Final Year Project Thesis
Mo S H A Di

Shape-Shifting Human Addressable Digital Money

by

Chishala Matete Mpundu

29071011

Thesis submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Computer Science at the Department of Computer Studies, under the School of Natural Sciences, University of Zambia.

Supervisor

Mr. David M. Zulu

2013

DECLARATION

I, the undersigned here declare that the Shape-Shifting Human Addressable Digital Money System is my own work, that it has not been submitted for any degree or examination in any other university, and that all the sources I have used or quoted have been indicated and acknowledged by complete references.

Name: Chishala Matete Mpundu **Signature**:..............

Supervisor: Mr. David M. Zulu **Signature**:.............

Date: September 2013

ACKNOWLEDGEMENT

I would like to thank all the lecturers at the Department of Computer Studies for teaching me most of what I know and instilling a level of discipline. I thank my supervisor, Mr. David M. Zulu for the guidance and help he offered in this undertaking and I thank my family for being there for me in the good and bad times when it all seemed impossible.

I thank God for bringing me this far because years back I never thought I'd even have an opportunity of having tertiary education.

Contents

Abstract

MoSHADi is a technologically centred approach to the money system intending to preserve the role of financial institutions and appointed authorities in the transition towards digital economies. It is designed around various means for users to define what money is to them but retaining the fundamental properties of money to the system in control.

A major focus is on the layer whose purpose is to facilitate means of describing, accessing and utilising monetary value and the layers allowing value to flow securely, distinctly and efficiently over supporting digital communications network. The concept picks up from conventional aspects of money, both the physical and current electronic kind, and relying on the existence of regular objects, mechanisms and user defined patterns to render and carry value.

The ability to assume various tangible and intangible forms achieves the form or shape-shifting property, adding an aspect of personalisation and inspiring pride in money ownership by allowing users to define what form or pattern constitutes the value of their money.

While seemingly placing user interests at the centre from the surface of it, the system, to degree, indirectly encourages hard work to earn money and spending it to keep value in circulation in a transparent way and so bringing the interests of an entire financial system into perspective.

Chapter 1

1.1 Introduction

In the current technological age where distance is less of a factor to reckon with in terms of trade, the demand for money to conduct transactions has never been higher than it is today. As the world becomes more globalised, more people from different parts of the world are actively engaging in trade, cross border trade. Important elements fuelling this activity are the digital communications infrastructure and information systems which make it possible to digitally transfer value from one individual or entity to another. MoSHADi assimilates those features and takes into account the insecurities associated with what has been identified as global currencies, taking an approach that allows users to address monetary value in forms the system can understand. Internally, the system monitors flowing value across a domain of use and that provides a starting point for analysis of the financial data gathered.

It is obvious that political squabbles over trade currencies will continue if nations can't get over their differences and agree on common money to use. For this reason, MoSHADi is a deliberate attempt, within the confines of the technology available, to 'money blanket' a world in steady transition towards an all-out digital economy and usher an age where value translation and handling exchange rates are problems of the past.

MoSHADi also encourages and brings to life a vital characteristic of human beings which is the realisation of the power of money and the degree of control they have over it by allowing them to effectively personalise its outward representation.

With growing global human population comes the need to allow wealth to be distributed in a transparent and accountable manner for the greater good of mankind. In this context, the ability to have monetary value flowing efficiently and across all areas, a system like MoSHADi may play a central role, with integrated means of discouraging hoarding and providing data sufficient for analysis in the misappropriations of money and funding of illegal activities. The system also indirectly aims to protect the environment by reusing technology to carry money value.

1.2 Motivation and Significance

Motivation

One of the major inspirations for developing this system was *hyperinflation*, a special kind of inflation sometimes triggered by large increase in money supply which is not necessarily supported by Gross Domestic Product (GDP). If left unchecked, it causes prices to increase at ridiculously high rate and the currency rapidly loses value. Another way it is triggered is through the loss of confidence in a nation's ability to maintain its currency in the aftermath of war so traders demand risk premiums to accept the currency and in the process raise their prices. Nations that experience it experience massive developmental set back and in the context of the global economy, affect the global market directly and indirectly. A few countries that experienced it are Zimbabwe, Germany and Brazil.

Figure 1 Effects of hyperinflation: Extremely weak currency. User pictured needs lots of money to buy simple goods

The threat of hyperinflation is especially horrifying when global currencies threaten to take a dive. If its allowed to happen to any of them, especially the U.S. dollar, the world will be plunged into chaos. Placing measures to control and determine precisely the money circulating in the system provided inspiration to develop a money system to address these problems before they can happen.

Further inspiration came from the fact that a digital money system encouraging participation and equal say in economic matters for many more nations could propel the world towards adopting a solid global money whose strength collectively depends on the health of global trade.

Also the fact that criminal activity thrives through financial resources put together, quickly analysing value usage and handling characteristics of suspicious individuals and entities could be made possible with a system like MoSHADi.

Figure 2 Unscrupulous individuals and criminals use the power of money to fund operations

Deforestation, desertification and pollution were the other concern that compelled the development of a system that relies less on newly manufactured money carrier components and a lot more on existing technology and resources. A money system like MoSHADi would encourage reuse of technology and materials.

Figure 3 Paper production affects the environment through deforestation. Money uses paper and other materials

Significance

A system like MoSHADi once in place can provide a rich data mine that could help authorities trace value flow, profile economic activities and financial trends, conduct statistical work with what is seen on the system and enhance knowledge of what's happening on the ground and ultimately help policymakers in decision making. It would allow individuals to move between economic zones and countries with no problems of translating the money they have into that of the place they're in. It can see an end to the production of counterfeit money through secure mechanisms such as distinct registered digital identities for money but see the emergence of means to scam the system into believing a given value exists for which new means to fight digital counterfeiting would be derived. Security agencies can utilise logs and data available to monitor suspicious activities and profile individuals and institutions or applications may be developed on top of the details generated and used in the system to build profiles and gauge characteristics.

By utilising existing things around us, the system will help reduce the amount of damage to the environment by minimising the amount of new material for use in financial systems. Allowing money value to be officially recognised all over the world over digital communication systems sets the stage for a global currency whose stability depends on the combined economic effort of all actively trading people - nations and controlled with all Governments and regional economic bodies having an equal say. Finally, the system would put in place a stepping stone for an official currency for a new world order.

1.3 Scope

A system designed to emulate money for a global financial system ideally would have many components involved. From aspects of economics, through banking and down to the information, power management and communications systems would have to be taken into account. To narrow the scope to the area necessary to realise this system within specified time, attention was paid towards the development of access and user systems, a channeller system to accommodate and communicate with existing e-money systems allowing value to flow and the supporting server applications.

Under the Client systems, the work area included laying down model applications for financial institutions such as banks, money transfer centres, retail terminal systems, and applications for individual users. These inject and handle money as value as it floats across a world, exchanging hands of users known across the system. Two approaches were taken, standalone applications and web applications. This approach was taken to distribute functionality and allow access from different levels of the system.

The channeller system, designed to iron out differences in technologies and mechanisms used by various existing e-banking and e-money systems, provide a glue to allow value to cross over easily. Work under this area included setting up what would be called a *gate* to which existing systems see and can pass transaction details on a given value and receive value from others and the system, preserving integrity and ensuring changes to value are universally known.

Under the server applications work included was that of developing the main server systems holding value details, identified user details (effectively all living known people of all ages as identified by Government assigned numbers, both static and maturing in nature), details on financial institutions and central banks of regions and nations and those corporate and retail companies handling value and participating in trade.

1.4 Problem Statement

Can an elegant user-centred and computer network oriented system be developed that allows monetary value to flow conveniently and efficiently from anywhere and anyone like real money? If so, how can available technologies be used to provide rich interaction to embody the desired system?

1.5 Aims and Objectives

- **Aim:** To provide an adequate system that forms the foundation for an inclusive and convenient form of digital money.

Objectives:

- Create scalable and maintainable back-end to support the system.
- To provide multiple means of access and interaction for users with digital money.
- Be able to resolve common and ambiguous addressing for money from both institutions and individuals.
- Create means to pass value between individuals, individuals and institutions and between institutions.
- Create logs to save details for statistical and investigative purposes.

-**Aim:** To establish means for interaction with existing e-money systems.

Objectives:

- Provide template mechanism for external applications to communicate with system.
- Provide mechanism for system to notify customer's institution of changes to value customer aligned to system through e-money application.

1.6 Organisation of Thesis

This thesis is structured in a way that first provides insight to what the project is about, what inspired the work involved and then proceeds towards revealing the details of the system, bringing into perspective the various aspects of the system.

It begins with a literature review and then details of how the system was designed, implemented and tested. It ends with a discussion and a final conclusion.

1.7 Summary

The MoSHADi system intends to provide an insight into a way of translating conventional aspects of money to the digital realm with a user centred approach in an efficient way that maximises on the use of various means of human-computer interaction with technology available and demonstrate the possibility of a money suitable for an information age, comfortable for the common man and regulated by authoritative bodies.

Chapter 2

2.1 Literature Review and Related Works

Literature Review

Brief History of Money Systems

Ever since trade began, man has used many systems of payment to settle trade transactions [1] [2]. Except for the barter system, all of them used a distinct form of money. Money in and of itself is nothing but the value people attach to formal representations adopted to describe it.

For a brief period, value of money was aligned with the amount of Gold a territory had [3]. However, after the two World Wars, empires fell and a shift from the use of Gold in determining the strength of a currency was dropped, technologies emerged and the world began its solid transition towards an information age. By the mid-20th century, advanced research in magnetic strip technology and digital scanning systems laid the foundation for the development of a digital form of money relying on plastic cards which came to be known as credit and debit cards. The development of Automatic Teller machines reached an advanced stage by the 1960's, with Barclays Bank testing out the first one in 1967 at a London branch [21]. On the political scene, the world was shifting from the *Gold Standard* [3] and world financial power firmly shifted from Europe to the Americas. Money value was no longer aligned with how much Gold a nation had or required but solely on people's faith in it. This played a factor in accelerating the introduction and use of money value across electronic systems.

So what really is Digital Money and E-Money?

Firstly, seeing that many people interchange Digital / E-Money with E-Payment systems, it must be clarified that e-payment has to do with mechanisms used to settle transactions while the former serve as carriers of value needed to conduct transactions.

To define digital money, it is money stored and spent electronically [5]. It is a cash equivalent with stored value on an electronic device that may be local or remotely located such as on a server [6]. It's a record of funds or value available to a consumer for use over computer networks such as the Internet [7]. Any means of payment that exists purely in electronic form. Digital money is not tangible like a dollar bill or a coin. It is accounted for and transferred using computers [8].

Furthermore, some important terminologies when dealing with digital/electronic money that must be understood are; Mobile Payment, Electronic Funds Transfer, Internet Payments and Payment

Cards [18]. Mobile payment involves payments made via mobile technologies such as phones, laptops and tablets. Internet Payment refers to payments made online via linkage to bank accounts, third party institutions acting as trade middle men and anonymous systems such as Bitcoin [4]. Electronic Funds Transfer has to do with the transfer of value electronically from one bank account to another through devices such as computers, automated teller machines and point of sale systems to mention a few.

Electronic money systems have traditionally been used for making small payments although credit card systems are an exception. There are generally two categories of electronic money: note type and balance type [5].

The note type resembles a note with fixed value that is stored on a physical device. The balance type is the kind that allows for value to accumulate and charges to be made against that value. Electronic money can also be identified by how it interacts with the physical electronic devices. Some rely on dedicated hardware and include electronic wallets and purses, *NFC Chips* [12] and smartcards and surgically embeddable biochips and others completely rely on software, sometimes called virtual money.

Trends:

A wide variety of e-money technologies have come into existence, mostly confined to specific regions. If continental aspects are to be considered, Asia has the highest usage of e-money and density of e-money devices. This is due partly because of the structuring and enforcement of policies as well as the culture of the people who are ever ready to try out new technologies [13]. Despite taking the lead in the market for e-money, Europe and North America rank as the most innovative in the area with more systems emerging from those territories more often than anywhere else and lasting a lot longer than those from other active continents.

For a long time digital money or e-money has been used mostly to settle small bills or making small payments. With the potential to handle larger amounts, this will change inevitably. Already, online social networking and shopping allow users to purchase goods and services with electronic versions of currency. It's just a matter of time 'til many more kinds of transactions can be settled 'over-the – air'.

As the world goes through financial reforms in the wake of the financial crises, policies will likely become more flexible to accommodate these technologies and make the use of electronic money easier and common. With effective regulation, it can be a viable alternative. As a case in point, some regional bodies like European Union are reviewing their regulation structures and looking to advance electronic money technologies [15].

With digital money and payment systems, third world countries are also steadily making progress with cashless technology and helping them jump the digital divide. Exemplary systems providing the catalyst for this include MPESA of Kenya. Launched in 2007 by the company Safaricom, it is a mobile commerce system that allows people to pay for goods and services and transfer funds via mobile phones [16].

Reference to Technologies and Models incorporated in MoSHADi

MoSHADi system was developed to integrate a number of multimedia oriented technologies and interaction models (styles). Below, some of the details of those technologies and models are provided.

Human-Computer Interaction

This is an area dedicated to studying means of communication between computer systems and humans. It involves work from a number of fields such as psychology, ergonomics, cognitive and behavioural sciences to mention a few, and is constantly evolving to analyse and accommodate new ways of interaction in devices developed. It defines various models of interaction, methodologies and analysis techniques to evaluate means of interaction.

Among the most influential models of interaction is that of Donald Norman called Norman's Model [24]. It basically takes a user centred approach [25], mirroring human intuition. It is based on user formulation of a plan of action and then executing it. In this, it defines seven stages:

1. Establishing Goal
2. Forming Intention
3. Specifying Action Sequence
4. Executing the Action
5. Perceiving System State
6. Interpreting System State
7. Evaluating System State

Such models help develop understanding behind interactive systems from a given perspective. With this in mind, most systems are developed to maximise positive end user experience with a target of making it intuitive. In this respect the system developed in this project is no exception.

Producing a flexible, powerful and intuitive means of interaction makes it easier for users to fully utilise (and explore) a system and its functionality. Important is the degree of control that users can express through the provided means of interaction which can be realised through various channels of communication on which interfaces are built and the intelligence of underlying interface definition systems.

These channels can be one way implying details or commands necessary for the interaction can flow one way ideally from user to machine with some cases where it is machine to user or they may be bi-directional implying information regarding interaction flows both ways as though a request-response pair.

Although it is important for systems to have this quality, ultimately the purpose of the system and its economic aspects with respect to its intended market and role in society determine how complex the means of interaction and functionality are [23].

With proper marketing and pricing, technologies harnessing multiple means of interaction are available and integrated on user systems. The most common utilise vision and include ways that allow users to exert control through logical representations of instructions, in form of say icons, selectable/callable via hardware such as a mouse, joystick, touch screens and sensors and cameras in the case of motion sensing and gesture control.

Under vision based interactive systems is the developing field known as Computer Vision. Although still in infancy, areas of much activity in it are real time image processing with face recognition and detection and object detection, with projects such as OpenCV library. This is a cross platform open source library developed in the languages C++ and C by Intel for real time image processing in 1999. It is widely used on the Android platform for image processing applications [26] as well as research in robotics and vision based surveillance systems. This library forms an abstraction that allows developers to code without directly engaging the hardware and includes sophisticated algorithms and frameworks such as face-detection algorithms, the Viola-Jones real time visual object detection framework [27] based on *haar* classification techniques. These have made it possible to make vision based interaction via image processing using feed from basic hardware like webcams and digital cameras such as those on mobile devices and face detection and recognition are now a new means of interaction with computing systems.

To explain a little more on face detection, it works by identifying features on images that resemble computer models (algorithmically defined) for human faces known as Haar features, classified by Haar Classifiers in a cascade of stages. If identified, a rectangular section called a face-candidate (sub-window of the original image) is derived and the image passed onto the next stage of processing. An integral image, a summation of pixel values, is used in processing many more haar features in constant time, quickly by-passing non-face candidates. Comparators are used to determine if stage outputs succeed against a threshold for the next stage and ultimately to qualify as faces [28].

In the context of MoSHADi, a PHP-based face detection mechanism was integrated that would detect faces in pictures and associate details regarding user specified money value and numerical information identified from the face in a picture as an internal identity in the system. This was one approach taken to bring the shape shifting aspect to life, in a way, by taking a face and associating money value with it and at the same time locking that particular instance of the user's face only to the user upon *linkage*, which is a term used in the context of this project to describe the process of aligning a value with objects of interest.

One problem with face detection technologies of the day however, is that faces that are not upright are difficult to detect and sometimes objects in images that are not necessarily human faces are detected as faces and so the degree of accuracy is quite low but is still a great candidate for interaction and machine learning.

Another vision based feature incorporated in the MoSHADi system was the use of *QR-Codes* (Qucik Response Codes) as an approach to providing another alternative of describing money value. Here, an *alias* – the term used to describe a name a user gives to his or her money, would be encoded into a 2-Dimensional QR-Code using a java-based bar code generator programme built on the open source library from the ZXing (Zebra Crossing) project, and rendered as a JPEG format image in a downloadable from the web page tasked with the functionality of dealing with handling of encoded forms of money value. This aspect of the system harnesses the power and versatility of 2-D QR-Codes and the algorithms that make their generation possible wrapped in the core libraries.

QR-Codes have patterns laid out on a white background that can be understood by scan processing software working with digital cameras. The patterns are analysed with error correction mechanisms aided by the correction keys (codewords) in the image itself until suitable information is interpreted and data extracted. In this project, the data extracted are strings of characters through a third party barcode reader. The patterns describe version and format information, data and error correction codewords and alignment and timing patterns [32].

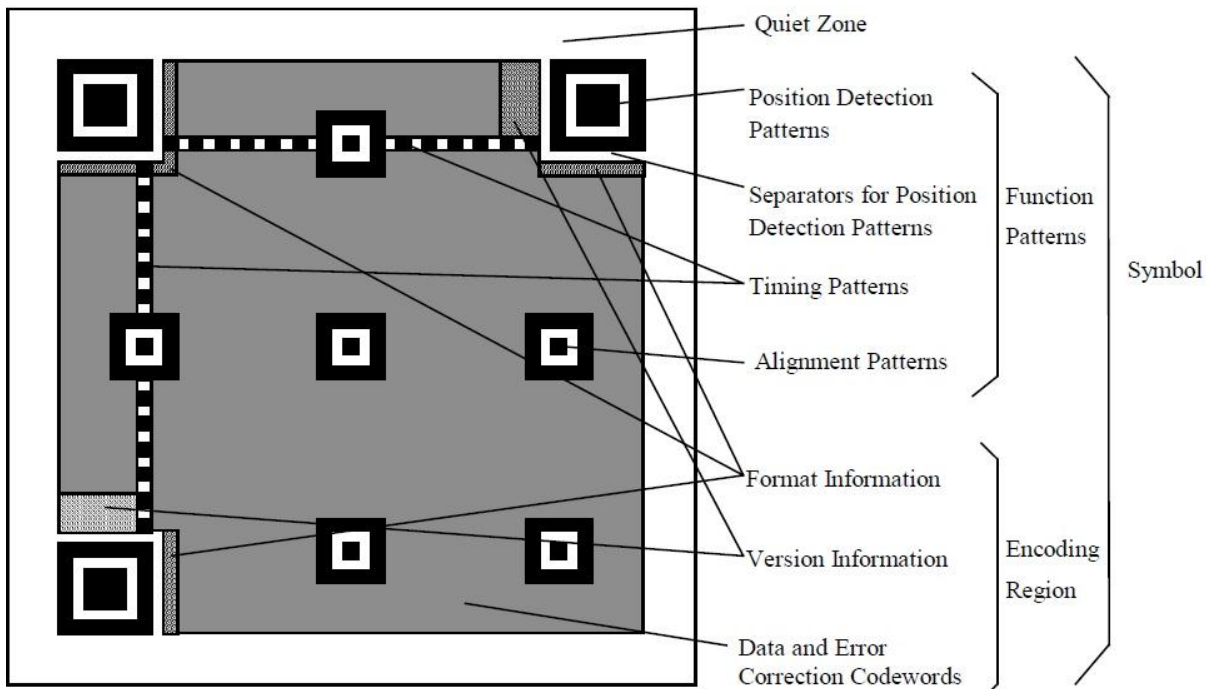


Figure 4 QR Code Structure

Version information goes with the scale used. Forty versions are specified from version one which scales to 21x21 QR-Codes through to 400x400 QR-Codes or version forty. Implications of this are that a stretched QR-code file with dimensions other than those specified by the versions will not be recognizable by scanning systems.

It is typical for a system to include multiple means of interaction and to this end other areas of much interest and of use in this project are Voice Recognition, Speech synthesis and Gesture control through patterns.

Voice recognition is technology that allows machines to hear and understand spoken information. Once understood, the machine can be designed to act upon the information it has. These actions may be defined by the developer or may be learnt through artificial intelligence. The system works with four basic stages; Analysis, feature extraction, modelling and testing [29]. Feature extraction tries to reduce the dimensionality of the input vector while preserving the strength of the signal. Modelling generates speaker models and conducts speaker identification. Noise cancellation and adaptive shaping is often used to increase accuracy. Often training through utterance of words to build internal approximate representations and aid the identification is done to increase accuracy. To take advantage of the power of distributed computing, some voice recognition applications use servers to process information they gather. Take for example, Applie Inc.'s *Siri* used on the iOS system. Siri relies on Server side technology to help process information and provide vital feedback on how well the application is operating on user devices. Siri makes use of another technology called Speech Synthesis. This is the opposite of speech recognition and is the generation of speech from presented information, usually in text format where it is describe as *Text-to-Speech, TTS.*

Two approaches are used when it comes to speech synthesis of type TTS-the first relies on a known set of words with recorded pronunciations and the second relies on specialised algorithms to generate speech dynamically, allowing previously unknown words to be spoken [30]. The dynamic approach is taken often for performance sake with synthesizing algorithms embodied in a speech programme or speech engine.

In this project, an open source java library known as Freetts was used. Freetts stands for Free Text-to-Speech and is based on Sun Microsystem's Java Speech API and can be used with MBROLA voices [31]. With domain knowledge limited to logic defined around action-response, this library was used to bring to life a speaking guide feature to aid visually impaired people in the quest for universality. For the Web version of the project, the speech API of Javascript was used that also uses the MBROLA and taking concepts from research into means of interaction, enlarged icon objects to narrow the time at which a user, in particular a visually impaired one, would arrive at an object of interest on the interface masking functionality of the system.

Networking and Data Storage Technologies

Persistent data is data stored for durations longer than the execution of the application using it. Files and database systems make it possible for data to be in a persistent state. Both files and databases have a number of advantages in their use but using both provides greater control. MoSHADi system does this, making use of Operating System file structures to hold multimedia files and handling other details such as value and user details on relational databases on the server applications. This approach was taken with a future in terms of size and user number scalability in mind because clustering rich data in form of binary files in databases can have performance penalties but delegating some responsibility to the governing Operating System eases the load.

Relational databases, as mentioned, are databases based on the set theory of mathematics [33]. They were used in the development of the system in this project because of the interlinked nature of the information handled. Since the foundation of everything in the system is a *value* belonging to a *user* known to be an economic participant and citizen of a region, all other details are built around these two pieces of information. Value transfers between individuals embodying transactions, as with real money, are possible through adjustments to records representing each individual in these databases and actions beamed across any other systems handling the value in question.

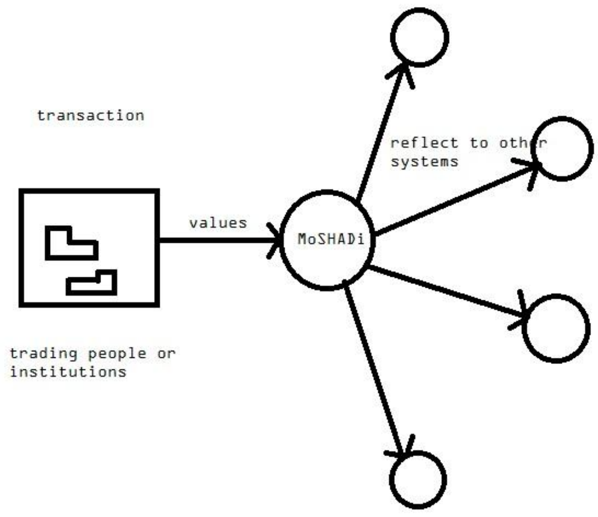


Figure 5 Nature of transactions in the context of MoSHADi

The graphical illustration above describes the outward nature of a deal fixed between individuals with a value known in the system and held on other systems, such as those of banks. The simplicity and elegance of relational databases simplifies the code required to make the update. The illustration also brings to light the aspect of networking, a cornerstone of the system ensuring integrity and functionality.

Computer networks are networks of communicating computer systems connected by radio or cabling. Today, these are used for a variety of purposes from entertainment to conflict. They are the foundation of money transfer systems and various electronic money forms. As the speed and capability of these systems continues to grow, many more systems will be built on top of them.

