University of Fort Hare

Together in Excellence

DEVELOPMENT OF AN M-PAYMENT SYSTEM PROTOTYPE FOR A MARGINALIZED REGION (DWESA CASE STUDY)

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF THE DEGREE

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

UNIVERSITY OF FORT HARE

BY

HANDSOME C. MPOFU

DECEMBER 2011

Acknowledgements

I would like to express my sincere gratitude to the Telkom Center of Excellence and all its industry partners for the financial support and academic activities that made this project a success.

I extend my thanks to the Computer Science department: my research mates, staff members and especially my supervisor, Dr. M. Thinyane, for his guidance. I offer a special thanks to my friends Chakes, Kaizer, Mpiza, Siseko and many more for brightening my social life.

I dedicate this work to my special daughter, Lerato and my late friend Chakalisa Ndlovu.

University of Fort Hare
Together in Excellence

PLAGIARISM DECLARATION

1. I know that plagiarism means taking and using the ideas, writings, works or inventions of another as if they were one's own. I know that plagiarism not only includes verbatim copying, but also the extensive use of another person's ideas without proper acknowledgement (which includes the proper use of quotation marks). I know that plagiarism covers the sort of material found in textual sources and from the internet.

2. I acknowledge and understand that plagiarism is wrong.

3. I understand that my research must be accurately referenced. I have followed the rules and conventions concerning referencing, citation and the use of quotations as set out in the Departmental Guide.

4. This assignment is my own work. or I acknowledge that copying someone else's research or part of it, is wrong, and that submitting identical work to others constitutes form of plagiarism.

5. I have not allowed, nor will I in the future allow, anyone to copy my work with the intention of passing it off as their own work.

Surname and Initials……. Signature……

Date…….

Abstract

Modern ICT innovations boast of major technological advancements which aim to improve services and social interaction in all societies. We live in the information age, where so many of these technologies play a major role in business operations, transactions and communication. As the complexity of these innovations intensifies, they create what is known as a digital divide, where people from marginalized regions have limited or no access to these ICTs. The ICT for Development framework gave birth to the development of innovations which target marginalized regions. This project follows the footprints of The ICT for Development and offers the marginalized community of Dwesa, a rural community in the Eastern Cape Province of South Africa, a mobile device application meant to improve the banking services for and electricity acquisition by this rural community. The M-Payment System is a financial application aiming to reduce travelling costs for rural people to town, by making banking and purchasing electricity a local transaction using mobile devices. It is a standalone system that is user defined, easy to navigate interface with system requirements meeting the needs of the technologically and services-deprived community of Dwesa. It has the characteristics of many systems introduced for the same purpose but also is unique in its non-integration into existing traditional financial services, using the system database in a local server as a bank repository. The system design is modeled around the spiral software development life-cycle for future improvements and enhancements, and uses Java Micro Edition, WAMP server 2.0 technologies as part of the migration from electronic commerce to mobile commerce.

Table of Contents

List of Figures

List of Tables

List of Acronyms

3G ..... Third Generations
BREW..... Binary Runtime Environment for Wireless
CDC ..... Connected Device Configuration
CLDC..... Connected Limited Device Configuration
EB ..... Electricity Buying
GAP ..... GSM Access Point
GPRS ..... General Packet Radio Service
GSM..... Global System for Mobile Communication
GUI ..... Graphical User Interface
HLR ..... Home Location Register
ICT..... Information and Communication Technology
ICT4D ..... Information and Communication Technology for Development
IDE..... Integrated Development Environments
IP..... Internet Protocol
J2ME..... Java 2 platform, Micro Edition
MIDP ..... Mobile Information Device Profile
PHP ..... Hypertext Preprocessor
SMS ..... Short Messaging Service
SMSC..... Short Message Service Center
TDMA..... Time Division Multiple Access
VMT ..... Virtual Money Transfer
WAP ..... Wireless Application Protocol
WMA ..... Wireless messaging API

CHAPTER 1: INTRODUCTION

Information Technology (IT) has seen a rapid change in fortune for many organizations and communities at large. Businesses now require some form of IT to function everyday and be sustainable. Although the influence and effects of IT are felt almost everywhere, there are still marginalized regions that barely know of the existence of some technologies and their benefits. Information and Communication Technology for Development (ICT4D) is a framework that has resulted in the technological advancement of isolated rural regions through innovations that target rural inhabitants in order to improve their socio-economic well being (Mpofu et al., 2010). This chapter introduces a project which stems from this framework and is meant to provide rural communities with a mobile device application for banking services and the remote purchasing of electricity. The structure of the entire thesis is also outlined in this chapter.

1.1 Background

In the Global digital information age, access to information and communication technology offers new opportunities through access to information and economic development strategies. At present, business ideas and strategies take advantage of the penetration of some technologies into their target markets. Internet, for example, is one technology that has seen a major growth and penetration into global markets. It is the most common means by which different entities in the business world stay in touch with their clients, and is used as a marketing strategy for the entities' business engagements.

Recently, there has been a major shift from wired network to wireless network based technologies. In developing countries, internet as a wired technology did not have the same penetration rate as wireless mobile device technology. The introduction of mobile device technology for communication, information retrieval and mobile commerce meant that more people had access to a technology that generally performs similar tasks to fixed networks, but has an advantage of mobility and affordability. The wireless technology ensured greater network coverage beyond the traditional wired network rollout.

With the global economy aiming to raise the international standards of living of people from different races, geographic locations and cultures, among other attributes, the mobile device presents a technology that has seen many applications and solutions developed and launched for the general public. Although these technologies serve their purpose for ICT-aware clients, most people from marginalized regions have no knowledge of the use of these technologies; hence the latest technologies target or are useful to urban users. Because of this, there is digital exclusion of rural communities as they struggle to cope with the rapid technological revolution, which does not ease the harsh conditions in which they exist. This digital exclusion is termed the *digital divide*, and offers developers a challenge to invent sustainable applications for marginalized and poor communities around the world.

Marginalized communities in South Africa suffer from almost the same impoverishing factors. They are characterized by poor infrastructure, isolation, poverty and low literacy levels. Business prospects and projects are not feasible or profitable for them, as investors fear major losses in rural-based business turnover. However, these communities are in need of services that can alleviate poverty and lessen the hardships they face on a daily basis. These services should be sustainable and accessible, at any time, at affordable rates. Wireless technology, together with mobile devices, offers a platform that can be used to reach out to those in marginalized regions and, after a socio-economic evaluation of the community, develop solutions in order to solve some of the concerns expressed by the community. This concept is a framework known as ICT for Development (IDT4D) which aims to initiate innovations for marginalized communities to improve their socio-economic well-being.

In this project, the ICT4D framework was used to pilot a mobile device application for the sending of money to a person without a bank account, in a rural area, and for a person based in a rural area to buy electricity from an entrepreneur in town. Traditional banks are located in urban areas, making it costly for people from marginalized regions to access such traditional ICTs. This project uses the existing infrastructure to provide a low cost mobile device solution that has two modules: the money transfer and electricity buying modules.

1.2 Problem statement

Dwesa is a completely rural region in the Eastern Cape that is characterized by most of the attributes of a rural reality in South Africa. The region has a high level of unemployment, low literacy levels, poor infrastructure, lack of common or established government structures and no business prospects beyond backyard supermarkets. Since the essential infrastructure, like banks and shops for basic commodities, are all in urban areas, the people staying in Dwesa have to travel to town to access these. The cost of travelling to and from town is R60. Let us take an example of a person who has R100 deposited into his account and has to use R60 to travel to town just for a withdrawal. Similarly, a person has to use R60 on transport to town to purchase electricity worth less than R60. It becomes expensive for these people to travel for these services, considering their socio-economic status. The purpose of this study is to develop a user-driven mobile application, M-Payment System, which is meant to convert banking and the buying of electricity in Dwesa into a locally based transaction. The M-Payment system is a direct consequence of a socio economic survey done in the region, where the villagers expressed their need for banking infrastructure as well as an easy way to buy electricity without commuting. It is a community driven initiative hence the term *user-driven* application. The study should address the following questions:

- How will the M-Payment system integrate into the existing traditional banking infrastructure?
- What technologies are going to be used in implementing this application?
- What are the specification requirements in terms of usability?
- What methodology is going to be used to meet the system specification requirements?
- Who are the players in this system and how will using this system benefit them?
- How will the system ease the socio-economic well-being of the Dwesa community?

- What are the outlined tests to make sure the system meets the user requirements?

- What security measures are going to be taken to ensure access to and manipulation of data is controlled, as this is a financial system.

1.3 Research Objectives

The objective of this project is to develop and deploy a mobile device prototype that meets certain usability and user acceptance metrics. The metrics are classified as follows:

1.3.1 Usability Metrics

- *Effectiveness*
  - Prototype should enable users to complete desired tasks with little or no assistance.
- *Efficiency*
  - Prototype should enable users to complete tasks and achieve desired results in record time.

1.3.2 User-acceptance Metrics

- *Usefulness*
  - Prototype should address the needs of the target community. It should satisfy the user needs and be tailored to solve community problems.
- *Interest the user*
  - Prototype must invigorate an interest in the intended users for it to be sustainable.

1.3.3 Other Metrics

- *User friendly*

- Prototype should offer a user friendly interface and navigation structure for ease of use.

- *Cost- effective*
    - Prototype should not be expensive to use.

- *Embraces data integrity*
    - Data should be protected from manipulation by unauthorized persons.
- *Security*
    - Prototype should possess a certain level of security, as financial transactions are susceptible to fraudulent operations.
- *Localization of the system*
    - Prototype should have a language option, which includes the dominant language of the Dwesa community.

1.4 Methodology

The idea of this project emanated from the Baseline Study conducted by prominent researchers from Fort Hare and Rhodes Universities, in the marginalized area of Dwesa. The study set out to assess the socio-economic status of the community at large over a period of time. The research methodology is outlined as follows:

- *Questionnaires and informal interviews* – These were used to come up with conclusive evidence of the status of the community. This project focused on solving a part of the results obtained, focusing on the unavailability of banking infrastructure and the buying of electricity.
- *Literature review* – This enabled the assessment of already existing case studies and methodologies in an attempt to customize some of the ideas to the problems identified in Dwesa.

- *Planning* – This is the preliminary investigation and project commencement stage. It involves assessing the system requirements according to the information gathered from users.
- *Prototype Designing* – The prototype is developed or structured using the outlined plan and guided by the requirements elicitation. This stage requires the use of various technologies:
    - *Java technology* - To develop the client application.
    - *PHP Scripting language* –To develop the middleware.
    - *WAMP 2.0 server* – To implement the server side functionalities.
- *Simulation and Evaluation* - The prototype was simulated and evaluated on usability, user acceptance and other performance metrics.

1.5 Dissertation Structure

The remainder of the dissertation is structured as follows:

Chapter Two: Literature Review

This chapter explores the theory surrounding the research topic. It looks at the principles of Information for Communication and Technologies for Development (ICT4D) and the various network types among other sections.

Chapter Three: Mobile Computing Technologies

This chapter discusses the various technologies used in the development of mobile device applications.

Chapter Four: Service Description and Design

This chapter gives an overview of components of the M-Payments system. It discusses the functional as well as the non-functional attributes.

Chapter Five: Implementation of the Money Payment Support Components

The M-Payment system which has been developed has two modules: the money transfer and electricity buying modules. This chapter discusses the implementation of the system components that are common between the two system modules, e.g. the login function.

Chapter Six: Money Transfer Module Implementation

This chapter focuses on the implementation of the component used to send money to a person based in the rural areas, without using a bank.

Chapter Seven: Electricity Buying Module Implementation

This chapter discusses the implementation of the electricity buying module used to buy electricity from an urban entrepreneur who uses the same system.

Chapter Eight: Experiments and Results

This chapter discusses the different experiments that were carried out, so as to necessitate the presentation of the outcomes, in graphical form, as results of the experiments.

Chapter Nine: Discussion and Conclusion

This chapter discusses the project outcomes, in order to ascertain whether the research objectives have been met or not. It lists the successes and failures of this project, and possible future extensions to the project. The chapter concludes the dissertation.

CHAPTER 2: LITERATURE REVIEW

This chapter reviews the literature and related work on mobile computing. At the core of alleviating poverty in marginalized regions in the developing world is the concept of introducing Information and Communication Technology (ICT) to targeted regions. The readiness of rural communities for information exchange technologies poses a big challenge to introducing ICT related solutions to them. This chapter looks at, but is not limited to, Information and Communication Technologies for Development, wireless networks, the Global System for Mobile communication (GSM) technology and other work related to this research.

2.1 Information and Communication Technologies for Development (ICT4D)

Information and knowledge exchange technologies equip people with means to do business and understand their social and political values. Rural areas in various regions of Africa are under-developed in terms of infrastructure, have under-utilized resources that include agricultural land, as well as natural tourist attractions and many more.

Marginalized rural areas lack information acquiring and sharing techniques and technologies which are best suited to their needs. Rural people are associated with low levels of education, high illiteracy levels and poor infrastructure such as roads, poor or no proper government offices and communication structures (Banerjee et al., 2004). Impoverishing forces are numerous and diversified in rural areas; they tend to inhibit the diffusion of technology, information, supply of inputs, credit and extension services as well as market integration. The fusion of these factors and other cultural and macro-economic problems, such as foreign debts and currency depreciation subjects the rural poor majority to chronic poverty (Banerjee et al., 2004).

ICT4D has seen several organizations embarking on a mission to use ICT to fight poverty and bring about development in marginalized rural regions (Tella et al., 2010; Zhen-Wei et al., 2006). If implemented properly, ICTs are meant to integrate rural economies into the global

economy by promoting better service delivery, beyond urban city centers. It is also meant to empower people from all walks of life to seek, evaluate, use and create information effectively to achieve their personal, social, occupational and educational goals (Ľudovít et al., 2010). ICT4D serves to lessen the gap between the information haves and have-nots, which is referred to as the digital divide.

2.1.1 Digital Divide

The digital divide refers to the gap between those people who have access to digital technologies and information on the Internet, and those who do not. Figure 1 shows global internet usage statistics for the year 1999, with the bulk of internet users coming from first world or developed countries (Singh, 2004).

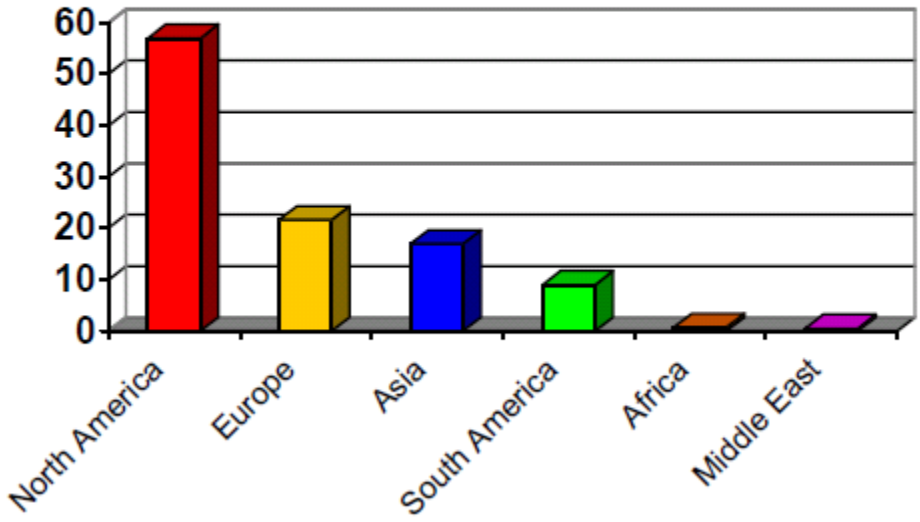


Figure 1: Internet users worldwide (Source: Singh A. M, 2004).

Developing countries only account for 4.25% of Internet users worldwide. Figure 2 below shows the top ten internet usage statistics of African countries by March 2008.

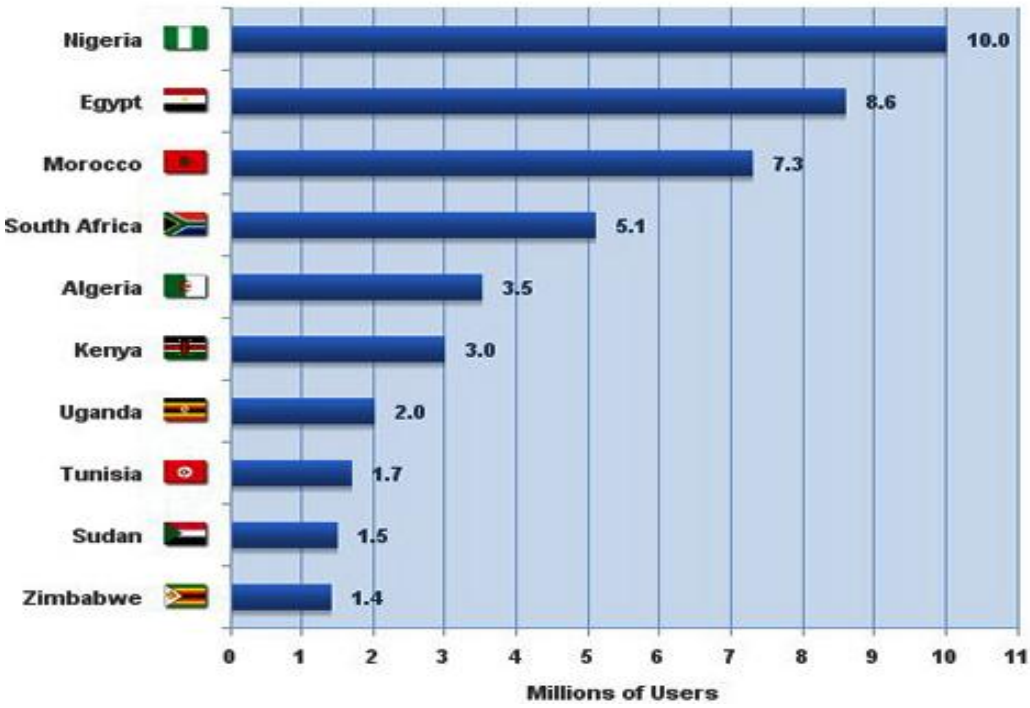


Figure 2: Internet use statistics in Africa. (Source: www.internetworldstats.com)

The digital divide is not only a consequence of the difference between the first world and the third world, but there are many factors contributing to it. According to Singh (2004) and Wertlen (2008), these include:

- High levels of poverty.
- Lack of telecommunications infrastructure.
- High costs of connectivity.
- High levels of illiteracy.

2.1.1.1 Information and communication technology (ICT) in South Africa

South Africa has a large population living in its rural areas – approximately 19 million (or 42% of the total population) in 2001, according to Statistics South Africa report (Kok et al., 2006). From the same report, a strong urbanization trend was identified, with up to 12% of the rural population migrating to the cities.

To address the problem of the digital divide in South Africa, an Information and Communication Technology (ICT) initiative was established to offer connectivity through the internet (Singh, 2004). However, this initiative has proven to be costly to low income earners and has been deemed an out-of-reach development for rural areas, in comparison to the low cost and mobility of a wireless network (Mansel, 2001).

As an increasingly important contributor to South Africa's gross domestic product (GDP), the country's ICT and electronics sector is both sophisticated and developing. With a network that is 99.9% digital and includes the latest in wireless and satellite communication, the country has the most developed telecoms network in Africa (ICT and electronics in South Africa, 2008).

2.1.2 Siyakhula Living Lab

Siyakhula Living Lab is a joint venture between the Telkom Centres of Excellence at the University of Fort Hare and Rhodes University, with the support of the Technology and Human Resources for Industry Programme (THRIP) of the Department for Science and Technology of South Africa and the Cooperation Framework on Innovation Systems between Finland and South Africa (COFISA). The distributed Living Lab focuses on providing a cost effective and robust, integrated e-business/telecommunications platform for rural South African communities. The Siyakhula Living Lab was piloted and is now a successful project in the Dwesa community.

2.1.2.1 Dwesa: Research Location

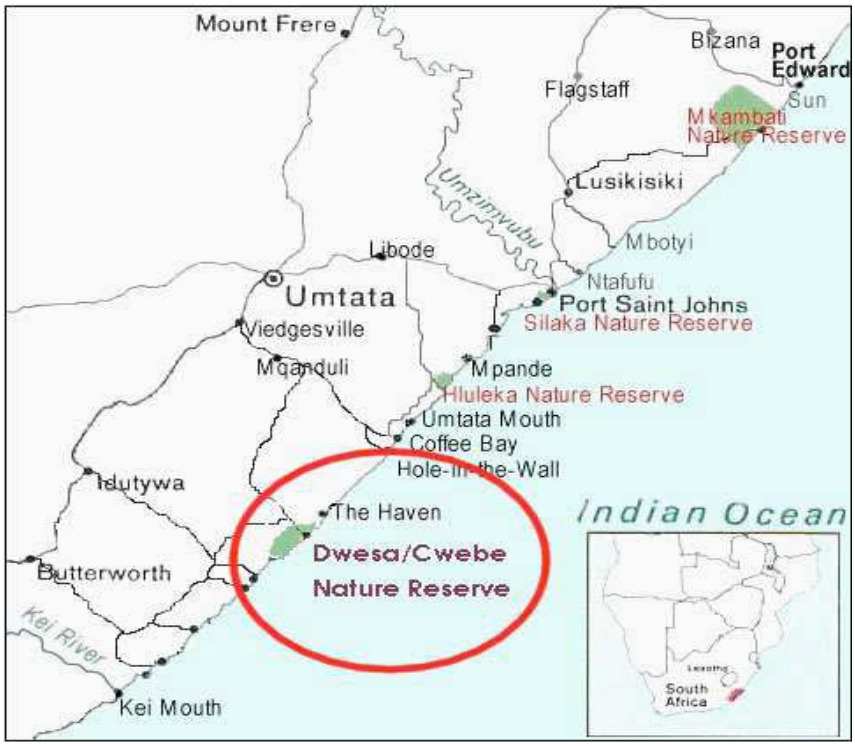


Figure 3: Transkei and Dwesa region

Dwesa is a completely rural community located on the Wild Coast of the former homeland of Transkei, in the Eastern Cape Province of South Africa (see Figure 3 above). It is characterized by a lack of infrastructure like roads and electricity, low literacy levels, high unemployment levels and intense poverty, among other things, which are typical of many rural regions in South Africa. The region features a coastal nature reserve and it was the site of one of the first restitution projects in post-apartheid South Africa. The region has significant potential for both economic and cultural tourism due to the rich cultural heritage and the marine conservation project undertaken at the nature reserve (Dalvit et al., 2007).

2.1.2.2 Services Currently in Dwesa

In Dwesa, the points of presence so far are the schools: Mpume, Ngwane, Ntokwane and Nodobo (see Figure 4).

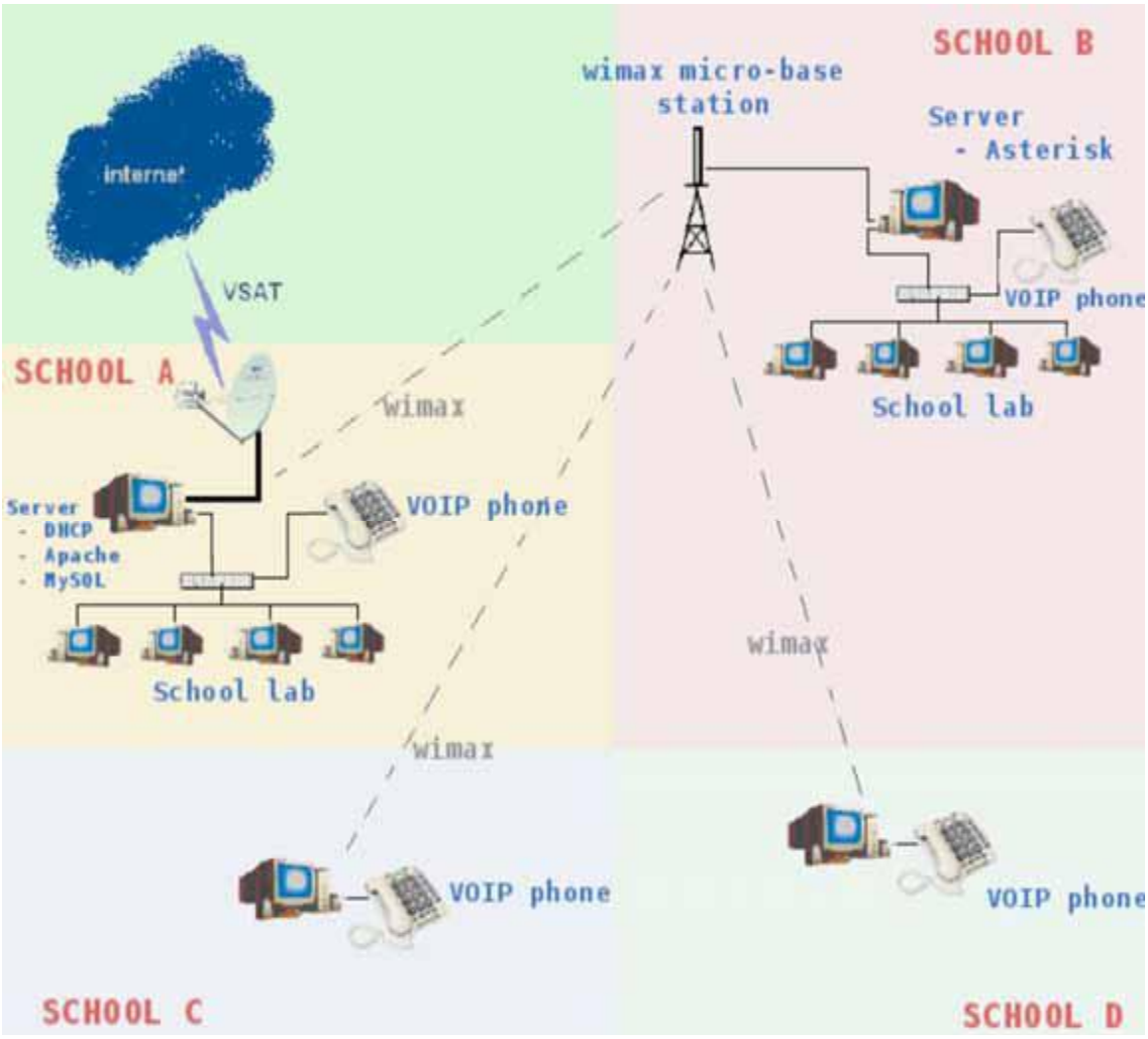


Figure 4: Dwesa Network (Dalvit et al., 2007).

The network is a converged IP network consisting of both wired and wireless technologies. In the setup, VSAT forms part of the backhaul connectivity to the internet. WiMAX forms the last-mile access to the other schools. It is a point to multipoint wireless network capable of providing a seamless wireless backhaul for the cellular network. It would be a perfect setup if the mobile devices to be used by the community were already equipped with WiMAX chipsets so they can be treated as actual subscriber units in the network. This would enable devices to create direct access to the network resources. Researchers from Fort Hare and Rhodes Universities develop, test and deploy various projects in Dwesa as a way to improve the socio-economic status of the community. Some of the projects include:

Deployed Projects:

- e-Commerce platform
- e-Government

Under development:

- e-Health
- Mobile applications
- e-Commerce additional plug-in e.g. billing module and the rewarding and negotiating module
- e-Judiciary

2.2 Wireless Networking Overview

Wireless networking technologies serve the same purpose as that of conventional wired networking except that they have the added advantage of mobility. A wireless network can operate over a long distance, such as voice and data networks like Global Systems for Mobiles (GSM), or a short distance using short range wireless connections, such as infra red technology.

2.2.1 Wireless Network Types

The three categories of wireless networks that will be discussed in this study are: Wireless Wide Area Networks (WWAN), Wireless Local Area Networks (WLAN) and Wireless Personal Area Networks (WPAN).

2.2.1.1 WWANs

A WWAN is wireless connectivity to the internet achieved through using a cellular tower or carrier technology to transmit signals over a range of several miles, to a mobile device (Deepak, 2006). It enables users to achieve connectivity over remote public or private networks. The third generation (3G) technologies are the current wireless technologies and they follow global standards and roaming capabilities.

3G networks enable network operators to offer users a wider range of more advanced services, while achieving greater network capacity through improved spectral efficiency (Jordan et al., 2002). They are integrated with voice and data transmission capabilities; the GSM network is an example of this (described in detail in section 2.3).

Example: WiMAX

WiMAX is a broadband or wide area network technology that serves to provide an alternative to wired technologies such as cable and DSL services. A WiMAX system consists of the following parts (Deepak, 2006):

- WiMAX tower – a single tower provides coverage to a very large area, as big as 8,000 km.
- WiMAX receiver – receiver and antenna could be a small box or PCMCIA card. They could also be built into a laptop.

2.2.1.2 WLANs

WLAN is a wireless network connection within a corporate or campus building, or in a public place such as an airport (Ajose, 2005). It provides the same data rates as traditional wired networks and adds the ability and advantage of mobility for users. WLANs are not a replacement of the Wired LANs; they are merely an extension or supplement to it, especially where the installation of conventional wired or cabled networks would be prohibitive.

Example: Wi-Fi

This is a wireless local area network technology providing distribution and service to end–user devices in a smaller footprint e.g. a campus or a school. Wi-Fi can be used for network backhaul and network extension via a bridge or mesh (Deepak, 2006). Wi-Fi hot spots are very small and there can be hundreds within a Wi-Fi infrastructure. It uses a technique known as Direct-Sequence Spread Spectrum (DSSS) to transmit data at up to 11Mbps across a 22 MHz channel (Ajose, 2005). The main problem with Wi-Fi access is that network coverage can be sparse.

