Implementing a Software Switch and a Mobile Application to Prevent Frauds and Control the Usage of Electronic Transactions

G T C Liyanage

149218J

Dissertation submitted to the Faculty of Information Technology, University of Moratuwa, Sri Lanka for the partial fulfillment of the requirements of the Degree of Master of Science in Information Technology

May 2017

Declaration

I confirm that this thesis is a presentation of my original research work for the degree of M.Sc. in Information Technology. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions.

The work was done under the guidance of Mr. B. H. Sudantha, at the Faculty of Information Technology, University of Moratuwa, Sri Lanka.

G. T. C. Liyanage

Date:

…

In my capacity as supervisor of the candidate's thesis, I certify that the above statements are true to the best of my knowledge.

Mr. B. H. Sudantha

Date:

Acknowledgement

First of all I would like to express my sincere gratitude to Mr. B. H. Sudantha (Senior Lecturer at Faculty of IT, University of Moratuwa and Course Coordinator) for giving me the opportunity to work under him and to be the supervisor for this research project without second thoughts. His patience and guidance have helped me a lot towards the successful project selection, continuation and completion.

Further I would like to thank especially Prof. A. S. Karunanda (Dean of Kothalawala Defense University) for guiding towards how to do research and how to write thesis. Additionally I would like to thank Mr. S. Premarathne (Senior Lecturer) and all the lecturers who have helped throughout the course by giving unlimited support.

I would also like to thank my superior Mr. Kanishka Weeramunda of PayMedia for helping and guiding in selecting a suitable research area, Mr. Nuwan Wickramanayake and Dilan Ekanayake from InfoTech Department of Sampath Bank for their insights on this project.

Last, but not least I would like to than my wife for enormous support and patience she have been holding throughout my work. And also I wish to thank my parents, brother and all the colleagues for encouraging me in all aspects of life apart from this.

G T C Liyanage

Dedicated

To Mr. B. H. Sudantha

&

To My Wife and Parents

&

To those who lost their valuable money
Because of they had no control over their plastic card

Table of Contents

Table of Figures

List of Tables

Glossary of Terms

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ATM	Automated Teller Machine
CRUD	Create, Read, Update, Delete
DDC	DieBold® Direct Connect
DSS	Data Security Standard
FOS	Free and Open Source
GPL	General Public License
ISO	International Standards Organization
MD5	Message digests - 5
NDC	NCR® Direct Connect
PCI	Payment Cards Industry
POS	Point of Sales
SHA	Secure Hash Algorithm
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
WCF	Windows Communication Foundation
XFS	eXtensions for Financial Services

Abstract

The growth of the electronic payment methods have rapidly increased in past decades. This has been resulted in growth in electronic transactions fraud. The main source of electronic transactions is card payments. People are reluctant to do card payments online because they think twice about the security. On the other hand if a person loses his card or identity of any electronic payment system, he/she has a higher chance of losing their money. ePaySwitch is less complicated but reliable method of preventing a card fraud which gives the control of the card to the user.

ePaySwitch gives the control of the card to the end customer through the mobile application. ePaySwitch server is installed on premises of the bank or financial institute. Server has two main sub modules as ‘Payments Processor’ and the ‘Customer and Cards Manager’ which implemented on two separate servers. These two modules are connected to the database which runs on a separate server. Combination above two modules with database and mobile application together, we call the ePaySwitch.

Customer can keep their cards turned on or off using the mobile application. They even can set whether can perform transactions Online, Offline or for Withdrawal. Additionally they have the facility to set maximum transaction values for each of the above transaction types. This feature enables the customers to protect their money even their cards have lost or stolen.

Mobile application communicates with the server via APIs bind to the mobile application. These APIs are written Java and hosted on an Apache Tomcat server. Payments Processor and the Customers and Cards Manager have written in PHP and hosted on an Apache web server. Database is a MySql GPL version for the current version of ePaySwitch.

Simulation shows how the system will work on real environment. Answers to the questionnaire we prepared shows that some customers’ biggest paint point is fear of losing their money because of cards. Some have showed that they have no idea on their spending when using the cards. But somehow they are willing to user a solution like ePaySwitch if they had a chance.

Finally it is shown that ePaySwitch is a practical solution for prevent frauds and control transactions in the electronic environment.

1 Introduction

1.1 Prolegomena

This research is to provide a solution to the users who are willing to keep a control on their credit cards. Even though there are so many influences to use electronic transaction media, people are reluctant to use these due to two main factors which are fraud and risk of no control over those media.

There have been lots of research done in past to detect fraudulent activity. But there are enough evidences available to show that those researches are not much affected in the actual environment.

When people have no control over their spending in transaction media, they have no idea of how much they will spend, they have spent and what are their limits. The main reason behind this is the invisibility of physical money.

The approach described in this research is novel to the electronic payments systems. This addresses both the above mentioned issues under one system. The method mentioned in the rest of this document can be applied to any financial institute and any type of electronic transactions channel. Nevertheless we are trying to implement it for card based media.

1.2 Background and Motivation

There are several modes of electronic money transactions have been adopted in the market. The most widely available method is usage of a card in form of credit or debit. Few decades back these cards were only available to few businessmen. But with the adaption of technology and as fact of convenience to the customer, now banks and financial institutes are convincing their customers to have a plastic card.

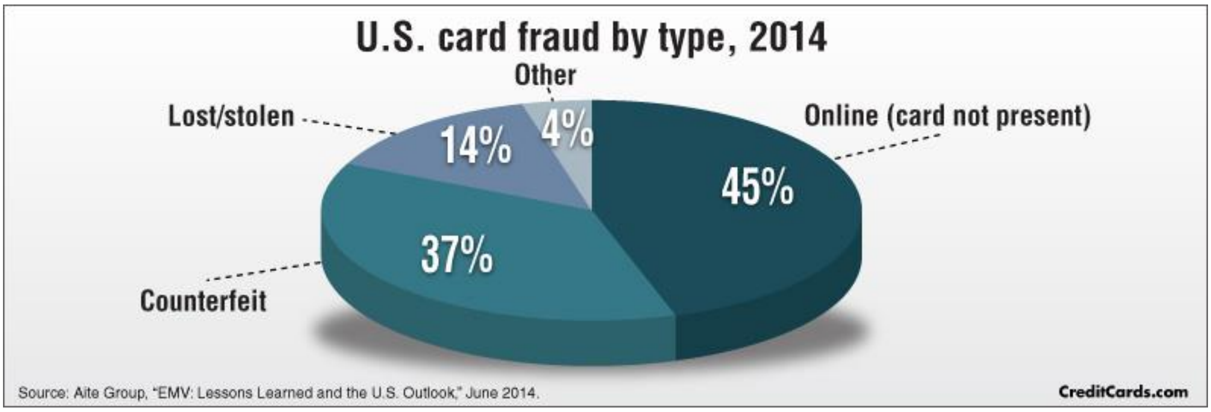
In Sri Lankan aspect when it comes to credit cards, there were only .8M cards in five years back. Later in September 2015 there are around 1.1M active card users by all the providers. However not everybody is convinced. Sri Lankans do not trust electronic means of transactions 100% due to security counter measures. People are not willing to use cards for online transactions because they fear of losing their hard earned money. (Kotelawala, 2016).

This is a global phenomenon. According to Jason they have listed down 9 reasons why most of the Americans do not use credit cards (Jason Steele, 2011). 7 out of the 9 reasons are due to risk of debts. These include risk of debt and budgetary complexity. Other two are due to credit card fatigue and risk of identity theft.

To overcome these facts financial institutes provide rewards for the purchases and loyalty for using the cards. Even though the companies have these programs people are concerned about their security of their transactions and about the spending they make (Holmes, 2016).

"With compromised credit cards and data breaches dominating the headlines in the past couple of years, it's hard not to have some concern about fraud. Technology such as EMV promises to make some payments safer, but experts predict fraud will remain a growing problem for years to come." (Holms, n.d.)

Even though the numbers of frauds happening are increasing, people cannot avoid using cards for the transactions, as cash moves from physical presence to electronic derivatives. There are two types of transactions that are happening with cards as Cards Present (CP) transactions and Cards Not Present (CNP) transactions. Regardless of the way of transaction, the security is an issue.



Source: Aite Group, "EMV: Lessons Learned and the U.S. Outlook," June 2014.

Figure 1-1: U. S. Card Fraud by Type

The main way of fraud where card is present is POS transactions. In the transactions where card is not present are eCommerce and other online payment options. According to the Figure 1-1 extracted from CreditCards.com the CNP transactions will have significant growth in frauds.

These frauds results in millions of dollars of cost to the victims. As per the data from LexisNexis 2004 extracted from CreditCards.com the number of victims in 2010 due to identity theft is 10.2Mn with a value of $20Bn.

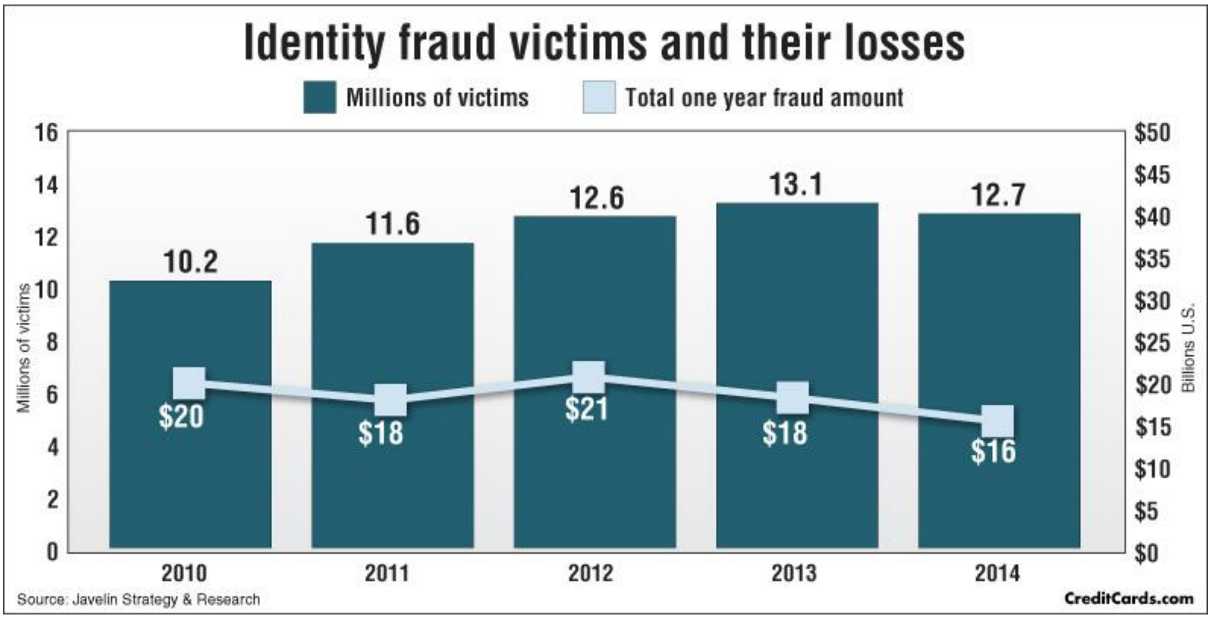


Figure 1-2: ID Fraud Victims and their Loses

With this situation in hand people need a secure way to do transactions in the places where electronic transactions are taking place.

The research is to identify a mechanism to control and increase the cards usage of a customer by integrating existing technologies in financial industry present in a helpful way to secure the electronic transaction of the customers.

1.3 Types of Credit Card Frauds

There are numerous ways to perform a fraudulent transaction in the online environment. Whatever the fraud type all those are come under three main types. Card related, merchant related and internet related (Bhatla et al., 2003).

1.3.1 Card Related Frauds

1.3.1.1 Application related Frauds

Application fraud can be committed in three ways:

- Assumed identity, where an individual illegally obtains personal information of another individual and opens accounts in his or her name, using partially legitimate information.
- Financial fraud, where an individual provides false information about his or her financial status to acquire credit.
- Not-received items (NRIs) also called postal intercepts occur when a card is stolen from the postal service before it reaches its owner’s destination.

This type of fraud occurs when a person falsifies an application to acquire a credit card.

1.3.1.2 Lost or Stolen cards

This type of fraud is in essence the easiest way for a fraudster to get hold of other individual's credit cards without investment in technology. Card is lost/stolen when a legitimate account holder receives a card and loses it or someone steals the card for criminal purposes. It is also perhaps the hardest form of traditional credit card fraud to tackle.

1.3.1.3 Account Takeover

The fraudster takes control of (takeover) a legitimate account by either providing the customer’s account number or the card number. The fraudster then contacts the card issuer, masquerading as the genuine cardholder, to ask that mail be redirected to a new address. This type of fraud occurs when a fraudster illegally obtains a valid customers’ personal information. The fraudster reports card lost and asks for a replacement to be sent.

1.3.1.4 Counterfeit Cards

The creation of counterfeit cards, together with lost / stolen cards poses highest threat in credit card frauds. Fraudsters are constantly finding new and more innovative ways to create counterfeit cards.

Some of the techniques used for creating false and counterfeit cards are listed below;

- Erasing the magnetic strip
- Creating a fake card
- Altering card details
- Skimming

1.3.2 Merchant Related Frauds

1.3.2.1 Merchant Collusion

This type of fraud occurs when merchant owners and/or their employees conspire to commit fraud using their customers' (cardholder) accounts and/or personal information. Merchant owners and/or their employees pass on the information about cardholders to fraudsters.

1.3.2.2 Triangulation

This process is designed to cause a great deal of initial confusion, and the fraudulent internet company in this manner can operate long enough to accumulate vast amount of goods purchased with stolen credit card numbers. Once fraudsters receive these details, they order goods from a legitimate site using stolen credit card details. The customer while placing orders online provides information such as name, address and valid credit card details to the site. The fraudster then goes on to purchase other goods using the credit card numbers of the customer.

1.3.3 Internet Related Frauds

With the expansion of trans-border or 'global' social, economic and political spaces, the internet has become a New World market, capturing consumers from most countries around the world. The Internet has provided an ideal ground for fraudsters to commit credit card fraud in an easy manner.

The most commonly used techniques in internet fraud are described below;

- Site cloning
- False merchant sites
- Credit card generators

Al of these transaction frauds are falling into two types.

- Card Present (CP) Frauds
- Card Not Present (CNP) Frauds

1.4 Problem Statement

Background specifies the reasons behind people are reluctant to use credit cards for their transactions in any channel. Even though there are so many researches have been done in the area of card fraud detection, there are lesser number of researches in prevention and controlling of electronic transaction media.

This emphasizes the problem as there is no proper mechanism to prevent electronic transactions fraud on cards other than detecting them and there is no way of taking control over electronic transaction by the users themselves.

1.5 Hypothesis

Introducing a software switch which can be controlled by a mobile application to the end user to add and control their electronic transaction channels for cards will help in prevention of electronic transactions fraud and also will increase the electronic transactions usage as they have the control over their different channels.

1.6 Aims and Objectives

Main goal of the study is to provide a software switch to the financial institutes to lie on top of their existing infrastructure and a mobile application to access this switch.

- The customers will be able to maintain their card repository in one mobile application.
- Using this repository they will be able to control following factors for each card saved in the repository;
    - Enable or Disable (On or Off) a particular card for transactions
    - Enable or Disable (On or Off) a particular card for a particular type of transaction (E.g. POS, Web Transaction)
    - Set a maximum value for transactions
- In a case where a customer has lost any card, he/she will be able to disable that card using the mobile application. So no one can use the card for any transaction anymore.
- Motivate the customer to use cards for the transactions without any fear.
- To provide security in terms of following types for both customer and the card issuer;
    - Card Present (POS)
    - Card Not Present (Web/Online)

Apart from above mentioned this study will include:

- A study on how others have implemented previous systems
- What are the trends in electronic transaction fraud prevention
- How current electronic transaction control systems are established
- Develop a DSS compatible software switch
- Develop a DSS compatible mobile application
- Evaluate the validity and adaptability of the solution

1.7 Software Switch and Mobile Application Based Approach

The overall solution stands in two pillars. One is the software switch named ePaySwitch and the ePaySwitch mobile application.

The solution addresses the problem before a fraud happens or the user controls the transaction channel beforehand. The mobile application is just an interface which communicates with the ePaySwitch. The communication channel is encrypted. The ePaySwitch holds the necessary data that has been already provided by the user.

When a user needs to do a transaction using the credit card first he/she has to login to the system using the mobile application. It will list down all the registered payment channels. User has to select the credit card option and need to activate it and also need to set a limit to the transaction if necessary. When the transaction is done, the system first looks whether the card is active and then the maximum limit. If these criteria met, then the message is forwarded to the core system. Otherwise sends a decline message back.

1.8 Structure of Thesis

Chapter 2 reviews the fraud detection; prevention and transaction control systems and identify the research problem. Then chapter 3 discuss about the technology underlying the base of the proposed solution. Chapter 4 includes the approach to this novel solution and chapter 5 has the design of system. Chapter 6 and 7 has the implementation of the system and evaluation of the new system. Chapter 8 presents the conclusion and future work.

1.9 Summary

In this chapter the introduction to the problem and solution presented with background to the research, problem definition, hypothesis and brief overview of the solution was presented.

In next chapter the review of the others work is critically analyzed and presented.

2 Study on Electronic Transaction Frauds and Control of Transactions

2.1 Introduction

This chapter emphasizes what other researchers have done in the past in order to overcome the card frauds. Further the chapter explains the algorithms used in the industry and also manual processes which currently in use.

Additionally the problems of the current systems are also discussed.

2.2 Related Work on Credit Card Fraud Detection

Over the past two decades there have been numerous researches have been done in order to detect card frauds. On doing the literature survey we found that there are several methods available to detect these frauds.

Researchers developed many credit card fraud detection techniques based on data mining loom. Ghosh and Relley (FFNN), which requires long training time. CardWatch proposed that a neural network based database mining scheme which was a trial product for database mining system developed for credit card fraud detection application and is concerned that it requires one network per customer (Aleskerov et al., 1997). BLAST-SSAHA hybridization technique (Kundu et al., 2009) of credit card fraud by online detection Rilly have proposed credit card fraud detection with a three-layer approach, feed-forward neural network.

BLAST-SSAHA approach improves the fraud detection by combining both peculiarities as well as misuse detection techniques. Chiu et al have introduced web-services based collaborative scheme for fraud detection in the Banks (Chiu and Tsai, 2004). The proposed scenario supports the sharing of knowledge about fraud pattern with the participant banks in a heterogeneous and distributed environment. Hidden Markov model (HMM) (Srivastava et al., 2008) for credit card fraud detection which shows 80% accuracy over a large variation in the input data. An approach to enhanced speed by using equivalent coarse neural network of data mining and knowledge discovery process (KDP) for credit card fraud detection and achieve reasonable speed up to 10 processors only & more number of processors introduces load imbalance problem (Syeda et al., 2002). Markov Model and time series are not scalable to large size data sets due to their time complexity.

The application of distributed data mining in credit card fraud detection (Chan et al., 1999) and improve the efficiency of highly distributed databases and detection system as this approach uses Boosting algorithm name Ada Cost. Ada Cost uses large number of classifiers and requires more computational resources during detection. Brause combines advanced data mining techniques and neural network algorithms (Brause et al., 1999). Study of Stolfo intimate a credit card fraud detection system using various meta- learning techniques to learn models of fraudulent credit card transactions (Stolfo et al., 1998). To achieve high fraud finding along with low forged alarm Elkan suggest Naïve Bayesian approach for credit card fraud detection. Further, Elkan and Witten presents that NB algorithm is very effective in many real world data sets as well as extremely capable in linear attributes. Bayesian networks were faster and accurate to train but are slower when applied to new instances/occurrence in an online system. Vatsa have currently proposed a game- theoretic approach to credit card fraud detection (Vatsa et al., 2005). Wen-Fang have recommended a research on credit card swindle detection model which is based on outlier detection mining on distance sum, which shows that it can detect credit card fraud better than anomaly detection based on clustering. Jianyun have shown outline for detecting falsified transactions. In his paper work describes an FP tree based method to effectively create user profile for finding of fraud. However on the other hand, this method doesn’t be familiar with atypical patterns i.e. short term behavioral changes of genuine card holders.

Today, some of the existing credit card fraud detection techniques which use labeled data to train the classifiers are unable to detect new kinds of frauds. Supervised learning has some drawback, that they require human involvement to optimize parameters. On another hand, decision tree do not require any parameter setting from the user and can build faster compared to other techniques.

2.2.1 Bayesian Networks

First, a Bayesian network is constructed to model behavior under the assumption that the user is fraudulent (F) and another model under the assumption the user is a legitimate (NF). For the purpose of fraud detection, two Bayesian networks to describe the behavior of user are constructed. By inserting evidence in these networks and propagating it through the network, the probability of the measurement x less than two above mentioned hypotheses is obtained. During operation user net is adapted to a specific user based on emerging data.

2.2.2 Hidden Markov model

HMM (Srivastava et al., 2008), Baum Welch algorithm is used for training purpose and K-means algorithm for clustering. HMM sores data in the form of clusters depending on three price value ranges low, medium and high (Bhusari and Patil, 2011). If an incoming credit card transaction is not accepted by the trained Hidden Markov Model with sufficiently high probability, it is considered to be fraudulent transactions. A Hidden Markov Model is a double embedded stochastic process with used to model much more complicated stochastic processes as compared to a traditional Markov model.

2.2.3 Genetic Algorithm

Genetic algorithms, inspired from natural evolution were first introduced by Holland (1975). These algorithms are evolutionary algorithms which aim at obtaining better solutions as time progresses. Fraud detection problem is classification problem, in which some of statistical methods many data mining algorithms have proposed to solve it. GA has been used in credit card fraud detection for minimizing the wrongly classified number of transactions (Duman and Ozcelik, 2011). And is easy accessible for computer programming language implementation, thus, make it strong in credit card fraud detection. GA is used in data mining mainly for variable selection and is mostly coupled with other DM algorithms (Duman and Ozcelik, 2011).

2.2.4 Decision Tree

Decision trees are predictive decision support tools that create mapping from observations to possible consequences. Decision tree usually separates the complex problem into many simple ones and resolves the sub problems through repeatedly using (Raj and Portia, 2011; Sahin and Duman, 2011). Classification rules, extracted from decision trees, are IF-THEN expressions and all the tests have to succeed if each rule is to be generated (Raj and Portia, 2011). The work demonstrates the advantages of applying the data mining techniques including decision trees and SVMs to the credit card fraud detection problem for the purpose of reducing the bank's risk. Decision trees are statistical data mining technique that express independent attributes and a dependent attributes logically AND in a tree shaped structure. The results show that the proposed classifiers of C&RT and other decision tree approaches outperform SVM approaches in solving the problem under investigation. There are number of popular classifiers construct decision trees to generate class models.

2.2.5 Neural Networks

An artificial neural network (Chang et al., 2007; Raj and Portia, 2011) consists of an interconnected group of artificial neurons .The principle of neural network is motivated by the functions of the brain especially pattern recognition and associative memory (Patidar and Sharma, 2011). The neural network recognizes similar patterns, predicts future values or events based upon the associative memory of the patterns it was learned. The advantages of neural networks over other techniques are that these models are able to learn from the past and thus, improve results as time passes. Fraud detection methods based on neural network are the most popular ones. Among the reported credit card fraud studies most have focused on using neural networks. In more practical terms neural networks are non-linear statistical data modeling tools. By employing neural networks, effectively, banks can detect fraudulent use of a card, faster and more efficiently.

On other hand, the unsupervised techniques do not need the previous knowledge of fraudulent and non-fraudulent transactions in database. In supervised training, samples of both fraudulent and non-fraudulent records are used to create models. In contrast, unsupervised training simply seeks those transactions, which are most dissimilar from the norm. There are two phases in neural network (Guo and Li, 2008) training and recognition. There are two types of NN training methods supervised and unsupervised. Learning in a neural network is called training. And they need a long training dataset.

2.3 Related work on Credit Card Fraud Prevention and Control

While fraudsters are using sophisticated methods to gain access to credit card information and perpetrate fraud, new technologies are available to help merchants to detect and prevent fraudulent transactions. Fraud detection technologies enable merchants and banks to perform highly automated and sophisticated screenings of incoming transactions and flagging suspicious transactions.

2.3.1 Manual Review

Moreover, manual review is unable to detect some of the more prevalent patterns of fraud, such as use of a single credit card multiple times on multiple locations (physical or web sites) in a short span. This method consists of reviewing every transaction manually for signs of fraudulent activity and involves an exceedingly high level of human intervention.

2.3.2 Address Verification System

Address Verification System (AVS) matches the first few digits of the street address and the ZIP code information given for delivering/billing the purchase to the corresponding information on record with the card issuers.

2.3.3 Card Verification Methods

The purpose of Card Verification Method (CVM) is to ensure that the person submitting the transaction is in possession of the actual card, since the code cannot be copied from receipts or skimmed from magnetic stripe. CVM consists of a 3- or 4-digit numeric code printed on the card but is not embossed on the card and is not available in the magnetic stripe.

2.3.4 Negative and Positive Lists

A negative list is a database used to identify high-risk transactions based on specific data fields.

An example of a negative list would be a file containing all the card numbers that have produced chargebacks in the past, used to avoid further fraud from repeat offenders. Similarly a merchant can build negative lists based on billing names, street addresses, emails and internet protocols (IPs) that have resulted in fraud or attempted fraud, effectively blocking any further attempts.

Another popular example of negative list is the SAFE file distributed by MasterCard to merchants and member banks.

This list contains card numbers, which could be potentially used by fraudsters, e.g., cards that have been reported as lost or stolen in the immediate recent past.

2.3.5 Payer Authentication

The program is based on a Personal Identification Number (PIN) associated with the card, similar to those used with ATM cards, and a secure direct authentication channel between the consumer and the issuing bank. The first implementation of this type of service is the Verified by Visa (VbV) or Visa Payer Authentication Service (VPAS) program, launched worldwide by Visa in 2002.

2.4 Problem Definition

So far we have discussed about various techniques that have been suggested by different researchers. Those researches have both successful and difficult to implement solutions as well.

Most of these theories are based on advanced algorithmic functions. The of fraud detection methods using KDA (Vadoodparast et al., 2015) has suggested a more complex theory.

Even though there are enough research for the card fraud detection, there are less research on prevention. The research on Cryptographic Algorithm (Meshram and Yenganti, 2013) is a prevention technique which can be used only for ATM transactions.

Other prevention techniques are mostly pen and paper methods which are practically available (Bhatla et al., 2003). For a digital era these methods cannot be applied at every scenario.

Therefore the final outcome from the literature survey is that there is gap of research for prevention of the card related frauds. The problem stands here is how a simple but reliable fraud prevention method can be invented.

2.5 Summary

In this chapter we have discussed about the fraud detection and prevention techniques that are currently in use and their problems. We have identified the research gap in the related research area.

Next we will discuss about the technological aspect of the available electronic payments methods.

3 Technology of Banking in Electronic Environment

3.1 Introduction

This chapter describes about the technology used in the industry as well as the used in the ePaySwitch. The chapter further explains about the banking systems technologies and how they have been adopted in exposing their services to the public.

3.2 Standard Banking Systems

Several banking systems use several methodologies to develop banking systems, namely Core Banking System. In this standard banking systems, underlying communication and transaction handling protocol is an ISO standard protocol. Above mentioned protocol is globally known as ISO8583 Financial transaction card originated messages - Interchange message specification. This specification is a standard for electronic transactions.

3.2.1 ISO8583 Protocol

There are several methodologies used in the industry to access core banking functions via ISO8583 in payment systems. Most popular among ATMs are NDC & DDC which are proprietary wrappers for ATM switches. Other methodology is CEN[1]/XFS. And there some other wrappers and technologies which are used in POS machines and web. Whatever the wrapper is every transaction communicates with the core banking system with ISO8583 (“ISO 8583,” 2016) which is an ASCII based protocol. This protocol mainly uses socket communication.

The implementation of this protocol differs from one to another. However the logic behind the communication is common for all the implementations. ISO8583 message consists of following main components that will be used in the ePaySwitch implementation which describes later.

3.2.1.1 MTI – Message Type Indicator

This is the first for digits of an ISO8583 message which is used to identify the version of the protocol and especially ye type of the message. Each digit of the MTI has different meaning as below.

[1] CEN is a European Committee for Standardization

0xxx - version of ISO 8583 (for example: 1987 version)
x2xx - class of the Message (for example: Transaction Message)
xx1x - function of the Message (for example: Request/Response)
xxx0 - who began the communication (for example: Acquirer)

3.2.1.2 Bitmap(s)

After MTI, there will be one; two or three (later versions) bitmaps consists of 64, 128 or 192 digits respectively. Each digit represents the presence or the absence of the data elements which will be passed to the core banking system using 1 or 0.

Some implementations have used series of binary numbers up to 64, 128 or 192. Some have converted these binary series to hexadecimal series to shorten the message length as shown below.

<u>Binary representation of bitmap for 128 data elements.</u>
111110101011110001000100110000010010100000100000110000000000000000000000
00000000000000000000000000000100000000000000000000000000

<u>Hexadecimal representation of bitmap for above.</u>
FABC44C12820C000000000004000000

In above representation, every data element represented by 1 is sent to core system and 0s will be omitted. Bitmap it used for alert the core system that what elements will be attached in the rest of the message.

<u>Elements carried in the above message includes;</u>
1-2-3-4-5-7-9-11-12-13-14-18-22-25-26-32-35-37-43-49-50-103

3.2.1.3 Data Elements

Data elements are the individual fields which carries the transactional information. Some data elements have a specific meaning and some data elements are variable according to the implementation. Some of those data elements are as follows.

[001] Bitmap (Binary 64, 128 or 192)
- 111110101011110001000100110000010010100000100000110000000000000000
  00000000000000000000000000000000000100000000000000000000000000

[002] Primary Account Number (Numeric 19 max with 2 digits length)
- 164691760100091433 (first two digits are the length of account number)

[003] Processing Code (Numeric 6)
- 011000

[004] Transaction Amount (Numeric 12)

- 000000500010 (5000.10)

[049] Currency code, transaction (Alphanumeric 3)

- LKR

[43] Card Accepter Terminal Identification (Alphanumeric or Special 40)

- TWELLAW3COMMERCIAL_BANKWELLAWATTA_3__BR_____WELLAWATTA___COLK

3.2.1.4 Response Codes

Every ISO8583 request has a response. Response is sent back in the field 39. There are different types of response codes available in the standard ISO8583 specification. Following Table 3-1 shows the codes which will be used by the ePaySwitch.

Code	**Meaning**
00	Successful
07	Declined
51	Not Enough Funds
59	Suspicious

Table 3-1: ISO8583 Common Response Codes

Above mentioned communication protocol is vital for the development for the ePaySwitch as it will be connected to the core banking system which will be explained later chapters.

3.3 APIs for Public Access

Banks or financial institutes have opened some of their banking operations for their customers as self-service functionalities. Most common way of opening those into the public is via secure APIs. In this way bank can secure their core banking functionalities.

In the industry there are several kind of implementation in form of Java based APIs, WCF based APIs etc. Even though development effort is less in WCF, most widely used technology is Java as it is FOS.

3.3.1 Usage of Identity servers and API managers

Financial institutes are using identity servers to secure the electronic transactions via tokenization. For further security, API managers have been used. API manager hides the real APIs from the public with the help of identity management and releases a mock API to the public.

Any user who is willing to access a service should have a valid token. This mechanism increases the security. Nevertheless, this is not helpful in preventing and controlling electronic transactions.

3.4 Personal Banking

Via public APIs and other services opened to the public, banks have given personal banking services to the customers. The technologies used in this scenario are different from one institute to another based on their policies and capacity to implement such services. Advances in technology allows the delivery of banking products and services more conveniently and effectively to the customer than ever before - thus creating new bases of competition.

3.4.1 Web Presence

To make the system user friendly to all clients, banks have used a Graphical User Interfaces (GUI) with attractive and user-friendly layouts. Customers can access their bank details on their own computers, make money transfers from one account to another, print bank statements and inquire about their financial transactions using online servises.

Most common and widely used technologies here are proprietary or open source client server technologies integrated with identity servers and/or API managers.

3.4.2 Mobile Presence

Mobile representation of banks and financial institutes is the most convenient and most trending means of interacting with the bank. Mobile Applications are the widely used form of mobile presence of banking functionalities.

When the number of mobile users increases, number of treats is also increasing. Technology creates a barrier for those treats for some extent. But as discussed in Chapter 2, number of frauds (treats) has increased over time.

3.5 Encryption Functions

There are several industry standard encryption functions available. Most common types are MD5 (“MD5,” 2017) and SHA (“SHA-1,” 2017). A comparison between these two standards is shown in Table 3-2.

Key	**MD5**	**SHA1***
Security	Less secure than SHA	More secure than MD5
Message Digest Length	128 bits	160 bits
Attack required to find original message	2^{128} bit operations	2^{160} bit operations
Attacks to try and find two messages producing the same MD	2^{64} bit operations	2^{80} bit operations
Speed	Faster. 64 iterations	Slower 80 iterations
Successful attacks so far	Reported to some extent	No reported attacks so far

Table 3-2: MD5 and SHA Comparison

*SHA1 is the first version of SHA. Later SHA2 was released with more security in forms of SHA128, SHA256, SHA384 and SHA512. It works the same way as SHA1 but is stronger and generates a longer hash.

