-m-d H:i:s");
      $transactionM->cancelation =$message;
      $transactionM->bank =  $request->bank;
      $transactionM->save();
      $text = Helper::SmsText($request,$status,$message);
      Helper::sendSMS($cstmr->phone_number,$text);
          if($transactionM->status == 0)
          {
            $text = Helper::SmsText($request,null,null,"receiver");
             Helper::sendSMS($request->receiver,$text);
          }


      return response()->json([
          'message' => $message,
          'transactionOut' =>$transactionM,
          'transactionIn' => $transactionIn,
          'status' => $state
      ]);
  }

  public function addtransactionreceiver($request,$cstmr,$receiver,$isIn)
  {
      $customerM = new Customer;
      $transactionM = new Transaction;
      if($isIn == true ){
          $receiverE = $customerM->find($receiver);
          $receiverE->update(["credit" => $receiverE->credit + $request->amount ]);
      $transactionM->id = time();
      $transactionM->sender = $cstmr->id;
      $transactionM->is_bank = $request->is_bank;
      $transactionM->receiver = $receiver;
      $transactionM->status = 0;
      $transactionM->type = 'IN';
      $transactionM->amount = $request->amount;
      $transactionM->start_time = $request->start_time;
      $transactionM->end_time = date("Y-m-d H:i:s");
      $transactionM->cancelation =' ';
      $transactionM->bank =  $request->bank;
      $transactionM->save();
      return $transactionM;
      }else {
       $receiverE = $customerM->where('phone_number',$request->receiver)->first();
          $receiverE->update(["credit" => $receiverE->credit + $request->amount ]);

      $transactionM->sender = $request->sender;
      $transactionM->is_bank = $request->is_bank;
      $transactionM->receiver = $receiverE->id;
      $transactionM->status =0;
      $transactionM->type = 'IN';
      $transactionM->start_time = date("Y-m-d H:i:s");
      $transactionM->end_time = date("Y-m-d H:i:s");
      $transactionM->amount = $request->amount;
      $transactionM->cancelation =' ';
      $transactionM->bank =  $request->bank;
      $transactionM->save();

      return $transactionM;
      }
  }

   public function add_transactionReceiver(Request $request){


     $transactionM =  $this->addtransactionreceiver($request,null,null,false);
          if($transactionM){
              $text = Helper::SmsText($request,null,null,"receiver");
             return response()->json([
                 'status'  => true,
                 'transaction' => $transactionM
             ]);
          }else {
            return response()->json([
                'status'  => false,
                'message' => "There's a problem!!!",
            ]);
        }


}
public function endtransactionsender($request){
  $customerM = new Customer;
  $cstmr =  $customerM->find($request->sender);
  if($cstmr)
  {
    $cstmr->update(["credit" => $cstmr->credit - $request->amount ]);
    return response()->json([
        'message' => "transaction end with success",
        'status' => true
    ]);
  }else {
    return response()->json([
        'message' => "customer doesn't exsit ",
        'status' => false
    ]);
  }

}
public function rollbacktransaction($request){
    $transactionM = new Transaction;
  $transaction =  $transaction->find($request->id);
  if($transaction)
  {
    $transaction->update(["status" => 1,'cancelation' => "Transaction failed, Technical Error!"]);
    return response()->json([
        'message' => "Transaction failed, Technical Error!",
        'status' => true
    ]);
  }else {
    return response()->json([
        'message' => "transaction doesn't exsit ",
        'status' => false
    ]);
  }

}
public function sendCustomerPin($phone){

     $sender = Customer::where('phone_number',$phone)->first();
        if($sender){
              if($sender->pin_code != 0)
              {
              $text = "The pin code for the transaction is : ". $sender->pin_code;
                 Helper::sendSMS($phone,$text);
              }else {
                $length = 4;
                $randomletter = substr(str_shuffle("912837465"), 0, $length);
                $sender->last_pin_code = $randomletter ;
                  $sender->save();
                    $text = "The pin code for the transaction is : ". $sender->last_pin_code;
                 Helper::sendSMS($phone,$text);
              }
                    return response()->json([
                           'status' => true,
                           'pin_code' => $sender->last_pin_code
                       ]);

        }else {
            return response()->json([
                'message' => 'Customer Doesn\'t find !!',
                  'status' => false
            ]);
        }
         return response()->json([
                'message' => 'Error !!',
                  'status' => false
            ]); }}
```

Listing B.8: MOP Bank Customer Controller.

```
 namespace App\Http\Controllers;
use App\Customer;
use App\Transaction;
use Illuminate\Http\Request;
use Validator;
use App\Helpers\Helper;
use MercurySeries\Flashy\Flashy;
use Illuminate\Support\Facades\URL;
class CustomerController extends Controller
{

     /**
      * Create a new controller instance.
      *
      * @return void
      */
     public function __construct()
     {
         $this->middleware('auth');
     }

   /**
    * Display a listing of the resource.
    *
    * @return \Illuminate\Http\Response
    */
   public function index()
   {
        $customers = Customer::All();
        return view('customers',compact("customers"));
   }

   /**
    * Show the form for creating a new resource.
    *
    * @return \Illuminate\Http\Response
    */
   public function create()
   {
        return view('add');
   }

   /**
    * Store a newly created resource in storage.
    *
    * @param  \Illuminate\Http\Request  $request
    * @return \Illuminate\Http\Response
    */
   public function store(Request $request)
   {
      Validator::make($request->all(), [
      'name' => ['required', 'string', 'max:255'],
      'pin_code' => ['required', 'string', 'max:6'],
      'phone_number' => ['required', 'string', 'max:255'],
      'national_id' => ['required', 'string', 'max:255'],
      'credit' => ['required', 'Numeric'],
      'limit_transfer' => ['required', 'Numeric'],
      'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
        ])->validate();
        $data =$request->all();
        $data['bank'] = 1 ;
        $customer = new Customer;
        $url = env("URL_OPERATOR")."adduser";
        $response = Helper::callApi("POST",$url,$data);
        if($response){
            if($response->status)
            {
              $customer->id =  $response->user->id;
              $customer->name =  $request->name;
              $customer->account_number = $response->user->account_number;
              $customer->phone_number = $request->phone_number;
              $customer->national_id = $request->national_id;
              $customer->credit =$request->credit;
              $customer->email = $request->email;
            $customer->pin_code = $request->pin_code;
            $customer->limit_transfer = $request->limit_transfer;
            $customer->save();
            Flashy::primary("Customer Added with success ");
                return redirect(URL::previous());
          }else {
            Flashy::primary($response->message);
                return redirect(URL::previous());
          }
      }else {
        Flashy::primary("There's a technical problem, please try again");
                return redirect(URL::previous());
      }


  }

  /**
   * Display the specified resource.
   *
   * @param  int  $id
   * @return \Illuminate\Http\Response
   */
  public function show($id)
  {
    $customer = Customer::find($id);
    if($customer)
    return view("show",compact("customer"));

    abort(404);
  }

  /**
   * Show the form for editing the specified resource.
   *
   * @param  int  $id
   * @return \Illuminate\Http\Response
   */
   public function edit($id)
   {
      $customer = Customer::find($id);
      if($customer)
      return view("edit",compact("customer"));

      abort(404);
   }

   /**
    * Update the specified resource in storage.
    *
    * @param  \Illuminate\Http\Request  $request
    * @param  int  $id
    * @return \Illuminate\Http\Response
    */
   public function update(Request $request, $id)
   {
     $customer =  Customer::find($id);
     Validator::make($request->all(), [
     'name' => ['required', 'string', 'max:255'],
     'pin_code' => ['required', 'string', 'max:6'],
     'phone_number' => ['required', 'string', 'max:255'],
     'national_id' => ['required', 'string', 'max:255'],
     'credit' => ['required', 'Numeric'],
     'limit_transfer' => ['required', 'Numeric'],
     'email' => ['required', 'string', 'email', 'max:255', 'unique:users,email,'.$customer->email],
        ])->validate();
        $data =$request->all();
        $data['id'] =$id;

        $url = env("URL_OPERATOR")."edituser";
        $response = Helper::callApi("POST",$url,$data);
        if($response){
           if($response->status)
           {
             $customer->name =  $request->name;
             $customer->phone_number = $request->phone_number;
             $customer->national_id = $request->national_id;
             $customer->credit =$request->credit;
             $customer->email = $request->email;
             $customer->pin_code = $request->pin_code;
             $customer->limit_transfer = $request->limit_transfer;
             $customer->save();
             Flashy::primary("Customer Edited with success ");
                 return redirect(URL::previous());
           }else {
             Flashy::primary($response->message);
                 return redirect(URL::previous());
           }
        }else {
          Flashy::primary("There's a technical problem, please try again");
                 return redirect(URL::previous());
        }
   }

   /**
    * Remove the specified resource from storage.
    *
    * @param  int  $id
    * @return \Illuminate\Http\Response
    */
   public function destroy($id)
   {
     $customer = Customer::find($id);
     if($customer)
      {
        Transaction::where('sender', $id)
             ->update(['sender' => $customer->phone_number]);
        Transaction::where('receiver', $id)
                 ->update(['sender' => $customer->phone_number]);
        $url = env("URL_OPERATOR")."deleteuser/".$id;
        $response = Helper::callApi("GET",$url,null);
           if($response) {
              if($response->status)
              {
                 $customer->delete();
                 Flashy::primary("Customer Edited with success ");
                     return redirect(URL::previous());
              }else {
                 Flashy::primary($response->message);
                     return redirect(URL::previous());
              }
           }else {
             Flashy::primary("There's a technical problem, please try again");
                     return redirect(URL::previous());
           }
      }

           abort(404);
   }
}
```

Listing B.9: MOP Bank API Calls

```
        use Illuminate\Http\Request;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| is assigned the "api" middleware group. Enjoy building your API!
|
*/

   Route::get('/customer/{id}', 'Api\BankController@get_customer')->name('getcostumer');
   Route::get('/checkcustomer/{phone}', 'Api\BankController@check_customer')->name('checkcustomer');
   Route::get('/sendcustomerpin/{phone}', 'Api\BankController@sendCustomerPin')->name('sendcustomerpin');
   Route::post('/savecustomerdetails', 'Api\BankController@add_customer')->name('addcostumer');
   Route::post('/endtransactionsender', 'Api\BankController@end_transactionSender')->name('end_transactionSender');
   Route::post('/rollbacktransaction', 'Api\BankController@rollbacktransaction')->name('rollbacktransaction');
   Route::post('/addtransactionsender', 'Api\BankController@add_transactionSender')->name('addtransactionsender');
   Route::get('/gettransaction/{id}', 'Api\BankController@get_transactions')->name('gettransaction');
   Route::post('/addtransactionreceiver', 'Api\BankController@add_transactionReceiver')->name('addtransactionreceiver');
```

Appendix C: Critical Code for the “Bank” module of the MOP Prototype

In this appendix, we give critical sections of the code in the “Bank” module.

Listing C.1: MOP Bank Helper.

```
namespace App\Helpers;
use App\Customer;

class Helper
{
  static public function callApi($method,$url,$data)
   {
      $client = new \GuzzleHttp\Client();
      if ($method == "POST") {
          $response = $client->post($url, ['form_params' => $data]);
          $reason = $response->getReasonPhrase();
          if ($reason == "OK")
              return json_decode($response->getBody()->getContents());


          return false;
      } elseif ($method == "GET") {
          $response = $client->request($method, $url);
          $reason = $response->getReasonPhrase();

          if ($reason == "OK") {

              $response = json_decode($response->getBody()->getContents());
              return $response;
          }
      }

      return true;
   }

    static public function sendSMS($mobileNumber,$text){
              $message = urlencode($text);
              //API URL

$url="http://josmsservice.com/smsonline/msgservicejo.cfm?numbers=".$mobileNumber.",&senderid=FlexService&AccName=
flexserv&AccPass=flexserv123&msg=".$message."&requesttimeout=5000000";
              $response = self::callApi("GET",$url,null);
                  if($response)
                   return $response;

              // init the resource
             // dd($response);
              //Print error if any
                  }
   static public function SmsText($req,$status,$message,$whome = "sender"){

              if($whome == "sender")
              {
                  if($status ==0)
                      return "the transaction Completed with Success, You transfer ".$req->amount. " To ".
                              $req->receiver;
                  else {
                      return "The transaction failed Cause: ".$message;
                        }
                  }else {
                   $sender = Customer::find($req->sender);
                    return "You recieved  ".$req->amount. " USD From : ".$req->phone_number;
              }
          }
   static public function CheckPin($pin_code,$sender){
               if($sender->pin_code != 0)
               {
                  if($sender->pin_code == $pin_code)
                      return true;
                  else
                      return false;
               }else{
                  if($sender->last_pin_code == $pin_code)
                      return true;
                  else
                      return false;
               }
          }
}
```

Listing C.2: MOP Bank API.