The MoSHADi system is designed to rely extensively on them and their high degree of sophistication and dependability. The nature of mesh type computer networks allows a system to be built in such a way that it would be available almost at all times because failure of some links would not shutdown access from other links. The diagram below shows this [34]:

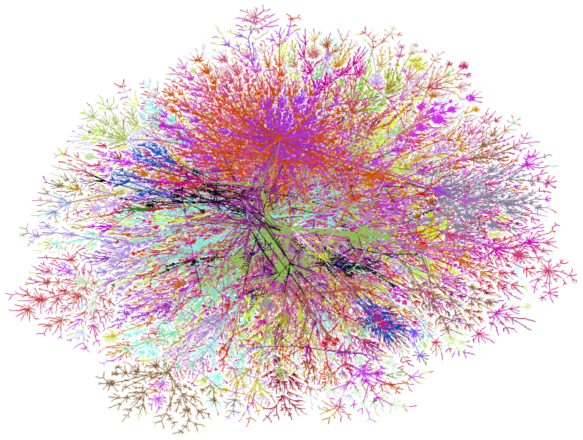


Figure 6 Graphical result of an algorithm showing how the Internet appears through IP addresses of active nodes and participant systems

Extensive networks dedicated to keeping the system afloat could be built around the existing infrastructure, forming an Autonomous System. If that is achieved, protocols such as Border Gateway Protocol, BGP [38] would be extensively used to allow communication with other large scale networks and have data flow through them and into the system in a secure and efficient manner.

Networks also provide a platform to implement more security. It is natural for applications to have in built security features relying on cryptography however, multistage security mechanisms can be achieved by placing network level security features such as the use of secure encrypted protocols allowing the transfer of data in encrypted form. Network level security can also filter out suspicious requests and implement techniques to counter excessive requests. Also at this level, recovery can be implemented in connection oriented protocols during communication sessions ensuring that the correct data is transferred to the requesting application.

Some secure protocols commonly used include SSH - the secure shell protocol and the hyper-text transfer protocol-secure or https. These use encryption layers to transform the data into cipher text and allow for remote administration and client-server communication.

2.2 Related Works

A number of technologies have been developed around the concept of electronic money, a large number of them driven by optimism. While some were developed to genuinely make trade easier, others were business opportunities and as it is in nature, only the fittest and strongest survive. Each contributed something and affected, in special ways directly or indirectly, the way we live. In this sense, a few of them with aspects similar to this are discussed below.

1. Bitcoin:

Perhaps the most significant similar project of recent times is *Bitcoin.* It was developed in 2009 by a young Japanese cryptographic expert by the name *Satoshi Nakamoto* [4] as an anonymous peer-to-peer electronic cash system for online use. It's used for making payments and purchasing items off the internet and in some areas of the physical world where it is accepted, paying for services through provisions such as payment apps on mobile devices on advanced mobile operating systems such as Google's Android and Apple's iOS. The bitcoin system is software oriented and relies on peering systems of individuals participating in it anonymously. Only a few bitcoins are released at a time, periodically produced in a lottery like manner. With more people trying to get them, the more difficult it becomes to acquire them and the value of bitcoins changes from time to time, with the general value known publicly.

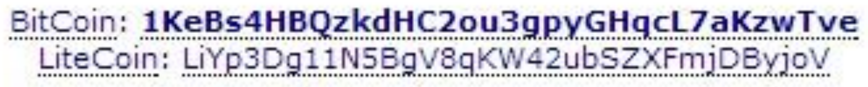

BitCoin: **1KeBs4HBQzkdHC2ou3gpyGHqcL7aKzwTve**
LiteCoin: LiYp3Dg11N5BgV8qKW42ubSZXFmjDByjoV

Figure 7 17th July 2013 Bitcoin and Litecoin on website piratereverse.info

It by-passes the dependency on trusted 3rd party entities acting as middle-men and tries to prevent *double spending* [11] through the use of hash based peering mechanisms incorporating a trail or work sheet described through a proof-of-work algorithm.
Despite being designed for general use and a fast and transparent alternative means of payment, it has sparked a lot of controversy as the Federal Government of the United States of America and a number of other governments have identified it as a threat to currency and a means of promoting illegal and illicit activities, one example of which is the Silk Road – an anonymous online market place where illegal goods are traded [22]. The main reason for this is that the identity of users is kept secret and that provides an opportunity for people with bad motives to take advantage of as has been the case. In contrast, MoSHADi aims to keep the identity of users known across the system.

2. Mondex:

It is an electronic financial system that utilizes smart cards as electronic purses. Each card stores financial value (equivalent to cash) as electronic information on a microchip and provides operations for making financial transactions with other cards via a communication device [9]. It was originally developed by Tim Jones and Graham Higgins for the National Westminster Bank of the United Kingdom in 1991 and later sold to MasterCard International. Electronic cash is stored securely on a smart card and designed to complement credit and debit cards operating efficiently and presenting marketing opportunities for retailers and service providers. The value is stored on Integrated Circuits (I.C.) called Electronic Purses and each participant in the system is given one.

The system is structured to rely on both software and a comprehensive set of hardware ranging from telephones to cashless Automated Teller Machines and Point of Sale Systems to function efficiently. *Cryptography* is used in the system where for each transaction message exchanged is a unique key that could be used only once. The e-purses security is consolidated by a mechanism that damages the contents upon intrusion [10]. The biggest weakness of Mondex is the fact that it is heavily isolated from other digital money technologies and depends completely on infrastructure and devices developed to specially accommodate it. In this respect, the MoSHADi system irons out the problem by using open source technology and providing means to interact with existing systems.

3. Octopus:

Octopus is a contactless electronic payment device originally developed for public transport systems in Hong-Kong. Launched in 1997, Octopus cards enabled commuters to travel across more than one aligned transport system in the region. It has since been adapted for use in areas other than transport such as retail.

Like Mondex, it uses electronic cards that have chips in them maintaining a record of payment information and monetary value. The system makes use of *RFID* technology that allows proximity communication. Essentially it integrates pervasive or ubiquitous computing as a fundamental aspect [14]. Like Mondex, Octopus is proprietary and isolated.

4. Google Wallet:

Developed by the tech-giant, Google, Google Wallet is essentially a “wallet for the cloud” [19] used to purchase things and pay for services off the internet. It provides a link between the physical world’s economy (real world economy) and the virtual world and aligns clients debit and credit card numbers with the accounts they create on it and this allows the system to not be restricted to a virtual domain.

The system also allows money to be sent through the email service Gmail [20] although at the time of writing, only available to users in the United States of America. The wallet utilises NFC technology in real world situations to make it easy for users to operate. One area where it is widely used is in the purchasing of user applications for mobile devices off web stores.

The problem with Google Wallet is that it isn't directly available to people outside Google's system and it is heavily regional in that some features and functionality can't be accessed from certain regions.

2.3 Summary

Money has come a long way since its first use and it has taken various forms of representation with different systems of regulation developed in the past few years. Today, as we live in an information age, various technologies are developing around computerised systems based on controlled data networks allowing money value to be used openly across a global economic system, facilitating trade.

Chapter 3

Methodology

3.1 Introduction

The approach taken was protoyping with module based development. Developed modules were put together to form test systems, effectively prototypes, on which various aspects testing functionality were tried out. Working components were maintained and improved upon, while non-functioning components were disabled and others ultimately dropped.

Basically a hybrid approach involving the use of module based development and prototyping led to the development of a shadow system was evolved into the final working system. At the heart of the entire process was rapid coding with multiple alternative implementations of the procedures embodying the algorithms developed for the project and the most suitable assimilated into the system.

The image below describes the whole approach taken and to some degree, illustrating how the final system was developed.

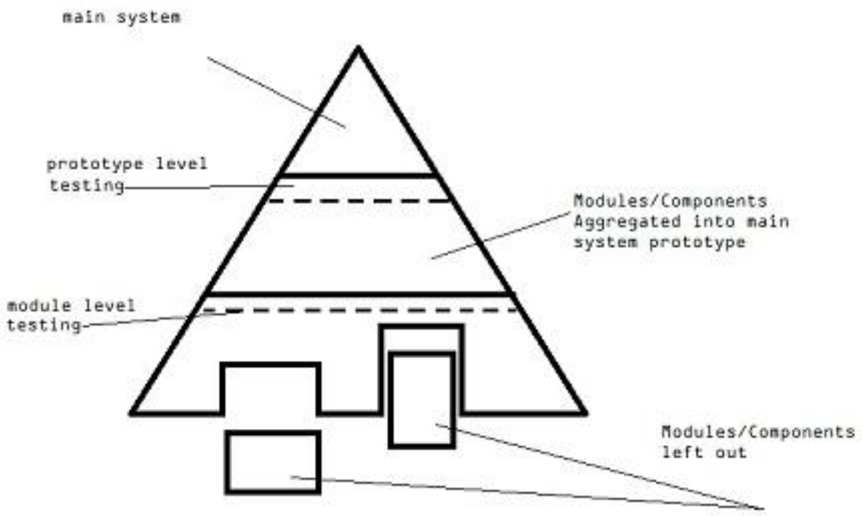

main system
prototype level testing
Modules/Components Aggregated into main system prototype
module level testing
Modules/Components left out

Figure 8: Approach Taken

The wide base of the pyramid describing the model symbolises the multiple routes taken to implement algorithms describing how to handle and achieve functionality required in the system. Some components and modules developed but not integrated were released and others continued to the test phase. Despite loss of extra functionality masked behind omitted components (*see discussion*), vital to this undertaking was the learning experience and proof that multiple forms of information representation and interaction with computer systems with a goal of improving the way our finances are handled in a digitally inclined society could be achieved.

Module based testing was done to ensure each component performed according to plan before integration into the shadow system (main prototype). This was done in order to quickly identify problems more isolated way and have them handled. Modules that passed module level testing were then 'glued' into a shadow system forming a visible skeleton of the final system and tested for unified operation. Ironing out problems identified at prototype level testing; the shadow system was evolved into the main working system.

3.2 System Requirements

3.2.1 Functional Requirements

Functional requirements of a system describe the functionality and services provided by the system, they describe what the system does. With respect to this system, the functional requirements include:

- Allow users to access their values of money with ease. Identification, verification and validation of users across the system grants access

- Express information about value in a variety of ways according to users liking, within the confines of the system's current level of understanding with regard to interaction channels.
- Accommodate means of transferring value between people, people and institutions and between institutions and facilitate flow of value.
- Present users with interaction mechanisms across different technologies; standalone and web to begin with.
- Allow financial institutions to easily handle value floated through them

Use case diagrams to model these requirements:

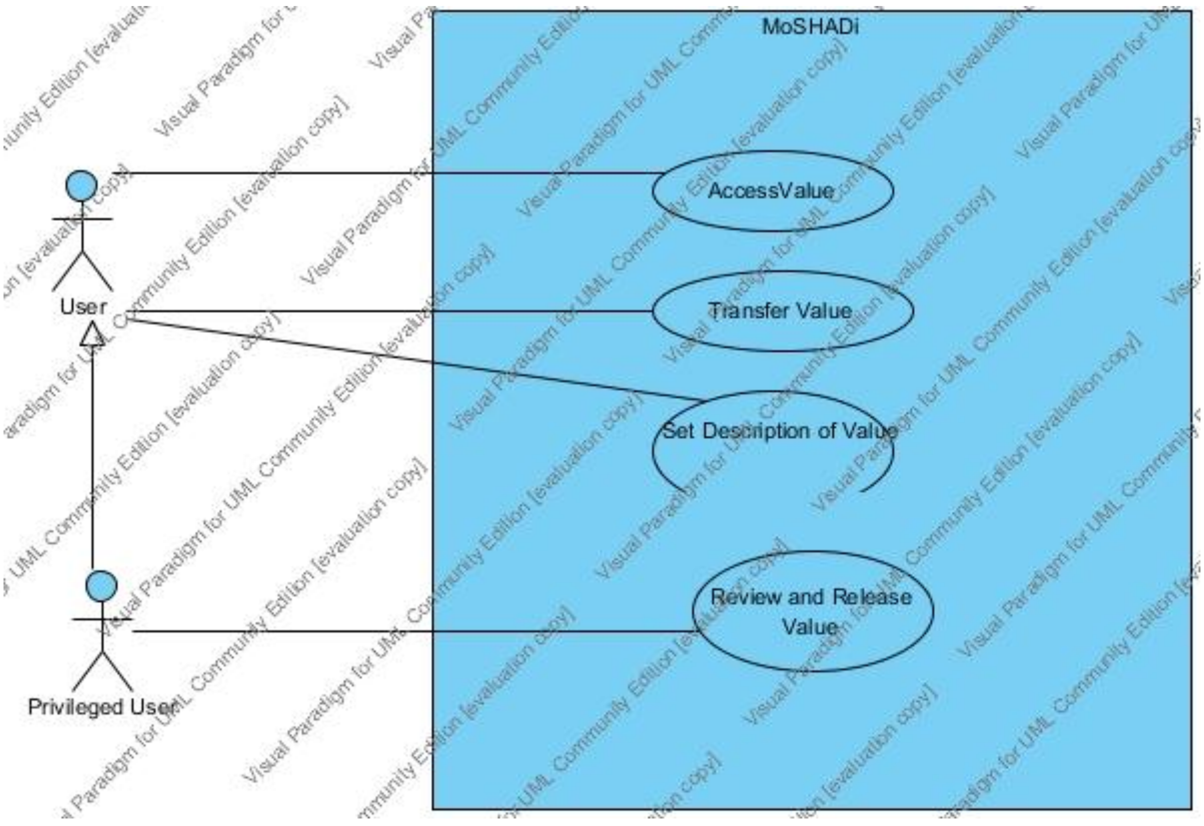


Figure 9 Use Case diagrams for User interaction with system

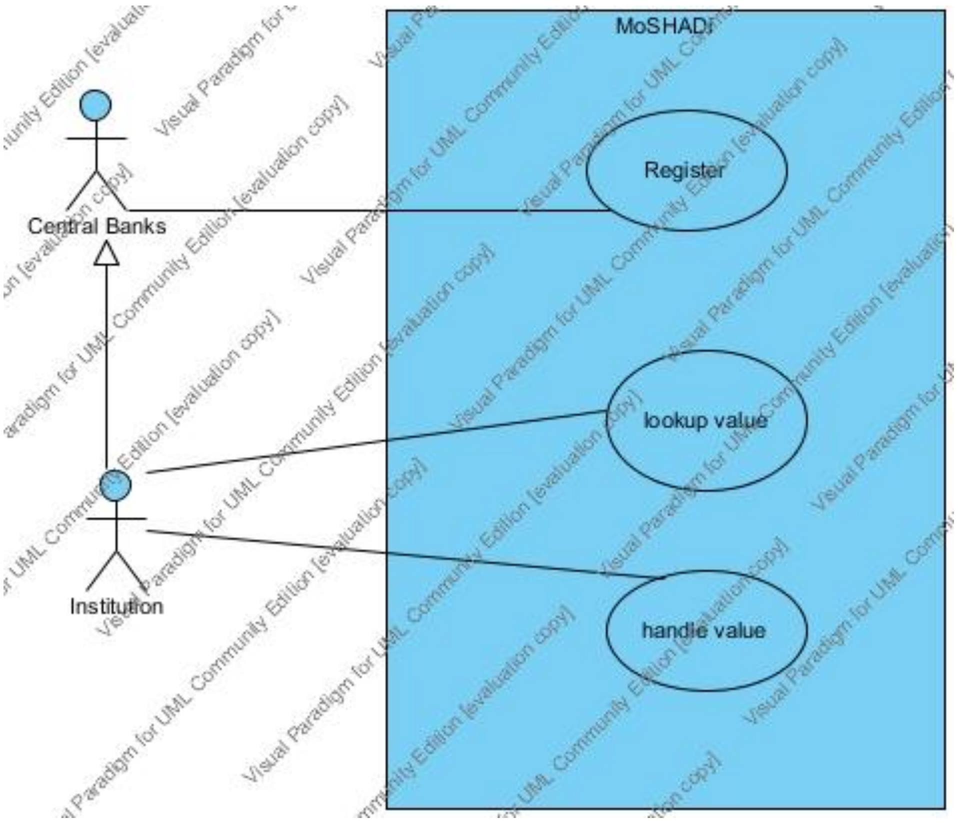


Figure 10 Use Case for accommodating institutions

3.2.2 Non-functional Requirements

Non-functional requirements have to do with aspects of the desired system that manifest at levels beyond the confines of specific functions of the system, more as emergent properties. With respect to this system for which this literature is based, the non-functional requirements include:

- Portability; This is a fundamental requirement of the system in order for it to be used across a variety of capable devices.
- Responsiveness and Reliability; In order to present resilient, ever ready system, accessibility and responsiveness are key requirements for the system.
- A high degree of usability must be achieved for the intended target is primarily the common man. Basic intuition and initiative should be embodied at all levels of interaction.
- The system must reflect high levels of integrity, ensuring consistency in values owned users at all times.

In order to run the client side applications, certain specifications need to be met. Among them:

- At least 1 Gigabyte of Primary Memory
- In built or installed Web Cam
- Integrated or installed stereo speakers
- Windows XP or later
- Screen resolution of at least 640 x 480
- Java Runtime Environment installed, jre, version 6 or higher
- An internet connection with minimum download speed of about 460 kilobits per second and upload speeds minimum 56 kilobits per second.
- Browsers supporting php, javascript and html 5
- Administrator level user

Mobile devices can access the system's web version and a specialised Java Server system designed to do number processing works with the web server. This system requires a Java Runtime Environment 6 or higher installed and to operate on ports 5050, 5051 and 5052 and an instance of MySQL server software.

3.3 System Design & Implementation

To describe the design of the system, Unified Modelling Language, UML 2.0 was used. Using this, the database systems were modelled through Entity Relationship Diagrams and then the package and interaction diagrams for the java based applications and finally a model of the web application.

Database Models:

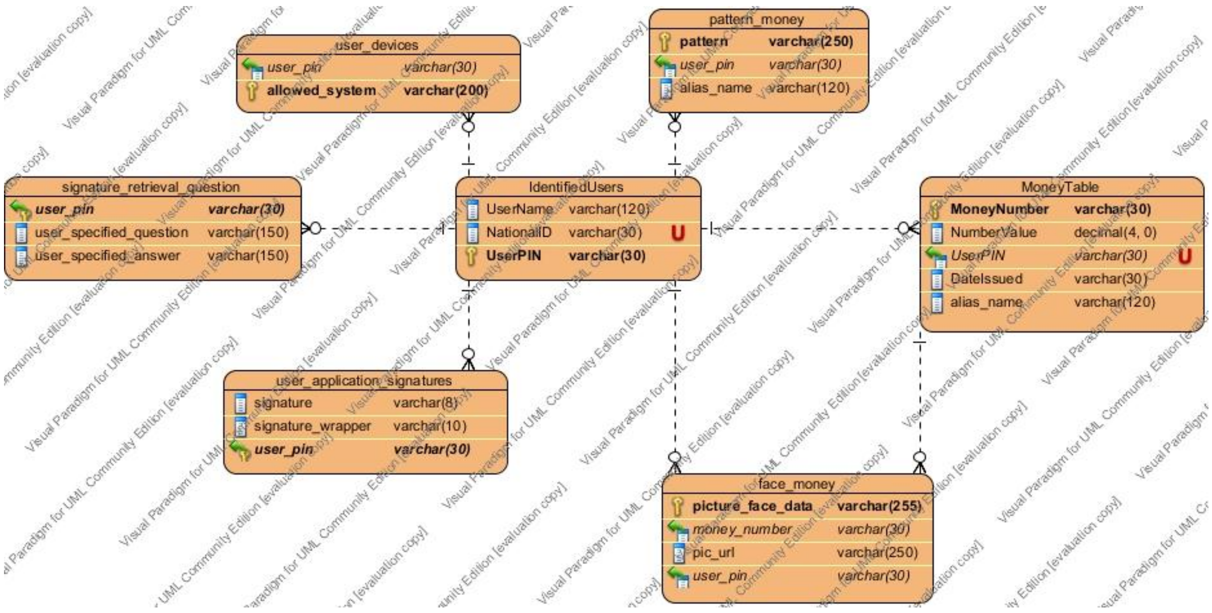


Figure 11 Relational Database model for the system

This diagram shows the structure of the database tables providing the foundation of the system. It is arranged around the user who is uniquely identified in the system and has their details included in the table *identifiedusers*. The money table *moneytable* holds system information of values in use and owned by a known user.

A single user can have many values of money and those can be described in alternative ways through patterns and faces. To do this, information relating to a given value is stored in the *pattern_money* table for pattern-described money and *face_money* for money value identified through user faces in pictures.

There are tables relating to device management that hold information on devices from where users can access the system beyond those that are open to the general public. The table *user_devices* holds the device details belonging to a user and the table *user_application_signatures* and *signature_retrieval_*question hold signature information and question-answer details relating to means of recovering signature numbers or requesting for new ones. The signature is a number similar to a licence key for software only it is a guiding number sent together with user login details and system information for validation and verification during login. It is designed around an identity revealing algorithm specially developed and called *This is who I am.* It basically uses the supplied signature which is just a large number, as a wrapper that is unwrapped to reveal a hidden number in it known to the system and allows it to trace a given user. It is described below:

```
Input signatureWrapper

Output identity

set qIndex = 0,cIndex = 0,counter = 0,array digits;
```

```
qIndex = Integer.parseInt(signatureWrapper.substring(0,1));
cIndex = Integer.parseInt(signatureWrapper.substring(1,2));
digits = signatureWrapper.toCharArray();
digits[qIndex] = '?';
digits[cIndex] = ':';
counter = 0;
set string extractedSignature = "";
character firstDigit = digits[0];
character lastDigit = digits[digits.length-1];
digits[0] = digits[qIndex];
digits[qIndex] = firstDigit;
digits[digits.length-1] = digits[cIndex];
digits[cIndex] = lastDigit;

while(counter < digits.length){
    extractedSignature = extractedSignature+""+digits[counter];
    counter++;
}
return extractedSignature;
```

This feature was included as a means of providing security against access to a user's value of money by unauthorized parties.

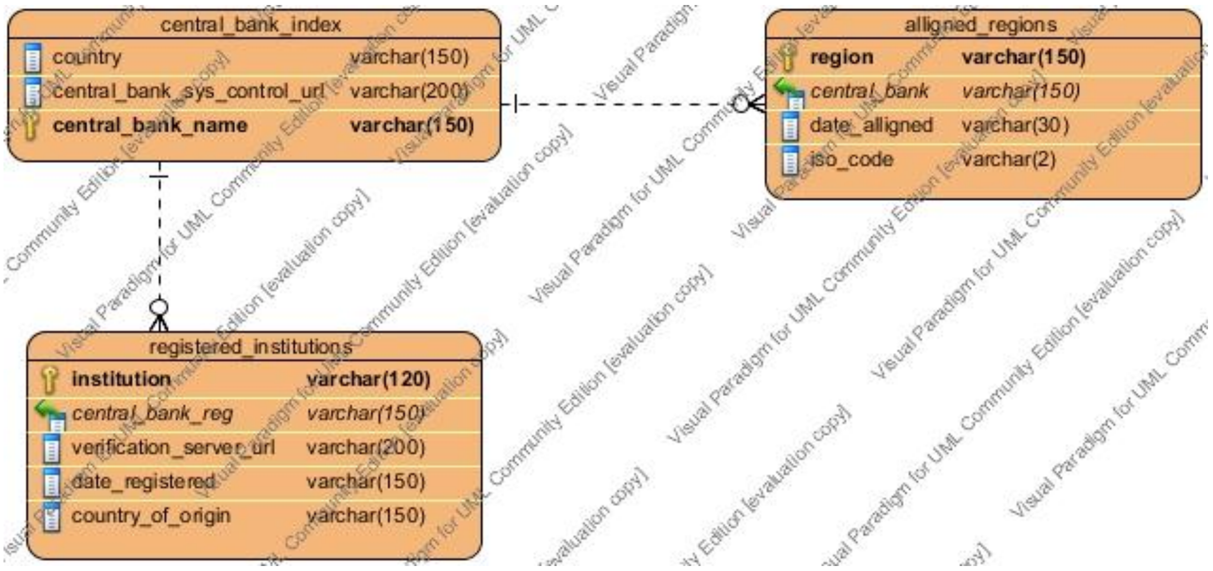

central_bank_index
country varchar(150)
central_bank_sys_control_url varchar(200)
central_bank_name varchar(150)
alligned_regions
region varchar(150)
central_bank varchar(150)
date_alligned varchar(30)
iso_code varchar(2)
registered_institutions
institution varchar(120)
central_bank_reg varchar(150)
verification_server_url varchar(200)
date_registered varchar(150)
country_of_origin varchar(150)

Figure 12 Relational database model for system accommodating institutions

This Entity Relationship Diagram shows institutions aligned to handle flow. Each one is registered through the central bank of the country or region of origin and each central bank is identified under aligned regions of the world. Details of all regions and nations of the world in terms of name and ISO codes were keyed into the system and the actual central banks for those countries/regions mapped to them.