2.2.1.3 WPANs

WPAN is a short distance wireless network specifically designed to support portable and mobile computing devices such as Personal Computers (PCs), Personal Digital Assistants (PDAs), wireless printers, storage devices, cell phones and a variety of consumer electronics or equipment (Jordan et al., 2002).

WPANs are what are referred to as cable replacement technologies that permit communication within a distance of about ten meters (10m), in all directions.

Example: Bluetooth

Bluetooth is a wireless standard that uses the radio waves located in the 2.4 GHz (2400 - 2483.5 MHz) frequency band. It was developed to provide wireless interconnectivity between small mobile devices and their peripherals. Target markets were the mobile computer, the mobile phone, small personal digital assistants and peripherals (Ajose, 2005). Today, more and more consumer electronic devices are equipped with Bluetooth capabilities.

2.3 Global System for Mobile Communication (GSM)

GSM is a digital cellular system widely used in Europe and other parts of the world. GSM uses a variation of time division multiple access (TDMA). It digitizes and compresses data, then sends it down a channel with two other streams of user data, each in its own time slot (Dogac et al., 2002).

GSM was first launched in Finland in 1991 and by end of 1997 its rollout covered more than 100 countries (Singh, 2009). According to the GSM Association, there were 564.6 million GSM subscribers by the end of July 2001 and by June 2008 there were more than 3 billion mobile subscribers worldwide, which the organization said is more than 86 percent of the world's cellular market.

GSM technology is operating in the 900 MHz and 1800 MHz frequency band and is limited due to its low data transmission speed; therefore, with the growth in data services the long term future of GSM is uncertain, unless it is changed to offer high bandwidth data services (ITU Telecommunication Indicators Update, 2005).

Apart from GSM, other wireless mobile telecommunication technologies include High-Speed Circuit-Switched Data (HCSD), General Packet Radio System (GPRS), Enhanced Data GSM Environment (EDGE) and Universal Mobile Telecommunications Service (UMTS) (Akhila et al., 2008).

2.3.1 Components of the GSM System

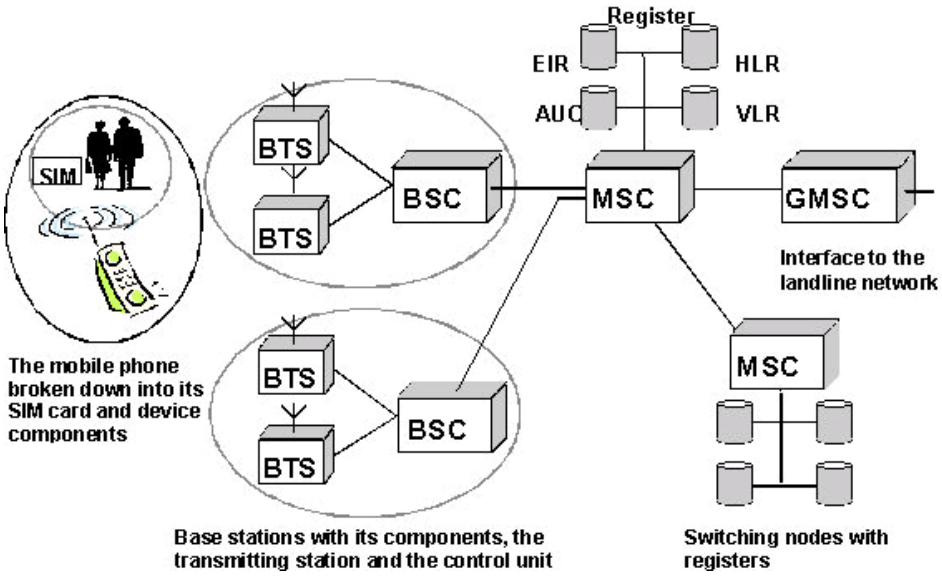


Figure 5: GSM architecture (Source: http://www.bsi.bund.de/literat/doc/gsm/gsm_e2.jpg)

GSM architecture consists of many components that deal with the various communication aspects of its functionality. Figure 5 above shows the various components of the GSM system. The GSM architecture consists of the following subsystems and their subsequent components **(**Rahnema, 1993**):**

- **Subscriber Equipment**

  - Mobile Station (MS) - The mobile telephone. These digital telephones include vehicle, portable and hand-held terminals.

- **The network and Switching System (SS)**

  This subsystem provides switching between the GSM subsystem's network and other networks. It consists of the following:

  - **Home Location Register (HLR)**- A database which stores data about GSM subscribers.
  - **Mobile Services Switching Center (MSC)** - The network element which performs the telephony switching functions of the GSM network.
  - **Visitor Location Register (VLR)** - A database which stores temporary information about roaming GSM subscribers.
  - **Authentication Center (AUC)** - A database which contains the International Mobile Subscriber Identity (IMSI), the Subscriber Authentication key (Ki) and the defined algorithms for encryption.
  - **Equipment Identity Register (EIR)** - A database which contains information about the identity of mobile equipment in order to prevent calls from stolen, unauthorized, or defective mobile stations.

- **The Base Station System (BSS)**

  This is a collection of devices that support the switching networks radio interface. The system consists of the following components:

  - **Base Station Controller (BSC)** - The network element which manages the radio activities of several Base Transceiver Stations (BTS). The BSC provides functions such as handover, cell configuration data, and control of radio frequency (RF) power levels in BTS.

- **Base Transceiver Station (BTS) -** The network element which handles the radio interface to the mobile station. The BTS is the radio equipment (transceivers and antennas) needed to service each cell in the network.

- **The Operation and Support System (OSS)**

  - **Message Center (MXE) -** A network element which provides Short Message Service (SMS), voice mail, fax mail, email, and paging.
  - **Mobile Service Node (MSN) -** A network element which provides mobile intelligent network (IN) services.

  - **Gateway Mobile Services Switching Center (GMSC) -** A network element used to interconnect two GSM networks.

  - **GSM Inter Networking Unit (GIWU) -** The network element which interfaces to various data networks.

2.3.2 Wireless Village Concept

The global communication industry, by 2015, aims to have five billion people connected anywhere and anytime using their devices. However, according to mobiThing (2012), there are now 5.9 billion mobile subscribers, estimated to be 87% of the world population. Approximately two billion people living in globally emerging markets do not have access to basic telephony services (Nokia Siemens Village Connection, 2008). The wireless village connection provides a cost efficient addition to existing GSM networks; it extends coverage beyond where conventional network roll-out would be too expensive. The wireless village connection involves three players, namely:

- Local entrepreneur
- Micro finance entity
- An operator

In a wireless village, a local entrepreneur collaborates with various stakeholders or micro finance entities to acquire a low cost GSM access point (GAP) within the village. The village connection architecture transfers responsibility for local network and business functions to a local operator (Nokia Siemens Village Connection, 2008). Subscribers are then acquired and mobile content applications are developed to suit their needs. The model requires an initial cash investment as a start button, but with time revenue for sustainability will be generated through controlled service charges based on the applications deployed

The wireless village targets low income earners and serves to provide them with mobile access while ensuring that the project is self-sustaining, needing some initial funding at the pilot stage (Nokia Siemens Village Connection, 2008).

2.4 Services in Marginalized Rural Areas

The question of how ICT access and use can bring about development in marginalized regions is of major concern (World Bank, 2005). The availability of ICT enables people to acquire knowledge that can be used to increase their social and economic standing in society, as they are able to use the knowledge to tap into various income generating opportunities and to influence the services they require from government. There is a correlation between information and poverty, whereby information leads to resources and opportunities that generate resources; access to information thus leads to access to resources and opportunities that generate more resources.

In this research, effective ICT services are classified as:

- Traditional - these ICTs include televisions and radios and are merely used for entertainment and information updates on day-to-day events e.g. politics, news, music and weather. This class of ICT is dependent on electricity and can thus be in-effective when it comes to information and knowledge exchange in rural areas that do not have access to electricity.

- Mobile – these use simple handheld devices that are not directly dependable on electricity as they can be charged and used anywhere at any time. Mobile devices are discussed in more detail in section 2.4.1

2.4.1 Mobile Devices in General

We can define a mobile device as a computing device which varies in size from small cell phones with limited resources to high-powered laptops with lots of resources. These devices are portable, which means that one can carry it around (Imielinski, 1994).

The mobile computing hardware's physical characteristics are defined by the size, weight, screen size, processor speed, primary and secondary storage, means of input and output, battery life span, communication capabilities and durability of the device (Gorlenko et al., 2003).

To a large extent, desktops and laptops tend to share many of the same usage patterns but laptops have some characteristics of mobile devices. The use of laptops is more similar to that of desktops than it is to mobile devices. Laptops have large screen sizes for exploring information and relatively large keyboards in comparison to handheld devices, such as PDAs, that have small screens and keyboards with a limited number of buttons (Gorlenko et al., 2003).

Even though laptops can be used as mobile devices, they are too big, compared to handheld devices, and they need a flat surface, such as a table, for the user to be comfortable. Whereas handheld devices fit well in the palm of the hand and can be operated by using one or two hands, while on the move; flat surfaces such as tables are not necessary. Wearable devices which support wireless connectivity such as mobile phones, PDAs and any other handheld devices are considered mobile devices, in this work (Gorlenko et al., 2003).

Mobile devices, using wireless technology, are on the verge of spurring an information and communication revolution in these regions. Mobile devices have proven to be more suited for rural conditions than personal computers; this is vastly due to the following mobile device attributes (Parikh et al., 2006):

- Long battery life,
- Wireless connectivity,
- Solid state memory,
- Low price,
- Immediate utility.

The popularity of mobile device use, in developing countries, has grown at an alarming rate. The low usage and ownership costs, as well as the reduced restrictions of use and mobility of these devices have enabled their use to extend to marginalized rural regions.

2.4.2 Constraints of Mobility

Wireless communication is vulnerable to the surrounding environment that introduces noise and echo to the transmission signal. Some implications of using wireless communication are vulnerability to disconnection and security threats.

2.4.3 Disconnection

A mobile computing device can disconnect voluntarily or involuntarily (Barbara, 1999).

- Voluntary disconnection - this happens when a device goes into sleeping mode to save power or when it detects a weak battery, or when the user switches it off.
- Involuntary disconnection - this occurs when a mobile device roams out of the wireless network signal boundary or when there is hardware or software failure.

2.4.4 Security Threats

Security threats to mobile devices and the data they store are on the increase in the form of theft or the unintentional loss of a mobile device. As a result, the sensitive or private data stored in the device ends up in the wrong hands.

2.4.5 Power Source

Mobile devices rely on a finite power source for energy, in the form of a battery. The power in a battery will last longer if the device is not intensely used, otherwise it will be used up quickly. Battery power can be depleted while the user is in the middle of collecting data, downloading material or running any other application installed on the device.

2.4.6 Operation Environment

While mobile clients use a wireless network, servers use a typically fixed wired network. The two thus communicate with each other using an integrated wireless/wired network. The new environment that results from this blending has limitations due to the nature of the resource, poor computing devices or low bandwidth, high latency and unreliable network connections (Imielinski, 1996).

The complexity of the new environment is because it involves two networks, with different bandwidths and configurations, which exchange data; a fixed wired network with abundant bandwidth and a wireless network with limited bandwidth.

2.5 Services Development

The development and deployment of financial systems involves a lot, for example deciding what software's to use, what the target devices are, how to test the system and so on. Once the system has been developed and tested it needs to be deployed to be used in the real financial world. The financial regulatory framework of South Africa supervises and enforces measures that the micro entities have to adhere to while offering financial services to the people. The following sections discuss the financial regulatory framework and the various ways in which mobile applications can be developed and deployed in the real world.

2.5.1 Financial Regulatory Framework

The existence of an efficient and effective financial regulation system is of great importance to any nation. This makes it possible to regulate financial institutions and financial service providers in order to nourish the nation's economic status (Falkena et al., 2000). Through these financial service providers individuals and entities manage to invest their savings on a day to day basis. The Policy Board for Financial Services and Regulation was formed in 1993, and part of their mission is to promote and maintain a safe and sound financial system which will be fair to investors, effective in supplying financial services to all, well structured and co-ordinated in terms of financial and regulatory matters (Falkena et al., 2000).

The regulation framework has many policies that have to be adhered to. For example, the securities regulation has the following three objectives (Policy Document, 2011):

- Protection of investors – regulation protects investors from misleading, fraudulent or manipulative practices

- Making sure markets are fair, efficient and transparent – regulation detects, deters and penalizes market manipulation and other unfair trading practices.

- Reduction of systematic risks- regulation should provide a mechanism that reduces the risk of financial market failure.

The financial regulation framework provides a guideline for financial services providers, as well as investors on how the financial system operates. It offers some principles that all parties have to be aware of and have to adhere to. According to one policy, for an entity to be a financial service provider it has to be appropriately licensed to operate inside the regulatory perimeter (Policy Document, 2011). According to (Falkena et al., 2000), there is a policy that promotes maximum competition among service providers and also maintains competitive neutrality among all financial entities, ensuring fairness. The M-Payment system is directly regulated by the Regulation of Branchless Banking in South Africa.

2.5.2 Regulation of Branchless Banking in South Africa

South Africa has had many branchless banking models developed and deployed to work hand in hand with the different banking institutions. According to Consultative Group to Assist the Poor (CGAP) there are two models of mobile banking that deliver financial services outside the traditional banks, and these differ on the following questions (Sultana R., 2009):

- Who will establish the relationship (account opening etc.) with the customer? The bank or the non-bank (Telecommunications Company)?

- What is the nature of agreement between the bank and the non-bank?

The two models are (Sultana R., 2009):

- Bank based- customer has a direct contractual relationship with a licensed and regulated financial service provider (bank), and may deal with a third party service provider equipped to communicate directly with a bank

- Non-bank based- customer has no contractual relationship with a licensed and regulated financial service provider, and deals with a third party or retail agent for transactions.

The South African regulatory framework for the Branchless Banking governs the following (CGAP, 2010):

- Use of agents- banks can use nonbank third parties to offer banking services beyond their traditional banking rollout as agents or through outsourcing arrangements

- Anti-money laundering (AML) – the Financial Intelligence Center Act (FICA) governs that a financial institution may not establish a business relationship or process a single transaction unless the institution has taken the prescribed steps to verify the identity of the client.

- Banks, nonbanks and E-money- only banks registered under the Banks Act are prescribed to be engaged in the business of banking, which includes collection of deposit funds from the public. Entrepreneurs willing to do take deposits must do so alongside the banks.

The M-Payment system currently is a proposal, a prototype that will, before its implementation, have to meet the above mentioned regulatory frameworks.

2.6 Classes of Mobile Applications

The following sections discuss the different classes of mobile applications.

2.6.1 SIM based

These are applications that run and are encrypted on SIM cards. SIM based applications offer the following advantages (Martineau, 1999; Thomas, 2006):

- Ease of use i.e. they are user-friendly and there is no need for subscribers to remember specific procedures.
- Powerful tool for digital security through identity verification and encryption, which are necessary for secure electronic commerce.
- Easy to deploy on the largest number of mobile devices.
- Opens up new revenue opportunities – positions the operator as a gateway for value-added services.
- Inspires confidence – offers anywhere, anytime services with the certainty that all transactions are fully secured.

SIM-based services include, but are not limited to (Thomas, 2006):

- Information on Demand
- Location-based services
- Mobile Banking
- Mobile Payment
- Email

- Browsing

2.6.2 Text based

SMS based applications use strings of text between subscribers or network connections to perform a required action in the network. Text messages can also be used for sending binary data over the air, and there are specific applications in the phone that handle binary data; for example, downloading ringtones and exchanging picture messages.

Typical SMS applications include (Brown et al., 2007):

- Consumer applications
    - Person to person messaging
    - Entertainment services (download a ring tone)
    - Location-based services (based on handset location)
- Corporate applications
    - Notification and alert services (emergency broadcast)
    - Managing contacts and appointments (SMS integration with Microsoft outlook)
    - Vehicle location (tracking)
- Cell operator applications
    - SIM updates – network operator remotely updates data stored on the SIM card
    - WAP push – push a URL of the content to be displayed and a browser on the mobile phone represents content to the subscriber

2.6.3 Audio based

Mobile devices range from cell phones with simple tone generation to PDAs and web tablets with advanced audio and video rendering capabilities. These manage the audio playback of various audio data and digitally recorded audio, simultaneously (Salami et al, 2006). Audio based applications also include the conventional call transactions that are billed using different rates according to the call plan subscribed to.

2.6.4 Handset based

These are applications that are embedded in the handset and can be accessible even if there is no SIM card inserted into the device. These applications include, but are not limited to, some of the following from the device menu applications:

- Calendar
- Games
- Calculator

2.7 Examples of Deployed Mobile Banking Services

There are many examples of financial mobile applications that are deployed globally. Most of these applications are not suitable for marginalized areas as they turn to concentrate on the commercial aspect and value in the urban markets. Lately, applications for marginalized and isolated regions have been on the increase and have seen so much interest in research fields, with researchers digging for ways to improve the socio-economic well-being of people living in deserted parts of their countries.

Mobile phones provide technological services that reduce costs, while increasing income and mobility. They can help to extend social and business networks, and they are clearly substitutes for journeys and for brokers, traders and other business intermediaries. This section lists some examples of the mobile applications that have been developed and deployed in such areas around the globe.

- **WIZZIT (CGAP, 2010), South Africa** - is a non-bank application developed and deployed in December 2004 by two entrepreneurs. It was designed to offer low cost banking services to the low income earners in South Africa. WIZZIT continues to sign up clients today but at a slow rate due to the strict interpretation of the anti-money laundering and combating the financing of terrorism (AML/FCT) regulation. Customers recruited and issued with a maestro-branded debit card that can be used for the following:

- Transferring money to third party accounts
- Checking balance
- Buying prepaid electricity
- Buying airtime for prepaid mobile phone

WIZZIT has no branches of its own, but works alongside ABSA bank and the Post Office. There are approximately 3500 deposit sites, and since customers possess a debit card they can even withdraw their money from any South African ATM machine.

- **MTN MobileMoney (CGAP, 2010), South Africa-** this application was launched in 2005 by MTN, one of south Africa's mobile operators, in a joint venture with Standard bank. The application is housed in Standard bank, and is available to all MTN subscribers. The application is SIM-based, hence it is embedded into the MTN SIM card and only MTN subscribers can open this MobileMoney account. New client information is captured using the mobile application, with his photograph and identity sent to the offices. Existing accounts have a daily transaction limit of R5000, and with the option of the MasterCard customers can make withdrawals from any ATMs in South Africa.

- **Text-A-Withdrawal, Philippines** (Ndiwalana, 2007) – This system is mobile phone banking service that uses GCash, a mobile money platform in Philippines, to turn mobile phones into virtual wallets. In this way users can buy or cash-in GCash credits, converting cash to GCash

- **M-PESA, Kenya** (Hughes et al., 2007;Sultana R., 2009) – This system is a text messaging (SMS) application that enables customers to complete simple financial transactions or micro-payments via a mobile phone, thus making life simpler for the users.it is a non-bank system launched by Vodafone and Safaricom. Money collected by agents is deposited in a trust account in one of the commercial banks and is protected not to be used for the purposes of lending, investing or any other manner for the account.

Table 1 below shows a comparison of some of the above mentioned mobile financial initiatives (Sultana R., 2009).

Table 1: Comparison of deployed mobile financial systems

Deployment	**M-PESA**	**MTN BANKING**	**G CASH**
Launch date	2007	2005	2005
M-banking Model	Non-bank based	Bank based	Non-bank based
Key Features	➢ 6.3 million customers ➢ 9000 agents ➢ US$170 million transaction in February 2009	Not reported as yet	➢ 1.2 million registered users
Customer experience	➢ Faster (98%) ➢ Convenient(97%) and safer (98) than alternatives ➢ Main means for sending money for 50 % of Kenyans ➢ 4 out of 5 say not having it will have a negative impact on their lives	➢ High charges ➢ Fees range about 4 to 7 times greater than would be affordable	One of the most affordable among the m-banking products
Impact on the life of the unbanked	Medium/ High	Low	Medium/High

2.8 Conclusion

As the world becomes increasingly technologically advanced, the solutions that are introduced tend to be more complicated and of little significance to rural communities. This brings about a division between those who have access to technology and those who do not - termed the digital divide. ICT4D is a framework that facilitates the integration of technological solutions targeting marginalized communities. With the wireless network evolving and influencing global markets and business sectors, the rollout of this technology has presented an opportunity for technological innovations that can integrate marginalized regions into urban centers. Mobile devices are portable and cheap, and use GSM technology. They have been seen as a tool for bridging the digital divide by developing mobile device applications for marginalized regions.

CHAPTER 3: MOBILE COMPUTING TECHNOLOGIES

The traditionally separate technologies of the internet and mobile computing have now started to converge, bringing promises for seamless wireless internet access through portable devices (Gavals et al, 2007). Effectively mapping internet content to mobile wireless devices requires not only new technologies, but innovative solutions that reduce the cost and maximizes efficiency in order to benefit both service providers and consumer (Saha et al., 2001). This chapter focuses on the enabling technologies for mobile computing; it discusses the different programming or markup languages and the wireless development platforms associated with them.

3.1 Wireless Platforms and Programming Languages

Wireless internet has many challenges, some of which are due to the different web markup languages used in wireless programming, such as Wireless Markup Language (WML), Compact Hyper Text Markup Language (cHTML) and eXtensible Hyper Text Markup Language (XHTML). The sections that follow below discuss the various markup languages and their respective development platforms.

3.1.1 WML and Wireless Application Protocol (WAP)

Mobile device constraints, such as small displays, low memory, low battery and limited processing power have resulted in common internet standards, such as HTTP, TCP/IP and HTML, which are not fully compatible with wireless devices (Saha et al., 2001). WAP is a protocol suite which enables interactive mobile devices to create and send data over the internet. WML markup language is the primary content format for mobile devices that implement the WAP specification. Figure 6 below shows how a wireless web can be accessed using WAP technology.

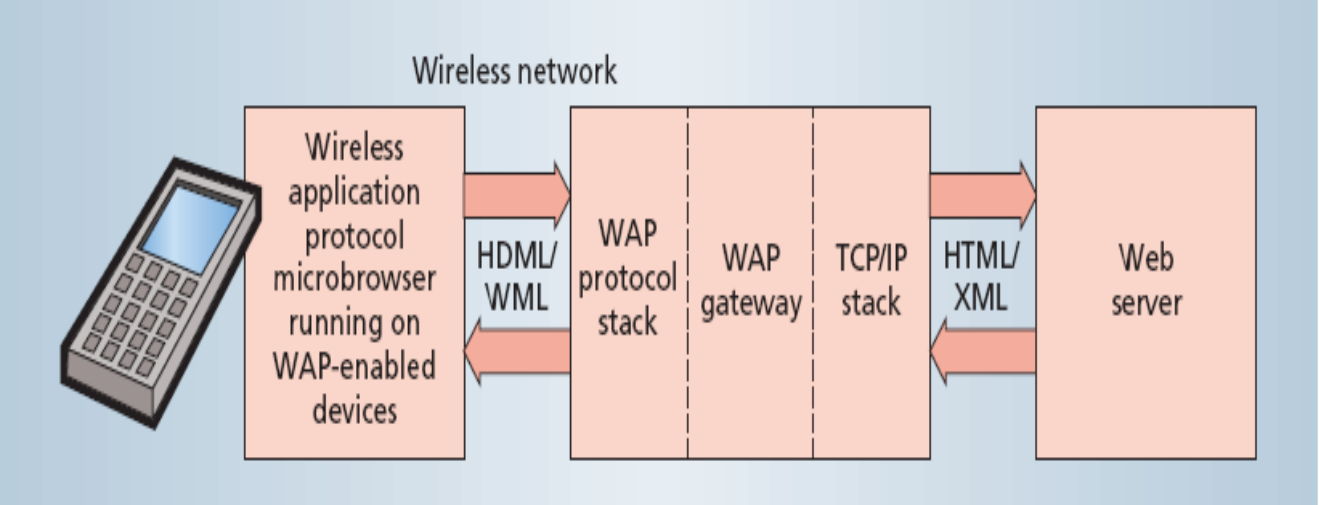


Figure 6: How WAP accesses the Wireless Web (Source: Saha et al., 2001).

The mobile device sends a request to the WAP application running on the application server by selecting the system's WAP address as locating a WML file (Woo et al., 2008). The WAP gateway then translates the requests from the WAP protocol stack to the TCP/IP stack that is then forwarded to the appropriate URL. The execution result of the response is then rerouted through the WAP gateway, translated to WAP, encoded and forwarded to the mobile client. Thus, the mobile device can display the data result as a response from the WAP address (Saha et al, 2001).

3.1.2 cHTML and iMode

cHTML is a subset of HTML 2.0, 3.2 and 4.0, and is used for iMode service that is supported by NTT DoCoMo (Woo et al., 2008). The fact that cHTML is a subset of HTML makes it easy for web developers to program mobile services using cHTML as its syntax is similar to that of HTML. The iMode service functions almost the same way as WAP. The user makes a request to the mobile internet from his iMode device. The request information is translated to cHTML and sent over the network to the Web servers at NTT DoCoMo, which processes the information and sends it back to the user (Deitel, 2002). With iMode, smart-phone users can browse the Net with a touch of a button. IMode has a transmission speed of 9.6 Kbps, utilizes a packet-switched connection, and has adopted CHTML as its markup language (McKitterick et al, 2003).

3.2 Mobile Runtime Environments

A runtime environment is a collection of subroutines and environment variables that enable support for application software at runtime. According to Efstathiadis et al. (2006), the runtime environment provides utilities, definitions, environmental variables, file system layouts, naming conventions and tool chains that allow programmers to test such scripts and build systems. This section discusses the three common runtime environments for mobile service development; the Java Platform, Micro Edition (Java ME) by Sun Microsystems, Binary Runtime Environment for Wireless by QUALCOMM based on C/C++ and the .NET Compact Framework by Microsoft Corporation based on ASP.NET Mobile.

3.2.1 Java Platform, Micro Edition (Java ME) Technology

Java ME is formerly known as the Java 2 platform, Micro Edition (J2ME). It is a Java platform that deals with constraints associated with developing applications for mobile devices such as PDAs, mobile phones, pagers and embedded devices (Perrucci, 2007). J2ME is not a piece of software or specification (Nylund, 2001). It is a platform, a collection of technologies, specifications and test suites defined by the Java Community Process (JCP) and combined to form a complete java runtime environment for different parts of the small device market (Fitzek, 2007) Figure 7 below shows the java platform and its different components including the Java ME.

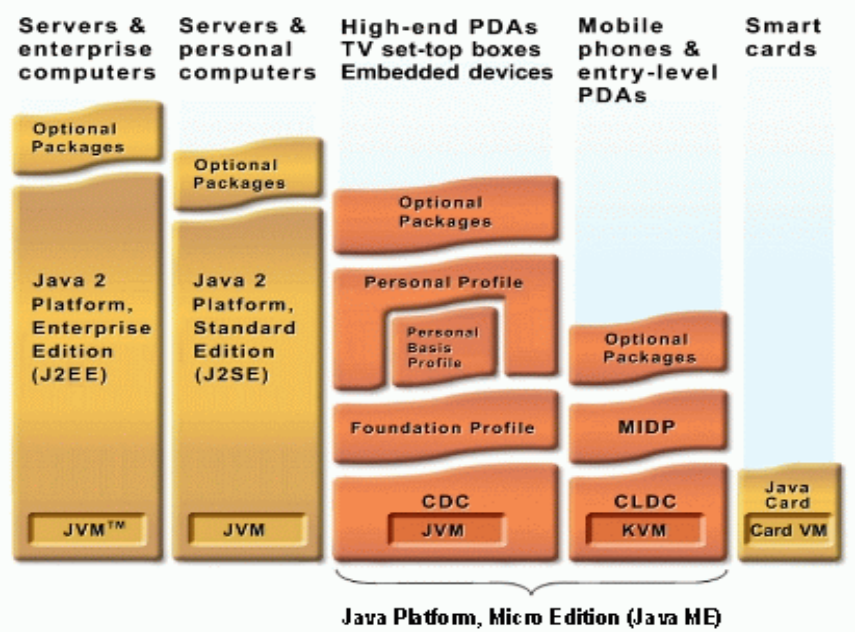


Figure 7: Java Platform (Java ME Technology
Source: http://javasun.com/javame/technology/index.jsp)

Figure 8 below shows the Java ME software.

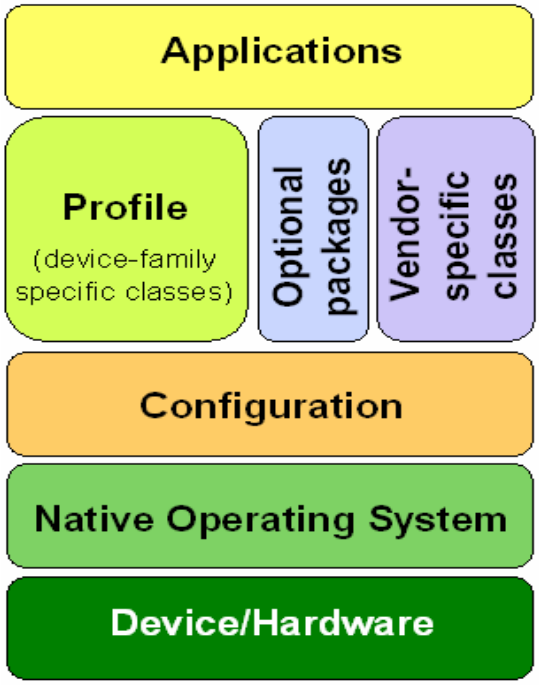


Figure 8: Java ME Software layers (Source: Isakow et al.).