```
namespace App\Http\Controllers\Api;

use App\Customer;
use App\Transaction;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App\Helpers\Helper;
use DateTime;
class BankController extends Controller
{

    public function add_customer(Request $request){

        $customer = new Customer;

        $customer->id =  $request->id;
        $customer->name =  $request->name;
        $customer->account_number = $request->account_number;
        $customer->phone_number = $request->phone_number;
        $customer->national_id = $request->national_id;
        $customer->credit =$request->credit;
        $customer->email = $request->email;
        $customer->pin_code = $request->pin_code;
        $customer->limit_transfer = $request->limit_transfer;
        $customer->save();

        return response()->json([
            'message' => 'Customer Added !!',
            'customer' => $customer,
            'status' => true
        ]);

    }

    public function get_customer($id){



        $customerEntity =  Customer::find($id);
            if($customerEntity){
                return response()->json([
                    'customer' => $customerEntity,
                    'status' => true
                ]);
            }else {
                return response()->json([
                    'message' => 'Customer Doesn\'t find !!',
                    'status' => false
                ]);
            }


    }

     public function check_customer($phone){

        $receiver = Customer::where('phone_number',$phone)->first();
            if($receiver){
                return response()->json([
                    'customer' => $receiver,
                    'status' => true
                ]);
            }else {
                return response()->json([
                    'message' => 'Customer Doesn\'t find !!',
                      'status' => false
                ]);
            }


    }

    public function get_transactions($id){
```

```
    $transactionEntity =  Transaction::with('senderC')->with('receiverC')->where('receiver',$id)->orWhere('sender',$id)->get();
        if($transactionEntity->first()){
            return response()->json([
                'status'  => true,
                'transactions' => $transactionEntity
            ]);
        }else {
            return response()->json([
                'status'  => true,
                'transactions' => [],
            ]);
        }


}

public function add_transactionSender(Request $request) {

    $customerM = new Customer;
    $transactionM = new Transaction;
    $message = "Transaction Added  Succesfully";
    $status = " ";
    $cstmr =  $customerM->find($request->sender);
    $receiver =  $request->receiver ;
    $isOnprocess = true;
    $transactionIn = null;
    $state = false;
    if(Helper::CheckPin($request->pin_code,$cstmr))
    {
        if($request->is_bank)
        {
            $receiver = $customerM->where('phone_number',$request->receiver)->first();
                if ($receiver) {
                    $receiver = $receiver->id;
                   // $this->addtransactionreceiver($request);
                } else {
                    $message = "The Receiver Account doesn't exist ";
                    $status  = 1;
                    $receiver = $request->receiver;
                    $isOnprocess = false ;
                }
        }
        if($isOnprocess){

            if( $cstmr->limit_transfer != 0 && $request->amount  >=  $cstmr->limit_transfer)
            {
                $message = "You across amount of limit transfer";
                $status  = 1;
            }else {
        if ($cstmr->credit >= $request->amount)
        {
            $cstmr->update(["credit" => $cstmr->credit - $request->amount ]);
            $status  = 0;
              $state = true;
            if($request->is_bank)
                    {
                        $transactionIn = $this->addtransactionreceiver($request,$cstmr,$receiver,true);
                    }
        }else {
            $message = "You don't have enough money to make this transaction";
            $status  = 1;
        }
    }
                    }
    }else {
        $message = "The pin code is wrong !";
        $status  = 1;
    }
    $transactionM->id = $request->idtrans;
    $transactionM->sender = $cstmr->id;
    $transactionM->is_bank = $request->is_bank;
    $transactionM->receiver = $receiver;
    $transactionM->status = $status;
    $transactionM->type = $request->type;
    $transactionM->amount = $request->amount;
    $transactionM->start_time = $request->start_time;
//    $transactionM->end_time =new DateTime();
    $transactionM->end_time = date("Y-m-d H:i:s");
    $transactionM->cancelation =$message;
    $transactionM->bank =  $request->bank;
    $transactionM->save();
    $text = Helper::SmsText($request,$status,$message);
    Helper::sendSMS($cstmr->phone_number,$text);
        if($transactionM->status == 0)
        {
           $text = Helper::SmsText($request,null,null,"receiver");
            Helper::sendSMS($request->receiver,$text);
        }


    return response()->json([
        'message' => $message,
        'transactionOut' =>$transactionM,
        'transactionIn' => $transactionIn,
        'status' => $state
    ]);
 }

 public function addtransactionreceiver($request,$cstmr,$receiver,$isIn)
 {
    $customerM = new Customer;
    $transactionM = new Transaction;
    if($isIn == true ){
        $receiverE = $customerM->find($receiver);
        $receiverE->update(["credit" => $receiverE->credit + $request->amount ]);
    $transactionM->id = time();
    $transactionM->sender = $cstmr->id;
    $transactionM->is_bank = $request->is_bank;
    $transactionM->receiver = $receiver;
    $transactionM->status = 0;
    $transactionM->type = 'IN';
    $transactionM->amount = $request->amount;
    $transactionM->start_time = $request->start_time;
    $transactionM->end_time = date("Y-m-d H:i:s");
    $transactionM->cancelation =' ';
    $transactionM->bank =  $request->bank;
    $transactionM->save();
    return $transactionM;
    }else {
     $receiverE = $customerM->where('phone_number',$request->receiver)->first();
        $receiverE->update(["credit" => $receiverE->credit + $request->amount ]);

    $transactionM->sender = $request->sender;
    $transactionM->is_bank = $request->is_bank;
    $transactionM->receiver = $receiverE->id;
    $transactionM->status =0;
    $transactionM->type = 'IN';
    $transactionM->start_time = date("Y-m-d H:i:s");
    $transactionM->end_time = date("Y-m-d H:i:s");
    $transactionM->amount = $request->amount;
    $transactionM->cancelation =' ';
    $transactionM->bank =  $request->bank;
    $transactionM->save();

    return $transactionM;
    }
 }

  public function add_transactionReceiver(Request $request){


   $transactionM =  $this->addtransactionreceiver($request,null,null,false);
        if($transactionM){
            $text = Helper::SmsText($request,null,null,"receiver");
           return response()->json([
               'status'  => true,
               'transaction' => $transactionM
           ]);
        }else {
            return response()->json([
                'status'  => false,
                'message' => "There's a problem!!!",
            ]);
        }


}
public function endtransactionsender($request){
  $customerM = new Customer;
  $cstmr =  $customerM->find($request->sender);
  if($cstmr)
  {
    $cstmr->update(["credit" => $cstmr->credit - $request->amount ]);
    return response()->json([
        'message' => "transaction end with success",
        'status' => true
    ]);
  }else {
    return response()->json([
        'message' => "customer doesn't exsit ",
        'status' => false
    ]);
  }
}

public function rollbacktransaction($request){
    $transactionM = new Transaction;
  $transaction =  $transaction->find($request->id);
  if($transaction)
  {
    $transaction->update(["status" => 1,'cancelation' => "Transaction failed, Technical Error!"]);
    return response()->json([
        'message' => "Transaction failed, Technical Error!",
        'status' => true
    ]);
  }else {
    return response()->json([
        'message' => "transaction doesn't exsit ",
        'status' => false
    ]);
  }
}
public function sendCustomerPin($phone){

     $sender = Customer::where('phone_number',$phone)->first();
        if($sender){
              if($sender->pin_code != 0)
              {
              $text = "The pin code for the transaction is : ". $sender->pin_code;
                 Helper::sendSMS($phone,$text);
              }else {
                $length = 4;
                $randomletter = substr(str_shuffle("912837465"), 0, $length);
                $sender->last_pin_code = $randomletter ;
                  $sender->save();
                    $text = "The pin code for the transaction is : ". $sender->last_pin_code;
                 Helper::sendSMS($phone,$text);
              }
                     return response()->json([
                            'status' => true,
                            'pin_code' => $sender->last_pin_code
                        ]);
        }else {
            return response()->json([
                'message' => 'Customer Doesn\'t find !!',
                  'status' => false
            ]);
        }

          return response()->json([
                'message' => 'Error !!',
                  'status' => false
            ]);
}}
```

Listing C.3: MOP Bank Customer Controller.

```
 namespace App\Http\Controllers;
use App\Customer;
use App\Transaction;
use Illuminate\Http\Request;
use Validator;
use App\Helpers\Helper;
use MercurySeries\Flashy\Flashy;
use Illuminate\Support\Facades\URL;
class CustomerController extends Controller
{

     /**
      * Create a new controller instance.
      *
      * @return void
      */
     public function __construct()
     {
         $this->middleware('auth');
     }

   /**
    * Display a listing of the resource.
    *
    * @return \Illuminate\Http\Response
    */
   public function index()
   {
        $customers = Customer::All();
        return view('customers',compact("customers"));
   }

   /**
    * Show the form for creating a new resource.
    *
    * @return \Illuminate\Http\Response
    */
   public function create()
   {
        return view('add');
   }

   /**
    * Store a newly created resource in storage.
    *
    * @param  \Illuminate\Http\Request  $request
    * @return \Illuminate\Http\Response
    */
   public function store(Request $request)
   {
     Validator::make($request->all(), [
     'name' => ['required', 'string', 'max:255'],
     'pin_code' => ['required', 'string', 'max:6'],
     'phone_number' => ['required', 'string', 'max:255'],
     'national_id' => ['required', 'string', 'max:255'],
     'credit' => ['required', 'Numeric'],
     'limit_transfer' => ['required', 'Numeric'],
     'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
        ])->validate();
        $data =$request->all();
        $data['bank'] = 1 ;
        $customer = new Customer;
        $url = env("URL_OPERATOR")."adduser";
        $response = Helper::callApi("POST",$url,$data);
        if($response){
            if($response->status)
            {
              $customer->id =  $response->user->id;
              $customer->name =  $request->name;
              $customer->account_number = $response->user->account_number;
              $customer->phone_number = $request->phone_number;
              $customer->national_id = $request->national_id;
              $customer->credit =$request->credit;
              $customer->email = $request->email;
            $customer->pin_code = $request->pin_code;
            $customer->limit_transfer = $request->limit_transfer;
            $customer->save();
            Flashy::primary("Customer Added with success ");
                return redirect(URL::previous());
          }else {
            Flashy::primary($response->message);
                return redirect(URL::previous());
          }
      }else {
        Flashy::primary("There's a technical problem, please try again");
                return redirect(URL::previous());
      }


  }

  /**
   * Display the specified resource.
   *
   * @param  int  $id
   * @return \Illuminate\Http\Response
   */
  public function show($id)
  {
    $customer = Customer::find($id);
    if($customer)
    return view("show",compact("customer"));

    abort(404);
  }

  /**
   * Show the form for editing the specified resource.
   *
   * @param  int  $id
   * @return \Illuminate\Http\Response
   */
   public function edit($id)
   {
      $customer = Customer::find($id);
      if($customer)
      return view("edit",compact("customer"));

      abort(404);
   }

   /**
    * Update the specified resource in storage.
    *
    * @param  \Illuminate\Http\Request  $request
    * @param  int  $id
    * @return \Illuminate\Http\Response
    */
   public function update(Request $request, $id)
   {
     $customer =  Customer::find($id);
     Validator::make($request->all(), [
     'name' => ['required', 'string', 'max:255'],
     'pin_code' => ['required', 'string', 'max:6'],
     'phone_number' => ['required', 'string', 'max:255'],
     'national_id' => ['required', 'string', 'max:255'],
     'credit' => ['required', 'Numeric'],
     'limit_transfer' => ['required', 'Numeric'],
     'email' => ['required', 'string', 'email', 'max:255', 'unique:users,email,'.$customer->email],
        ])->validate();
        $data =$request->all();
        $data['id'] =$id;

        $url = env("URL_OPERATOR")."edituser";
        $response = Helper::callApi("POST",$url,$data);
        if($response){
          if($response->status)
          {
            $customer->name =  $request->name;
            $customer->phone_number = $request->phone_number;
            $customer->national_id = $request->national_id;
            $customer->credit =$request->credit;
            $customer->email = $request->email;
            $customer->pin_code = $request->pin_code;
            $customer->limit_transfer = $request->limit_transfer;
            $customer->save();
            Flashy::primary("Customer Edited with success ");
                return redirect(URL::previous());
          }else {
            Flashy::primary($response->message);
                return redirect(URL::previous());
          }
        }else {
          Flashy::primary("There's a technical problem, please try again");
                return redirect(URL::previous());
        }
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
     $customer = Customer::find($id);
     if($customer)
      {
        Transaction::where('sender', $id)
             ->update(['sender' => $customer->phone_number]);
        Transaction::where('receiver', $id)
                 ->update(['sender' => $customer->phone_number]);
        $url = env("URL_OPERATOR")."deleteuser/".$id;
        $response = Helper::callApi("GET",$url,null);
           if($response) {
              if($response->status)
              {
                $customer->delete();
                Flashy::primary("Customer Edited with success ");
                    return redirect(URL::previous());
              }else {
                Flashy::primary($response->message);
                    return redirect(URL::previous());
              }
           }else {
             Flashy::primary("There's a technical problem, please try again");
                    return redirect(URL::previous());
           }
      }

          abort(404);
    }
}
```

Listing C.4: MOP Bank API Calls

```
 use Illuminate\Http\Request;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
*/