3.6 Technologies used in Novel Solution

ePaySwitch uses three technologies mentioned above.

1. Mobile Application
   - Solution includes a mobile application which will be used to control the cards of the customers. The application will be written on Android at first.
2. APIs for Mobile Access.
   - Solution has APIs which will be consumed by the mobile application. Those APIs are written in Java as most of the financial institutes running there servers on Linux platform and the Apache Tomcat is FOS.
3. Payments Processor
   - Payments processor uses ISO8583 message encoding and decoding using PHP. As mentioned above, Apache web server also free and hosted in most of the financial institutes.

Additionally ePaySwitch uses SHA256 encryption function to encrypt sensitive data. This is more important in achieving the PCI-DSS standard for the solution. PCI-DSS is further explained in Chapter 4.

3.7 Summary

ePaySwitch solution is combination of Android, Java and PHP, which will uses ISO8583 industry standard protocol to build a barrier to card based transaction frauds and give a control to the customers over their transactions.

In this chapter we have discussed about the technologies used in the industry and how those technologies are adapted in the ePaySwitch. Next we will elaborate the approach to ePaySwitch.

4 An Approach to Prevent Card Fraud and Misuse

4.1 Introduction

This chapter emphasizes the approach to the novel method to prevent transaction frauds and control their usage, which is out research problem. The sections covered here are the main inputs, outputs to the system, processes, users and features of the proposed solution.

Hypothesis

Introducing a software switch which can be controlled by a mobile application to the end user to add and control their electronic transaction channels will help in prevention of electronic transactions fraud and also will increase the electronic transactions usage as they have the control over their different channels.

4.2 ePaySwitch

Card is channel of electronic transactions. The novel approach that will be used to prevent card frauds and misuse of the cards which is going to discuss rest of this document will call ePaySwitch. The name suggests electronic payments can be switched at the customers' convenience.

4.3 Inputs to the system

Inputs to the ePaySwitch has three forms as inputs from user, baking officer and core banking system

4.3.1 User Inputs

User enters the data to the system using the ePaySwitch mobile application. The primary data would be;

- Registration information
    - First Name & Last Name
    - NIC Number
    - Mobile Number
    - Username & Password
- The login information
    - Username & Password
- Card information

    - Name in Card
    - Card Number
    - Expiry Date
    - Card Alias
- Active/Inactive state
    - On/Off
- Transaction Amounts
    - Online, Offline & Withdrawal Limits

4.3.2 Input from Banking Officer

Banking officer will access a module called ‘Customer and Cards Manager’ which will be described later. The inputs to the system from bank officer will be;

- Customer Activation/Deactivation
    - On/Off
- Card Activation/Deactivation
    - On/Off

4.3.3 Core System Inputs

The core system of the financial institute may push following data to ePaySwitch when a transaction is initiated.

- Card information
    - Plain test information from ISO8583 message
- Response from core system
    - Responses for a particular transaction as an ISO8583 message

4.4 Outputs of the system

Output from the system has two forms as Output from ePaySwitch and Mobile Application.

4.4.1 Output from ePaySwitch

Outputs from ePaySwitch include:

- Active/Inactive status
    - On/Off Status of Online, Offline & Withdrawals

- Accepted/Declined status
- Card information
  - Masked Card Number
  - Owner name
  - Expiry
- Transaction amounts
  - Online, Offline & Withdrawal Limits

4.4.2 Outputs from ePaySwitch Mobile Application

Outputs from ePaySwitch mobile application include:

- Active/Inactive status
  - On/Off Status of Online, Offline & Withdrawals
- Accepted/Declined status
- Card information
  - Masked Card Number
  - Owner name
  - Expiry
- Transaction amounts
  - Online, Offline & Withdrawal Limits

4.5 Processes

4.5.1 User Registration and Authentication

The institute have the ability to make customers ge registered in the ePaySwitch via the mobile application. Currently the system allows users to register under a single institute. Registration process will create an account in the ePaySwitch.

Then an officer from bank should activate that particular customer. Then the registered customer can login to the system by entering the username number and password. The system returns the status of the authenticity of the account to the mobile application.

A user can change the password number whenever needed.

4.5.2 Card Registration

Registering a card to the system is only possible through the mobile application. When a user needs to add a card to the system, he/she need to enter required details in the mobile application. An officer from bank needs to manually verify the card against the customer and activate it. This has been applied to prevent registration of stolen cards.

4.5.3 Card Control

Users can control their card by turning on or off to perform transactions and also setting the transactional limit to particular type of transactions. When a card is turned off then the payment channel will get a declined message. If the transaction limit exceeded then a not enough credit message (decline) message will be pushed.

4.5.4 Mobile Application to ePaySwitch Communication

The communication between the mobile application and the ePaySwitch is done via a public network using APIs. ePaySwitch opens the APIs to public which is bonded to the mobile application.

4.5.5 Communication with ePaySwitch and the Core System

Generally core system uses a socket communication protocol which is ISO8583. This protocol has preconfigured standard. The way is communicate with socket is different from one implementation to another. ePaySwitch takes the input from the mobile application and formats it according to the core system specification.

When the communication mechanism is not ISO8583 then standard APIs are used to communicate with the core system.

4.6 Users of system

The system is beneficial mainly for two sectors namely financial institutes and the end users.

4.6.1 Financial Institutes

Financial institutes can use ePaySwitch to provide a better value added service to the end users. They can get a competitive advantage over those who does not provide fraud preventions and control solution for cards.

4.6.2 End Users

The customers can use the solution in order to prevent their cards been exposed to the public unnecessarily. Even though, they have lost their card without knowing, there is nothing to worry about.

4.7 Features of ePaySwitch

ePaySwitch is developed to suit the general customer and also to comply with the industry standards of the domain. To comply with all the aspects in general following main features has to be considered.

Secure

End to end transmission from mobile application to the switch is encrypted using private key cryptography to ensure the security of the data and privacy.

Easy and Simple to Use

The interface of the mobile application is designed to suit almost each and every user. Minimized the complexity of the UI to adhere with industry standard guidelines.

Robust

The system should fault tolerant. Because, number of users and number of transactions in the actual implementation is too high.

ISO8583 Compatible

The system has the capability to communicate with any core banking system as per their configuration.

PCI-DSS Compatible

PCI-DSS is a set of guidelines that has published which needs to be embedded to any application for usage of finance purposes. The mobile application and server will have the minimum required security standards to meet this requirement.

4.8 Summary

In this chapter we have elaborated about the input, output, process, users and features of the ePaySwitch solution.

In the next chapter we will explain the design of ePaySwitch in detail.

5 Design of ePaySwitch

5.1 Introduction

Previous chapter described about the approach on what are the inputs, outputs and processes of the system. This chapter emphasizes those factors in a designer's point of view. The complete solution has two main components as the software switch and the mobile application.

5.2 Architecture of ePaySwitch

The following block diagram, use case diagram and database diagram illustrate the scenarios in the complete ePaySwitch solution.

Block Diagram

Figure 5-1 shows the block diagram of the overall solution. The design of individual component will be discussed later.

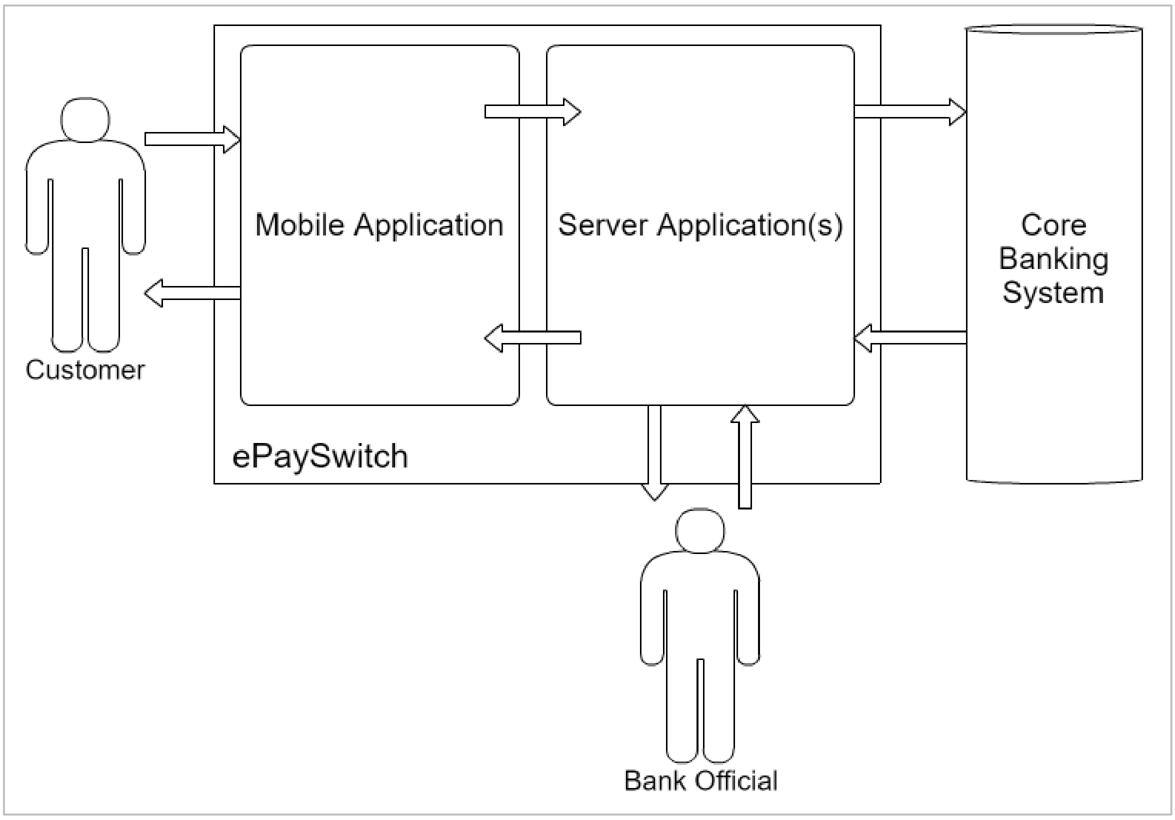


Figure 5-1: High Level Architecture

User interacts with the mobile application. The mobile application first connects to the ePaySwitch. The decision on whether to communicate with the core system or not will be decided upon the preferences saved by customer in ePaySwitch.

5.2.1 Server Architecture

ePaySwitch combines three different servers to run separate services as mentioned below.

1. Web-Server [Tomcat]
   - To be used as the platform for run web APIs to be consumed by the mobile application.
2. Web-Server [Apache]
   - Payments processor and the User and Card manager runs on top of an Apache web server.
3. Database server
   - A separate database server to be used to isolate data from services.

Figure 5-2 illustrates the total ePaySwitch server architecture in block diagram.

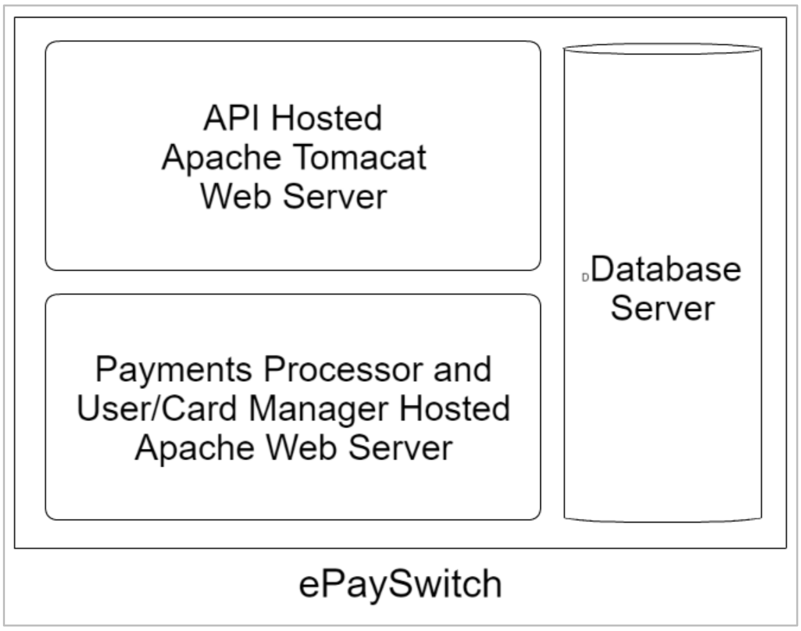
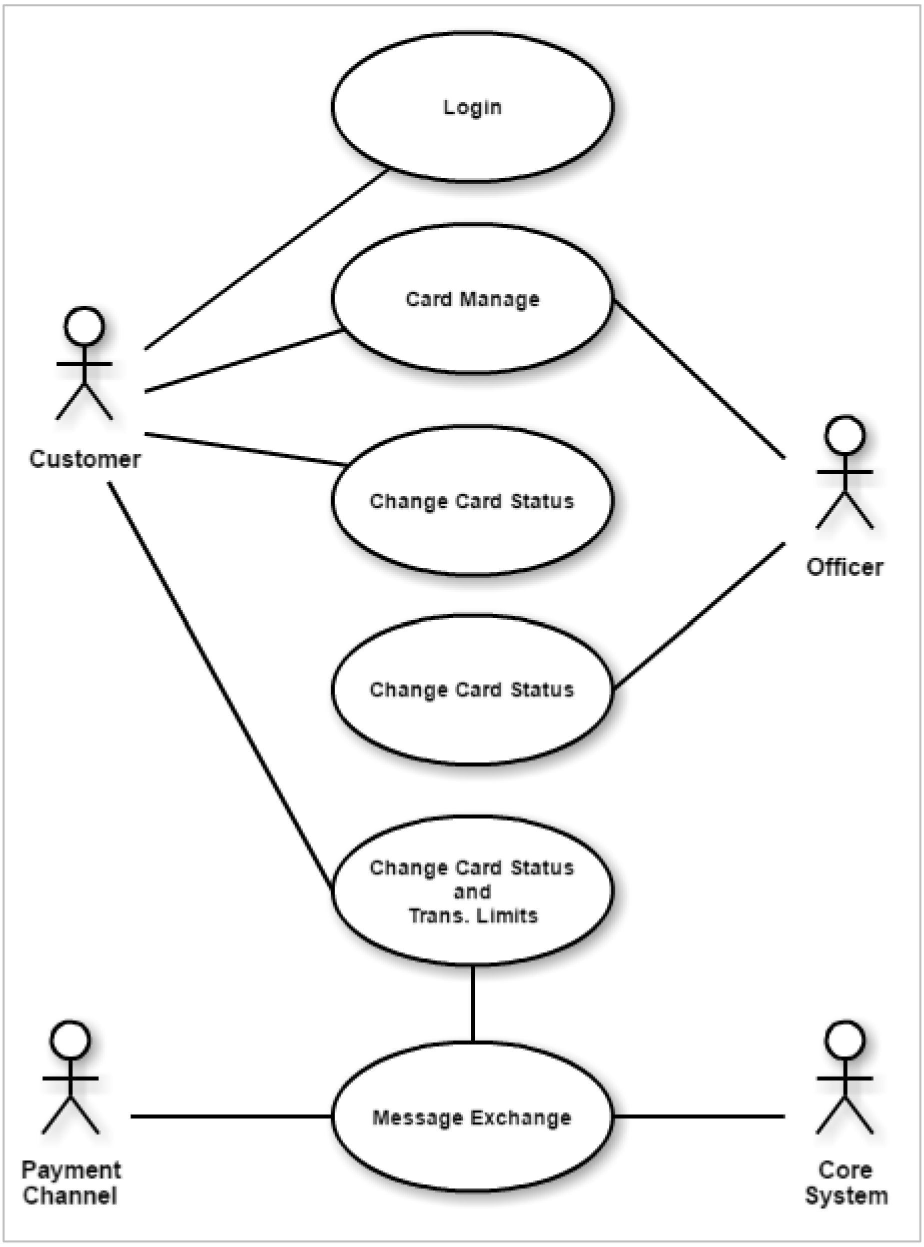


Figure 5-2: ePaySwitch Server Block Diagram

5.2.2 Use Case Diagram



Following

Figure 5-3 of use case diagram further elaborates the overall system usage. Mainly four parties get involved in a running ePaySwitch

The end user has the most of the authority over the control of credit cards. ePaySwitch manages the entire communication and other information processing requests according to the given criteria by the end user.

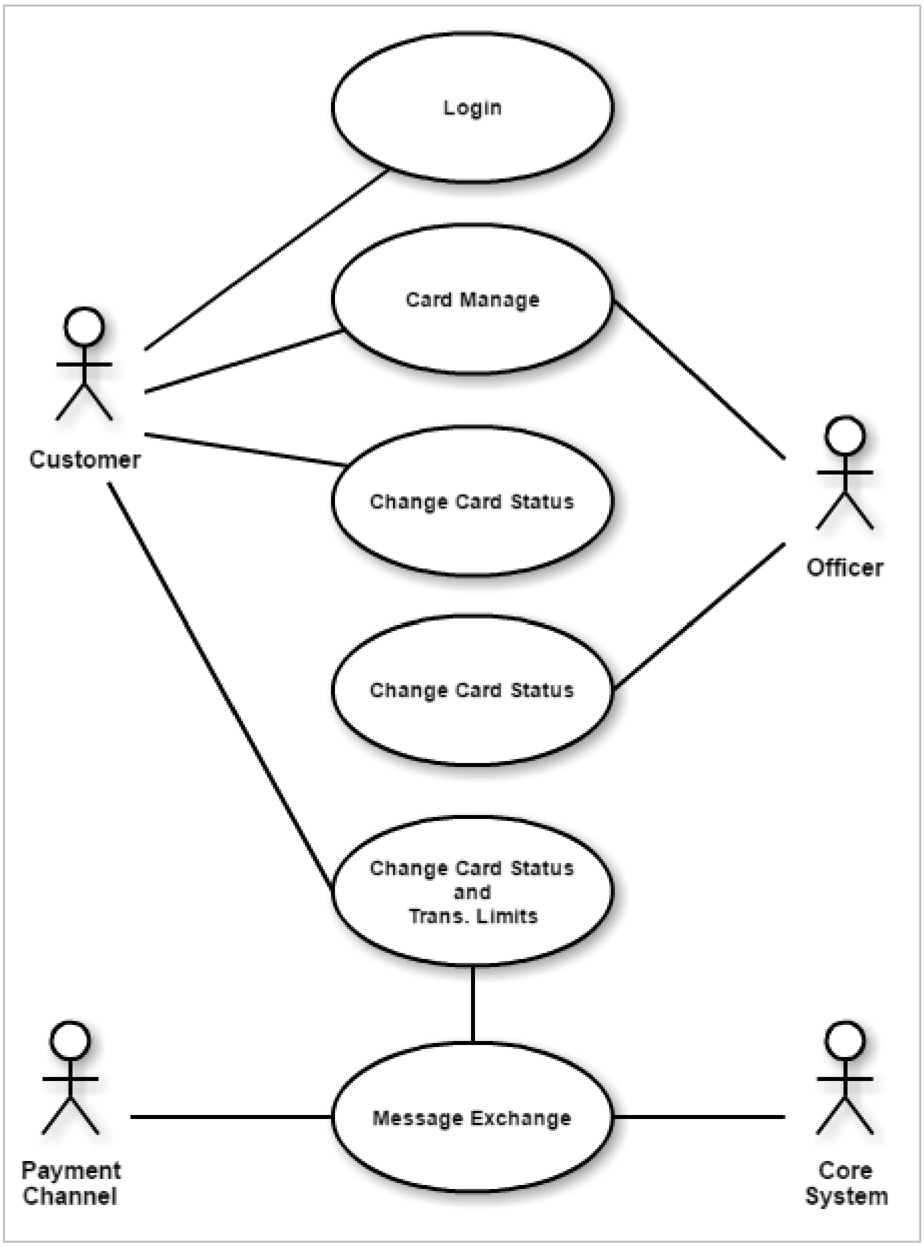


Figure 5-3: High Level Use Case Diagram

5.2.3 Activity Diagram

Figure 5-4 shows the activity diagram for ePaySwitch.

Server receives the transaction message and decodes it and check with rules saved by the customer and responds accordingly

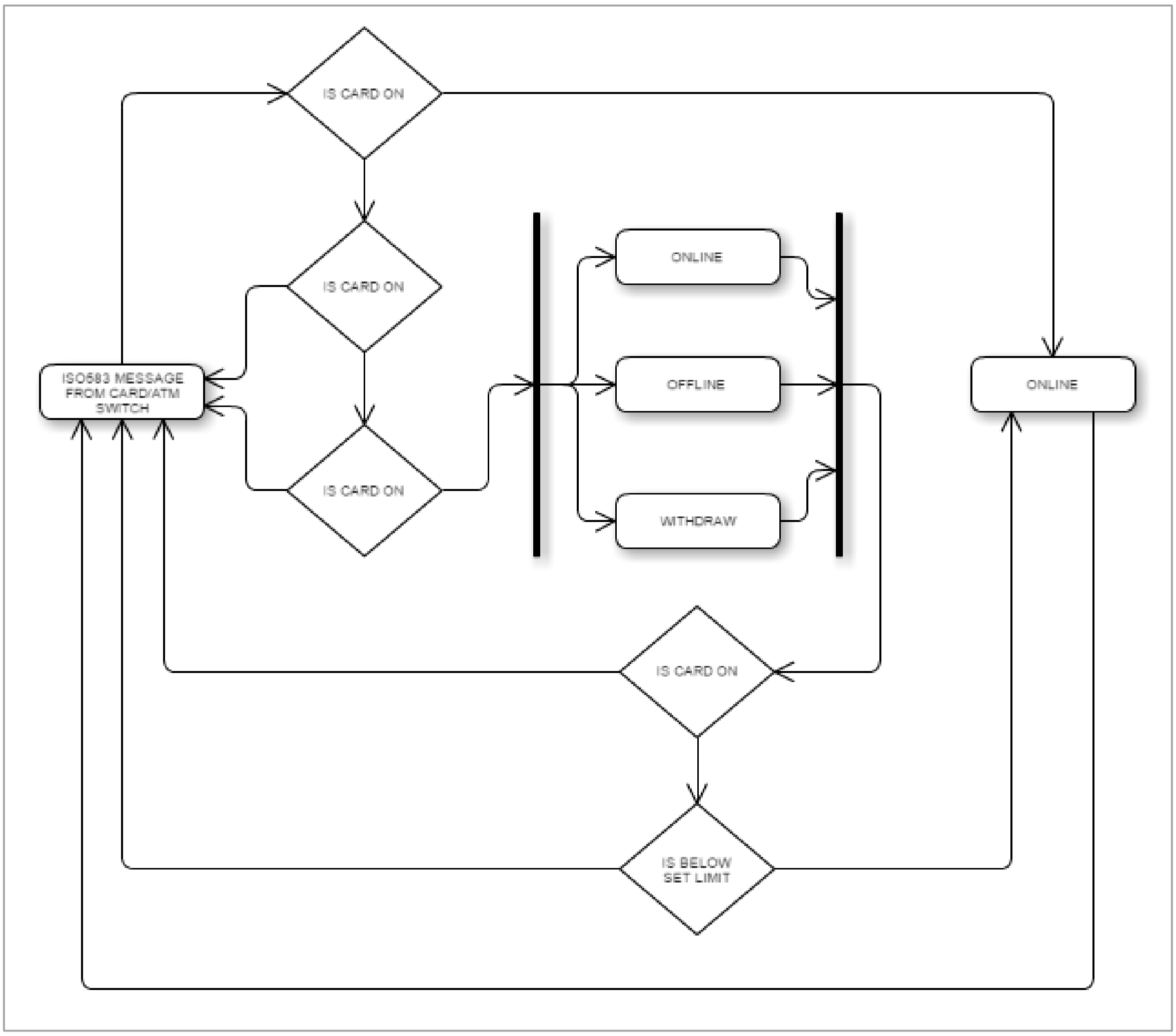


Figure 5-4: Activity Diagram

5.2.4 High Level Network Diagram

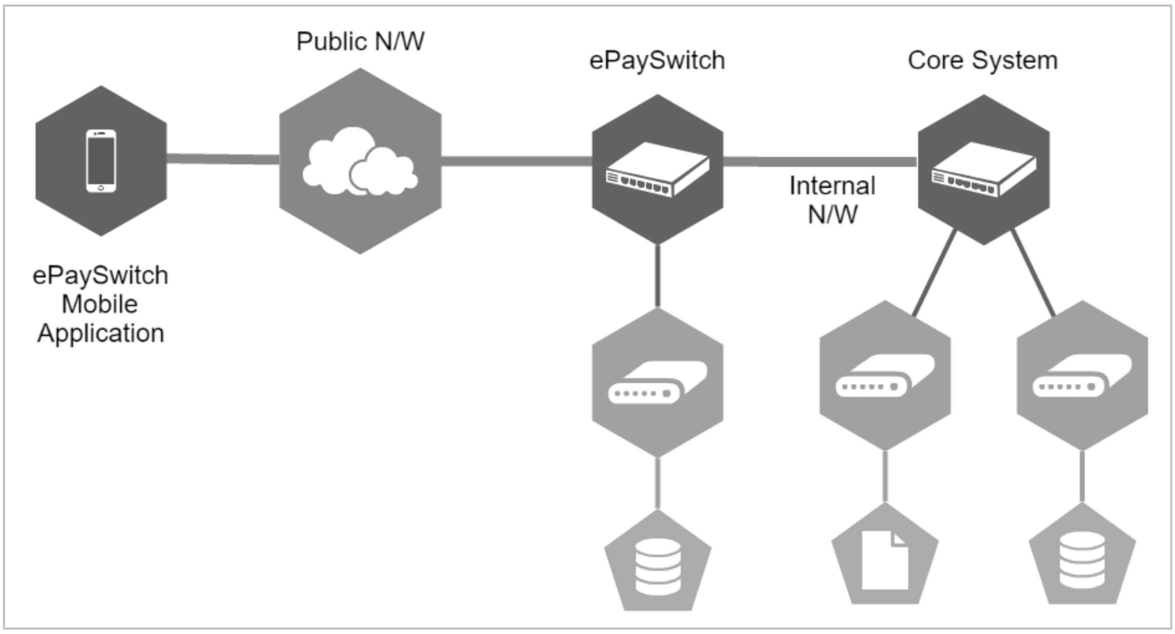


Figure 5-5: High Level Data Network Diagram

In the above Figure 5-5, it illustrates the how the network communication will take place. Mobile application will use the public network and send the information. Any information that will receive for rule validation is only via internal network.

5.3 Hierarchy of ePaySwitch

Software switch acts as the intermediator between the core financial system and the mobile application. The communication protocol between the core system and ePaySwitch is standard ISO8583.

5.3.1 Mobile application

Figure 5-6 shows the login screen and the card management screen of the proposed application. These wireframes may change according to the requirement of the institute.

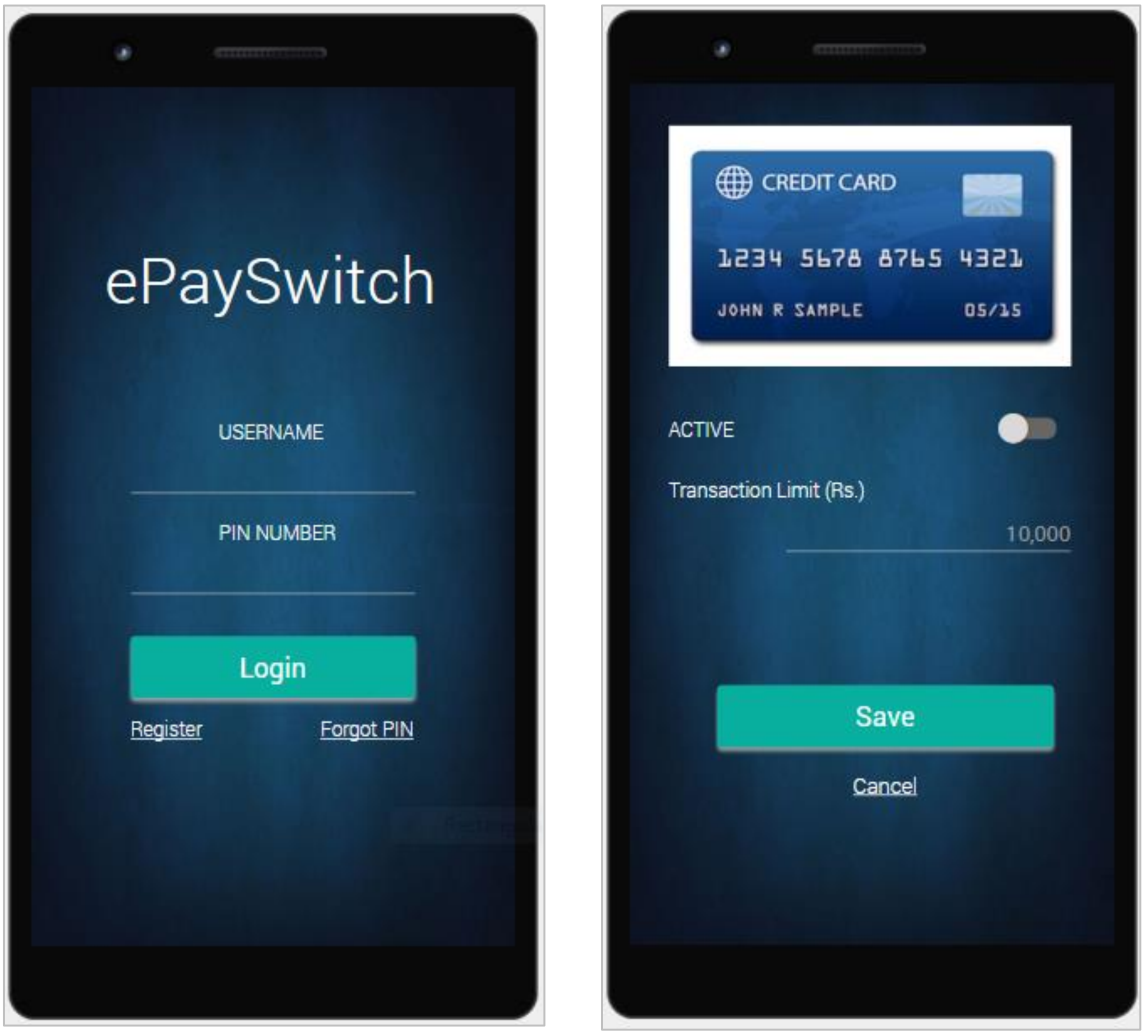


Figure 5-6: Wireframes of the Mobile Interface

5.3.1.1 User Registration and Authentication

A customer has to register with system by consuming a web service in ePaySwitch. He/She has to enter the required details in the mobile application. Those data will be saved in the ePaySwitch database as an inactive user. An officer form the bank has to decide whether to activate that particular customer. Until customer gets activated, he/she cannot login to the system.

Activated customer can login to the system by entering the username and the password. This process also consumes a web API. API will check the validity and grant or deny the access to the system accordingly.

The flow chart for the user login process is shown in Figure 5-7.

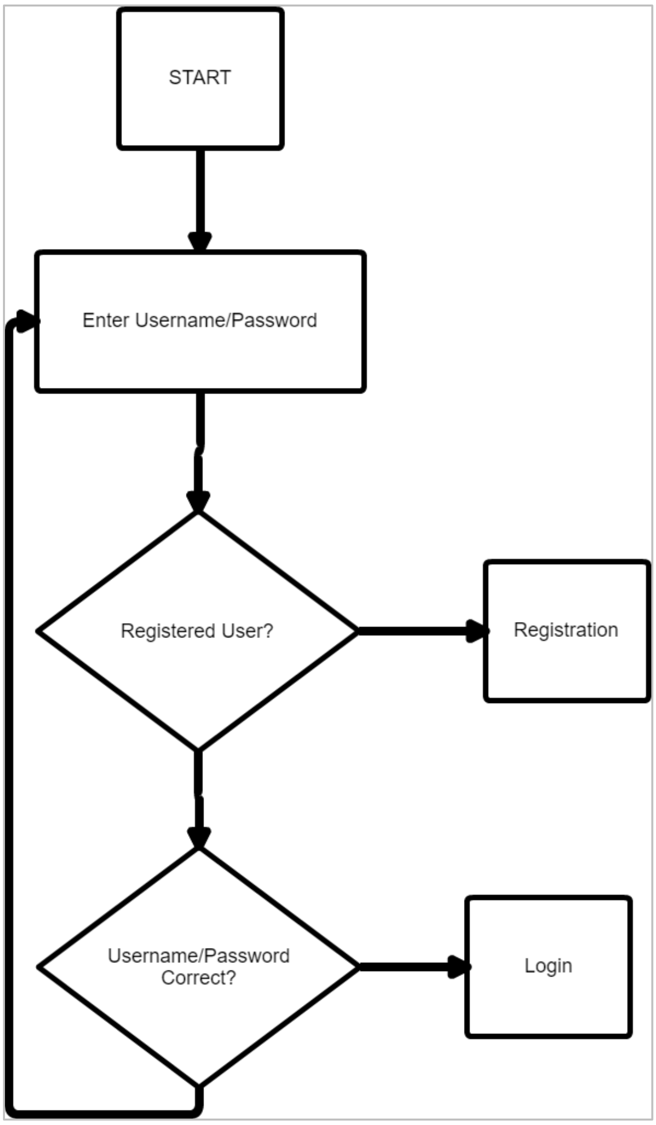


Figure 5-7: Login Flow Chart

5.3.1.2 Card Registration

A registered user can add cards to the system by contacting the card issuing institute. If the user have granted any cards, the ePaySwitch authorized user can add his/her card to the system. The default state of the card will be inactive and the transaction limit is set to zero.