The model of the web based application is summarised:

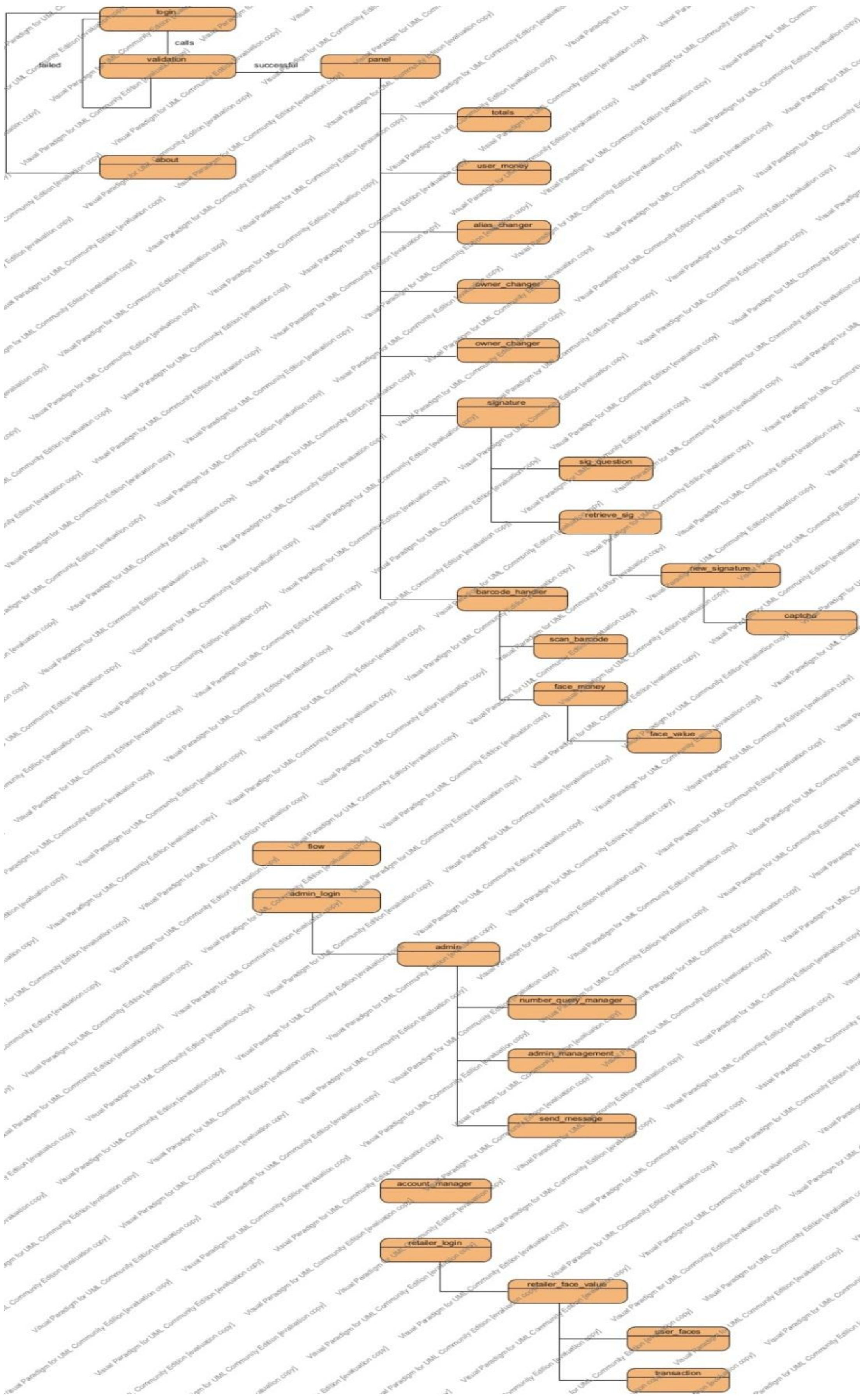


Figure 13 High level site map for Web application

Each component in the web application diagram represents a PHP page.

Class and Package Diagrams to some aspects of the system, the standalone components, are described below:

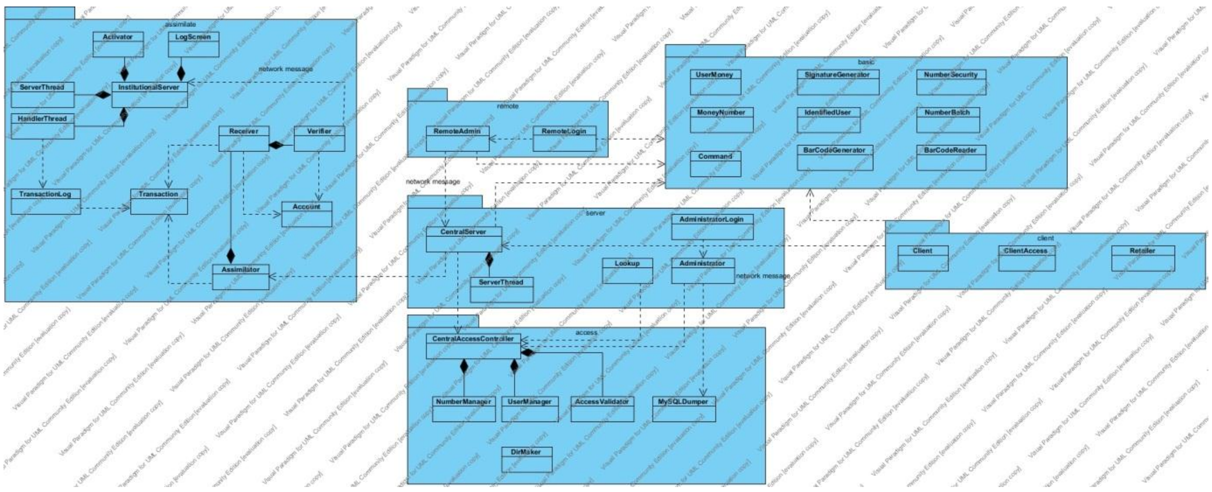


Figure 14 Outlining Package Diagram

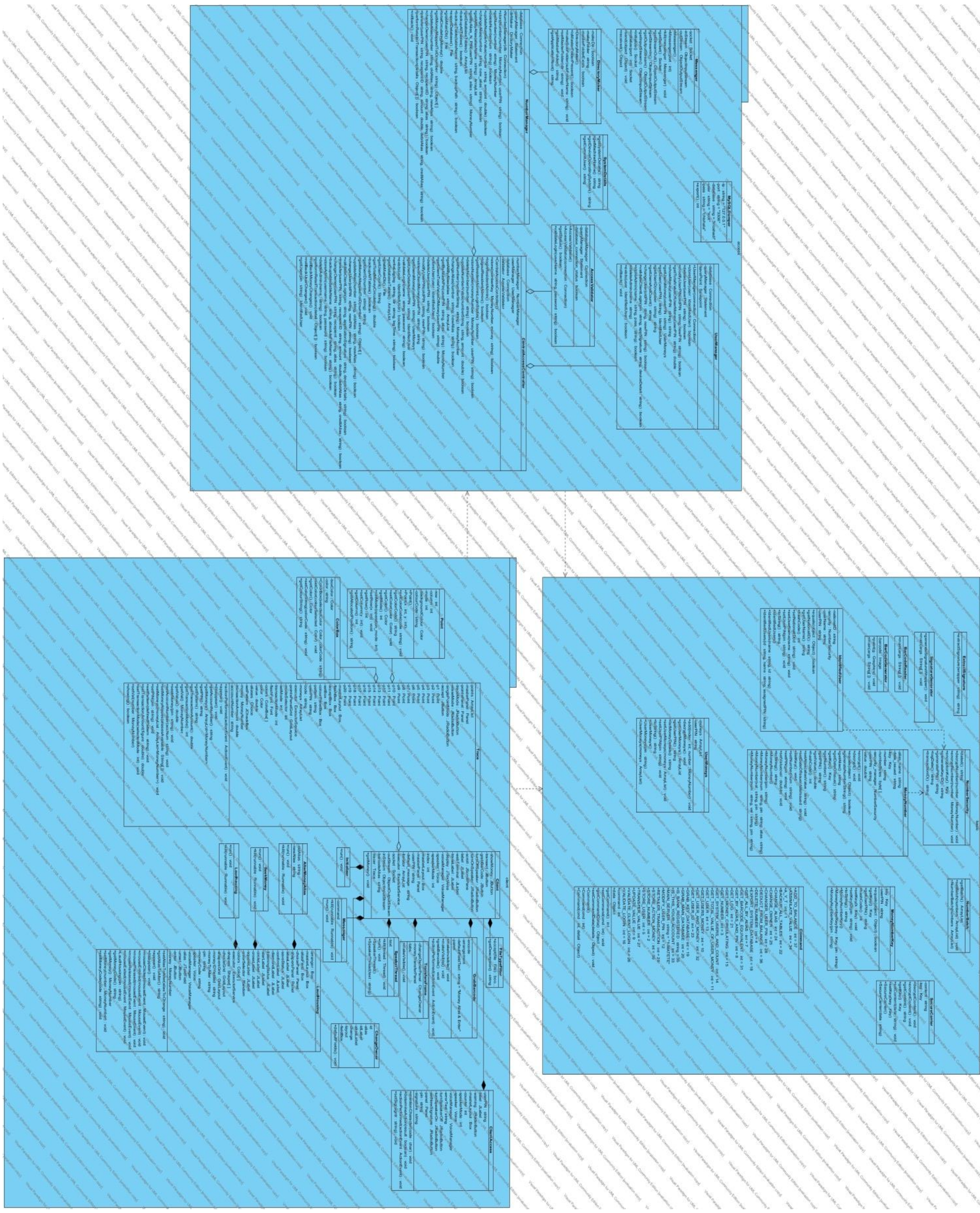


Figure 15 Master Package Diagram for the Java Standalone client application

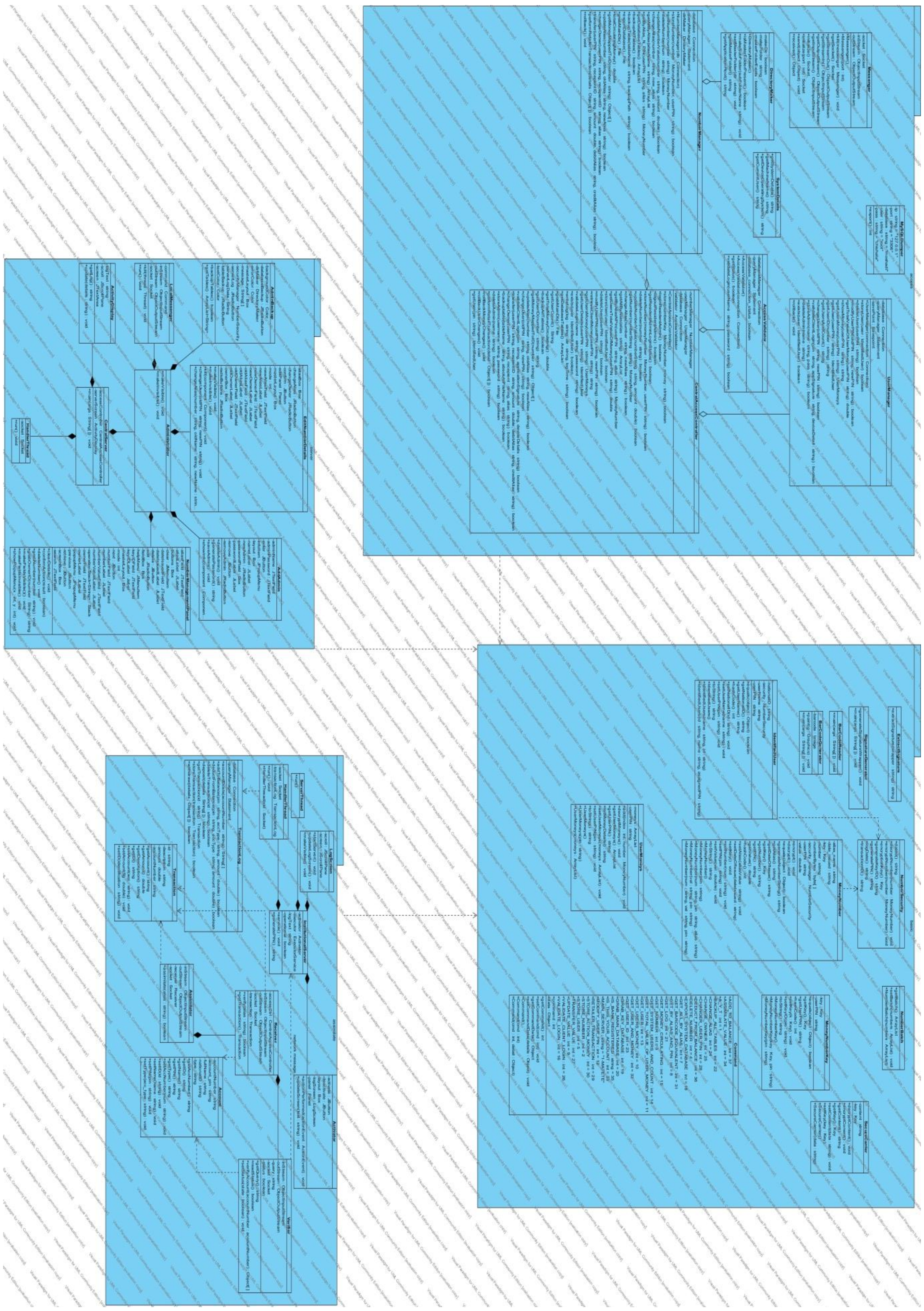


Figure 16 Master Package Diagram for the main Java based Server System

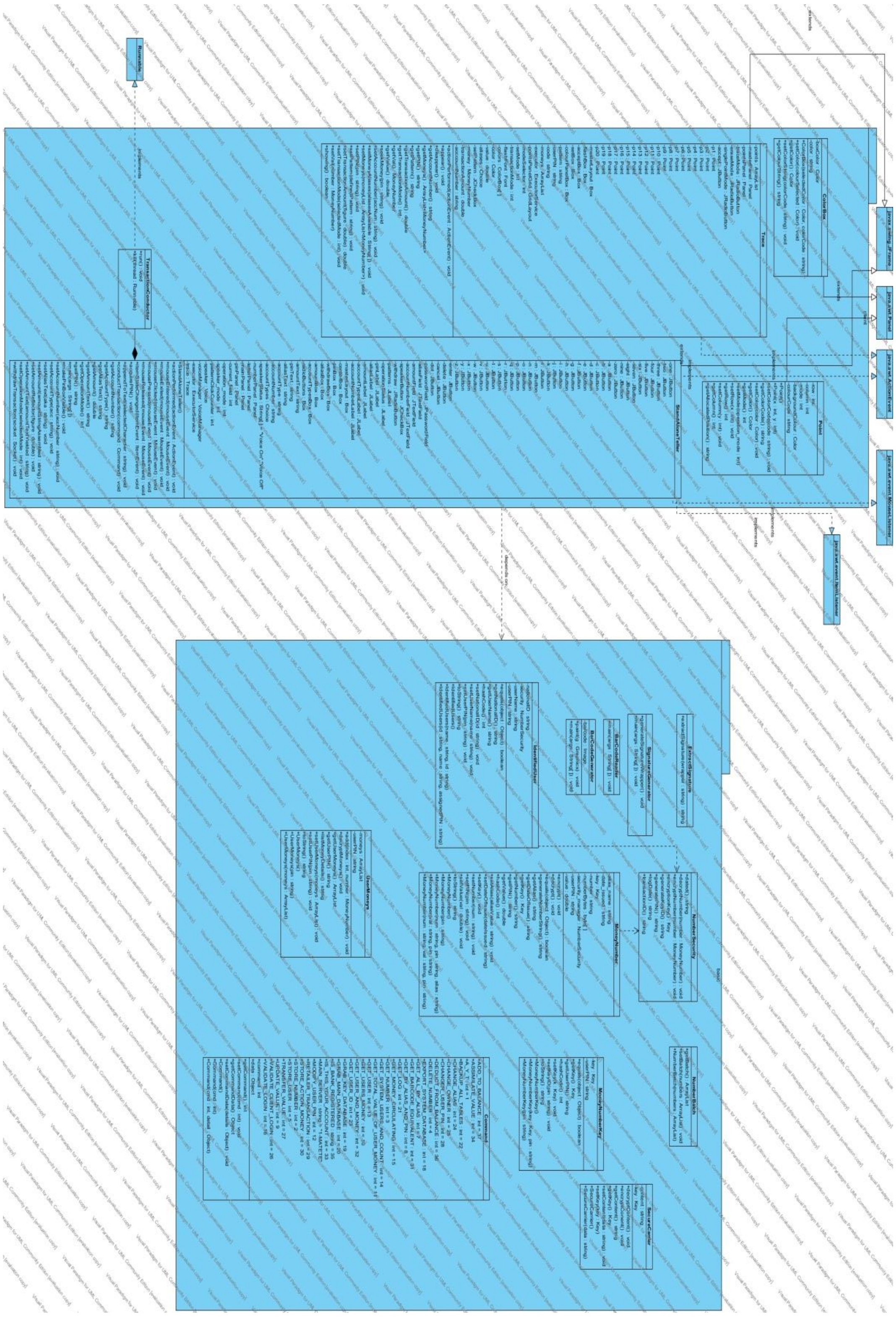


Figure 17 Package Diagram of a Standalone simulator of an adapted ATM

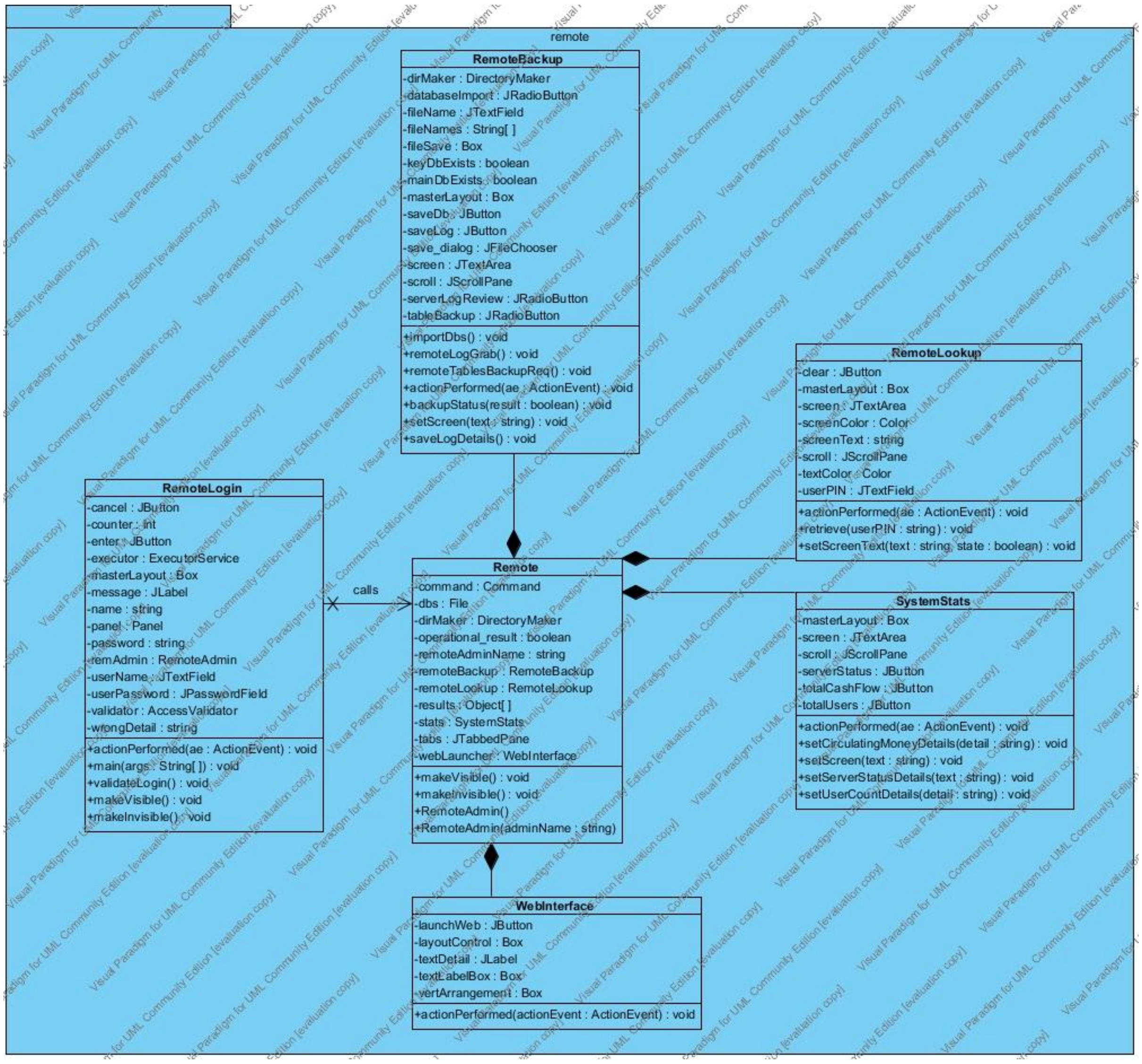


Figure 18 Package view diagram for Remote Administration standalone application

Sequence Diagrams

General sequence diagrams to show the time ordered interaction of components were developed and described below:

This sequence diagram shows how patterns are made on a grid by trapping mouse movements.

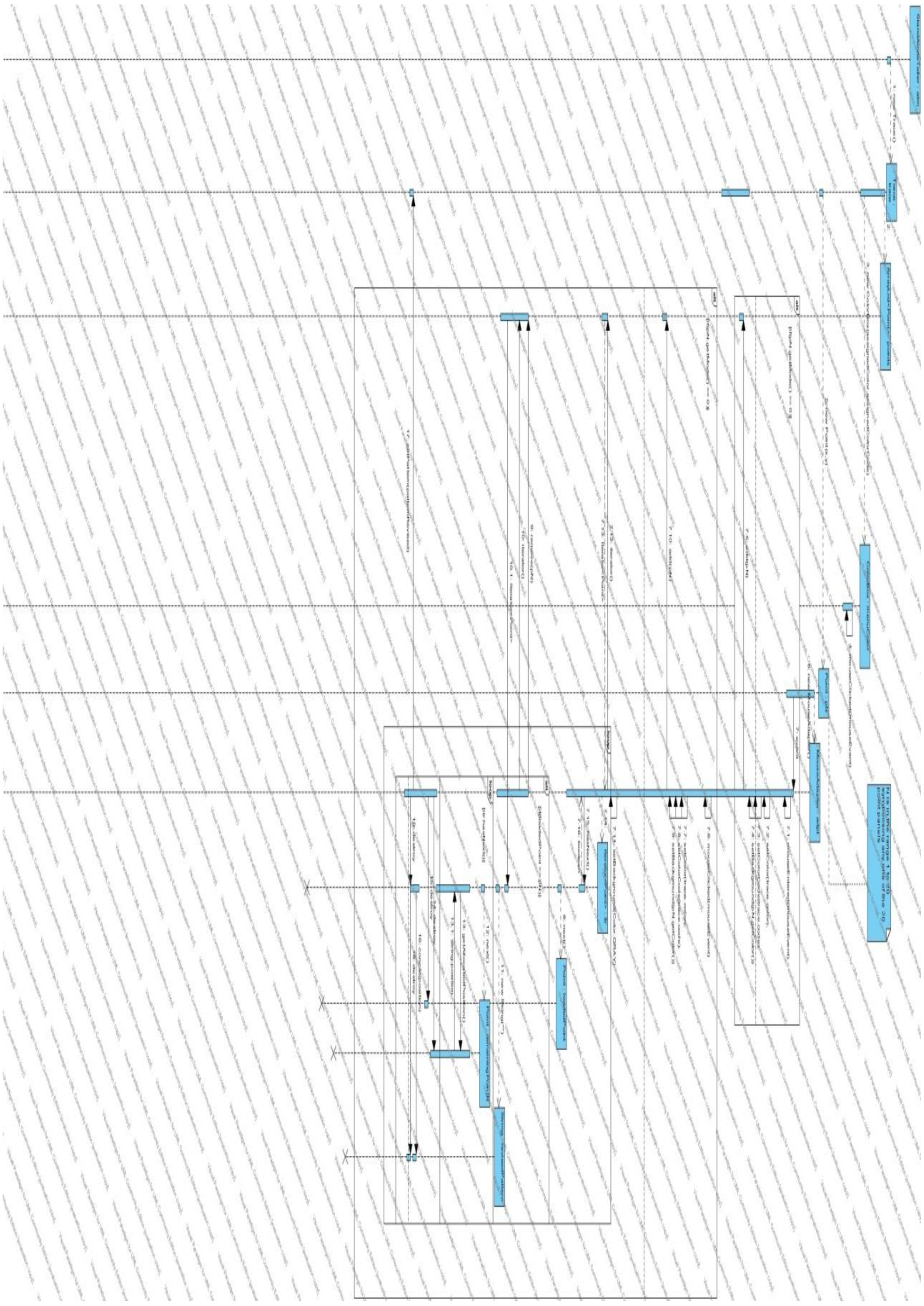


Figure 19 ATM Pattern control

This sequence diagram is for the ATM simulator, emulating an adapted ATM for this kind of money. It shows how patterns are used to query money value, effectively equivalent to reading a balance statement off the ATM.