    Route::get('/customer/{id}', 'Api\BankController@get_customer')->name('getcostumer');
    Route::get('/checkcustomer/{phone}', 'Api\BankController@check_customer')->name('checkcustomer');
    Route::get('/sendcustomerpin/{phone}', 'Api\BankController@sendCustomerPin')->name('sendcustomerpin');
    Route::post('/savecustomerdetails', 'Api\BankController@add_customer')->name('addcostumer');
    Route::post('/endtransactionsender', 'Api\BankController@end_transactionSender')->name('end_transactionSender');
    Route::post('/rollbacktransaction', 'Api\BankController@rollbacktransaction')->name('rollbacktransaction');
    Route::post('/addtransactionsender', 'Api\BankController@add_transactionSender')->name('addtransactionsender');
    Route::get('/gettransaction/{id}', 'Api\BankController@get_transactions')->name('gettransaction');
    Route::post('/addtransactionreceiver', 'Api\BankController@add_transactionReceiver')->name('addtransactionreceiver');
```

Appendix D: Simulator for the MOP system

In this appendix, we describe a simulator for the MOP system, which allows one to get a feel for using the actual system. Our simulator carries out account creation and transaction processing through the Internet. It makes use of four web services: one for the MOP operator, two for MOP banks, and one for the central bank. In the simulation, we have two banks and operator, as shown in figure D.1.

In order to create an account for a new user, the details of the customer and his account are fed to the bank website through a registration form, which is validated by the bank prior to addition to the database. Once the request is approved, the user can remit the amount and open the account. All operations are done by using web services. In order to transfer funds, the bank application gives the user ID and password in order to perform online transactions. These are explained later in this appendix.

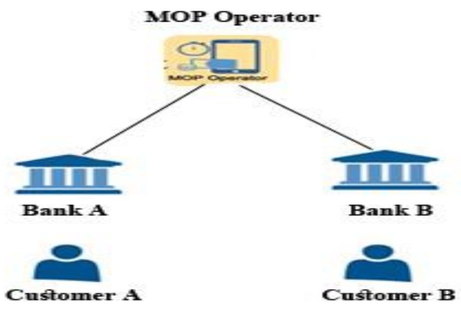


Figure D.1: Simulation scenario

D.1 Web Services Overview

A Web service is software designed to perform a set of tasks which is accessed using standard Internet technology. It provides a standardized way to communicate between the client and server applications on the internet. In our simulation we use the RESTfull Application Programming Interface (API's), which is based Representational State Transfer (REST) technology. figure D.2 shows how a RESTfull web service works.

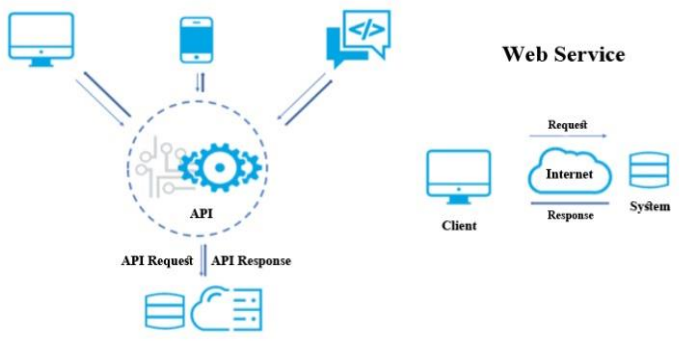


Figure D.2: Web service architecture.

D.2 Web Service Methods for MOP Operator

- `Adduser(name, email, bank, account_number, phone_number, national_id)`: used add a user to the operator database.
- `deleteUser(id)`: used to delete a user from operator database.
- `editUser(name, email, bank, account_number, phone_number, national_id )`: used to change user information in operator database.

D.3 Web Service Methods for MOP Bank

- `addCustomer (id, name, account_number, phone_number, national_id, credit, email, pin_code, limit_transfer, last_pin_code)`: used to add a customer to the bank database.
- `checkCustomer(phone_number)`: used to check customer by phone number.
- `getCustomer(id)`: it is used to get a customer details by customer ID.
- `sendPinCode (phone_number)`: used to send PIN for customer by phone number.
- `addtransactionsender (id, sender, receiver, is_bank, bank, amount, status, type, cancelation, start_time, end_time)`: used to add transaction to the sender bank.
- `addReceiverTransaction (id, sender, receiver, is_bank, bank, amount, status, type, cancelation, start_time, end_time)`: used to add receiver transaction to the bank.
- `addtransaction (id, senderbank, receiverbank, status, cancelation, start_time, end_time)`: used to add a transaction to the central bank.
- `endTransaction (id, sender, receiver, is_bank, bank, amount, status, type, cancelation, start_time, end_time)`: used to end transaction by edit customers credit.
- `getTransaction (id)`: used to get a transaction details by customer ID number.

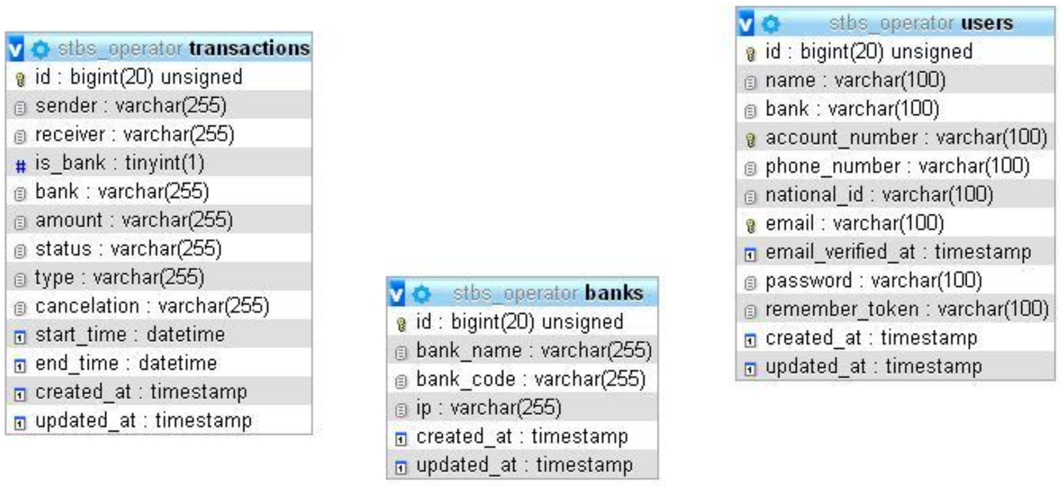
- `rollbackTransaction (id, sender, receiver, is_bank, bank, amount, status, type, cancelation, start_time, end_time)`: used to rollback the transaction for some technical error.

D.4 Web Service Methods for Central Bank

- `addtransaction (id, senderbank, receiverbank, status, cancelation, start_time, end_time)`: used to add a transaction to the central bank database.

D.5 Database Structure for MOP Components

Figures D.3, D.4 and D.5 depict the database structures for MOP operator, MOP bank and Central bank respectively.

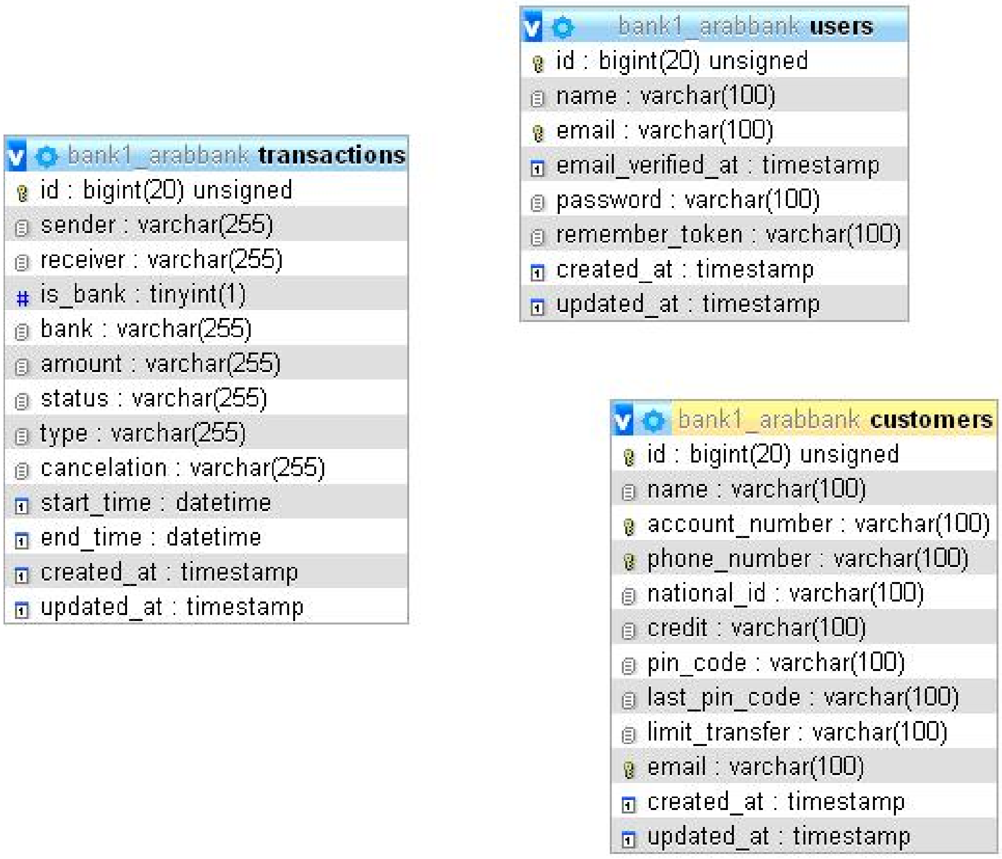


```
stbs_operator transactions
id : bigint(20) unsigned
sender : varchar(255)
receiver : varchar(255)
is_bank : tinyint(1)
bank : varchar(255)
amount : varchar(255)
status : varchar(255)
type : varchar(255)
cancelation : varchar(255)
start_time : datetime
end_time : datetime
created_at : timestamp
updated_at : timestamp

stbs_operator banks
id : bigint(20) unsigned
bank_name : varchar(255)
bank_code : varchar(255)
ip : varchar(255)
created_at : timestamp
updated_at : timestamp