Mobile application has to consume a web service in order to perform the card registration. User will be prompted to enter required details in the mobile application. This information will be sent to the server via public network and data will be saved in the ePaySwitch database. User will be notified on the status and the institute is getting a new inactive card that needs activation after a review.

Figure 5-8 shows the sequence of activities in customer and cards registration

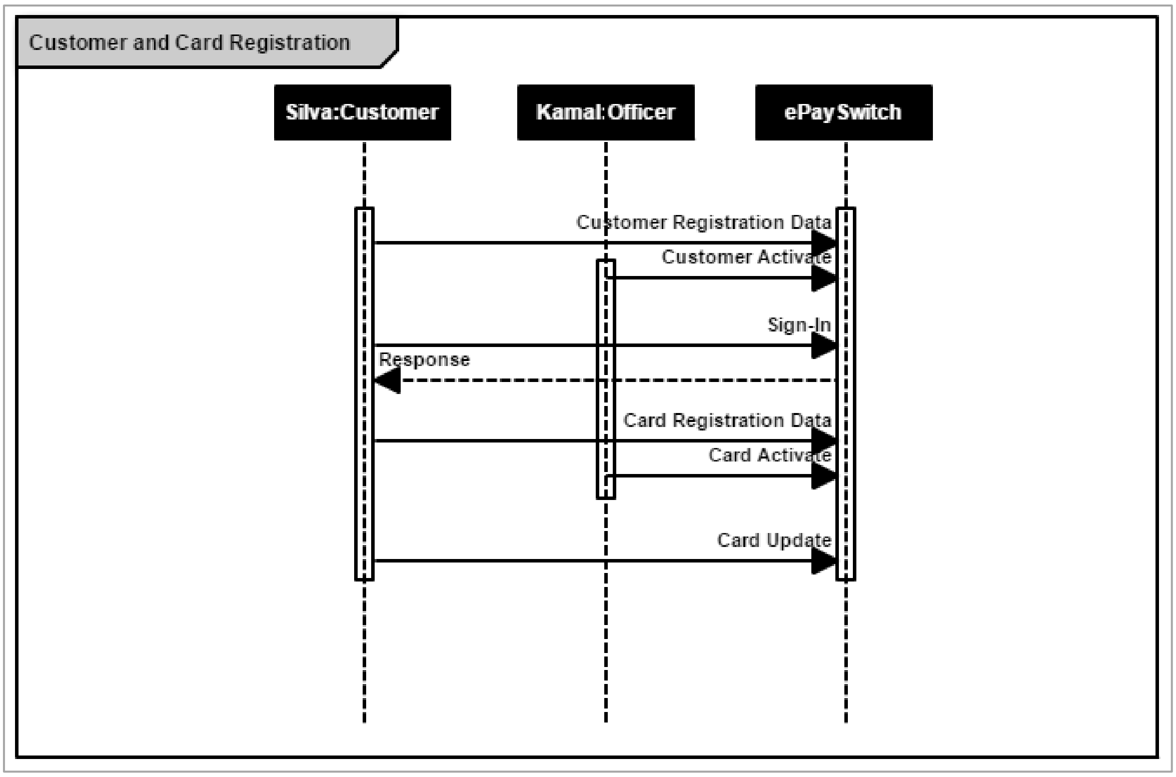


Figure 5-8: Customer and Card Registration Sequence Diagram

5.3.1.3 Card Control

After adding a card to the system particular customer can activate or set the transaction limit to the card. This will be effected when the user trying to do a transaction. If a card is turned oss the transaction will be declined. If the transaction limit is lower than the payment amount then also the transaction will be declined.

5.3.2 APIs for Mobile Access

APIs are written in java and hosted on Apache Tomcat. They will be bind to the mobile application at the development. API managers and identity servers have not used on top of those services as the ePaySwitch' approach is different. Nevertheless applying those additional security features are possible and it will give more manageability over the APIs.

5.3.3 Payments Processor

Payments processor is the most responsible modules which encodes and decodes the data which can be, POS/Internet and the core financial system. This module has written using PHP. Selection of this language is due to the competency of the development in communication handling with ISO8583 protocol.

Figure 5-9 shows a block diagram of the payments processor.

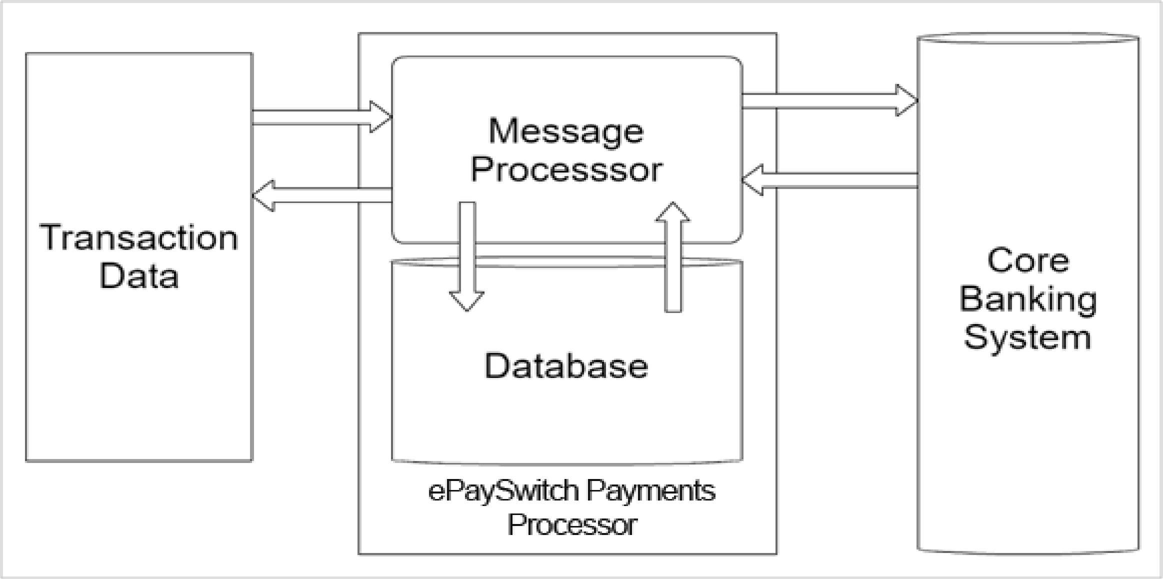


Figure 5-9: Payments Processor

5.3.4 Customers and Cards Manager

Customers and cards manager will be available to the officers in the bank to activate and deactivate customers and cards. This module needs to be attached to the process due to following two reasons.

1. When a user gets registered, then that particular user should be a customer of that particular institute.
2. When a card gets registered, then that particular card should be in the card repository of that particular institute and should also belong to the customer who added the card.

Since above verification is not automated in the ePaySwitch, manual interaction of a banking user is included.

5.3.5 Data Storage of ePaySwitch

ePaySwitch can use any commercially or freely available database in the market. The fact that is more important here is the security of the database. As the save data is sensitive and the number of transactions per day will be higher in commercial releases. It is recommended to go with a reputed, high performance database.

Figure 5-10 illustrates the high level database design for the ePaySwitch. This design is for the card based transactions only as this research touches only on the specified area.

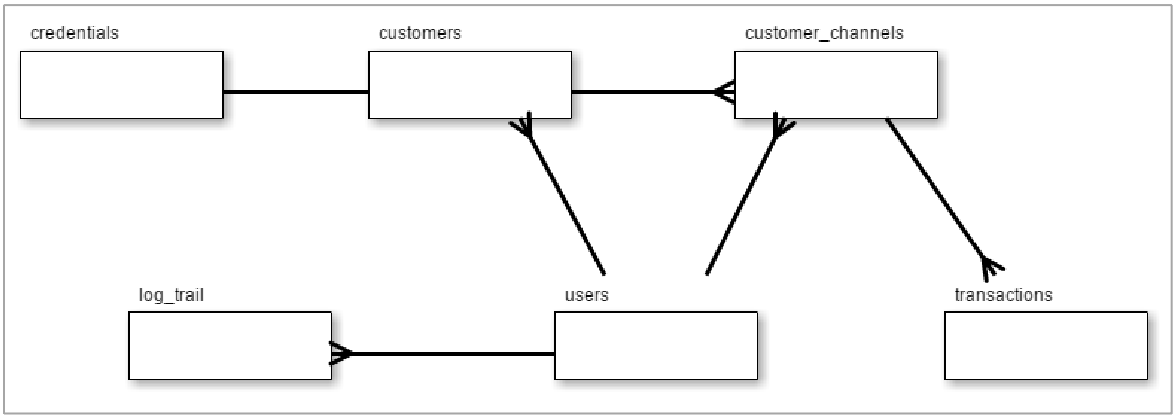


Figure 5-10: High Level Database Diagram

Customer information will be saves on the 'customers' table while the credentials will be saved separately. Card information will be saved in the 'customer_channels' table by considering the card is a payment channel. This table holds the rules for the transactions. User table stores the user information for access the Customer and Card Manager module discussed above. Table: 'log_trail' keeps the data regarding the changes done for 'customers' and 'customer_channels'. All the transactions which are processed through the payments processor will be saved in the 'transactions' table.

5.4 The Actual Transaction

When a user tries to produce his/her card for a transaction, the channel will directly connected to the merchants' payment system as shown in Figure 5-11.

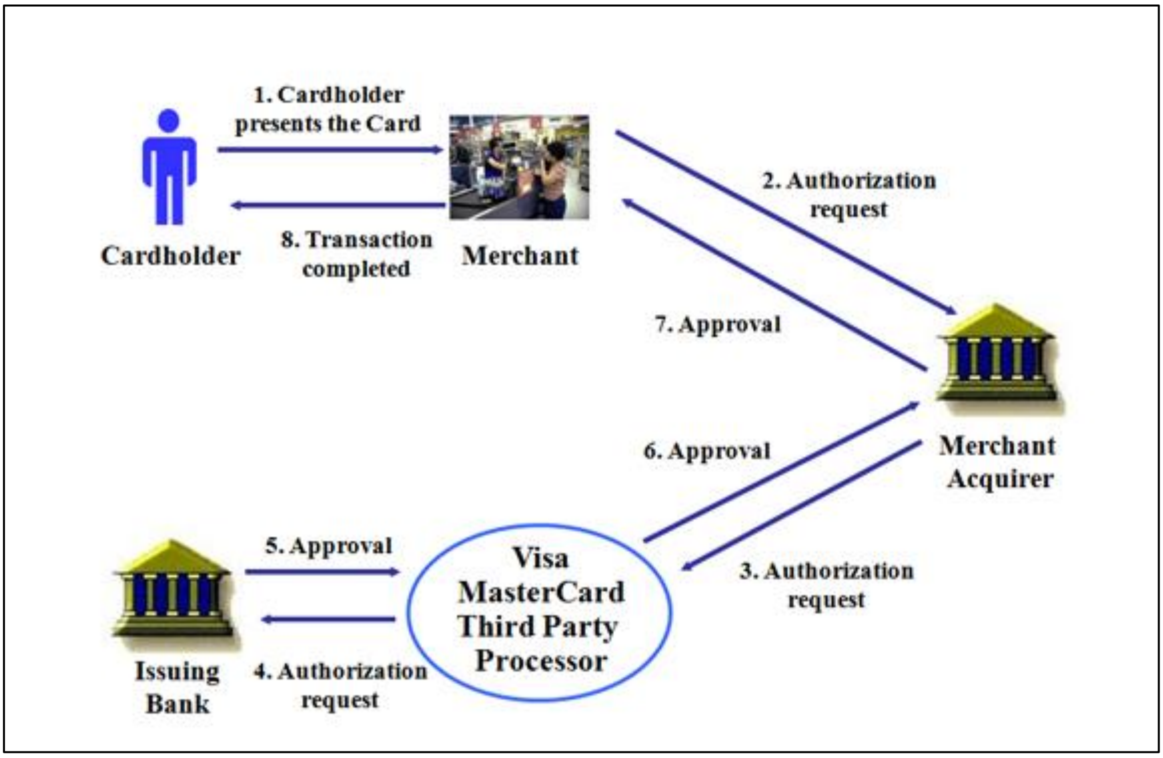


Figure 5-11: Conventional Card Transaction (Offline)

Then the message is passed to the issuing institute. The proposed solution ePaySwitch comes in to the action this point. The following Figure 5-12 and Figure 5-13 further elaborated the function of the ePaySwitch which will be described. ePaySwitch will reside in between the existing payment switch and the core banking system

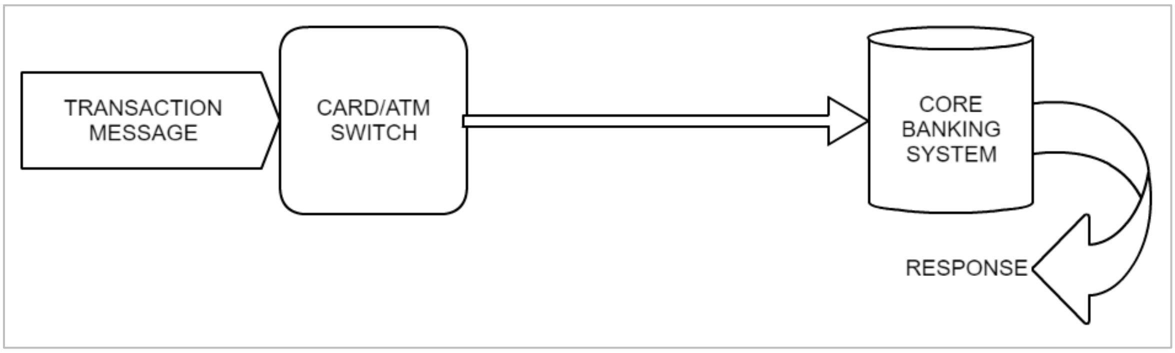


Figure 5-12: Conventional Message Flow

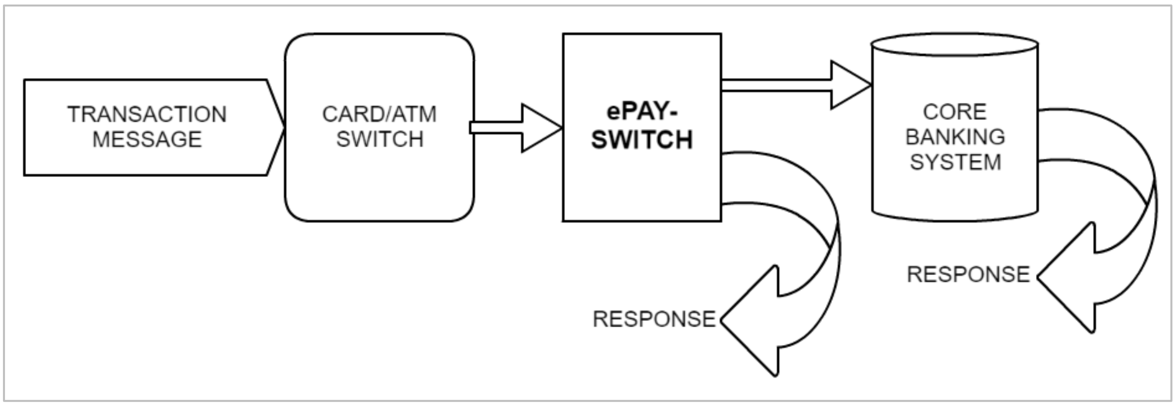


Figure 5-13: ePaySwitch Intermediator

When the details from the payment channel is acquired by the ePaySwitch;

1. First the message needs to be decoded.
2. Then the card number should be validated.
3. Status of the card should be validated.
4. After transaction amounts should be checked against the process code (Online, Offline or Withdraw)
5. There after ePaySwitch will decide whether to process the rest of the data with core system or send a declined message.
6. If the decision is to proceed, then the message will be converted to ISO 8583 and send to the core system.
7. Finally, according to the communication arrangement in core system, the response will sent back to the merchant.

5.5 Summary

This chapter discussed about the design decisions taken to implement the ePaySwitch in terms of different modules in the system. Server architecture, database design and the API design also discussed.

In next chapter we will look deep into the implementation of ePaySwitch.

6 Implementation of ePaySwitch

6.1 Introduction

Previous chapter gave full details on design of the ePaySwitch. This chapter elaborates about the implementations of each module described in above chapter namely ePaySwitch Mobile Application, APIs, Customers & Cards Manager and Payments Processor. Further the chapter has the main code segments used in the ePaySwitch.

6.2 Building Blocks of ePaySwitch

The solution here explains is for a single bank use. Whenever the application communicates with a server it belongs to one particular bank. Solution is completely developed using open source technologies as further described below.

6.2.1 Mobile Application

Mobile application is the interaction point between the server and the customer. On a commercial release this application can be downloaded via any application store where the app is released to. Because of this implementation is for a single bank, a customer who is willing to use cards from several banks has to have different applications per bank.

Therefore the mobile application is pre-configured to access one particular server and a port.

```
public static String getServerIP() {
    return "http://IP:PORT";
}
```

User registration and card registration used SHA-256 encryption standard as mentioned in the Chapter 4. Because reversing SHA-256 is actually considered malicious. SHA-256 has creation method used all over the solution is mentioned bellow.

```
private static final String Sha256(final String s) {
    final String SHA256 = "SHA-256";
    try {
        // Create SHA-256 Hash
        MessageDigest digest = java.security.MessageDigest.getInstance(SHA256);
        digest.update(s.getBytes());
        byte messageDigest[] = digest.digest();
        // Create Hex String
        StringBuilder hexString = new StringBuilder();
        for (byte aMessageDigest : messageDigest) {
            String h = Integer.toHexString(0xFF & aMessageDigest);
            while (h.length() < 2)
                h = "0" + h;
            hexString.append(h);
        }
        return hexString.toString();

    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    return "";
}
```

ePaySwitch mobile application is developed on Android platform. Android platform is free and open source. The community support is very high and the development effort is relatively less.

6.2.2 APIs for Mobile Access

ePaySwitch has three different public APIs to be consumed by the mobile application as;

1. card_pay_services_1/card_service_1
    a. AddCard [Method to add a new card]
    b. DeleteCard [delete an unconfirmed card]
    c. GetCardDetails [Get details of a particular card]
    d. GetCards [Get list of cards for the signed in user]
    e. UpdateCard [Update card details]
2. card_pay_services_1/user_service_1
    a. Register [Register new user]
    b. SignIn [User Sing In]
    c. ChangePass [Change the login password]
3. card_pay_services_1/transaction_service_1
    a. GetTransactions [Get details of transactions]

“card_service_1” is used for cards registration, update & deletion. “user_service_1” is used for user registration & update. “transaction_service_1” used for transaction details viewing. Each of these services has separate methods implemented for separate functions as mentioned above. Inputs and outputs of those services are attached in Appendix B.

Customer can turn on/off the cards, transactions and set the transaction values by consuming these services via ePaySwitch Mobile application.

These services have been written in Java as web services. Implementation platform is Apache Tomcat.

6.2.3 Payments Processor

Payments processor is the heart of ePaySwitch. Because, it is responsible for take the message that receives from the payment channel and decode it to cross check with the validation rules applied by the customer. If the rules do not match, then particular transaction originator should be notified. Otherwise the message should pass to the core banking system to further process and get the response and finally pass to the transaction originator.

6.2.4 Cards & Customers Manager

Cards and user manager is a sub system of the ePaySwitch, which is used to activate/deactivate the customers and cards. Access for this portal will be given to privileged personnel of the particular institute or department.

Module implemented on the Apacher web servier using PHP.

6.2.5 ePaySwitch Database

Database implementation of the ePaySwitch has done using MySql GPL version. Tables are accessible via stored procedures by the APIs. All the CRUD operations are handled using stored procedures.

Figure 6-1 illustrates the table implementation of the ePaySwitch database.

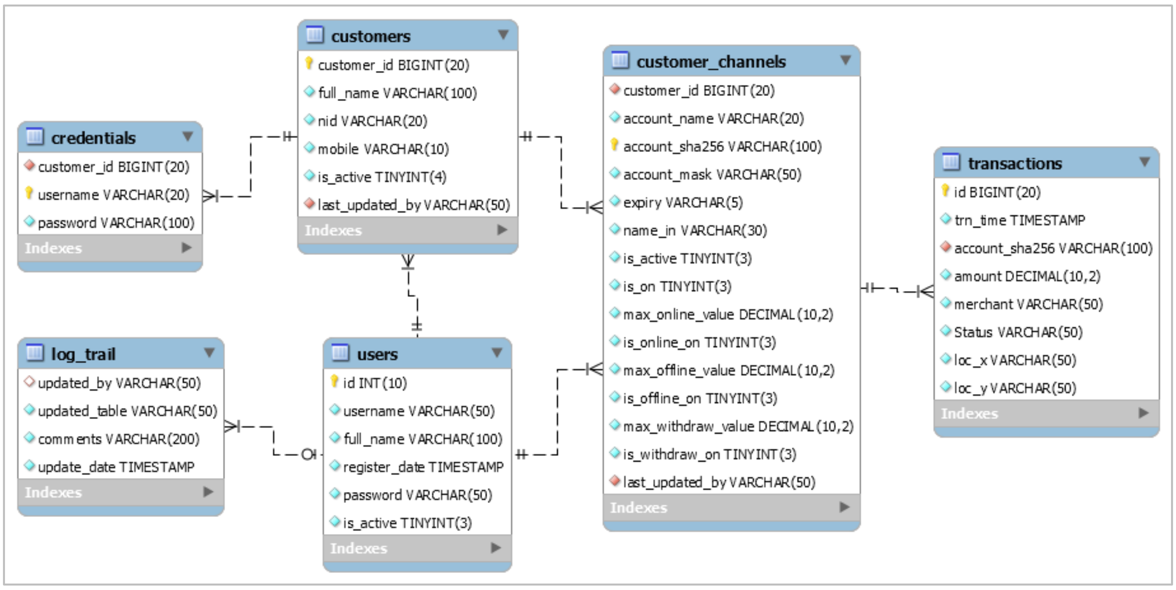


Figure 6-1: Database Table Implementations of ePaySwitch

Stored procedures for the API access are as tabulated in below Table 6-1.

Procedure Name	**Description**
sp_register	Customer Registration
sp_user_login	Customer Sign-In
sp_addcard	Register New Card
sp_updatecard	Update Card Rules
sp_deletecard	Delete Card
sp_get_cards	Get Active List of Cards
sp_cardview	View a Particular Card

Table 6-1: Database Stored Procedures

6.3 Implementation of ePaySwitch

6.3.1 User Registration

In order to access the ePaySwitch services particular user needs to be registered with the system. After user fills all the required fields as per the Image shown, customer may click on register button. This will consume ‘Register’ method in ‘user_service_1’ service.

User registration form in mobile application is shown in Figure 6-2.

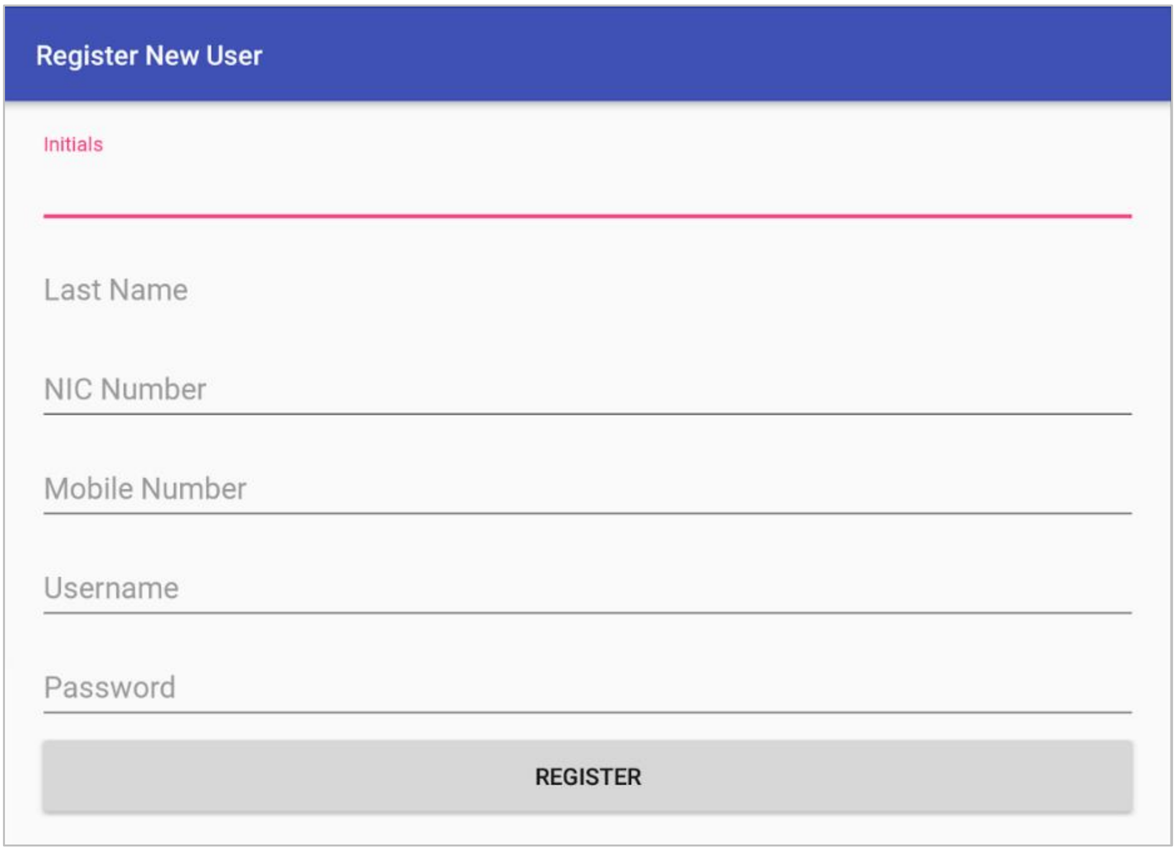
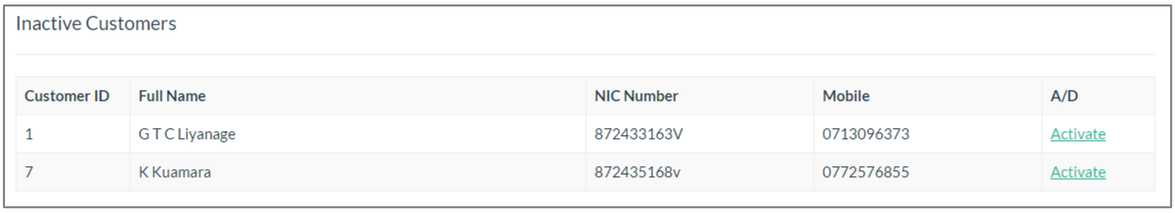


Figure 6-2: User Registration Form

All the information will be passed to the server via the API in plain text. Password is encrypted using SAH256. When customer clicks the 'Register' button, application encrypts the password and sends to the server together with other details.

If customer entered invalid details, it will be informed to the customer. Images of these user interfaces are attached in the Appendix C.

After a successful user registration, an officer from the bank needs to verify and confirm the details the customer has entered using the 'Cards & User Manager'. Figure 6-3 shows the customer activation portal.



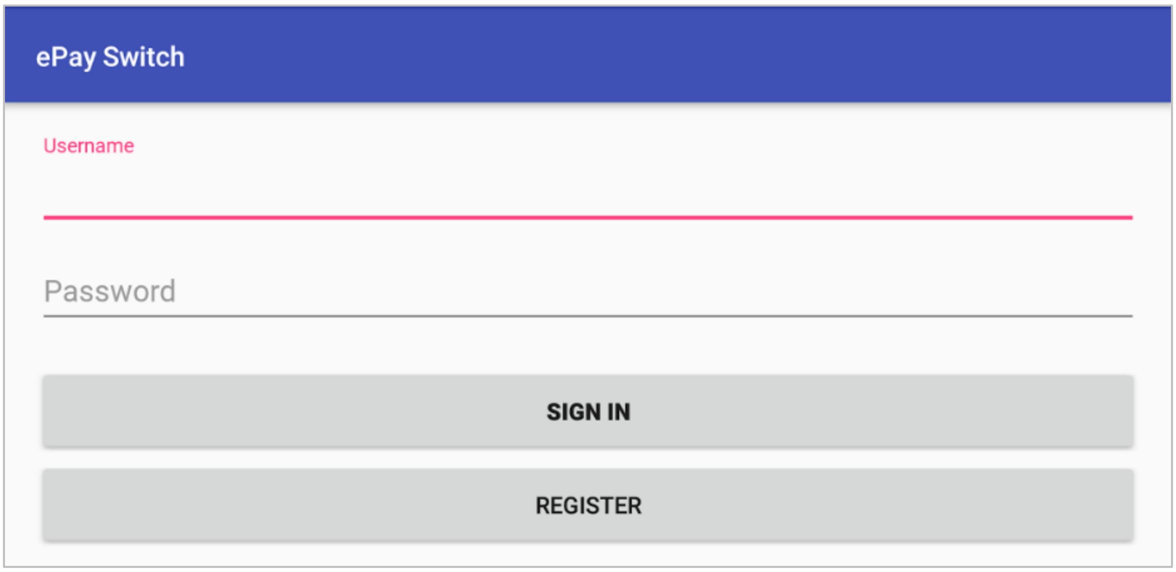
Inactive Customers

Customer ID	Full Name	NIC Number	Mobile	A/D
1	G T C Liyanage	872433163V	0713096373	Activate
7	K Kuamara	872435168v	0772576855	Activate

Figure 6-3: Customer Activation Portal

After activating the particular customer he/she will be able to login to the system using the username and the password given by him/her at the point of registration. Password will not be visible to anyone after registration as they will be stored encrypted.

6.3.2 Sign In



ePay Switch

Username

Password

Figure 6-4: User Sign-In Form

After activating the customer he/she may log in to the system using the sign-in form as shown in Figure 6-4. Username and password is same as provided at the registration. For the sign in process, application will consume ‘SignIn’ method in ‘user_service_1’.Username and password will pass to the server in plain text with encrypted password cipher. If the entered credentials are correct and the user is active then user will be signed in to the system. Otherwise user will be informed on the status..

6.3.3 Card Registration

After a successful sign in, customer will be able to add a card to his vault. This can be using the + sign on the home screen. User should enter the requested details according to the format in the Figure 6-5.

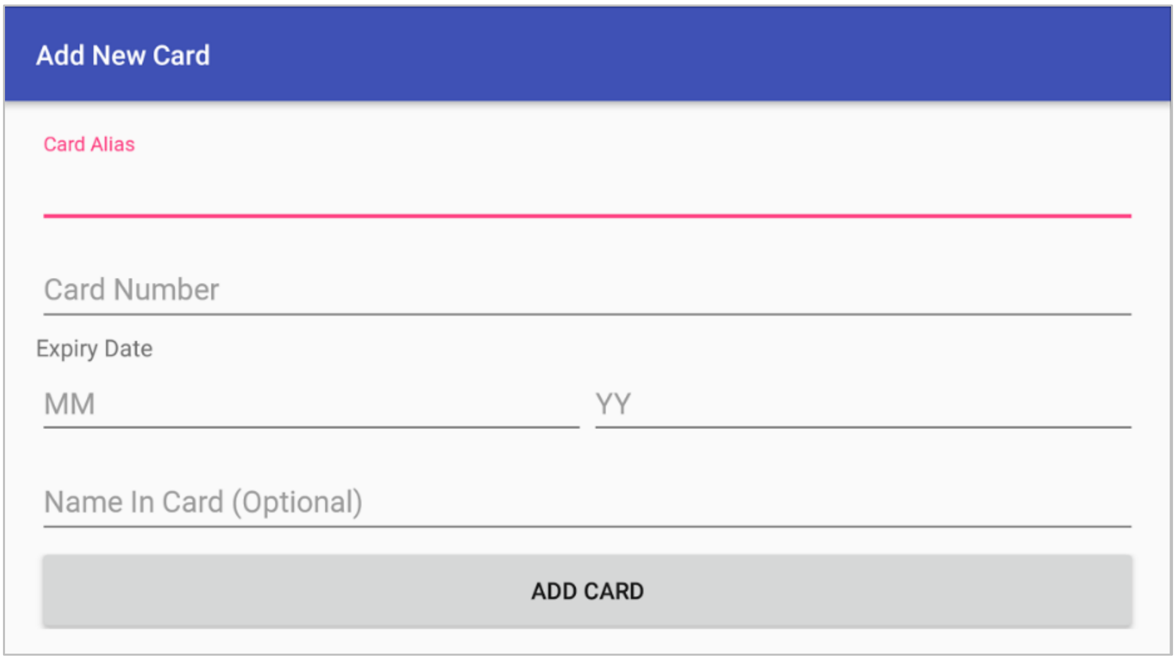


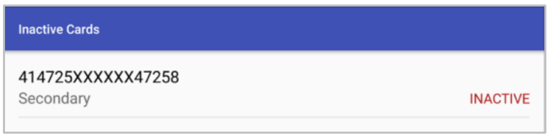
Figure 6-5: card Registration Form

Card registration uses ‘AddCard’ method in ‘card_service_1’ service. API will access SQL query of ‘sp_addcard’, service class and the cutomer channel class implementation for particular service.