Below is a brief description of some of the Java ME software layers shown in Figure 8 above:

- Configurations - these are specifications that provide a set of libraries or APIs and a java virtual machine to be used with a specific class of small devices (Perrucci, 2007). Chu (2001) defines the configuration as a specification that "defines a minimum platform for a "horizontal" category of devices, each with similar requirements on total memory budget and processing power". It defines the minimum set of libraries, but may not contain additional or optional features (Nylund, 2001). A configuration coupled with a specific profile forms the core application functionality required by mobile applications i.e. a java runtime environment and a rich set of APIs ((Fitzek, 2007). There are currently two configurations, namely; the Connected Limited Device Configurations (CLDC) and the Connected Device Configuration (CDC). Table 2 below compares the CLDC and CDC configurations.

Table 2: CLDC versus CDC. (Source: http//java.sun.com/products/cldc/overview.html.)

ATTRIBUTES	CLDC	CDC
RAM	RAM + ROM + Flash Memory must greater than 128K~512K	Equal or greater than 256K
ROM		Equal or greater than 512K
Power	Battery	No limit
Network	Short time period	No limit
User Interface	Small screen and use screen to input	According to device

Example: Connected Limited Device Configuration (CLDC)

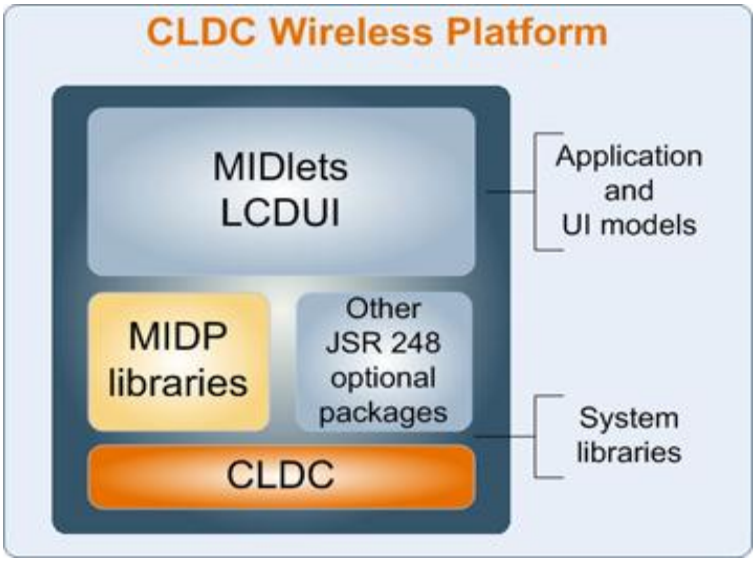


Figure 9: CLDC Platform (Java ME Technology.
Source: http://javasun.com/javame/technology/index.jsp).

Figure 9 above shows the CLDC platform. Most libraries from the CLDC configuration are inherited from Java Platform Standard Edition (J2SE) (Perrucci, 2007).

The CLDC specifies the use of the compact Kilobyte Virtual machine (KVM), an implementation of Java Virtual machine (JVM) that meets the CLDC specifications for small devices. The CLDC addresses the following areas (Topley, 2002):

- Java language
- Core Java libraries (java.lang.*, java.util.*)
- Input/output
- Networking
- Security
- Internationalization

Some of the features of the CLDC configuration are processed by profiles coupled with it, including user interface and event handling. The CLDC targets devices with the following additional capabilities (Topley, 2002):

- Intermittent connection and limited bandwidth
- A 16-bit or 32-bit processor with a clock speed of 16MHz or higher
- At least 192 KB of total memory available for the Java platform

- Profiles – a profile is a subset of a particular configuration. It enhances functionality, customization and extensibility in application development for specific device classes (Larsson et al., 2004). It defines a set of APIs for devices with the same uses (Isakow et al., 2008). According to Sun Microsystems, J2ME Building Blocks for Mobile Devices a profile provides a complete toolkit for implementing applications for a particular kind of device such as a pager or a cell phone. The profiles provide domain specific capabilities and the Mobile Information Device Profile (MIDP) is currently the only implemented profile in CLDC (Fitzek, 2007).

  Example: MIPD

- MIDP is a J2ME profile for small devices like cell phones and two way pagers (Kolsi et al., 2004). The MIDP profile defines a platform for dynamically and securely deploying optimized, graphical, networked applications (Sun Microsystems: Mobile Information Device Profile (MIDP); JSR 37, JSR 118). The MIDP profile layer includes a set of APIs related to user interface, storage and network capabilities for a particular category of devices e.g. cell phones. It allows for programmers to develop applications that can run on any platform – this is referred to as a "write once, run anyway" phenomenon. Devices that are MIDP compliant share the following characteristics (Muchow,2001; Sun Microsystems, 2000):

  - Low memory capacity
  - Low connectivity (9600 bps)
  - Low graphic capacity

- 128 KB of memory for MIDP components
- Data input alphanumeric very reduced
- 8 KB of memory for persistent application data storage
- 32 KB of memory for the Java stack in runtime

- Optional Packages - these define a set of APIs that enhance java application functionality, and support common behaviors not defined in a single configuration or profile. It extends functionality for Java ME configurations and profiles, creating a rich software stack. The following is a list of some of the optional packages available in Java ME technology:

    - Wireless messaging API (WMA); JSR 120, JSR 205 - provides platform-independent access to wireless communication resources like Short Message Service (SMS) (Sun Microsystems: Wireless messaging API (WMA); JSR 120, JSR 205).

    - Mobile 3D Graphics; JSR 184 - the API is used in the development of 3D graphics applications for Java ME platform and CLDC configured devices (Java Community Process).

3.2.2 Binary Runtime Environment for Wireless

BREW is Qualcomm's open source application development platform for enhanced cell phone services that run at the firmware level of Code Division Multiple Access (CDMA) technology. BREW runs between software application and the Application Specific Integrated Circuit (ASIC) level software (Sun, 2004). It manages all telephony functions without the developer needing to code to the system interface. In other words, BREW system developers do not need to understand embedded systems programming. Their focus is on the design and development of the system (Salvage, 2005).

Figure 10 below shows the BREW architecture and its relations to existing mobile applications.

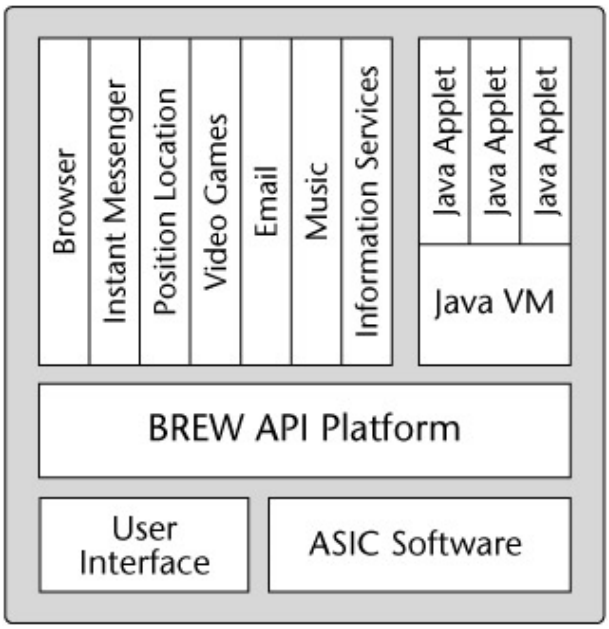


Figure 10: Brew Architecture.
(Source: www.developer.com/ws/brew/article.php/1454711/What-is-BREW.htm)

Unlike Java ME that runs on virtual machine, BREW runs on the chipset level hence it requires less processing power and is faster. On the phone, BREW acts like a thin client and is capable of running on devices that employ the other air interface standards, i.e. it can support GSM/GPRS, UTMS and CDMA (Salvage, 2005). The following are the benefits of using BREW in mobile application development (Sun, 2004):

- Consumers - BREW promotes a whole range of applications that a consumer has the privilege of downloading and personalizing in their devices. This allows the consumer to pay only for services or applications of interest to them e.g. instant messaging.
- Developers - application development is independent of the type of device, carrier or any physical device prototype. Developers use any programming language to create the various applications for CDMA enabled devices. The applications are tested for compatibility and quality before deployment.
- Handset manufacturers - their task is now to integrate the manufactured handsets or devices and BREW before quickly releasing the devices onto the market. There is also no application testing as the applications are deployed on the device at a later time.

- Wireless operator - these are the parties that provide wireless applications hence profiting from the revenues. They also benefit from the capability of providing a broader range of applications to be deployed over their networks.

3.2.3 Microsoft .Net Compact framework

The .NET Compact Framework (.NET CF) is a Microsoft product developed to lure corporate developers to windows for mobile devices (Thai et al., 2003). It is used to develop programming interfaces to applications and APIs for the windows platform. It inherits the .NET Framework of the Common Language runtime (CLR). The .NET CF architecture consists of the following distinct components:

- Optimized CLR - according to Kesic et al. (2009:4), this is an engine for running managed code on resource-constrained devices with limited memory, effectively using low battery.
- Class Libraries - these are the specific .NET CF libraries, combined with a subset of the .NET Framework and supporting such features as Windows Communication Features and Windows Forms (Thai et al., 2003).
- Windows CE operation System - according to Kesic et al. (2009:4), this is an operating system used by the .NET CF. Windows Forms, graphics and Web services, in order for them to run on mobile devices; they were developed rather than inherited from the full .NET Framework.

Figure 11 below shows the architecture of the .NET CF.

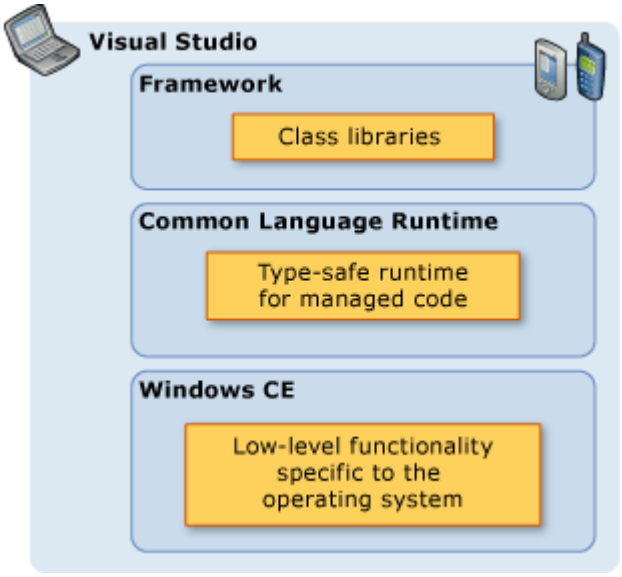


Figure 11: .NET CF Platform Architecture
(Source: http://msdn.microsoft.com/en-us/library/f44bbwa1.aspx)

3.3 Middle Tier Technologies

A middle tier is an application that connects a client component to the server component, adding or retrieving some data in the process. The middle tier participates in the workflow of the business processes and facilitates connectivity to the database and various back end systems for data storage and service access (Stal, 2000). This section will briefly look at the Java servlet and WAMP (Apache, PHP and MySQL for Windows) technologies.

3.3.1 Java Servlet Technology

A servlet is a module that runs inside request/response-oriented servers, such as Java-enabled web servers (Moss, 1999). Servlets are platform-independent and support the following Java features:

- Networking

- Multi-threaded processes
- Database connectivity
- Object orientation.

Servlets technology use classes and interfaces from two packages, namely the *javax.servlet* and *javax.servlet.http* (Hunter et al., 2001). The javax.servlet package has classes that support servlets independent of any protocol, with these classes extended by classes in the javax.servlet.http, thus enhancing HTTP functionality (Hunter et al., 2001). Figure 12 below shows the servlet architecture:

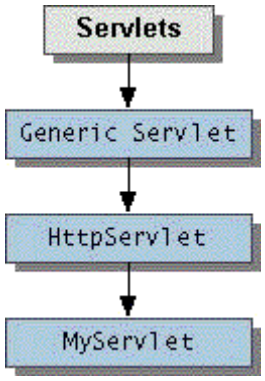


Figure 12: Servlet Architecture.
(Source: www.cse.iitb.ac.in/dbms/Data/Courses/DBIS/Software/servlets/servlet_tutorial.html)

To manage its communication with clients, the servlet interface offers the following methods (Hunter et al., 2001; Moss, 1999):

- *init()* – called only once when the servlet is initialized
- *destroy()* – called only once when the servlet destroys a servlet instance and makes sure that any persistent state is synchronized with the servlet's current in-memory state
- *service()* - this allows the servlet to respond to a request.
- *getServletConfig()* - returns a servlet config object containing initialization parameters and startup configuration for the servlet.

There are two types of servlets that can be implemented, i.e. the Generic servlet and the HTTP servlet interface. These servlets implement the *javax.servlet.Servlet* by extending the following classes:

- Javax.servlet.GenericServlet or
- javax.servlet.http.HttpServlet

The generic servlet extends the following servlet class:

- *Javax.servlet.GenericServlet* – defines a generic protocol independent servlet, with no support for HTTP or any other transport protocol. The generic servlet overrides the service method to handle requests, as appropriate for the servlet (Hunter et al., 2001).

Figure 13 below shows how a generic servlet handles requests:

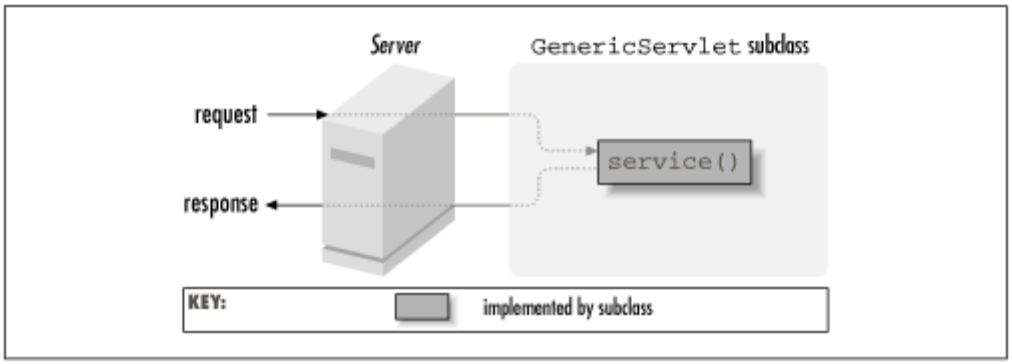


Figure 13: A generic servlet handling a request (Source: Hunter et al., 2001).

The HTTP servlet extends the following servlet class:

- *Javax.servlet.http.HttpServlet* - these have built-in support for HTTP and are useful in the browser environment. The HTTP servlet does not override the service() method, but offers some of the following methods that can be re-written as per the programmer's choice (Hunter et al., 2001;Moss, 1999):
  - *doGet()* - handles GET request.
  - *doPost()* – handles a POST requests
  - *doPut()*- handles a PUT request
  - *doDelete()* - handles DELETE request
  - *doOptions()*- handles an OPTIONS request
  - *doTrace()* - handles TRACE request

These methods are named according to the type of HTTP request received and users can override them to generate their own responses.

Figure 14 below shows how the HTTP servlet handles the GET and POST requests:

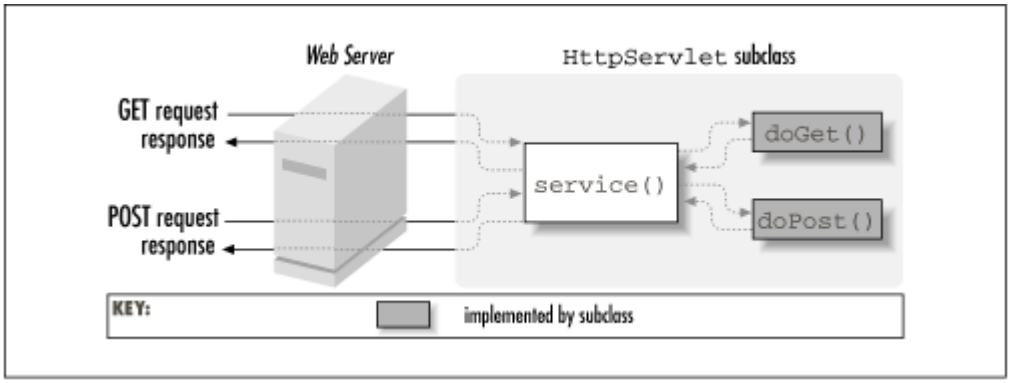


Figure 14: An HTTP servlet handling requests (Source: Hunter et al., 2001).

Servlets in the middle tier can be used to connect to and access resources in the server, adding or retrieving data from the database component. In this project the servlet can be used to connect the mobile client application to the database. The servlets use the Java Database Connectivity (JDBC) API to perform database related operations using java code. The JDBC API provides methods for querying and updating specific data in a Relational Database Management system such as SQL, Oracle etc**.** The JDBC assists developers in writing programs that can perform the following operations (Patel P. et al., 1997):

- Connect to data source e.g. database
- Send queries and update statements to database
- Retrieve and process results obtained from interaction with the database.

3.4 WAMP Technology

WAMP is an acronym for the following components (Austin et al., 2010):

- *Windows* – platform in which the WAMP software package is installed and run
- *Apache* - the web server

- *MySQL* – the database used as a data repository system.
- *Hypertext Preprocessor (PHP)* – scripting language that facilitates the connectivity and data flow between the client side application and the server.

WAMP is a mini-server bundle that runs on almost any Windows Operating System (Austin et al., 2010). The following discussion offers a brief description of the components of WAMP.

3.4.1 Apache

Apache is a web server, a project of the Apache Software Foundation, which provides support for open source projects. Apache is primarily used to serve both static content and dynamic Web pages on the World Wide Web. Many web applications are designed expecting the environment and features that Apache provides (Lakhani et al, 2003). The following is a list of some of the properties of an Apache server (Trichkova, 2009):

- Powerful and flexible
- HTTP/1.1 compliant
- implements the latest protocols
- customizable through writing 'modules' using the Apache module API
- offers access to full source code and an unrestrictive license
- runs on different operating systems

3.4.2 MySQL

MySQL is a *small, simple and compact database server* that can be used in development of small and medium sized applications (Trichkova, 2009). It is available as a simple binary or downloadable source code that can be modified to suit developers' application specifications. In addition to the above, MySQL also boasts a low total cost of ownership and high stability (Suehring, 2002). It is a database management system unmatched in speed, compactness, stability and ease of deployment (Trichkova, 2009).

MySQL has the following characteristics (Widenius et al., 2002):

- Supports standard SQL
- Compiles on a number of platforms
- Free to download and use
- Is ideal for both small and large applications

PHP and MySQL server can be used together in application development. The following PHP function is used to connect to the MySQL server, with the definition of arguments used following thereafter (Quigley et al., 2007; Lerdorf et al., 2006):

```
<?php
        $link = mysql_connect
($hostname,$username,$password);
?>
```

- ***$hostname*** = the domain name of the MySQL server. If both the web server and MySQL are located on the same machine/computer, you can simply use "localhost".
- ***$username*** = login ID/username of the MySQL server
- ***$password*** = password for the MySQL server

The function above allows for a connection to a database server to be created, after which several queries can be made to a specific database in the server e.g. adding data or updating information stored in the database.

3.4.3 PHP

This is a general purpose server side scripting language executed on the server that has the following attributes (Lerdorf et al., 2006):

- Support for many databases e.g. MySQL, postgreSQL.
- Open source software
- Can be embedded in HTML
- Has file extension of ".php" , ".php3", or ".phtml"

Much of PHP syntax is borrowed from Java, Perl and C although PHP has its own self defining pool of features. PHP can be used with MySQL queries to connect to, query and update the database system. A PHP script starts with <?php and ends with ?>. It may contain HTML tags, and has PHP code in it. Below is an example of a simple PHP script that sends a message "Computer Science" to the browser (Quigley et al., 2007, Lerdorf et al., 2006):

```
<html>
        <body>
                <?php
                    echo    "Computer        Science";
                ?>
        </body>
</html
```

3.5 Conclusion

This chapter discussed the various technologies used to implement wireless mobile systems. It covered theory on the mobile runtime environments and scripting languages. As this system uses Java ME technology and a WAMP server, a greater part of the discussion was based on these two technologies.

CHAPTER 4: SERVICE DESCRIPTION AND DESIGN

This chapter briefly describes the main components, requirement specifications and the tools used to implement the system prototype. The Java ME application (midlet) which was developed consists of two modules and the ensuing sections further discuss the midlet attributes.

4.1 System Goals

The prototype was developed to meet the needs of the rural community of Dwesa. It consists of the virtual money transfer and electricity buying modules. Hence, the application serves to act as a rural bank as well as a means of buying electricity using a mobile device to interact with various entities in the process.

4.2 System Architecture

The prototype was designed using a multi tier approach which offers distinction on the architectural layers. The system uses the three layer architecture; this consists of presentation, business logic and data attributes. Figure 15 below shows the system architecture used in the development of the prototype.

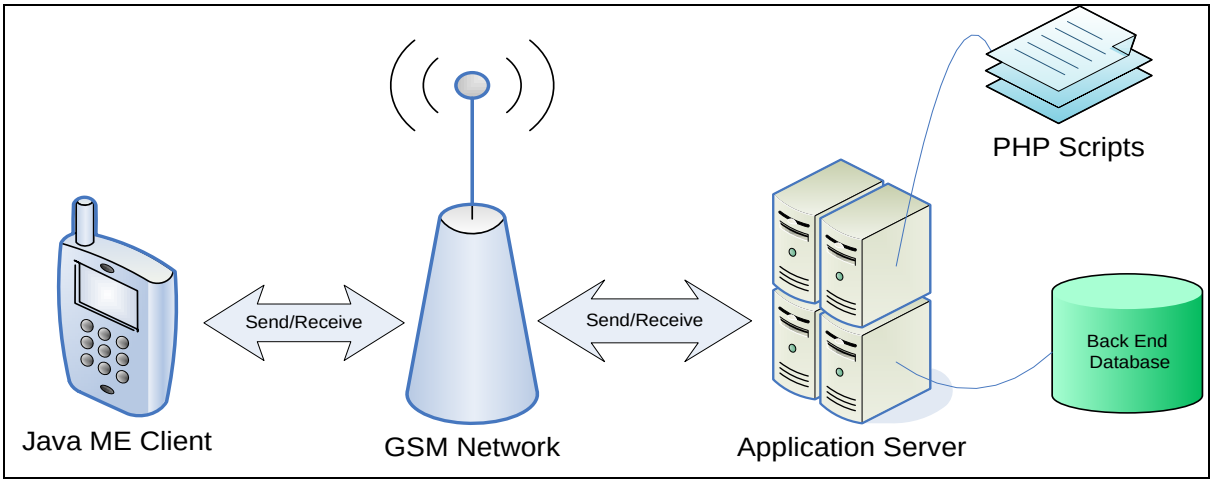


Figure 15: System architecture

The Java ME client resides on the mobile device. Once started, the client application can send or receive data through the GSM network to the backend. In this case the Java ME exists in the presentation layer. In order to access or manipulate data in the database some business logic scripts, written in PHP and residing in the server, are used. These are responsible for the two way communication between the GSM network and the backend database by executing specific queries to facilitate data flow and service access. This thus forms the business logic layer. The database forms the data layer, acting as a storage device for service data and the vast strings of text that the Java ME client sends to the server.

4.3 System Components

The prototype is called an M-Payment System and has two modules, as stated in previous chapters. The two modules are functionally independent but have certain components in common. Figure 16 shows the components of the M-Payment system. It consists of the Virtual Money Transfer (VMT) module and the Electricity Buying (EB) module. The menu components in green are for the VMT only, the ones in purple are for the EB module only, and those in yellow exist in both modules.

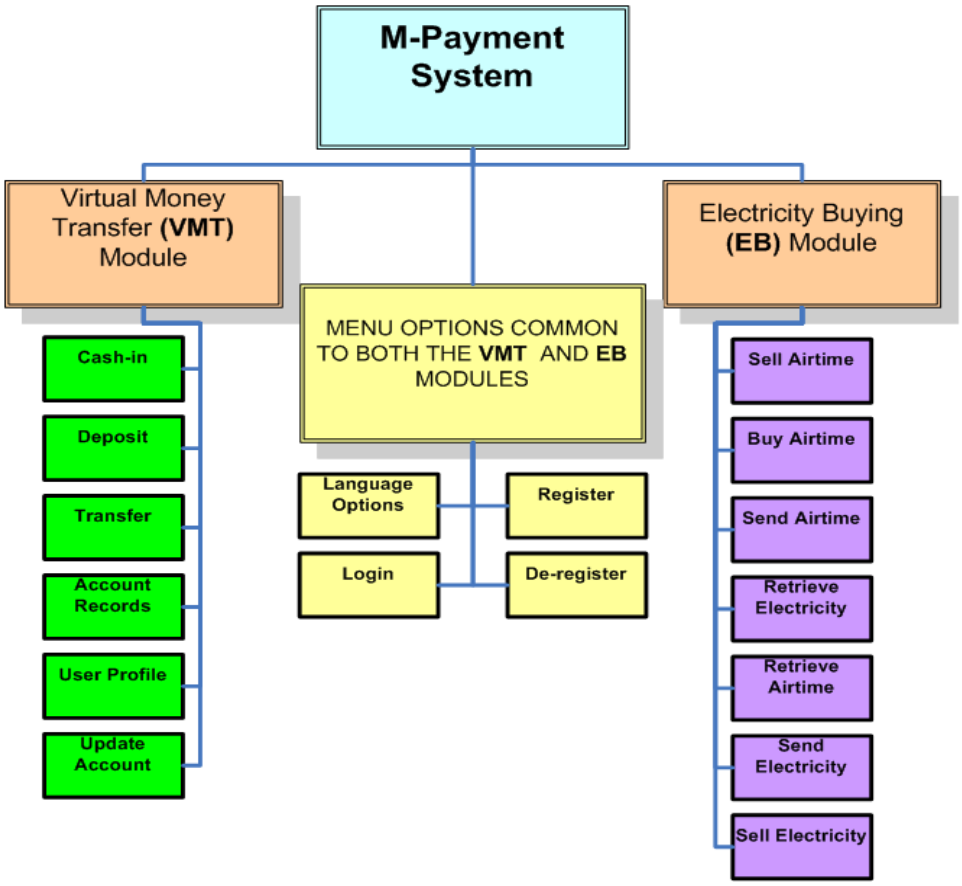


Figure 16: M-Payment System Components

4.4 Functional Requirements

This section provides a brief overview of the system's mandatory functional requirements. The project offers rural users with an M-Commerce application, consisting of the following modules:

- ***Virtual Money Transfer (VMT)*** *module* - offers banking transactions locally (in the rural community), without owning a bank account. Facilitates the sending of money from a town based user, with traditional banking infrastructure, to a rural based user.
- ***Electricity Buying (EB)*** *module* – facilitates the buying of electricity from a town entity while locally based.

The system is designed for three different types of users, namely the Rural Entrepreneur (RE), Town Entrepreneur (TE) and Subscribers. Some of the system's functions are only defined for system administrators (TEs and REs) while some are only for subscribers.

4.4.1 Virtual Money Transfer Functional Menus

Figure 17 below is a use case diagram for the virtual money transfer module. The pink cases represent cases common to both the VMT and the EB modules e.g. login menu. The blue cases are those that are only present in the VMT module.

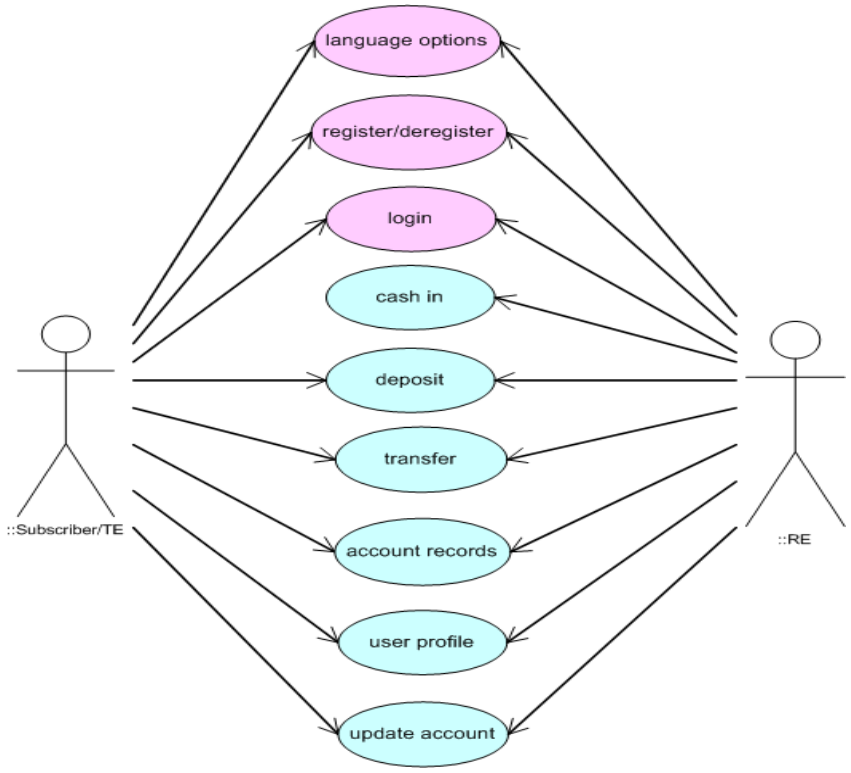


Figure 17: Use case for Virtual Money Transfer Module

The following is a brief description of the different menus in the VMT module:

- *Language options* - this allows users to select a language of preference in using the system resources. Since the target community for the application is a Xhosa community, the current application only has isiXhosa and English options.