stbs_operator users
id : bigint(20) unsigned
name : varchar(100)
bank : varchar(100)
account_number : varchar(100)
phone_number : varchar(100)
national_id : varchar(100)
email : varchar(100)
email_verified_at : timestamp
password : varchar(100)
remember_token : varchar(100)
created_at : timestamp
updated_at : timestamp
```

Figure D. 3: Database structure for MOP operator.

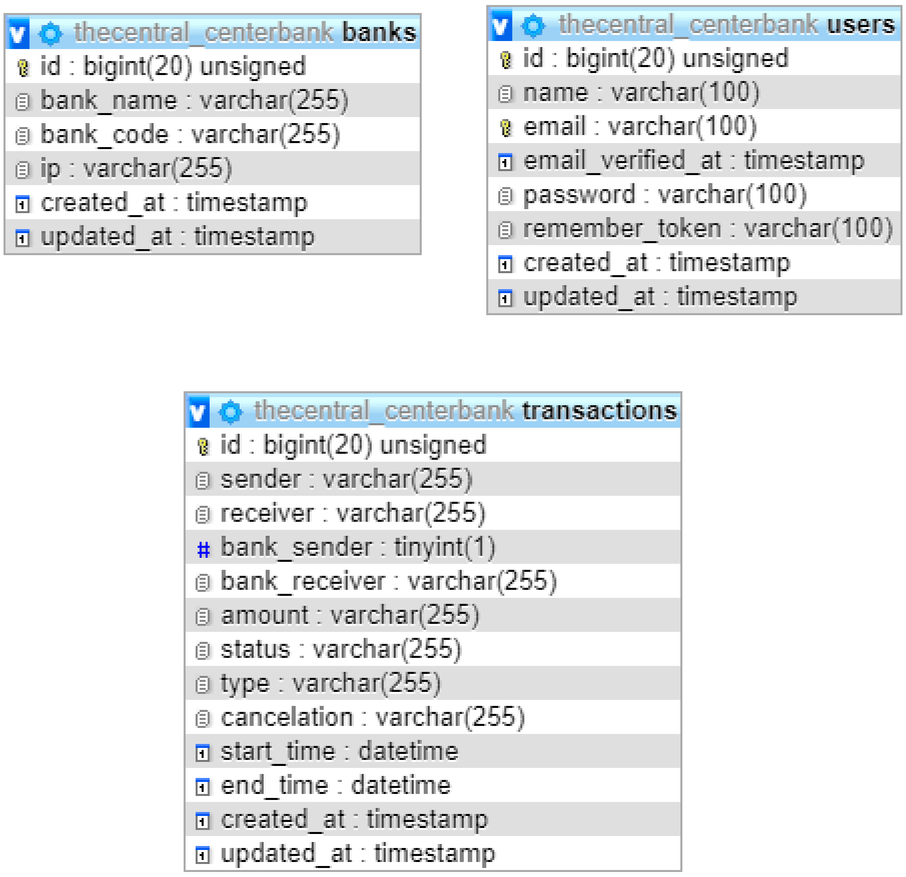


bank1_arabbank **users**
- id : bigint(20) unsigned
- name : varchar(100)
- email : varchar(100)
- email_verified_at : timestamp
- password : varchar(100)
- remember_token : varchar(100)
- created_at : timestamp
- updated_at : timestamp

bank1_arabbank **transactions**
- id : bigint(20) unsigned
- sender : varchar(255)
- receiver : varchar(255)
- is_bank : tinyint(1)
- bank : varchar(255)
- amount : varchar(255)
- status : varchar(255)
- type : varchar(255)
- cancelation : varchar(255)
- start_time : datetime
- end_time : datetime
- created_at : timestamp
- updated_at : timestamp

bank1_arabbank **customers**
- id : bigint(20) unsigned
- name : varchar(100)
- account_number : varchar(100)
- phone_number : varchar(100)
- national_id : varchar(100)
- credit : varchar(100)
- pin_code : varchar(100)
- last_pin_code : varchar(100)
- limit_transfer : varchar(100)
- email : varchar(100)
- created_at : timestamp
- updated_at : timestamp

Figure D.4: Database structure for MOP bank.



thecentral_centerbank **banks**
- id : bigint(20) unsigned
- bank_name : varchar(255)
- bank_code : varchar(255)
- ip : varchar(255)
- created_at : timestamp
- updated_at : timestamp

thecentral_centerbank **users**
- id : bigint(20) unsigned
- name : varchar(100)
- email : varchar(100)
- email_verified_at : timestamp
- password : varchar(100)
- remember_token : varchar(100)
- created_at : timestamp
- updated_at : timestamp

thecentral_centerbank **transactions**
- id : bigint(20) unsigned
- sender : varchar(255)
- receiver : varchar(255)
- bank_sender : tinyint(1)
- bank_receiver : varchar(255)
- amount : varchar(255)
- status : varchar(255)
- type : varchar(255)
- cancelation : varchar(255)
- start_time : datetime
- end_time : datetime
- created_at : timestamp
- updated_at : timestamp

Figure D.5: Database structure for Central bank.

D.6 Guide to using the MOP Simulator

In this section we describe the user guide for each part in simulator, and explain how the process works.

D.6.1 MOP Operator

The process starts in the operator by setting up a new bank server. An administrator logs in to the operator web-application (at www.1stbs.online) using admin email and user/password. The users table in the operator has a record for the admin with encrypted password.

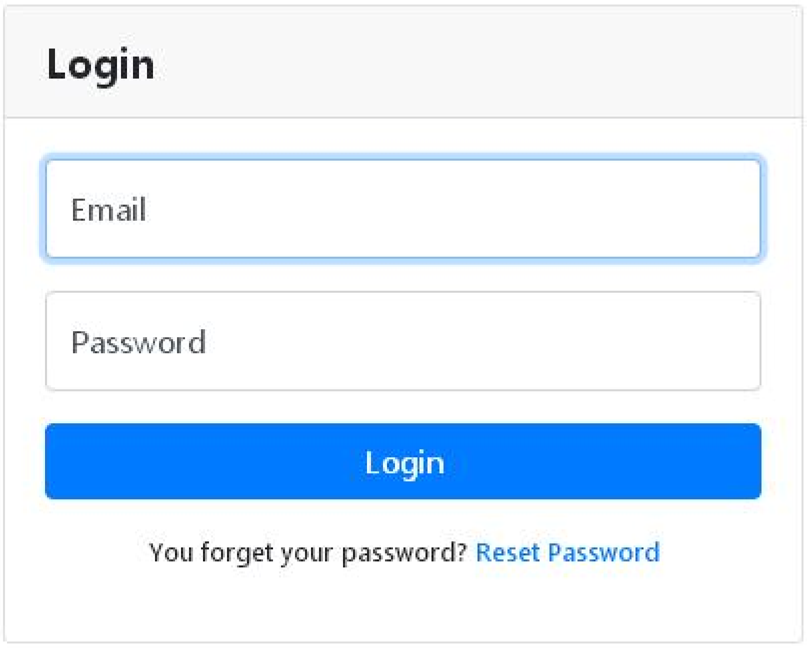


Figure D.6: MOP administrator login

Once the admin logs in to the web-application, the only page that appears in the system is adding banks inside the operator server. When we add a new bank record in the UI, this is reflected in the banks table in the database. As part of the setting up process, bank details such as the IP address are entered into the banks table.

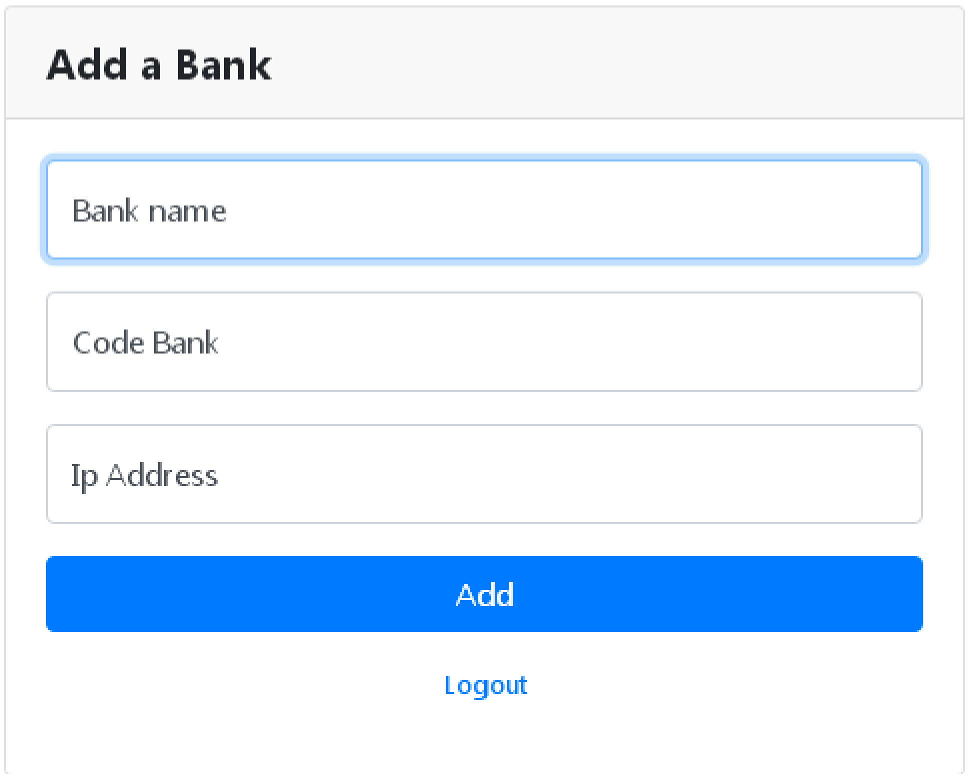
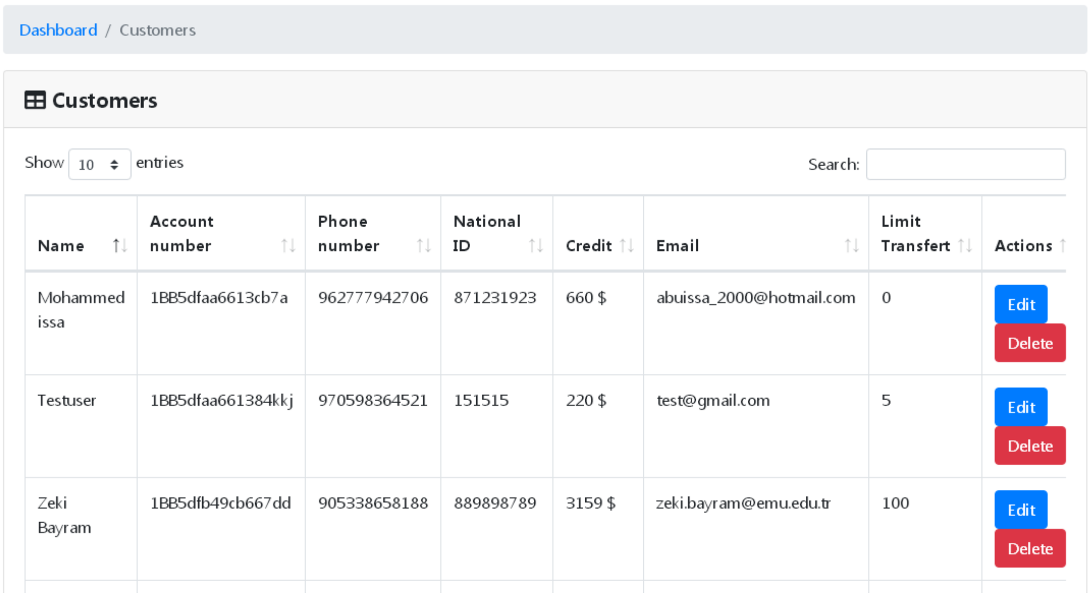


Figure D.7: Add bank interface.

D.6.2 MOP Bank

The bank administrator will login to the bank web application to setup the bank's customers, and synchronize the records with operator customers table. The bank admin can add/edit/delete customers. After synchronizing the customers with the operator database, a welcome message will be sent to the customer with the generated password in the operator database. After that, the process will continue on the bank side by the customer logging in to the web-application (at www.bank-1.online)



Dashboard / Customers

Customers

Show 10 entries Search:

Name	Account number	Phone number	National ID	Credit	Email	Limit Transfert	Actions
Mohammed issa	1BB5dfaa6613cb7a	962777942706	871231923	660 $	abuissa_2000@hotmail.com	0	Edit Delete
Testuser	1BB5dfaa661384kkj	970598364521	151515	220 $	test@gmail.com	5	Edit Delete
Zeki Bayram	1BB5dfb49cb667dd	905338658188	889898789	3159 $	zeki.bayram@emu.edu.tr	100	Edit Delete

Figure D.8: Customers information.

We can get the customer details by calling API (getCustomer(customer-id) method) from bank IP server, or we can call API (checkCustomer(phone) method) to check if the customer exists in the bank database.

The administrator of a bank can add a new customer in the same web-application (Add Customer Menu). This function is done using the API (addCustomer method) on the bank server.

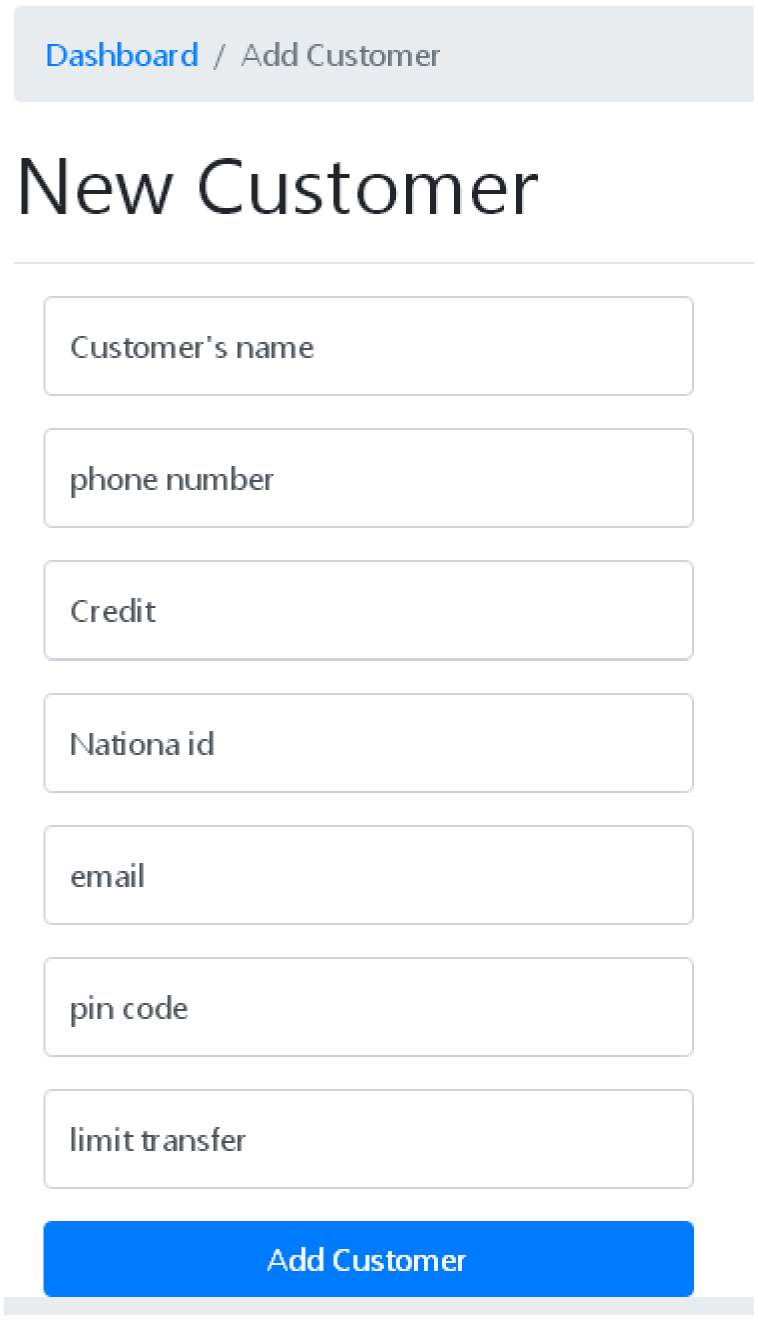
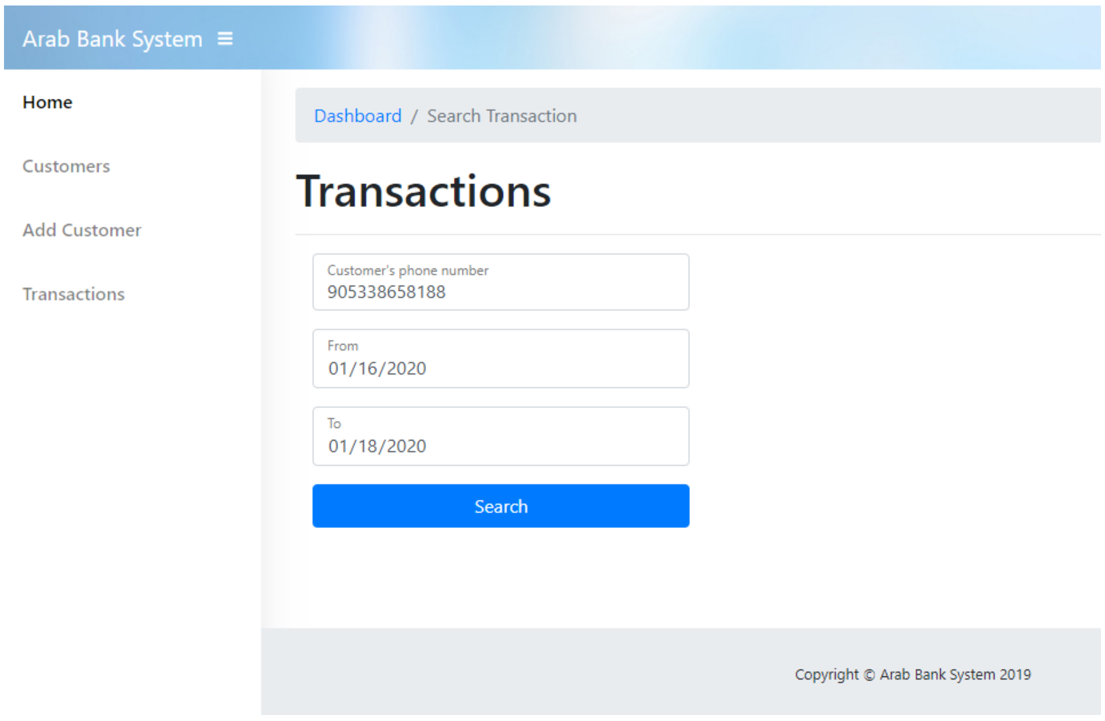


Figure D.9: Adding a customer.

When the administrator adds/edits/deletes any customer, users table inside the operator database will be modified by using API methods (addUser/editUser/deleteUser methods). The welcome SMS message will be sent to the user from the bank that a login was created, and the password will be sent to him to use for login on operator side. After that, the customer can login to operator web-application to manage his transaction between banks using his email and the same password that was sent to his mobile phone.

The administrator in the bank can search and see all transactions from one date to another date using transaction button in the left side.



Customer's phone number
905338658188

From
01/16/2020

To
01/18/2020

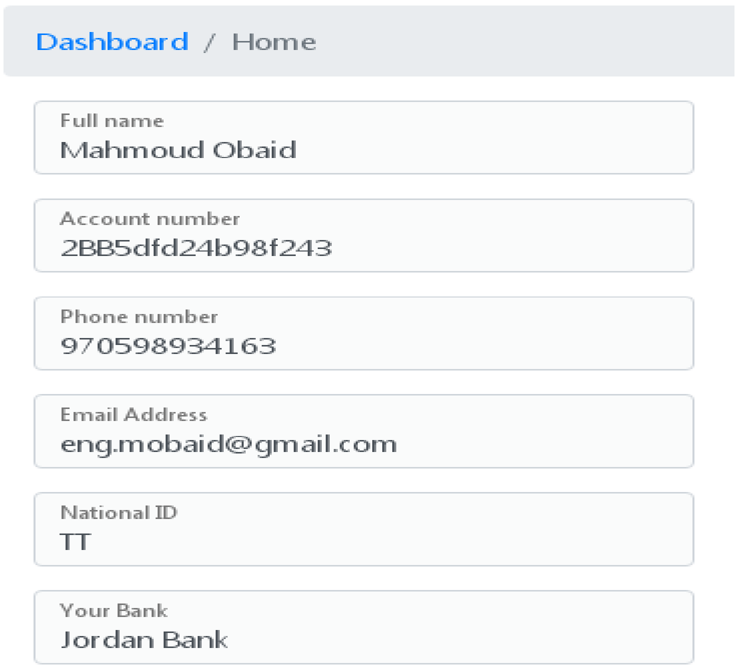
Figure D.10: Transaction details in the bank side.

D.6.3 MOP User

The customer can login to the operator web application to initiate a transaction, view transaction, and change password.

The transaction will start in the operator, which will save a new record in the transaction table with “pending” status and “Not started” in status description, and input amount and phone number to transfer the money. The operator will check that both customers exist in both banks depending on the records that exists at the operator, and update this in status description “Check customers” for the current transaction. The operator will ask the customer’s bank to send a PIN code to the customer, and update this in status description “Ask Pin Code” for the current transaction. The operator will ask the customer to enter the PIN code for the current transaction, the operator will check with the bank that the entered PIN code for current transaction is the same that was sent to client via SMS to transfer the amount, and update this in status description “Verify Pin Code” for the current transaction. The bank will record the transaction with same transaction ID from the operator, send the transaction with same ID to the central bank and