After inserting the card details, authorized person from the institute should activate the particular card. Otherwise customer cannot access any of the functionalites ragards to the particular card. Only available option customer will have is to remove the registered card before activating.

6.3.3.1 Card Remove

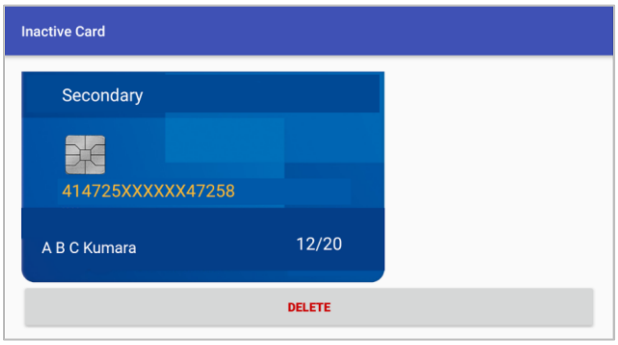
Customer can access the registered inactive cards by selecting ‘Inactive Cards’ section from the menu drawer as in Figure 6-6 and Figure 6-7. If the particular card is activated then he/she cannot delete the card.



Inactive Cards

414725XXXXXX47258
Secondary
INACTIVE

Figure 6-6: Inactive Cards List



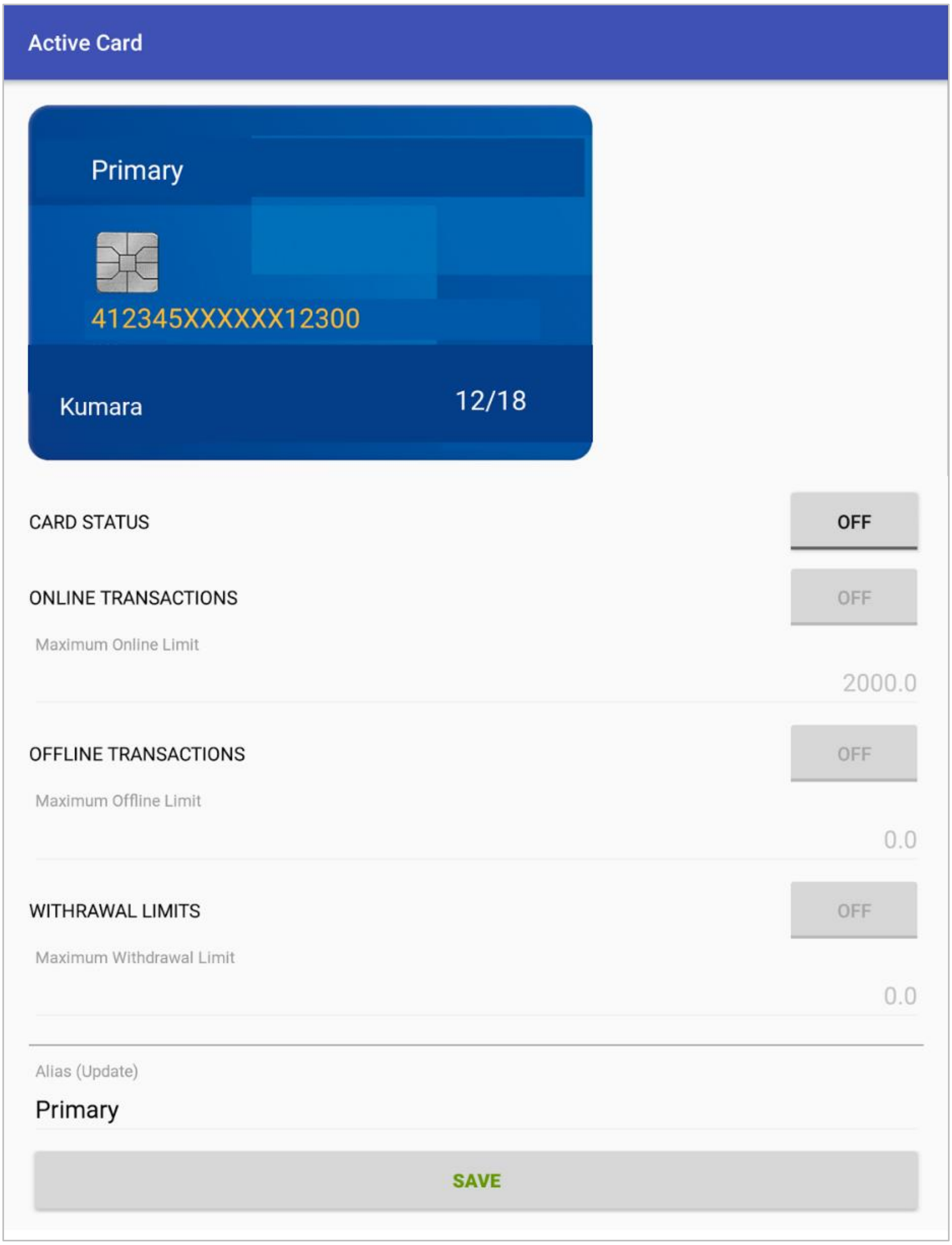
Inactive Card

Secondary
414725XXXXXX47258
A B C Kumara
12/20

Figure 6-7: Card Remove Form

6.3.4 Adding Rules and Card Controls (Update Card)

In the home page of the ePaySwitch Mobile application, customer can see all his/her activated cards. To apply the rules to a particular card customer has to tap on the particular card number. It will open the following page that can be used to apply all the rules for online, offline and withdrawal transactions. This one of the important implementation is ePaySwitch. The control of the electronic transactions are done the customer himself and the security of the card also can be controlled using this page. Figure 6-8 shows the card control form.



Active Card

Primary

412345XXXXXX12300

Kumara 12/18

CARD STATUS OFF

ONLINE TRANSACTIONS OFF

Maximum Online Limit 2000.0

OFFLINE TRANSACTIONS OFF

Maximum Offline Limit 0.0

WITHRAWAL LIMITS OFF

Maximum Withdrawal Limit 0.0

Alias (Update) Primary

Figure 6-8: Card On/Off and Adding Rules

Following are the implementation of the card control function.

<u>SQL Stored Procedure</u>

```
BEGIN
```

```
DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
ROLLBACK;
       SELECT "0x9100" AS resp;
END;
START TRANSACTION;
UPDATE customer_channels SET
       is_on = _on,
       max_online_value =_max_online_value,
       is_online_on = _online_on,
       max_offline_value =_max_offline_value,
       is_offline_on = _offline_on,
       max_withdraw_value =_max_withdraw_value,
       is_withdraw_on = _withdraw_on,
       account_name = _card_name
WHERE account_Sha256 = _card_Sha256 AND customer_id = _customer_id;
COMMIT;
       SELECT "0x9000" AS resp;
END
```

Service Implementation (card_service_1.java)

```
    @WebMethod(operationName = "UpdateCard")
    public String UpdateCard(
            @WebParam(name = "CustomerID") Long CustomerID,
            @WebParam(name = "CardName") String CardName,
            @WebParam(name = "CardSha256") String CardSha256,
            @WebParam(name = "IsOn") String IsOn,
            @WebParam(name = "MaxOnlineValue") String MaxOnlineValue,
            @WebParam(name = "IsOnlineOn") String IsOnlineOn,
            @WebParam(name = "MaxOfflineValue") String MaxOfflineValue,
            @WebParam(name = "IsOfflineOn") String IsOfflineOn,
            @WebParam(name = "MaxWithdrawValue") String MaxWithdrawValue,
            @WebParam(name = "IsWithdrawOn") String IsWithdrawOn) {
        CustomerChannels ccUpdateCard = new CustomerChannels();
        ccUpdateCard.setCustomerId(CustomerID);
        ccUpdateCard.setAccountName(CardName);
        ccUpdateCard.setAccountSha256(CardSha256);
        ccUpdateCard.setIsOn(Short.parseShort(IsOn));
        ccUpdateCard.setMaxOnlineValue(Float.parseFloat(MaxOnlineValue));
        ccUpdateCard.setIsOnlineOn(Short.parseShort(IsOnlineOn));
        ccUpdateCard.setMaxOfflineValue(Float.parseFloat(MaxOfflineValue));
        ccUpdateCard.setIsOfflineOn(Short.parseShort(IsOfflineOn));
        ccUpdateCard.setMaxWithdrawValue(Float.parseFloat(MaxWithdrawValue));
        ccUpdateCard.setIsWithdrawOn(Short.parseShort(IsWithdrawOn));
        return ccUpdateCard.UpdateCard(ccUpdateCard);
    }
```

Service Implementation (CustomerChannels.java)

```
    public String UpdateCard(CustomerChannels ccUpdateCard) {
        dbAccess db = new dbAccess();
        Connection con = db.GetConnection();
        String response = "";
        try {
            Statement stmnt = con.createStatement();
            ResultSet result;
            result = stmnt.executeQuery("CALL sp_updatecard("
                    + "'" + ccUpdateCard.getAccountSha256() + "', "
                    + "'" + ccUpdateCard.getCustomerId() + "', "
                    + "'" + ccUpdateCard.getAccountName() + "', "
                    + "'" + String.valueOf(ccUpdateCard.getIsOn()) + "', "
                    + "'" + ccUpdateCard.getMaxOnlineValue() + "', "
                    + "'" + String.valueOf(ccUpdateCard.getIsOnlineOn()) + "', "
                    + "'" + ccUpdateCard.getMaxOfflineValue()+ "', "
                    + "'" + String.valueOf(ccUpdateCard.getIsOfflineOn()) + "',
"
                    + "'" + ccUpdateCard.getMaxWithdrawValue()+ "', "
                    + "'" + String.valueOf(ccUpdateCard.getIsWithdrawOn()) + "'
"
                    + ")");
            while (result.next()) {
                response = result.getString("resp");
            }
        } catch (SQLException ex) {
            Logger.getLogger(card_service_1.class.getName()).log(Level.SEVERE,
null, ex);
        }
        return response;
    }
```

Mobile App Implementation

```
   public class UpdateCard extends AsyncTask<Void, Void, Boolean> {

        private CustomerChannel ccCard = new CustomerChannel();

        UpdateCard(String CardAlias, String CardNumber_sha256, Boolean IsOn,
Double MaxOnlineValue, Boolean IsOnlineOn,
                   Double MaxOfflineValue, Boolean IsOfflineOn, Double
MaxWithdrawValue, Boolean IsWithdrawOn, Long UserId) {
            ccCard.setCustomerId(UserId);
            ccCard.setAccountName(CardAlias);
            ccCard.setAccountSha256(CardNumber_sha256);
            if (IsOn)
                ccCard.setIsOn((short) 1);
            else
                ccCard.setIsOn((short) 0);
            ccCard.setMaxOnlineValue(MaxOnlineValue);
            if (IsOnlineOn)
                ccCard.setIsOnlineOn((short) 1);
            else
                ccCard.setIsOnlineOn((short) 0);
            ccCard.setMaxOfflineValue(MaxOfflineValue);
            if (IsOfflineOn)
                ccCard.setIsOfflineOn((short) 1);
            else
                ccCard.setIsOfflineOn((short) 0);
            ccCard.setMaxWithdrawValue(MaxWithdrawValue);
            if (IsWithdrawOn)
                ccCard.setIsWithdrawOn((short) 1);
            else
                ccCard.setIsWithdrawOn((short) 0);
        }

        @Override
        protected Boolean doInBackground(Void... params) {
            final String SOAP_ACTION = "http://services.epay.com/UpdateCard";
            final String METHOD_NAME = "UpdateCard";
            final String NAMESPACE = "http://services.epay.com/";
            final String URL = Actions.getServerIP() +
"/card_pay_services_1/card_service_1?wsdl";

            SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
            request.addProperty("CustomerID", ccCard.getCustomerId());
            request.addProperty("CardName", ccCard.getAccountName());
            request.addProperty("CardSha256", ccCard.getAccountSha256());
            request.addProperty("IsOn", String.valueOf(ccCard.getIsOn()));
            request.addProperty("MaxOnlineValue",
ccCard.getMaxOnlineValue().toString());
            request.addProperty("IsOnlineOn",
String.valueOf(ccCard.getIsOnlineOn()));
            request.addProperty("MaxOfflineValue",
ccCard.getMaxOfflineValue().toString());
            request.addProperty("IsOfflineOn",
String.valueOf(ccCard.getIsOfflineOn()));
            request.addProperty("MaxWithdrawValue",
ccCard.getMaxWithdrawValue().toString());
            request.addProperty("IsWithdrawOn",
String.valueOf(ccCard.getIsWithdrawOn()));
            SoapSerializationEnvelope envelope = new
SoapSerializationEnvelope(SoapEnvelope.VER11);
            envelope.setOutputSoapObject(request);
            HttpTransportSE ht = new HttpTransportSE(URL);
            try {
                ht.call(SOAP_ACTION, envelope);
                SoapPrimitive anyType1 = (SoapPrimitive) envelope.getResponse();
                response = anyType1.toString();
                return true;

            } catch (Exception e) {
                return false;
            }
        }
```

This rules or card controls are the inputs to ‘Payments Processor’ which will be discussed next.

6.3.5 Payments Processor

Payments processor is the heart of the solution which decides continuation of a transaction based on the rules set applied by the customer. Also this is the switch which decodes the bit stream from the payment channel and routes accordingly to core system or back to the payment initiator. Flow chart for the payment processor is shown in Figure 6-9.

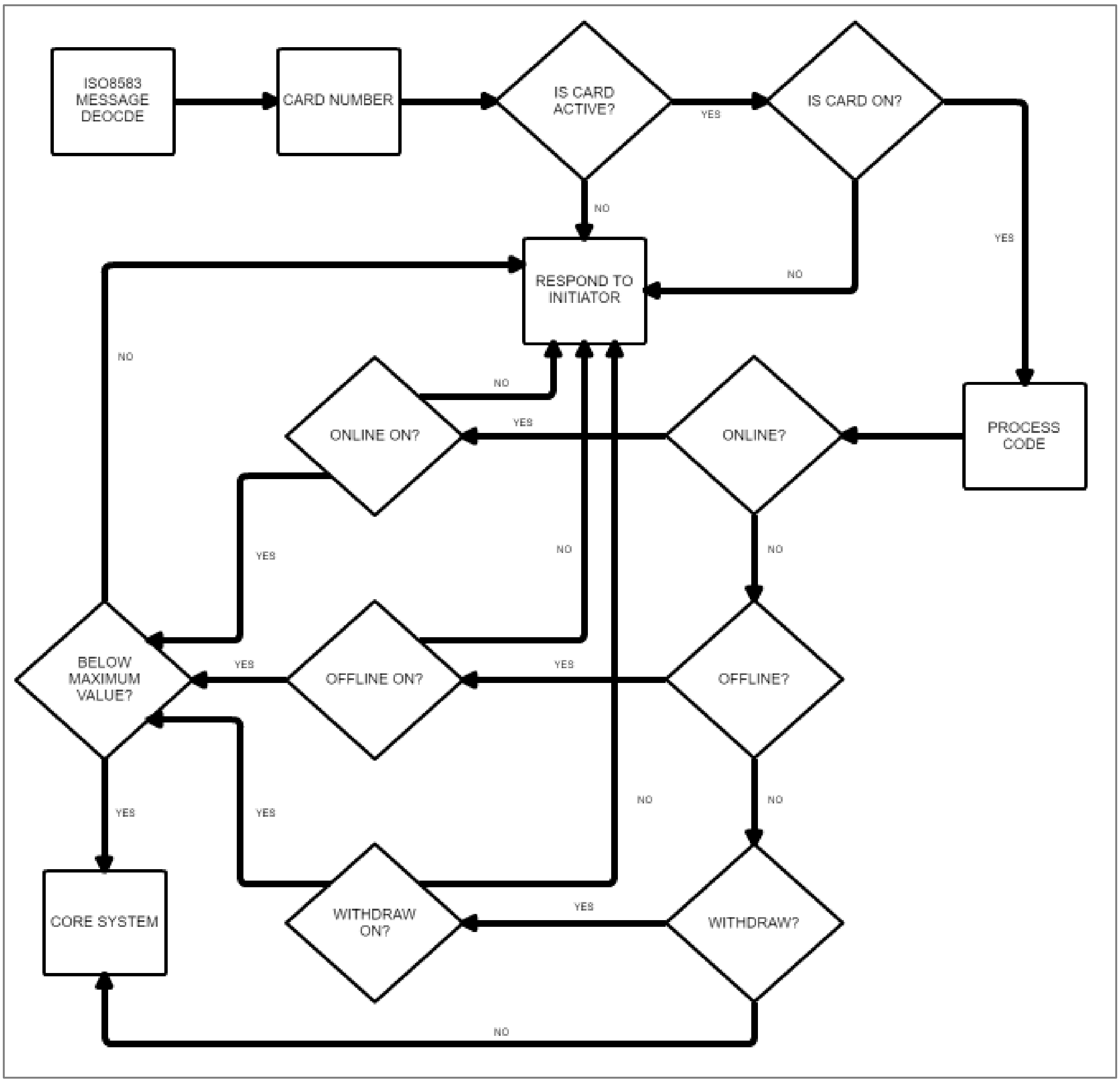


Figure 6-9: Payment Processor Flow

Detailed implementation details of above flow diagram is discussed later in this chapter.

Payment processor has been implemented in PHP in a separate apache server. This was to keep mobile application channel and the payment channel separately. Code segment used for card control is given below.

```
$sha256 = substr($payDetails_, 2, substr($payDetails_, 0, 2));
$PayDetails = substr($payDetails_, substr($payDetails_, 0, 2)+2);

$amount_s = substr($PayDetails, 60, 12);
$amount = floatval((substr($amount_s, 0, 10).".".substr($amount_s, 10, 2)));

if($sha256 == hash("sha256", $PayDetails)){

        $cardNumber = substr($PayDetails, (strlen($PayDetails)-16), 16);
        $cardNumber_sha256 = hash("sha256", $cardNumber);
        $merchant = substr($PayDetails, 188, 36);
        $processCode = substr($PayDetails, 54, 6);
        $Cards = CustomerChannel::FindCustomerChannelById($cardNumber_sha256);
        $cardNumAvailable= 0;
        $cardIsActive = 0;
        $cardIsOn = 0;
        $onlineOn = 0;
        $onlineValue = 0;
        $offlineOn = 0;
        $offlineValue = 0;
        $widtrawOn = 0;
        $widtrawValue = 0;

        $cardNumAvailable = $Cards->account_sha256;
        $cardIsActive = $Cards->is_active;
        $cardIsOn = $Cards->is_on;
        $onlineOn = $Cards->is_online_on;
        $onlineValue = $Cards->max_online_value;
        $offlineOn = $Cards->is_offline_on;
        $offlineValue = $Cards->max_offline_value;
        $widtrawOn = $Cards->is_withdraw_on;
        $widtrawValue = $Cards->max_withdraw_value;

        $status = "";
        $response = "";
        $isSave = false;

        if($cardNumAvailable == $cardNumber_sha256){ // CARD AVAILABLE
                if($cardIsActive == 1){ // CARD ACTIVE
                        if($cardIsOn == 1){ // CARD ON
                                if($processCode == 200000 || $processCode == 300000 ||
$processCode == 400000){
                                        if($processCode == 200000){
                                                if($onlineOn == 1){ //ONLINE ENABALED
                                                        if($onlineValue >= $amount){ //
BELOW LIMIT
                                                                $status = "ONLINE SUCCESS";
                                                                $response = "00";
                                                                $isSave = true;
                                                        } else { // ABOVE LIMIT
                                                                $status = "ONLINE LIMIT";
                                                                $response = "51";
                                                                $isSave = true;
                                                        }
                                                } else { //ONLINE OFF
                                                        $status = "ONLINE OFF";
                                                        $response = "07";
                                                        $isSave = true;
                                                }
                                        }
                                        if($processCode == 300000){
                                                if($offlineOn == 1){ //OFFLINE ENABALED
                                                        if($offlineValue >= $amount){ //
BELOW LIMIT
                                                                $status = "OFFLINE
SUCCESS";
                                                                $response = "00";
                                                                $isSave = true;
                                                        } else { // ABOVE LIMIT
                                                                $status = "OFFLINE LIMIT";
                                                                $response = "51";
                                                                $isSave = true;
                                                        }
                                                } else { // OFFLINE OFF
                                                        $status = "OFFLINE OFF";
                                                        $response = "07";
                                                        $isSave = true;
                                                }
                                        }
                                        if($widtrawOn == 400000){
                                                if($widtrawOn == 1){ //WITHDRAW ENABALED
                                                        if($onlineValue >= $amount){ //
BELOW LIMIT
                                                                $status = "WITHDRAW
SUCCESS";
                                                                $response = "00";
                                                                $isSave = true;
                                                        } else { // ABOVE LIMIT
                                                                $status = "WITHDRAW LIMIT";
                                                                $response = "51";
                                                                $isSave = true;
                                                        }
                                                } else { // WITHDRAW OFF
                                                        $status = "WITHDRAW OFF";
                                                        $response = "07";
                                                        $isSave = true;
                                                }
                                        }
                                } else { // NON OF THE CONFIGURED TRANSACTION TYPES
                                        $response = "00";
                                }
                        } else { // CARD OFF
                                $status = "CARD OFF";
                                $response = "07";
                                $isSave = true;
                        }
                } else { // CARD NOT ACTIVE
                        $status = "CARD INACTIVE";
                        $response = "07";
                        $isSave = true;
                }
        } else { // CARD NOT AVAILABLE
                $response = "00";
        }
} else { // SUSPICIOUS
        $response = "59";
        $isSave = true;
}
```

Parameter for hash() used in comparison is the ISO8583 bit stream received from the payment channel. This will be crosschecked for hash values in order to confirm that no data is changed by a man-in-middle attack. If the hash values do not match then, the payment processor will return suspicious response code back to the message originator.

Otherwise it would breakdown the ISO8583 bit stream and extracts the card number (field number differs from one implement to another), amount [field 004] and the processing code [field 003]. Processing code is used to identify the type of transaction. This value differs on the implementation. However have a 6 digits fixed length numeric value. This value is used to identify the transaction type as online payment, offline payment or cash withdrawal.

Next four sections describe how the implementation of payments processor responds to different rules applied by the customer and how they can be used to prevent frauds and controlling card based transactions in a useful manner.

6.3.6 Card Availability and Active/Inactive State

When a transaction is processed online, at some point that particular transaction receives at the issuing bank for validation purposes. When this message arrives to the switch of that particular bank without further processing with their core banking system, it will be routed to ePaySwitch. ePaySwitch Payments Processor will decode the message and extract the data for further processing.

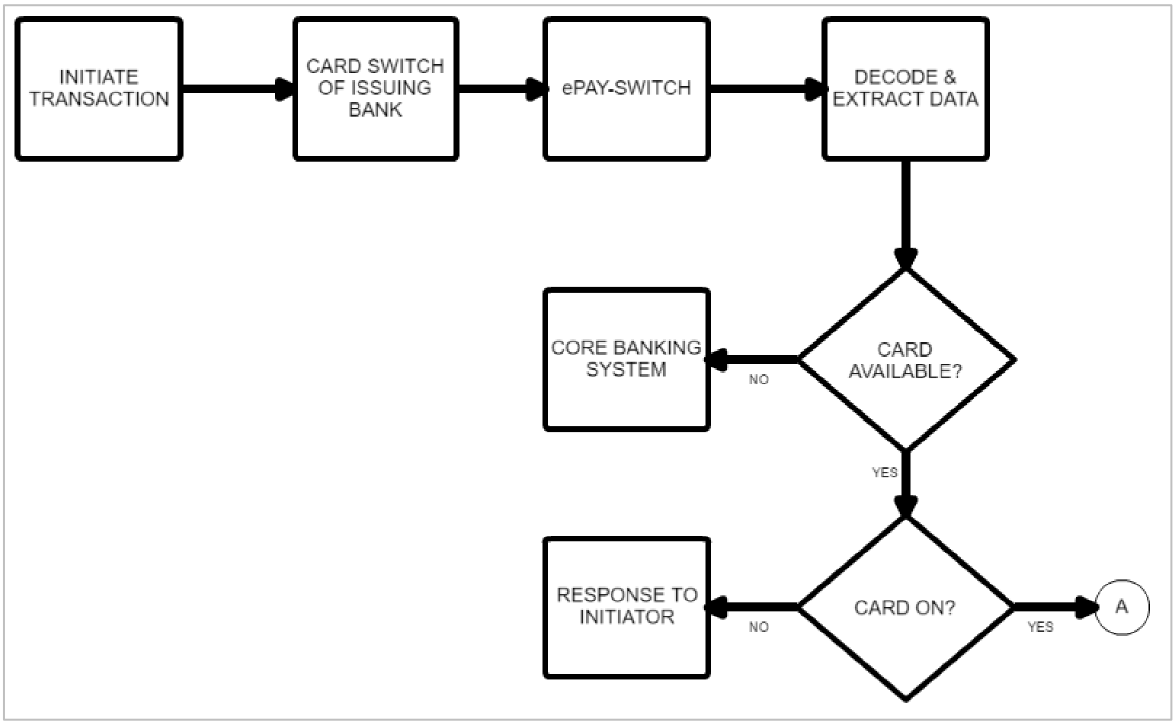


Figure 6-10 illustrates the implementation of card availability and active/inactive status and how the response takes place. The box in #A represents rest of the activity that will be linked with next four sections.

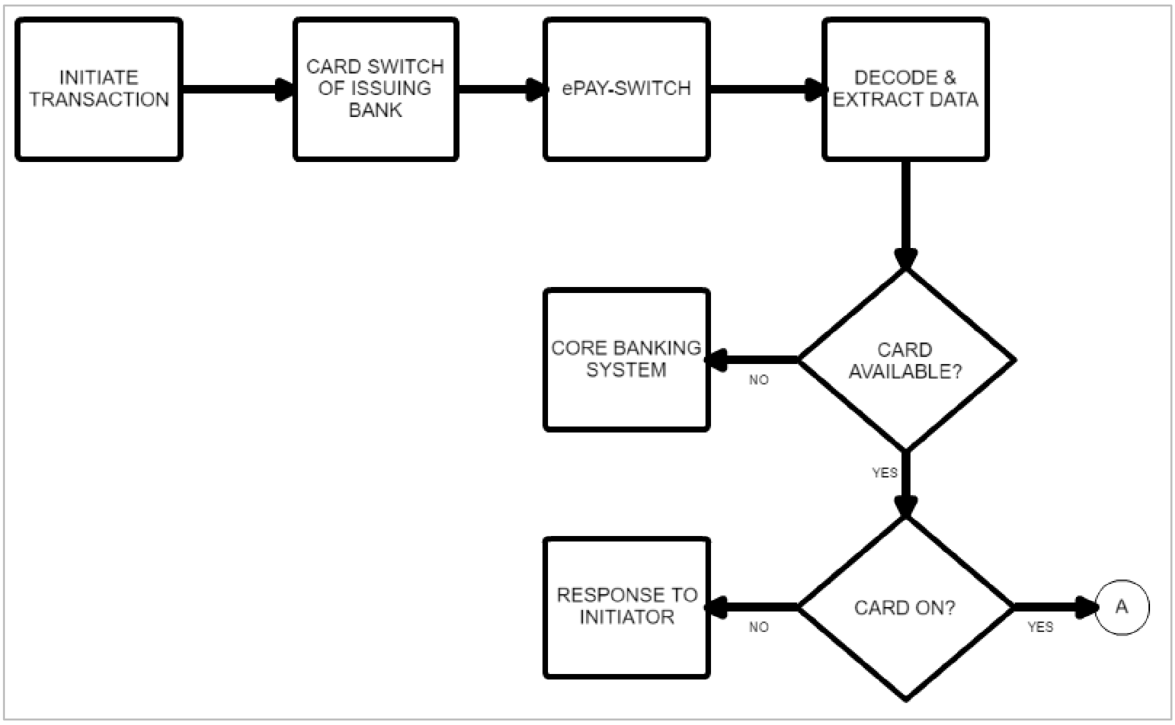


Figure 6-10: Card Availability and Active/Inactive State

First the card number will be validated against the values stored in ePaySwitch database and if the card is not available (not registered) then the message will be directly passed to the core banking system for further processing.

```
$cardNumber_sha256 = hash("sha256", $cardNumber);
…
$Cards = CustomerChannel::FindCustomerChannelById($cardNumber_sha256);
$cardNumAvailable= 0;
$cardIsActive = 0;
…
$cardNumAvailable = $Cards->account_sha256
$cardIsActive = $Cards->is_active;
…
if($cardNumAvailable == $cardNumber_sha256){ // CHECK CARD AVAILABLE
       // CARD AVAILABLE
       if($cardIsActive == 1){ // CARD ACTIVE
              … // CARD ACTIVE. CONTINUE WITH REST OF THE PROCESS
       }
} else {
       // CARD NOT AVAILABLE. FORWORD TO CORE BANKING SYSTEM
}
```

If the card is available, then the Payments Processor will continue to check the validity of card active/inactive status.

If the particular card is activated then payments processor will process rest of the validations as described in next sections. Otherwise it will respond to the initiator with a response code. Depending on the availability and active/inactive status of the card, responses are shown in Table 6-2.

Status	**Response Code**
Card Not Available	Response Code from Core Bank
Card Available	Proceed to next. Response from #A
Card Inactive	Response Code 07
Card Active	Proceed to next. Response from #A

Table 6-2: Card Availability and Active/Inactive State Response Codes

Card active/inactive status will be recorded in the ePaySwitch database for further references.

6.3.7 Card On/Off

Turning off the card will totally shut down the usage of that particular card. This will not effect in the core banking functions related to the cards as those rules are applied in a separate environment.

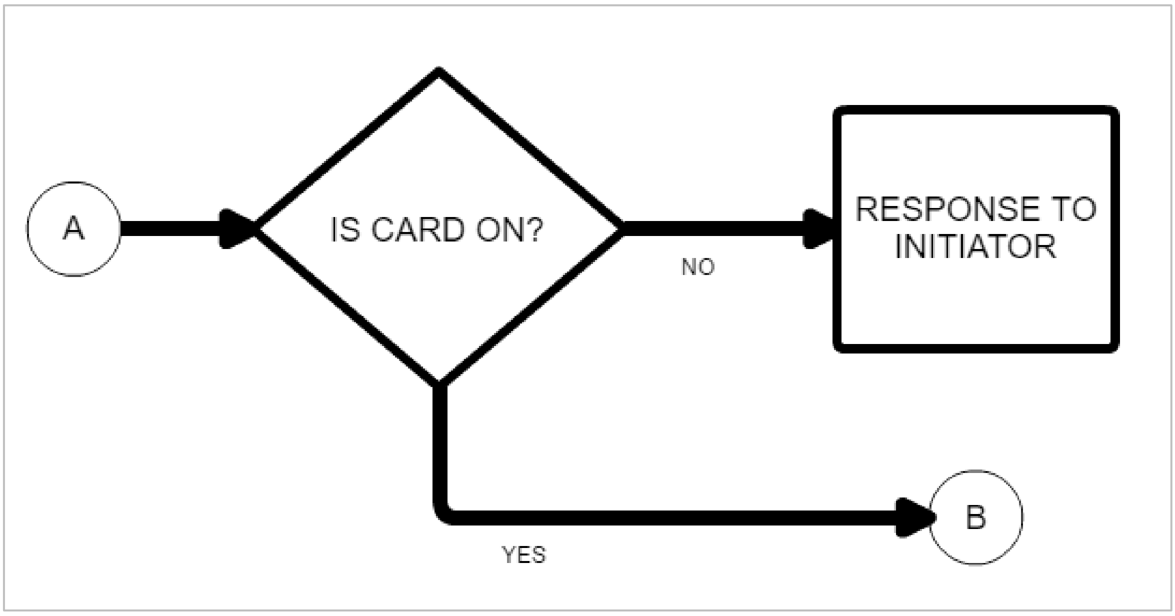


Figure 6-11: card On/Off Flow

If the card is turned off then then a response code will be sent back to the transaction originator. Otherwise will wait for the response after #B. card turn on off can be done by the customer himself. For this purpose customer can use the mobile application which will use ‘UpdateCard’ method in ‘card_service_1’ service. Payments processor code for the above is as follows.

```
$onlineOn = 0;
$onlineValue = 0;
…
$onlineOn = $Cards->is_online_on;
$onlineValue = $Cards->max_online_value;
…
if($processCode == 2000000){ // ONLINE TRANSACTION TYPE
        if($onlineOn == 1){ //ONLINE ENABALED
                if($onlineValue >= $amount){ // BELOW LIMIT
                        $status = "ONLINE SUCCESS";
                        $response = "00";
                        $isSave = true;
                } else { // ABOVE LIMIT
                        $status = "ABOVE ONLINE LIMIT";
                        $response = "51";
                        $isSave = true;
                }
        } else { //ONLINE OFF
                $status = "ONLINE OFF";
                $response = "07";
                $isSave = true;
        }
}
```

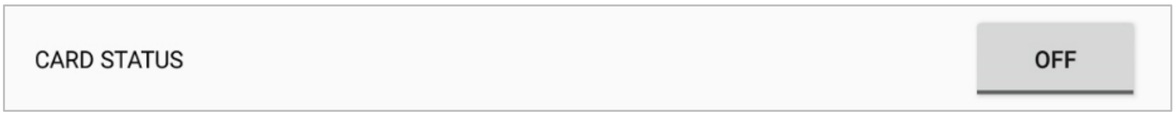

CARD STATUS OFF

Figure 6-12: Card Status Off

Customer can toggle the status by tapping on the on/off button, and clicking on the ‘Save’ button at the bottom of the form as in Figure 6-12 and Figure 6-13.