- *Register/ Deregister* – register allows users to register their personal information to be eligible to access system resources. Deregister option removes all information about the registered user from the database, hence becoming ineligible to use system.
- *Login* - used to authenticate users. Uses a cell phone number and secret password for user authentication in the database.
- *Cash in* - allows user to withdraw or claim some money at cash in point. Only used by the cashier (RE in this case).
- *Deposit* - used when depositing virtual money into a user's account in the database, after a successful deposit in the traditional banking infrastructure in town.
- *Transfer* - used to transfer money from one user account, in the database, to another.
- *Account records* - shows details of user account; it is a mini bank statement
- *User profile* - shows user personal information like name, surname and other basic data.
- *Update account* - used to update user profile data. Does not update sensitive data like balances.

Figure 18 shows the simple sequence diagram for the VMT as used in performing a deposit, by a town based user to a rural based user.

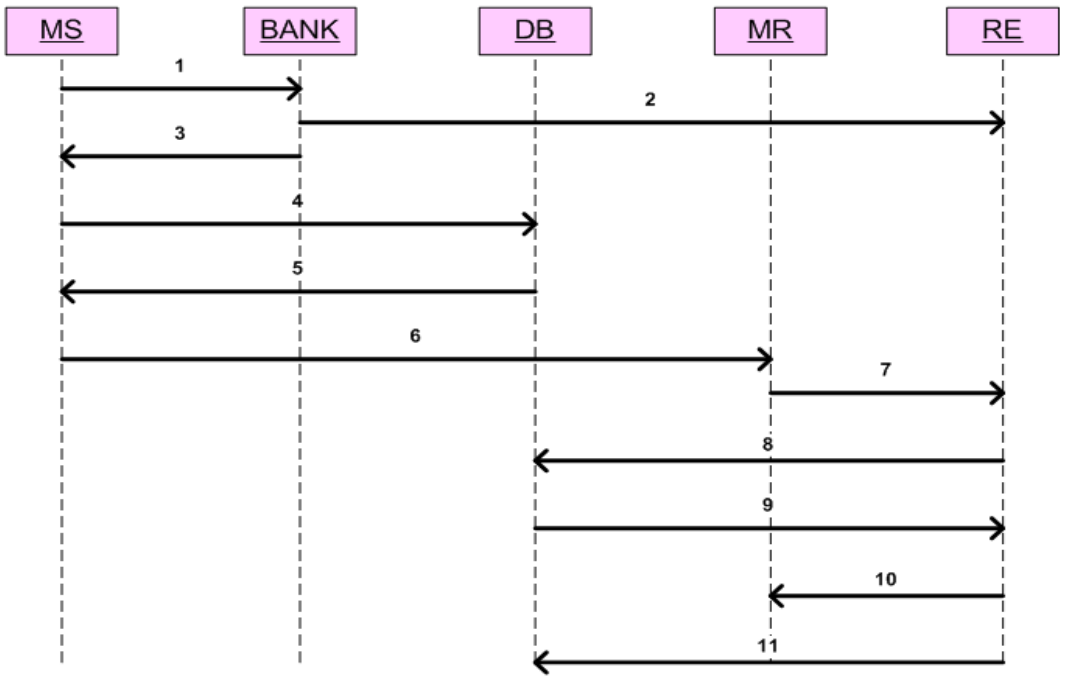


Figure 18: Sequence Diagram for the VMT module

The actors involved in the deposit transaction are:

- *Rural Entrepreneur* ***(RE)*** - this is a rural based shop owner who will act as a rural bank or cash in point.
- *Money Sender* ***(MS)*** - this is a person based in any town with banking infrastructure, and is responsible for sending money, through a deposit, to any person staying in a rural area with no banks.
- *Money Recipient* ***(MR)*** - this is the person abiding in the rural area to which the money is being sent to by the **MS** based in a town with a bank.
- *Bank* – banking infrastructure in a town. This is where the **RE** has opened an account into which the **MS** will deposit the money destined for the **MR** in the same location as the **RE**. The bank account must be configured to send an SMS notification to the **RE** after each deposit transaction.
- *MySQL Database* ***(DB)*** - once registered to use the M-Payment system, each subscriber owns an account which has its details stored in the database. The account details include

mobile phone number, deposit references, transaction dates and balances. These details exist in one or more related tables inside the system database.

To access M-Payment system resources, a subscriber logs into system using a unique *username* and a *password* stored in the system database. Since every mobile phone number is unique, it represents the username, with the password being any string of text specified by the subscriber during registration. Once successfully logged in, the subscriber can perform any transaction of choice as provided by the system. The following lists the sequential steps taken when **MS** deposits money to **MR** and how, after a successful deposit, the **MR** cashes in from the **RE** (as illustrated in Figure 18). The numbers used in the list below represent those on the arrows in Figure 18.

1. **MS** first starts the M-Payment application. After successful login, deposit option is selected. A deposit form, with an automatically generated unique reference number, is shown on the device. **MS** fills the bank deposit slip for a deposit of **R*X*** ***(x Rands)*** into **RE**'s account. As the deposit reference he uses the automatically generated reference digits from the system.
2. Once the deposit to the **RE**'s bank account has been successful, an SMS automatically generated by the bank is sent to the **RE** to confirm the deposit. This serves as an alert to the **RE** that an account in the database might have been updated after the deposit of **R*X***.
3. **MS** receives a bank deposit slip as evidence of a transaction in the primary bank facilities.
4. **MS** sends the reference digits, **MR**'s phone number, **RE's** name and the amount deposited to the database using the deposit form.
5. A response is sent back confirming the sending of the deposit data to the database.
6. If step 5 above is successful, **MS** then notifies **MR** of a successful deposit by using an SMS which contains the deposit details.
7. **MR** then consults the **RE** about the deposit.

8. **RE** logs into the M-payment system to access the *cash-in* menu. The cash-in form requires the **MR** to enter his phone number and secret transaction pin specified on registration. The database is then queried for all the amounts owed to **MR** by the **RE**.
9. If the deposit was successful the database will send a response showing the amount owed to **MR** by the **RE**.
10. If **MR** is keen to withdraw all or a part of the balance with the shop owner, then the amount **R*Y*** to withdraw is specified and is given to the **MR**.
11. The balance for **MR**'s account with the **RE** is then updated using the cash-in form, which after having successfully updated should show the balance as **R(*X*-*Y*).**

Once a **MR** has a balance with any of the various **RE**s, he can cash in part of the balance at any time. He can use the system to purchase from the **RE**'s shop or withdraw hard cash.

4.4.2 Electricity Buying (EB) System Modules

This module is part of the M-Payment system and is used by rural people to purchase electricity vouchers from town, without travelling. It involves four actors - Rural Entrepreneur (**RE**), Urban Entrepreneur (**UE**), Electricity Vendor (**EV)** and a Rural Electricity Consumer (**REC**). This module has three menus that are shared between it and the VMT module. The following menus are common to both modules and will not be included in the use case diagram for the EB module:

- Language options
- Login
- Register
- Deregister

Figure 19 below shows a use case diagram for the electricity buying module.

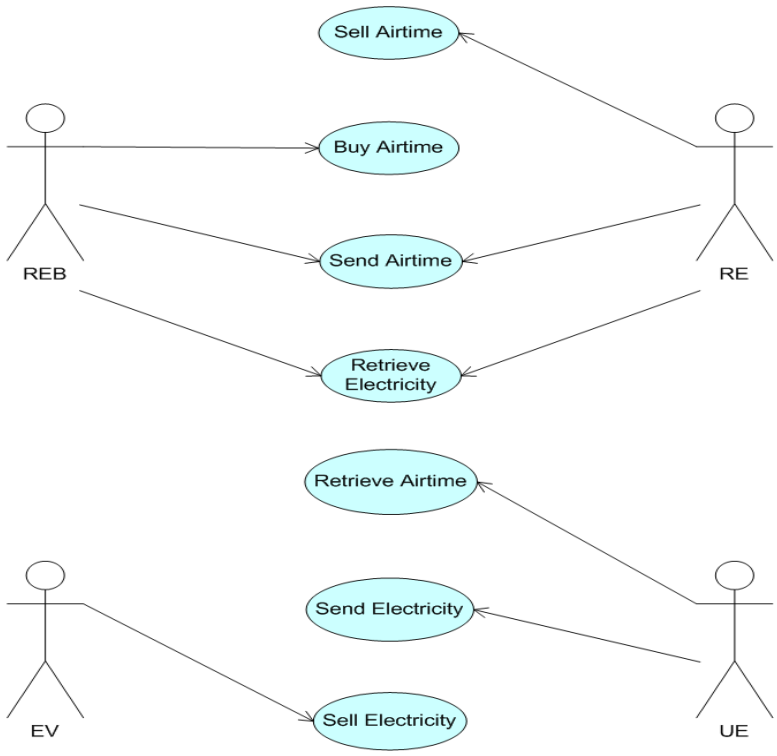


Figure 19: Use Case for Electricity Buying Module

The **REC** is a person abiding in an electrified homestead in a rural area isolated from a town from which to buy electricity. He therefore buys electricity by sending airtime voucher digits to a town based **UE**, who runs a phone shop into which he will load the airtime voucher. If the voucher is valid the **REC**'s account in the system database is updated with the electricity voucher, prompting the **REC** to retrieve it and load it in his electricity meter. The following section briefly describes the electricity module components shown in Figure 19 above:

- *Sell Airtime* - this is not implemented but is a process leading to the other functions of the module. This is the process by which the RE sells airtime vouchers to the REC.
- *Buy airtime* - also not implemented. It involves the REC buying airtime voucher from the RE.
- *Send airtime* - used by both the RE and the REC to send airtime voucher digits to the UE, thus purchasing electricity.

- *Retrieve electricity* - used by the RE and the REC when retrieving electricity voucher digits from the database accounts in the M-Payment system. The electricity voucher is sent into the database by the UE, after a request is made by either the RE or the REC.
- *Retrieve airtime* - this is used by the EU only and is for retrieving the airtime voucher sent into the database by either the RE or the REC when requesting electricity vouchers.
- *Send electricity* - Once the airtime voucher sent by the RE or REC has been loaded into the UE's pay phones, the EU uses this current module component to send an electricity voucher into the respective subscriber's account in exchange for the airtime voucher he successfully loaded onto his phones.
- *Sell electricity* -this is not a software implementation, but is part of the processes required for software implementation. This component represents the different entities that sell electricity around any town closest to the UE, for example Eskom.

Figure 20 offers a sequential diagram for the purchasing of an electricity voucher; a description of the steps follows thereafter.

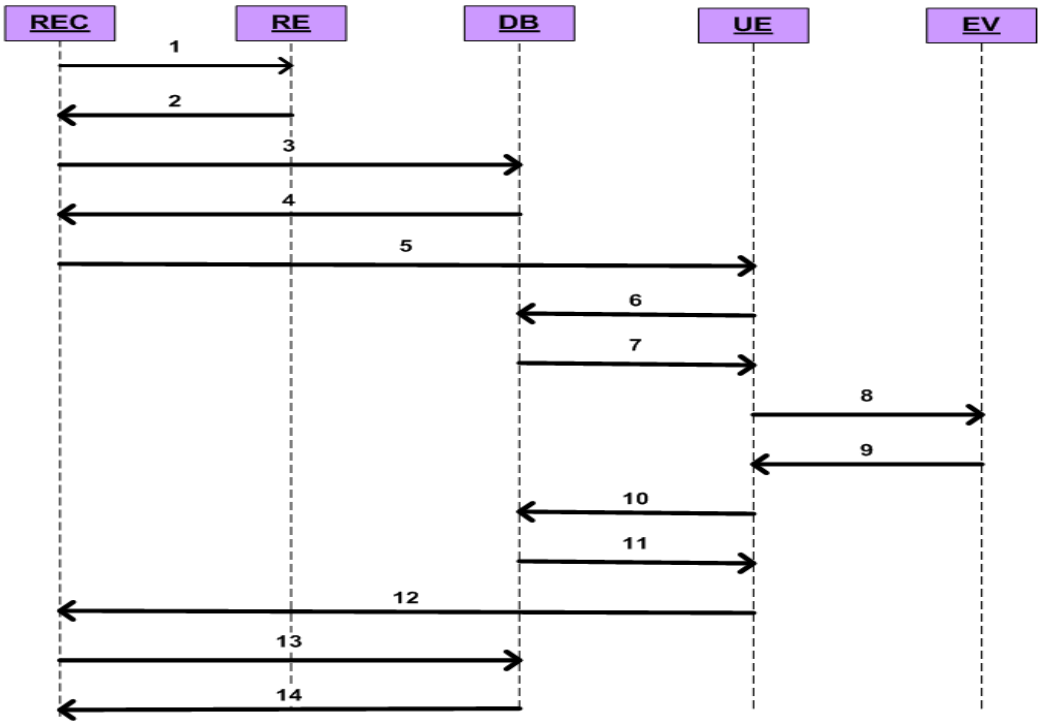


Figure 20: Sequence diagram for the EB system

The following lists the sequential steps of purchasing an electricity voucher by a **REC**, as indicated before. The numbers on the list below correspond to the numbers in the sequence diagram of figure 20.

1. **REC** goes to the local or rural shop in the community to buy an airtime voucher.
2. **RE** gives **REC** the airtime voucher.
3. **REC** launches the M-Payment system on the mobile device, logs in and selects the electricity buying option. He then sends the electricity voucher request information to an **UE**'s account in the database. This information constitutes the electricity meter number, phone number of the **REC**, date on which the request was sent, airtime voucher digits and the equivalent amount of the airtime digits. The various **UE**'s registered with the system are shown as an exclusive list in the electricity buying form of the system application.
4. A response is sent back to the mobile device of the **REC** showing a successful update on the database with the electricity voucher request information.
5. **REC** notifies the **UE** of a request using an SMS.
6. **EU** launches the system application and queries the database for any electricity voucher requests.
7. The **EU** retrieves the airtime voucher from the database and loads the digits into his pay phones.
8. If the airtime voucher is valid, and corresponds to the voucher amount specified, the **UE** goes to the nearest **EV**.
9. **UE** purchases an electricity voucher using the meter number for the **REC**.
10. **UE** updates the electricity account of the **REC** with the electricity voucher digits, equivalent electricity amount and the date of the transaction.
11. **UE** receives an update status response message from the database.
12. If the update was successful the **UE** notifies the **REC**, using an SMS, of the update on his electricity account.
13. **REC** queries the database to retrieve the electricity voucher digits from his updated account.

14. If the account has been updated, the **REC** receives the electricity voucher digits from the database.

4.5 Non-Functional Requirements

These are distinct attributes used to measure the success of the application. They are guidelines governing the operation requirements of the system and its performance as a whole, imposing constraints on the system design. The following are some of the non-functional attributes considered in the implementation of the M-Payment system.

4.5.1 Usability

Ease of use is an important aspect of any software application. To ensure ease of use in the M-Payment system the following were considered:

- Graphical User Interface (GUI) was used for the user interface
- Menu options were turned into lists to reduce typing errors
- Feedback on the success or failure of any particular operation was implemented to notify the user of the status of an operation or task being undertaken at that time
- Navigation within the application was made simple and uniform in the interface layout
- Input of data is done through the use of the mobile device's keypad

4.5.2 Utility

This is the measure of the satisfaction of the users of this software application. If the users express content at the effectiveness and usefulness of the product then the utility aspect is highly rated.

4.5.3 Performance

So many factors contribute to the measurement of the performance of any software application. Performance is dependent on the resources required for the running and operation of the

software. The M-Payment system is a mobile device application. Mobile devices are characterized by less memory and a slow CPU. This has an effect on the speed at which most of the application operations will be carried out, as most involve sending and receiving data over a wireless network. This also has an effect on system response times, which is the time measured from the time a request is sent using an application and the time a desired response is received. The higher the response time the greater the need to improve factors affecting network speed and CPU speed. Threading was used in this application to lessen the chances of deadlocks. At times, applications behave like they are frozen due to resource allocation conflicts, hence the use of threads to avoid these common cases.

4.5.4 Security

A wireless network has a disadvantage that it is more susceptible to attacks than the wired network. This is so because the wireless network can be detected by any unauthorized person with a device that has wireless capabilities, as long as the person is within range of the network. Security becomes an important attribute; its primary function is to control who has access to the resources and who does not. In this application, cryptographic techniques were not implemented. Only password protection and HASH functions were used to protect sensitive data which is sent over the network.

4.6 Conclusion

This chapter discussed the services offered by the M-Payment system. The M-Payment system has two modules in it, the virtual money transfer module and the electricity buying module. The architecture of both modules was discussed in detail, followed by a step-by-step analysis of how some operations are carried out in these modules.

Software requirements are grouped into two branches: functional and non-functional. The functional and non-functional requirements for the system where discussed, with greater emphasis being places on those that were implemented.

CHAPTER 5: IMPLEMENTATION OF THE M-PAYMENT SUPPORTING COMPONENTS

This chapter discusses the implementation of the supporting components of the M-Payment System. Before concentrating on the money transfer module and the electricity buying module, general system implementation will be discussed in this chapter. These include the language options functionality, database structure, split functions and many more. The various code samples will be analyzed and discussed, both for the midlet and the server side implementations.

5.1 Software requirements

To implement the prototype, Java ME technology and WAMP server 2.0 were employed. WAMP - an acronym for Windows, Apache, MySQL and PHP - is a server stack that has Apache web server, MySQL database and PHP scripting language functions. The various components of WAMP offer various services in this implementation, and the following sections highlight some of the important aspects of these components.

5.1.1 Client Side Software

Java ME was used to design and develop the front end system, which is the client midlet application that runs on the mobile device. It is a technology created for small portable devices like cell phones and pagers, offering flexibility and cross platform capabilities.

5.1.2 Middle Tier Software

The middle tier functionality was developed using a collection of PHP scripts to carry out connectivity to the server as well as facilitate data flow to and from the database.

5.1.3 Server Side Software

The server side software provides communication facilities and administrative functions to the backend system. This project made use of WAMP server 2.0 in its implementation, which has a component called *phpMyAdmin* that offers administrative procedures to backend services. With *phpMyAdmin* one can perform server maintenance, browse and manipulate databases and tables. Serving the same purpose of data manipulation and browsing in databases and tables is a software application known as *HeidiSQL*. It is an easy to use interface easily adaptable for the windows operating system used in this implementation. Figures 21 and 22 show the two design views for the *phpMyAdmin* and *HeidiSQL*, showing properties of the database table *banktrans.*

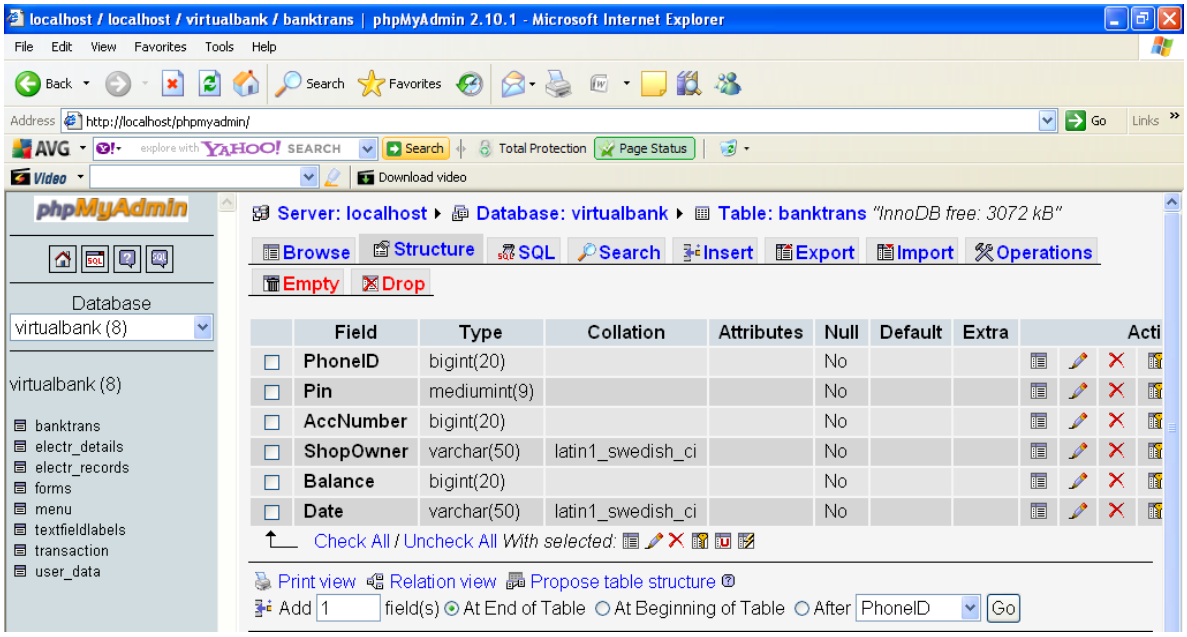


Figure 21: phpMyAdmin design view

The design view for *phpMyAdmin* uses a web browser to navigate the database and its tables, although it does not need internet connectivity for its use.

Below is the same information displayed using *HeidiSQL* which, like *phpMyAdmin*, does not require internet connectivity.

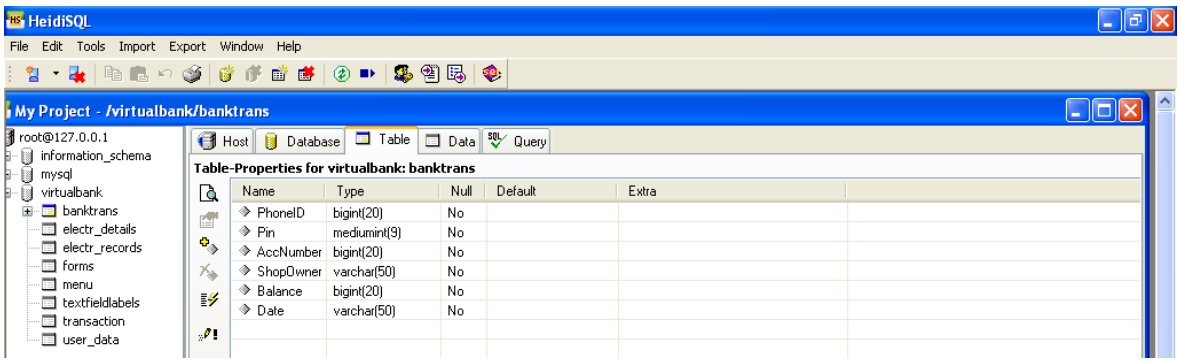


Figure 22: HeidiSQL design view

5.2 Communication between the Midlet and the Server

A midlet has to communicate with the server on a regular basis. There is communication when the midlet requires sending data to or retrieving data from the database tables. In order to do this, the midlet requests a connection to the database through the server.

5.2.1 HTTP connection and Passing of parameters

All the data used in this system is stored in database tables. The midlet has to connect to the server in order to retrieve or store data in the respective tables. The following sample code shows how a midlet requests an HTTP connection to the server side scripts:

```
1.  String parameter;
2.  String url = "http://127.0.0.1/bank/users.php?";
3.  parameter = "name=" + “lerato”;
4.  parameter += "&surname=" + “mpofu”;
5.  System.out.println (url+parameter);
6.  try {
7.          hc = (HttpConnection)Connector.open(url+parameter);
8.          in = hc.openInputStream();
9.          int ch;
10.         while ((ch = in.read ())!= -1) {
11.                 b.append ((char) ch);
12.          }
13. } catch (Exception e) {
14. e.printStackTrace ();
```

Communication is effected through the passing of *parameters* from the midlet to the PHP scripts in the server. These parameters are a combination of strings and numbers extracted from the midlet by the PHP scripts and sent to the database. *Line 2* represents the HTTP address to the location of the PHP scripts in the server. The PHP scripts are in the computer used to develop and test the application, which is why the code uses the loop back address *(127.0.0.1)*. If the client and server are on different machines then the IP Address used should be that of the server. These scripts are stored in a **www** folder automatically created during the installation of the WAMP 2.0 server, and the following is the folder directory of where the PHP scripts are, i.e. in the **bank** folder that was manually created:

```
C:\wamp\www\bank
```

The last part of the HTTP address shows that it is locating a PHP file by name of *users* in the abovementioned **bank** folder.

Lines 3 and *4* show the parameters of *name* and *surname*, which have the respective values of *lerato* and *mpofu.* These values are the ones that are sent to the *users.php* file. The HTTP connection request to the server is done using *line 6*, which connects over the network using the combination of the *url* and the *parameters*. The details of the connectivity to the server and the database are discussed in section 5.2.2 below. *Line 7* opens the input stream for the sending of data through the network connections, and *Line 10* facilitates the reading of the data from the input stream until it is empty.

5.2.2 Connecting to the server and database

In order for the midlet to be able to pass the parameters to the PHP scripts, first a connection to the server has to be established during an HTTP connection request to the server. Once connection has been established between the server and the midlet, the parameters that are to be sent to the database can now be passed on to the respective tables. First, a connection has to be established to the server and database using the following PHP script:

```
1. ?php
2. $con = mysql_connect(“localhost”,”tla”,”mpofu”);
3. if (!$con)
4. {
5.         die('Could not connect: ' . mysql_error());
6. }
7. mysql_select_db("virtualbank", $con);
8. ?>
```

A PHP code exists between the PHP tabs in *lines 1* and *7*. *Line 2* connects to the web server in the machine *localhost*, with a root *tla* and password *mpofu*. If connection fails, the *if* statement in *line 3* is executed, with a connection error “*Could not connect*”, plus a specific error as reported by the *mysql_error ()* part of the statement in *line 5*. When a connection to the *localhost* machine is successful then a connection is done to the database *virtualbank* as shown by *line 7*.

5.3 Database Structure

The database tables created store data for different purposes. The name of the database in the M-Payment system is called *virtualbank.* PHP scripts are used to communicate with the database – entering or retrieving data to or from it. The following section will briefly discuss the different tables in the system and their functions.

5.3.1 Database tables in the M-Payment System

- *user_data* - stores basic subscriber information like username, password, age and date. Information will differ on type of user; for example, a rural entrepreneur will have “admin” data while a subscriber will have “user” data in the *userID* field.
- *Forms* - this table stores the names of the different forms designed in the system. The names are either in English or Xhosa. Choosing either language loads the forms in that language format.
- *Menu* - stores the different menu options, with the table presenting the menus in both English and Xhosa. Just like the *forms* table, the language used is optional.
- *textfieldlabels* - used to store the text field labels, in both Xhosa and English.
- *banktrans* - stores the money transfer module account details. It stores the balance each subscriber has with the different rural entrepreneurs. It is a virtual bank account.

- *transactions* - stores records for all the deposits made to a subscriber's database account.
- *electr_details* - stores the phone number and meter number records for the subscriber, for use when purchasing electricity.
- *electr_records* - stores various electricity purchasing data like meter number, electricity voucher, airtime voucher and many more.

5.3.2 Database Relationships

The M-Payment System database consists of the various tables already discussed in the previous section. The discreet tables are related to each other as certain primary key fields in one table are foreign fields in another table. Figure 23 shows the relationship between all the tables in the *virtualbank* database.

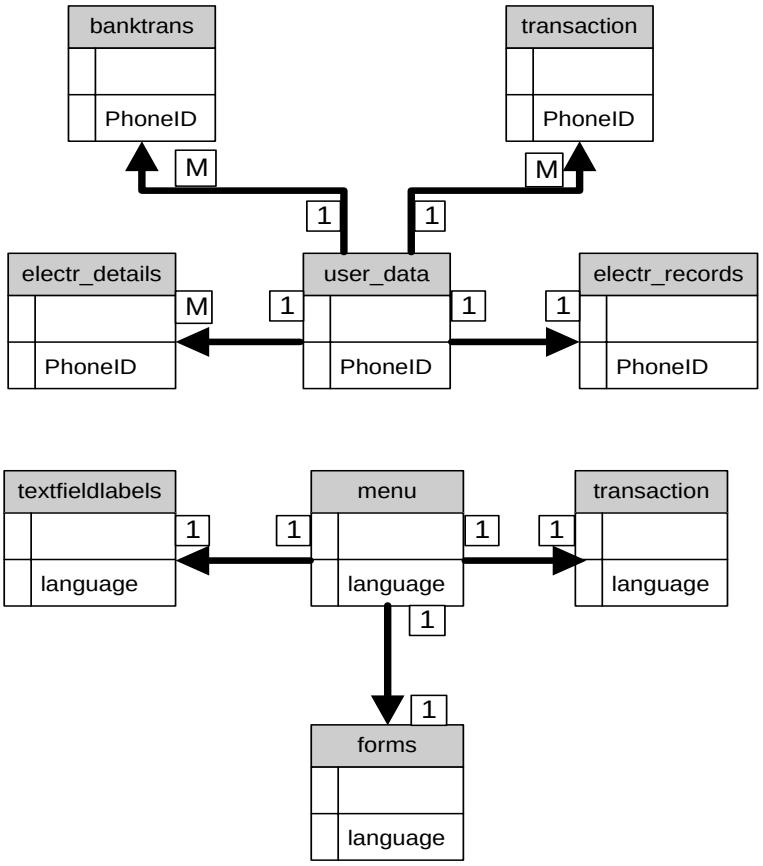


Figure 23: System's Database Relationship structure

The *user_data* table uses the phone number in the *PhoneID* field as its primary key. This primary key is then used to create relationships with other tables in the system that contains the *PhoneID* field as their foreign keys. Figure 23 shows that the *user_data* table has One to Many relationships with three (3) tables and a single One to One relationship with one (1) table. The term 'One to Many' relationship means that one phone number in the *user_data* table can have many records in the respective tables related to it, while a 'One to One' relationship with the *electr_details* means that the user can have many records in the *electr_details* table.

The *menu* table uses the language stipulated in the *language* field as its primary key. This primary key is then used to create relationships with the other tables that contain the *language* field as their foreign key. The *menu* table has 'One to One' relationships with all the other tables shown in Figure 23; this means that a language in the *menu* table can have a single record in the other tables in the various relationships.