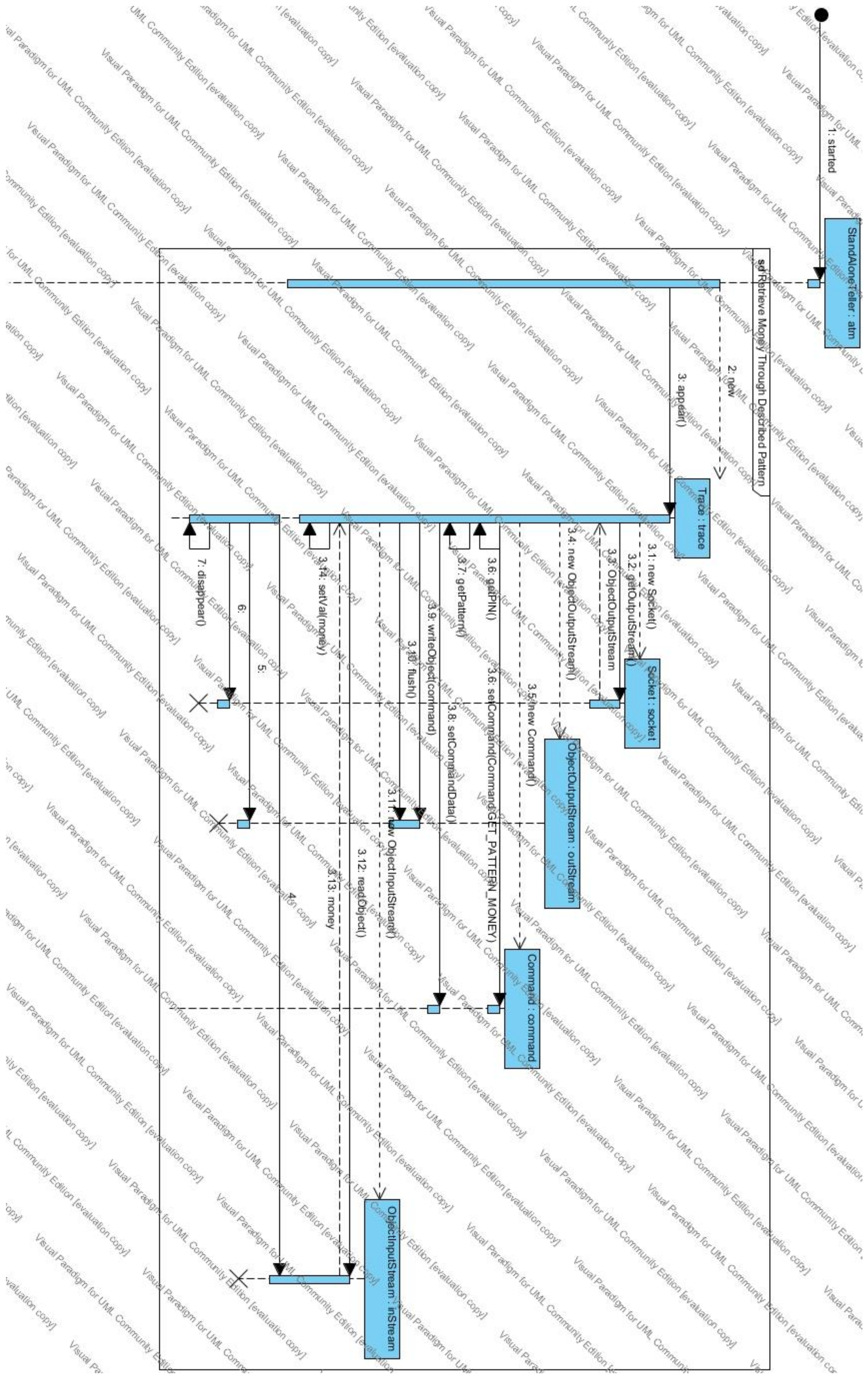


Figure 20 ATM Pattern interaction

The following sequence diagram shows what happens when trying to finalise a withdrawal:

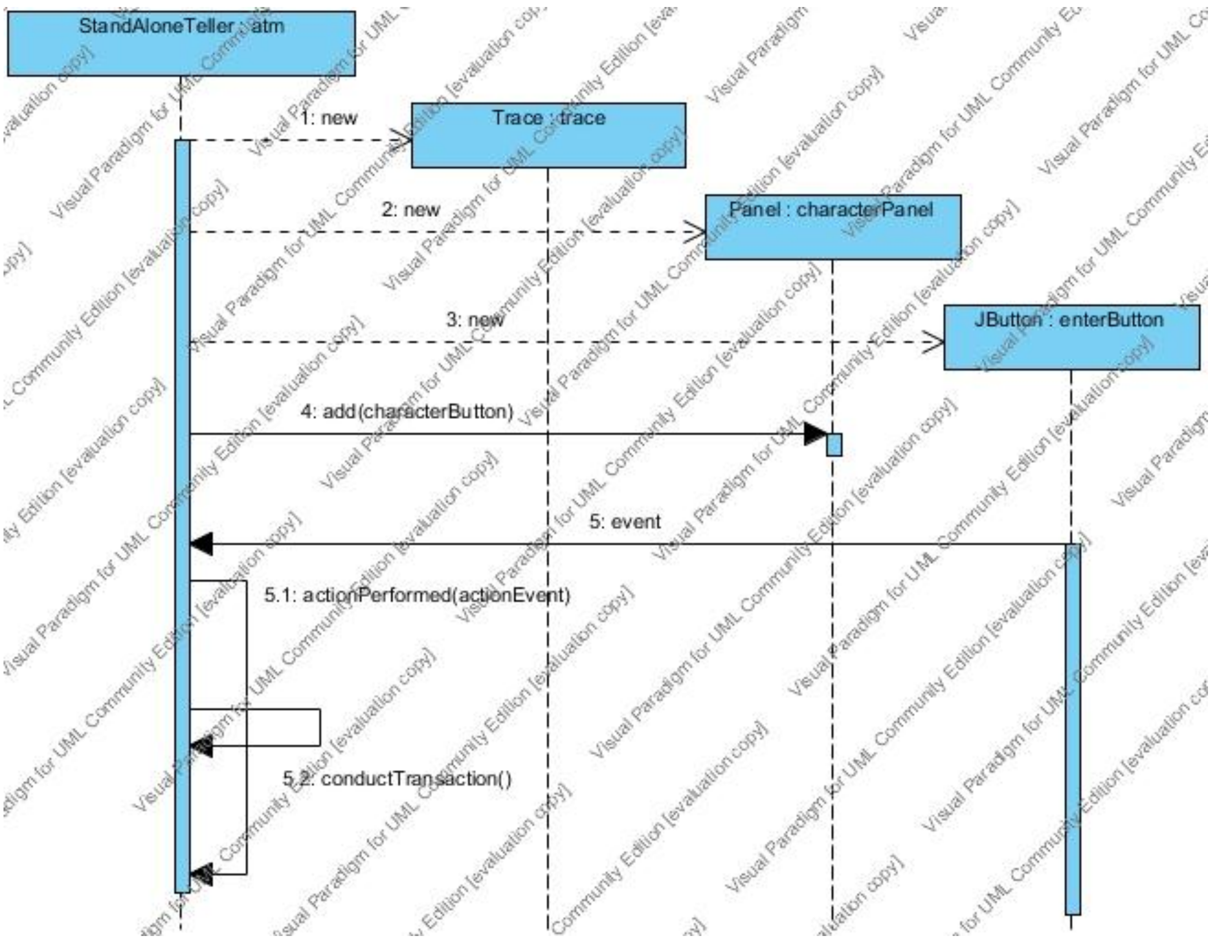


Figure 21 Finalising withdrawals for simulated ATM adapted to MoSHADi

The sequence diagram for text entry on focus of a field is shown below:

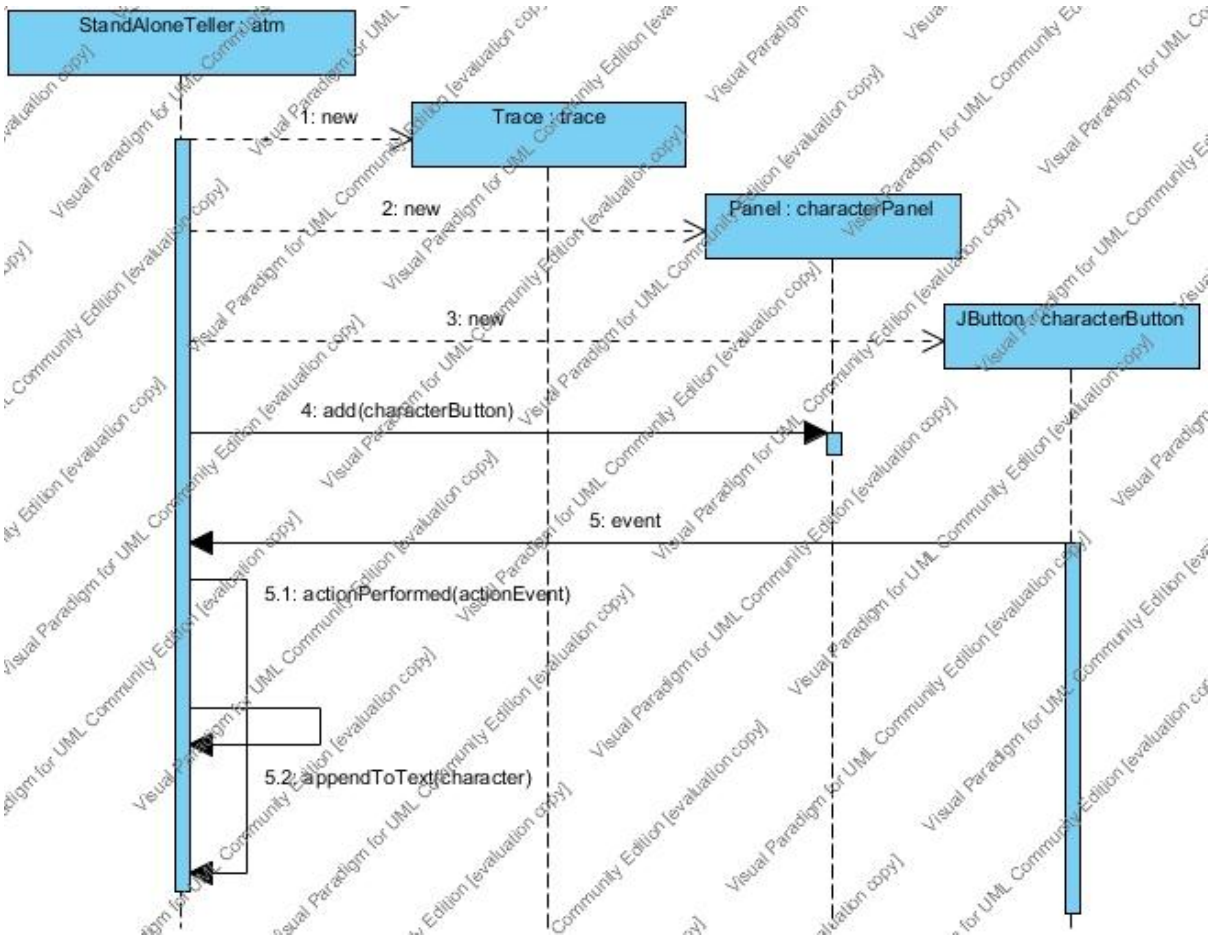


Figure 22 Text Entry

The sequence diagram for the client application login is shown below:

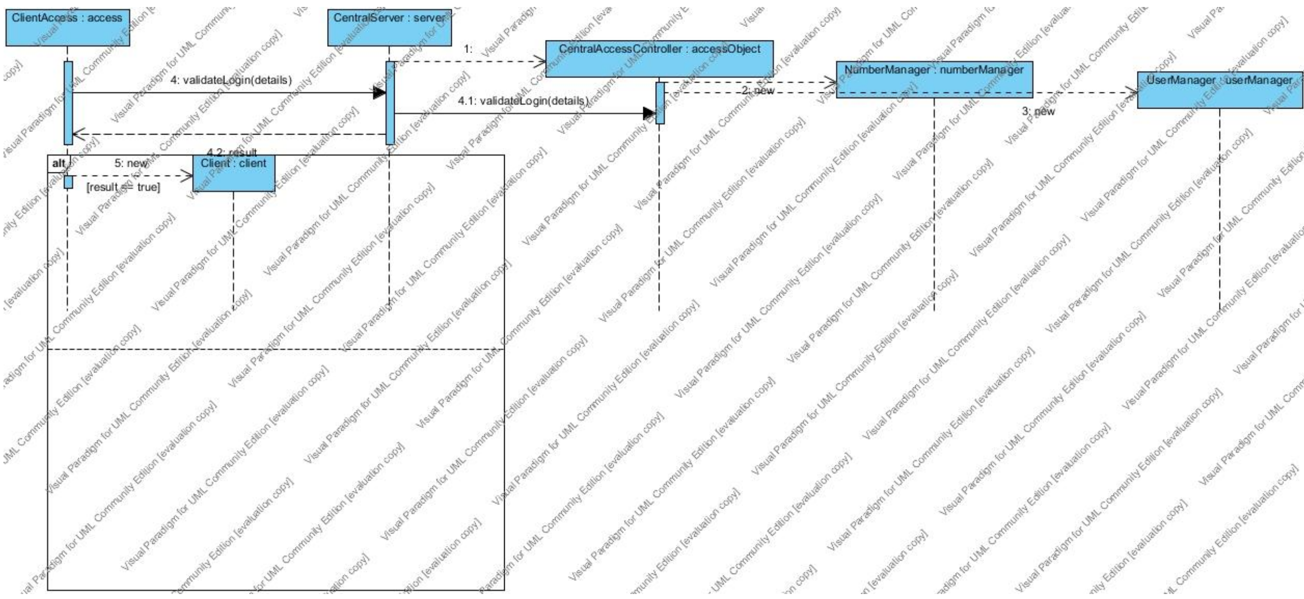
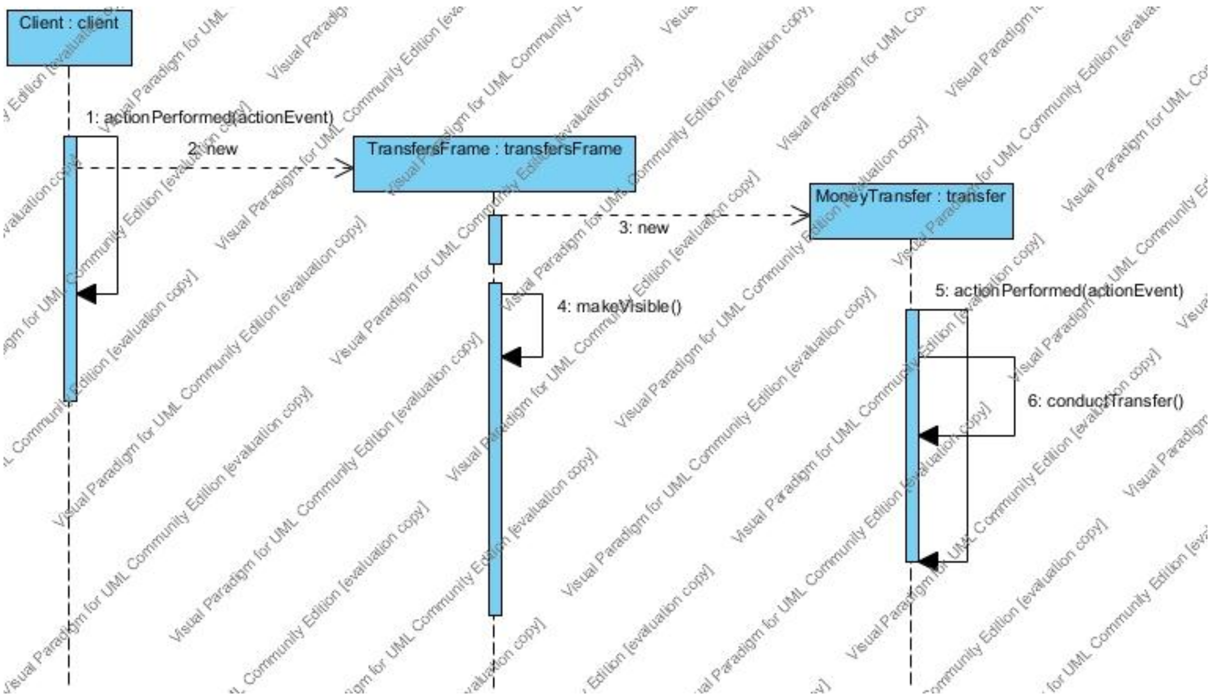

ClientAccess : access
CentralServer : server
CentralAccessController : accessObject
NumberManager : numberManager
UserManager : userManager
1:
4: validateLogin(details)
4.1: validateLogin(details)
2: new
3: new
4.2: result
alt
5: new
Client : client
[result == true]

Figure 23 Client Login

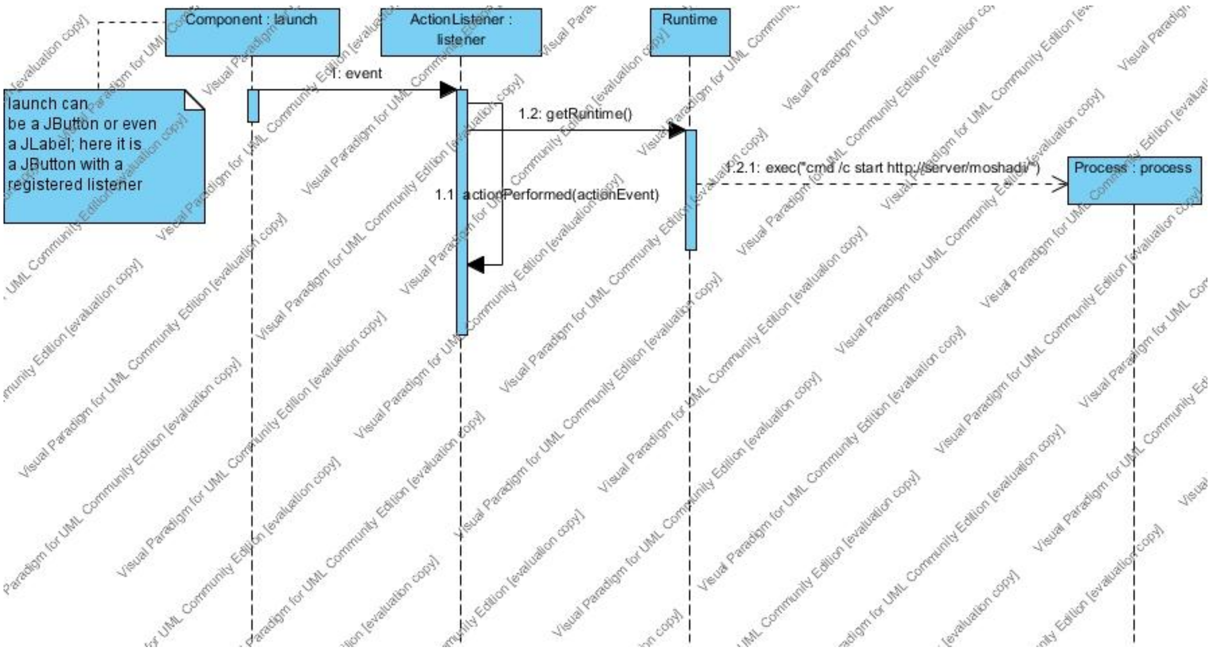
`



Client : client
1: actionPerformed(actionEvent)
2: new
TransfersFrame : transfersFrame
3: new
MoneyTransfer : transfer
4: makeVisible()
5: actionPerformed(actionEvent)
6: conductTransfer()

Figure 24 Transferring value

The sequence diagram below describes the interaction that leads to the standalone client directing a user over to the web application:



Component : launch
ActionListener : listener
Runtime
launch can be a JButton or even a JLabel; here it is a JButton with a registered listener
1: event
1.2: getRuntime()
1.1: actionPerformed(actionEvent)
1.2.1: exec("cmd /c start http://server/moshadi/")
Process : process

Figure 25 Directing user to web application

State Diagrams

Some state diagrams describing the various states that the Java based components of the system may be in are shown below:

1.

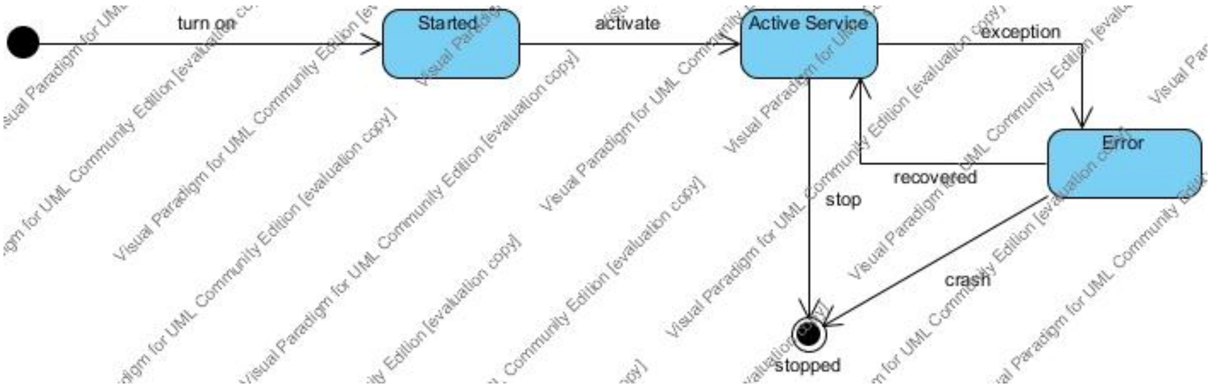


Figure 26Java Server State Machine Diagram

This state diagram shows the various states that the Java based server component may enter during operation.

2.

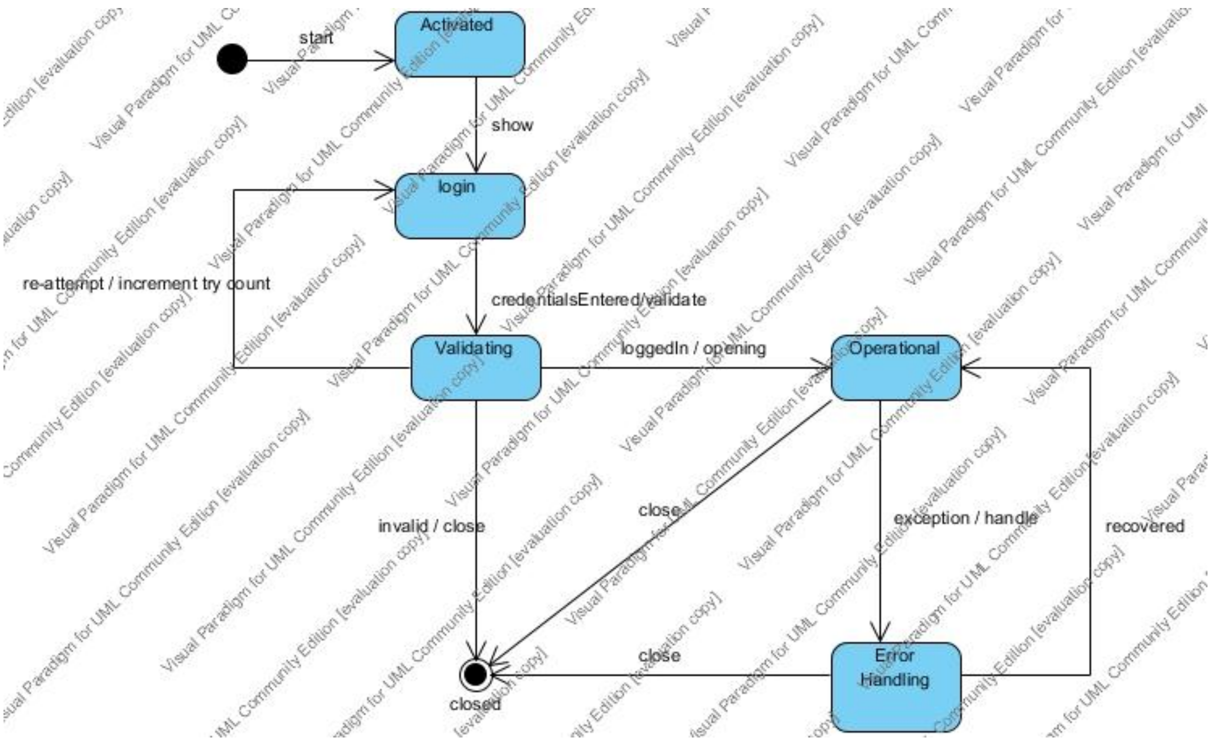


Figure 27 Client Standalone State Machine Diagram

This state diagram illustrates the states that a client standalone system may enter during operation. First it validates credentials supplied on the login window. If correct and valid, it opens the application for usage, otherwise it closes after 3 attempts. In operational state, any exceptions encountered lead to the error handling state where recovery may be attempted or closure in the case of fatal and unrecoverable exceptions.