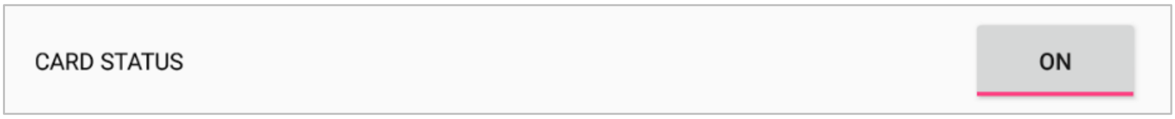

CARD STATUS ON

Figure 6-13: Card Status On

Status	**Response Code**
Card Off	Response Code 07
Card On	Proceed to next. Response from #B

Table 6-3: Card On/Off Status

Theses information will be saves in the ePaySwitch database. Turning on the card will enable the next three rules to be updated.

6.3.8 Online Transactions

Customers can set whether he/she can perform online transactions on a particular card. If the card is enabled for online transactions and also has set the maximum value, then the online transactions that involve the particular card will control according to the rules.

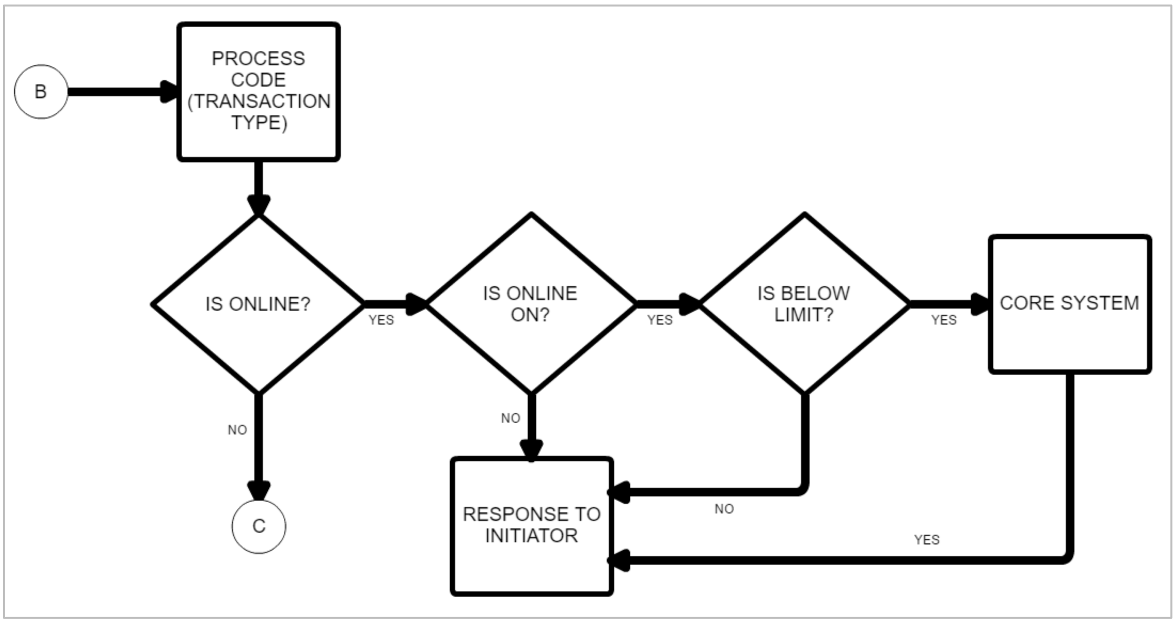
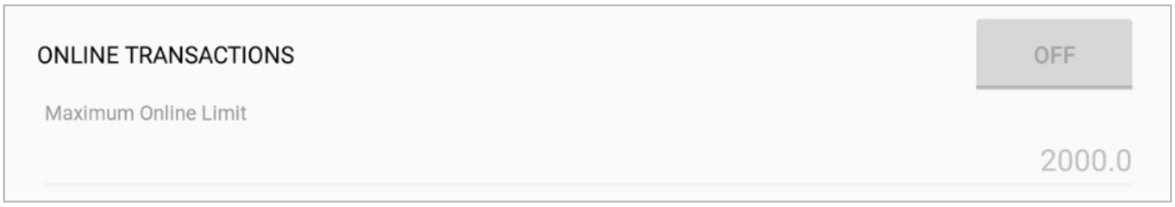


Figure 6-14: Online Transaction Rules Flow

Payments processor will extract the processing code from the ISO8583 message and check it to confirm as an online transaction type. If the transaction is online then payments processor will check whether customer has enabled the online facility for that particular card.


ONLINE TRANSACTIONS

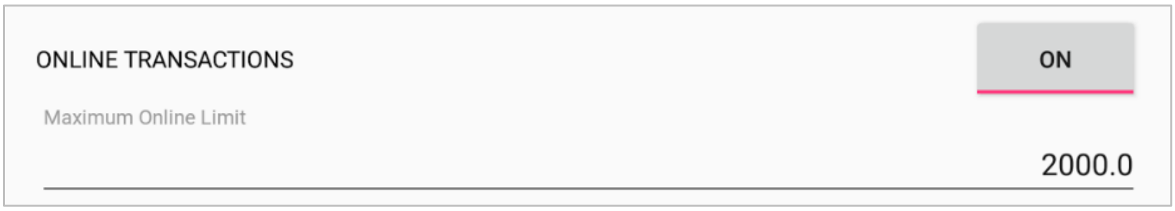
OFF

Maximum Online Limit

2000.0

Figure 6-15: Online Transactions Off

If the online transaction option is off, then the transaction originator will get a response code as mentioned in below table. Customer can turn on the online option by tapping on the on/off button.



ONLINE TRANSACTIONS

ON

Maximum Online Limit

2000.0

Figure 6-16: Online Transaction On and Rules Apply

After turning on the online option, customer can set the maximum limit per online transaction. When ePaySwitch validates the rules, it will check whether the incoming online transaction value is below the value set by the customer. Based on this comparison payment processor will decide to continue to core banking system if the transaction value is below the rule value or otherwise respond to transaction initiator. Table 6-4 shows the response codes for online transaction rules.

Status	**Response Code**
Online Off	Response Code 07
Online On Above Limit	Response Code 51
Online On Below Limit	Proceed to Core System. Core System Response Code

Table 6-4: Online Transactions Response Codes

6.3.9 Offline Transactions

Same way as mentioned above, a customer can apply a rule to perform offline transactions. An Offline transaction refers to a transaction where the card is present. Most common method of offline transactions are POS transactions. Customer can tap on the on/off button to turn on or off a selected card to perform online transactions. For card which is turned on can set a maximum per transaction value. This will consume the same method and service mentioned above.

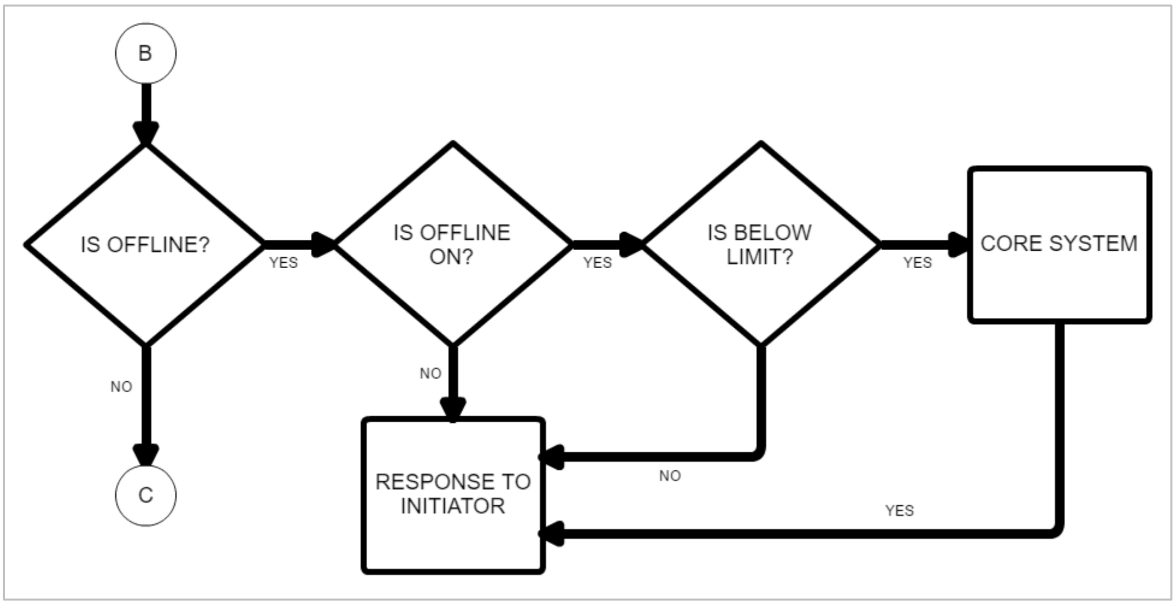


Figure 6-17: Offline Transations Flow

As showed in Figure 6-17, payments processor will decide the operation of the message to be passed depending on the rules applied by the customer. Code level implementation of the offline transactions flow is mentioned below.

```
if($processCode == 3000000){
        if($offlineOn == 1){ // OFFLINE ENABLED
                if($offlineValue >= $amount){ // BELOW LIMIT
                        $status = "OFFLINE SUCCESS";
                        $response = "00";
                        $isSave = true;
                } else { // ABOVE LIMIT
                        $status = "ABOVE OFFLINE LIMIT";
                        $response = "51";
                        $isSave = true;
                }
        } else { // OFFLINE OFF
                $status = "OFFLINE OFF";
                $response = "07";
                $isSave = true;
        }
}
```

Following Table 6-5 shows the response codes that will return to the transaction originator from the payments processor.

Status	Response Code
Offline Off	Response Code 07
Offline On Above Limit	Response Code 51
Offline On Below Limit	Proceed to Core System. Core System Response Code

Table 6-5: Offline Transactions Response Codes

6.3.10 Withdrawals

Cash withdrawal rule is applied as a control mechanism for withdrawals. Because, for withdrawals the PIN of particular card and the presence of an ATM machine is required. However, customer can keep the withdrawal function completely turn off using the mobile application which uses the same method and service as same way explained in section 6.3.8. Figure 6-18 illustrates the withdrawal flow.

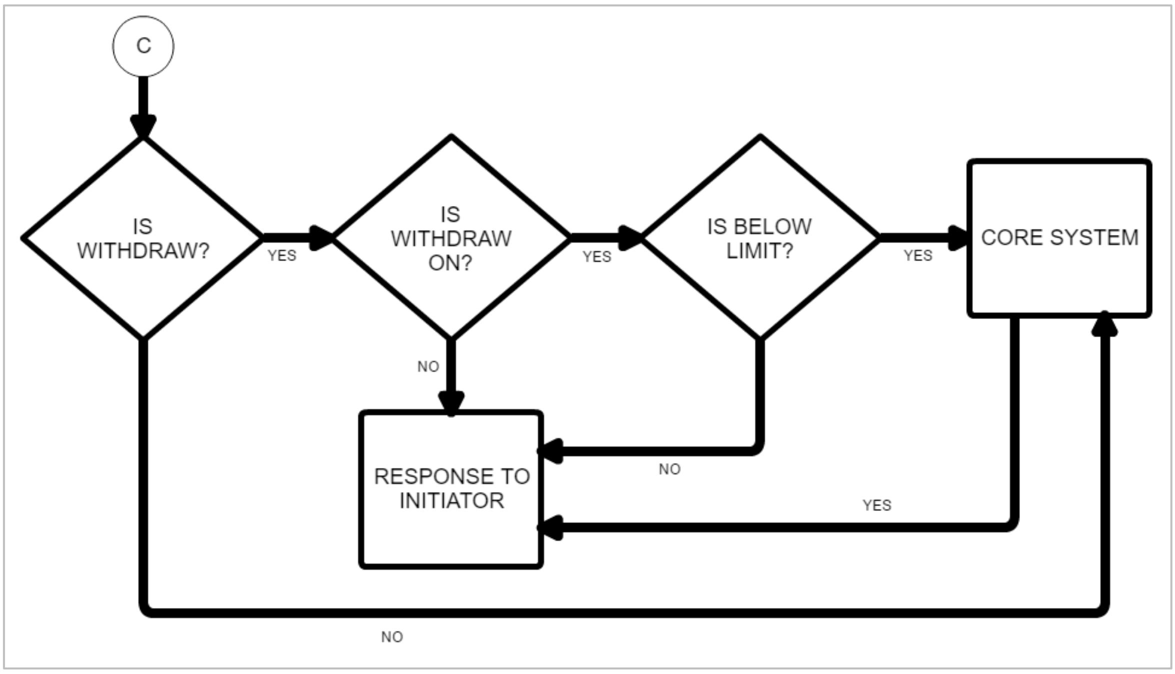


Figure 6-18: Withdrawal Flow

Code level implementation of withdrawal is as follows.

```
if($widtrawOn == 4000000){
	if($widtrawOn == 1){ //WITHDRAW ENABALED
		if($onlineValue >= $amount){ // BELOW LIMIT
			$status = "WITHDRAW SUCCESS";
			$response = "00";
			$isSave = true;
		} else { // ABOVE LIMIT
			$status = "ABOVE WITHDRAW LIMIT";
			$response = "51";
			$isSave = true;
		}
	} else { // WITHDRAW OFF
		$status = "WITHDRAW OFF";
		$response = "07";
		$isSave = true;
	}
}
```

Payments processor will check whether the card is enabled for withdrawals and the limit is below the rule value and responds accordingly as mentioned in the Table 6-6.

Status	Response Code
Withdrawal Off	Response Code 07
Withdrawal On Above Limit	Response Code 51
Withdrawal On Below Limit	Proceed to Core System. Core System Response Code

Table 6-6: Withdrawals Response Codes

When the processing code does not meet any of above, then payments processor will route the message directly to the core banking system and gets the response. Those processing codes are balance inquiry, transaction history etc.

6.4 Overall Implementation

Previous sections have described the individual implementation of each module in the ePaySwitch. Figure 6-19 illustrates the overall implementation of the ePaySwitch in the actual environment.

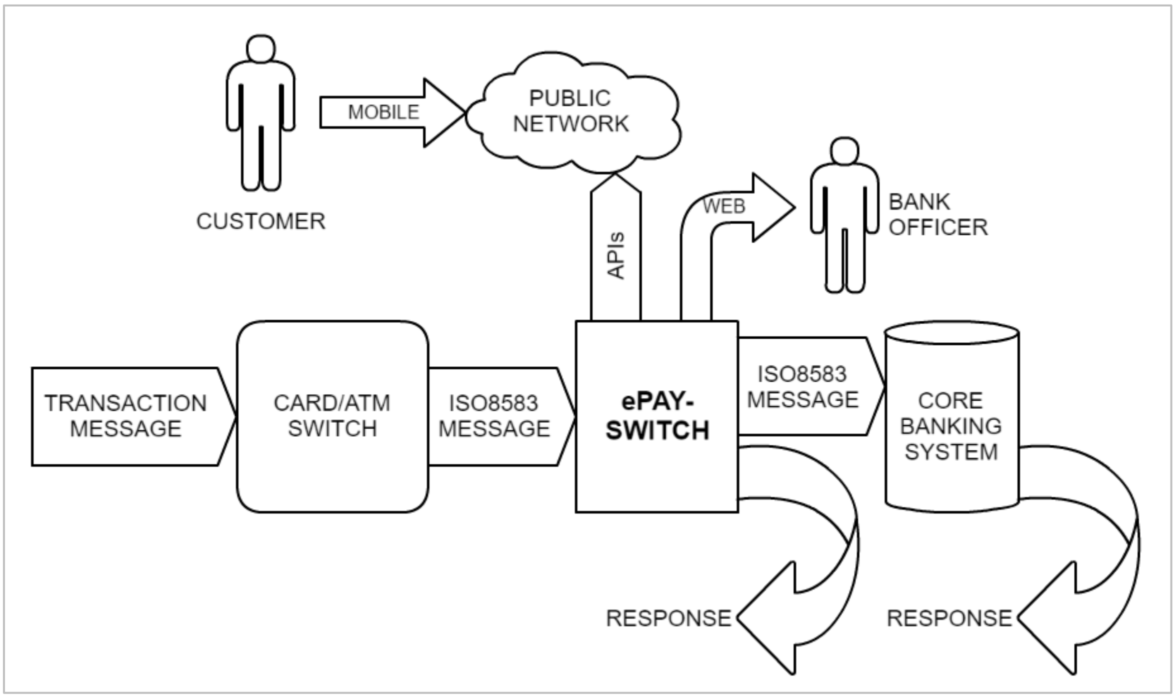


Figure 6-19: Overall Implementation

Customer will access the ePaySwitch to register, add and control cards via a public network. ePaySwitch give access by APIs. Banking officer have access to Customer and Cards Manager portal via intranet. Web portal is hosted within the banking domain.

When a transaction message arrives, card/ATM switch will format that message to ISO8583 specification according to particular core banking implementation. ePaySwitch will capture this message before passing to the core bank, decode and do the needful as per the implementation explained above. When needed ePaySwitch will reformat the message and send to the core banking system. According the response from ePaySwitch or the core banking system, ePaySwitch will send the response ISO8583 message back to the card/ATM switch.

6.5 Summary

This chapter focused on the implementation of ePaySwitch in detail. Individual module implementation, their behavior and the overall implementation also discussed.

Next chapter is about the testing and evaluation of the ePaySwitch which discusses the impact of the solution.

7 ePaySwitch Testing & Evaluation

7.1 Introduction

This chapter discusses the evaluation procedure for the ePaySwitch. Evaluation techniques, collected data and results are presented as software simulation and questionnaire and the test cases. The results have presented under two sections for the above mentioned two techniques in this chapter.

7.2 Software Simulation

Software simulator is tailor made software which uses the ISO8583 messaging for testing purposes. Since the actual simulators need a core banking system to test, a custom simulator is used which uses an actual message. The message has been decoded and replaced the following fields with the data that arrives with the transaction.

Field Number	**Description**
[003] Processing Code	Online – 200000
	Offline – 300000
	Withdraw – 400000
[004] Transaction Amount	*Values as described below*
[043] Card Acceptor Terminal Identification	Online EPAYSWITCH **ONLN** LOCATION COLOMBO LKA
	Offline EPAYSWITCH **OFLN** LOCATION COLOMBO LKA
	Withdraw EPAYSWITCH **WDRW** LOCATION COLOMBO LKA
[103] Card number	*Values as described below*

Payments processor has also configured to accept the above processing codes and identify the transaction type accordingly.

7.2.1 Simulation Test Cases

Simulation software has been used for evaluate the ePaySwitch working conditions. Simulation evaluates all three transaction types discussed throughout the solution.

Common Test Cases for Simulation

Customer & Card Details	
Customer Name	A B C Silva
Card Number	Online - 4123-4567-8914-7258
	Offline – 4216-8xxx-xxxx-0575
	Withdraw – 4216-8xxx-xxxx-0575
Maximum Online Value	Rs. 10,000/-
Maximum Online Value	Rs. 20,000/-
Maximum Withdraw Value	Rs. 30,000/-

Rest of the evaluation process assumes that the customer 'A B C Silva' registered and activated in the system.

Card Not Registered

Online	Below Value	Rs. 9,000/-
	Above Value	Rs. 11,000/-
Offline	Below Value	Rs. 19,000/-
	Above Value	Rs. 21,000/-
Withdraw	Below Value	Rs. 29,000/-
	Above Value	Rs. 31,000/-

Card Registered - Not Activated

Online	Below Value	Rs. 9,000/-
	Above Value	Rs. 11,000/-
Offline	Below Value	Rs. 19,000/-
	Above Value	Rs. 21,000/-
Withdraw	Below Value	Rs. 29,000/-
	Above Value	Rs. 31,000/-

Card Activated - Turned Off (by Default or by Customer)

Online	Online Status	Off
Offline	Offline Status	Off
Withdraw	Withdraw Status	Off

Card Turned On - Online Transactions

Online Turned ON	Online Below Value	Rs. 9,000/-
	Online Above Value	Rs. 11,000/-
Online Turned OFF	Offline Below Value	Rs. 9,000/-
	Offline Above Value	Rs. 11,000/-

Card Turned On - Offline Transactions

Offline Turned ON	Offline Below Value	Rs. 19,000/-
	Offline Above Value	Rs. 21,000/-
Offline Turned OFF	Offline Below Value	Rs. 19,000/-
	Offline Above Value	Rs. 21,000/-

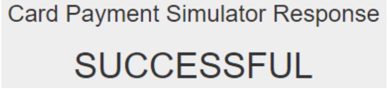
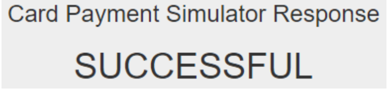
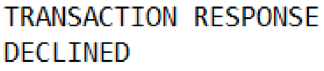
Card Turned On - Withdrawal Transactions

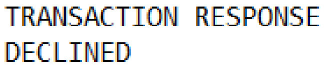
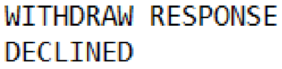
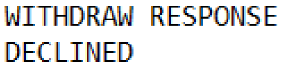
Withdrawal Turned ON	Withdrawal Below Value	Rs. 29,000/-
	Withdrawal Above Value	Rs. 31,000/-
Withdrawal Turned OFF	Withdrawal Below Value	Rs. 29,000/-
	Withdrawal Above Value	Rs. 31,000/-

Above tabulated test cases have been simulated in the software simulator and results are mentioned in the bellow tables in order.

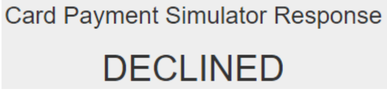
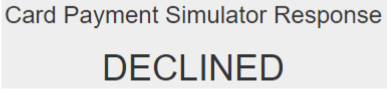
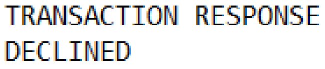
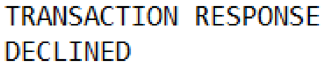
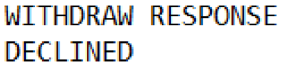
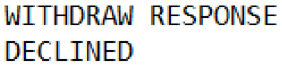
7.2.2 Simulation Test Output

Card Not Registered

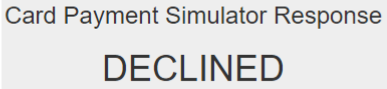
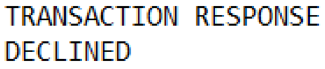
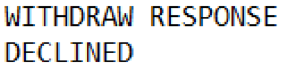
Online	Below Value	 Card Payment Simulator Response SUCCESSFUL
	Above Value	 Card Payment Simulator Response SUCCESSFUL
Offline	Below Value	 TRANSACTION RESPONSE DECLINED

	Above Value	 TRANSACTION RESPONSE DECLINED
Withdraw	Below Value	 WITHDRAW RESPONSE DECLINED
	Above Value	 WITHDRAW RESPONSE DECLINED

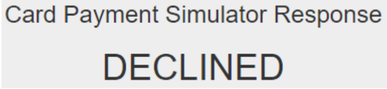
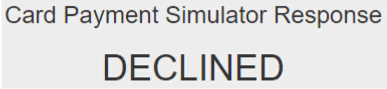
Card Registered - Not Activated

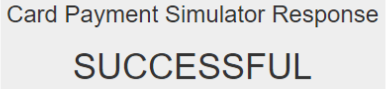
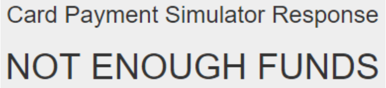
Online	Below Value	 Card Payment Simulator Response DECLINED
	Above Value	 Card Payment Simulator Response DECLINED
Offline	Below Value	 TRANSACTION RESPONSE DECLINED
	Above Value	 TRANSACTION RESPONSE DECLINED
Withdraw	Below Value	 WITHDRAW RESPONSE DECLINED
	Above Value	 WITHDRAW RESPONSE DECLINED

Card Activated - Turned Off (by Default or by Customer)

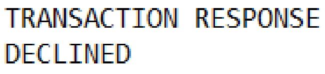
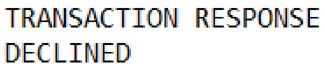
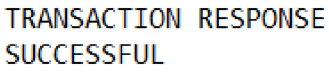
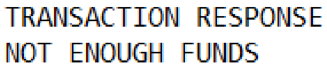
Online	Online Status	 Card Payment Simulator Response DECLINED
Offline	Offline Status	 TRANSACTION RESPONSE DECLINED
Withdraw	Withdraw Status	 WITHDRAW RESPONSE DECLINED

Card Turned On - Online Transactions

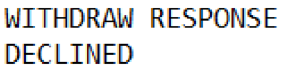
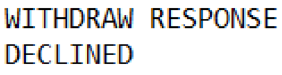
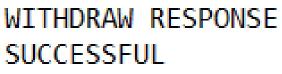
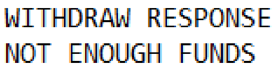
Online Turned OFF	Online Below Value	 Card Payment Simulator Response DECLINED
	Online Above Value	 Card Payment Simulator Response DECLINED

Online Turned ON	Offline Below Value	 Card Payment Simulator Response SUCCESSFUL
	Offline Above Value	 Card Payment Simulator Response NOT ENOUGH FUNDS

Card Turned On - Offline Transactions

Offline Turned OFF	Offline Below Value	 TRANSACTION RESPONSE DECLINED
	Offline Above Value	 TRANSACTION RESPONSE DECLINED
Offline Turned ON	Offline Below Value	 TRANSACTION RESPONSE SUCCESSFUL
	Offline Above Value	 TRANSACTION RESPONSE NOT ENOUGH FUNDS

Card Turned On - Withdrawal Transactions

Withdrawal Turned OFF	Withdrawal Below Value	 WITHDRAW RESPONSE DECLINED
	Withdrawal Above Value	 WITHDRAW RESPONSE DECLINED
Withdrawal Turned ON	Withdrawal Below Value	 WITHDRAW RESPONSE SUCCESSFUL
	Withdrawal Above Value	 WITHDRAW RESPONSE NOT ENOUGH FUNDS

Test results are discussed later in this this chapter.

7.3 Questionnaire

In order to gather feedback regarding the solution, an online questionnaire as a Google form published to random online users. Target was to gather 50 responses. Questions and answers are listed in the Appendix D for reference. Analytics of the results are presented in section 7.4.3 and 7.4.4 under topic 7.4.

7.4 Results

Following section reveals the result analysis and the conclusion of that results in software simulation and questionnaire.

7.4.1 Simulation

Software simulation showed that the ePaySwitch works as per the implementation and gives the desired outputs. Following are the findings of the simulation in tabulated form.

Input	**Output**
Card is not added to the system	As per the core system
Card added to the system (Inactive)	Card Not Active (Declined)
Card activated in the system (Off)	Card off (Declined)
Card on (online, offline & withdraw off)	-See below -
Online Transactions Online off	Card online off (Declined)
Online Transactions Online on – Above limit	Above limit (Not enough funds)
Online Transactions Online on – Below limit	As per the core system
Offline Transactions Offline off	Card offline off (Declined)
Offline Transactions Offline on – Above limit	Above limit (Not enough funds)
Offline Transactions Offline on – Below limit	As per the core system
Withdraw Transactions Offline off	Card withdraw off (Declined)
Withdraw Transactions Offline on – Above limit	Above limit (Not enough funds)
Withdraw Transactions Offline on – Below limit	As per the core system

Table 7-1: Simulation Results

7.4.2 Software Simulation Conclusion

ePaySwitch a technically viable solution for implementing card control and a switch for control card based solutions that will result in preventing card frauds happen due to loss and steel of cards. Additionally it can be used to control the transaction amount by transaction type for a particular card.

7.4.3 Questionnaire

From the collected 50 responses there were 44 people who use credit or debit cards. Among them, 27 people use their cards frequently. 24 of these people think somehow their cards can be stolen/lose and some other intruder can perform transaction, but keep on using. It is a percentage of 88.8%.

When we asked if they faced a similar kind on incident and whether they have any way of deactivating the particular card from above people, 91.7% of the their reaction was they have to call the particular banks card center. Rest has responded as they have no way of deactivating.

From people who was keep using the cards knowing the treat, we asked whether will like an application to control the application, 50% said that is a great idea and 33.3% may need to try it.

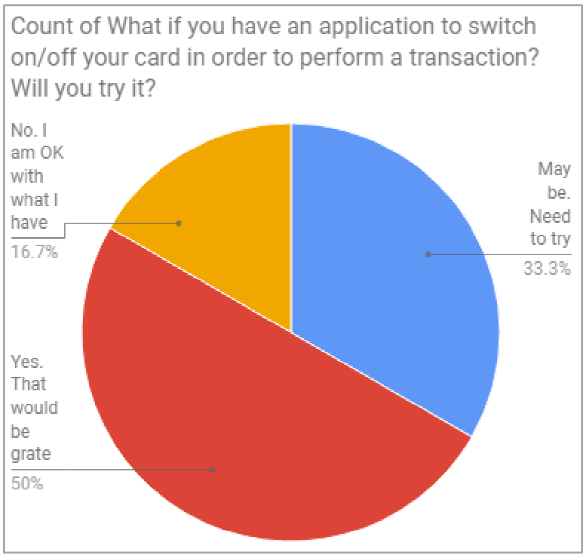


Figure 7-1: People who need to try ePaySwitch

With that positive reaction we gave a question showing how it works and how it feels to that 50% + 33.3%. Result was amazing.

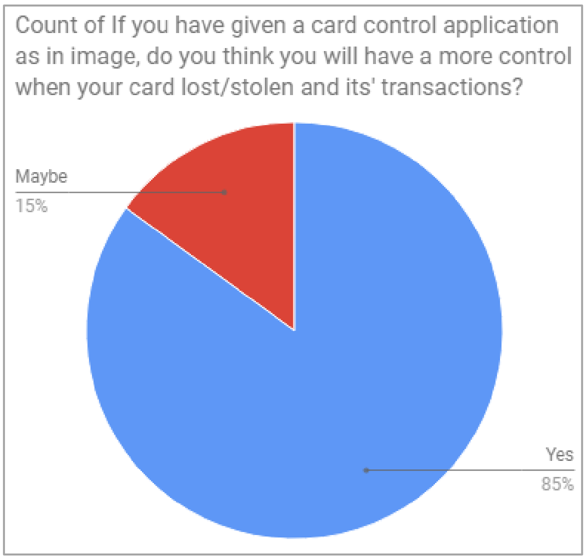


Figure 7-2: People who commented on ePaySwitch

At this point we have filtered 20 people. As the most suitable comment for the solution we received 19/20 saying “Worth it. May try”. And that is a percentage of 95%.

Among the frequent cards users there is around 29.6% people who have lost their card. 62.5% of them said they definitely use the solution. Other 37.5% said they would try it.

Result of the question on whether people have control over their transaction using card is shown in Figure 7-3.

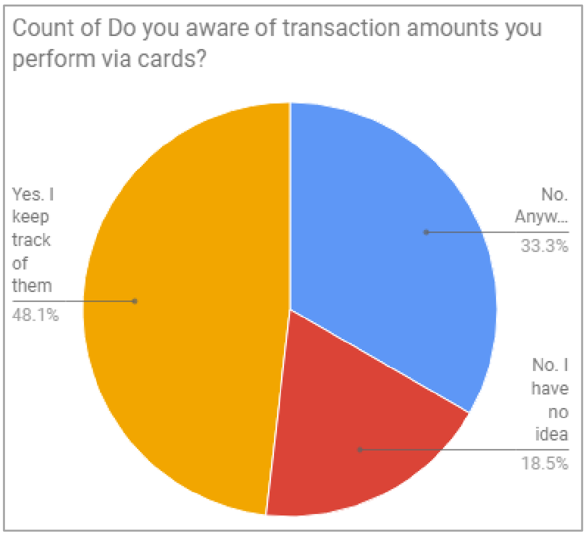


Figure 7-3: People's reaction for transaction amounts

From people who said that they have no idea on their spending, 80% have said they anyway use it and 100% have said this solution is worth using.