5.4 Starting the Midlet

The application offers the option of selecting a language of preference on startup. Since the target users from Dwesa are predominantly Xhosa speakers, this application currently has English and Xhosa language options only to choose from. The language terms are all stored in different tables in the database and are retrieved as soon as a specific language is selected. Figure 24 below shows the screen for selecting the language of preference.

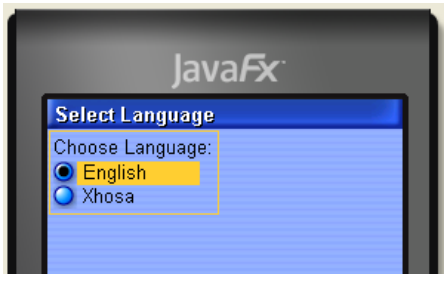


Figure 24: Selecting system language screen

5.4.1 Retrieving Language options from the database

Once a language is selected, the *LanguageClass.java* class program connects to the *getLanguage.php* script in the server and then searches the database tables for records that have the selected language as its key, e.g. Xhosa. It then loads all the terms for Xhosa in an array in the server side PHP scripts.

Below is part of the code for specifying what language is to be used on logging into the system:

```
1.  for(int i=0;i<2;i++) {
2.      if(language.isSelected(i)) {
3.              lang = language.getString(i);
4.              display.setCurrent(loginForm);
5.      }
6.  }
```

Line 1 shows that there are two languages in the list for selection, that is why the values for *i* are 0 and 1 in the *for loop.* The variable *lang* then takes the value of the specified language of choice and is used in the *LanguageClass.java* class as a parameter.

The *LanguageClass.java* class also takes a second variable as a parameter, depending on what is to be retrieved from the database. The choice of the second parameter can be a variable for the retrieval of the following from the database in the preferred language:

- Form names – the parameter sent is *formNames* and has the value "*formNames"*. All form names are retrieved and appear on the respective forms as they are used in the midlet.
- Command button – the parameter sent is *start* and has the value of "*start*". The data loaded when this parameter is used is for the menu command buttons in the midlet.
- Label names – the parameter sent is *labelNames* and has the value "labels". This parameter facilitates the loading (from the database) of the label names used alongside the text fields.

- Account holders - two different parameters can be passed to the middleware. There is an *accnumsBank* parameter used to retrieve account holder names for rural entrepreneurs in the money transfer module; it has a value of *"*accounts*"*. Then there is an *accnumsElectr* parameter used to retrieve names of the urban based entrepreneurs used in the electricity buying module; it has a value of "accounts2". All the names will appear in the forms as they are used in the midlet.

Below is a sample of the code that is used together with the previous language selection sample code, between *lines 3* and *4*. It is used to specify what information has to be retrieved from the database tables on starting up the midlet.

```
1. String formNames = "formNames";
2. LanguageClass langClass = new LanguageClass(lang,formNames );
3. String str = langClass.getResult().trim();
4. split(str,";");
```

Line 2 calls the *LanguageClass.java* class using the parameters for the selected language (*lang*) and the type of data to be retrieved (*formName*s). *Line 3* gets the response from the database as a string (*str*) through the *LanguageClass.java* class. The result is an array of strings retrieved from the database which has to be split into separate strings using the split function called in *line 4*. The distinct strings represent the different form names which are then embedded in their respective forms on login.

Similar code is used when retrieving the label names, account holders names and the command button names. They all use the *lang* parameter and a second specific parameter, as previously described, e.g. *labelNames*.

5.4.2 Array Function in the PHP Scripts

As the data is retrieved from the database, it is put in an array of strings and sent back to the midlet for use. Below is a sample of the code used for creating the array elements. The PHP sample code is in *getLanguage.php*, which takes parameters from the *LanguageClass.java* class.

The code is a sample for the array formed after using the parameters *lang* and *formNames* from the *LanguageClass.java* class.

```
1.  if($str == "formNames"){
2.      $sql="Select * FROM forms where language='$lang'";
3.      $res = mysql_query($sql);
4.      while($result_set=mysql_fetch_array($res)){
5.          $str1 =$result_set['deposit'];
6.          $str2 =$result_set['register'];
7.          $str3 =$result_set['login'];
8.          $str3 =$result_set['mainMenu'];
9.          $str4 =$result_set['meterNumber'];
10.         $str5 =$result_set['deregister'];
11.      }
12.    $arrayMenu = array($str1, $str2, $str3, $str4,$str5);
13.    for ($x = 0; $x < sizeof($arrayMenu); $x++){
14.    echo "$arrayMenu[$x];";
15. }
```

Line 2 is an SQL query that *uses* the *lang* parameter to select all the records in the database table *forms* that are in the specified language of choice. The query is executed using *line 1b*. The use of *msql_fetch_array()* in *line 4* serves the purpose of buffering the results from the database into an array so that one can easily call the required data by its field name. The while loop that starts from *line 4* to *line 11* reads records from the table until there are no records left for the specified query. It assigns the data read from the database to variables *$str1* to *$str5*, as indicated. The data is read from the various fields as shown in the code, e.g. deposit, register and login fields. *Line 12* creates an array of strings, called *$arrayMenu*, using the data from the database table. *Lines 13 and 14* read from the array and print the array back to the *LanguageClass.java* class.

5.4.3 Array split Function

The function below is used to split the array of strings sent from the *getLanguage.php* script into separate strings, and then creates a new array that can be used in the system functions.

The function makes use of a delimiter to separate the strings. In this system the delimiter character used between the distinct strings is the semicolon ";". That is why when the split function is called it takes the array of strings and the delimiter as its parameters.

```
public  String[] split(final String string, final String delim,final int limit) {
    int count = count(string, delim) + 1;
    if (limit > 0 && count > limit) {
      count = limit;
    }
    strings = new String[count];
    int begin = 0;
    for (int i = 0; i < count; i++) {
       int end = string.indexOf(delim, begin);
      if (end == -1 || i + 1 == count)
        end = string.length();
      if (end == 0)
        strings[i] = null;
      else
        strings[i] = string.substring(begin, end);
      begin = end + 1;
    }
    timesSplitUsed++;
    initialiseMenu(strings,aray);
    return strings;
```

5.4.4 Using the New Array

The following shows an example of the array containing form names:

```
s[0] =  DEPOSIT INFORMATION
s[1] =  REGISTER
s[2] =  LOGIN
s[3] =  CASH IN MONEY
s[4] =  CASH IN AMOUNT
s[5] =  TRANSFER DETAILS
s[6] =  TRANSFER MONEY
s[7] =  UPDATE ACCOUNT
s[8] =  UPDATE ACCOUNT
s[9] =  M-PAYMENT SYSTEM MENU
s[10] =  MONEY TRANSFER MENU
s[11] =  ELECTRICITY MENU
s[12] =  AIRTIME VOUCHER
s[13] =  ELECTRICITY VOUCHER
s[14] =  RETRIEVE AIRTIME
s[15] =  RETRIEVE ELETRICITY
s[16] =   METER NUMBER
s[17] =  DE-REGISTER
```

The following code shows a sample of how the new array created by the split function is used to initialize the menu options:

```
public void initialiseMenu(String []s, int arayValue){
    depositForm     = new Form(s[0]);
    registerForm    = new Form(s[1]);
    loginForm       = new Form(s[2]);
    cashInForm      = new Form(s[3]);
    cash            = new Form(s[4]);
    transferRefForm = new Form(s[5]);
    transferForm    = new Form(s[6]);
    userUpdateForm  = new Form(s[7]);
 }
```

The above method takes the split function array output and the array length as its parameters. In this instance, the array *s(arayValue)* contains form names. The above code shows how midlet forms are declared and given the distinct names *s(arayValue)* contained in the array, with each name corresponding to the use of the form. This method is used for the language initialization that was discussed in previous sections, e.g. initializing labels and command buttons in Xhosa. It deals with array elements and, on starting the midlet, initializes the application language component as opposed to hard coding the language strings in the midlet.

5.5 Registering and Deregistering

To be able to use the M-Payment system, a person should be registered and have his details stored in the database tables. Once registered, the person becomes a subscriber and should be able to access the system resources by logging in, or terminating subscription by deregistering. After having started the midlet, and selecting the language of choice, the menu options as shown in Figure 25 become available.

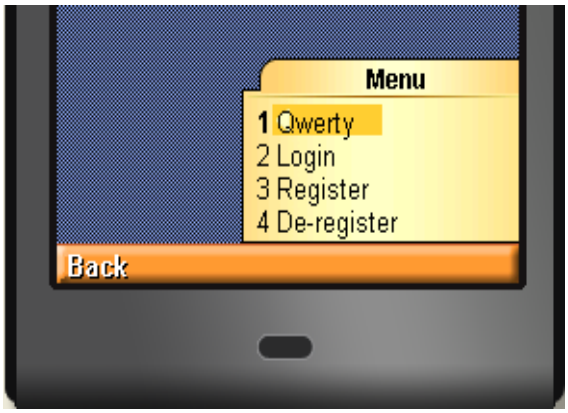


Figure 25: Start Up Menu Options

Menu options available at this stage are: login, register and de-register.

The following is a sample of *Register.java* code for specifying registration parameters to be sent to the *register.php* script which, in turn, enters the data into the database table *user_data*.

```
String parameter;
String url = "http://127.0.0.1/bank/register.php?";
parameter = "rp=" + rp;
parameter += "&lang=" +lang;
parameter += "&rpass=" +rpass;
parameter += "&rpin=" + rpin;
parameter += "&rttle=" + rttle;
parameter += "&rnme=" + rnme;
parameter += "&rsnme=" + rsnme;
parameter += "&rage=" + rage;
```

Table 3 below shows the parameters sent to the database by the midlet during registration. These parameters are subscriber details.

Table 3: Parameters sent on registration

Parameter	rp	lang	rpass	rpin	rttle	rrnme	rsnme	rage	rloc
Description	Phone Number	Language Type	Login Password	Transaction Pin Number	Title	Name	Surname	Age	Location

Figure 26 below shows the registration form screenshot, indicating the details required on registration.

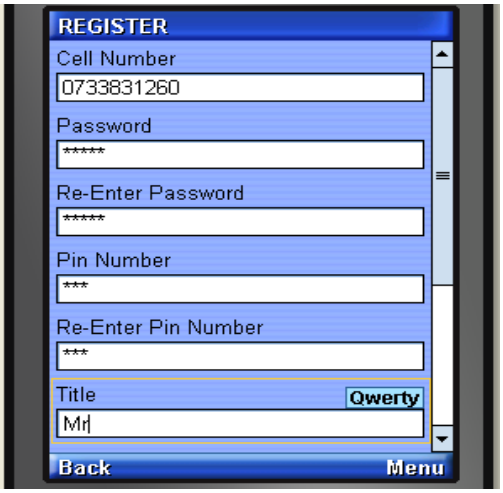


Figure 26: Registration form

The details in the text fields are sent to the *register.php* script as values for the parameters shown in Table 2. The following sample SQL query code in *register.php* inserts the parameter values into the database table *user_data* as part of the registration procedure:

```
1. $query="SELECT * FROM user_data where PhoneID='$rp'";
2. $result = mysql_query ($query);
3. $num = mysql_numrows ($result);
4. $userID = "user";
5. $my_t=getdate (date ("U"));
6. $d = (date ("dS-F-Y-h: i: s A"));
7. if($num == 0){
8.     $s_query = "INSERT INTO user_data (PhoneID, Password, TransPin, Title, Name, Surname, Age, userID, Balance, Location, AccNumber, Date) VALUES ('$rp','$rpass','$rpin','$rttle','$rnme','$rsnme','$rage','$userID',0,'$rloc',0,'$d')";
9. }
```

Lines 1 and *2* check the database for any records or rows of data registered with the cell phone represented by the parameter *rp*. *Line 3* counts the number of records that are registered using *rp* phone number. If *rp* has no account yet then the value for the variable *$num* will be ZERO. Only when *$num* is zero can the registration process continue successfully since a phone number can

only be registered once. *Lines 5* and *6* generate the current date to record as date of registration in the database.

If *$num* equals zero then *lines 7* and *8* insert all the registration details into the database table *user_data*. Once the details are sent to the database, the subscriber can use the login or de-register menu options.

De-registration is the exact opposite of registration. Instead of entering data into the database, deregistration deletes all the data for a specific mobile number from the database. Figure 27 shows a form for entering the phone number and password to initiate deregistration. Figure 28 shows confirmation messages for a successful deregistration.

Figure 27: De-registration form

Figure 28: De-registration confirmation

Below is a sample PHP code from *deregister.php*, which is used to de-register a specific subscriber using the phone number and the password:

```
1.  $query="SELECT Name, Password FROM user_data where PhoneID='$phone' AND Password='$pwd'";
2.  $result=mysql_query ($query);
3.  $num=mysql_numrows ($result);
4.  if ($num ==1) {
5.      mysql_query ("DELETE FROM user_data WHERE PhoneID='$phone' AND Password='$pwd'");
6.      mysql_query ("DELETE FROM banktrans WHERE PhoneID='$phone'");
7.      mysql_query ("DELETE FROM electr_details WHERE PhoneID='$phone'");
8.      mysql_query ("DELETE FROM electr_records WHERE PhoneID='$phone'");
9.      mysql_query ("DELETE FROM transaction WHERE Recipient='$phone'");
10.     mysql_query ("DELETE FROM transaction WHERE Sender='$phone'");
11. }
```

Lines 1 and *2* search for records with the phone number and password specified in Figure 5.6. If there are no errors then the number of records found (*$num*) should be equal to one, as determined by *line 3*. If the records found are equal to one (line 4), then *lines 5* to *10* will delete all the records that have either the phone number and password or just the phone number from all the tables in the database. A confirmation message is then sent back to the midlet. If unsuccessful, an error message will be sent. The function for showing error messages will be discussed in section 5.6.

5.6 Login functionality

A successful log in to the system ensures that the subscriber has access to the midlet resources. The subscriber should have successfully completed registration. The phone number and the password are used when logging in, and are sent to the *login.php* as parameters. Figure 29 shows the form used by subscribers to login.

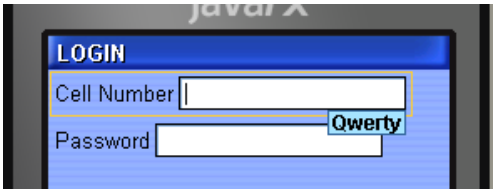


Figure 29: Login form

During login, a string representing user identity is retrieved from the database and sent back to the midlet. That string is then used to determine which system menus should be loaded and which ones should not be for this type of user. The menus therefore differ according to the different types of users. There are two categories of subscribers, namely:

- *User* – this is a normal subscriber with no administrative duties. This subscriber is identified by the string "*user*" in the *userID* field of the *user_data* table in the database.
- *Administrators* - there are two types of administrators, the rural entrepreneur and the urban entrepreneur. The rural entrepreneur has administrative tasks in the money transfer module, and is identified by the string "admin" in the database. The urban entrepreneur has administrative tasks in the electricity buying module, and is identified by the string "adminadmin" in the database table.

The following sample code searches the *user_data* table in the database for the login credentials:

```
1.  $sql="Select * FROM user_data where PhoneID='$phone' AND Password='$pwd'";
2.  $res = mysql_query($sql);
3.  while($result_set=mysql_fetch_array($res)){
4.          $userID =$result_set['userID'];
5.  }
6.  if($userID == "user"){
7.          echo "user";
8.  }else if($userID == "adminadmin"){
9.          echo "adminadmin";
10. }
11. else{
12.         echo "admin";
13. }
```

Lines 1 and *2* search the *user_data* table for the phone number and password entries used on the login form. If the entries exist in the database then *lines 3* and *4* retrieve the user identification string from the record that has the login entries. This string could be any of the three strings which define a distinct subscriber. *Lines 6* to *13* use an *if-else* statement to check the retrieved string, and then print it out using the echo statement e.g. *echo "admin".* The echoed string is used by the midlet to initialize the menu according to the type of user.

5.7 Error Reporting Functions

The system uses functions and a single form to report errors encountered during the running of the system. This reduces the number of forms used and consequently reduces the size of the file. The following diagram offers a depiction of the codes for two functions used to report errors, with a third sample code on how the functions are used in the system.

Function for converting error message and form name to selected language:

```
public String[] convertLingu(String language, String result1, String result2,String formStr1, String formStr2){
        String error, formStr;
        if(lang.equals("English")){
            error = result1;
            formStr = formStr1;
          }else{
            error = result2;
            formStr = formStr2;
          }
        String []res = {error, formStr};
        return (res);
 }
```

The above function takes five parameters – two in English, two in Xhosa, and one representing the language used in the current session. For each set of parameters in the same language, one is an error message and the other is the name of the error form. In the code, *result1* and *result2* are similar error messages in different languages and likewise *formstr1* and *formstr2* are similar form names in different languages.

If the current session is in English, only the English error message and the form name will be used to create an array *[]res*, which is returned by the function. The same applies when the Xhosa language is used.

Function that uses convertLingu and then calls outputForm to create an error form with error message:

```
String [] res;
res = convertLingu (lang,"Invalid number","Inombolo engalunganga","LOGIN ERROR","UKUPHAZAMA EKUNGENENI");
StringItem strItem =new StringItem ("","+res [0]);
```

The above function calls the *convertLingu* function using the parameters shown. If English is the language used in the current session, then the results from *convertLingu* function will be an array *[]res* with the following strings:

res[0] = "Invalid Number"

res[1] = "LOGIN ERROR"

If Xhosa is used then *[]res* will have the following strings:

res[0] = "Inombolo engalunganga"

res[1] = "UKUPHAZAMA EKUNGENENI"

A string item is then created using *res[0]*. The *outputForm* function is then called using the *string item*, *res[1*] and a command button of choice as parameters. This function creates a form with the error message (*string item*), form name (*res[1]*) and the command button.

Function for creating an error form:

```
public void outputForm(String str, Command c, StringItem s){
        formName = str;
        output = new Form(formName);
        output.deleteAll();
        output.append(s);
        output.addCommand(c);
        output.setCommandListener(this);
        display.setCurrent(output);
```

outputForm creates a new form with the form name *str* and error message *s.* The form is then displayed as shown in Figure 30 below.

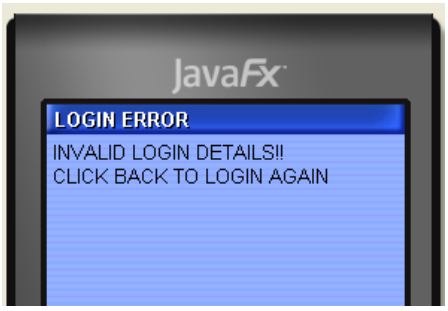


Figure 30: Error reporting form created using functions

5.8 Conclusion

This chapter focused on the general implementation of some components common to both the VMT and EB modules. It covered the database concepts, connectivity between the client application and the server and some basic functionality like logging in, registering and deregistering. Some useful sample codes were discussed so as to highlight how certain operations are structured.

CHAPTER 6: MONEY TRANSFER MODULE IMPLEMENTATION

This chapter discusses how the various functions of the money transfer module were implemented. The module has menus that are common to both a normal user and an administrator, or rural entrepreneur. The various forms are shown and a few sample codes discussed in this chapter. From henceforth we shall call a normal user a 'subscriber' and an administrator an 'entrepreneur'.

6.1 Money Transfer Module Menus

The menu options for both the subscriber and the entrepreneur are similar except for the fact that the entrepreneur has an extra menu function. Figures 31 and 32 show the two menu lists for the two different users.

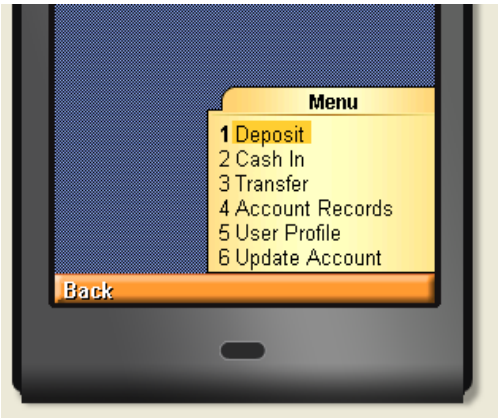


Figure 31: Entrepreneur menu options

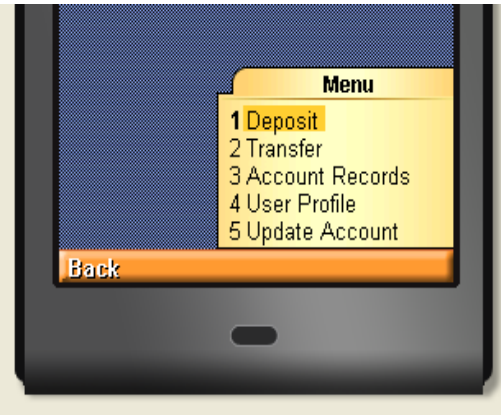


Figure 32: Subscriber menu options

The menu options for the entrepreneur have one more option: the *cash in* menu. Otherwise, all the other menu options are the same. The following sections discuss some of the module's menu options in detail.

6.1.1 Deposit

This option is used by a subscriber based in a town with traditional banking infrastructure. Figure 33 below shows the form used when making a deposit transaction using the M-Payment money transfer module.

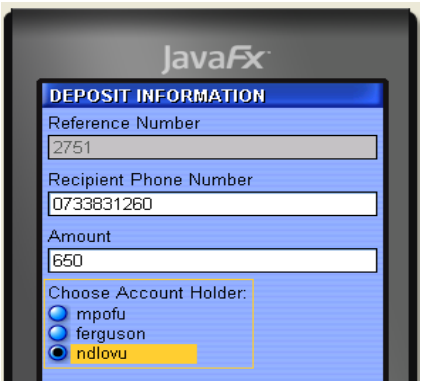


Figure 33: Deposit form

Once the deposit menu is selected the deposit form shows an automatically generated reference number for the deposit. This is a unique number in the database table that identifies a specific

deposit transaction. Below is a sample code from *generateRef.php* for generating the reference number:

```
$random = (rand () %10000);
 $qry="Select * from transaction where Reference='$random'";
 $r=mysql_query($qry);
 while (mysql_numrows ($r)> 0){
     $random = (rand()%10000);
     $qry="Select * from transaction where Reference='$random'";
     $r=mysql_query ($qry);
 }
```

The reference number is generated using the random number function *rand().* This generates a number between 0 and 1, but is multiplied by 10 000 in this system to give the reference number. After generating the reference number the code checks whether the number does not already exist in the database table *transactions.* If it already exists then the value of the variable *$r* should be greater than zero, hence the code in the while loop is executed, generating the reference number until the value of *$r* is zero. That then becomes the reference number for that particular deposit. The code below, from *deposit.php*, further queries the database for the records with the generated reference number before entering deposit data into it; this is done to check that there are no records with that particular reference.

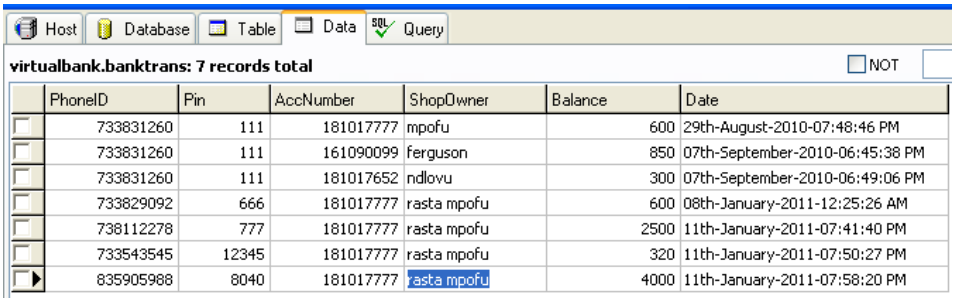
```
$query="Select * from transaction where Reference='$ref' AND Sender
='$sender'";
$result=mysql_query ($query);
if (mysql_num_rows ($result)==0){

    DEPOSIT CODE IN HERE
}
```

The subscriber also has to enter the recipient phone number and the amount to be deposited in the text fields shown in Figure 33. The recipient number has to be of another registered subscriber in the system otherwise the deposit will not be successful.

At the bottom of the form is an exclusive list of entrepreneurs available in the database, whose bank account numbers can be used for the deposit. In Figure 33 above entrepreneur *ndlovu* is selected for the deposit. Clicking the *OK* button updates the database with the specified deposit details. Multiple deposits to a particular subscriber, by different users, into different entrepreneurs' accounts will lead to what is termed *mini-accounts* in the database.

6.1.1.1 Mini-Accounts in the Database

In Figure 33, there are three entrepreneurs present from which an urban based subscriber can choose to deposit money for any rural based subscriber. If three different deposits are made to one subscriber using all three entrepreneurs' account numbers, then three mini-accounts for the rural subscriber will be created in the database. This simply means that the subscriber will now have account balances with three different entrepreneurs from whom he can claim the money, at any time. Figure 34 shows the structure of the *banktrans* table in the database after deposits had been made to three different entrepreneurs for the subscriber mobile phone 073 383 1260.


Host | Database | Table | Data | Query

virtualbank.banktrans: 7 records total NOT

PhoneID	Pin	AccNumber	ShopOwner	Balance	Date
733831260	111	181017777	mpofu	600	29th-August-2010-07:48:46 PM
733831260	111	161090099	ferguson	850	07th-September-2010-06:45:38 PM
733831260	111	181017652	ndlovu	300	07th-September-2010-06:49:06 PM
733829092	666	181017777	rasta mpofu	600	08th-January-2011-12:25:26 AM
738112278	777	181017777	rasta mpofu	2500	11th-January-2011-07:41:40 PM
733543545	12345	181017777	rasta mpofu	320	11th-January-2011-07:50:27 PM
835905988	8040	181017777	rasta mpofu	4000	11th-January-2011-07:58:20 PM

Figure 34: Mini-accounts in banktrans table

From Figure 34, it can be seen that the subscriber with phone number 0733831260 has a mini account with entrepreneur *Mpofu*, where he is owed or has a balance of R600 from all his deposits made into *Mpofu's* account. The same applies to his mini-accounts with *Ferguson* and *Ndlovu*.

6.1.2 Transfer

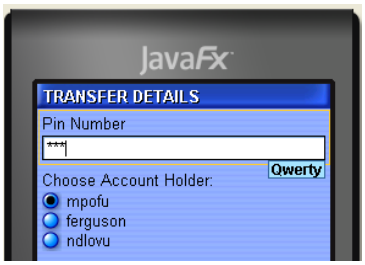


Figure 35: Transfer form

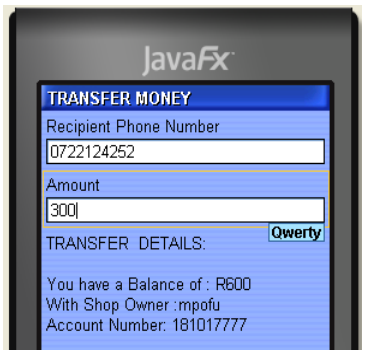


Figure 36: Transferring to another subscriber

Transferring money is an inter-account process of deducting part of a mini-account balance of *subscriber A* and depositing the deducted amount into a mini-account of *subscriber B* with the same entrepreneur. Figure 35 above shows the initial transfer form, which requires a secret transaction *pin number* for a transfer to be processed. The form also shows a list of three entrepreneurs with whom one might have mini-accounts with, from which the transfer can be done.

Figure 36 above shows the subscriber's mini-account balance with *Mpofu*, and requires the subscriber to specify the recipient phone number of the transfer and the amount to be transferred. Once the transfer of R300 is made, the mini-account balance of the subscriber, with *Mpofu*, changes to R300.

6.1.3 Account Records

The account records menu option displays the mini-accounts for the subscribers and entrepreneurs. In addition to these mini-accounts, the account records for the entrepreneur also displays the amounts owed to the subscribers i.e. mini-accounts' balances owed. Figures 37 and 38 below show the two types of account records displays:

Figure 37: Subscriber account records

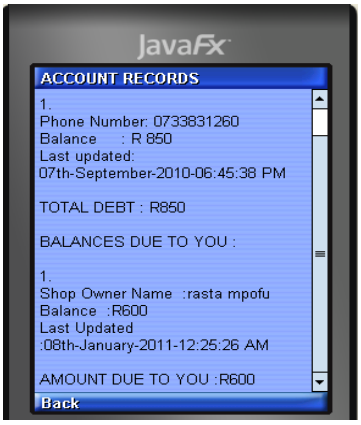


Figure 38: Entrepreneur account records

Figure 37 shows that the subscriber has three mini-accounts with the three entrepreneurs, and is owed a total of R1750.00 by the entrepreneurs. In Figure 38 the entrepreneur owes R850 to the subscriber with the mobile number 0733831260. In addition, the entrepreneur has a mini-account with another entrepreneur, R. Mpofu, who owes him R600.

6.1.4 Cash In

A subscriber can only cash in from any entrepreneur with whom he has a mini-account balance. Cashing in is like a cash withdrawal in traditional banks, only that the bank repository system here is the system database, with the entrepreneur acting as a cashier for payouts to subscribers. As previously mentioned, the *cash in* menu is an entrepreneur-only menu option; hence, the subscriber does not have access to it on logging in. To cash in, the entrepreneur selects the *cash in* menu option. The resulting cash in form requires a subscriber to then enter his phone number and a secret transaction pin, only known by him. This is to avoid the direct manipulation of mini-accounts in the database by unauthorized subscribers and the entrepreneur. Figure 39 below shows the initial cash in form with cash in details already inserted.