return the 0 if success, or negative value for failed, then show error/success message on the operator web application. If the transaction was successful, the transaction log will be updated in the operator as “transaction completed” and recorded in Bank B with same transaction ID that was sent from the operator. Two messages will be sent via SMS (customer A and customer B) to inform them that the money is already transferred. If a connection error happened with banks or when checking amount or when asking PIN code, then the transaction record will be updated to failed with “Technical Error” in status description using rollback API. The total of amount of customer A will be decreased and the total money of customer B will be increased by the transfer amount after a successful transaction. In case of transaction failure, the transaction record will be updated with failure information in the three sites involved. The transaction log will be recorded in the bank and operator with same as transaction ID that was sent from bank, and one message will be sent via SMS to customer A with error message.

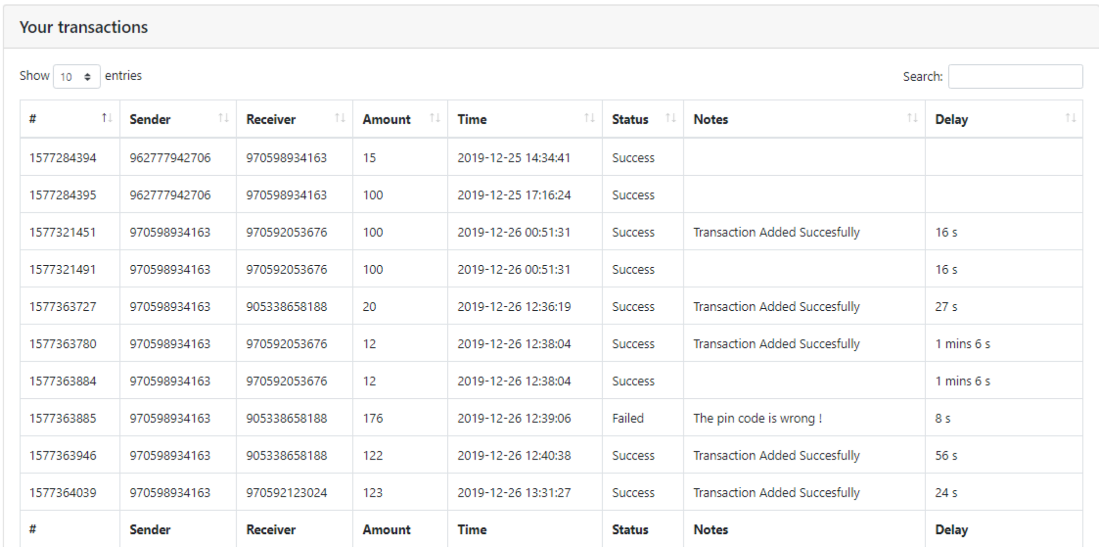
The customer can reset his password from the login screen, and the new password will send by SMS on his mobile. After customer login, the dashboard will appear with the basic information for the current user; this information is obtained by calling API (getCustomer method).



Full name	Mahmoud Obaid
Account number	2BB5dfd24b98f243
Phone number	970598934163
Email Address	eng.mobaid@gmail.com
National ID	TT
Your Bank	Jordan Bank

Figure D.11: User main page.

The customer can view his transaction from the transaction in the left menu, obtained by a call to API (getTransaction method).


Your transactions

Show 10 entries

Search:

#	Sender	Receiver	Amount	Time	Status	Notes	Delay
1577284394	962777942706	970598934163	15	2019-12-25 14:34:41	Success		
1577284395	962777942706	970598934163	100	2019-12-25 17:16:24	Success		
1577321451	970598934163	970592053676	100	2019-12-26 00:51:31	Success	Transaction Added Succesfully	16 s
1577321491	970598934163	970592053676	100	2019-12-26 00:51:31	Success		16 s
1577363727	970598934163	905338658188	20	2019-12-26 12:36:19	Success	Transaction Added Succesfully	27 s
1577363780	970598934163	970592053676	12	2019-12-26 12:38:04	Success	Transaction Added Succesfully	1 mins 6 s
1577363884	970598934163	970592053676	12	2019-12-26 12:38:04	Success		1 mins 6 s
1577363885	970598934163	905338658188	176	2019-12-26 12:39:06	Failed	The pin code is wrong !	8 s
1577363946	970598934163	905338658188	122	2019-12-26 12:40:38	Success	Transaction Added Succesfully	56 s
1577364039	970598934163	970592123024	123	2019-12-26 13:31:27	Success	Transaction Added Succesfully	24 s
#	Sender	Receiver	Amount	Time	Status	Notes	Delay

Figure D.12: Transaction details for user.

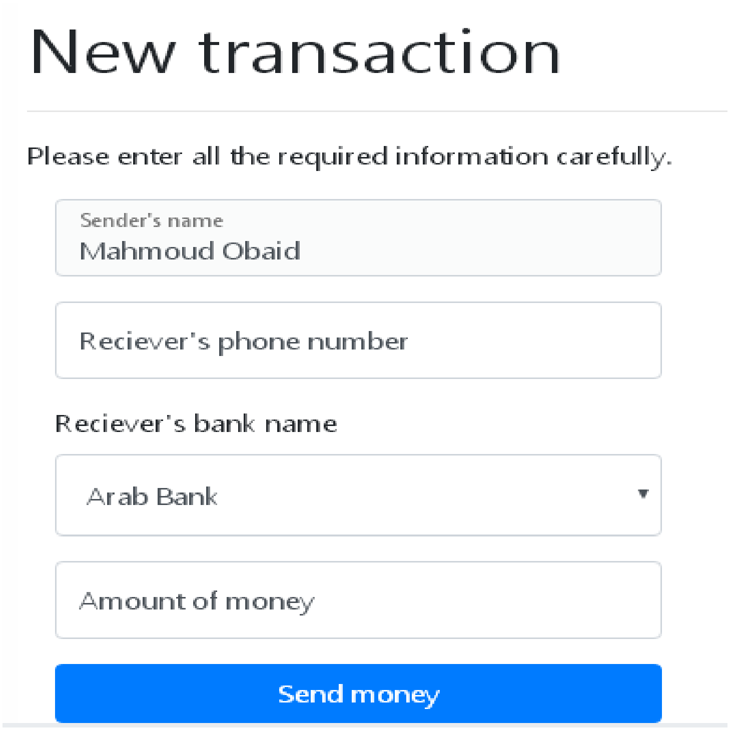
The customer can transfer money to any mobile number on the MOP network, which can be completed by calling below APIs:

i. addTransactionSender Method: add transaction record in sender bank database.

ii. addTransactionReceiver Method: add transaction record in receiver bank database.

iii. add_transaction Method: add transaction record in central bank database.

iv. endTransactionSender Method: calculate amount for sender/receiver and complete transaction successfully, or cancel the transaction by calling API (rollbackTransaction method) to roll back the transaction at all database actors.


New transaction

Please enter all the required information carefully.

Sender's name: Mahmoud Obaid

Reciever's phone number

Reciever's bank name: Arab Bank

Amount of money

Figure D.13: New transaction page.

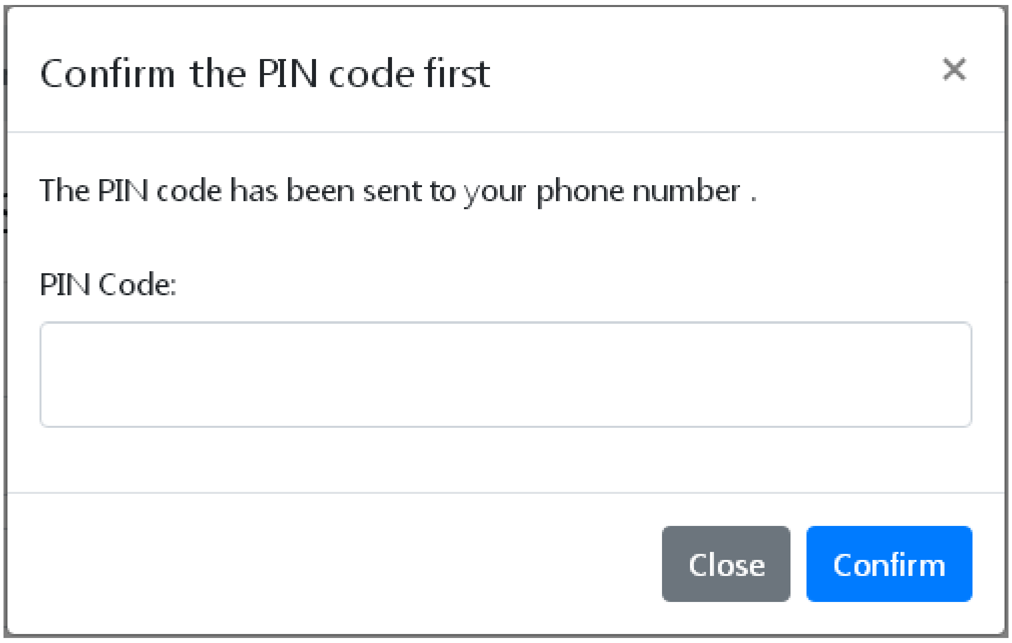
When the customer clicks on send money button in the UI, the operator will connect sender bank to:

i. Check the sender customer balance is above than the amount that the customer wants to send.

ii. Check if the sender limit is greater than the amount that wants to send.

iii. Check if there is a valid record for receiver customer.

If the above is successful, the PIN code will be sent to the customer on his mobile via SMS; the PIN can be fixed, or generated if the bank administrator put 0 in the PIN field for this customer, by calling API (sendCustomerPIN Method).


Confirm the PIN code first

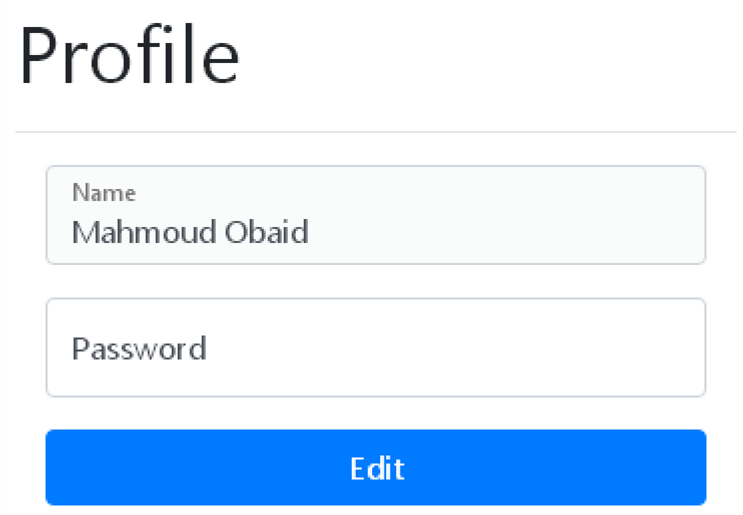
The PIN code has been sent to your phone number .

PIN Code:

Figure D.14: Insert PIN code page.

When the customer receives the SMS and enters the correct PIN code, then the operator will connect with the bank to do the money transfer with the entered PIN code. After that, the system will show a message that transfer is done, or any message for errors (Technical error). If success, the SMS messages will be sent to the sender and receiver on their mobile numbers to let them know that the money already transferred. If failed, the SMS message will be sent to the sender’s mobile that the money was not transferred.

The customer can change his password by clicking on the right corner icon, and select edit password.



Profile

Name

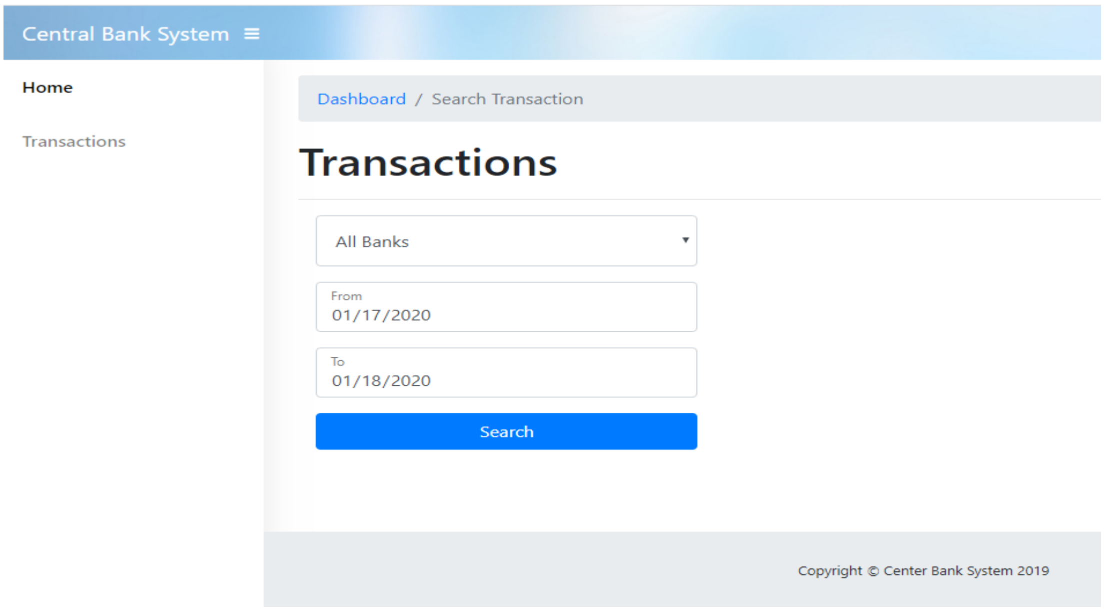
Mahmoud Obaid

Password

Figure D.15: Change password page.

D.6.4 Central Bank

The central bank has the banks information and banks transaction only if transaction is done between a customer in bank-1 and a customer in bank –2, in which case the transaction detail will be insert in the central bank transaction table. The transaction is inserted in the central bank only if it is successfully completed. There is no need to add any transaction if both customers are from the same bank. The administrator can see all transactions from one date to another date using transaction button in the left side.



Transactions

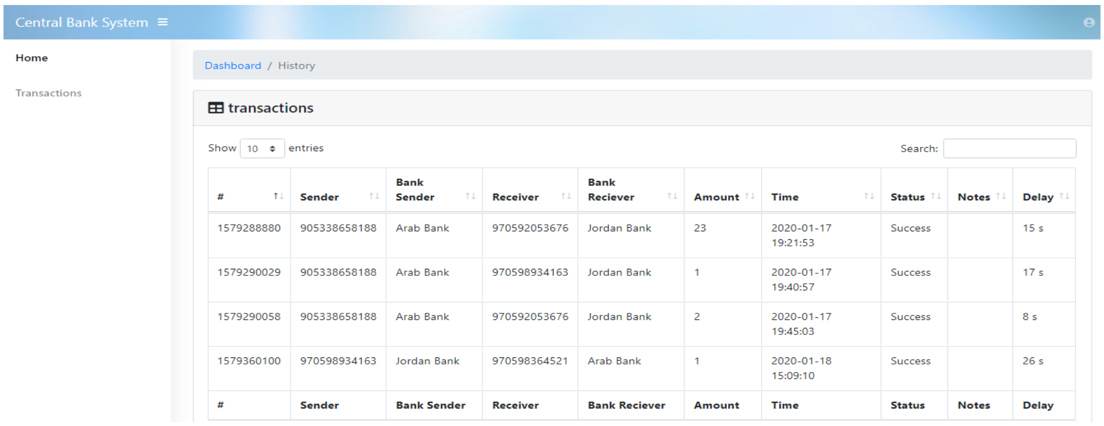
All Banks

From 01/17/2020

To 01/18/2020

Copyright © Center Bank System 2019

Figure D. 16: Search for transaction in central bank.



#	Sender	Bank Sender	Receiver	Bank Reciever	Amount	Time	Status	Notes	Delay
1579288880	905338658188	Arab Bank	970592053676	Jordan Bank	23	2020-01-17 19:21:53	Success		15 s
1579290029	905338658188	Arab Bank	970598934163	Jordan Bank	1	2020-01-17 19:40:57	Success		17 s
1579290058	905338658188	Arab Bank	970592053676	Jordan Bank	2	2020-01-17 19:45:03	Success		8 s
1579360100	970598934163	Jordan Bank	970598364521	Arab Bank	1	2020-01-18 15:09:10	Success		26 s
#	Sender	Bank Sender	Receiver	Bank Reciever	Amount	Time	Status	Notes	Delay

Figure D.17: Transaction details in central bank.

Appendix E: Critical Code for the Simulator

In this appendix we give the critical code for the simulator.

Listing E.1: MOP Operator Helper.