3.

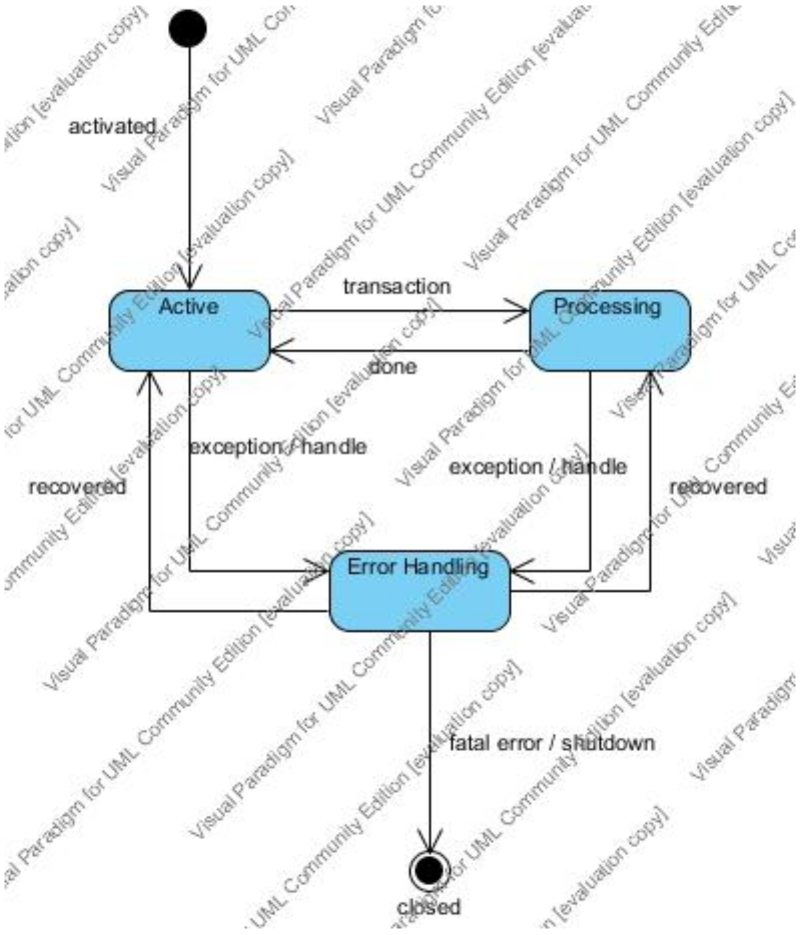


Figure 28 ATM Simulator State Diagram

The above state diagram shows the states that an adapted ATM application could enter during operation.

Deployment Diagram

The final system is designed to run against a Server, with client applications on devices such personal computers and mobile communication devices communicating with the server via network links and the internet. To start with, the system is designed in a centralised manner (but could be expanded towards a distributed system of servers). The deployment diagram below shows the arrangement of the system and the various components involved:

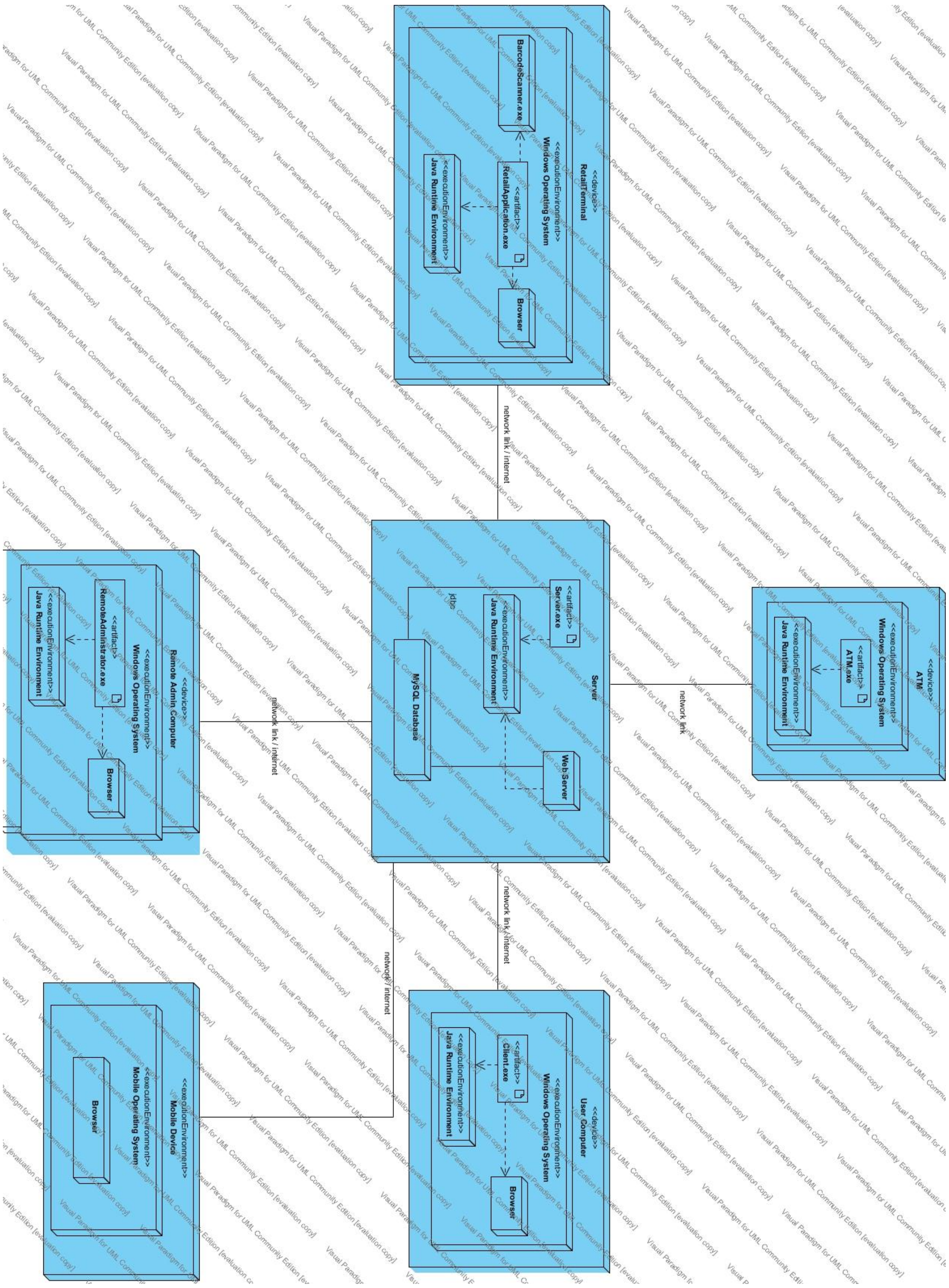


Figure 29 Deployment Diagram

Applications are wrapped up as Windows executables with dependencies on the Java Runtime Environment. Server components all see one relational database based on MySQL Database Management system and also make use of the Java Runtime Environment.

Moving on to the implementation, once the system was fully modelled, work began to realise it. Using a software called Xampp installed on a desktop computer, a web server was made and ready for development of the dynamic web application. Java SE was used to develop a java based Server working side by side with the web server and interfacing the same data set contained in MySQL databases.

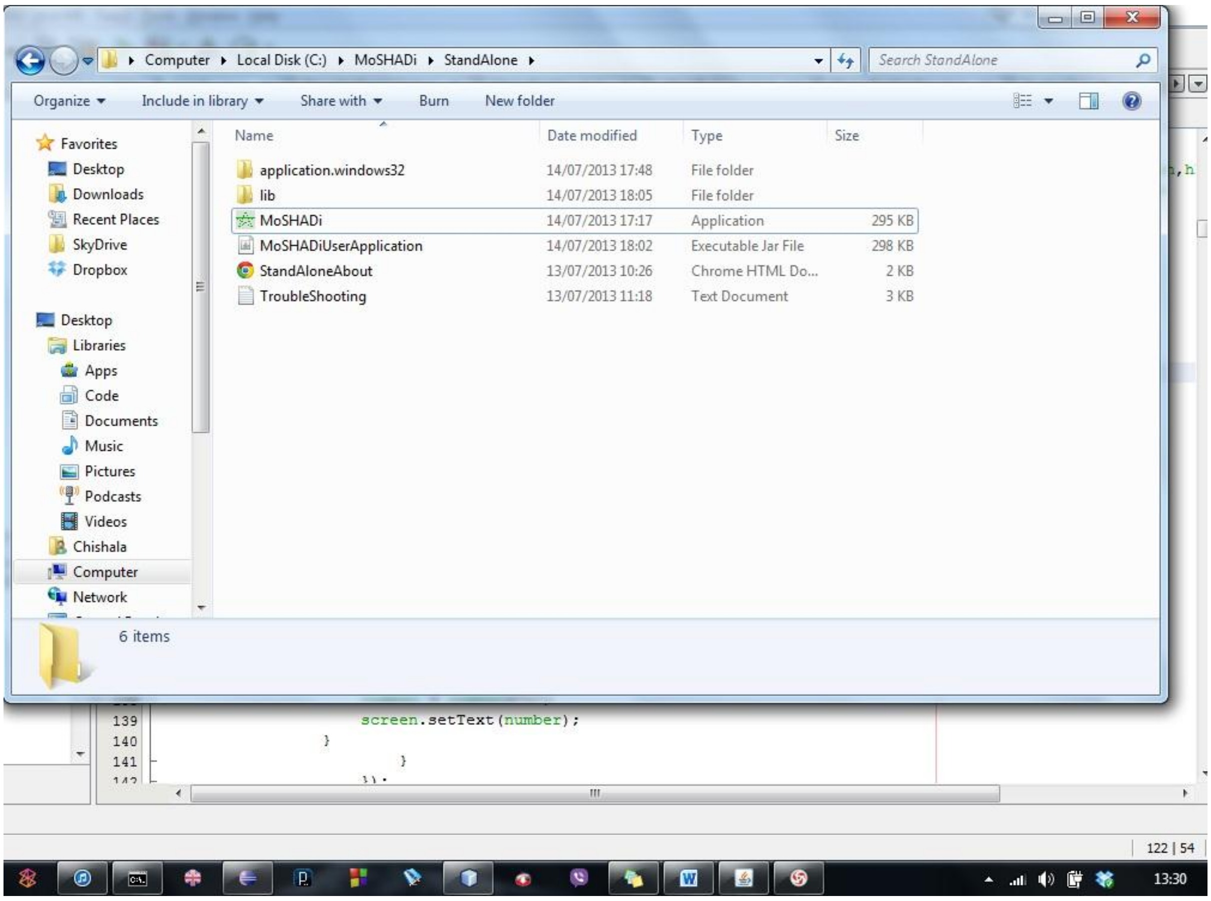
In terms of the Web application, PHP was used with a little JavaScript to realise speech synthesis on the web application and some aspects of face detection. The web application was also made the centre for downloading standalone applications and registering new users. For the java based standalone applications, the prototype standalone systems were made with the required libraries; the text to speech library, speech recognition library and the barcode code generation and scanning library. Client systems were then produced from the dependent packages, bundled into executable jar files and invoked using Windows Executable launcher programmes developed by wrapping executable jar files into .exe files with the help of a software called Launch4J.

For each standalone application, a number of interaction models were defined and incorporated and for those that couldn't be accommodated, the functionality was shifted to the web application and a clickable link to that feature placed on the standalone application.

Firstly, for action oriented interaction, pattern generation was made available by placing small color panels that would turn a certain color set by the user when the mouse is moved across them in a grid. The pattern generated is internally described by combining the panel number and color code as the single first letter of the color itself and concatenating them to form a textual representation of the pattern. Secondly, a feature to accommodate static facial recognition was incorporated by first providing a component on the standalone applications to take pictures of people and allow those pictures to be analysed and aligned with a value of money through the web application. Thirdly, speech controlled descriptions of money were incorporated via interaction with the system microphone. An integrated listener would pick up commands and compare them to a limited language dictionary and reflect them with a graphical rendition of the command. A textual pattern made by combining the commands uttered would internally represent the value aligned with the user's speech at the point of saving the speech controlled description and a text to speech component confirming the description the user just made. Fourthly, the web application was equipped with the capability of generating and scanning Quick Response (QR) codes masking a textual alias describing a value of money. The codes could also be photographed and presented to a code scanner utilising the Webcam of a computer and the alias of the value well recognized and used in the system to transact.

Sample screenshots of the applications at work are provided below:

Client Standalone Java Application:

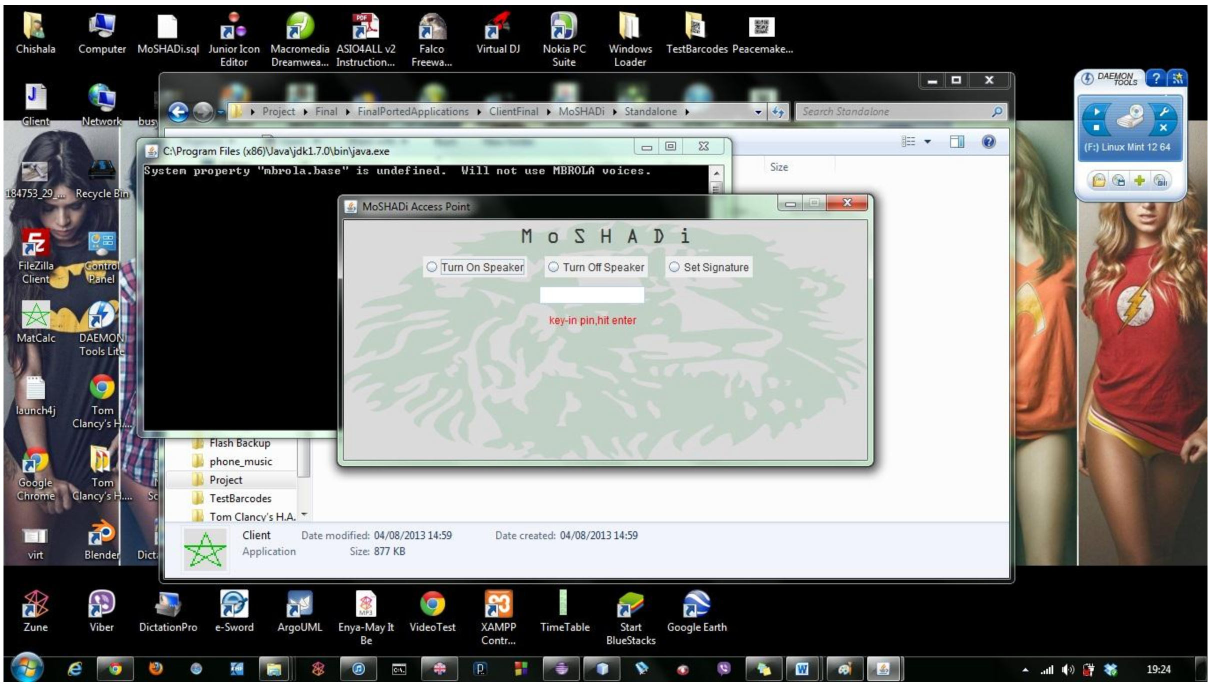


Name	Date modified	Type	Size
application.windows32	14/07/2013 17:48	File folder	
lib	14/07/2013 18:05	File folder	
MoSHADi	14/07/2013 17:17	Application	295 KB
MoSHADiUserApplication	14/07/2013 18:02	Executable Jar File	298 KB
StandAloneAbout	13/07/2013 10:26	Chrome HTML Do...	2 KB
TroubleShooting	13/07/2013 11:18	Text Document	3 KB

```
                screen.setText(number);
            }
                }
```

Figure 30 Starting the client application

Once setup, the directory for the client side is arranged like that of the image above were the drive C is used to hold the folders and iles used by the application. The directory precisely created is C:/MoSHADi/. The application to run is the one called MoSHADi of type .exe. Double clicking it, produces a launcher screen, which is hooked to the application file.

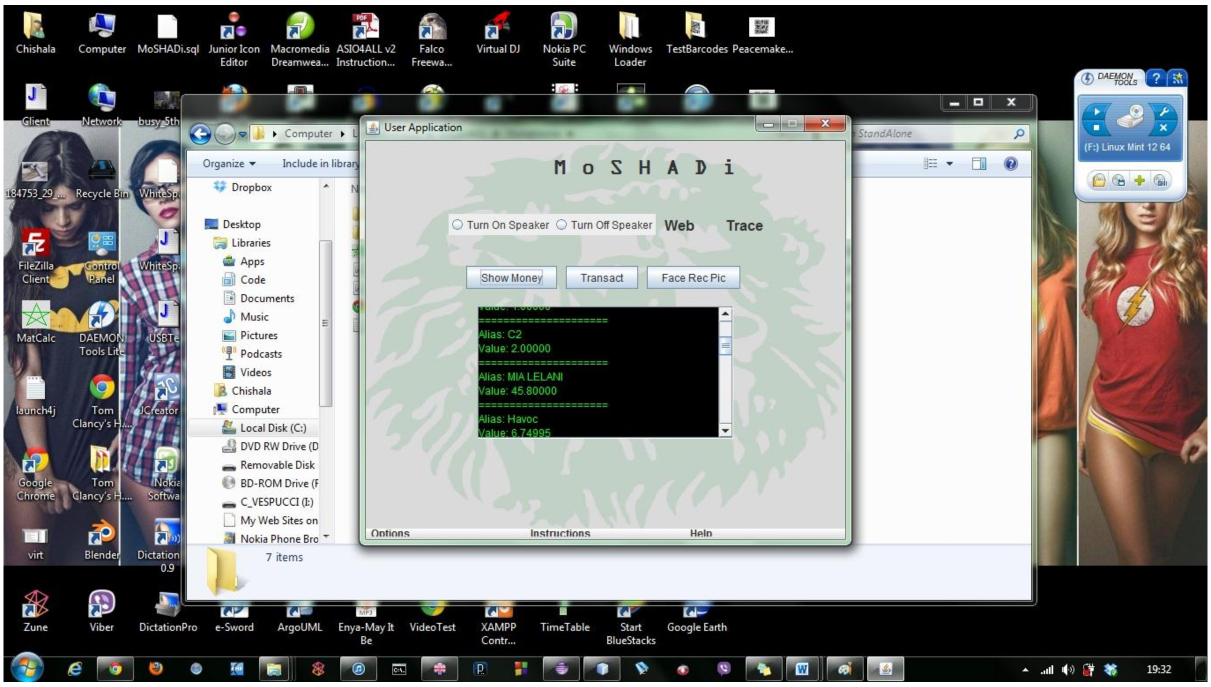

C:\Program Files (x86)\Java\jdk1.7.0\bin\java.exe
MoSHADi Access Point
M o S H A D i
Turn On Speaker
Turn Off Speaker
Set Signature
key-in pin,hit enter
Project ▸ Final ▸ FinalPortedApplications ▸ ClientFinal ▸ MoSHADi ▸ Standalone ▸
Client
Application
Date modified: 04/08/2013 14:59
Date created: 04/08/2013 14:59
Size: 877 KB

```
System property "mbrola.base" is undefined.  Will not use MBROLA voices.
```

Figure 31 Logging into the application

The login screen appears and by default, a speech guide is set to the ON state. This is done as a provision for the visually disabled but can be turned off by clicking "turn off speaker" option. Once login details are keyed in, they are sent over to the server for validation and a Boolean result indicating true for valid details or false for invalid details is received. If false, a counter reduces the login tries by 1 out of 3.

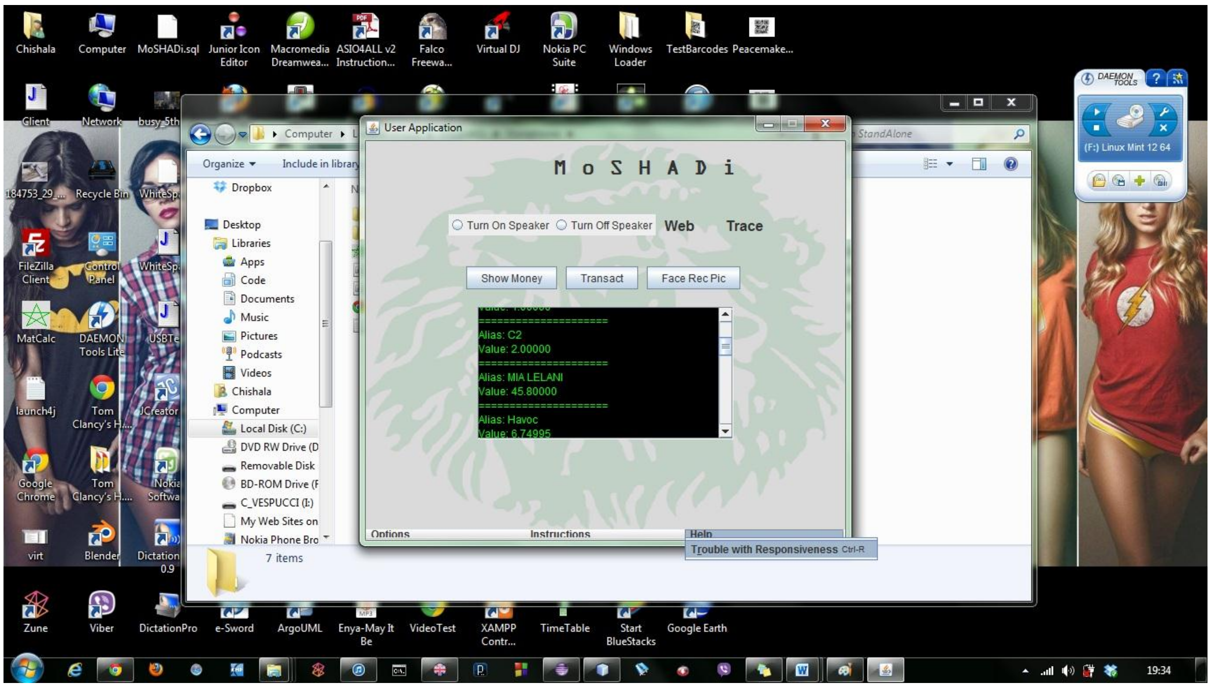
If login is successful, the main screen of interaction opens up, as shown below. Clicking the "show money" button retrieves all money of the user and displays them as text aliases with their values.



```
Value: 1.00000
=====================
Alias: C2
Value: 2.00000
=====================
Alias: MIA LELANI
Value: 45.80000
=====================
Alias: Havoc
Value: 6.74995
```

Figure 32 The Application as it appears after a successful log in

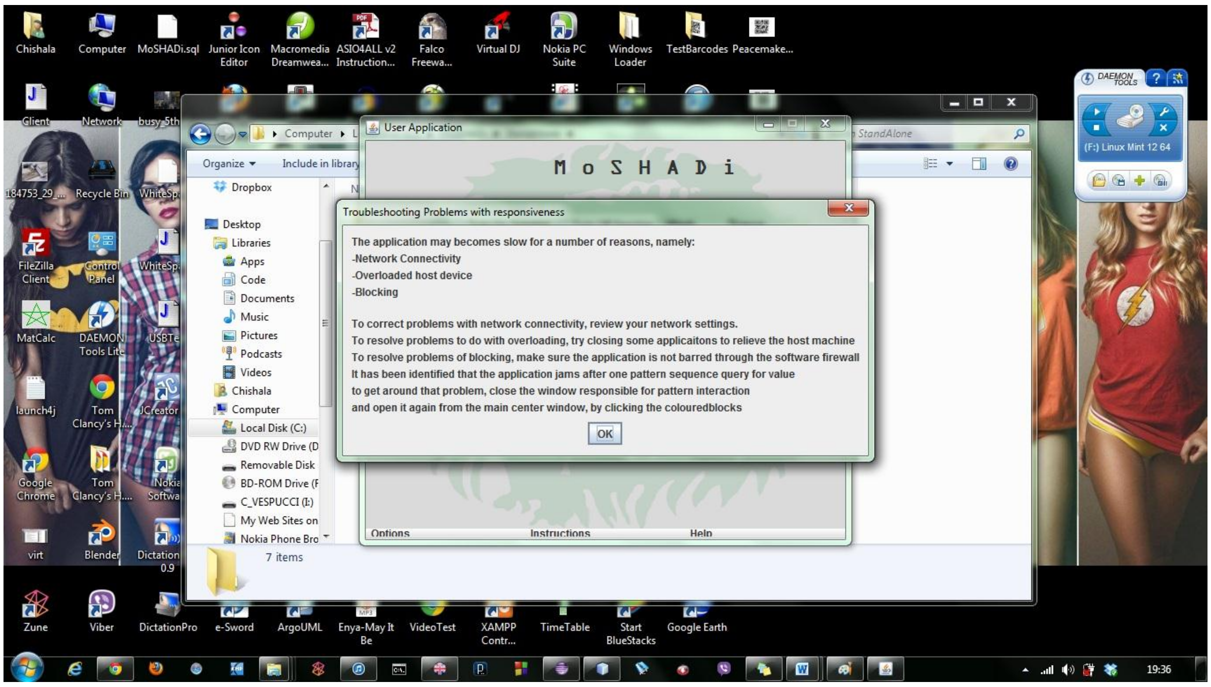
There is a menu at the bottom which provides guidance and options for the user.



```
Value: 1.00000
=====================
Alias: C2
Value: 2.00000
=====================
Alias: MIA LELANI
Value: 45.80000
=====================
Alias: Havoc
Value: 6.74995
```