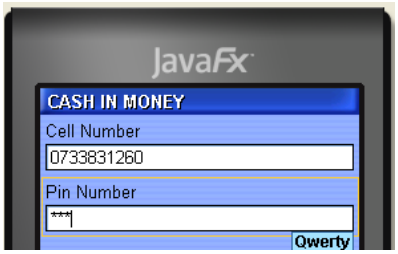


Figure 39: Cash in form

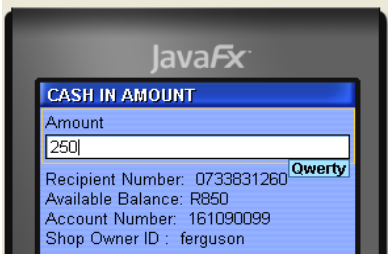


Figure 40: Cashing in an amount

Figure 40 is a form which shows whether the cash in credentials inserted in Figure 39 are valid. It shows that the cash in credentials were correct, hence displaying the mini-account balance details of the subscriber with the phone number 0733831260 in a new form. The form offers an amount field into which the amount to be cashed is inserted. The subscriber can cash in any

amount less than or equal to the available mini balance; in this case, the available balance is R850.

After cashing in R250, the mini-account balance will be R600. Adding all the mini balances for a single subscriber gives an overall account balance which is stored in the *user_data* table. The following sample code shows how this overall balance is updated on cashing in:

```
1.  $qry="SELECT * FROM user_data where PhoneID='$phoneID'";
2.  $res=mysql_query($qry);
3.  $nums=mysql_numrows($res);
4.  if($nums ==1){
5.        while($result_set=mysql_fetch_array($res)){
6.              $bal=$result_set['Balance'];
7.        }
8.       $newBal = $bal - $amount;
9.        mysql_query("UPDATE user_data SET Balance = '$newBal' WHERE PhoneID = '$phoneID'");
10. }
```

We will use the subscriber 0733831260 cashing-in, in Figures 39 and 40 above. *Lines 1* and *2* select all the fields from the *user_data* table where the phone number is equal to 0733831260. *Line 3* equates a variable *$nums* to the count of the number of records in the table that have the subscriber phone number, which should always be equal to 1. *In line 6* if *$num* is equal to one then the variable *$bal* is given the value of the entry in the *Balance* field of the set of records found. *$bal* is the overall balance (R850) after adding all the mini-accounts of the subscriber. In *line 8* a new balance *$newBal* is calculated, deducting the amount *$amount* to be cashed in from the original balance *$bal*. In this case, *$amount* is equal to R250. *Line 9* then updates the *Balance* field in the *user_data* table with the new balance *$newBal* (R600)

6.1.5 User profile

This menu option outputs the basic subscriber information or data on the profile form. The data is just basic, and is that which was specified during registration.

The following sample code is used to query the database for user data:

```
$query="SELECT*FROM user_data where
PhoneID='$phoneID'";
$result=mysql_query($query);
$num=mysql_numrows($result);
if($num ==0){
while($result_set=mysql_fetch_array($result)){
      $title=$result_set['Title'];
      $name=$result_set['Name'];
      $surname=$result_set['Surname'];
      $age=$result_set['Age'];
      $location=$result_set['Location'];
      $phone=$result_set['PhoneID'];
      $acc=$result_set['AccNumber'];
      $bal = $result_set['Balance'];
   }
```

It queries the database table *user_data* for records with a certain value of a phone number *$phoneID*. If the number of rows of such a record equals one i.e. *$num* equals one, then the data under the fields shown in the while loop is retrieved and given to specified variables, e.g. variable *$title* holds the data from the *Title* field. The sample code below then prints out the variables containing the subscriber data retrieved from the *user_data* table.

```
echo "Title              :".$title."\n";
echo "Name               :".$name."\n";
echo "Surname            :".$surname."\n";
echo "Age                :".$age."\n";
echo "Location           :".$location."\n";
echo "Phone Number   :0".$phone."\n";
if($acc != 0){
  echo "Account Number:".$acc."\n";
}
```

Subscribers have an account number equal to zero. If the value of the account number field *AccNumber* is not equal to zero then it is also printed out, as shown in the *if* statement above.

Figure 41 below shows the resulting profile output for an entrepreneur:

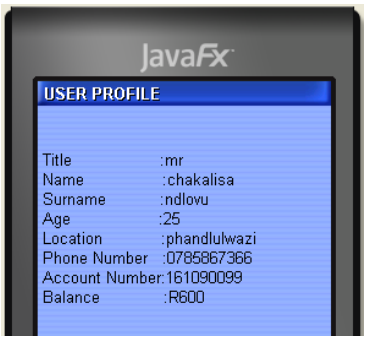


Figure 41: User profile

6.1.6 Update

This function updates the basic subscriber or entrepreneur information in the database. It does not update sensitive data related to money, as it maintains the data integrity of the application. The entrepreneur has the option of updating his account number in case of a change in banking details. After a successful update, the database will contain the latest information on the user. Figure 42 shows entrepreneur data fields that can be updated in the database:

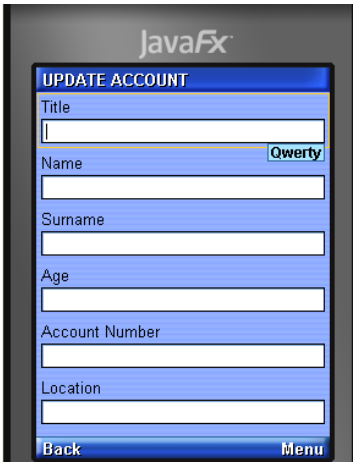


Figure 42: Update form

6.2 Conclusion

This chapter discussed the implementation of the various components in the money transfer module. There are various menu options available to the subscriber and entrepreneur. Most of these functionalities are common to both user types, but the entrepreneur has an additional *cash in* menu option. Code samples and forms for the various menus were discussed and a brief description of their functionality was covered.

CHAPTER 7: ELECTRICITY BUYING MODULE IMPLEMENTATION

This chapter focuses on the implementation of the electricity buying module. It analyses the various menu options available, whilst discussing the corresponding code samples and forms used. The module was designed to be used by two types of users: the ***subscriber*** *and the urban* ***entrepreneur****.*

7.1 Electricity Buying Module Menus

The module is designed to be used by the entrepreneur and the subscriber. Menu components are different for the two users as they perform different tasks in the electricity buying process. Figures 43 and 44 show the menu options available in the system:

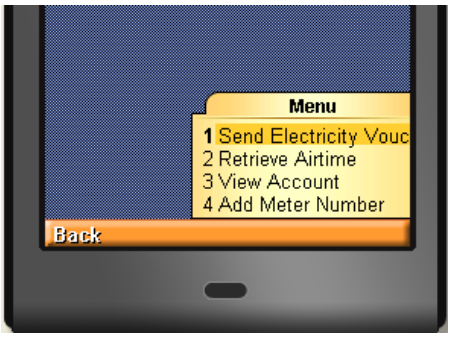


Figure 43: Entrepreneur menu options

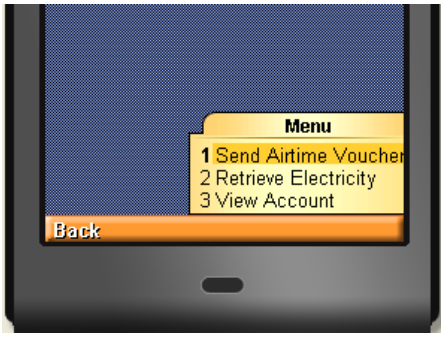


Figure 44: Subscriber menu options

An entrepreneur can do the following:

- Send electricity voucher - this is when the entrepreneur updates the subscriber's electricity account in the database with electricity voucher digits.
- Retrieve airtime – the entrepreneur retrieves airtime voucher digits, sent by a subscriber, from the database.
- View account – the entrepreneur views his account to see his outstanding debts.
- Add meter number - this function allows the entrepreneur to create an electricity account for a user using the subscriber's meter number and phone number.

A subscriber can do the following:

- Send airtime voucher – the subscriber sends airtime voucher digits to the database for retrieval by the entrepreneur, prior to buying the electricity voucher for him.
- Retrieve electricity – the subscriber retrieves the electricity voucher, purchased for him by the entrepreneur, from the updated database.
- View account – the subscriber views his account status to see any transaction updates.

7.2 Description of the Menu options

The following sections discuss the various electricity buying menu options in detail. In order to deepen the understanding of how the system works. The discussion will follow the order in which the menu options are used in the buying process.

7.2.1 Add Meter Number

To use the electricity buying module, the subscriber must first register his meter number with the urban based entrepreneur. Once this has been done the subscriber can now have access to the electricity menu options. Figure 45 below shows the mobile interface used by the entrepreneur to register a subscriber.

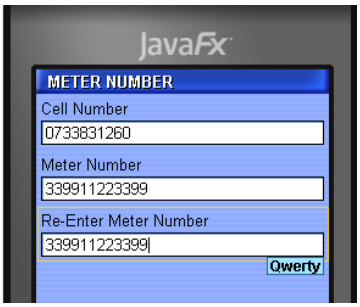


Figure 45: Add meter number form

The form requires the entrepreneur to add the subscriber's mobile number and meter number, all to be sent to the database. On sending these details to the database, a PHP script checks if the user trying to create an account is an entrepreneur. The following sample code from *addMeterNumber.php* does the verification:

```
$query="SELECT * FROM user_data where
PhoneID='$vendorPhone'";
$result = mysql_query($query);
if (mysql_num_rows($result)==1){
    while($result_set=mysql_fetch_array($result)){
        $userID=$result_set['userID'];
 }
```

The code first queries the table *user_data* for the records containing the currently logged in user's mobile number *$vendorPhone*. If there is exactly one record, as expected, then the user identity string is retrieved from the table. This can be any of the following strings:

- *user* - subscriber
- *admin* - rural entrepreneur (money transfer administrator)
- *adminadmin* - urban entrepreneur (electricity buying administrator)

The following sample code from *addMeterNumber.php* then uses the retrieved string to clarify the type of user currently trying to create an electricity account. If the user identity string is "adminadmin" the entrepreneur is granted the right to create an account for the subscriber.

```
if ($userID=="adminadmin" && $phoneID! = $vendorPhone){
    $my_t=getdate (date ("U"));
    $d = (date ("dS-F-Y-h: i: s A"));
    $s_query = "INSERT INTO electr_details (PhoneID,
MeterNumber) VALUES ('$phoneID','$meterNum')";
```

The subscriber phone number and meter number are entered into the *electr_details* table using the *$s_query* statement in the code. The subscriber can now use the system to buy electricity.

7.2.2 Send Airtime Voucher

This option allows a subscriber to send airtime voucher digits to the entrepreneur through the use of the system database. Figure 46 shows a form used to send the airtime voucher digits to the database.

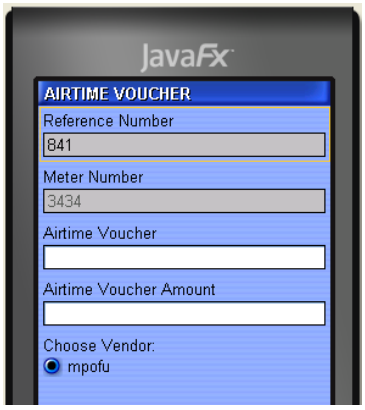


Figure 46: Send airtime voucher form

The reference number is automatically generated in *generateElectrRef.php* and inserted in the corresponding reference number field on the form. The meter number is also retrieved from the *electr_records* table and inserted in the corresponding meter number field on the form. The following code is used to retrieve the meter number from the table and sending it to the midlet where it is inserted in the text field, as shown in Figure 46.

```
$query="SELECT * from electr_details where
PhoneID='$phone'";
$result=mysql_query ($query);
$num=mysql_numrows ($result);
if($num ==1) {
  while ($result_set=mysql_fetch_array($result)){
      $meterNum =$result_set['MeterNumber'];
   }
  echo $meterNum;
```

The above code checks for records in the *electr_details* table containing the phone number (*$phone)* of the currently logged in subscriber. If the number of records found is exactly one, i.e. *$num = 1*, then the meter number is retrieved from the *MeterNumber* field in the table, and its value given to a variable *$metreNum*. The meter number is then printed out to be used by the midlet. If *$num* is not equal to one it means the subscriber is not registered to use the electricity module, prompting the message "No electricity Meter Number" to be sent to the midlet.

Below the form is a list of all the currently available urban entrepreneurs to whom you can send airtime voucher digits to. In Figure 46, there is currently only one entrepreneur, *Mpofu*.

7.2.3 Retrieve Airtime Voucher

To retrieve airtime voucher digits from *electr_records*, the entrepreneur needs to enter the reference number of the record. Figures 47 and 48 below show the forms for entering the reference number and showing the airtime voucher retrieval results respectively.

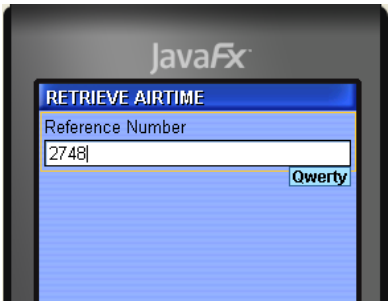


Figure 47: Retrieve airtime form

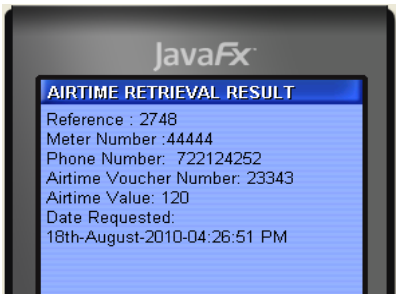


Figure 48: Airtime retrieval results

The following sample code is used to search for records with reference 2748 from the *electr_records* table and then output the results as shown in Figure 48.

```
$mq= "Select * from electr_records where Reference='$ref'";
$myq =mysql_query($mq);
while($result_set = mysql_fetch_array($myq)){
        $ref2=$result_set['Reference'];
        $metNum=$result_set['MeterNum'];
        $fone =$result_set['PhoneID'];
        $pin =$result_set['ElectrVoucher'];
        $airVal =$result_set['ElectrValue'];
        $d =$result_set['ElectrDate'];
}    echo "Reference: ".$ref2."\n";
            echo "Meter Number :".$meterNum."\n";
            echo "Phone Number:  ".$fone."\n";
            echo "Electricity Voucher Number: ".$pin."\n";
```

The structure of the code will not be discussed any further as it closely resembles previously analyzed code samples. Take note that for the output in Figure 48 the value of the reference 2748 will be passed on to the reference parameter *$ref* used in the code.

7.2.4 Send Electricity Voucher

After retrieving the airtime voucher digits and loading it in his pay phones, the entrepreneur purchases electricity for the subscriber in any electricity vendor. He then sends the electricity voucher digits to the subscriber by updating the subscriber's electricity account in the database.

Figure 49 below shows the form used to send the electricity voucher digits in order to update the subscriber's electricity account in the database.

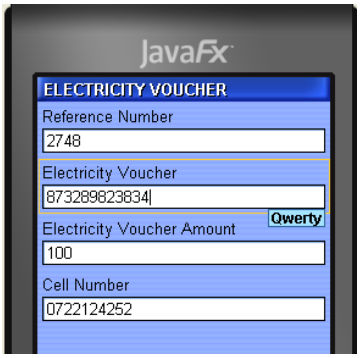


Figure 49: Sending electricity voucher form

The form requires the entrepreneur to enter the reference number for the airtime voucher (that was sent by the subscriber requesting for electricity), the electricity voucher digits, electricity voucher amount and the mobile number of the subscriber.

The sample code from the *sendElectrVoucher.php* script below is used to query the *electr_records* table for the reference number and the mobile number before updating it with the electricity voucher details.

```
$mq= "Select * from electr_details where PhoneID='$cellnum'";
$myq =mysql_query ($mq);
if (mysql_num_rows($myq)==1) {
        $query="SELECT * from electr_records  where PhoneID='$cellnum' AND Reference =
'$re'";
        $result=mysql_query($query);
        $num=mysql_numrows($result);
       if($num ==1) {
                        while($result_set=mysql_fetch_array($result)) {
                        $meterNum =$result_set['MeterNum'];
                         }
```

The code first queries the table *electr_details* for records with phone number *$cellnum* of the subscriber who requested the electricity voucher. If the number of records found is one, the meter number is then retrieved from the *MeterNum* field and its value given to the variable

$meterNum. The following code shows how the database is then updated with the electricity voucher using a query statement in the *sendElectrVoucher.php* script:

```
$my_t=getdate (date ("U"));
$d = (date ("dS-F-Y-h: i: s A"));
 if ($meterNum> 0) {
    $sql="UPDATE electr_records SET ElectrVoucher = '$elecPin', ElectrValue = '$elecVal', ElectrDate
= '$d' WHERE Reference = '$re'";
```

The code above first generates the current date to be used in the update query. If the meter number retrieved in the previous code is not less than zero then an update query is executed to send the electricity details to the *electr_records* table.

The value of the electricity need not be the same as the value of the airtime that was initially sent in request for the electricity. This will be discussed in detail in the sections to come, focusing on the benefits of the M-Payment system.

7.2.5 Retrieve Electricity Voucher

Once the entrepreneur has updated the subscriber's electricity account with the electricity voucher digits, the subscriber has to retrieve those digits using this electricity menu option. The reference number, generated when sending airtime voucher digits to database, is used to retrieve the electricity voucher digits here. Figure 50 shows the form used to enter the reference number in order to retrieve the electricity voucher, and Figure 51 shows the results of a successful retrieval.

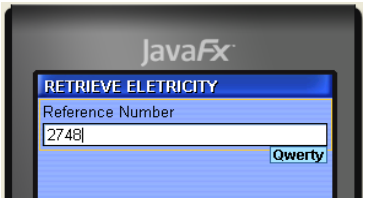


Figure 50: Retrieving electricity voucher form

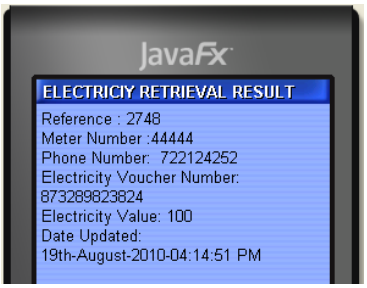


Figure 51: Electricity voucher retrieval result

The code samples of the above functionality will not be discussed as they closely resemble those for the retrieval of the airtime voucher by the entrepreneur.

7.2.6 View Account

The *view account* menu option allows both the subscriber and entrepreneur to view the outstanding electricity transactions. Only the electricity account records that have not been updated with the electricity voucher digits by the entrepreneur are displayed as the results on the form.

Figures 52 and 53 show the two different views for the electricity account:

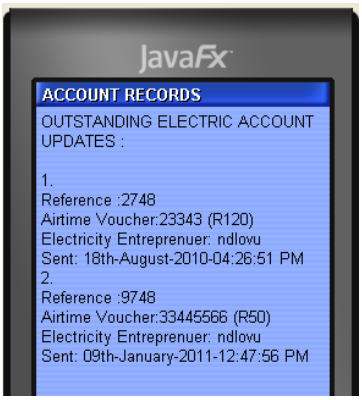


Figure 52: Subscriber records view form

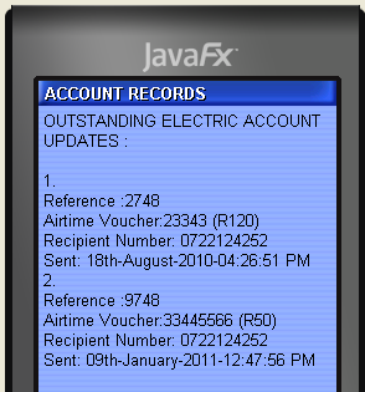


Figure 53: Entrepreneur records view form

The details shown for both are the same, with only the third line of each record differing. For the entrepreneur, it states who the electricity voucher recipient is, while in the subscriber's output it states the name of the entrepreneur who is supposed to update the table with the electricity voucher.

Figure 52 shows that the subscriber has two requests for an electricity voucher that are yet to be processed by the entrepreneur. The first request used an airtime voucher of R120 while the second used a R50 airtime voucher. Both requests were sent to a single entrepreneur, *Ndlovu*, on different dates. Figure 53 shows that entrepreneur *Ndlovu* still owes subscriber 0722124252 two electricity voucher updates for his two requests that were made, as previously stated.

7.3 Conclusion

This chapter focused on the implementation of the menu options available in the electricity buying module. The menu options are different for the subscriber and entrepreneur but complement each other in some operations. This module is independent of the money transfer module; it is an add-in application where the entrepreneur first has to create an account for an already registered subscriber. Once the account has been created the vouchers can be sent to and from the subscriber and entrepreneur as they trade the airtime voucher digits for the electricity voucher digits.

CHAPTER 8: EXPERIMENTS AND RESULTS

This chapter highlights the experiments undertaken in the M-Payment system in order to conclude certain aspects of the system. The results obtained provide an overview of some of the most important attributes which the system possesses and how the system can be improved accordingly.

8.1 Test Objective

The sole purpose of the experiments or tests that were carried out was to evaluate the usefulness and usability of the M-Payment system, hence obtaining detailed feedback on certain attributes constituted in the tests. User experience was also evaluated using questionnaires and interviews to get feedback; this was then used to reach certain conclusions and effect the implementation of certain strategies to improve the system.

8.2 Test Sample and Environment

The tests were conducted in two places. The total sample size of 15 constituted of 10 participants from the rural community of Dwesa while an additional 5 were undergraduate students from the University of Fort Hare. Table 4 below shows information regarding the test participants.

Table 4: Sample Demography

Participant	Gender	Age	Occupation Status	Computer Literacy Level	Mobile Device Familiarity Level
1	Male	15-20	Unemployed	Average	Average
2	Male	21-30	Student	Very Good	Good
3	Male	21-30	Student	Good	Good
4	Male	21-30	Employed	Good	Good
5	Male	21-30	Employed	Average	Average
6	Male	31-40	Unemployed	Average	Average
7	Male	41-60	Pensioner	Poor	Average
8	Female	21-30	Student	Average	Average
9	Female	21-30	Student	Very Good	Good
10	Female	21-30	Student	Very Good	Good
11	Female	21-30	Employed	Good	Good
12	Female	31-40	Employed	Good	Good
13	Female	31-40	Employed	Good	Good
14	Female	41-60	Employed	Average	Average
15	Female	41-60	Pensioner	Poor	Average

8.3 Evaluation Techniques

Evaluation of the system is a process that follows after system development has been done. It is not a once-off practice, hence the importance of the feedback to implement further reinforcements on the system and carry out more evaluation techniques. In this project the techniques used involved some briefing on the application in general and the purpose of the tests. Then, demonstrations on the use of the application were done for the benefit of the participants; these ranged from starting the application to navigating the menu options. After the demonstration, participants were each given a chance to use the system, and certain test results were recorded in the process. A questionnaire was given to each participant that had questions based on their basic information and the application in general. Most of the questions in the questionnaire had the following response options to choose from:

- Yes
- No
- Not sure
- Maybe
- Other

The above answer options helped in narrowing down the responses and making it simpler to group and analyze the results.

8.4 Measurement Metrics

To evaluate the system, certain measurement metrics were considered which acted as a guideline in analyzing the results of the tests. The following section discusses certain metrics used in certain tests.

8.4.1 Usability metrics

To measure usability the following metrics were considered;

- Effectiveness
- Efficiency

8.4.1.1 Effectiveness

This test was designed to assess the extent to which or level at which the participants used the system functionalities in an attempt to complete certain tasks. The tests that were carried out concentrated on:

- *Task Completion* - measured the percentage of participants who were able to complete certain tasks in the system.
- *Assisted menu navigation* - this measured the percentage of participants who needed assistance in navigating the system menu and other functions.

8.4.1.2 Efficiency

This is a measure of the effectiveness of the system and we used the following test:

- *Task time* - the time taken to complete certain tasks.

8.4.2 User Acceptance Metrics

This test was solely based on the user's feelings towards the entire project, including its concepts and perceptions. The following tests were used to analyze user acceptance levels:

- *Usefulness* - measures how the user rates the project in terms of its usefulness to him and the community at large.
- *Interest in the system* - measures the levels of appreciation of and interest that users have in the entire application.

8.5 Data analysis

Once the data had been collected, through the use of a questionnaire, it was grouped then analyzed in order to come to certain conclusions using the resulting information. The ensuing sections of this chapter summarize the results of the tests that were conducted.

8.5.1 Demography

This offers a gender analysis of the participants. Figure 54 below shows that the survey had more female than male participants. There are 8 females and 7 males, with percentages shown in the data labels.

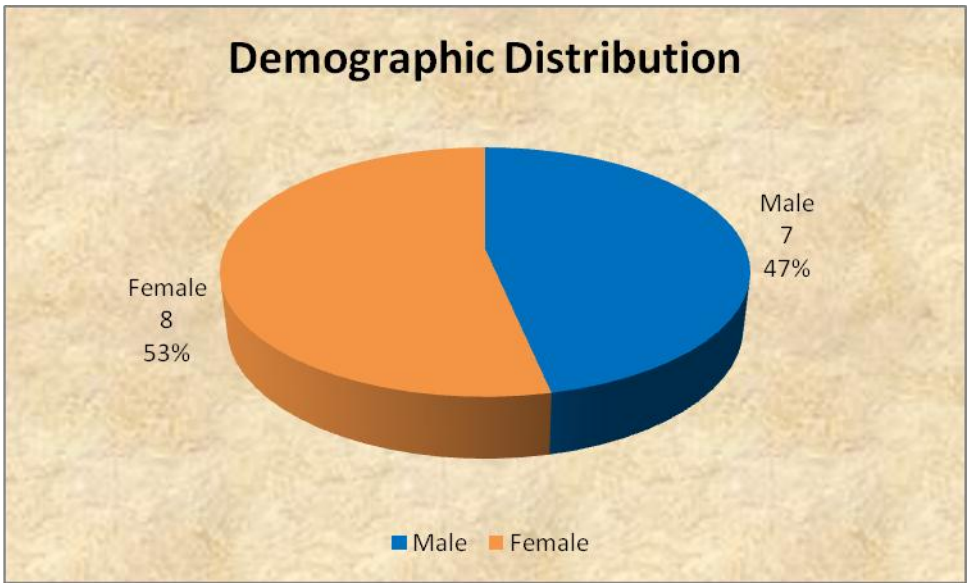


Figure 54: Demographic distribution

8.5.2 Age Groups

The age group analysis collates the data gained from the participants by using a defined range of ages. Figure 55 below shows the distribution of ages in the sample used.

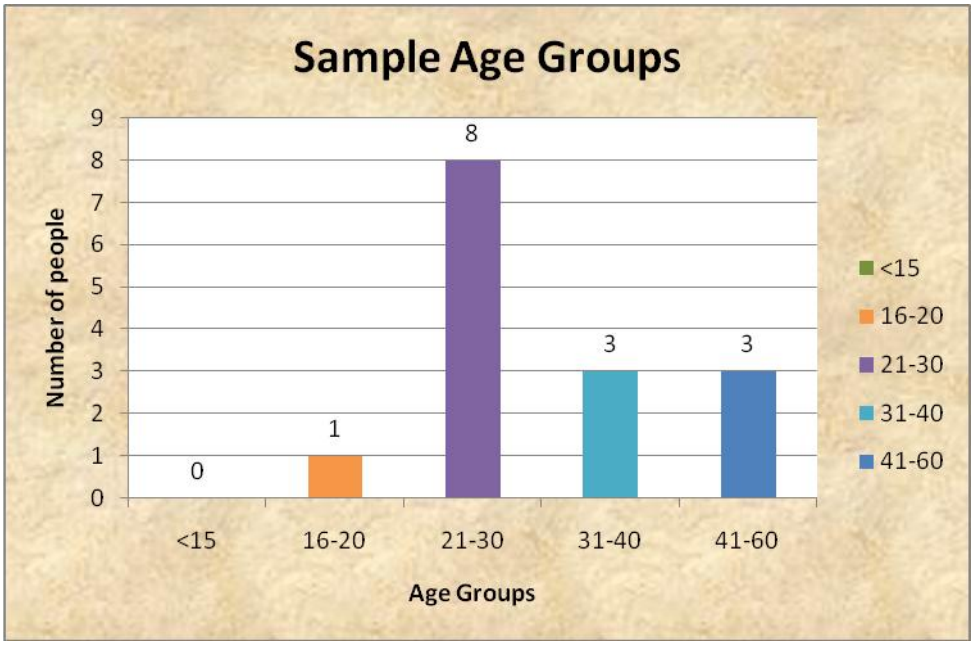


Figure 55: Age distribution results

8.5.3 M- Commerce Knowledge

This test analyzed the participants' response to their knowledge on the use of mobile devices, like cell phones, in carrying out mobile commerce transactions e.g. using mobile device to buy airtime from First National Bank (FNB).

Figure 56 shows the results of the grouped data. 8 participants (53%) had knowledge of M-Commerce, 3 (20%) had no knowledge of the subject at all, while 4 (27%) were not sure if they knew or did not know what M-Commerce is about.