```
namespace App\Helpers;
use App\Bank;
use App\Transaction;
use DateTime;
class Helper
{
    static public function callApi($method,$url,$data)
    {
     try {
        $client = new \GuzzleHttp\Client();
        if ($method == "POST") {
            $response = $client->post($url, ['form_params' => $data]);
            $reason = $response->getReasonPhrase();

            if ($reason == "OK")
                return json_decode($response->getBody()->getContents());


            return false;
        } elseif ($method == "GET") {
            $response = $client->request($method, $url);
            $reason = $response->getReasonPhrase();

            if ($reason == "OK") {
                $response = json_decode($response->getBody()->getContents());
                return $response;
            }
        }
     } catch (\Throwable $th) {
       return false ;
     }




    }

    static public function getUrlApi($bank,$operation)
    {
      $bank = Bank::find($bank);
         if($bank){
            return $bank->ip.$operation;
         }else {
            return false;
         }
    }

    static public function addTransaction($transaction)
    {
        $transactionM = Transaction::find($transaction->id);
        if(!$transactionM)
        {
            $transactionM = new Transaction;
            $transactionM->id = $transaction->id;
        }

        $transactionM->sender = $transaction->sender;
        $transactionM->is_bank = $transaction->is_bank;
        $transactionM->receiver = $transaction->receiver;
        $transactionM->status = $transaction->status;
        $transactionM->type =  $transaction->type;
        $transactionM->amount =  $transaction->amount;
        $transactionM->cancelation = $transaction->cancelation;
        $transactionM->bank =  $transaction->bank;
        $transactionM->start_time =  $transaction->start_time;
        $transactionM->end_time =  $transaction->end_time;
        $transactionM->save();

    }

    static public function getStatus($status) {
        switch($status){
            case 0 : return "Success" ;
            case 1 : return "Failed"  ;
            case 2 : return "pending"  ;
            case 3 : return "Check Customers"  ;
            case 4 : return "Ask Pin Code";
            case 5 : return "Check Pin Code";
        }
    }

    static public function sendSMS($mobileNumber,$text,$senderId = "TestApp" ){
                //API URL
                $message = urlencode($text);

$url="http://josmsservice.com/smsonline/msgservicejo.cfm?numbers=".$mobileNumber.",&senderid=FlexService&AccName=
flexserv&AccPass=flexserv123&msg=".$message."&requesttimeout=5000000";
                $response = self::callApi("GET",$url,null);
                    if($response)
                     return $response;
                    }
   static public   function format_interval($start_date,$end_date) {
                    $first_date = new DateTime($start_date);
                    $second_date = new DateTime($end_date);

            $interval = $first_date->diff($second_date);
                        $result = "";
                        if ($interval->y) { $result .= $interval->format("%y years "); }
                        if ($interval->m) { $result .= $interval->format("%m months "); }
                        if ($interval->d) { $result .= $interval->format("%d days "); }
                        if ($interval->h) { $result .= $interval->format("%h hours "); }
                        if ($interval->i) { $result .= $interval->format("%i mins "); }
                        if ($interval->s) { $result .= $interval->format("%s s "); }

                        return $result;
                    }
}
```

Listing E.2: MOP Operator API.

```
namespace App\Http\Controllers\Api;

use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App\User;
use App\Bank;
use App\Transaction;
use App\Helpers\Helper;

use Illuminate\Support\Facades\Hash;
class ApiController extends Controller
{
    public function EditUser(Request $request){
        try {
            $user = User::find($request->id);
            $userMail = User::where('email',$request->email)->first();

            if($userMail)
              {
                 if($userMail->id != $request->id){
                  return response()->json([
                      'message' => 'Email already exsit',
                      'status' => false
                  ]);
                  }
              }
            if($user)
            {
              $user->phone_number = $request->phone_number;
              $user->national_id = $request->national_id;
              $user->name = $request->name;
              $user->save();
              return response()->json([
                'message' => 'User Edited !!',
                'status' => true
            ]);
            }else {
                return response()->json([
                    'message' => 'User Does\'nt exsit !!',
                    'status' => false
                ]);
            }
        } catch (\Throwable $th) {
            return response()->json([
                'message' => 'Error !',
                'status' => false,
                'object' => $th
            ]);
        }

    }

    public function AddUser(Request $request){
              $userc = User::where('email',$request->email)->first();
              if($userc)
              {
                return response()->json([
                    'message' => 'Email Exist !!',
                    'status' => false
                ]);
              }

              $pass = str_random(8);
              $codebank = $this->CodeBank($request->bank);
              $hashed_random_password = Hash::make($pass);


            $user =  User::create([
                'name' => $request->name,
                'email' => $request->email,
                'account_number' => $codebank,
                'password' => $hashed_random_password,
                'bank' => $request->bank,
                'phone_number' => $request->phone_number,
                'national_id' => $request->national_id,
            ]);

            $text = "Hello ".$request->name." Your Account on OperatorBank Created Successufully : your password is: ".
            $pass;
            if($request->pin_code != 0)
            $text .= "|| Your pin code is :".$request->pin_code;
            Helper::sendSMS($request->phone_number,$text);
            return response()->json([
                'message' => 'User Added !!',
                'status' => true,
                'user' => $user,
            ]);


    }

    private function CodeBank($bank) {
        return $bank."BB".uniqid();
        }


    public function deleteUser($id)
        {
          $user = User::find($id);
          if($user)
           {
             Transaction::where('sender', $id)
                   ->update(['sender' => $user->phone_number]);
             Transaction::where('receiver', $id)
                         ->update(['sender' => $user->phone_number]);
                         $user->delete();
              return response()->json([
                            'message' => 'User Deleted !!',
                            'status' => true,
                        ]);
           }else {
            return response()->json([
                'message' => 'User doesn\'t exist !!',
                'status' => false,
            ]);
           }
        }

}
```

Listing E.3: MOP operator HomeController.

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Customer;
use App\Helpers\Helper;
use Validator;
use MercurySeries\Flashy\Flashy;
use App\Bank;
use App\Transaction;
use App\User;
use Illuminate\Support\Facades\URL;
use Illuminate\Support\Facades\Hash;
use DateTime;
use Illuminate\Support\Facades\Session;
class HomeController extends Controller
{
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
      // $this->middleware('auth');
    }

    /**
     * Show the application dashboard.
     *
     * @return \Illuminate\Contracts\Support\Renderable
     */
    public function index()
    {
        $user = auth()->user();
        $data = null;

        $url = Helper::getUrlApi($user->bank,config("api.bank.getCustomer").$user->id);

        $data =   Helper::callApi("GET",$url,null);
         $status = $data->status;
          if($data->status) {

            $data =$data->customer;
            return view('home',compact("data","user","status"));
           }
          $data =$data->message;
        return view('home',compact("data","user",'status'));
    }

    public function gettransaction()
    {
        $user = auth()->user();
        $transactions = null;

        $url = Helper::getUrlApi($user->bank,config("api.bank.gettransactions").$user->id);

        $transactions =   Helper::callApi("GET",$url,null);

       $transactions = $transactions->transactions;
    //  dd($transactions);
        return view('getTransactions',compact("transactions","user"));
    }

    public function addTransaction()
    {
        $user = auth()->user();
        if(!Session::has('transaction'))
        {

            $transactionM = new Transaction;
            $transactionM->id = time();
            $transactionM->sender = $user->id;
            $transactionM->status = 2;
        $transactionM->bank = $user->bank;
        $transactionM->cancelation = "Not Started";
        $transactionM->type = "OUT";
        $transactionM->receiver = "-";
        $transactionM->amount = "-";
        $transactionM->is_bank = false;
        $transactionM->start_time = date("Y-m-d H:i:s");
        $transactionM->end_time =  date("Y-m-d H:i:s");
        $transactionM->save();
        session()->put('transaction', $transactionM->id);
      }

    $banks = Bank::get();

    return view('addTransaction',compact("user","banks"));
}

public function saveTransaction(Request $request)
{

    Validator::make($request->all(), [
        'receiver' => 'required',
        'amount' => 'required',
        'pin_code' => 'required',
    ])->validate();
        /// Check Pin Code ///
    $transaction = Transaction::find(Session::get('transaction'));
    $transaction->status = 5;
    $transaction->cancelation = "Check Pin Code";
    $transaction->save();
        /// Check Pin Code ///
    $response = null;
    $data = $request->all();
    $user = auth()->user();
    $data['sender'] = $user->id;
    $data['type'] = 'OUT';
    $data['start_time'] = date("Y-m-d H:i:s");
     $is_bank = 0 ;
    if($user->bank == $request->bank){
        $is_bank = 1;
    }

    $data['senderphone'] =  $user->phone_number;
    $data['is_bank'] = $is_bank;
    $data['idtrans'] = Session::get('transaction');
    $url = Helper::getUrlApi($user->bank,config("api.bank.addTransaction"));

        if($is_bank){
            $response = Helper::callApi("POST",$url,$data);
           if($response)
            {
                Helper::addTransaction($response->transactionOut);
                Helper::addTransaction($response->transactionIn);
            }else {
                Flashy::primary("There's a technical problem, please try again");
                  return redirect(URL::previous());
            }
          $message = $response->message;
        } else {
            //dd($data);
            /*$url = Helper::getUrlApi($req->bank,config("api.bank.checkCustomer"). $data['receiver']);
            $checkreceiver = Helper::callApi("GET",$url,null);
                if(!$checkreceiver->status)
                Flashy::primary("The Receiver doesn't Exist !");
                return redirect(URL::previous());*/
           // var_dump($data);

            $responseIn = Helper::callApi("POST",$url,$data);
            if(!$responseIn)
            {
                Flashy::primary("There's a technical problem, please try again");
                  return redirect(URL::previous());
            }
            $urlIn = Helper::getUrlApi($request->bank,config("api.bank.addTransactionin"));
            $data['sender'] = $user->phone_number;
            $data['type'] = 'IN';
            if($responseIn->status)
            {
               $responseOut = Helper::callApi("POST",$urlIn,$data);
                     if(!$responseOut)
                  {
                     $dataroll['id'] = $responseIn->transactionOut->id;
                     $UrlRoll = Helper::getUrlApi($user->bank,config("api.bank.rollbacktransaction"));
                     $rollbackCall = Helper::callApi("POST",$UrlRoll,$dataroll);

                     Flashy::primary($rollbackCall->message);
                     return redirect(URL::previous());
                  }else {
                     if($responseOut->status)
                     {
                        $Urlend = Helper::getUrlApi($user->bank,config("api.bank.endtransactionsender"));
                        $end = Helper::callApi("POST",$Urlend,$data);
                        if(!$end)
                         dd("there's a problem !!");
                     }else {
                        $dataroll['id'] = $responseIn->transactionOut->id;
                        $UrlRoll = Helper::getUrlApi($user->bank,config("api.bank.rollbacktransaction"));
                        $rollbackCall = Helper::callApi("POST",$UrlRoll,$dataroll);

                        Flashy::primary($rollbackCall->message);
                        return redirect(URL::previous());
                     }
                  }
               Helper::addTransaction($responseOut->transaction);
            }
            Helper::addTransaction($responseIn->transactionOut);
            $message = $responseIn->message;
          }


      session()->forget('transaction');

      Flashy::primary($message);
      return redirect(URL::previous());
   }

  public function profile(){
   $user = auth()->user();

   return view('profile',compact("user"));
  }

  public function editprofile(Request $request){
   $user = auth()->user();


   $obj_user = User::find($user->id);
   if($request['password'] != null)
   {
   $obj_user->password = Hash::make($request['password']);
   $obj_user->save();
   }
   Flashy::primary("Update information with success");
   return redirect(URL::previous());
  }


}
```

Listing E.4: MOP Operator API Calls.

```
return [

    /*
    |--------------------------------------------------------------------------
    | Describe Api Calls
    |--------------------------------------------------------------------------
    |
    |
    |
    */

    'bank' => [
        'addCustomer' => "/api/savecustomerdetails",
        'getCustomer' => "/api/customer/",
        'gettransactions' => "/api/gettransaction/",
        'addTransaction' => "/api/addtransactionsender",
        'addTransactionin' => "/api/addtransactionreceiver",
        'checkCustomer' => "/api/checkcustomer/",
        'sendCustomerPin' => "/api/sendcustomerpin/",
        'endtransactionsender' => "/api/endtransactionsender/",
        'rollbacktransaction' => "/api/rollbacktransaction/",

    ],



];;
```

Listing E.5: MOP Operator Web route using API.

```
/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Auth::routes();

Route::post('/resetpassword', 'Auth\LoginController@resetpassword')->name('resetpassword');

Route::group(['middleware' => ['check.user','web']], function () {
    Route::get('/', 'HomeController@index')->name('home');
    Route::get('/addtransaction', 'HomeController@addTransaction')->name('transaction');
    Route::post('/addtransaction', 'HomeController@saveTransaction')->name('addtransaction');
    Route::get('/transactions', 'HomeController@gettransaction')->name('gettransaction');
    Route::post('/checkbeforesendpin', 'AjaxController@CheckBeforeSendPin')->name('checkbeforesendpin');
    Route::get('/profile', 'HomeController@profile')->name('profile');
    Route::post('/profile', 'HomeController@editprofile')->name('editprofile');

});

Route::group(['middleware' => ['check.admin','web']], function () {
    Route::get('/addbank', 'AdminController@bank')->name('bank');
    Route::post('/addbank', 'AdminController@addBank')->name('addbank');
});
```