Unfortunately we were unable to collect decent amount of responses from the people are not using the cards or occasionally using cards to be analyzed. However from the responses we found that some are using both cash and cards while others are due to fraudulent transactions.

7.4.4 Questionnaire Conclusion

After analyzing the questionnaire we come to a conclusion that 95% of people who use cards often knowing the risk of cards are willing to accept the ePaySwitch solution. Additionally 100% of people who have no idea on their spending using cards also have accepted the Solution.

When we consider overall responses, following chart proved that ePaySwitch is practical and smart solution for Prevent Fraud & Control Electronic Transactions

7.5 Summary

This chapter discussed about the testing and evaluation of ePaySwitch. From the conclusions of simulator and the questionnaire, it is proven that the solution is technically viable and the customers would like to use such a solution.

8 Conclusion

8.1 Introduction

In previous chapter we discussed about the evaluation of the ePaySwitch. Throughout this chapter we will discuss about the overall achievements, problems encountered, limitations and further work.

8.2 ePaySwitch Achievements

Complete solution is PCI-DSS and ISO8583 compatible. Those compatibility requirements are compulsory for commercial implantations. Mobile application is tested and robust. And it is easy to use.

ePaySwitch solution is a workable solution for prevent card based fraudulent transactions and keep a control over the transactions. Test result shown that the solution is an acceptable and a practical solution that could be used as control technique of cards.

8.3 Problems and Limitations

When implementing in the actual environment, ePaySwitch Payments Processor requires an input from a card switch or from an ATM switch. Forwarding this message to the ePaySwitch should be done by the particular switch vendor. For some changes it will take time and cost. Nevertheless it will benefit the institute and customers.

For credit cards, the solution needs to be implemented inside the particular switch. Messages to and from the core system is not available in this context and it will be handled by the switch. This may limit the implementation as direct involvement of the switch vendor is required.

8.4 Improvements and Further Work

For further security, an API manager with identity server can be integrated with the ePaySwitch APIs. This will open virtual APIs to the public and the transactions will be secured using tokenization.

Further, this solution can be expanded beyond cards to control e-wallets, mobile-wallets and other forms of electronic transaction mediums. And also the solution can be expanded as a cloud solution to facilitate multiple financial institutes in a single mobile application.

8.5 Summary

Throughout the document we have discussed the existing solutions, research gaps, novel approach as ePaySwitch, technologies, design implementation and evaluation. Further discussed about the achievements, limitations and opened the dooer for further improvements.

This chapter concludes the research thesis under topic “Implementing a Software Switch and a Mobile Application to Prevent Frauds and Control the Usage of Electronic Transactions”.

References

Aleskerov, E., Freisleben, B., Rao, B., 1997. CARDWATCH: a neural network based database mining system for credit card fraud detection, in: Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr). Presented at the Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr), pp. 220–226. doi:10.1109/CIFER.1997.618940

Bhatla, T.P., Prabhu, V., Dua, A., 2003. Understanding Credit Card Frauds. Cards Bus. Rev. 1.

Bhusari, V., Patil, S., 2011. Study of Hidden Markov Model in Credit Card Fraudulent Detection. ResearchGate 20. doi:10.5120/2428-3263

Brause, R., Langsdorf, T., Hepp, M., 1999. Neural Data Mining for Credit Card Fraud Detection, in: ResearchGate. Presented at the Tools with Artificial Intelligence, 1999. Proceedings. 11th IEEE International Conference on, pp. 103–106. doi:10.1109/TAI.1999.809773

Chang, R.-I., Lai, L.-B., Su, W.-D., Wang, J.-C., Kouh, J.-S., 2007. Intrusion Detection by Backpropagation Neural Networks with Sample-Query and Attribute-Query. ResearchGate 3, 6–10. doi:10.5019/j.ijcir.2007.76

Chiu, C.-C., Tsai, C.-Y., 2004. A Web services-based collaborative scheme for credit card fraud detection, in: 2004 IEEE International Conference on E-Technology, E-Commerce and E-Service, 2004. EEE ’04. Presented at the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service, 2004. EEE ’04, pp. 177–181. doi:10.1109/EEE.2004.1287306

Duman, E., Ozcelik, M.H., 2011. Detecting credit card fraud by genetic algorithm and scatter search. Expert Syst. Appl. 38, 13057–13063. doi:10.1016/j.eswa.2011.04.110

Guo, T., Li, G.-Y., 2008. Neural data mining for credit card fraud detection, in: 2008 International Conference on Machine Learning and Cybernetics. Presented at the 2008 International Conference on Machine Learning and Cybernetics, pp. 3630–3634. doi:10.1109/ICMLC.2008.4621035

Holmes, T.E., 2016. Credit card rewards and cardholder satisfaction statistics [WWW Document]. URL http://www.creditcards.com/credit-card-news/cardholder-satisfaction-rewards-security-statistics-1276.php (accessed 8.14.16).

Holms, T., n.d. Credit card fraud and ID theft statistics [WWW Document]. www.CreditCards.com. URL http://www.creditcards.com/credit-card-news/credit-card-security-id-theft-fraud-statistics-1276.php (accessed 6.27.16).

ISO 8583, 2016. . Wikipedia.

Jason Steele, 2011. Why Most Americans Should Not Use Credit Cards. MoneyCrashers.com.

Kotelawala, H., 2016. Credit Card Usage in Sri Lanka: A Breakdown. Roar.lk.

Kundu, A., Panigrahi, S., Sural, S., Majumdar, A.K., 2009. BLAST-SSAHA Hybridization for Credit Card Fraud Detection. IEEE Trans. Dependable Secure Comput. 6, 309–315. doi:10.1109/TDSC.2009.11

MD5, 2017. . Wikipedia.

Meshram, P.L., Yenganti, T., 2013. Credit and ATM Card Fraud Prevention Using Multiple Cryptographic Algorithm. Int. J. Adv. Res. Comput. Sci. Softw. Eng. 3, 1300–1305.

Patidar, R., Sharma, L., 2011. Credit Card Fraud Detection Using Neural Network. Int. J. Soft Comput. Eng. IJSCE 1, 32–38.

Raj, S.B.E., Portia, A.A., 2011. Analysis on credit card fraud detection methods, in: 2011 International Conference on Computer, Communication and Electrical Technology (ICCCET). Presented at the 2011 International Conference on Computer, Communication and Electrical Technology (ICCCET), pp. 152–156. doi:10.1109/ICCCET.2011.5762457

Sahin, Y., Duman, E., 2011. Detecting Credit Card Fraud by Decision Trees and Support Vector Machines. ResearchGate 1, 442–447.

SHA-1, 2017. . Wikipedia.

Srivastava, A., Kundu, A., Sural, S., Majumdar, A., 2008. Credit Card Fraud Detection Using Hidden Markov Model. IEEE Trans. Dependable Secure Comput. 5, 37–48. doi:10.1109/TDSC.2007.70228

Stolfo, S.J., Fan, D.W., Lee, W., Prodromidis, A.L., Chan, P.K., 1998. Credit Card Fraud Detection Using Meta-Learning: Issues and Initial Results. ResearchGate.

Syeda, M., Zhang, Y.-Q., Pan, Y., 2002. Parallel granular neural networks for fast credit card fraud detection, in: Proceedings of the 2002 IEEE International Conference on Fuzzy Systems, 2002. FUZZ-IEEE’02. Presented at the Proceedings of the 2002

IEEE International Conference on Fuzzy Systems, 2002. FUZZ-IEEE’02, pp. 572–577. doi:10.1109/FUZZ.2002.1005055

Vadoodparast, M., Hamdan, A.R., Hafiz, 2015. Fraudulent Electronic Transaction Detection Using Dynamic KDA Model. Int. J. Comput. Sci. Inf. Secur. 12.

Vatsa, V., Sural, S., Majumdar, A.K., 2005. A Game-theoretic Approach to Credit Card Fraud Detection, in: Proceedings of the First International Conference on Information Systems Security, ICISS’05. Springer-Verlag, Berlin, Heidelberg, pp. 263–276. doi:10.1007/11593980_20

Appendix A – ISO8583 Message Sample

Sample of Online Payment

Sample Request Message

```
0200FABC44C12880C00000000000040000001641234567891472582000000000100000
0000000000001214012627610000006605440127461214190960110510004069144423 24
691760100091433 =190922116459321005053486 2736EPAYSWITCH ONLN
LOCATION COLOMBO LKA14414416412345678 9147258
```

Field	Value
MTI	0200
[001] Bitmap (hex 62)	FABC44C12820C0000000000004000000
[002] Primary Account Number (n..16)	16 4123456789147258
[003] Processing Code (n6)	200000
[004] Amount, Transaction (n12)	000000100000
[005] Amount, Settlement (n12)	000000000000
[007] Trans. Date/Time (n10)	1214012627
[009] Conversion rate (n8)	10000006
[011] System trace audit number (n6)	605440
[012] Time (n6)	012746 (HHmmss)
[013] Date (n4)	1214 (MMdd)
[014] Date, expiration (n4)	1909
[018] Merchant type (n4)	6011
[022] Point of service entry mode (n3)	051
[025] Point of service condition code (n2)	00
[026] Point of service capture code (n2)	04
[032] Acquiring institution identification code (n..6)	06 914442
[035] Track 2 data (n..32)	32 4691760100091433 =19092211645932
[037] Retrieval reference number (an12)	100505348627

[043] Card acceptor terminal identification (ans..32)	36 EPAYSWITCH ONLN LOCATION COLOMBO LKA
[049] Currency code, transaction (n3)	144
[050] Currency code, settlement (n3)	144
[103] Account identification 1 (n..16)	16 4123456789147258

Sample Response Message

0200FABC44C12880C000000000000400000016412345678914725820000000001000000000000000000121401262761000000660544012746121419096011051000406914442324691760100091433 =19092211645932100505348627**00**36EPAYSWITCH ONLN LOCATION COLOMBO LKA144144

Field	Value
MTI	0210
[001] Bitmap (hex 62)	FABC44C12A20C0000000000000000000
[002] Primary Account Number (n..16)	16 4123456789147258
[003] Processing Code (n6)	200000
[004] Amount, Transaction (n12)	000000100000
[005] Amount, Settlement (n12)	000000000000
[007] Trans. Date/Time (n10)	1214012627
[009] Conversion rate (n8)	10000006
[011] System trace audit number (n6)	605440
[012] Time (n6)	012746 (HHmmss)
[013] Date (n4)	1214 (MMdd)
[014] Date, expiration (n4)	1909
[018] Merchant type (n4)	6011
[022] Point of service entry mode (n3)	051
[025] Point of service condition code (n2)	00
[026] Point of service capture code (n2)	04
[032] Acquiring institution identification code (n..6)	06 914442
[035] Track 2 data (n..32)	32 4691760100091433 =19092211645932
[037] Retrieval reference number (an12)	100505348627
[039] Response Code (an2)	Success: 00
	Declined: 07
	Not enough funds: 51
	Suspicious: 59
[043] Card acceptor terminal identification	36 EPAYSWITCH ONLN LOCATION

(n..32)	COLOMBO LKA
[049] Currency code, transaction (n3)	144
[050] Currency code, settlement (n3)	144

Appendix B – Important Code Segments

API Service Implementation

Customers.java

```
public class Customers implements Serializable {

    private static final long serialVersionUID = 1L;
    private Long customerId;
    private String fullName;
    private String nid;
    private String mobile;
    private short isActive;
    private String username;
    private String password;
    private String respo;

    public Customers() {
    }
    public Long getCustomerId() {
        return customerId;
    }
    public void setCustomerId(Long customerId) {
        this.customerId = customerId;
    }
    public String getFullName() {
        return fullName;
    }
    public void setFullName(String fullName) {
        this.fullName = fullName;
    }
    public String getNid() {
        return nid;
    }
    public void setNid(String nid) {
        this.nid = nid;
    }
    public String getMobile() {
        return mobile;
    }
    public void setMobile(String mobile) {
        this.mobile = mobile;
    }
    public short getIsActive() {
        return isActive;
    }
    public void setIsActive(short isActive) {
        this.isActive = isActive;
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public String getRespo() {
        return respo;
    }
    public void setRespo(String respo) {
        this.respo = respo;
    }

    public Customers[] SignIn(String username, String password) {
        Customers[] ccArray = null;
        dbAccess db = new dbAccess();
        Connection con = db.GetConnection();
        try {
            Statement stmnt = con.createStatement();
            ResultSet result;
            result = stmnt.executeQuery("CALL sp_login('" + username + "',  '" +
password + "')");
            ccArray = new Customers[1];
            while (result.next()) {
                Customers cc = new Customers();
                if (result.getString("resp").equals("0x9300")) {
                    cc.setRespo(result.getString("resp"));
                } else {
                    cc.setRespo(result.getString("resp"));

cc.setCustomerId(Long.parseLong(result.getString("customer_id")));
                    cc.setUsername(result.getString("username"));
                    cc.setFullName(result.getString("full_name"));

cc.setIsActive(Short.parseShort(result.getString("is_active")));
                }
                ccArray[0] = cc;
            }
        } catch (SQLException ex) {
            Logger.getLogger(card_service_1.class.getName()).log(Level.SEVERE,
null, ex);
        }
        return ccArray;
    }

    public String Register(Customers customers) {
        dbAccess db = new dbAccess();
        Connection con = db.GetConnection();
        String response = "";
        try {
            Statement stmnt = con.createStatement();
            ResultSet result;
            result = stmnt.executeQuery(
                    "CALL sp_register("
                            + "'" + customers.fullName + "', "
                            + "'" + customers.nid + "', "
                            + "'" + customers.mobile + "', "
                            + "'" + customers.username + "', "
                            + "'" + customers.password + "' "
                            + ")");
            while (result.next()) {
                response = result.getString("resp");
            }
        } catch (SQLException ex) {
            Logger.getLogger(card_service_1.class.getName()).log(Level.SEVERE,
null, ex);
        }
        return response;
    }
    String ChangePass(String CustomerID, String OldPassword, String NewPassword)
{
        dbAccess db = new dbAccess();
        Connection con = db.GetConnection();
        String response = "";
        try {
            Statement stmnt = con.createStatement();
            ResultSet result;
            result = stmnt.executeQuery(
                    "CALL sp_change_password("
                            + "'" + CustomerID + "', "
                            + "'" + OldPassword + "', "
                            + "'" + NewPassword + "' "
                            + ")");
            while (result.next()) {
                response = result.getString("resp");
            }
        } catch (SQLException ex) {
            Logger.getLogger(card_service_1.class.getName()).log(Level.SEVERE,
null, ex);
        }
        return response;
    }
}
```

CustomerChannels.java

```
public class CustomerChannels implements Serializable {

    private static final long serialVersionUID = 1L;
    private long customerId;
    private int channelId;
    private String accountName;
    private String accountSha256;
    private String accountMask;
    private String expiry;
    private String nameIn;
    private short isActive;
    private short isOn;
    private Float maxOnlineValue;
    private short isOnlineOn;
    private Float maxOfflineValue;
    private short isOfflineOn;
    private Float maxWithdrawValue;
    private short isWithdrawOn;

    public CustomerChannels() {
    }
    public long getCustomerId() {
        return customerId;
    }
    public void setCustomerId(long customerId) {
        this.customerId = customerId;
    }
    public int getChannelId() {
        return channelId;
    }
    public void setChannelId(int channelId) {
        this.channelId = channelId;
    }
    public String getAccountName() {
        return accountName;
    }
    public void setAccountName(String accountName) {
        this.accountName = accountName;
    }
    public String getAccountSha256() {
        return accountSha256;
    }
    public void setAccountSha256(String accountSha256) {
        this.accountSha256 = accountSha256;
    }
    public String getAccountMask() {
        return accountMask;
    }
    public void setAccountMask(String accountMask) {
        this.accountMask = accountMask;
    }
    public String getExpiry() {
        return expiry;
    }
    public void setExpiry(String expiry) {
        this.expiry = expiry;
    }
    public String getNameIn() {
        return nameIn;
    }
    public void setNameIn(String nameIn) {
        this.nameIn = nameIn;
    }
    public short getIsActive() {
        return isActive;
    }
    public void setIsActive(short isActive) {
        this.isActive = isActive;
    }
    public short getIsOn() {
        return isOn;
    }
    public void setIsOn(short isOn) {
        this.isOn = isOn;
    }
    public Float getMaxOnlineValue() {
        return maxOnlineValue;
    }
    public void setMaxOnlineValue(Float maxOnlineValue) {
        this.maxOnlineValue = maxOnlineValue;
    }
    public short getIsOnlineOn() {
        return isOnlineOn;
    }
    public void setIsOnlineOn(short isOnlineOn) {
        this.isOnlineOn = isOnlineOn;
    }
    public Float getMaxOfflineValue() {
        return maxOfflineValue;
    }
    public void setMaxOfflineValue(Float maxOfflineValue) {
        this.maxOfflineValue = maxOfflineValue;
    }
    public short getIsOfflineOn() {
        return isOfflineOn;
    }
    public void setIsOfflineOn(short isOfflineOn) {
        this.isOfflineOn = isOfflineOn;
    }
    public Float getMaxWithdrawValue() {
        return maxWithdrawValue;
    }
    public void setMaxWithdrawValue(Float maxWithdrawValue) {
        this.maxWithdrawValue = maxWithdrawValue;
    }

    /**
```

```
     *
     * @return
     */
    public short getIsWithdrawOn() {
        return isWithdrawOn;
    }

    /**
     *
     * @param isWithdrawOn
     */
    public void setIsWithdrawOn(short isWithdrawOn) {
        this.isWithdrawOn = isWithdrawOn;
    }

    /**
     *
     * @param userID
     * @param IsActive
     * @return
     */
    public CustomerChannels[] GetCards(Long userID, String IsActive) {
        CustomerChannels[] ccArray = null;
        dbAccess db = new dbAccess();
        Connection con = db.GetConnection();
        try {
            Statement stmnt = con.createStatement();
            ResultSet result, backResult;
            result = backResult = stmnt.executeQuery(
                    "CALL sp_get_cards("
                    + "'" + userID + "', "
                    + "'" + IsActive + "' "
                    + ")");
            int j = 0;
            while (result.next()) {
                j++;
            }
            ccArray = new CustomerChannels[j];
            int i = 0;

            while (backResult.previous()) {
                CustomerChannels cc = new CustomerChannels();
                cc.setAccountSha256(backResult.getString("account_sha256"));
                cc.setAccountMask(backResult.getString("account_mask"));
                cc.setAccountName(backResult.getString("account_name"));

cc.setIsActive(Short.parseShort(backResult.getString("is_active")));
                cc.setIsOn(Short.parseShort(backResult.getString("is_on")));
                ccArray[i] = cc;
                i++;
            }
        } catch (SQLException ex) {
            Logger.getLogger(card_service_1.class.getName()).log(Level.SEVERE,
null, ex);
        }
        return ccArray;
    }

    /**
     *
     * @param cardData
     * @return
     */
    public String AddCard(CustomerChannels cardData) {
        dbAccess db = new dbAccess();
        Connection con = db.GetConnection();
        String response = "";
        try {
            Statement stmnt = con.createStatement();
            ResultSet result;
            result = stmnt.executeQuery(
                    "CALL sp_addcard("
                    + "'" + cardData.getCustomerId() + "', "
                    + "'" + cardData.getAccountName() + "', "
                    + "'" + cardData.getAccountSha256() + "', "
                    + "'" + cardData.getAccountMask() + "', "
                    + "'" + cardData.getExpiry() + "', "
                    + "'" + cardData.getNameIn() + "' "
                    + ")");
            while (result.next()) {
                response = result.getString("resp");
            }
        } catch (SQLException ex) {
            Logger.getLogger(card_service_1.class.getName()).log(Level.SEVERE,
null, ex);
        }
        return response;
    }

    /**
     *
     * @param cardSha256
     * @return
     */
    public CustomerChannels[] GetCardDetails(String cardSha256) {
        CustomerChannels[] ccArray = new CustomerChannels[1];
        dbAccess db = new dbAccess();
        Connection con = db.GetConnection();
        try {
            Statement stmnt = con.createStatement();
            ResultSet result;
            result = stmnt.executeQuery("CALL sp_cardview('" + cardSha256 +
"')");
            while (result.next()) {
                CustomerChannels cc = new CustomerChannels();
                cc.setAccountSha256(result.getString("account_sha256"));
                cc.setAccountMask(result.getString("account_mask"));
                cc.setAccountName(result.getString("account_name"));
                cc.setExpiry(result.getString("expiry"));
                cc.setNameIn(result.getString("name_in"));
                cc.setIsOn(Short.parseShort(result.getString("is_on")));

cc.setMaxOnlineValue(Float.parseFloat(result.getString("max_online_value")));

cc.setIsOnlineOn(Short.parseShort(result.getString("is_online_on")));

cc.setMaxOfflineValue(Float.parseFloat(result.getString("max_offline_value")));

cc.setIsOfflineOn(Short.parseShort(result.getString("is_offline_on")));

cc.setMaxWithdrawValue(Float.parseFloat(result.getString("max_withdraw_value")))
;

cc.setIsWithdrawOn(Short.parseShort(result.getString("is_withdraw_on")));
                ccArray[0] = cc;
            }
        } catch (SQLException ex) {
            Logger.getLogger(card_service_1.class.getName()).log(Level.SEVERE,
null, ex);
        }
        return ccArray;
    }

    public String UpdateCard(CustomerChannels ccUpdateCard) {
        dbAccess db = new dbAccess();
        Connection con = db.GetConnection();
        String response = "";
        try {
            Statement stmnt = con.createStatement();
            ResultSet result;
            result = stmnt.executeQuery("CALL sp_updatecard("
                    + "'" + ccUpdateCard.getAccountSha256() + "', "
                    + "'" + ccUpdateCard.getCustomerId() + "', "
                    + "'" + ccUpdateCard.getAccountName() + "', "
                    + "'" + String.valueOf(ccUpdateCard.getIsOn()) + "', "
                    + "'" + ccUpdateCard.getMaxOnlineValue() + "', "
                    + "'" + String.valueOf(ccUpdateCard.getIsOnlineOn()) + "', "
                    + "'" + ccUpdateCard.getMaxOfflineValue()+ "', "
                    +  "'"  +  String.valueOf(ccUpdateCard.getIsOfflineOn())  +  "',
"
                    + "'" + ccUpdateCard.getMaxWithdrawValue()+ "', "
                    +  "'"  +  String.valueOf(ccUpdateCard.getIsWithdrawOn())  +  "'
"
                    + ")");
            while (result.next()) {
                response = result.getString("resp");
            }
        } catch (SQLException ex) {
            Logger.getLogger(card_service_1.class.getName()).log(Level.SEVERE,
null, ex);
        }
        return response;
    }

    String DeleteCard(String CardSha256, String CustomerID) {
        dbAccess db = new dbAccess();
        Connection con = db.GetConnection();
        String response = "";
        try {
            Statement stmnt = con.createStatement();
            ResultSet result;
            result = stmnt.executeQuery(
                    "CALL sp_deletecard("
                    + "'" + CardSha256 + "', "
                    + "'" + CustomerID + "' "
                    + ")");
            while (result.next()) {
                response = result.getString("resp");
            }
        } catch (SQLException ex) {
            Logger.getLogger(card_service_1.class.getName()).log(Level.SEVERE,
null, ex);
        }
        return response;
    }
}
```

user_service_1.java

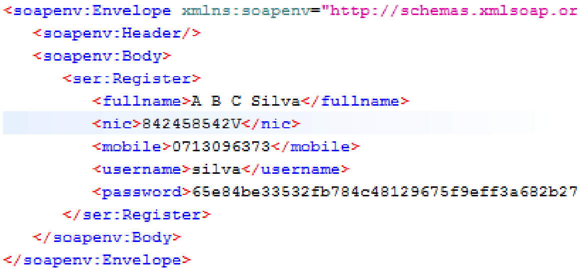
```
public class user_service_1 {

    /**
     * This is a sample web service operation
     * @param username
     * @param password
     * @return
     */
    @WebMethod(operationName = "SignIn")
    public Customers[] SignIn(
            @WebParam(name = "username") String username,
            @WebParam(name = "password") String password) {
        Customers ccSignIn = new Customers();
        return ccSignIn.SignIn(username, password);
    }

    @WebMethod(operationName = "Register")
    public String Register(
            @WebParam(name = "fullname") String fullname,
            @WebParam(name = "nic") String nic,
            @WebParam(name = "mobile") String mobile,
            @WebParam(name = "username") String username,
            @WebParam(name = "password") String password) {
        Customers ccRegister = new Customers();
        ccRegister.setFullName(fullname);
        ccRegister.setNid(nic);
        ccRegister.setMobile(mobile);
        ccRegister.setUsername(username);
        ccRegister.setPassword(password);
        return ccRegister.Register(ccRegister);
    }

    @WebMethod(operationName = "ChangePass")
    public String ChangePass(
            @WebParam(name = "CustomerID") String CustomerID,
            @WebParam(name = "OldPassword") String OldPassword,
            @WebParam(name = "NewPassword") String NewPassword) {
        Customers ccSignIn = new Customers();
        return ccSignIn.ChangePass(CustomerID, OldPassword, NewPassword);
    }
}
```

card_service_1.java

```
public class card_service_1 {

    /**
     * This is a sample web service operation
     *
     * @param UserID
     * @param IsActive
     * @return CustomerCannels_
     */
    @WebMethod(operationName = "GetCards")
    public CustomerChannels[] GetCards(
            @WebParam(name = "userID") Long UserID,
            @WebParam(name = "IsActive") String IsActive) {
        CustomerChannels ccInactive = new CustomerChannels();
        return ccInactive.GetCards(UserID, IsActive);
    }

    /**
     *
     * @param CardSha256
     * @return
     */
    @WebMethod(operationName = "GetCardDetails")
    public   CustomerChannels[]   GetCardDetails(@WebParam(name   =   "CardSha256")
String CardSha256) {
        CustomerChannels ccInactive = new CustomerChannels();
        return ccInactive.GetCardDetails(CardSha256);
    }

    /**
     *
     * @param CustomerID
     * @param CardName
     * @param CardSha256
     * @param CardMask
     * @param Expiry
     * @param NameInCard
     * @return
     */
    @WebMethod(operationName = "AddCard")
    public String AddCard(
            @WebParam(name = "CustomerID") Long CustomerID,
            @WebParam(name = "CardName") String CardName,
            @WebParam(name = "CardSha256") String CardSha256,
            @WebParam(name = "CardMask") String CardMask,
            @WebParam(name = "Expiry") String Expiry,
            @WebParam(name = "NameInCard") String NameInCard) {
        CustomerChannels ccAddCard = new CustomerChannels();
        ccAddCard.setCustomerId(CustomerID);
        ccAddCard.setAccountName(CardName);
        ccAddCard.setAccountSha256(CardSha256);
        ccAddCard.setAccountMask(CardMask);
        ccAddCard.setExpiry(Expiry);
        ccAddCard.setNameIn(NameInCard);
        return ccAddCard.AddCard(ccAddCard);
    }

    /**
     *
     * @param CustomerID
     * @param CardName
     * @param CardSha256
     * @param IsOn
     * @param MaxOnlineValue
     * @param IsOnlineOn
     * @param MaxOfflineValue
     * @param IsOfflineOn
     * @param MaxWithdrawValue
     * @param IsWithdrawOn
     * @return
     */
    @WebMethod(operationName = "UpdateCard")
    public String UpdateCard(
            @WebParam(name = "CustomerID") Long CustomerID,
            @WebParam(name = "CardName") String CardName,
            @WebParam(name = "CardSha256") String CardSha256,
            @WebParam(name = "IsOn") String IsOn,
            @WebParam(name = "MaxOnlineValue") String MaxOnlineValue,
            @WebParam(name = "IsOnlineOn") String IsOnlineOn,
            @WebParam(name = "MaxOfflineValue") String MaxOfflineValue,
            @WebParam(name = "IsOfflineOn") String IsOfflineOn,
            @WebParam(name = "MaxWithdrawValue") String MaxWithdrawValue,
            @WebParam(name = "IsWithdrawOn") String IsWithdrawOn) {
        CustomerChannels ccUpdateCard = new CustomerChannels();
        ccUpdateCard.setCustomerId(CustomerID);
        ccUpdateCard.setAccountName(CardName);
        ccUpdateCard.setAccountSha256(CardSha256);
        ccUpdateCard.setIsOn(Short.parseShort(IsOn));
        ccUpdateCard.setMaxOnlineValue(Float.parseFloat(MaxOnlineValue));
        ccUpdateCard.setIsOnlineOn(Short.parseShort(IsOnlineOn));
        ccUpdateCard.setMaxOfflineValue(Float.parseFloat(MaxOfflineValue));
        ccUpdateCard.setIsOfflineOn(Short.parseShort(IsOfflineOn));
        ccUpdateCard.setMaxWithdrawValue(Float.parseFloat(MaxWithdrawValue));
        ccUpdateCard.setIsWithdrawOn(Short.parseShort(IsWithdrawOn));
        return ccUpdateCard.UpdateCard(ccUpdateCard);
    }

    /**
     *
     * @param CardSha256
     * @param CustomerID
     * @return
     */
    @WebMethod(operationName = "DeleteCard")
    public String DeleteCard(
            @WebParam(name = "CardSha256") String CardSha256,
            @WebParam(name = "CustomerID") String CustomerID
    ) {
        CustomerChannels ccDeleteCard = new CustomerChannels();
        return ccDeleteCard.DeleteCard(CardSha256, CustomerID);
    }
}
```

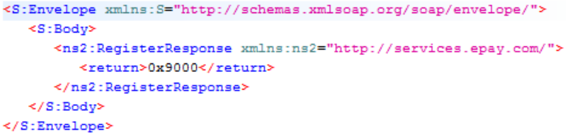
SOAP Service Requests

“Register” Request



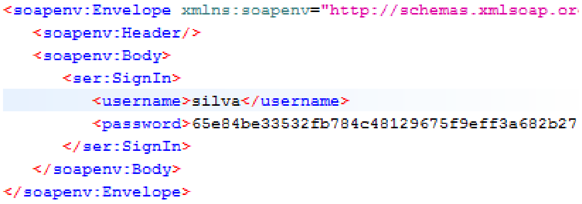
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.or
   <soapenv:Header/>
   <soapenv:Body>
      <ser:Register>
         <fullname>A B C Silva</fullname>
         <nic>842458542V</nic>
         <mobile>0713096373</mobile>
         <username>silva</username>
         <password>65e84be33532fb784c48129675f9eff3a682b27
      </ser:Register>
   </soapenv:Body>
</soapenv:Envelope>
```

“Register” Response



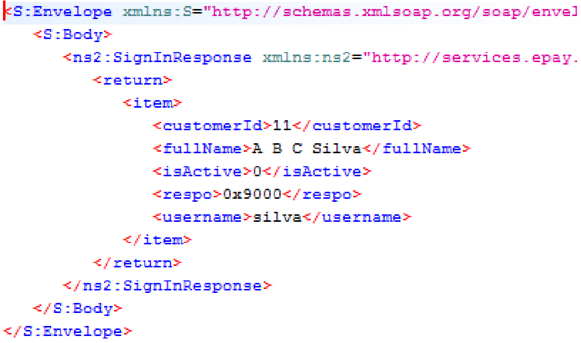
```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:RegisterResponse xmlns:ns2="http://services.epay.com/">
         <return>0x9000</return>
      </ns2:RegisterResponse>
   </S:Body>
</S:Envelope>
```

“SignIn” Request



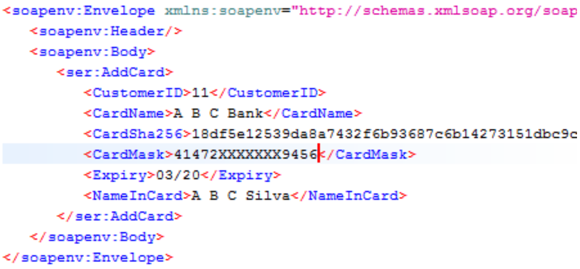
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.or
   <soapenv:Header/>
   <soapenv:Body>
      <ser:SignIn>
         <username>silva</username>
         <password>65e84be33532fb784c48129675f9eff3a682b27
      </ser:SignIn>
   </soapenv:Body>
</soapenv:Envelope>
```

“ SignIn” Response



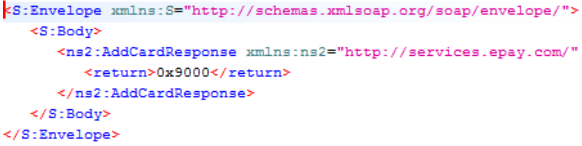
```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envel
   <S:Body>
      <ns2:SignInResponse xmlns:ns2="http://services.epay.
         <return>
            <item>
               <customerId>11</customerId>
               <fullName>A B C Silva</fullName>
               <isActive>0</isActive>
               <respo>0x9000</respo>
               <username>silva</username>
            </item>
         </return>
      </ns2:SignInResponse>
   </S:Body>
</S:Envelope>
```