Figure 33 Using the menu options available on the application

Clicking on (or using the combination CTRL+R) the option help option for example, produces:

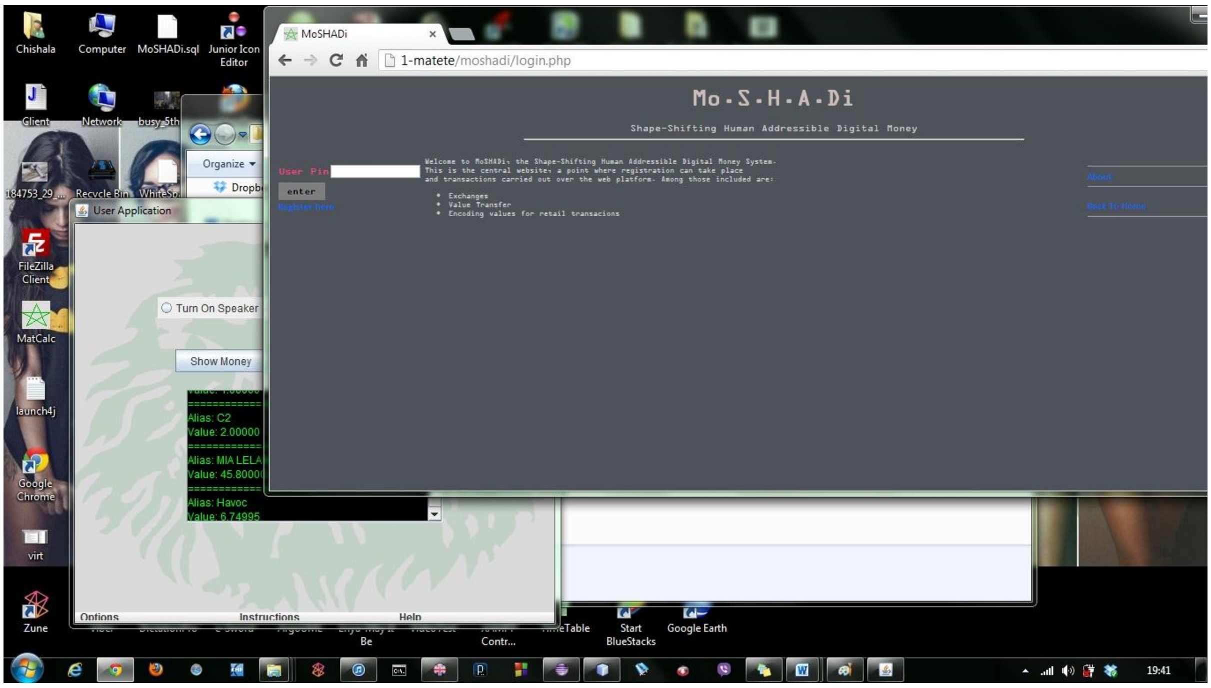

Troubleshooting Problems with responsiveness

The application may becomes slow for a number of reasons, namely:
-Network Connectivity
-Overloaded host device
-Blocking

To correct problems with network connectivity, review your network settings.
To resolve problems to do with overloading, try closing some applicaitons to relieve the host machine
To resolve problems of blocking, make sure the application is not barred through the software firewall
It has been identified that the application jams after one pattern sequence query for value
to get around that problem, close the window responsible for pattern interaction
and open it again from the main center window, by clicking the colouredblocks

Figure 34 Relevant information displayed after the help option is selected

Clicking the text "Web" leads the user to the web version of the application. This feature requires connection to the internet.


MoSHADi

1-matete/moshadi/login.php

Mo·S·H·A·Di

Shape-Shifting Human Addressible Digital Money

Welcome to MoSHADi, the Shape-Shifting Human Addressible Digital Money System.
This is the central website, a point where registration can take place
and transactions carried out over the web platform. Among those included are:

- Exchanges
- Value Transfer
- Encoding values for retail transacions

Alias: C2
Value: 2.00000
Alias: MIA LELA
Value: 45.8000
Alias: Havoc
Value: 6.74995

Figure 35 Web Application login screen after invoking the Web feature on the Java free form application

Wrapped behind the "transact" button, is a feature that allows value transfer with other people as shown here:

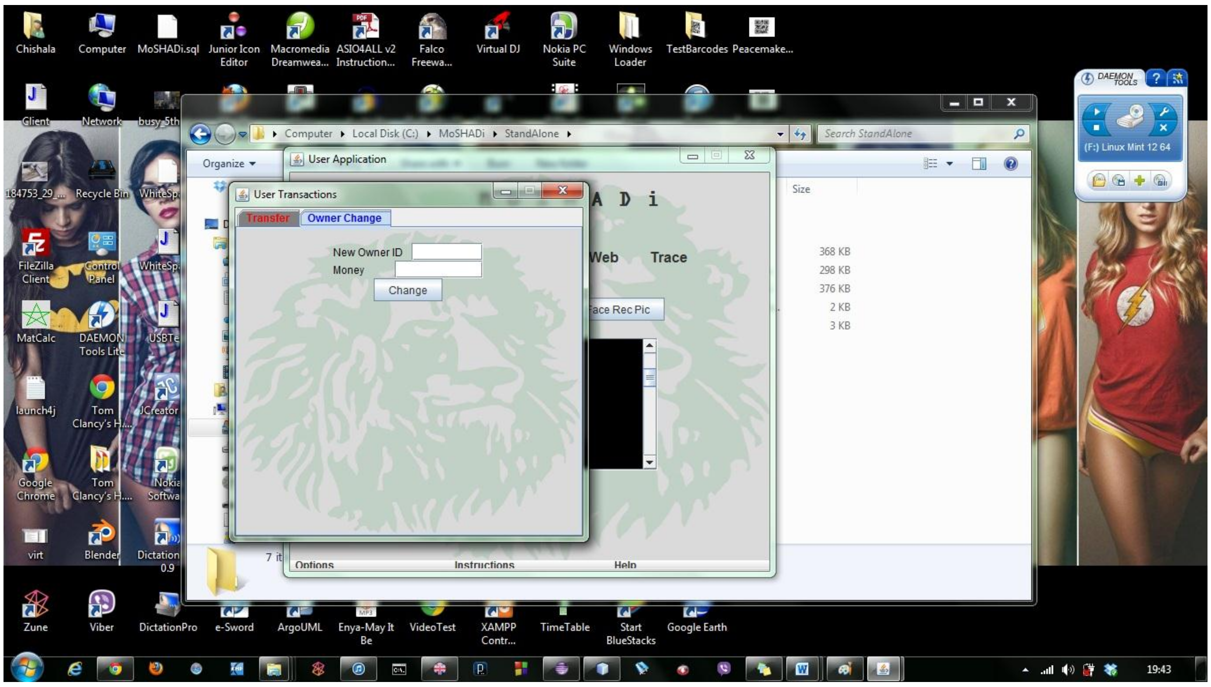
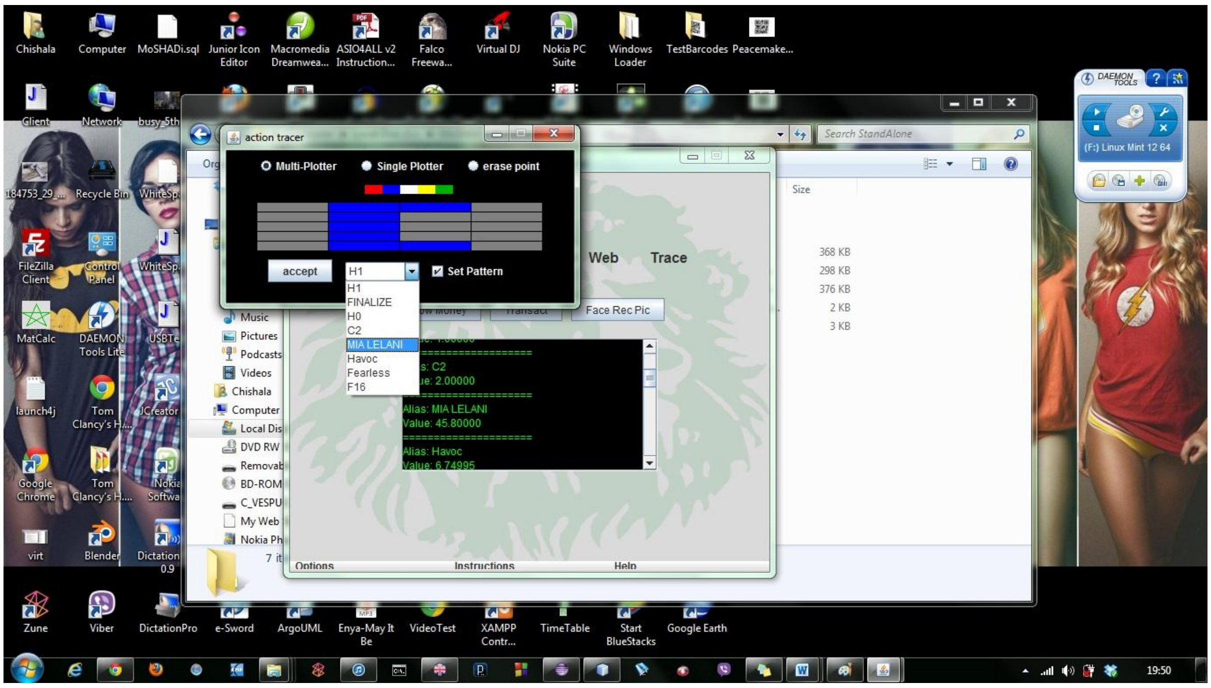


Figure 36 Changing value owner through the textual representation, the alias

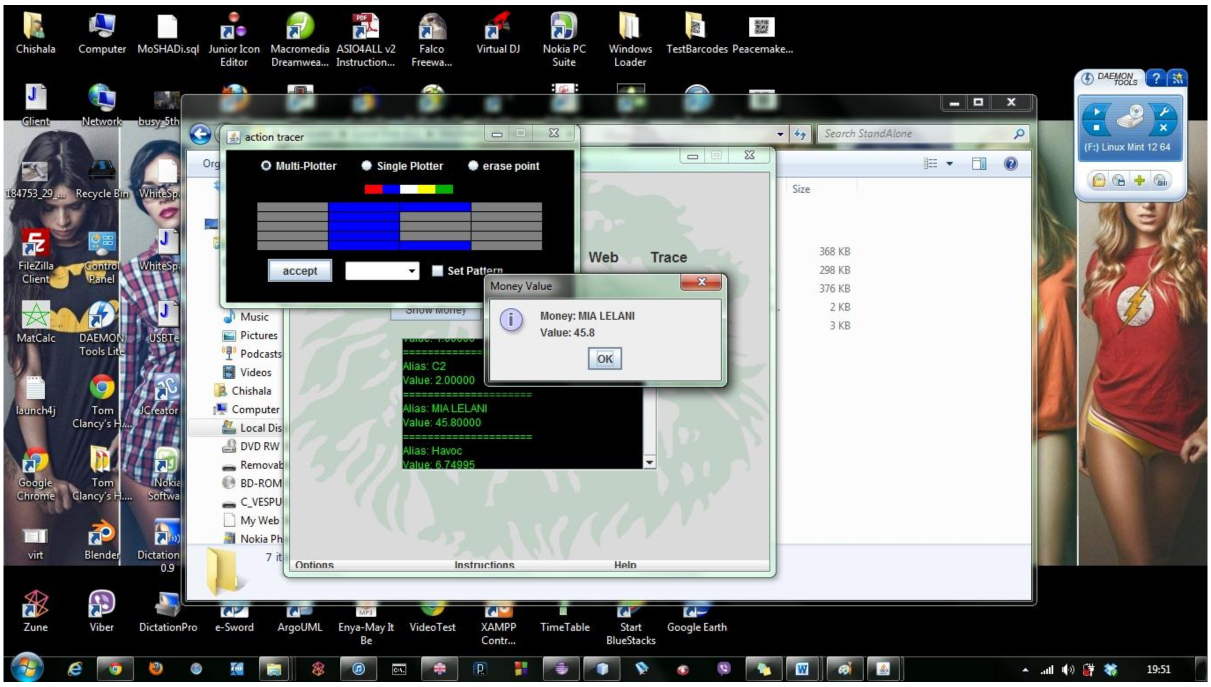
The icon consisting of coloured blocks represents pattern based interaction with value through the mouse on a grid. Below is an image showing how a pattern like the letter C in blue is set to represent a value 0f 45.8 and is also aliased as "Mia Lelani".



```
==================
s: C2
ue: 2.00000
==================
Alias: MIA LELANI
Value: 45.80000
==================
Alias: Havoc
Value: 6.74995
```

Figure 37 Using patterns made on the grid to associate money value with

Patterns can also be used to query. For example, the same C pattern shown above can be used to get the alias and value associated. Making out the pattern and pressing accept returns the results of the value and alias represented by the pattern:



Money: MIA LELANI
Value: 45.8

```
=====================
Alias: C2
Value: 2.00000
=====================
Alias: MIA LELANI
Value: 45.80000
=====================
Alias: Havoc
Value: 6.74995
```

Figure 38 Result of the value aligned with the pattern on the grid

returned together with the alias of the value

The web application has extra functionality that couldn't be wrapped behind the standalone application such as the Face value feature where faces in pictures are used as a representation for a given money value;

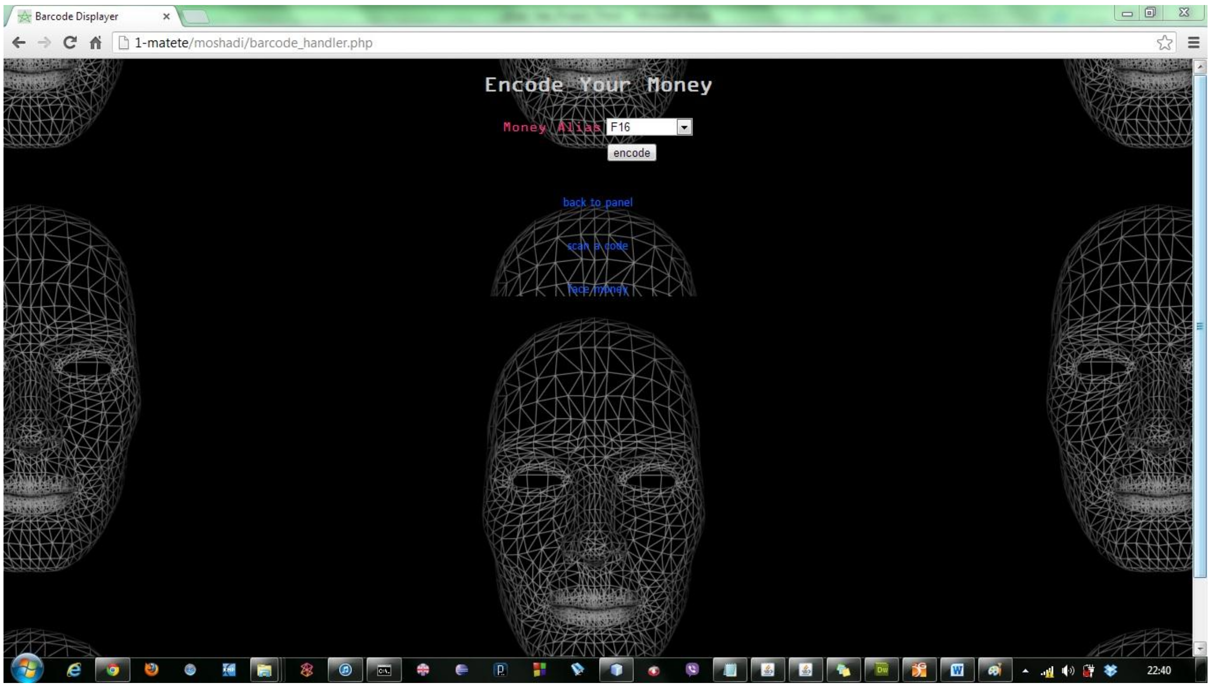


Figure 39 Web Application page responsible for manipulating faces of people and Quick Response codes to align with money

Also associated functionality on the encoding is the ability to encode money as Quick Response Codes or QR-Codes. Here the QR-Code equivalent for the money aliased as "Mia Lelani" is shown

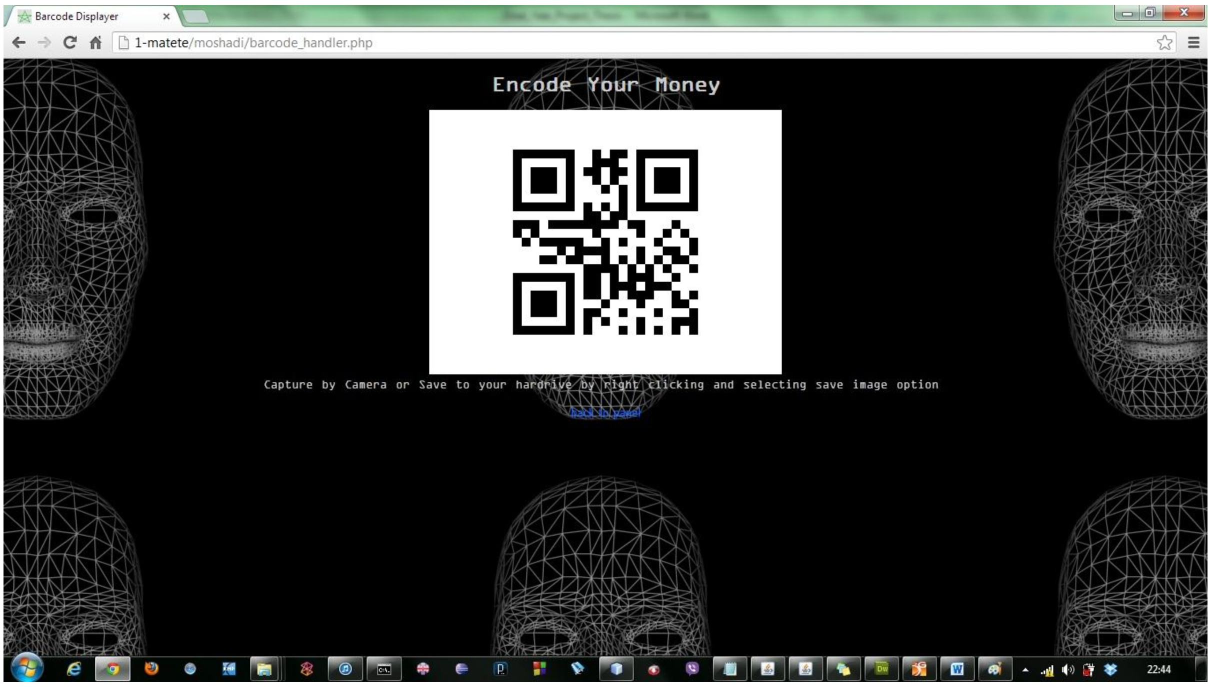

Barcode Displayer
1-matete/moshadi/barcode_handler.php
Encode Your Money
Capture by Camera or Save to your hardrive by right clicking and selecting save image option
22:44

Figure 40 QR Code generated for money aliased as 'Mia Lelani'

A user can query the system with a form of this QR Code saved on his or her device or can take a snapshot using a camera and present it over a QR barcode reader like:

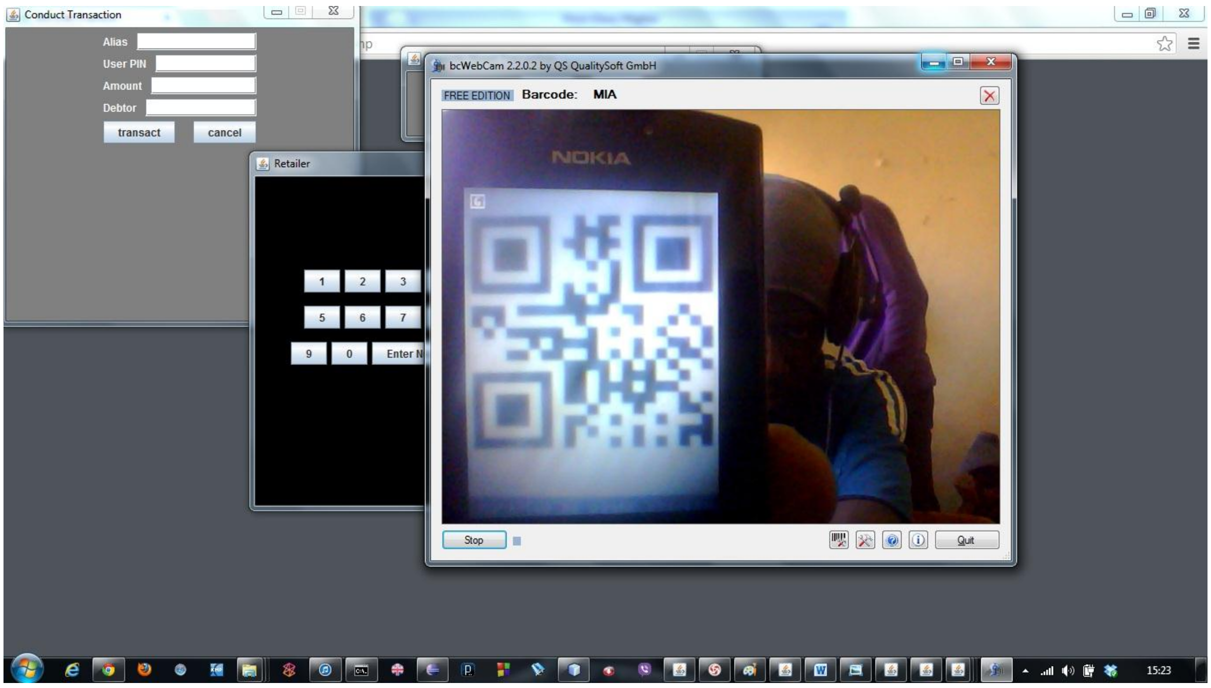


Figure 41 Picture taken by phone of QR Code generated and presented to a scanner for a simulated retailer terminal application and recognized

Here the a picture of the QR type barcode generated was taken and presented to a web cam based barcode reader and it recognized the alias associated with it. Once the alias is understood, a variety of transactions can be conducted with it. Under the faces functionality, a user can select pictures under the C:/MoSHADi/Pictures directory (placed there by copy paste or loaded through the picture taking feature) and align value with them. To show this, here is a screen shot:

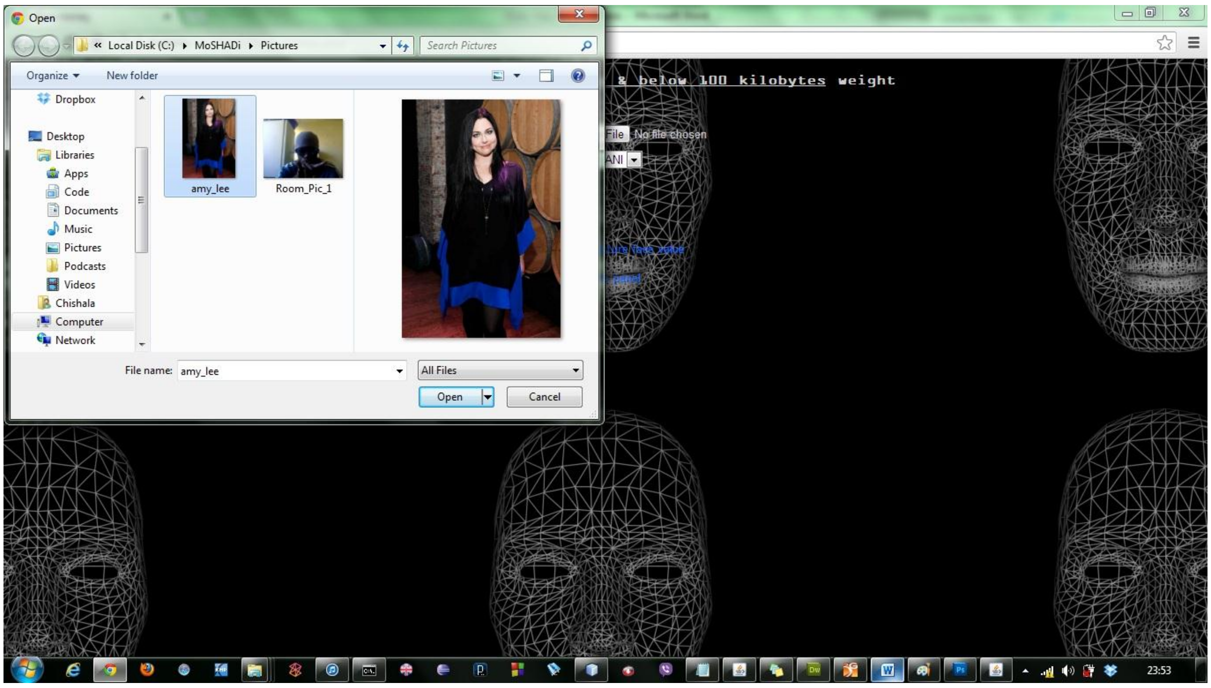


Figure 42 Aligning a face in a picture to a value of money

Here an image of Amy Lee, the lead for the rock band Evanescence placed under the directory, is selected and aligned to a money value by selecting an alias from a drop down menu. To get a value, the user queries the system with the image containing a face, like:

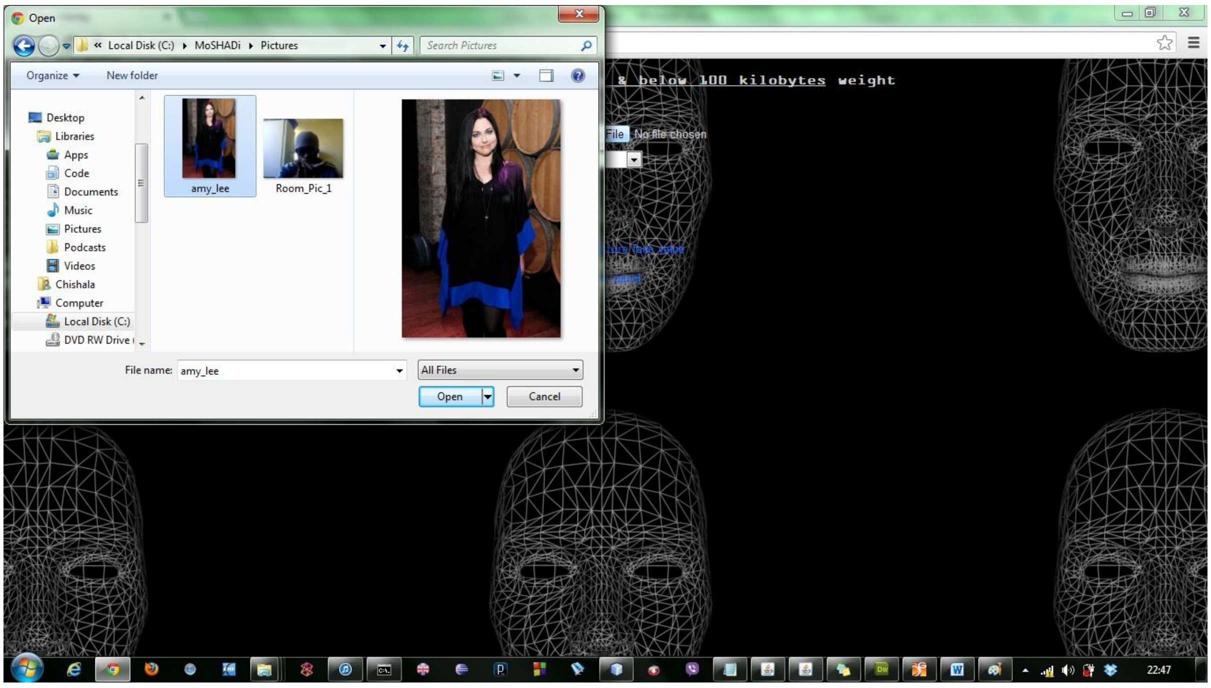


Figure 43 Querying the system with the face to get a value

The result on this example is a value of 100.00 as shown:

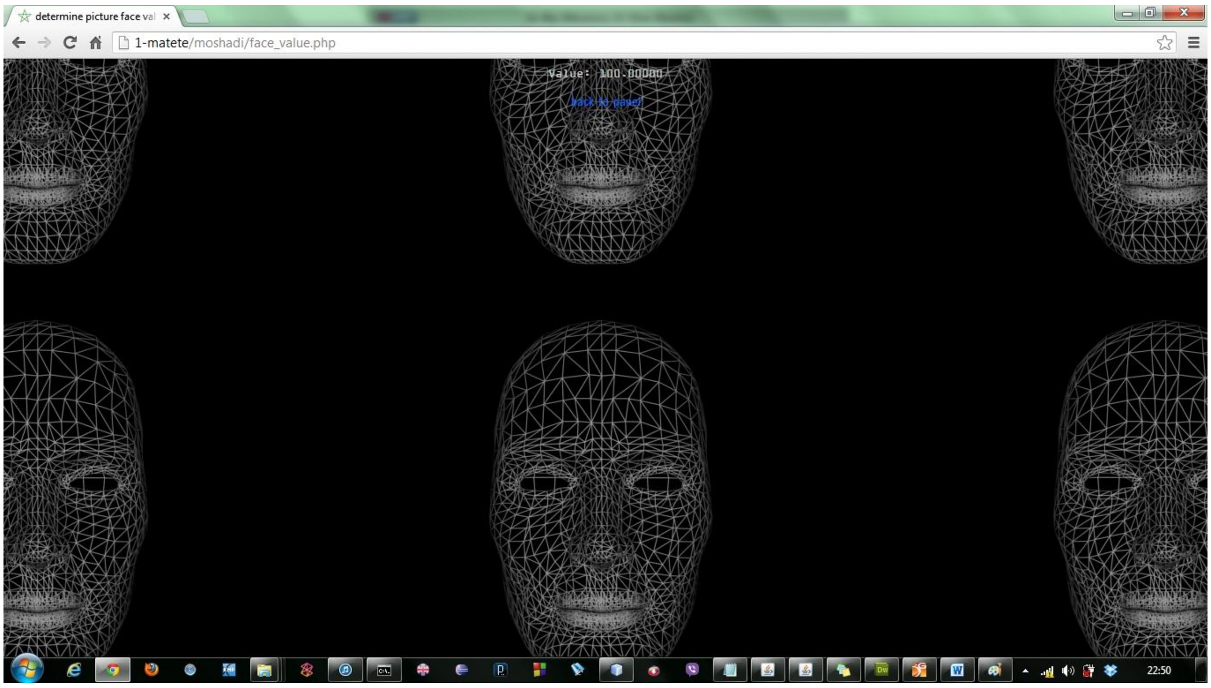


Figure 44 Value returned as 100 units

The static face identification feature can also be used for paying as demonstrated by the following images, where a web based retail terminal form is displayed:

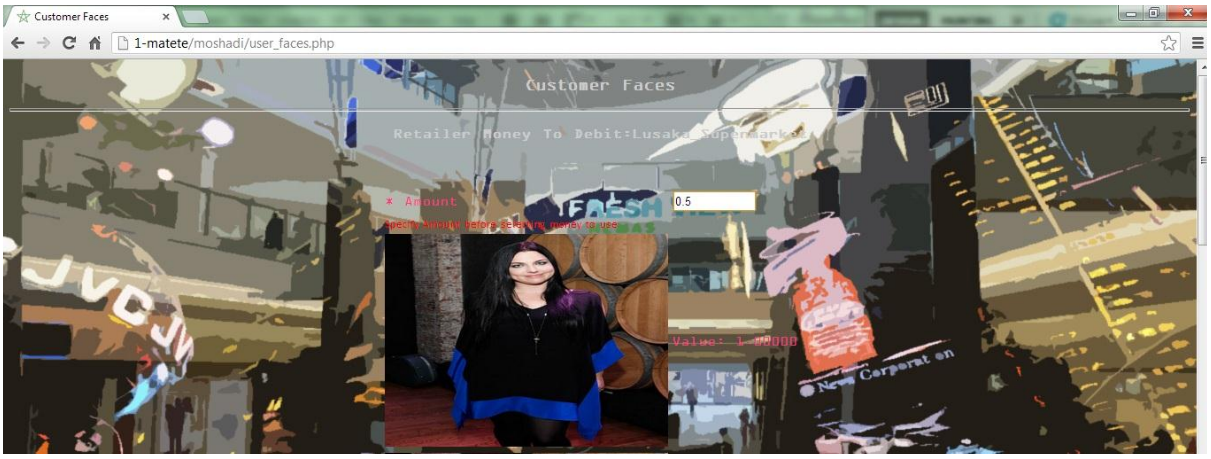

Customer Faces
1-matete/moshadi/user_faces.php
Customer Faces
Retailer Money To Debit:Lusaka_Supermarket
* Amount
0.5
Specify Amount before selecting money to use
Value: 1.00000

Figure 45 Using the face in web based terminal application to pay amount from associtaed value into retailer's money value

Then when a picture containing the face aligned is selected, the transaction is conducted against the money value it represents deducting the specified amount.

An ATM developed to simulate how ATMs could be adapted to the new form of money is described below:

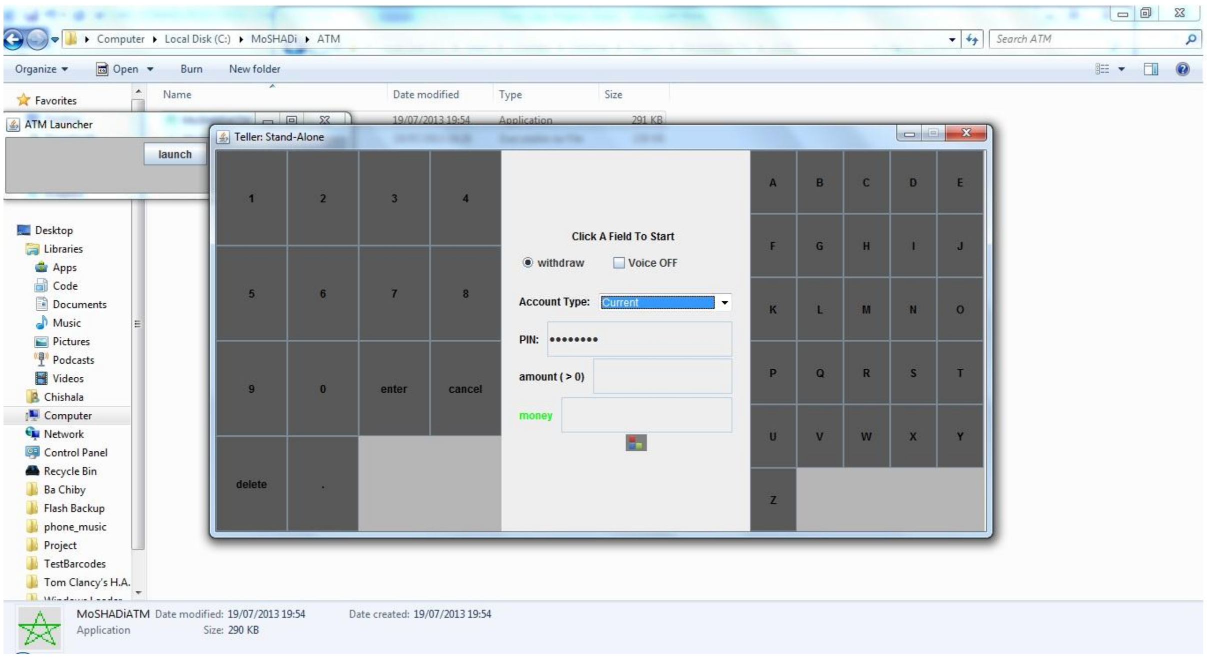


Figure 46 Withdraw Operation for an adapted simulator ATM

Each button on the left and right flank is controlled by the mouse. This is a design characteristic to allow, in future, for the code to be adapted to touch screen interfaces by modifying event handling. Pressing buttons with the appropriate field selected leads to the character that button represents to be fed into the field. This is shown below:

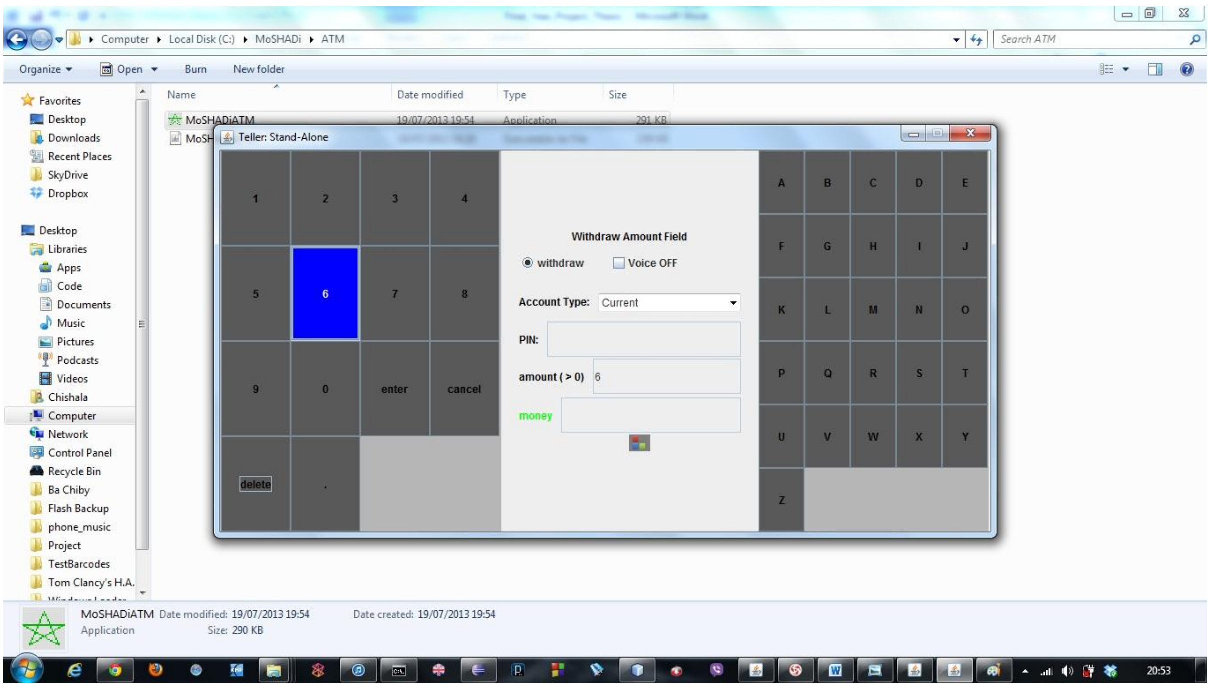
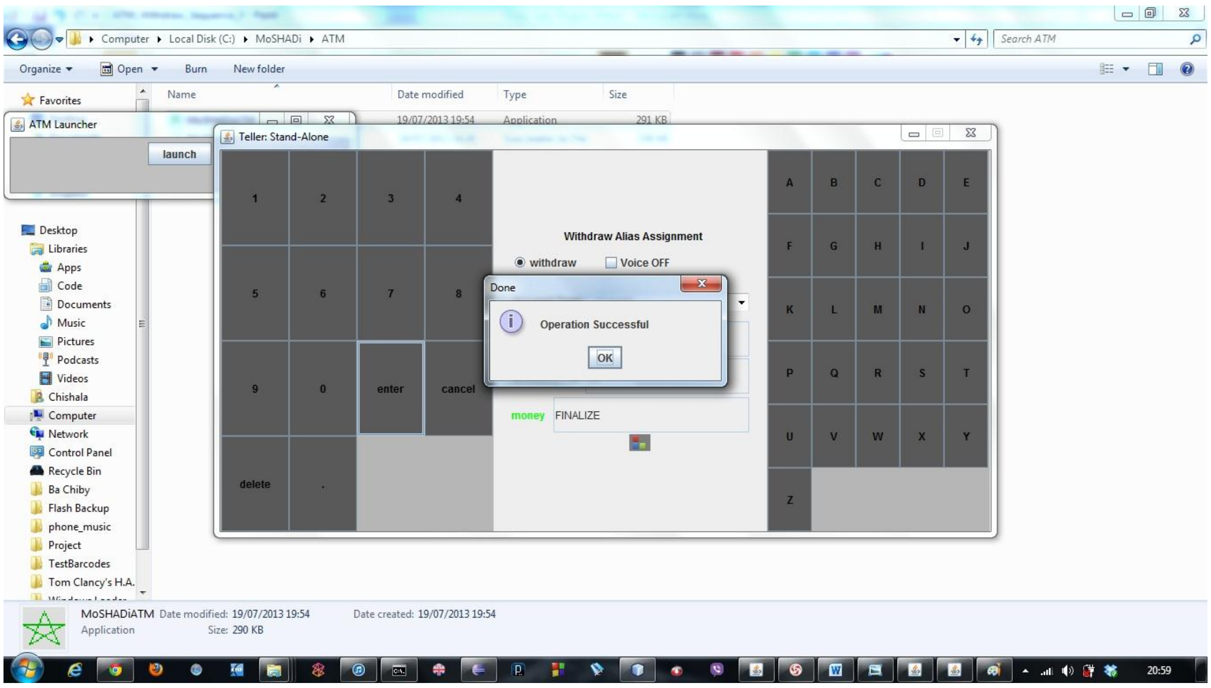


Figure 47 Specifying amount, money alias to represent the withdrawn value and other details for the withdrawal

Function buttons are the enter button, which sends the details of the transaction to generate a new number aliased by the name specified in the field ‘money’ with the green label and assuming the value specified deducting from an existing value. Once this value representation (alias) is created at withdrawal, it can be used anywhere until its value goes down to zero.



Withdraw Alias Assignment

withdraw   Voice OFF

Done

Operation Successful

OK

money FINALIZE

MoSHADiATM Date modified: 19/07/2013 19:54   Date created: 19/07/2013 19:54
Application   Size: 290 KB

Figure 48 Transaction successful, the specified money alias representing value withdrawn is now visible across MoSHADi

The notification "Operation Successful" show that the withdrawal has been conducted and the alias can be used in the system. It may also be represented with other forms such as patterns and faces and QR type barcodes.

The delete a character inserted into a field, clicking the delete button removes the character from the field of focus. Cancel removes all details and defaults the application.

The little icon with the blocks below the fields indicates pattern based control just like the standalone client application.

3.4 Summary

Standalone and web applications were designed and created to work around data relating to money value with the necessary functionality needed to represent information about the values and means of interaction to achieve multi varied form of monetary representation wrapped into them.

Chapter 4

System Testing

Tests were conducted to see how well the developed system would handle some situations identified as part of meeting the non-functional requirements listed. The tests were done sequentially starting with tests for the ability of the system to handle a variety of request and service operations reflecting responsiveness of the system, followed by usability and portability testing and the graphs showing the data were generated from Netbeans Performance Calibration tool.

Since the application is network based, the tests were conducted via a public wireless network between the test server computer and the test client machine.

The specifications of the machines used in testing:

- Client machine with dual core Intel Celeron P4600 processor @ 2.0 GigaHertz and 2 GB RAM and 120 GB free secondary memory.
- Dedicated mini test server machine with dual core Intel Pentium R processor @ 2.9 GigaHertz and 2 GB RAM and 150 GB free secondary memory.
- 15 and 18 Inch Wide Screen LCD Monitors for interface usability and visual interface tests.
- High Definition stereo sound speakers and headsets
- Standard PC Microphone

The conditions of the tests were:

- Broadband Wireless Network with speed at least 54 Megabits per second speed, lightly loaded.
- ~ 65% Free Primary Memory on Client Test Machine, lightly loaded system
- ~ 10% Free primary memory on Server Test Machine, free server system

Client Standalone Application Testing

For the client applications, performance testing was conducted in hierarchical fashion from the top most class to the inner classes providing functionality.

client.ClientAccess.java

Performance testing of the client system was done in terms of CPU utilisation, Memory Usage and Active Threads, using the Netbeans Profiler.

Under ClientAccess class, the profile calibrator was set at the Messenger .run() method responsible for validating user credentials,SpeakerThread.run() method responsible for providing voice control and welcoming message and finally ClientAccess.determineAction() method responsible for determining whether to open the main client application window if proven to be valid credentials or to produce an error message with a counter showing the number of chances left until the application closes detecting invalid credentials.

The results obtained are shown here:

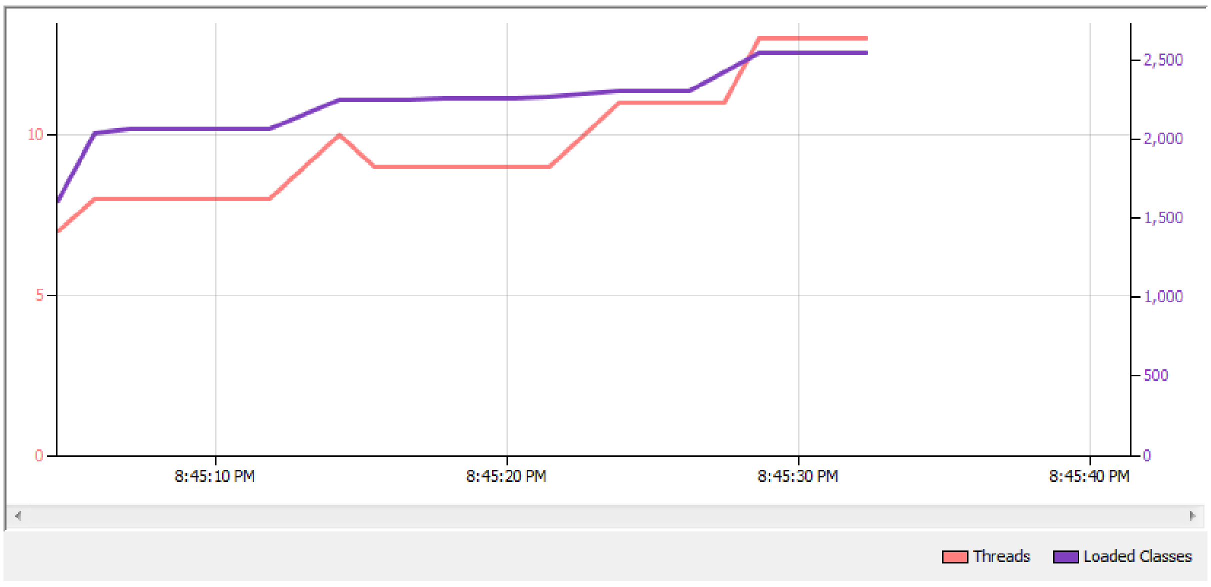


Figure 49 Loaded Classes and Threads for client access test

The graph above shows the number of Threads used against the number of classes loaded over a time frame spread across 10 second intervals. A maximum of 13 threads were used with 2,555 classes loaded. This was during the point where a client feeds in their credentials, the application sends them to the server for validation, reads the response and opens the main application window.

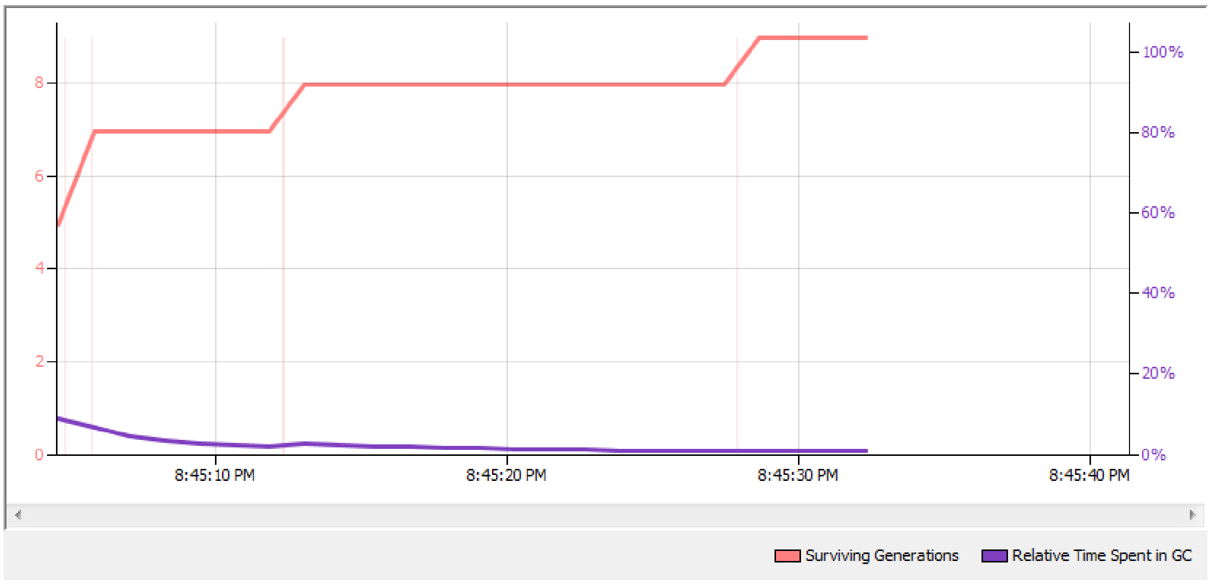


Figure 50 Garbage Collection stats for testing client access performance tests

This graph shows the Memory usage with respect to Garbage Collection, GC. It determined that a maximum of 9.1% relative time was spent in Garbage Collection and an overall average of 1.5% of time and there was approximately a maximum of 9 surviving generations.