Listing E.6: MOP Bank Helper.

```
namespace App\Helpers;
use App\Customer;

class Helper
{
  static public function callApi($method,$url,$data)
   {
      $client = new \GuzzleHttp\Client();
      if ($method == "POST") {
         $response = $client->post($url, ['form_params' => $data]);
         $reason = $response->getReasonPhrase();
         if ($reason == "OK")
             return json_decode($response->getBody()->getContents());


         return false;
      } elseif ($method == "GET") {
         $response = $client->request($method, $url);
         $reason = $response->getReasonPhrase();

         if ($reason == "OK") {

             $response = json_decode($response->getBody()->getContents());
             return $response;
         }
      }

      return true;
   }

    static public function sendSMS($mobileNumber,$text){
             $message = urlencode($text);
             //API URL

$url="http://josmsservice.com/smsonline/msgservicejo.cfm?numbers=".$mobileNumber.",&senderid=FlexService&AccName=
flexserv&AccPass=flexserv123&msg=".$message."&requesttimeout=5000000";
             $response = self::callApi("GET",$url,null);
                 if($response)
                  return $response;

             // init the resource
            // dd($response);
             //Print error if any
                 }
   static public function SmsText($req,$status,$message,$whome = "sender"){

             if($whome == "sender")
             {
                 if($status ==0)
                     return "the transaction Completed with Success, You transfer ".$req->amount. " To ".
                             $req->receiver;
                 else {
                     return "The transaction failed Cause: ".$message;
                        }
                 }else {
                  $sender = Customer::find($req->sender);
                   return "You recieved  ".$req->amount. " USD From : ".$req->phone_number;
             }
         }
   static public function CheckPin($pin_code,$sender){
              if($sender->pin_code != 0)
              {
                 if($sender->pin_code == $pin_code)
                     return true;
                 else
                     return false;
              }else{
                 if($sender->last_pin_code == $pin_code)
                     return true;
                 else
                     return false;
              }
         }
}
```

Listing E.7: MOP Bank API.

```
namespace App\Http\Controllers\Api;

use App\Customer;
use App\Transaction;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App\Helpers\Helper;
use DateTime;
class BankController extends Controller
{

    public function add_customer(Request $request){

        $customer = new Customer;

        $customer->id =  $request->id;
        $customer->name =  $request->name;
        $customer->account_number = $request->account_number;
        $customer->phone_number = $request->phone_number;
        $customer->national_id = $request->national_id;
        $customer->credit =$request->credit;
        $customer->email = $request->email;
        $customer->pin_code = $request->pin_code;
        $customer->limit_transfer = $request->limit_transfer;
        $customer->save();

        return response()->json([
            'message' => 'Customer Added !!',
            'customer' => $customer,
            'status' => true
        ]);

    }

    public function get_customer($id){



        $customerEntity =  Customer::find($id);
            if($customerEntity){
                return response()->json([
                    'customer' => $customerEntity,
                    'status' => true
                ]);
            }else {
                return response()->json([
                    'message' => 'Customer Doesn\'t find !!',
                    'status' => false
                ]);
            }


    }

     public function check_customer($phone){

        $receiver = Customer::where('phone_number',$phone)->first();
            if($receiver){
                return response()->json([
                    'customer' => $receiver,
                    'status' => true
                ]);
            }else {
                return response()->json([
                    'message' => 'Customer Doesn\'t find !!',
                      'status' => false
                ]);
            }


    }

    public function get_transactions($id){
```

```
    $transactionEntity =  Transaction::with('senderC')->with('receiverC')->where('receiver',$id)->orWhere('sender',$id)->get();
        if($transactionEntity->first()){
            return response()->json([
                'status'  => true,
                'transactions' => $transactionEntity
            ]);
        }else {
            return response()->json([
                'status'  => true,
                'transactions' => [],
            ]);
        }


}

public function add_transactionSender(Request $request) {

    $customerM = new Customer;
    $transactionM = new Transaction;
    $message = "Transaction Added  Succesfully";
    $status = " ";
    $cstmr =  $customerM->find($request->sender);
    $receiver =  $request->receiver ;
    $isOnprocess = true;
    $transactionIn = null;
    $state = false;
    if(Helper::CheckPin($request->pin_code,$cstmr))
    {
        if($request->is_bank)
        {
            $receiver = $customerM->where('phone_number',$request->receiver)->first();
                if ($receiver) {
                    $receiver = $receiver->id;
                   // $this->addtransactionreceiver($request);
                } else {
                    $message = "The Receiver Account doesn't exist ";
                    $status  = 1;
                    $receiver = $request->receiver;
                    $isOnprocess = false ;
                }
        }
        if($isOnprocess){

            if( $cstmr->limit_transfer != 0 && $request->amount  >=  $cstmr->limit_transfer)
            {
                $message = "You across amount of limit transfer";
                $status  = 1;
            }else {
        if ($cstmr->credit >= $request->amount)
        {
            $cstmr->update(["credit" => $cstmr->credit - $request->amount ]);
            $status  = 0;
              $state = true;
            if($request->is_bank)
                    {
                        $transactionIn = $this->addtransactionreceiver($request,$cstmr,$receiver,true);
                    }
        }else {
            $message = "You don't have enough money to make this transaction";
            $status  = 1;
        }
    }
                    }
    }else {
        $message = "The pin code is wrong !";
        $status  = 1;
    }
    $transactionM->id = $request->idtrans;
    $transactionM->sender = $cstmr->id;
    $transactionM->is_bank = $request->is_bank;
    $transactionM->receiver = $receiver;
    $transactionM->status = $status;
    $transactionM->type = $request->type;
    $transactionM->amount = $request->amount;
    $transactionM->start_time = $request->start_time;
//    $transactionM->end_time =new DateTime();
    $transactionM->end_time = date("Y-m-d H:i:s");
    $transactionM->cancelation =$message;
    $transactionM->bank =  $request->bank;
    $transactionM->save();
    $text = Helper::SmsText($request,$status,$message);
    Helper::sendSMS($cstmr->phone_number,$text);
        if($transactionM->status == 0)
        {
          $text = Helper::SmsText($request,null,null,"receiver");
           Helper::sendSMS($request->receiver,$text);
        }


    return response()->json([
        'message' => $message,
        'transactionOut' =>$transactionM,
        'transactionIn' => $transactionIn,
        'status' => $state
    ]);
 }

 public function addtransactionreceiver($request,$cstmr,$receiver,$isIn)
 {
    $customerM = new Customer;
    $transactionM = new Transaction;
    if($isIn == true ){
        $receiverE = $customerM->find($receiver);
        $receiverE->update(["credit" => $receiverE->credit + $request->amount ]);
    $transactionM->id = time();
    $transactionM->sender = $cstmr->id;
    $transactionM->is_bank = $request->is_bank;
    $transactionM->receiver = $receiver;
    $transactionM->status = 0;
    $transactionM->type = 'IN';
    $transactionM->amount = $request->amount;
    $transactionM->start_time = $request->start_time;
    $transactionM->end_time = date("Y-m-d H:i:s");
    $transactionM->cancelation =' ';
    $transactionM->bank =  $request->bank;
    $transactionM->save();
    return $transactionM;
    }else {
     $receiverE = $customerM->where('phone_number',$request->receiver)->first();
        $receiverE->update(["credit" => $receiverE->credit + $request->amount ]);

    $transactionM->sender = $request->sender;
    $transactionM->is_bank = $request->is_bank;
    $transactionM->receiver = $receiverE->id;
    $transactionM->status =0;
    $transactionM->type = 'IN';
    $transactionM->start_time = date("Y-m-d H:i:s");
    $transactionM->end_time = date("Y-m-d H:i:s");
    $transactionM->amount = $request->amount;
    $transactionM->cancelation =' ';
    $transactionM->bank =  $request->bank;
    $transactionM->save();

    return $transactionM;
    }
 }

  public function add_transactionReceiver(Request $request){


   $transactionM =  $this->addtransactionreceiver($request,null,null,false);
        if($transactionM){
             $text = Helper::SmsText($request,null,null,"receiver");
            return response()->json([
                'status'  => true,
                'transaction' => $transactionM
            ]);
        }else {
            return response()->json([
                'status'  => false,
                'message' => "There's a problem!!!",
            ]);
        }


}
public function endtransactionsender($request){
  $customerM = new Customer;
  $cstmr =  $customerM->find($request->sender);
  if($cstmr)
  {
    $cstmr->update(["credit" => $cstmr->credit - $request->amount ]);
    return response()->json([
        'message' => "transaction end with success",
        'status' => true
    ]);
  }else {
    return response()->json([
        'message' => "customer doesn't exsit ",
        'status' => false
    ]);
  }

}

public function rollbacktransaction($request){
    $transactionM = new Transaction;
  $transaction =  $transaction->find($request->id);
  if($transaction)
  {
    $transaction->update(["status" => 1,'cancelation' => "Transaction failed, Technical Error!"]);
    return response()->json([
        'message' => "Transaction failed, Technical Error!",
        'status' => true
    ]);
  }else {
    return response()->json([
        'message' => "transaction doesn't exsit ",
        'status' => false
    ]);
  }
}
public function sendCustomerPin($phone){

     $sender = Customer::where('phone_number',$phone)->first();
        if($sender){
              if($sender->pin_code != 0)
              {
              $text = "The pin code for the transaction is : ". $sender->pin_code;
                 Helper::sendSMS($phone,$text);
              }else {
                $length = 4;
                $randomletter = substr(str_shuffle("912837465"), 0, $length);
                $sender->last_pin_code = $randomletter ;
                  $sender->save();
                    $text = "The pin code for the transaction is : ". $sender->last_pin_code;
                 Helper::sendSMS($phone,$text);
              }
                     return response()->json([
                            'status' => true,
                            'pin_code' => $sender->last_pin_code
                        ]);
        }else {
            return response()->json([
                'message' => 'Customer Doesn\'t find !!',
                  'status' => false
            ]);
        }
          return response()->json([
                'message' => 'Error !!',
                  'status' => false
            ]);
}}
```

Listing E.8: MOP Bank Customer Controller.

```
namespace App\Http\Controllers;
use App\Customer;
use App\Transaction;
use Illuminate\Http\Request;
use Validator;
use App\Helpers\Helper;
use MercurySeries\Flashy\Flashy;
use Illuminate\Support\Facades\URL;
class CustomerController extends Controller
{

     /**
      * Create a new controller instance.
      *
      * @return void
      */
     public function __construct()
     {
         $this->middleware('auth');
     }

   /**
    * Display a listing of the resource.
    *
    * @return \Illuminate\Http\Response
    */
   public function index()
   {
        $customers = Customer::All();
        return view('customers',compact("customers"));
   }

   /**
    * Show the form for creating a new resource.
    *
    * @return \Illuminate\Http\Response
    */
   public function create()
   {
        return view('add');
   }

   /**
    * Store a newly created resource in storage.
    *
    * @param  \Illuminate\Http\Request  $request
    * @return \Illuminate\Http\Response
    */
   public function store(Request $request)
   {
     Validator::make($request->all(), [
     'name' => ['required', 'string', 'max:255'],
     'pin_code' => ['required', 'string', 'max:6'],
     'phone_number' => ['required', 'string', 'max:255'],
     'national_id' => ['required', 'string', 'max:255'],
     'credit' => ['required', 'Numeric'],
     'limit_transfer' => ['required', 'Numeric'],
     'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
        ])->validate();
        $data =$request->all();
        $data['bank'] = 1 ;
        $customer = new Customer;
        $url = env("URL_OPERATOR")."adduser";
        $response = Helper::callApi("POST",$url,$data);
        if($response){
            if($response->status)
            {
              $customer->id =  $response->user->id;
              $customer->name =  $request->name;
              $customer->account_number = $response->user->account_number;
              $customer->phone_number = $request->phone_number;
              $customer->national_id = $request->national_id;
              $customer->credit =$request->credit;
              $customer->email = $request->email;
            $customer->pin_code = $request->pin_code;
            $customer->limit_transfer = $request->limit_transfer;
            $customer->save();
            Flashy::primary("Customer Added with success ");
                return redirect(URL::previous());
          }else {
            Flashy::primary($response->message);
                return redirect(URL::previous());
          }
      }else {
        Flashy::primary("There's a technical problem, please try again");
                return redirect(URL::previous());
      }


  }

  /**
   * Display the specified resource.
   *
   * @param  int  $id
   * @return \Illuminate\Http\Response
   */
  public function show($id)
  {
    $customer = Customer::find($id);
    if($customer)
    return view("show",compact("customer"));

    abort(404);
  }

  /**
   * Show the form for editing the specified resource.
   *
   * @param  int  $id
   * @return \Illuminate\Http\Response
   */
   public function edit($id)
   {
      $customer = Customer::find($id);
      if($customer)
      return view("edit",compact("customer"));

      abort(404);
   }

   /**
    * Update the specified resource in storage.
    *
    * @param  \Illuminate\Http\Request  $request
    * @param  int  $id
    * @return \Illuminate\Http\Response
    */
   public function update(Request $request, $id)
   {
     $customer =  Customer::find($id);
     Validator::make($request->all(), [
     'name' => ['required', 'string', 'max:255'],
     'pin_code' => ['required', 'string', 'max:6'],
     'phone_number' => ['required', 'string', 'max:255'],
     'national_id' => ['required', 'string', 'max:255'],
     'credit' => ['required', 'Numeric'],
     'limit_transfer' => ['required', 'Numeric'],
     'email' => ['required', 'string', 'email', 'max:255', 'unique:users,email,'.$customer->email],
        ])->validate();
        $data =$request->all();
        $data['id'] =$id;

        $url = env("URL_OPERATOR")."edituser";
        $response = Helper::callApi("POST",$url,$data);
        if($response){
           if($response->status)
           {
             $customer->name =  $request->name;
             $customer->phone_number = $request->phone_number;
             $customer->national_id = $request->national_id;
             $customer->credit =$request->credit;
             $customer->email = $request->email;
             $customer->pin_code = $request->pin_code;
             $customer->limit_transfer = $request->limit_transfer;
             $customer->save();
             Flashy::primary("Customer Edited with success ");
                 return redirect(URL::previous());
           }else {
             Flashy::primary($response->message);
                 return redirect(URL::previous());
           }
        }else {
           Flashy::primary("There's a technical problem, please try again");
                 return redirect(URL::previous());
        }
   }

   /**
    * Remove the specified resource from storage.
    *
    * @param  int  $id
    * @return \Illuminate\Http\Response
    */
   public function destroy($id)
   {
     $customer = Customer::find($id);
     if($customer)
      {
        Transaction::where('sender', $id)
             ->update(['sender' => $customer->phone_number]);
        Transaction::where('receiver', $id)
                 ->update(['sender' => $customer->phone_number]);
        $url = env("URL_OPERATOR")."deleteuser/".$id;
        $response = Helper::callApi("GET",$url,null);
           if($response) {
              if($response->status)
              {
                $customer->delete();
                Flashy::primary("Customer Edited with success ");
                    return redirect(URL::previous());
              }else {
                Flashy::primary($response->message);
                    return redirect(URL::previous());
              }
           }else {
             Flashy::primary("There's a technical problem, please try again");
                    return redirect(URL::previous());
           }
      }

           abort(404);
   }
}
```

Listing E.9: Describe MOP Bank API Calls

```
use Illuminate\Http\Request;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| is assigned the "api" middleware group. Enjoy building your API!
|
*/

    Route::get('/customer/{id}', 'Api\BankController@get_customer')->name('getcostumer');
    Route::get('/checkcustomer/{phone}', 'Api\BankController@check_customer')->name('checkcustomer');
    Route::get('/sendcustomerpin/{phone}', 'Api\BankController@sendCustomerPin')->name('sendcustomerpin');
    Route::post('/savecustomerdetails', 'Api\BankController@add_customer')->name('addcostumer');
    Route::post('/endtransactionsender', 'Api\BankController@end_transactionSender')->name('end_transactionSender');
    Route::post('/rollbacktransaction', 'Api\BankController@rollbacktransaction')->name('rollbacktransaction');
    Route::post('/addtransactionsender', 'Api\BankController@add_transactionSender')->name('addtransactionsender');
    Route::get('/gettransaction/{id}', 'Api\BankController@get_transactions')->name('gettransaction');
    Route::post('/addtransactionreceiver', 'Api\BankController@add_transactionReceiver')->name('addtransactionreceiver');
```

Appendix F: Raw Timing Data for Transactions in the Prototype Implementation.