“AddCard” Request



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap
   <soapenv:Header/>
   <soapenv:Body>
      <ser:AddCard>
         <CustomerID>11</CustomerID>
         <CardName>A B C Bank</CardName>
         <CardSha256>18df5e12539da8a7432f6b93687c6b14273151dbc9c
         <CardMask>41472XXXXXXX9456</CardMask>
         <Expiry>03/20</Expiry>
         <NameInCard>A B C Silva</NameInCard>
      </ser:AddCard>
   </soapenv:Body>
</soapenv:Envelope>
```

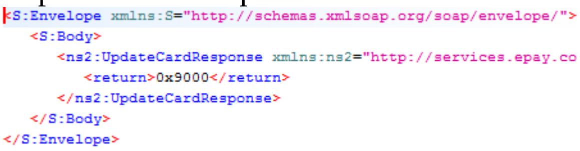
“AddCard” Response



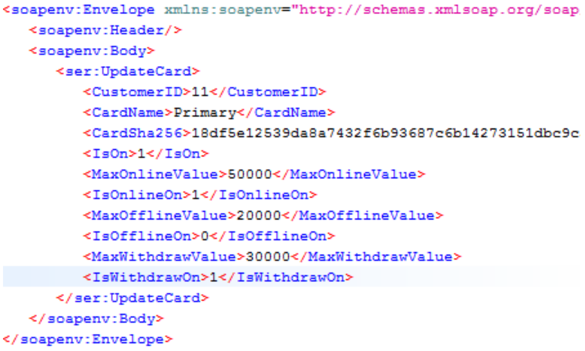
```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:AddCardResponse xmlns:ns2="http://services.epay.com/"
         <return>0x9000</return>
      </ns2:AddCardResponse>
   </S:Body>
</S:Envelope>
```

“UpdateCard” Request

“UpdateCard” Response

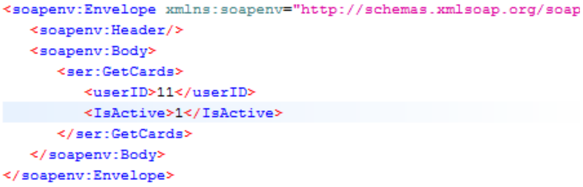


```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:UpdateCardResponse xmlns:ns2="http://services.epay.co
         <return>0x9000</return>
      </ns2:UpdateCardResponse>
   </S:Body>
</S:Envelope>
```



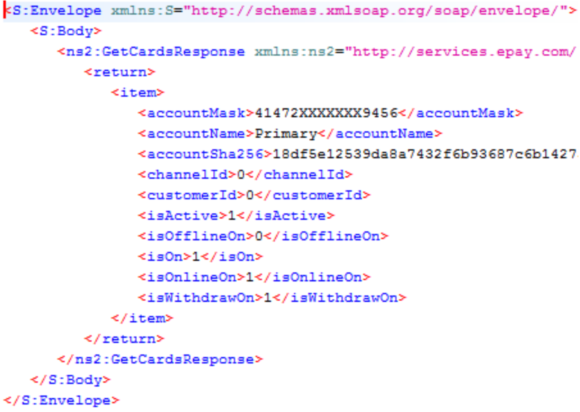
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap
   <soapenv:Header/>
   <soapenv:Body>
      <ser:UpdateCard>
         <CustomerID>11</CustomerID>
         <CardName>Primary</CardName>
         <CardSha256>18df5e12539da8a7432f6b93687c6b14273151dbc9c
         <IsOn>1</IsOn>
         <MaxOnlineValue>50000</MaxOnlineValue>
         <IsOnlineOn>1</IsOnlineOn>
         <MaxOfflineValue>20000</MaxOfflineValue>
         <IsOfflineOn>0</IsOfflineOn>
         <MaxWithdrawValue>30000</MaxWithdrawValue>
         <IsWithdrawOn>1</IsWithdrawOn>
      </ser:UpdateCard>
   </soapenv:Body>
</soapenv:Envelope>
```

“GetCards” List Request



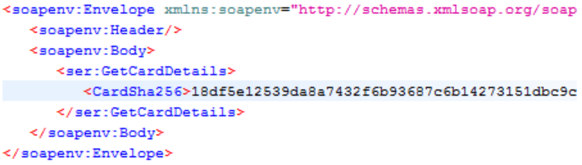
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap
   <soapenv:Header/>
   <soapenv:Body>
      <ser:GetCards>
         <userID>11</userID>
         <IsActive>1</IsActive>
      </ser:GetCards>
   </soapenv:Body>
</soapenv:Envelope>
```

“GetCards” List Response



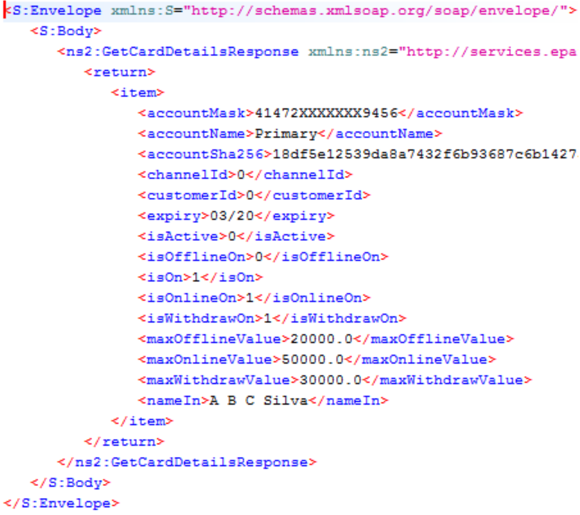
```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:GetCardsResponse xmlns:ns2="http://services.epay.com/
         <return>
            <item>
               <accountMask>41472XXXXXXX9456</accountMask>
               <accountName>Primary</accountName>
               <accountSha256>18df5e12539da8a7432f6b93687c6b1427
               <channelId>0</channelId>
               <customerId>0</customerId>
               <isActive>1</isActive>
               <isOfflineOn>0</isOfflineOn>
               <isOn>1</isOn>
               <isOnlineOn>1</isOnlineOn>
               <isWithdrawOn>1</isWithdrawOn>
            </item>
         </return>
      </ns2:GetCardsResponse>
   </S:Body>
</S:Envelope>
```

“GetCardDetails” Request



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap
   <soapenv:Header/>
   <soapenv:Body>
      <ser:GetCardDetails>
         <CardSha256>18df5e12539da8a7432f6b93687c6b14273151dbc9c
      </ser:GetCardDetails>
   </soapenv:Body>
</soapenv:Envelope>
```

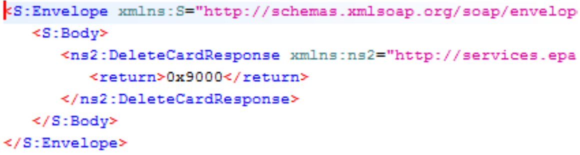
‘GetCardDetails” Response



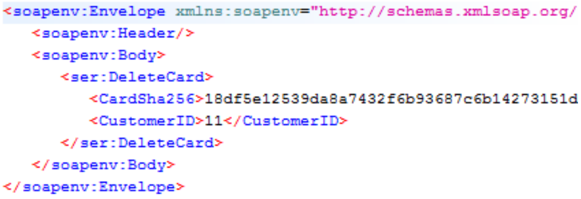
```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:GetCardDetailsResponse xmlns:ns2="http://services.epa
         <return>
            <item>
               <accountMask>41472XXXXXXX9456</accountMask>
               <accountName>Primary</accountName>
               <accountSha256>18df5e12539da8a7432f6b93687c6b1427
               <channelId>0</channelId>
               <customerId>0</customerId>
               <expiry>03/20</expiry>
               <isActive>0</isActive>
               <isOfflineOn>0</isOfflineOn>
               <isOn>1</isOn>
               <isOnlineOn>1</isOnlineOn>
               <isWithdrawOn>1</isWithdrawOn>
               <maxOfflineValue>20000.0</maxOfflineValue>
               <maxOnlineValue>50000.0</maxOnlineValue>
               <maxWithdrawValue>30000.0</maxWithdrawValue>
               <nameIn>A B C Silva</nameIn>
            </item>
         </return>
      </ns2:GetCardDetailsResponse>
   </S:Body>
</S:Envelope>
```

“DeleteCard” Request

“DeleteCard” Response

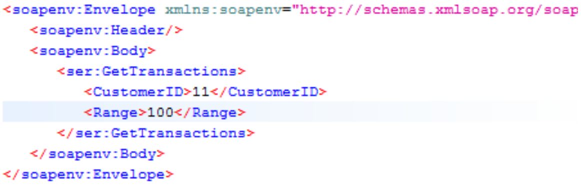


```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelop
   <S:Body>
      <ns2:DeleteCardResponse xmlns:ns2="http://services.epa
         <return>0x9000</return>
      </ns2:DeleteCardResponse>
   </S:Body>
</S:Envelope>
```



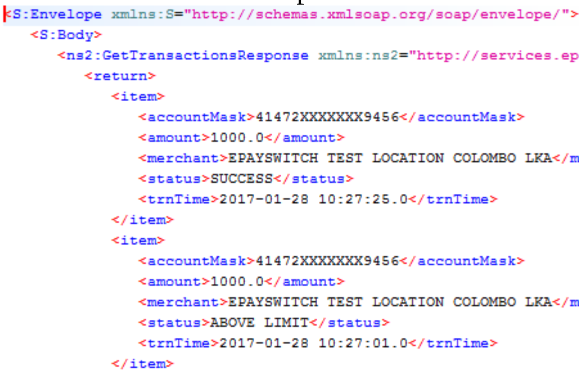
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/
   <soapenv:Header/>
   <soapenv:Body>
      <ser:DeleteCard>
         <CardSha256>18df5e12539da8a7432f6b93687c6b14273151d
         <CustomerID>11</CustomerID>
      </ser:DeleteCard>
   </soapenv:Body>
</soapenv:Envelope>
```

“GetTransactions” Request



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap
   <soapenv:Header/>
   <soapenv:Body>
      <ser:GetTransactions>
         <CustomerID>11</CustomerID>
         <Range>100</Range>
      </ser:GetTransactions>
   </soapenv:Body>
</soapenv:Envelope>
```

“GetTransactions” Response

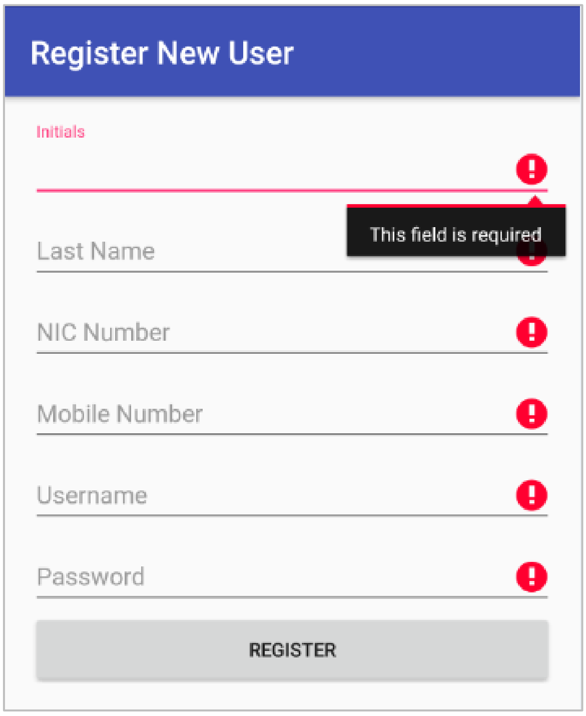


```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:GetTransactionsResponse xmlns:ns2="http://services.ep
         <return>
            <item>
               <accountMask>41472XXXXXXX9456</accountMask>
               <amount>1000.0</amount>
               <merchant>EPAYSWITCH TEST LOCATION COLOMBO LKA</m
               <status>SUCCESS</status>
               <trnTime>2017-01-28 10:27:25.0</trnTime>
            </item>
            <item>
               <accountMask>41472XXXXXXX9456</accountMask>
               <amount>1000.0</amount>
               <merchant>EPAYSWITCH TEST LOCATION COLOMBO LKA</m
               <status>ABOVE LIMIT</status>
               <trnTime>2017-01-28 10:27:01.0</trnTime>
            </item>
```

Appendix C – User Interfaces

Mobile Application

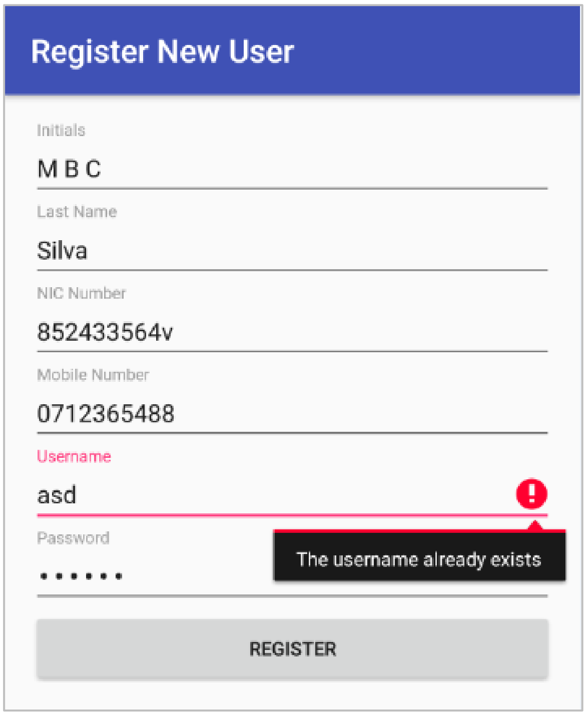
Register All Fields Required



Register New User

This field is required

Register Username Exists



Register New User

Initials: M B C

Last Name: Silva

NIC Number: 852433564v

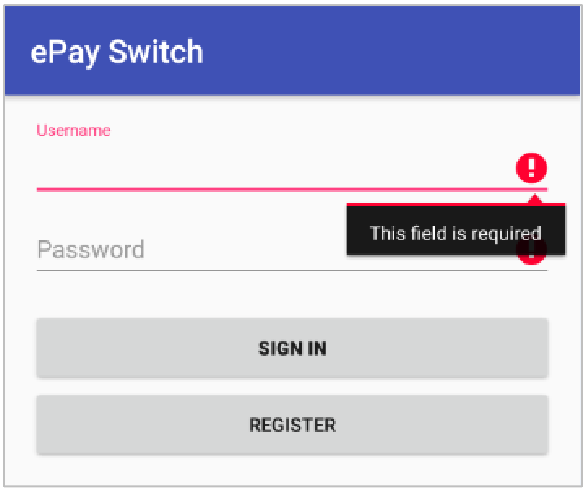
Mobile Number: 0712365488

Username: asd

Password: ••••••

The username already exists

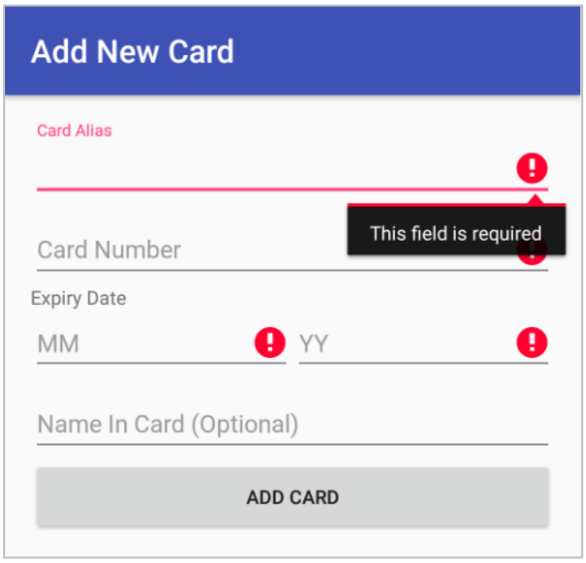
Sign In All Fields Required



ePay Switch

This field is required

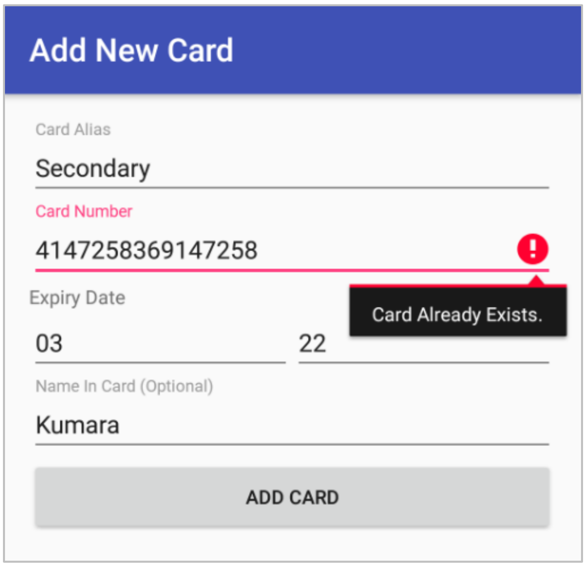
Add New Card All Fields Required



Add New Card

This field is required

Add New Card Already Exists



Add New Card

Card Alias
Secondary

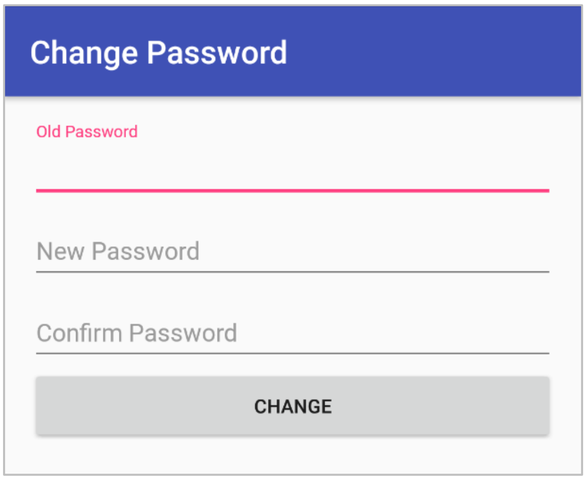
Card Number
4147258369147258

Card Already Exists.

Expiry Date
03 22

Name In Card (Optional)
Kumara

Password Change



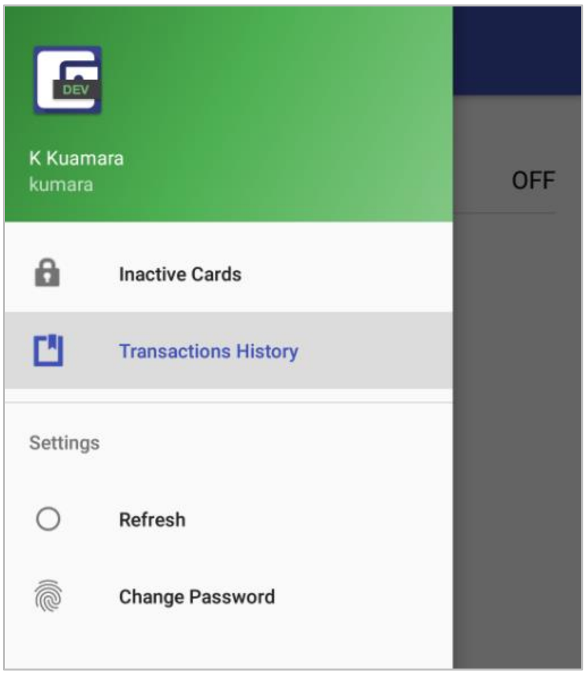
Change Password

Old Password

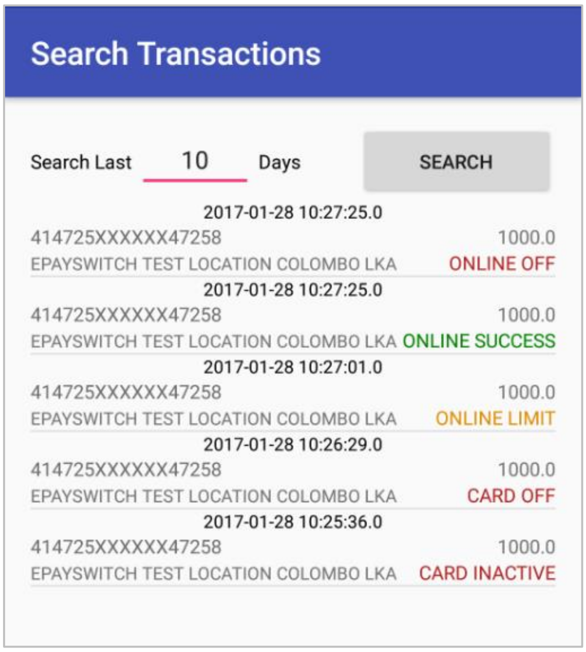
New Password

Confirm Password

Menu Drawer



Transaction Summary



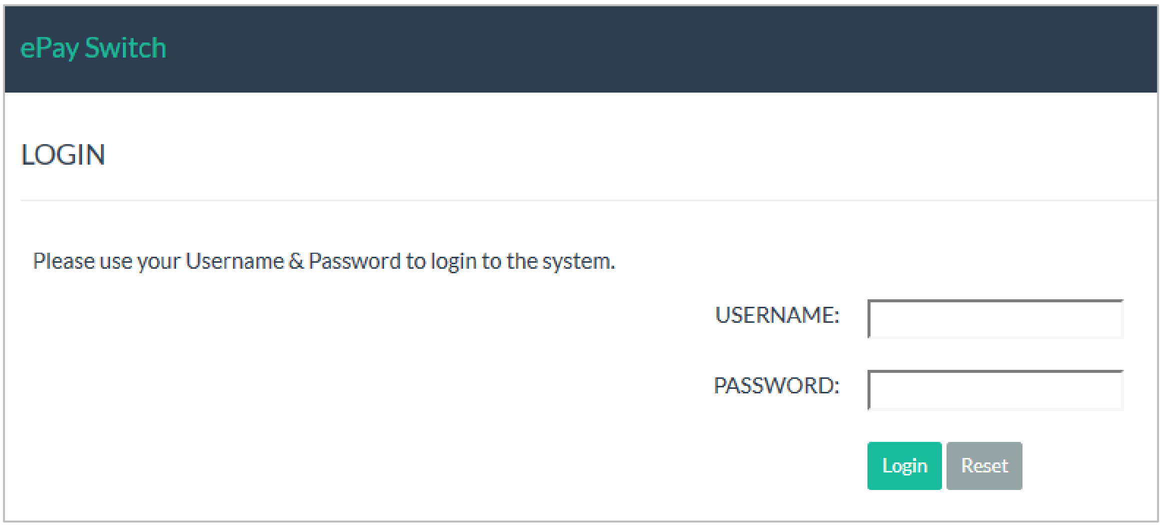
Search Transactions

Search Last 10 Days

```
          2017-01-28 10:27:25.0
414725XXXXXX47258                          1000.0
EPAYSWITCH TEST LOCATION COLOMBO LKA   ONLINE OFF
          2017-01-28 10:27:25.0
414725XXXXXX47258                          1000.0
EPAYSWITCH TEST LOCATION COLOMBO LKA ONLINE SUCCESS
          2017-01-28 10:27:01.0
414725XXXXXX47258                          1000.0
EPAYSWITCH TEST LOCATION COLOMBO LKA  ONLINE LIMIT
          2017-01-28 10:26:29.0
414725XXXXXX47258                          1000.0
EPAYSWITCH TEST LOCATION COLOMBO LKA     CARD OFF
          2017-01-28 10:25:36.0
414725XXXXXX47258                          1000.0
EPAYSWITCH TEST LOCATION COLOMBO LKA CARD INACTIVE
```

Customer and Cards Manager

User Login



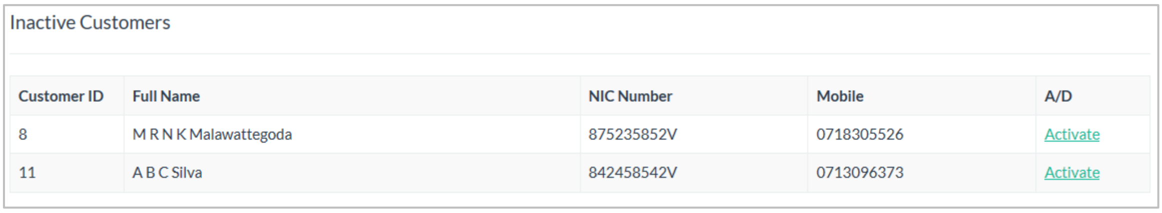
LOGIN

Please use your Username & Password to login to the system.

USERNAME:

PASSWORD:

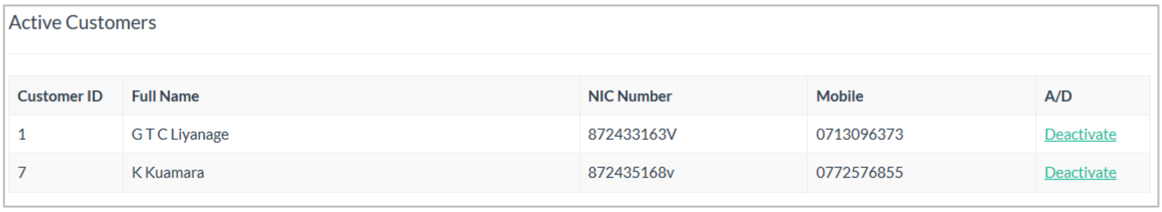
Inactive Customers (Activate Customers)



Inactive Customers

Customer ID	Full Name	NIC Number	Mobile	A/D
8	M R N K Malawattegoda	875235852V	0718305526	Activate
11	A B C Silva	842458542V	0713096373	Activate

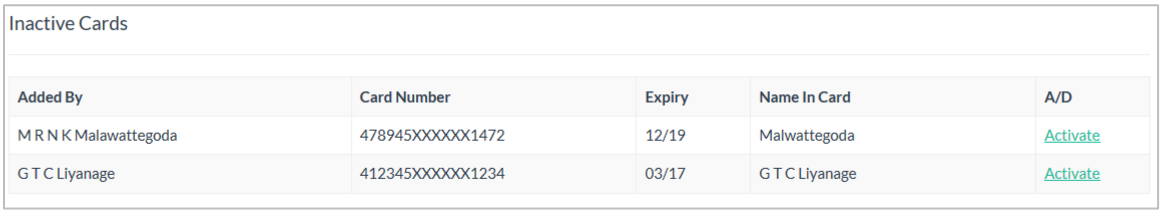
Active Customers (Deactivate Customers)



Active Customers

Customer ID	Full Name	NIC Number	Mobile	A/D
1	G T C Liyanage	872433163V	0713096373	Deactivate
7	K Kuamara	872435168v	0772576855	Deactivate

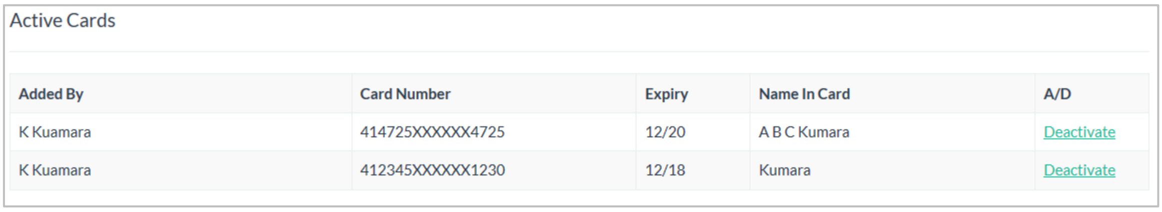
Inactive Cards (Activate Cards)



Inactive Cards

Added By	Card Number	Expiry	Name In Card	A/D
M R N K Malawattegoda	478945XXXXXX1472	12/19	Malwattegoda	Activate
G T C Liyanage	412345XXXXXX1234	03/17	G T C Liyanage	Activate

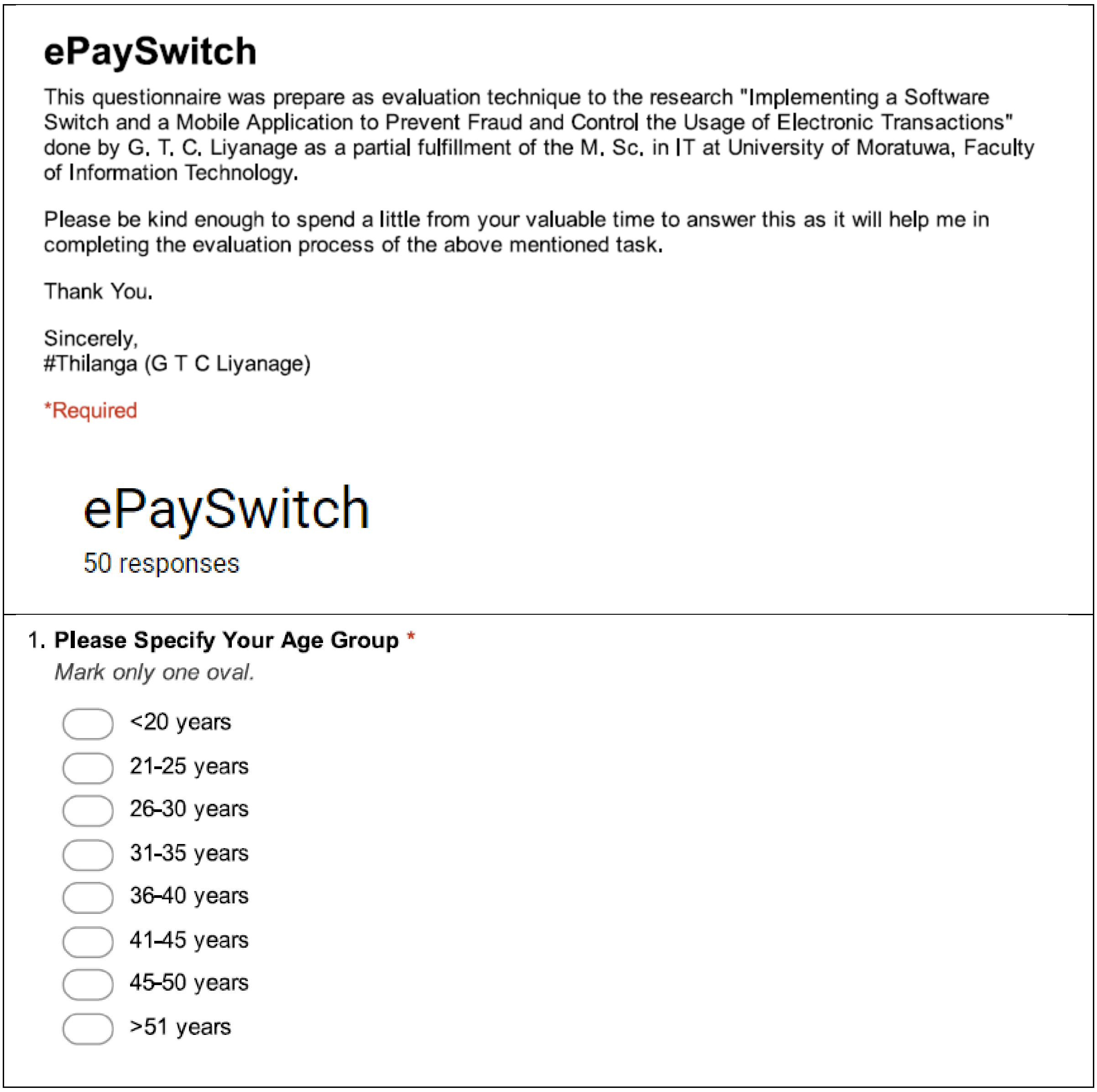
Active Cards (Deactivate Cards)



Active Cards

Added By	Card Number	Expiry	Name In Card	A/D
K Kuamara	414725XXXXXX4725	12/20	A B C Kumara	Deactivate
K Kuamara	412345XXXXXX1230	12/18	Kumara	Deactivate

Appendix D – Questionnaire and Responses



ePaySwitch

This questionnaire was prepare as evaluation technique to the research "Implementing a Software Switch and a Mobile Application to Prevent Fraud and Control the Usage of Electronic Transactions" done by G. T. C. Liyanage as a partial fulfillment of the M. Sc. in IT at University of Moratuwa, Faculty of Information Technology.

Please be kind enough to spend a little from your valuable time to answer this as it will help me in completing the evaluation process of the above mentioned task.

Thank You.