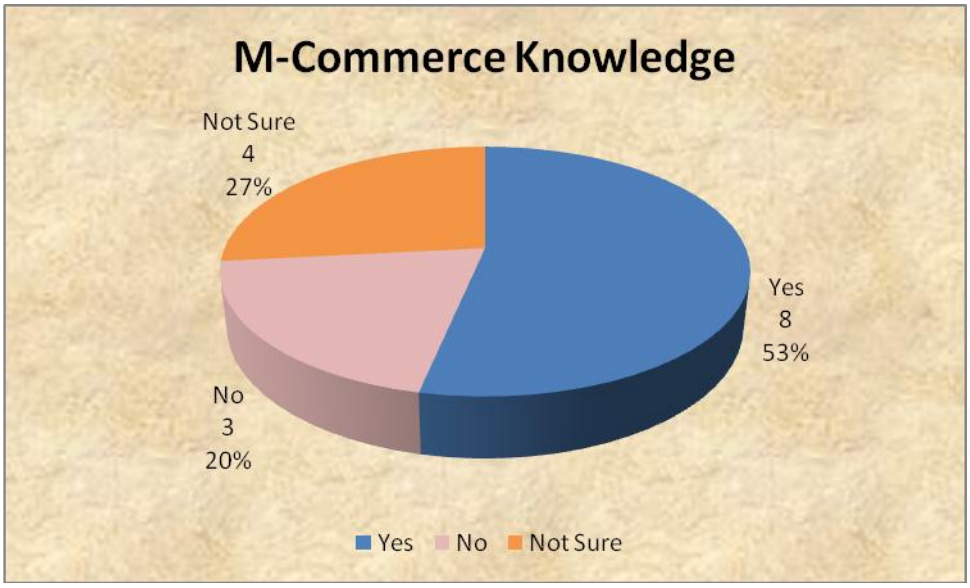


Figure 56: M-Commerce knowledge analysis

8.5.4 WAP Usage

The use of WAP by the participants was also assessed, and the results are represented in the graph below:

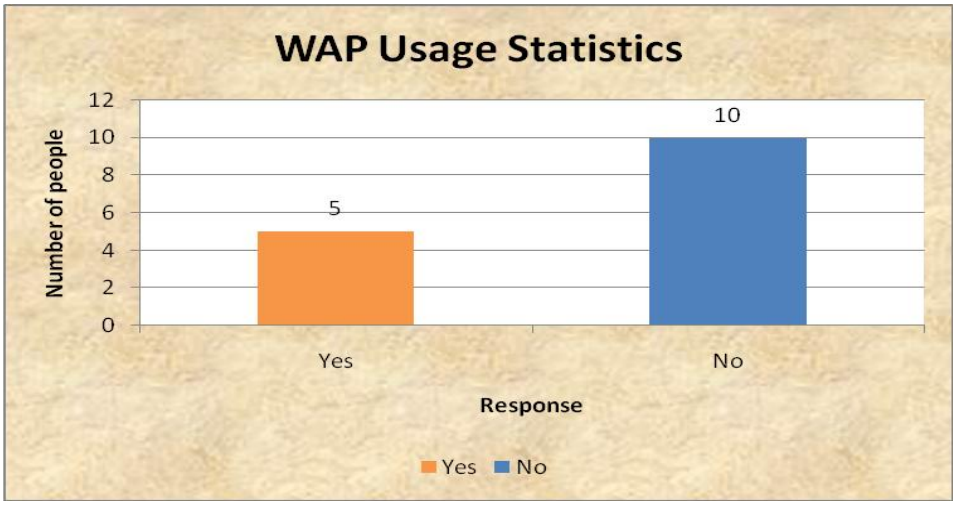


Figure 57: WAP usage statistics

WAP usage in applications is not a common technology in the rural regions, as is proved by the results in Figure 57. Only the students from the university responded positively to the use of WAP technology.

8.5.5 Access to Information Technology (IT)

This analysis grouped data according to the type of technology the participants had access to on a daily basis. There are traditional ITs like radios and televisions, and there are the latest technologies that include the internet and mobile devices. Figure 58 below shows the levels of access to certain IT technologies.

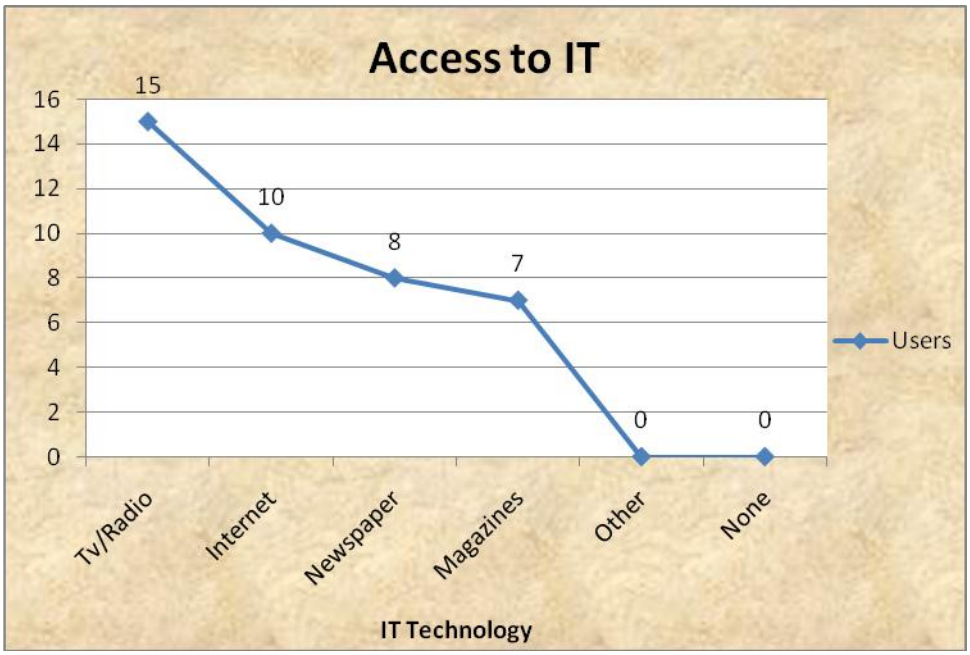


Figure 58: IT access statistics

As evident, the participants have a higher level of access to radio and television ITs (100%), with the magazines having the lowest level of access (46.7 %). Internet access is second highest (66.7), and this is attributed to the fact that some of the schools are benefiting from the networks which have been set up in them. These networks allow people from the local community to gain access to the internet.

8.5.6 Usability

The results will be analyzed separately, according to the distinct metrics used.

8.5.6.1 Effectiveness

Task completion - users were asked to complete various tasks on the use of the application; the aim of this was to assess their capability to carry out those tasks to completion. 13 (87%) participants successfully completed the tasks, while the other 2 (13%) did not manage to do so. Figure 59 below shows the results for the task completion test:

Figure 59: Task completion results

Assisted menu navigation - this tests and records the number of times each user was assisted while navigating through the different menus. This is a real test of the understanding and ease of use of the system by the participants. Every participant was assisted, with the pensioners getting the highest number of assists than anyone else. Figure 60 below shows the variations in the assists for the participants.

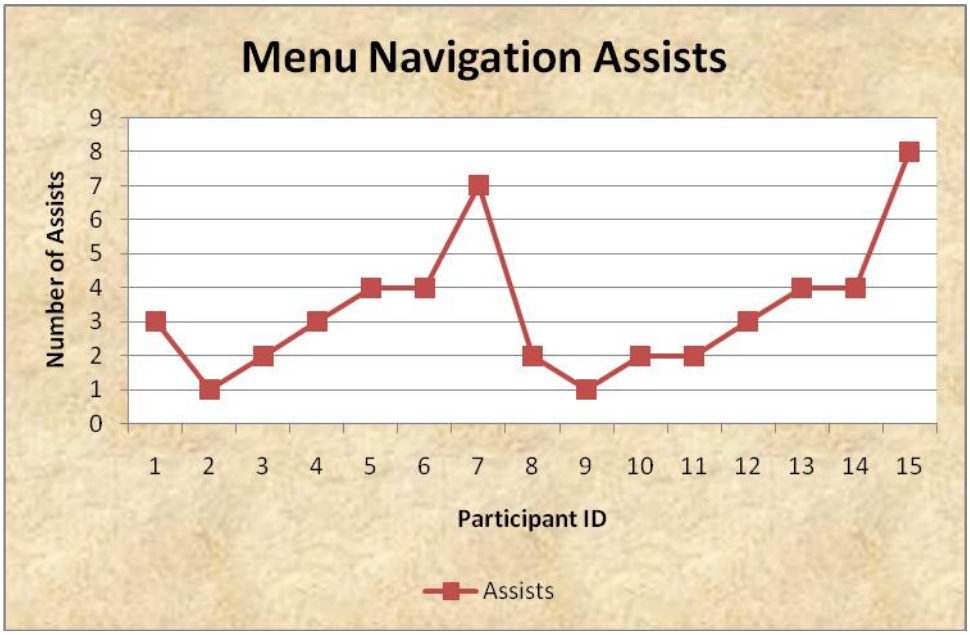


Figure 60: Assisted menu navigation results

The results show that the highest number of assists per participant was 8, with the lowest being 1. The average assist per participant was approximately 3 during the menu navigation task.

8.5.7 User Acceptance

The following metrics were used and results were analyzed for the user acceptance test.

Usefulness - participants had different opinions on the application. Nonetheless, the results showed that most of the participants valued the usefulness of this application regardless of the complexity it might initially present to them.

Figure 61 shows the results for usefulness, as analyzed from the responses obtained. 10 participants (67%) said they find the application useful to them and the community, while 4 (27%) were not sure if it is useful or not. Only one person said the application is useless in solving their problems.

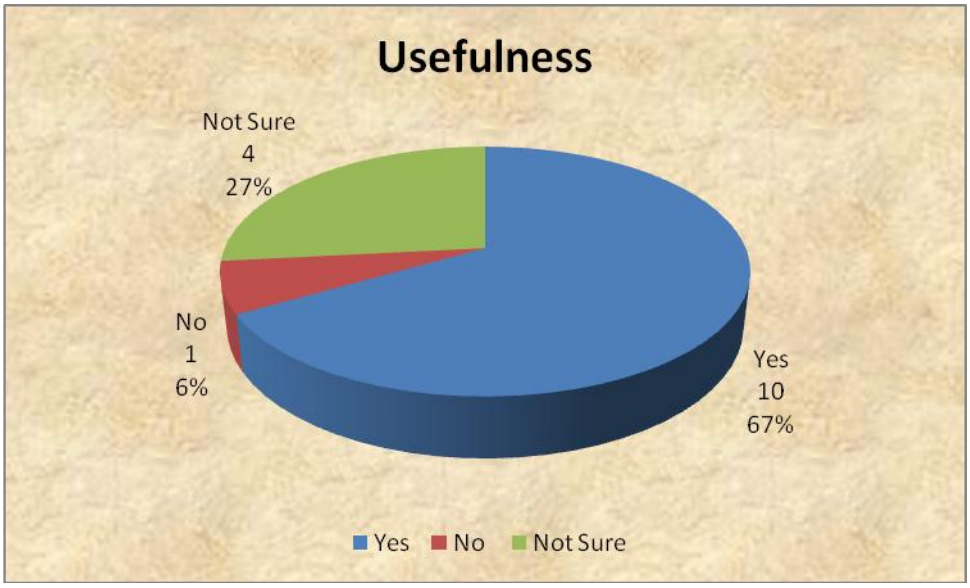


Figure 61: Usefulness results

Interest in the system - Figure 62 shows results of how the participant's interest in the system varies.

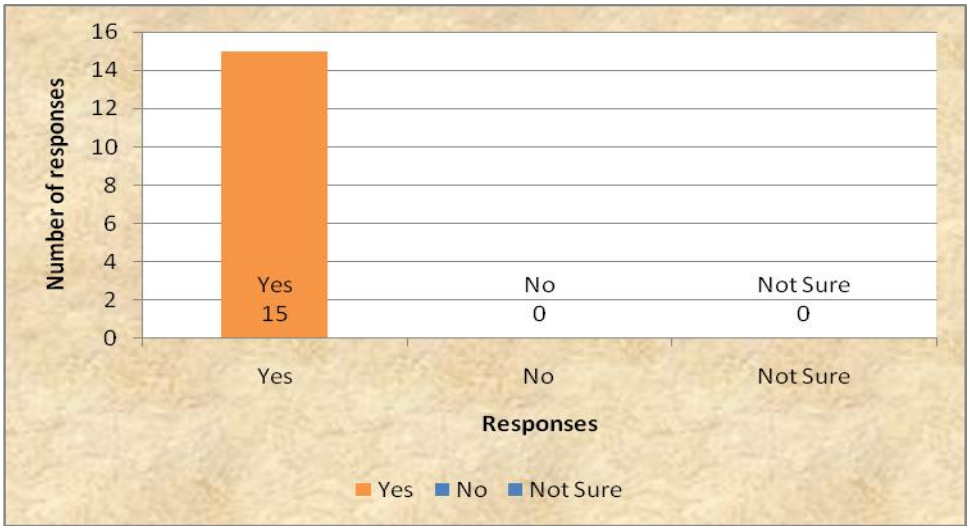


Figure 62: Interest in the system results

Every participant expressed some enthusiasm in the system.

8.5.8 Average Completion Time per Task

Several application use scenarios were set up for the survey participants. Each participant was timed for the duration of each task, and the average time for completing the task was calculated. The following sections provide a summary of the records for four different tasks. Each recording starts from the selection of language of preference to the completion of the desired task.

8.5.8.1 Registering Task

Each participant was timed for the duration of their registration on the system, using legitimate details. Table 5 below shows the results obtained:

Table 5: Registration times

Participant	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Time(secs)	106	85	90	101	112	125	189	96	81	93	96	103	118	168	192
Average Time(secs)	**117**														

8.5.8.2 Depositing Task

Each participant used his/her registration details to log into the system and perform a deposit to a number that already exists in the database. Table 6 shows the results of the various times taken to perform this task.

Table 6: Deposit times

Participant	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Time(secs)	52	43	48	55	50	67	118	54	46	54	61	65	75	52	105
Average Time(secs)			63												

8.5.8.3 Transferring Task

Each participant's account was credited with a small balance with an existing entrepreneur in the database. This was to make it possible for the participants to transfer a fraction of their existing balance to another registered user. A single recipient number was used in this task and Table 7 shows the times obtained in the tests.

Table 7: Transfer times

Participant	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Time(secs)	66	57	62	72	76	80	115	71	58	64	65	76	90	96	110
Average Time(secs)			77												

8.5.8.4 Analysis of Timed Tasks

From the results, it can be generalized that the younger participants took a shorter period of time than the older ones to complete the given tasks. The following are some of the factors that we feel influenced these results:

- *Familiarity or exposure to a variety of mobile devices* - the old people are not very familiar with latest technology.

- *Level of education* - the more educated a person is the higher their mental aptitude for learning new things faster.
- *Age* - this influences the memory lapse rate.

Figure 63 is a graph depicting the average time it took to complete each task.

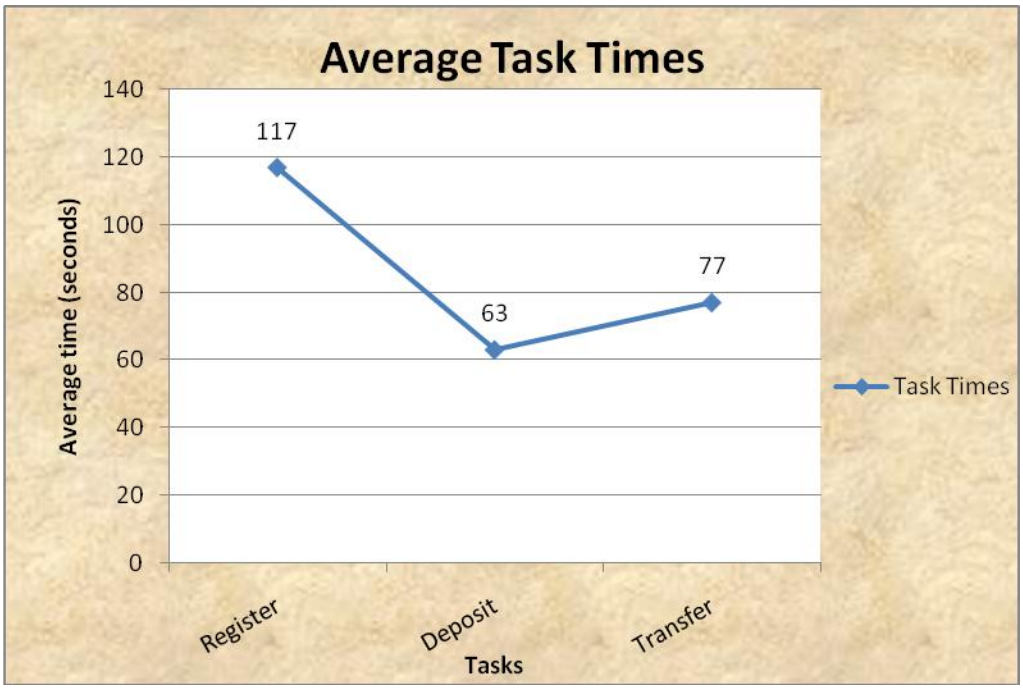


Figure 63: Average task times

The registering task took longer than the other tasks to complete. All tasks where timed from appearance of the language selection menus. As shown in Figure 63, registration had the highest average task time of 117 seconds, followed by transferring with an average task time of 77 seconds and then depositing with an average of 63 seconds.

8.6 Performance Testing

Figure 64 below shows the system specifications for the computer used in the development and testing of some of the already achieved results.

Figure 64: Computer system specification

This system has a Windows XP operating system and an Intel Pentium(R) CPU with a processor speed of 3.20GHz. It also has memory of 448MB of dynamic RAM.

8.6.1 Memory Usage

As the application runs, it utilizes some of the computer's resources like memory. Memory usage depends on the application's resource requirements at that particular time. The following figures show the different memory usage statistics for the system, while being run on the computer.

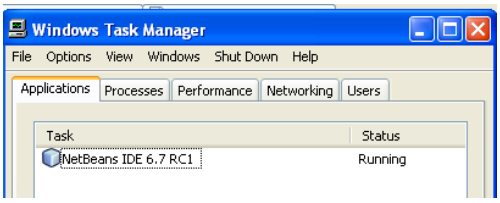


Figure 65: NetBeans application running alone

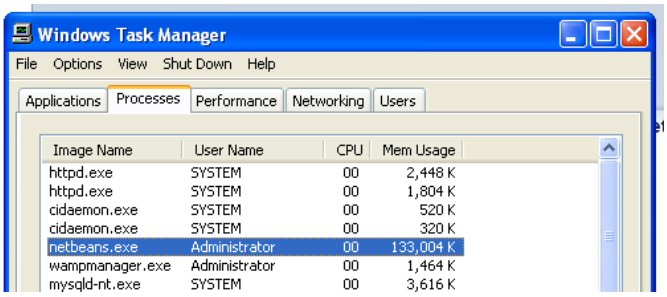


Figure 66: NetBeans memory usage

Figure 65 shows the task window with the NetBeans IDE as the only application running. Figure 66 shows that the memory usage by the NetBeans IDE equals to 133.004 Kb.

When the midlet is building, the memory usage also changes. Figure 67 below shows the memory usage for the application when building.

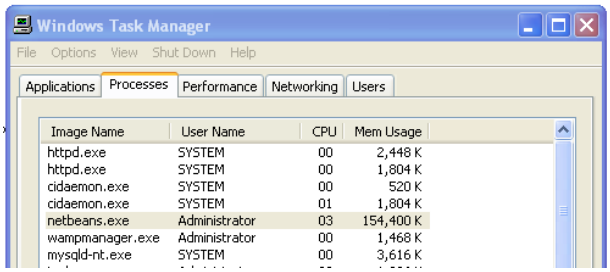


Figure 67: Memory usage while building the midlet

It utilizes 154.400 Kb of memory space.

When running the emulator, both the IDE and the emulator utilize some memory in the system. Figure 68 below shows memory usage statistics for when the emulator is started while running the midlet.

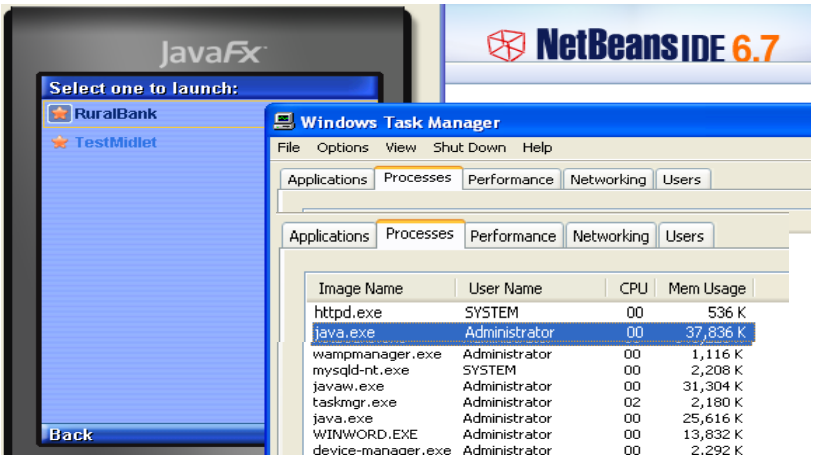


Figure 68: Memory usage when running midlet (emulator)

The emulator utilizes 37,836Kb memory compared to the 148,228Kb by the NetBeans IDE.

8.6.2 Obfuscation

To reduce the size of the Jar file, an obfuscation technique had to be implemented in this project. The technique reduces the size of the jar files by renaming certain classes and methods to use shorter names. In the NetBeans IDE there is a built in obfuscator, although it is possible to have a plug in module of an obfuscator. This section will show the jar and JAD file sizes for the different obfuscation levels in NetBeans 6.7 IDE.

8.6.2.1 Obfuscation OFF

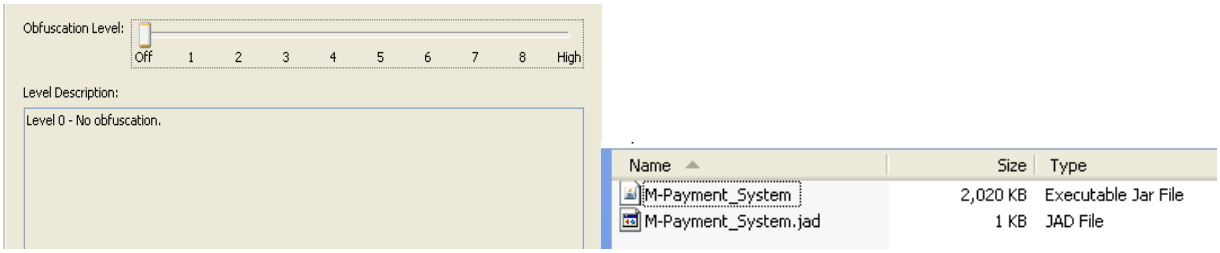


Figure 69: Jar and JAD file sizes with Obfuscation OFF

The jar file is 2,020KB in size when obfuscation is off. One part of the diagram shows the properties that are changed or modified in the process, which is none in this case. The other part shows the sizes of the jar and JAD files.

8.6.2.2 Obfuscation Level 4

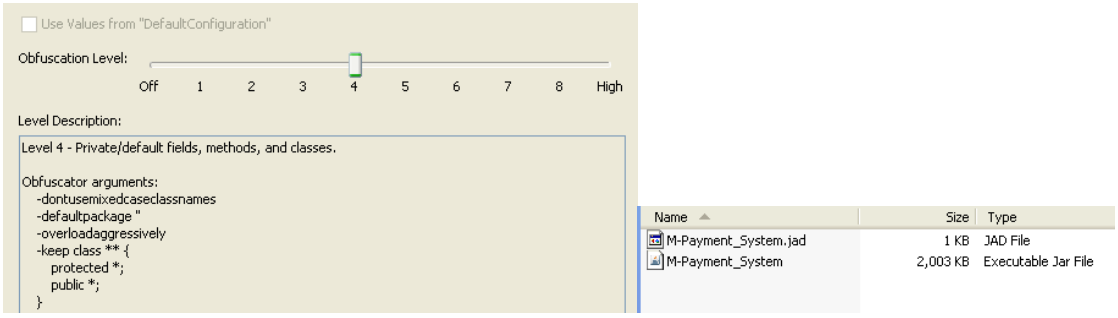


Figure 70: jar and JAD file sizes for obfuscation level 4 and 8

When obfuscation is increased to level 4 or 8 the size of the jar file decreases to 2,003KB.

8.6.2.3 High obfuscation (level 9)

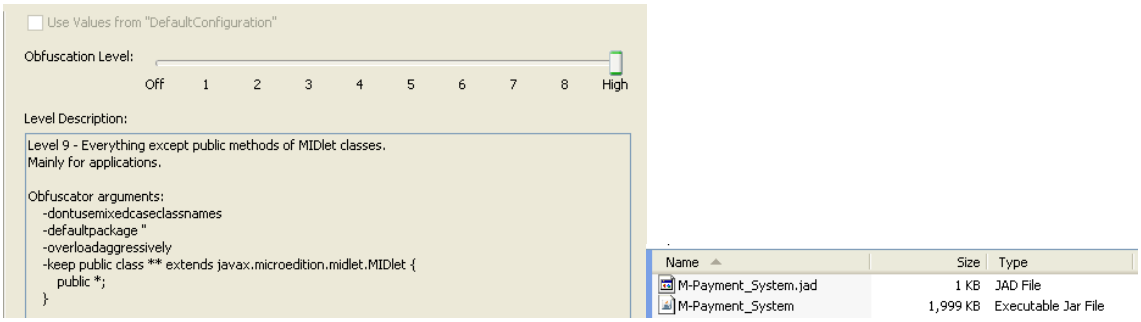


Figure 71: Jar and JAD sizes at maximum obfuscation level (level 9)

This is the maximum level of obfuscation giving a jar file of 1,999KB.

8.7 Conclusion

This chapter dealt with various results obtained through a survey conducted on a sample of 15 participants. It discussed the usability and user acceptance tests that were conducted using certain metrics. The results are presented primarily in the form of graphics, and range from simple demographic statistics to the performance evaluations through the use of timed tasks.

CHAPTER 9: DISCUSSION AND CONCLUSION

This chapter discusses the business model of the M-Payment system and the benefits it offers to the subscribers and entrepreneurs who use it. It also outlines some future work and concludes the study.

9.1 Project Summary

This project aims to improve the socio-economic aspect of the people in marginalized regions. The project uses Dwesa as a case study. Dwesa is a rural community in the Eastern Cape Province of South Africa, and is characterized by poverty, a high unemployment rate and no formal infrastructure, among other things. This study focused on an M-Payment system, which is a mobile application meant to perform banking transactions and purchasing electricity locally using a mobile device. The system is aims at to curbing the travelling costs, from Dwesa to remote towns and from. A database located in the already existing network infrastructure in the schools acts as a bank repository system. With the growing popularity of mobile devices, this project makes it feasible for the developed M-Payment system to be accessed by any person in this rural community as long as they have a WAP enabled device. The advantages of using the mobile devices are its mobility and affordability. The following sections discuss some of the important aspects of the system.

9.1.1 Technologies

A detailed study and review of the technologies was conducted prior to choosing the technologies to use (see chapter 3: *Mobile Computing technologies*).The M-Payment system was developed using Java ME technology and WAMP 2.0 server. The software's used are open source, motivating their choice thereof. Java has the advantage of being a cross-platform technology over some of the reviewed technologies. The database used for storage of financial data was MSQL, with HeidiSQL application used to interface the database components during development.

9.1.2 Platform Integration

The system is an independent financial application. Its functions are not connected to the existing banking services or electricity purchasing, although some input for some transactions depend on some services like:

- Deposit- the person deposing money for a rural based user has to first deposit it into an entrepreneur's traditional bank account in a bank.

- The electricity voucher for the rural based user has to be purchased from any electricity vendor based in urban areas, by an urban based user operating a phone shop.

The M-payment system currently has no functional integration to the existing financial systems but requires some input gathered from external systems.

9.1.3 Benefits

The M-Payment system involves the following players:

- Rural based entrepreneur (shop owner)

- Urban based entrepreneur (phone shop owner)

- Rural based subscriber (money and electricity recipient)

- Urban based user (sender of money)

The system is a possible source of revenue for the two types of entrepreneurs, while at the same time saving time and money for rural based subscribers who no longer need to go to a bank or electricity vendor in town. Since the bank account being used for deposits belongs to the rural based entrepreneur, the subscribers can be charged a special fee for a deposit transaction into entrepreneur's bank account. The subscribers will be charged for the bank charges in addition to

the service charge, for using the entrepreneur as a cashier. The Dwesa people will save a lot of money that can be used on other important things, with the shop entrepreneurs making a profit through charges incurred on subscribers. This somehow improves the socio-well being status of the community. This project is very convenient and user driven. Apart from the financial benefits, the project also helped the researcher to develop as a programmer.

9.1.4 System specification requirements

The system specification requirements are stated in the objectives at the beginning of the research. In this section the requirements are coupled with an analysis of their outcome in order to gauge the success of the project.

- ***Usability*** – with 83% of the users able to complete the designated tasks, the prototype proves to be effective in satisfying the usability metric. With the average assists per person equal to 3, during prototype navigation, it shows that the users can easily adapt to the application with time.
- ***Acceptance*** – with 67% of the users finding the prototype useful and 1% finding it useless, it leaves one with the desire to convince the remaining 27% into accepting the solution as useful to their everyday lives. Coupled with the 100% interest shown by users to the innovation, the percentages are a big encouragement for the further development, improvement and deployment of the final prototype in future.
- ***User defined*** – *the system should address the needs of the target community*. The system does address the needs of the Dwesa people as they needed a banking facility or application of this nature. The community expressed great appreciation on the introduction of this innovation as components of the system are user-specified requirements.
- ***User friendly*** - *ease of use should be a necessity*. The application uses simple menu lists and minimum typing on the keypad is expected. It also has the language choice function which allows the user to opt for their language of choice.
- ***Cost-effective*** - *should not be expensive to use.* The application reduces travel costs to and from town. The only costs incurred are for the sending of data to the database, using

the application installed in the mobile phone, and cannot be compared to the stipulated sixty rands (R60) that it would cost the user to travel to and from town for the traditional services.

- ***Embraces data integrity and security*** - *data should be protected from manipulation by unauthorized persons*. Use of passwords and the transaction pin code entails some level of integrity and security. For example, the entrepreneur has the cash in menu but has no access to subscriber accounts as he does not possess their transaction pins for the withdrawal and transfer processes.
- ***Localization of the system*** – the *system should have a language option, which includes the dominant language in Dwesa*. The system currently has two language options – isiXhosa and English. This will also help the illiterate learn their language as well as English as they attempt to use the system.