In this appendix we give the raw data relating to 20 transactions carried out in the prototype implementation of MOP. Listing F.1 gives the table definition and table contents as provided by the database server. Table F.2 contains the relevant part of the data provided by the database server in tabular form.

Listing F.1: Database table for transaction details in the prototype implementation.

```
-- Table structure for table `cdr`
--

DROP TABLE IF EXISTS `cdr`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `cdr` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `duration` varchar(10) DEFAULT NULL,
  `dst_mopid` varchar(100) DEFAULT NULL,
  `src_mopid` varchar(100) DEFAULT NULL,
  `amount` varchar(10) DEFAULT NULL,
  `start` datetime DEFAULT NULL,
  `answer` datetime DEFAULT NULL,
  `end` datetime DEFAULT NULL,
  `src` varchar(50) DEFAULT NULL,
  `clid` int(11) DEFAULT NULL,
  `channel` varchar(50) DEFAULT NULL,
  `billsec` int(10) DEFAULT NULL,
  `disposition` varchar(255) DEFAULT NULL,
  `status` enum('0','1','2','3','4','5','6','7') NOT NULL DEFAULT '0' COMMENT '1 = Success Payment | 2 = waiting call back | 0
= Error | 3 = Calling Now | 4 = Request | 5 = Error from CB | 6 = Error from receiver bank | 7 = Froud',
  `trx_method` int(11) DEFAULT NULL,
  `extra_data` varchar(255) DEFAULT NULL,
  `country_id` int(11) NOT NULL,
  `currency` int(3) DEFAULT NULL,
  `account` int(11) DEFAULT NULL,
  `receiver_bank_ip` varchar(255) DEFAULT NULL,
  `sender_bank_ip` varchar(255) DEFAULT NULL,
  `account_receiver` varchar(255) DEFAULT NULL,
  `account_sender` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`,`country_id`)
) ENGINE=MyISAM AUTO_INCREMENT=191 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;
-
-- Dumping data for table `cdr`
--

LOCK TABLES `cdr` WRITE;
/*!40000 ALTER TABLE `cdr` DISABLE KEYS */;
INSERT INTO `cdr` VALUES (320,'71','4008','4006','12','2018-12-10 18:47:54','2018-12-10 18:48:16','2018-12-10
18:49:05','0786945106',0,'DAHDI/63-
1',49,'ANSWERED','1',1,",1,400,1,'192.168.15.102',NULL,'889977','1010905'),(321,'44','4006','4004','30','2018-12-10
18:55:01','2018-12-10 18:55:20','2018-12-10 18:55:46','0799113799',0,'DAHDI/63-
1',25,'ANSWERED','1',1,",1,1,1,'192.168.15.102',NULL,'1010905','1010905'),(322,'67','4006','4004','10','2018-12-10
18:56:48','2018-12-10 18:57:07','2018-12-10 18:57:56','0799113799',0,'DAHDI/63-
1',49,'ANSWERED','1',1,",1,1,1,'192.168.15.102',NULL,'1010905','1010905'),(323,'87','4006','4004','40','2018-12-10
20:05:24','2018-12-10 20:05:56','2018-12-10 20:06:51','0799113799',0,'DAHDI/63-
1',54,'ANSWERED','1',1,",1,1,1,'192.168.15.102',NULL,'1010905','1010905'),(324,'70','4006','4009','20','2018-12-10
21:00:35','2018-12-10 21:00:53','2018-12-10 21:01:46','0799010693',0,'DAHDI/63-
1',53,'ANSWERED','1',1,",1,400,1,'192.168.15.102',NULL,'1010905','5555444'),(325,'0','4009','4004','25','2018-12-11
08:56:00','2018-12-11 08:56:00','2018-12-11
08:56:00','0799113799',0,'0',0,'','2',1,'',1,400,1,'192.168.15.102',NULL,'5555444','1010905'),(326,'70','4006','4004','12','2018-
12-11 09:05:52','2018-12-11 09:06:10','2018-12-11 09:07:02','0799113799',0,'DAHDI/63-
1',52,'ANSWERED','1',1,'',1,400,1,'192.168.15.102',NULL,'1010905','1010905'),(327,'46','4006','4004','14','2018-12-11
09:30:11','2018-12-11 09:30:30','2018-12-11 09:30:58','0799113799',0,'DAHDI/63-
1',27,'ANSWERED','0',1,'',1,400,1,'192.168.15.102',NULL,'1010905','1010905'),(328,'0','4006','4004','35','2018-12-11
09:30:06','2018-12-11 09:30:06','2018-12-11
09:30:06','0799113799',0,'0',0,'','3',1,'',1,1,1,'192.168.15.102',NULL,'1010905','1010905'),(329,'0','4006','4004','14','2018-12-11
09:39:05','2018-12-11 09:39:05','2018-12-11
09:39:05','0799113799',0,'0',0,'','1',1,'',1,1,1,'192.168.15.102',NULL,'1010905','1010905'),(330,'0','4006','4004','12','2018-12-11
09:41:47','2018-12-11 09:41:47','2018-12-11
09:41:47','0799113799',0,'0',0,'','1',1,'',1,1,1,'192.168.15.102',NULL,'1010905','1010905'),(331,'118','4006','4004','15','2018-12-
11 09:49:02','2018-12-11 09:49:21','2018-12-11 09:51:01','0799113799',0,'DAHDI/63-
1',99,'ANSWERED','1',1,'',1,1,1,'192.168.15.102',NULL,'1010905','1010905'),(332,'66','4006','4004','15','2018-12-11
09:54:20','2018-12-11 09:54:39','2018-12-11 09:55:27','0799113799',0,'DAHDI/63-
1',48,'ANSWERED','1',1,'',1,1,1,'192.168.15.102',NULL,'1010905','1010905'),(333,'65','4006','4004','17','2018-12-11
10:10:14','2018-12-11 10:10:32','2018-12-11 10:11:19','0799113799',0,'DAHDI/63-
1',47,'ANSWERED','1',1,'',1,1,1,'192.168.15.102',NULL,'1010905','1010905'),(334,'70','4006','4004','40','2018-12-11
12:16:01','2018-12-11 12:16:21','2018-12-11 12:17:11','0799113799',0,'DAHDI/63-
1',50,'ANSWERED','1',1,'',1,400,1,'192.168.15.102',NULL,'1010905','1010905'),(335,'80','4006','4006','11','2018-12-11
12:34:17','2018-12-11 12:34:48','2018-12-11 12:35:38','0786945106',0,'DAHDI/63-
1',50,'ANSWERED','1',1,'',1,400,1,'192.168.15.102',NULL,'1010905','1010905'),(336,'0','4008','4006','88','2018-12-11
12:31:52','2018-12-11 12:31:52','2018-12-11
12:31:52','0786945106',0,'0',0,'','2',1,'',1,400,1,'192.168.15.102',NULL,'889977','1010905'),(337,'0','4006','4004','20','2018-12-
11 01:17:24','2018-12-11 01:17:24','2018-12-11
01:17:24','0799113799',0,'0',0,'','3',1,'',1,1,1,'192.168.15.102',NULL,'1010905','1010905'),(338,'0','40012','4006','43','2018-12-
11 01:17:40','2018-12-11 01:17:40','2018-12-11
01:17:40','0786945106',0,'0',0,'','2',1,'',1,1,1,'192.168.15.102',NULL,'44444444','1010905'),(339,'72','40012','4009','20','2018-
12-11 15:44:02','2018-12-11 15:44:20','2018-12-11 15:45:15','0799010693',0,'DAHDI/63-
1',54,'ANSWERED','1',1,'',1,400,1,'192.168.15.102',NULL,'44444444','5555444'),(340,'65','40012','4009','30','2018-12-11
15:53:13','2018-12-11 15:53:32','2018-12-11 15:54:19','0799010693',0,'DAHDI/63-
1',46,'ANSWERED','1',1,'',1,400,1,'192.168.15.102',NULL,'44444444','5555444'),(341,'85','4008','4006','66','2018-12-11
16:15:52','2018-12-11 16:16:11','2018-12-11 16:17:17','0786945106',0,'DAHDI/63-
1',65,'ANSWERED','1',1,'',1,400,1,'192.168.15.102',NULL,'889977','1010905'),(342,'0','4009','4004','12','2018-12-11
04:15:22','2018-12-11 04:15:22','2018-12-11
04:15:22','0799113799',0,'0',0,'','2',1,'',1,1,1,'192.168.15.102',NULL,'5555444','1010905'),(343,'0','4004','4008','20','2018-12-11
04:16:11','2018-12-11 04:16:11','2018-12-11
04:16:11','0785199013',0,'0',0,'','3',1,'',1,400,1,'192.168.15.102',NULL,'1010905','889977'),(344,'0','4006','4004','111','2018-12-
11 04:22:48','2018-12-11 04:22:48','2018-12-11
04:22:48','0799113799',0,'0',0,'','2',1,'extension=1',1,1,1,'192.168.15.102',NULL,'1010905','1010905'),(345,'78','4006','4004','12'
,'2018-12-11 17:53:26','2018-12-11 17:53:43','2018-12-11 17:54:45','0799113799',0,'DAHDI/63-
1',61,'ANSWERED','1',1,'',1,1,1,'192.168.15.102',NULL,'1010905','1010905'),(346,'0','4009','40012','10','2018-12-12
05:17:05','2018-12-12 05:17:05','2018-12-12
05:17:05','0796388863',0,'0',0,'','3',1,'',1,1,1,'192.168.15.102',NULL,'5555444','44444444'),(347,'68','4006','4004','20','2018-12-
12 13:10:13','2018-12-12 13:10:37','2018-12-12 13:11:21','0799113799',0,'DAHDI/63-
1',44,'ANSWERED','1',1,'',1,400,1,'192.168.15.102',NULL,'1010905','1010905'),(348,'0','4006','4004','11','2018-12-12
08:19:33','2018-12-12 08:19:33','2018-12-12
08:19:33','0799113799',0,'0',0,'','2',1,'',1,1,1,'192.168.15.102',NULL,'1010905','1010905'),(349,'60','4004','40012','12','2018-12-
12 16:42:56','2018-12-12 16:43:13','2018-12-12 16:43:57','0796388863',0,'DAHDI/63-
1',44,'ANSWERED','1',1,'',1,400,1,'192.168.15.102',NULL,'1010905','44444444'),(350,'55','4004','40012','7','2018-12-12
16:45:16','2018-12-12 16:45:33','2018-12-12 16:46:12','0796388863',0,'DAHDI/63-
1',39,'ANSWERED','1',1,'',1,1,1,'192.168.15.102',NULL,'1010905','44444444'),(351,'65','40012','4009','20','2018-12-12
17:09:48','2018-12-12 17:10:06','2018-12-12 17:10:54','0799010693',0,'DAHDI/63-
1',47,'ANSWERED','1',1,'',1,1,1,'192.168.15.102',NULL,'44444444','5555444'),(352,'70','40012','4009','200','2018-12-12
17:40:01','2018-12-12 17:40:25','2018-12-12 17:41:12','0799010693',0,'DAHDI/63-
1',47,'ANSWERED','1',1,'',1,1,1,'192.168.15.102',NULL,'44444444','5555444'),(353,'68','40012','4009','15','2018-12-12
17:43:53','2018-12-12 17:44:10','2018-12-12 17:45:01','0799010693',0,'DAHDI/63-
1',51,'ANSWERED','0',1,'',1,400,1,'192.168.15.102',NULL,'44444444','5555444'),(354,'0','4006','4009','50','2018-12-17
08:28:24','2018-12-17 08:28:24','2018-12-17
08:28:24','0799010693',0,'0',0,'','2',1,'',1,1,1,'192.168.15.102',NULL,'1010905','5555444');
/*!40000 ALTER TABLE `cdr` ENABLE KEYS */;
UNLOCK TABLES;
```

Table F.1: Relevant timing data in tabular format.

Number	Start Time	Answer Time	End Time	Duration
1	12/10/2018 18:47:54	12/10/2018 18:48:16	12/10/2018 18:49:05	71
2	12/10/2018 18:55:01	12/10/2018 18:55:20	12/10/2018 18:55:46	44
3	12/10/2018 18:56:48	12/10/2018 18:57:07	12/10/2018 18:57:56	67
4	12/10/2018 20:05:24	12/10/2018 20:05:56	12/10/2018 20:06:51	87
5	12/10/2018 21:00:35	12/10/2018 21:00:53	12/10/2018 21:01:46	70
6	12/11/2018 9:05:52	12/11/2018 9:06:10	12/11/2018 9:07:02	70
7	12/11/2018 9:30:11	12/11/2018 9:30:30	12/11/2018 9:30:58	46
8	12/11/2018 9:49:02	12/11/2018 9:49:21	12/11/2018 9:51:01	118
9	12/11/2018 9:54:20	12/11/2018 9:54:39	12/11/2018 9:55:27	66
10	12/11/2018 10:10:14	12/11/2018 10:10:32	12/11/2018 10:11:19	65
11	12/11/2018 12:16:01	12/11/2018 12:16:21	12/11/2018 12:17:11	70
12	12/11/2018 12:34:17	12/11/2018 12:34:48	12/11/2018 12:35:38	80
13	12/11/2018 15:44:02	12/11/2018 15:44:20	12/11/2018 15:45:15	72
14	12/11/2018 15:53:13	12/11/2018 15:53:32	12/11/2018 15:54:19	65
15	12/11/2018 16:15:52	12/11/2018 16:16:11	12/11/2018 16:17:17	85
16	12/11/2018 17:53:26	12/11/2018 17:53:43	12/11/2018 17:54:45	78
17	12/12/2018 13:10:13	12/12/2018 13:10:37	12/12/2018 13:11:21	68
18	12/12/2018 16:42:56	12/12/2018 16:43:13	12/12/2018 16:43:57	60
19	12/12/2018 16:45:16	12/12/2018 16:45:33	12/12/2018 16:46:12	55
20	12/12/2018 17:09:48	12/12/2018 17:10:06	12/12/2018 17:10:54	65

Average duration: 70.10 seconds

Standard Deviation of the duration: 15.67 seconds.