Sincerely,
#Thilanga (G T C Liyanage)

*Required

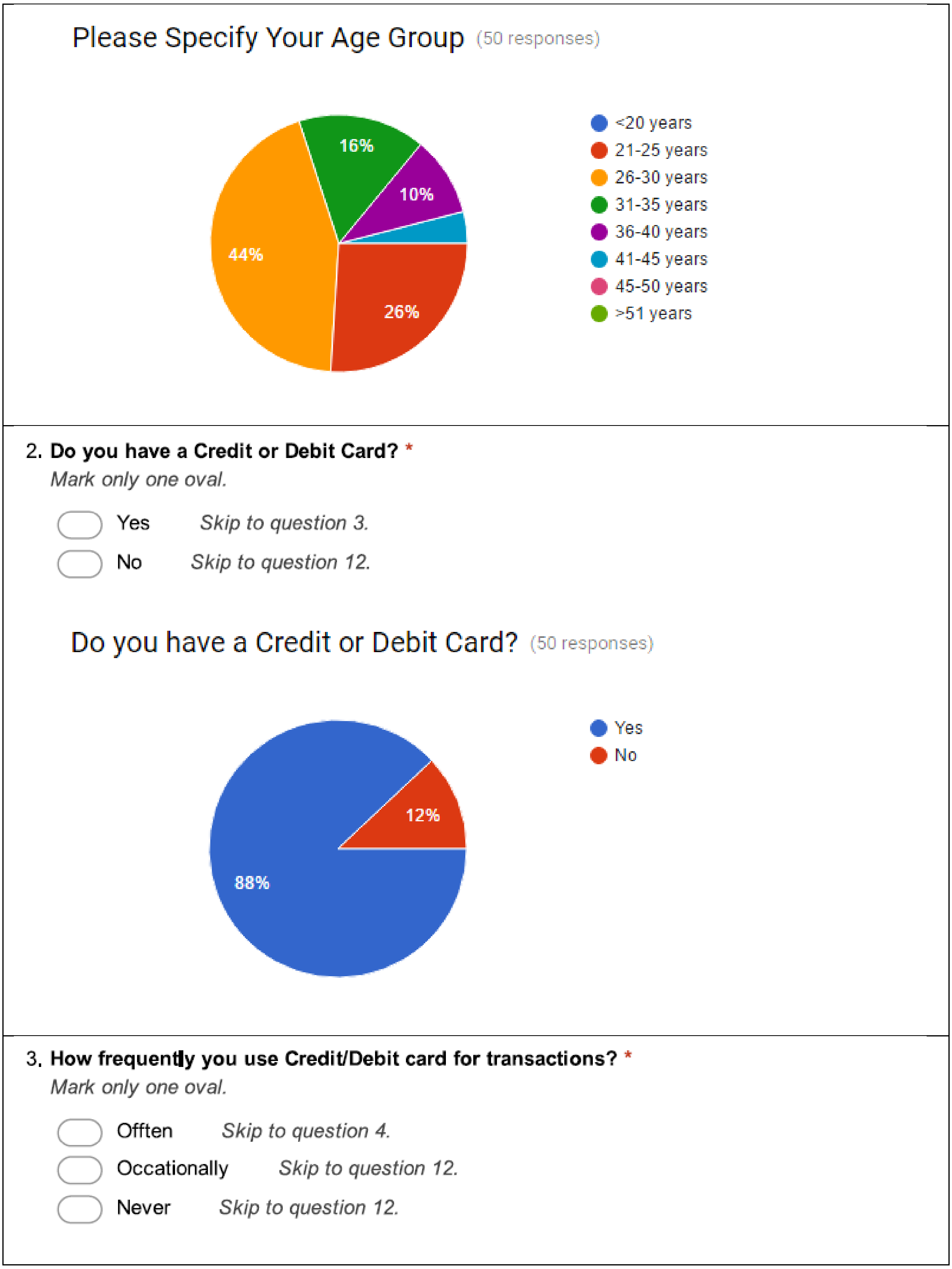
ePaySwitch

50 responses

1. **Please Specify Your Age Group** *

Mark only one oval.

○ <20 years
○ 21-25 years
○ 26-30 years
○ 31-35 years
○ 36-40 years
○ 41-45 years
○ 45-50 years
○ >51 years



Please Specify Your Age Group (50 responses)

2. **Do you have a Credit or Debit Card?** *

Mark only one oval.

○ Yes *Skip to question 3.*

○ No *Skip to question 12.*

Do you have a Credit or Debit Card? (50 responses)

3. **How frequently you use Credit/Debit card for transactions?** *

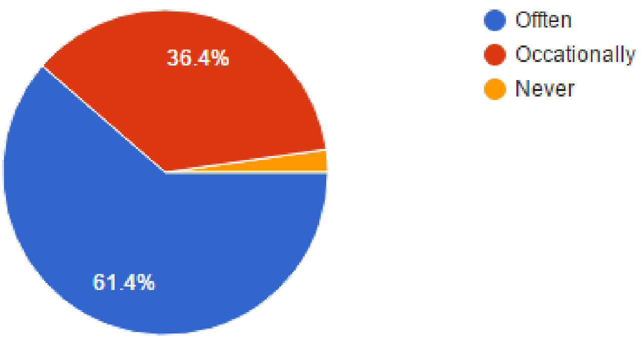
Mark only one oval.

○ Offten *Skip to question 4.*

○ Occationally *Skip to question 12.*

○ Never *Skip to question 12.*

How frequently you use Credit/Debit card for transactions?
(44 responses)



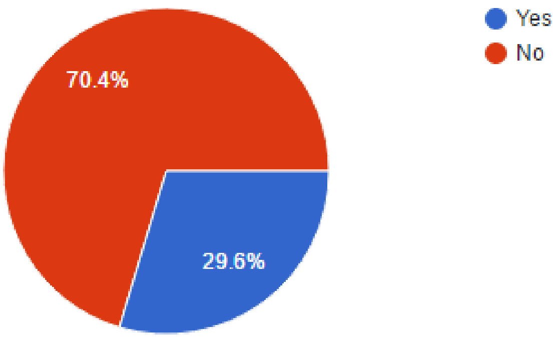
4. **Have you ever lost your card?** *

Mark only one oval.

○ Yes

○ No

Have you ever lost your card? (27 responses)



5. **Have you ever thought someone will steel your card & perform transaction before you know?** *

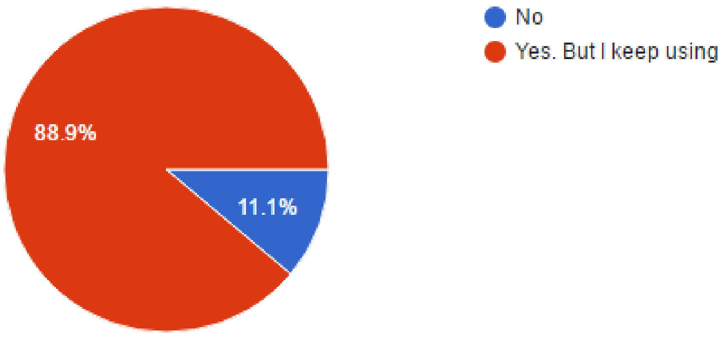
Mark only one oval.

○ No

○ Yes, But I keep using

Have you ever thought someone will steel your card & perform transaction before you know?

(27 responses)



6. **Do you aware of transaction amounts you perform via cards?** *

Mark only one oval.

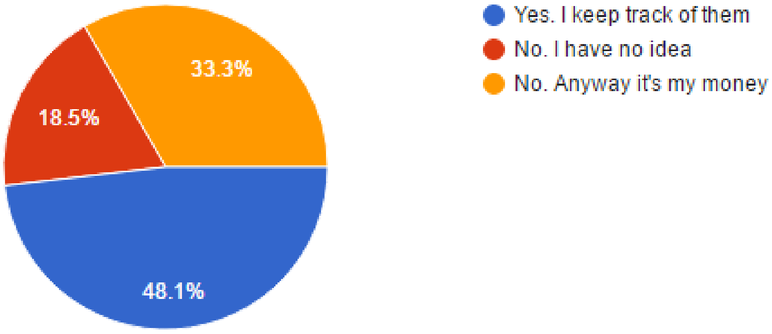
○ Yes. I keep track of them

○ No. I have no idea

○ No. Anyway it's my money

Do you aware of transaction amounts you perform via cards?

(27 responses)



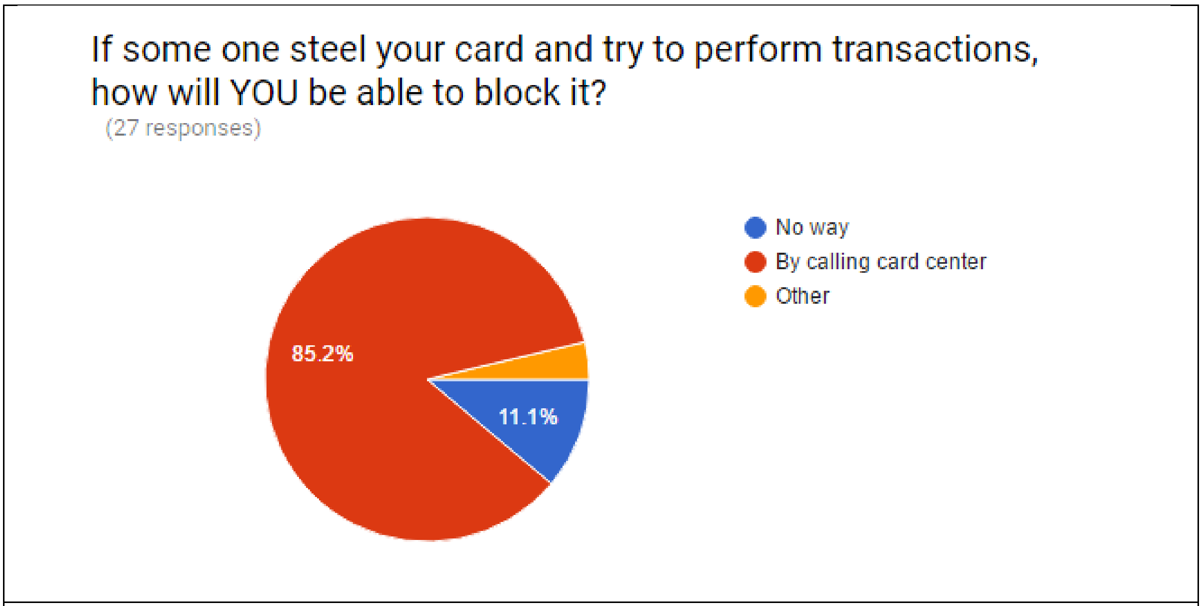
7. **If some one steel your card and try to perform transactions, how will YOU be able to block it?** *

Mark only one oval.

○ No way

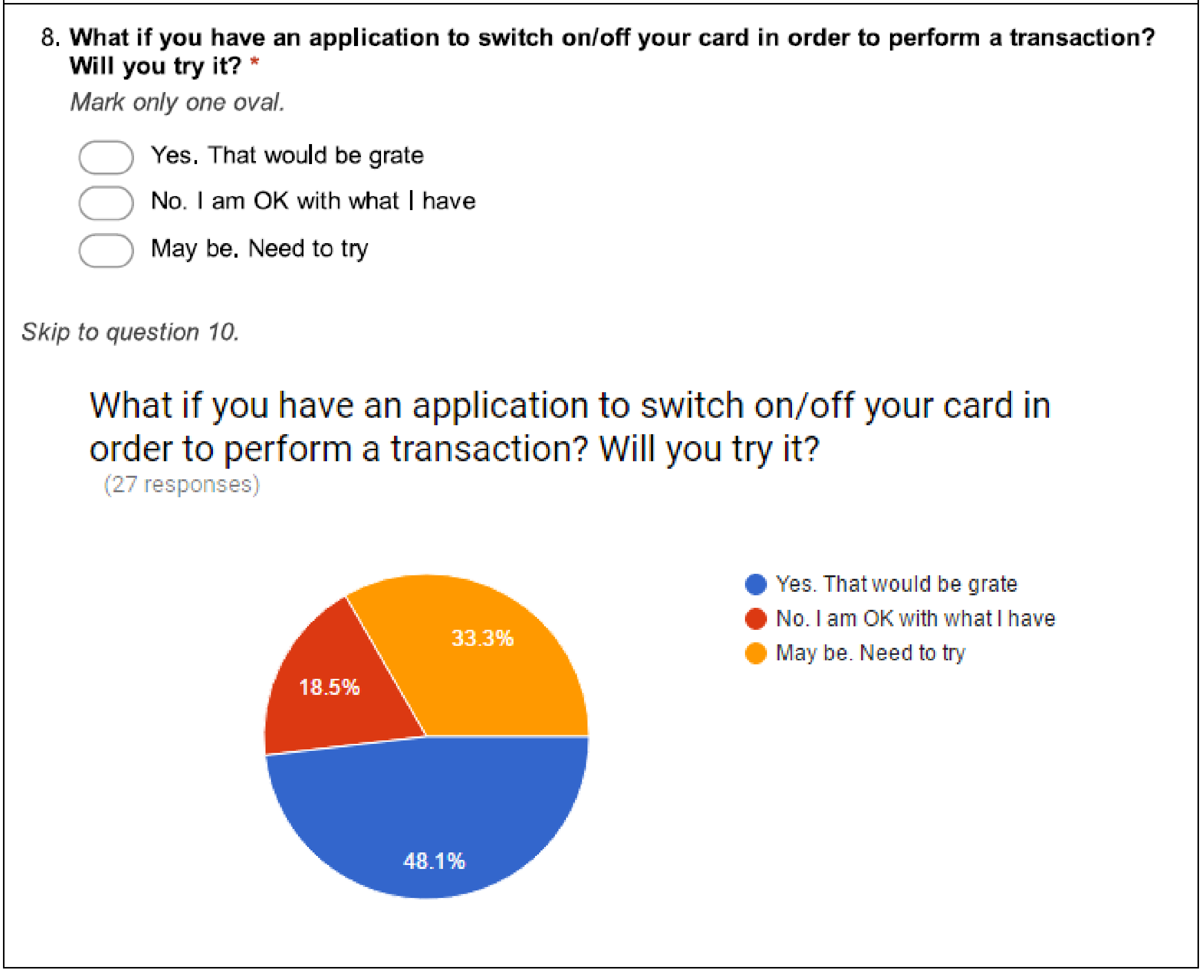
○ By calling card center

○ Other: ____________________



If some one steel your card and try to perform transactions, how will YOU be able to block it?

(27 responses)



8. **What if you have an application to switch on/off your card in order to perform a transaction? Will you try it?** *

Mark only one oval.

○ Yes. That would be grate

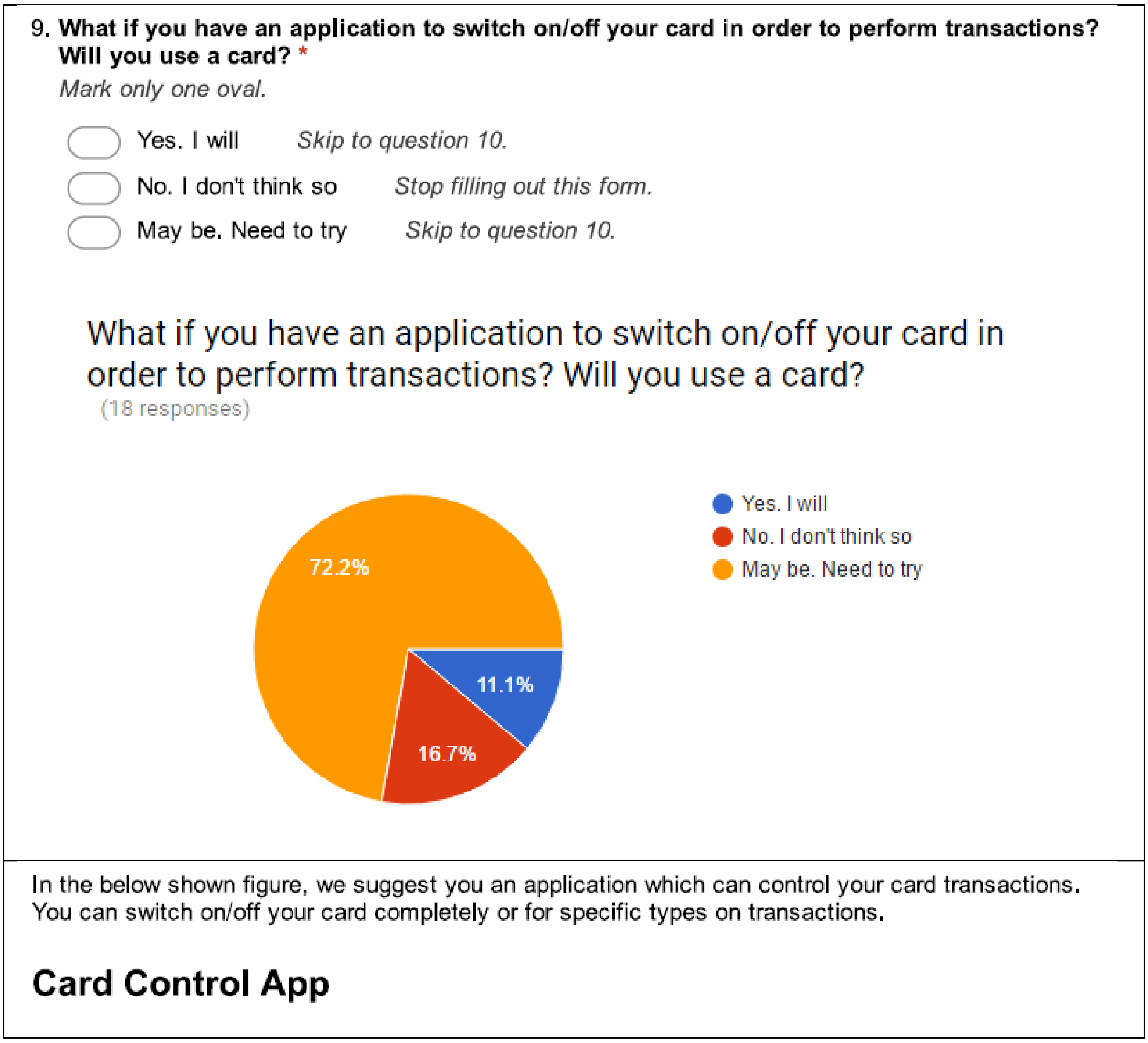
○ No. I am OK with what I have

○ May be. Need to try

Skip to question 10.

What if you have an application to switch on/off your card in order to perform a transaction? Will you try it?

(27 responses)



9. **What if you have an application to switch on/off your card in order to perform transactions? Will you use a card?** *

Mark only one oval.

○ Yes. I will *Skip to question 10.*

○ No. I don't think so *Stop filling out this form.*

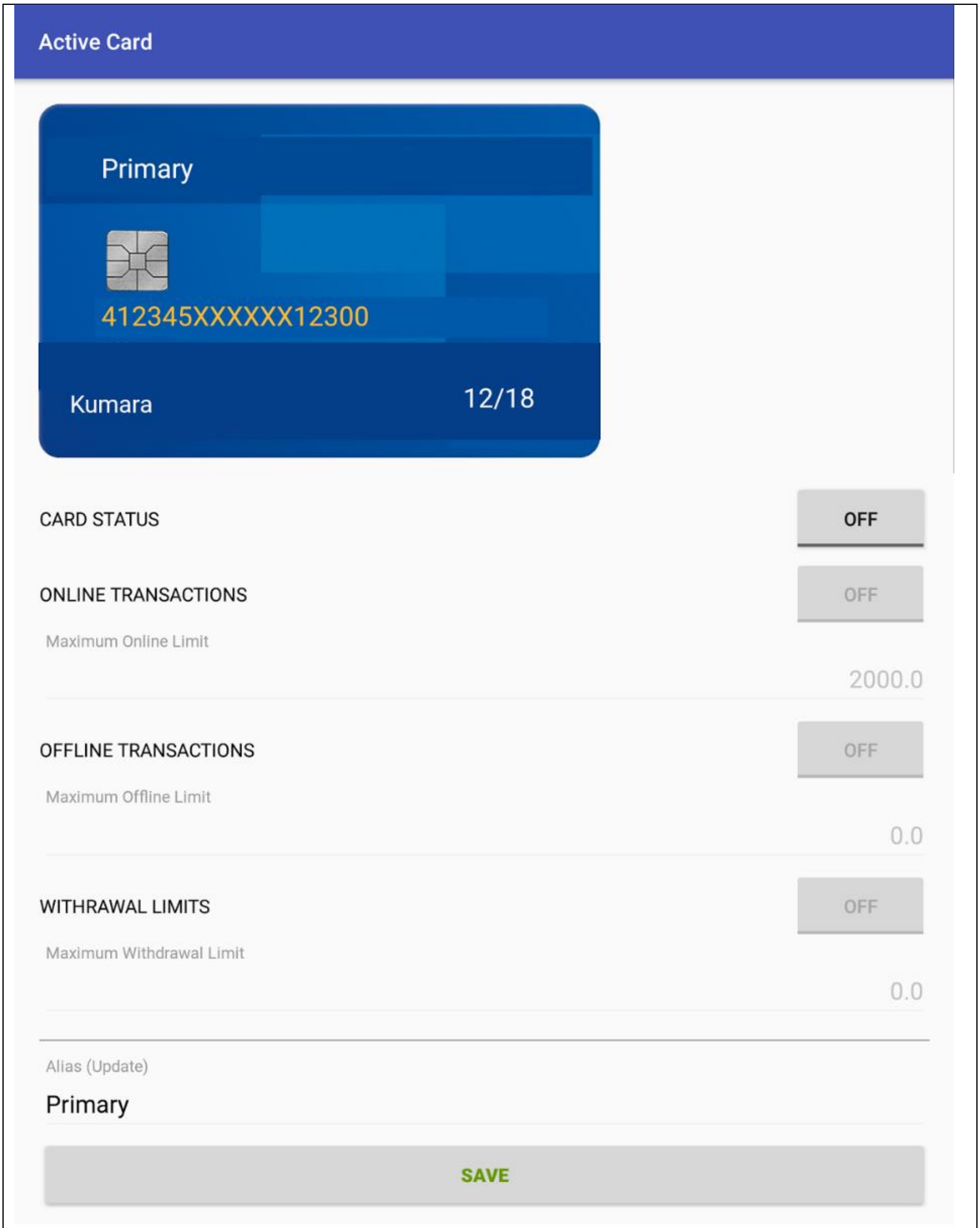
○ May be. Need to try *Skip to question 10.*

What if you have an application to switch on/off your card in order to perform transactions? Will you use a card?

(18 responses)

In the below shown figure, we suggest you an application which can control your card transactions. You can switch on/off your card completely or for specific types on transactions.

Card Control App


Active Card

Primary

412345XXXXXX12300

Kumara

12/18

CARD STATUS OFF

ONLINE TRANSACTIONS OFF

Maximum Online Limit

2000.0

OFFLINE TRANSACTIONS OFF

Maximum Offline Limit

0.0

WITHDRAWAL LIMITS OFF

Maximum Withdrawal Limit

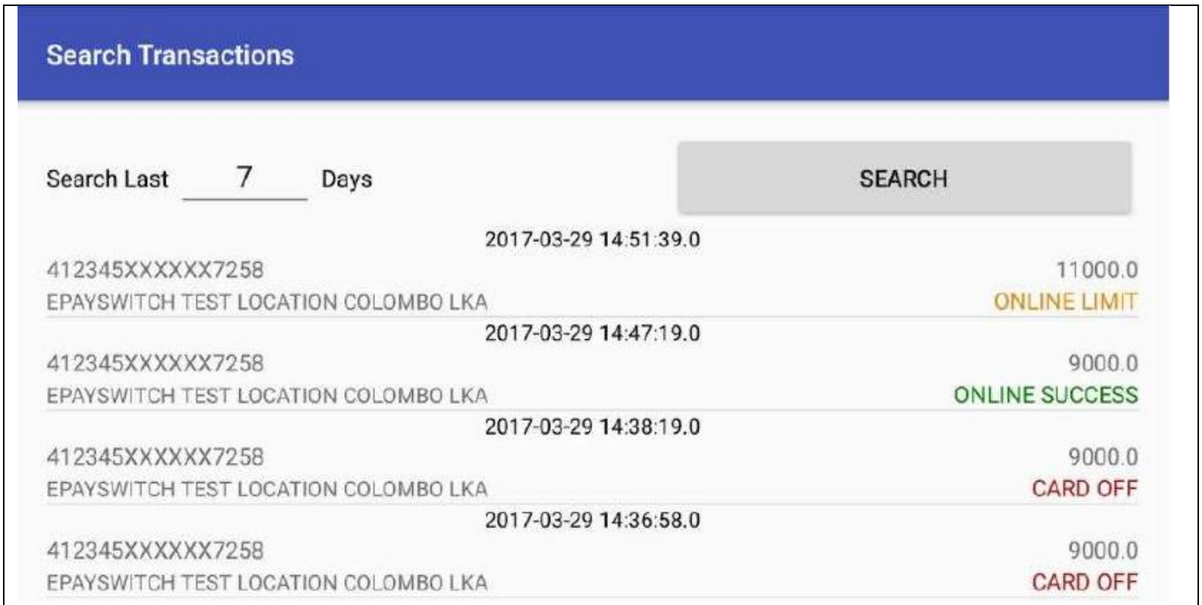
0.0

Alias (Update)

Primary

SAVE

Transactions History



Search Transactions

Search Last 7 Days SEARCH

2017-03-29 14:51:39.0
412345XXXXXX7258 11000.0
EPAYSWITCH TEST LOCATION COLOMBO LKA ONLINE LIMIT

2017-03-29 14:47:19.0
412345XXXXXX7258 9000.0
EPAYSWITCH TEST LOCATION COLOMBO LKA ONLINE SUCCESS

2017-03-29 14:38:19.0
412345XXXXXX7258 9000.0
EPAYSWITCH TEST LOCATION COLOMBO LKA CARD OFF

2017-03-29 14:36:58.0
412345XXXXXX7258 9000.0
EPAYSWITCH TEST LOCATION COLOMBO LKA CARD OFF

10. **If you have given a card control application as in image, do you think you will have a more control when your card lost/stolen and its' transactions?** *

Mark only one oval.

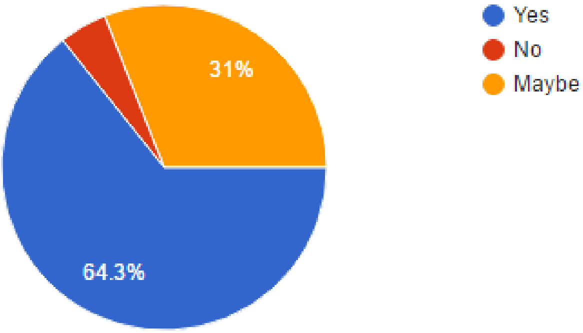
○ Yes

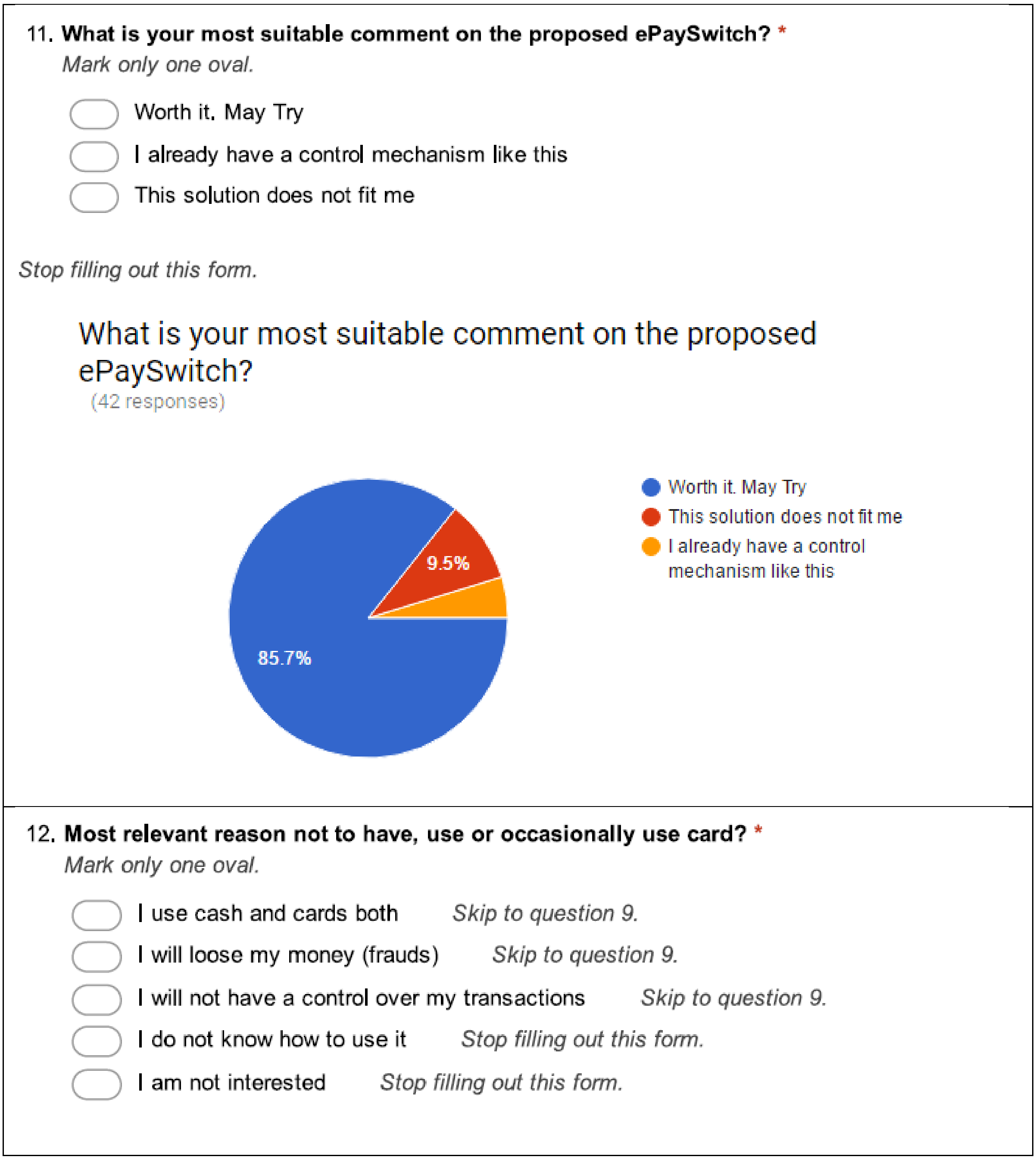
○ No

○ Maybe

If you have given a card control application as in image, do you think you will have a more control when your card lost/stolen and its' transactions?

(42 responses)





11. **What is your most suitable comment on the proposed ePaySwitch?** *

Mark only one oval.

○ Worth it. May Try

○ I already have a control mechanism like this

○ This solution does not fit me

Stop filling out this form.

What is your most suitable comment on the proposed ePaySwitch?

(42 responses)

9.5%
85.7%
● Worth it. May Try
● This solution does not fit me
● I already have a control mechanism like this

12. **Most relevant reason not to have, use or occasionally use card?** *

Mark only one oval.

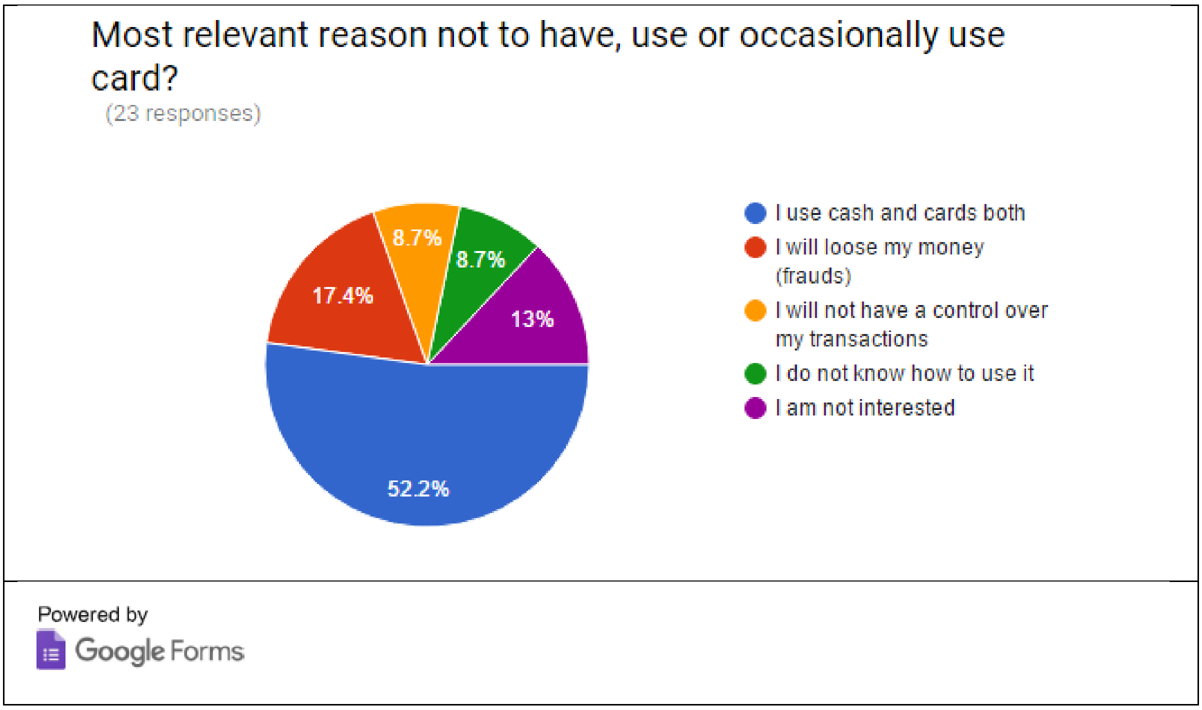
○ I use cash and cards both *Skip to question 9.*

○ I will loose my money (frauds) *Skip to question 9.*

○ I will not have a control over my transactions *Skip to question 9.*

○ I do not know how to use it *Stop filling out this form.*

○ I am not interested *Stop filling out this form.*



Most relevant reason not to have, use or occasionally use card?

(23 responses)

Powered by Google Forms