9.1.5 System testing

The M-Payment system was tested for many aspects. The users performed various timed tasks as to test the usability of the system. Through these tasks the average time to complete a task was calculated. Performance testing was not done on the mobile device as the system was only developed and tested on the desktop computer. However, the system performance on the computer was also done to check memory usage and compare the jar file sizes after different obfuscation levels.

9.1.6 Security

The system security was implemented using password protection to prevent unauthorized access to system resources at log in. During cashing in, a secret pin is also used as a way of allowing only legible users to manipulate their own accounts. Otherwise other levels of security involving cryptography have yet to be looked into as future improvements to the system.

9.1.7 Challenges

This project presented a number of challenges from the outset. Every innovation comes with challenges, and the following are some of the difficulties encountered during this research:

- *Language barrier* - Dwesa is a Xhosa community. With a low level of literacy, the language of communication had to be Xhosa throughout. Nonetheless, with the help of some Xhosa research mates and local school teachers the communication of information improved.
- *System development* - the system had to be a two in one, and both modules had to complement each other. The system has two different administrators. These factors made the development challenging, as the menus had to be specific to only a single type of user e.g. electricity entrepreneur or money transfer entrepreneur-each of their menus are different.
- *Deployment* - due to time constraints, real life deployment in a mobile device has not been possible, however, a framework exists for this in the future.

9.1.8 Future Work

The following are some of the recommendations for the future extension of this project:

- *Integration* - this is about having this system dependent directly on already existing banking infrastructure as regulated by the regulatory framework for Branchless Banking. For example, the deposit form can be removed from the midlet application and have the deposit function and database update done automatically in the bank. Through such integration the system can thus be able to be improved to work alongside other financial systems, for example WIZZIT, in transferring of funds from one system to another.
- *Security* – currently, the security on the system uses password and secret pin numbers. In future, cryptographic techniques need to be implemented for data sent over the network.

Database security has to be implemented to ensure protection of the M-Payment system database to improve trust and reliability.

- *Deployment* - this should come after implementing the security. The system deals with money; hence security is an integral aspect of it.
- *Sustainability*- an initial investment will get the application deployed and costs for maintenance of the resources included. Since this project is aligned to the Wireless Village concept, it has to sustain itself at some stage. Funds may come from the charges incurred on every person using the system. Ideas on how to ensure sustainability and their implementation remains a concern for the future.

9.2 Conclusion

The M-Payment system is a user driven system which is meant to benefit both the subscribers and the entrepreneurs. The system implementation is a success story, although deployment remains a future act. System development presented a number of challenges which bear many ideas. This project improves the socio-economic well-being of marginalized communities, with the pilot prototype designed to meet the specifications and requirements of the target community. This is in contrast to the traditional Information and Technology projects and applications that target the already profitable markets in urban areas, as this system is not a profit-oriented, but rather a poverty mitigation oriented application.

References

Ajose S.O, Ezebuiro I.I, Oluwo B.O (2005), "A Review of Wireless Local Area Network Technologies", *The Pacific Journal of Science and Technology* **6**(2)

Akhila S., Lakshminarayana M. (2008), "Averaging Mechanisms to Decision Making for Handover in GSM*", World Academy of Science, Engineering and Technology (42)*

Austin M., Baskerville T., Bevins Jean., Bevins Joyce, Evans R.(2010), "Evaluation and Implementation of Web 2.0 Technologies in Support of CReSIS Polar and Cyberinfrastructure Research Projects at Elizabeth City State University", *ECSU*

Banerjee P., Chau P. Y. K. (2004), "An evaluative framework for analyzing e-government convergence capability in developing countries", *Electronic Government – Inderscience***1**(1), pp 29-48

Barbara D. (1999), "Mobile Computing and Databases-A survey*", IEEE Transaction on Knowledge and Data Eng* 1(1), pp108-117

Brown J., Shipman B, Vetter R. (2007), "SMS: The short Message Service". *Innovative Technology for Computer professionals, IEEE Computer Society* **40**(12), pp 106-110

CGAP (2010), "Update on Regulation of Branchless Banking in South Africa" http://www.cgap.org/gm/document-1.9.42404/Updated_Notes_On_Regulating_Branchless_Banking_South_Africa.pdf (Accessed 27 March 2012)

Dalvit L., Thinyane M., Muyingi H., Terzoli A. (2007), "The Deployment of an e-Commerce Platform and Related Projects in a Rural Area in South Africa", *Strengthening the Role of ICT in Development* **l** (1), pp 27

Deepak P. (2006),"WiMAX: Broadband Wireless Access Technology", *IDEA Group*

Deitel (2002),"Wireless Communications Technologies" (Part 2), *Prentice Hall*

Dogac A., Laleci G., Kabak Y., Kurt G., A. Acar. (2002),"A Platform for Semantically Enriched Mobile Services", *Procs. of the First International Conference on Mobile Business*

Efstathiadis E., Holmgren D., Jo'o B., Simone j., Singh A., Watson C.(2006), "Common Runtime Environment Definition"

Falkena H., Bamber R., Lllewellyn D., Store T. (2001),"Financial Regulation in South Africa" *SA Financial Sector Forum*

Fitzek F. H. P., Reichert F. (2007), "Mobile phone programming and its application to wireless Networking", *Springer*

Gajwani Al. (2006), "Telecom Asia" http://findarticles.com/p/articles/mi_m0FGI/is_12_17/ai_n19052881 (Accessed 29 November 2011)

Gavals D., Economou D (2007), "The Technology landscape of Wireless Web". *International Journal of Mobile Communications* **5**(5), pp508 – 527

Gorlenko L, Merrick R. (2003), "No wires attached - Usability challenges in the connected mobile world", *IBM Systems Journal* **42**(4), pp 639 - 651

Hughes N., Lonie S. (2007), *M-PESA*, "Mobile Money for the "Unbanked" Turning Cell phones Into 24-Hour Tellers in Kenya", *Innovations: Technology, Governance, Globalization* **2**(1-2), pp 63-81

Hunter J., Crawford W. (2001), "The servlet API", *Java Servlet Programming (2nd edition),* O'Reilly & Associates

ICT and electronics in South Africa (2008) http://www.southafrica.info/business/economy/sectors/icte-overview.htm (Accessed 30 November 2011)

Imielinski T, Badrinath B.R (1994): "Mobile and Wireless Computing: Challenges in Data Management", *ACM* **37**(10), pp 18-28.

Imielinski T. (1996), "Mobile Computing: DataMan Project Perspective, Mobile networks and Applications", *Mobile Networks and Applications Journal* **1(**4), pp 359-369

Isakow A., Shi H. (2008),"Review of J2ME and J2ME-based Mobile Applications", *IJCSNS International Journal of Computer Science and Network Security* 8(2), pp 189 – 198

ITU Telecommunications Indicators Update (2005), "Africa reaches historic Telecom Milestone". *International Telecommunications Union (ITU),* Update 3.4

Jordan R., Abdallah C.T. (2002), "Wireless Communications and Networking: An Overview". *IEEE Antenna's and Propagation Magazine* **44** (1)

Kesic N., Peacy K. (2009), "U2 Compact Framework: UniObject for .NET Compact Framework for a smarter planet", *IBM Corporation*

Kolsi O., Virtanen T (2004), "MIDP 2.0 Security Enhancement", *Proceedings of the 37th Hawaii International Conference on System Sciences*, *IEEE*, pp 8

Kok P., Collision M. (2006), "Migration and Urbanization in South Africa*", Statistics South Africa Report*

Lakhani K. R., von Hippel E. (2003), "How open software works: "free" user-to-user Assistance*", Research Policy* **32**(6), pp 923 - 943

Lerdorf R., Tatroe K., MacIntyre P. (2006), "Programming PHP", *O'Reilly Media Inc.*

Ľudovít M., Miriam O. (2010)," What Engineering Education Needs More: COMPUTER OR INFORMATION LITERACY?", *Joint International IGIP-SEFI Annual Conference*, Trnava, Slovakia.

Mansel, R (2001)," Digital opportunities and the missing link for developing countries", *Oxford Review of Economic Policy* Volume **17**(2), pp 282-295.

Martineau P. (1999)," Prepaid card in a GSM based wireless telephone network and method for Operating prepaid cards", *Google Patents* (22), US Patent Number 5,915,226

McKitterick D., Dowling J. (2003),"State of the Art Review of Mobile Payment Technology", *Retrieved September* **14**(2003), pp 2003—14

mobiThinking (2012), "Global mobile statistics 2012: all quality mobile marketing research, mobile web stats, subscribers, ad revenue, usage, trends…" http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats (Accessed 28 march 2012)

Moss K. (1999), "*Java Servlets", (*2nd edition) *McGraw-Hill*

Mpofu H, Thinyane M, Terzoli A (2010)," Virtual Money Transfer System: A Component of the M-Payment System for a Marginalized Region", *SATNAC*

Muchow J. W. (2001),"Core J2ME Technology and MIDP", *Prentice Hall*

Ndiwalana A., Popov O. (2008)," Mobile Payments: A Comparison between Philippine and Ugandan Contexts*", IIMC International Information Management Corporation*

Nokia Siemens Village Connection (2008), "Bringing the benefits of affordable mobile access to rural communities", *Nokia Siemens Networks* Code 11486 – 04/2007

Nylund K (2001), "Developing Software for Mobile phones using J2ME", *ACM Classification*

Owens J., Balingit C (2007): "Technology series, Part 2: Virtual mobile Wallets" http://www.microfinancegateway.org/content/article/detail/43486 (Accessed 23 October 2011)

Parikh T. S., Lazowska E.D. (2006),"Designing an Architecture for Delivering Mobile Information Services to the Rural Developing World", *Proceedings of the 15th international conference on World Wide Web,* ACM , pp 791 - 800

Patel I., White G (2005)," *M-government-South African Approaches and Experiences*" in EURO mGOV, pp 313 - 323

Patel P., Moss K. (1997), "Java Database Programming with JDBC", *Coriolis Group Books*

Perrucci G., Häber A. (2007)," Mobile Phone Programming Part II", pp 63-93

Policy Document (2011), "Reviewing the regulation of Financial Markets in South Africa" http://www.fsb.co.za/finmarks/FinancialMarketsBill/FMBPolicyDocument.pdf (Accessed 27 March 2012)

Quigley E., Gargenta M. (2007)," PHP and MySQL by example", *Prentice Hall PTR*

Saha S., Jamtgaard M., Villasenor J. (2001),"Bringing the Wireless Internet to Mobile Devices, University of California", *IEEE* **34**(6), pp 54-58

Salami R., Lefebvre R., Lakaniemi A., Kontola K., Bruhn S., Taleb, A.(2006), " Extended AMR-WB for High-Quality Audio on Mobile Devices", *IEEE* **44**(5), pp 90-97

Salvage R (2005),"Flexible service integration with BREW*",* *TELEKTRONIKK* **101**(3/4), pp 78

Singh A. M. (2004): *Bridging the Digital Divide -The Role of Universities In Getting South Africa Closer to the Global Information Society.* South African Journal of Information Management, 2004.

Singh S. (2009),"Digital Divide in India: Measurement, Determinants and Policy for Addressing the Challenges in Bridging the Digital Divide", *Central Asia* **11** (20.1), pp 6--3

Stal M. (2000),"*Component Technologies for the Middle Tier: CCM, EJB, COM+",* Siemens AG, Corporate Technology. http://www.stal.de/Downloads/middletiercomponents.pdf (Accessed 29 November 2011)

Sultana R. (2009), "Mobile Banking: Overview of Regulatory Framework in Emerging Markets". *4th Communicartion Policy Research, South Conference.* Sri Lanka

Sun Microsystems (2000), "J2ME Building Blocks for Mobile Devices", *Sun Microsystems*, White Paper on KVM and the Connected, Limited Device Configuration (CLDC)

Sun R. (2004),"Brew and its Application Development", *Application Research of Computers,* pp 1

Suehring S. (2002),"MySQL Bible", *Wiley*

Tella A., Olorunfemi D. Y. O. (2010),"The future of ICT in developing world: Forecasts on sustainable solutions for global development", *India Journal of Library and Information Science* 4(2)

Thai T., Lam H. Q. (2003), ".NET Framework Essentials, Third Edition", *O'Reilly*

Thomas T. (2006): *Seminar Report on Survey of Smartcard and Mobile Payments.* KReSIT IIT Bombay

Trichkova E (2009), "Application of PHP and MySQL for search and retrieval Web services in Web information systems" *Citeseer*

Wertlen R. (2008), "An Overview of ICT Innovation for Developmental Projects in Marginalized Rural Areas", *Prato CIRN 2008 Community Informatics Conference*: ICTs for Social Inclusion: What is the Reality? Refereed Paper.

Widenius M., Axmark D. (2002),"MySQL Reference Manual: documentation from the source", (1st Edition), *O'Reilly*

Woo J., Jang M. (2008),"The Comparison of WML, cHTML and HTML in m-Commerce", *Journal of Software* 3(7), pp 22-29

World Bank (2005),"Using Information and Communication Technologies (ICT) to support Rural Livelihoods: Evidence, Strategies, Tools", *Workshop for World Bank Staff*

Zhen-Wei Q, Christine L., Bruno M., Michael S., Eric W., Bjorn C., George R. ; Halewood N.; Adamali A. ; Coffey J. O. ; Safdar, Z. (2006), "Global trends and policies-2006 : information and communications for development", **1** (35924)

Appendix A: GUI Components Sample Code

THE ARRAY S[1]….S[n] CONTAINS DIFFERENT STRINGS FROM THE DIFFERENT DATABASE TABLES

A1: Text fields and Command buttons

```
private electrAcc            = new Command (s[15], Command.OK,2);
private electricity          = new Command (s[16], Command.OK,2);
private moneyTransfer    = new Command (s[17], Command.OK,2);
private addMeterNumber = new Command (s[18], Command.OK,2);
private deregister           = new Command (s[19], Command.OK,2);
private deregisterUser    = new Command (s[8], Command.OK,1);

private userName          = new TextField(s[0], "", 10, TextField.ANY) ;
private password          = new TextField(s[1], "", 10, TextField.PASSWORD);
private regphoneID        =new TextField (s[0], "", 50,0);
private regpassword       = new TextField (s[1], "", 50,TextField.PASSWORD);
private regpassword2  = new TextField (s[2], "", 50,TextField.PASSWORD);
private regpinNumField = new TextField (s[3], "", 50,TextField.PASSWORD);
```

A2: Forms and Back Commands

```
private depositForm = new Form(s[0]);
private registerForm = new Form(s[1]);
private loginForm     = new Form(s[2]);
private cashInForm   = new Form(s[3]);
private cash              = new Form(s[4]);

private  back14 =  new Command (s[6], Command.BACK,1);
private  back15 =  new Command (s[6], Command.BACK,1);
private  back16 =  new Command (s[6], Command.BACK,1);
private  back17 =  new Command (s[6], Command.BACK,1);
private  back18 =  new Command (s[6], Command.BACK,1);
```

Appendix B: Adding Components to Forms

B1: Organizing GUI Components on the Form

```
loginForm.append(userName);
loginForm.append(password);
loginForm.addCommand(login);
loginForm.addCommand(register);
loginForm.addCommand(deregister);
loginForm.setCommandListener(this);

registerForm.append(regname);
registerForm.append(regsurname);
registerForm.append(regage);
registerForm.append(reglocation);
registerForm.addCommand(sendRegData);
registerForm.addCommand(back0);
registerForm.setCommandListener(this);
```

Appendix C: Money Transfer Code

```
public void commandAction(Command c, Displayable d) {
      String label = c.getLabel();
    if(c== exit) {
        try
        {   destroyApp(false);
            notifyDestroyed();
        }
        catch(Exception e) {
        }
  }
```

C1: Language Setup

```
  else if(c == getLanguage){
      for(int i=0;i<2;i++) {
        if(language.isSelected(i)) {
           lang = language.getString(i);
           String formNames = "formNames";
           System.out.println("u selected :" + language.getString(i));
           LanguageClass langClass3 = new LanguageClass(lang,formNames );
           String str3 = langClass3.getResult().trim();
           System.out.println("this is the Form Names "+  str3);
           timesSplitUsed = 0;
           split(str3,";");

           String start = "start";
           System.out.println("u selected :" + language.getString(i));
           LanguageClass langClass = new LanguageClass(lang,start );
           String str = langClass.getResult().trim();
           System.out.println("These are the command button Strings"+  str);
           split(str,";");

           String labelNames = "labels";
           LanguageClass langClass2 = new LanguageClass(lang,labelNames );
           String str2 = langClass2.getResult().trim();
           System.out.println("These are the labels  "+  str2);
           split(str2,";");

           String accnumsBank = "accounts";
           LanguageClass langClass4 = new LanguageClass(lang,accnumsBank );
```

```
                String str4 = langClass4.getResult().trim();
                System.out.println("These are the accounts for the bank  "+  str4);
                split(str4,";");

                String accnumsElectr = "accounts2";
                LanguageClass langClass5 = new LanguageClass(lang,accnumsElectr );
                String str5 = langClass5.getResult().trim();
                System.out.println("These are the accounts for electricity  "+  str5);
                split(str5,";");

                display.setCurrent(loginForm);
            }
          }
        }
```

C2: Login

```
        else if(c == login) {
            validateUser();
        }
    public void validateUser() {
        Thread t = new Thread(this);
        t.start();
    }
    public void run() {
        HttpConnection hc = null;
        StringBuffer b = new StringBuffer("");
        InputStream in = null;
        String pwd = password.getString();
        try{
            phone = Integer.parseInt(userName.getString());
        }catch(Exception exc){
            System.out.print(exc);
        }
        cellnumber = phone;
        String parameter;
        String result;
        String url = "http://127.0.0.1/bank/login.php?";
        parameter = "phone=" + phone;
        parameter += "&lang=" + lang;
        parameter += "&password=" + pwd.replace(' ', '+');
        System.out.println(url+parameter);
        try {
            hc = (HttpConnection)Connector.open(url+parameter);
            in = hc.openInputStream();
```

```
            int ch;
            while((ch = in.read()) != -1) {
               b.append((char) ch);
            }
         } catch (Exception e) {
            e.printStackTrace();
         }
        try {
            if(in != null) {
               in.close();
            }
            if(hc != null) {
               hc.close();
            }
         } catch (Exception e) {
            e.printStackTrace();
         }
         result = b.toString().trim();
         userID = result;
         if (userID.equals("admin")|| userID.equals("user")||userID.equals("adminadmin")) {
            display.setCurrent(menuForm);
         }
```

C3: Registration

```
        else if(c == register) {
            regphoneID.setString("");
            regpassword.setString("");
            regpassword2.setString("");
            regpinNumField.setString("");
            regpinNumField2.setString("");
            regtitle.setString("");
            regname.setString("");
            regsurname.setString("");
            regage.setString("");
            reglocation.setString("");
          display.setCurrent(registerForm);
        }
         else if(c == sendRegData){
            String rpass = regpassword.getString();
            String rpass2 = regpassword2.getString();
            String rpin = regpinNumField.getString();
            String rpin2 = regpinNumField2.getString();
            String rttle = regtitle.getString();
            String rnme = regname.getString();
            String rsnme = regsurname.getString();
            String rloc = reglocation.getString();
            if(!rpass.equals("")&& !rpass2.equals("")
```

```
                && !rpin.equals("")&& !rpin2.equals("")&& !rttle.equals("")
                && !rnme.equals("")&& !rsnme.equals("")&& !rloc.equals("")){
            int rage = Integer.parseInt(regage.getString());
            int rphone = Integer.parseInt(regphoneID.getString());
            Register reg = new Register(lang,rphone,rpass,rpin,rttle,rnme,rsnme,rage,rloc);
            reg.postIt();
            String str = reg.getResult();
        }
        else if (c ==back0){
          display.setCurrent(loginForm);
        }
       }
      }
```

C4: Deposit

```
        else if(c == deposit){
          GenerateRef gen;
          int n = cellnumber;
          gen = new GenerateRef(n);
          String s = gen.getResult();
          display.setCurrent(depositForm);
          ref.setString(s);
          phoneID.setString("");
          amount.setString("");
        }
        else if(c== sendDepositData){
          String accChoice = null;
          for(int i=0;i<numOfAccounts;i++) {
            if(accountChoice.isSelected(i)) {
              accChoice = accountChoice.getString(i);
              System.out.println ("1. The account chosen for deposit is for" + accChoice+"selected
from "+numOfAccounts+"choices");
            }
          }
          System.out.println ("2. The account chosen for deposit is for" + accChoice+"selected
from "+numOfAccounts+"choices");
          int ree = Integer.parseInt(ref.getString());
          int receiver = Integer.parseInt(phoneID.getString());
          int sender = cellnumber;
          int amnt = Integer.parseInt(amount.getString());
          String str = "";
          StringItem tr=new StringItem("","");
          if(amnt > 0){
             dep = new DepositAmount(lang,ree,sender,receiver,amnt,accChoice);
             str = dep.getResult().trim();
             tr=new StringItem("",""+str);
```

```
        }
```

C5: Cash-In

```
else if (c == withdraw){
    int amt = Integer.parseInt(cashInField.getString());
    if(amt >= 0){
          CashInAmount cinAmt;
          cinAmt = new CashInAmount(amt,refer,trnsNum, cellnumber );
          String str = cinAmt.getResult();
          String []res;
     }
```

C6: Account Details

```
else if (c == viewAccount){
  ViewAccount view;
  view = new ViewAccount(lang,cellnumber);
  view.postIt();
  String str = view.getResult();
  String []res;
  res = convertLingu(lang,"","","USER PROFILE","I-PROFAYILI YOMSEBENZISI");
  StringItem strItem =new StringItem("",""+str);
  outputForm(res[1],back2,strItem);
}
```

C7: Check user type for Updating Account

```
else if (c == update){
  CheckUserType check = new CheckUserType(cellnumber);
  check.postIt();
  String str = check.getResult();
  if (str.equals("user")){
      display.setCurrent(userUpdateForm);
      title.setString(null);
      name.setString(null);
      surname.setString(null);
      age.setString(null);
      location.setString(null);
      userUpdateForm.setCommandListener(this);
  }
  else{
      display.setCurrent(ownerUpdateForm);
      ownerTitle.setString(null);
```

```
            accNumber.setString(null);
            ownerNme.setString(null);
            ownerSnme.setString(null);
            ownerAge.setString(null);
            ownerLocation.setString(null);
            ownerUpdateForm.setCommandListener(this);
        }
    }
```

C8: Updating Subscriber

```
    else if (c == sendUserUpdateData){
            String t = title.getString();
            String n = name.getString();
            String sn = surname.getString();
            String agee = age.getString();
            int ag = Integer.parseInt(age.getString());
            String loc = location.getString();
            UpdateUser up;
            if(!t.equals("")&&!n.equals("")&&!sn.equals("")&&!loc.equals("")&& agee.equals("")){
                up = new UpdateUser(lang,t,n,sn,ag,loc,cellnumber);
                up.postIt();
                String str = up.getResult();

            }
    }
```

C9: Update Entrepreneur Account

```
    else if (c == sendOwnerUpdateData){
        String t = ownerTitle.getString();
        String n = ownerNme.getString();
        String sn = ownerSnme.getString();
        String loc = ownerLocation.getString();
        if(!n.equals("") && !sn.equals("")&& !sn.equals("")&& !loc.equals("")){
            int ag = Integer.parseInt(ownerAge.getString());
            int ac = Integer.parseInt(accNumber.getString());
            UpdateOwner upd;
            upd = new UpdateOwner(lang,t,n,sn,ag,loc,ac,cellnumber);
            upd.postIt();
            String str = upd.getResult();
        }
    }
```

C10: Transfer Money

```
    else if(c == transferMoney){
       display.setCurrent(transferRefForm);
       pinNumField.setString("");
    }
    else if(c == transSearch){
      transferForm.deleteAll();
      transferForm.append(transRecip);
      transferForm.append(transAmnt);
       String accChoice = null;
       for(int i=0;i<numOfAccounts;i++) {
          if(accountChoice2.isSelected(i)) {
             accChoiceForTransfer = accountChoice2.getString(i);
          }
       }
      int foneID = cellnumber;
      int t = Integer.parseInt(pinNumField.getString());
      transferPin = t;
      TransRefSearch trans;
       trans = new TransRefSearch(transferPin,foneID,accChoiceForTransfer);
      trans.postIt();
      String str = trans.getResult();
      }
    }

    }
    else if(c == transProcess){
       int fone = cellnumber;
       int transf = transferPin;
       int recip = Integer.parseInt(transRecip.getString());
       int money = Integer.parseInt(transAmnt.getString());
       String str = "";
       StringItem tr;
       if(money >= 0){
            Transfer trns = new Transfer(transf,fone,money,recip,accChoiceForTransfer );
            str = trns.getResult().trim();
            tr=new StringItem("",""+str);
            String []res;
            res = convertLingu(lang,"","","TRANFER RESULTS","IMPUMELO EKUFAKELENI");
            StringItem strItem =new StringItem("",""+str);
            outputForm(res[1],back2,strItem);
         }

}
```

C11: Outstanding Accounts

```
else if(c == owingUsers){
   int id = cellnumber;
   OutstandingAccnts out = new OutstandingAccnts(lang,id);
   out.postIt();
   String str = out.getResult();
   StringItem strItem =new StringItem("","+str);
   outputForm(res[1],back2,strItem);
  }
 }
```

Appendix D: Electricity Module Code

D1: Sending Airtime

```
else if(c == sendEtym){
   String vendor;
   for(int i=0;i<2;i++) {
   if(electrVendors.isSelected(i)) {
      vendor = electrVendors.getString(i);
      try{ int pin = Integer.parseInt(etymVouch.getString());
         int pinVal = Integer.parseInt(etymAmt.getString());
         if(pin > 0 && pinVal > 0){
           SendAirtime sndAir = new SendAirtime(lang,etymRef,meter,cellnumber,pin,pinVal,vendor);
           String str = sndAir.getResult();
           String []res;
           outputForm(res[1],back13,strItem);
         }
       }


    }
```

D2: Retrieving Airtime Voucher

```
}else if(c== retrEtym){
  String elecRef = electrREF.getString();
  int r = Integer.parseInt(elecRef);
  ReqAirtime req = new ReqAirtime(lang , r,1);
  String result = req.getResult();
   StringItem strItem =new StringItem("","+result);
   outputForm(res[1],back13,strItem);
  }
```

D3: Sending Electricity Voucher

```
}else if(c== sendElectrDtails){
   int elecPin = Integer.parseInt(elecVOUCHER.getString());
    int elecVal = Integer.parseInt(elecVALUE.getString());
    int cellnum = Integer.parseInt(elecFONENum.getString());
    int re = Integer.parseInt(elecREF.getString());
    if(elecPin > 0 && elecVal > 0){
          SendElectricity sndElec = new SendElectricity(lang,re,elecPin,elecVal,cellnum);
          String str = sndElec.getResult();
          StringItem strItem =new StringItem("","+res[0]);
          outputForm(res[1],back16,strItem);
     }
```

D4: Retrieve Electricity Voucher

```
}else if(c ==retrElectr){
          String elec = electrREF2.getString();
          int r = Integer.parseInt(elec);
          ReqAirtime reqs = new ReqAirtime(lang , r,2);
          String result = reqs.getResult();
          String []res;
          res = convertLingu(lang,"","","ELECTRICIY RETRIEVAL RESULT","UKUPHAZAMA EKUFUMANENI
INOMBOLO ZOMBANE");
          StringItem strItem =new StringItem("","+result);
          outputForm(res[1],back13,strItem);
 }
```

D5: Electricity Account Details

```
}else if(c ==electrAcc){
   int id = cellnumber;
   ElectricityAccDetails elAcdetails = new ElectricityAccDetails(lang,id);
   elAcdetails.postIt();
   String str = elAcdetails.getResult();
   String []res;
     res = convertLingu(lang,"","","ACCOUNT RECORDS","IMPUMELO YE- ACCOUNT");
     StringItem strItem =new StringItem("","+str);
     outputForm(res[1],back13,strItem);
  }
 }
```

D6: Add Meter Number

```
        else if(c == addMeterNumber ){
              phoneNum.setString("");
              meterNumTxt1.setString("");
              meterNumTxt2.setString("");
              display.setCurrent(addMeterNumForm);
        }else if (c ==sendMeterNumber ){
            int fone = Integer.parseInt(phoneNum.getString().trim());
            int meterNum1 = Integer.parseInt(meterNumTxt1.getString().trim());
            int meterNum2 = Integer.parseInt(meterNumTxt2.getString().trim());
            if(meterNum1==meterNum2){
               AddNewMeterNumber addMeter = new AddNewMeterNumber (lang, fone,cellnumber,meterNum1);
               String str = addMeter.getResult();
                res = convertLingu(lang,"","","ACCOUNT RECORDS","IMPUMELO YE ACCOUNT");
                StringItem strItem =new StringItem("",""+str);
                outputForm(res[1],back13,strItem);
               }
        }
    }
```

D7: Deregister

```
        }else if(c== deregister){
           display.setCurrent(deregisterForm);
        }else if(c== deregisterUser){
            String pwd = deregPwd.getString();
            int ph = Integer.parseInt(deregPhone.getString());
            Deregister dereg = new Deregister(lang, ph,pwd);
            String str = dereg.getResult();
                res = convertLingu(lang,"De-registration Complete","Ukuvalwa kwe Account kuphumelele","DE-
REGISTRATION RESULTS","IMPUMELO EKUVALENI I-ACCOUNT");
                StringItem tr=new StringItem("",""+res[0]);
                outputForm(res[1],back3,tr);

      }
```