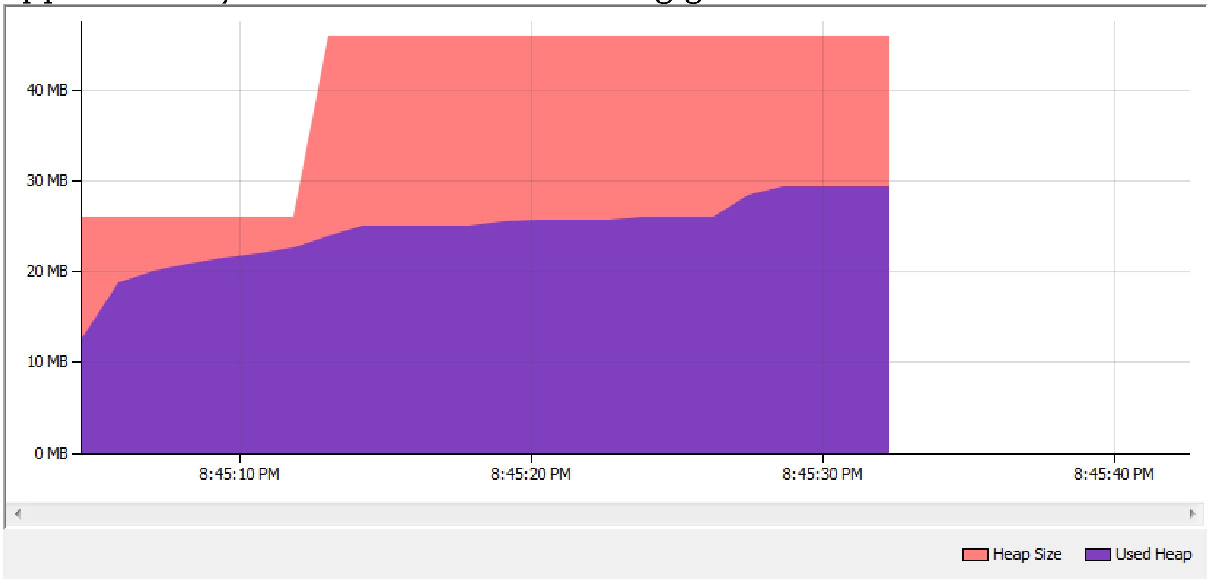
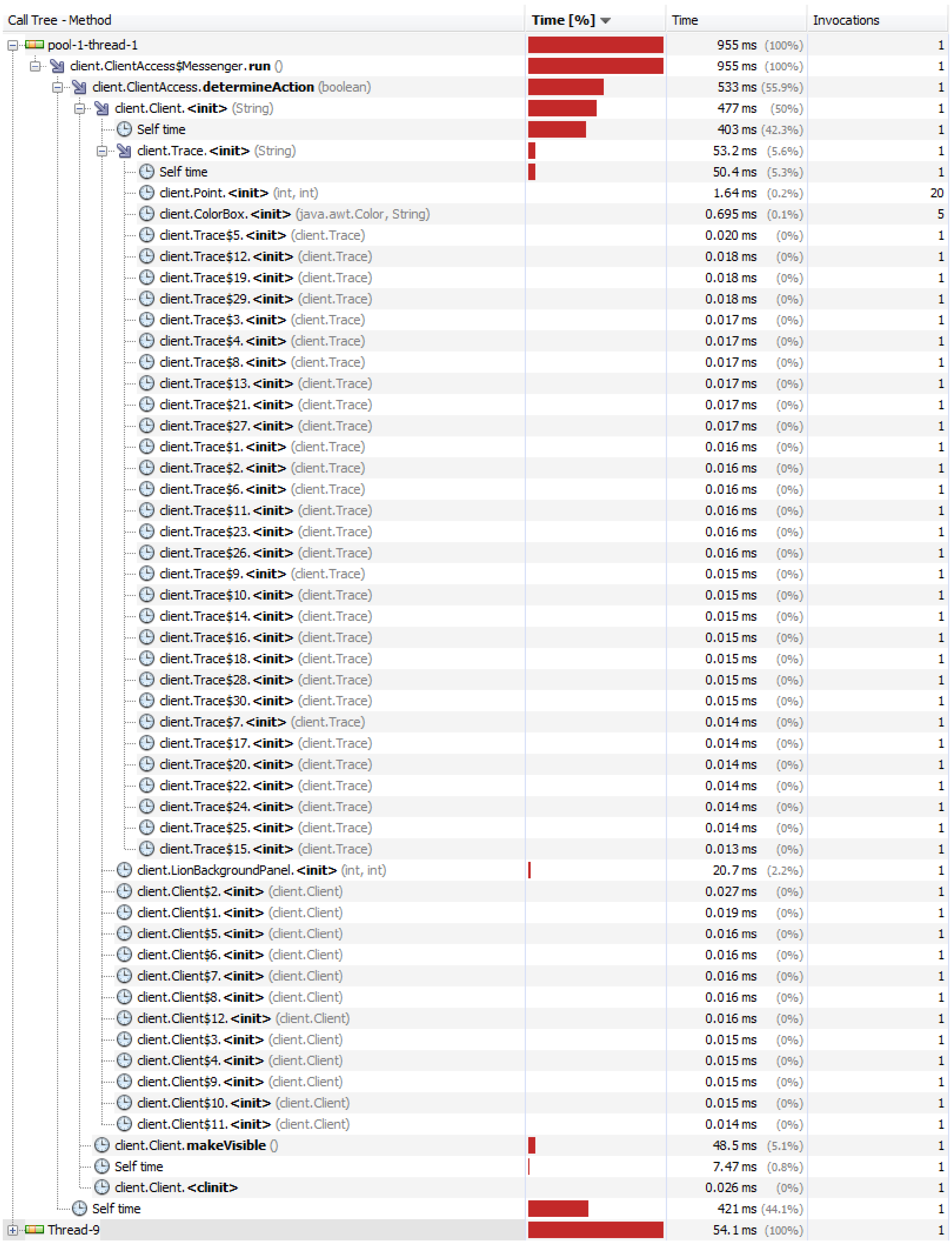


Figure 51 Memory Usage stats graph for client access performance tests

The graph above shows the heap usage of this part of the application. A maximum of 31 megabytes of the 48 megabyte heap were used to process the credentials and open up the main application window.

CPU timings were summarised and returned as here:



Call Tree - Method	Time [%]	Time	Invocations
pool-1-thread-1		955 ms (100%)	1
client.ClientAccess$Messenger.**run** ()		955 ms (100%)	1
client.ClientAccess.**determineAction** (boolean)		533 ms (55.9%)	1
client.Client. **<init>** (String)		477 ms (50%)	1
Self time		403 ms (42.3%)	1
client.Trace. **<init>** (String)		53.2 ms (5.6%)	1
Self time		50.4 ms (5.3%)	1
client.Point. **<init>** (int, int)		1.64 ms (0.2%)	20
client.ColorBox. **<init>** (java.awt.Color, String)		0.695 ms (0.1%)	5
client.Trace$5. **<init>** (client.Trace)		0.020 ms (0%)	1
client.Trace$12. **<init>** (client.Trace)		0.018 ms (0%)	1
client.Trace$19. **<init>** (client.Trace)		0.018 ms (0%)	1
client.Trace$29. **<init>** (client.Trace)		0.018 ms (0%)	1
client.Trace$3. **<init>** (client.Trace)		0.017 ms (0%)	1
client.Trace$4. **<init>** (client.Trace)		0.017 ms (0%)	1
client.Trace$8. **<init>** (client.Trace)		0.017 ms (0%)	1
client.Trace$13. **<init>** (client.Trace)		0.017 ms (0%)	1
client.Trace$21. **<init>** (client.Trace)		0.017 ms (0%)	1
client.Trace$27. **<init>** (client.Trace)		0.017 ms (0%)	1
client.Trace$1. **<init>** (client.Trace)		0.016 ms (0%)	1
client.Trace$2. **<init>** (client.Trace)		0.016 ms (0%)	1
client.Trace$6. **<init>** (client.Trace)		0.016 ms (0%)	1
client.Trace$11. **<init>** (client.Trace)		0.016 ms (0%)	1
client.Trace$23. **<init>** (client.Trace)		0.016 ms (0%)	1
client.Trace$26. **<init>** (client.Trace)		0.016 ms (0%)	1
client.Trace$9. **<init>** (client.Trace)		0.015 ms (0%)	1
client.Trace$10. **<init>** (client.Trace)		0.015 ms (0%)	1
client.Trace$14. **<init>** (client.Trace)		0.015 ms (0%)	1
client.Trace$16. **<init>** (client.Trace)		0.015 ms (0%)	1
client.Trace$18. **<init>** (client.Trace)		0.015 ms (0%)	1
client.Trace$28. **<init>** (client.Trace)		0.015 ms (0%)	1
client.Trace$30. **<init>** (client.Trace)		0.015 ms (0%)	1
client.Trace$7. **<init>** (client.Trace)		0.014 ms (0%)	1
client.Trace$17. **<init>** (client.Trace)		0.014 ms (0%)	1
client.Trace$20. **<init>** (client.Trace)		0.014 ms (0%)	1
client.Trace$22. **<init>** (client.Trace)		0.014 ms (0%)	1
client.Trace$24. **<init>** (client.Trace)		0.014 ms (0%)	1
client.Trace$25. **<init>** (client.Trace)		0.014 ms (0%)	1
client.Trace$15. **<init>** (client.Trace)		0.013 ms (0%)	1
client.LionBackgroundPanel. **<init>** (int, int)		20.7 ms (2.2%)	1
client.Client$2. **<init>** (client.Client)		0.027 ms (0%)	1
client.Client$1. **<init>** (client.Client)		0.019 ms (0%)	1
client.Client$5. **<init>** (client.Client)		0.016 ms (0%)	1
client.Client$6. **<init>** (client.Client)		0.016 ms (0%)	1
client.Client$7. **<init>** (client.Client)		0.016 ms (0%)	1
client.Client$8. **<init>** (client.Client)		0.016 ms (0%)	1
client.Client$12. **<init>** (client.Client)		0.016 ms (0%)	1
client.Client$3. **<init>** (client.Client)		0.015 ms (0%)	1
client.Client$4. **<init>** (client.Client)		0.015 ms (0%)	1
client.Client$9. **<init>** (client.Client)		0.015 ms (0%)	1
client.Client$10. **<init>** (client.Client)		0.015 ms (0%)	1
client.Client$11. **<init>** (client.Client)		0.014 ms (0%)	1
client.Client.**makeVisible** ()		48.5 ms (5.1%)	1
Self time		7.47 ms (0.8%)	1
client.Client. **<clinit>**		0.026 ms (0%)	1
Self time		421 ms (44.1%)	1
Thread-9		54.1 ms (100%)	1

Figure 52 Timing data for the complete test

This shows that the majority of the time was spent processing the instructions and data in the method determineAction( ), to be precise 55.9% or 553 of 995 milliseconds of time.

client.Client:

Under the Client class which was primarily responsible for the main standalone user application, the profiler was set at vital points of the application which were mainly methods and classes (also inner classes) responsible for transactions and value lookups.

The first to test was the lookup for all values under the method Client.getMoney(). The results in terms of memory usage are shown below:

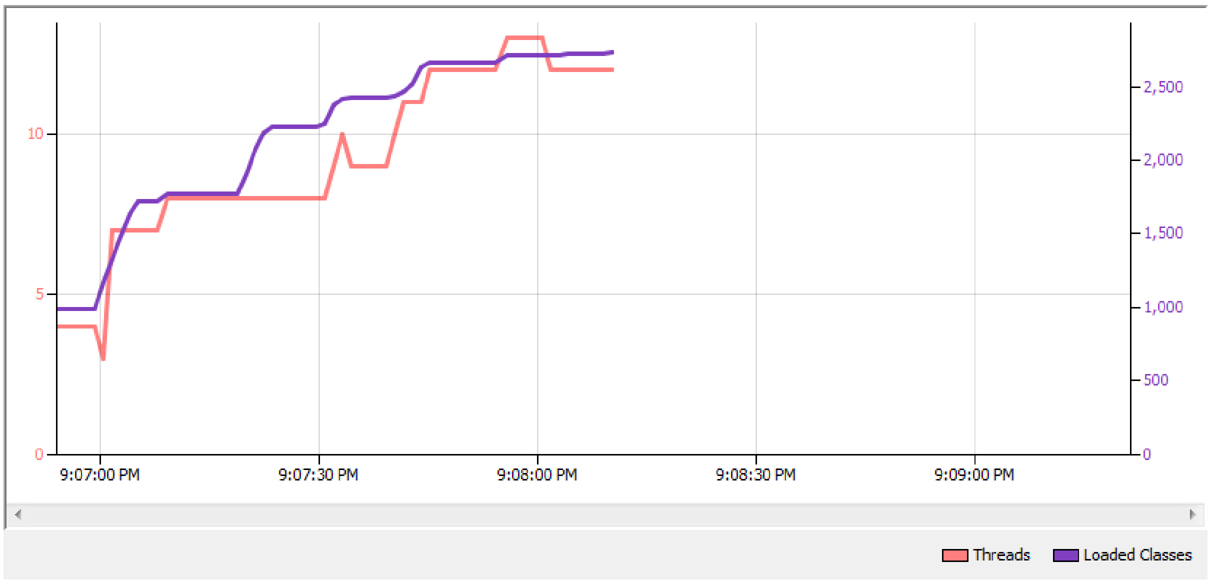


Figure 53 Threads and Classes stats for all users in getMoney method performance test

The graph shows the threads involved in a user lookup for all money belonging to a user. A maximum of 13 threads were used with 2,739 classes loaded by the java system, mostly system classes.

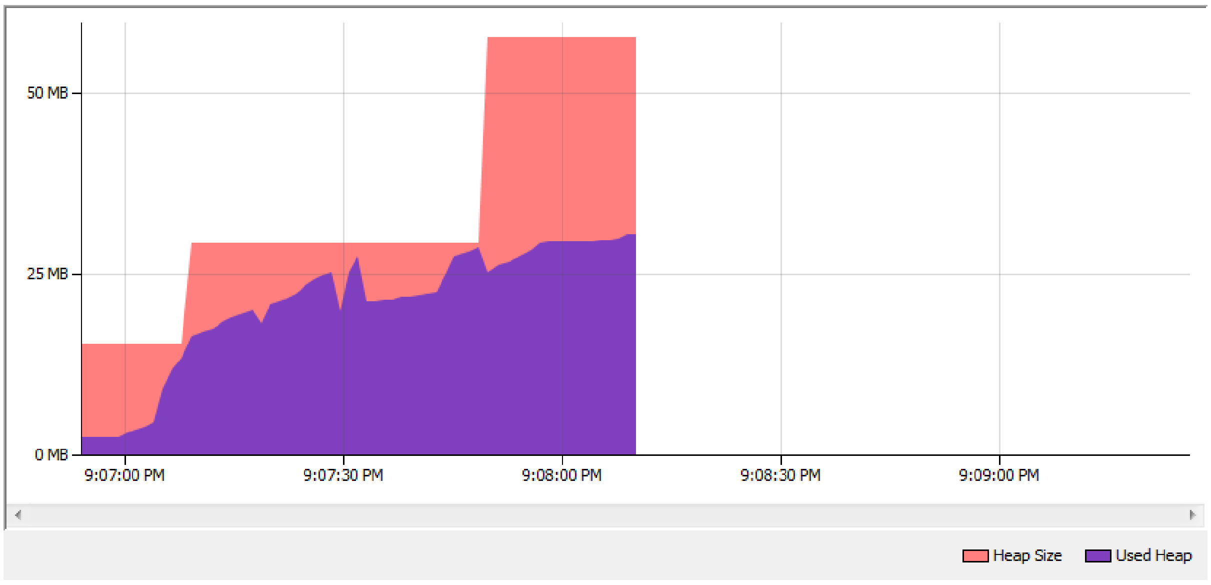


Figure 54 Memory usage stats for getMoney method performance test

The graph above shows the heap memory usage of the application during the lookup. Out of a total of 60.6 megabytes, a maximum of 31.1 megabytes were used, representing 51.3%.

The Application spent a maximum of 1% relative time in Memory Garbage Collection with at most 10 surviving generations. This is illustrated with the graph below:

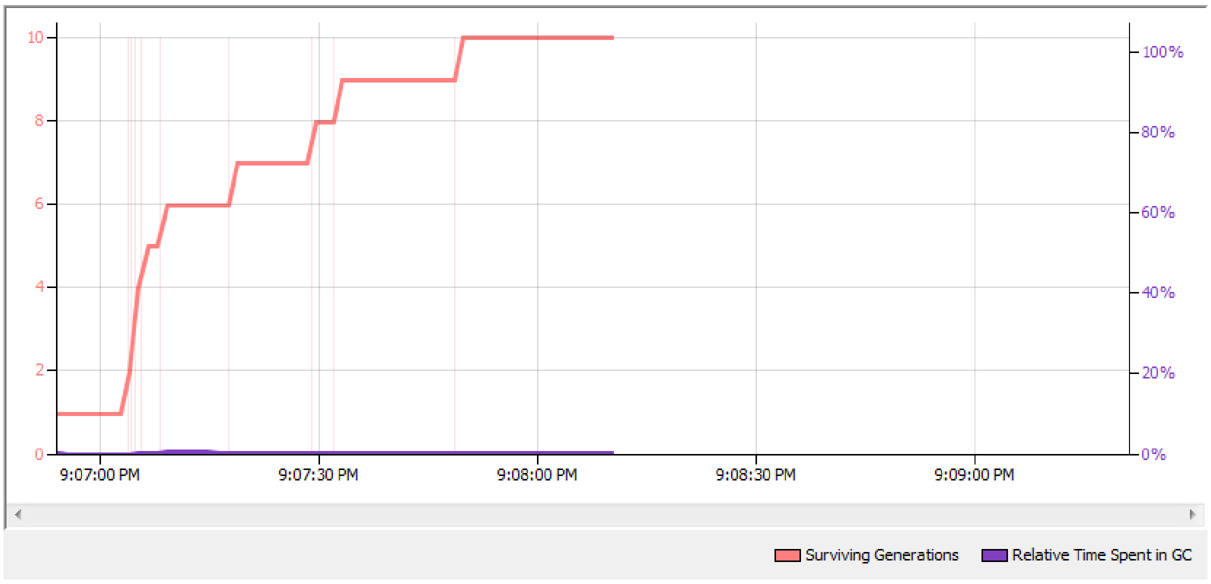
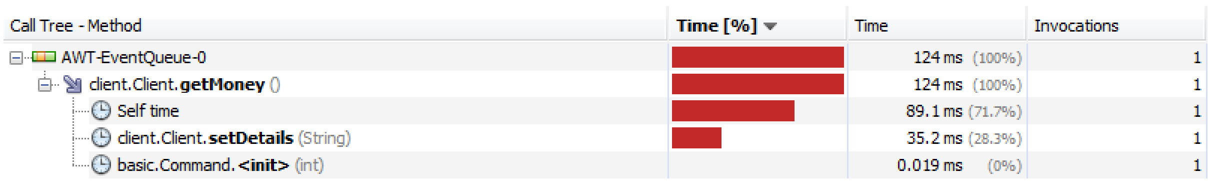


Figure 55 garbage collection stats for the getMoney method performance test

The CPU timings are show below:



Call Tree - Method	Time [%] ▼	Time	Invocations
AWT-EventQueue-0	🟥	124 ms (100%)	1
client.Client.**getMoney** ()	🟥	124 ms (100%)	1
Self time	🟥	89.1 ms (71.7%)	1
client.Client.**setDetails** (String)	🟥	35.2 ms (28.3%)	1
basic.Command.**<init>** (int)		0.019 ms (0%)	1

Figure 56 CPU timings for the test

The method executed within a time of 124 milliseconds and the major lag was attributed to the setting of looked up details onto a scrollable screen which involved a method call on the object responsible for that.

Testing was further conducted on the method responsible for transfer of value to another user. It must be said that the mechanism supporting this method did not require the receiver to be active.

The first graph show the Threads used:

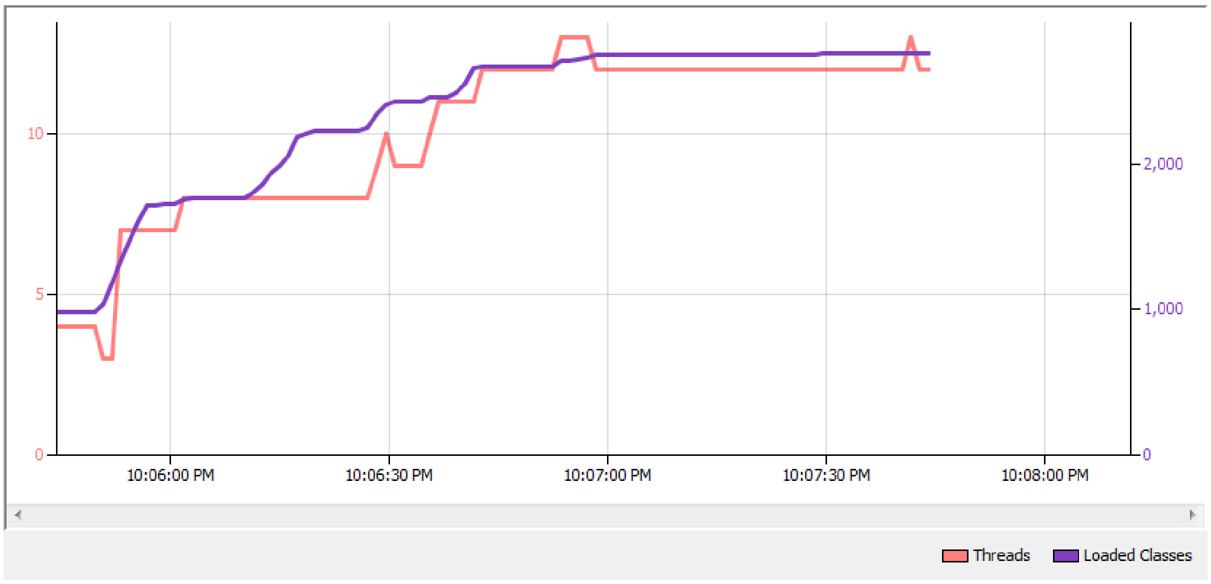


Figure 57 Threads and Classes stats for transfer value test

Remaining consistent, the application required a maximum of 13 Threads and 2,777 loaded classes. This trend of similar maximum thread count across different methods involving threads reflects a property of the Java Thread management system in which existing threads, if idle, can be called upon to handle a task.

The next graph shows the heap memory usage for the transfer method:

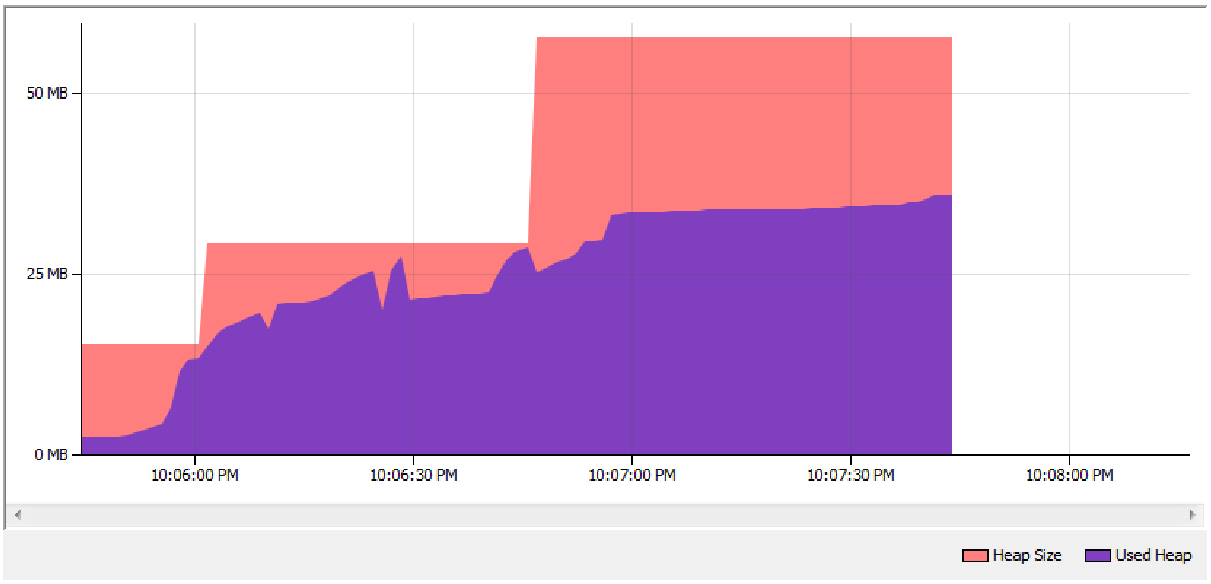


Figure 58 Memory usage stats for transfer value test

The value transfer method heap memory utilisation came in at a maximum of 37.8 megabytes over a total of 60.6 megabytes of allocated heap memory, representing 62.3%. This jump comes in partly because of the fact that verification to determine if the user actually specified transfer details was done within the method on top of socket and stream object creation, manipulation and closure.

The memory garbage collection graph reflects even more of this feature:

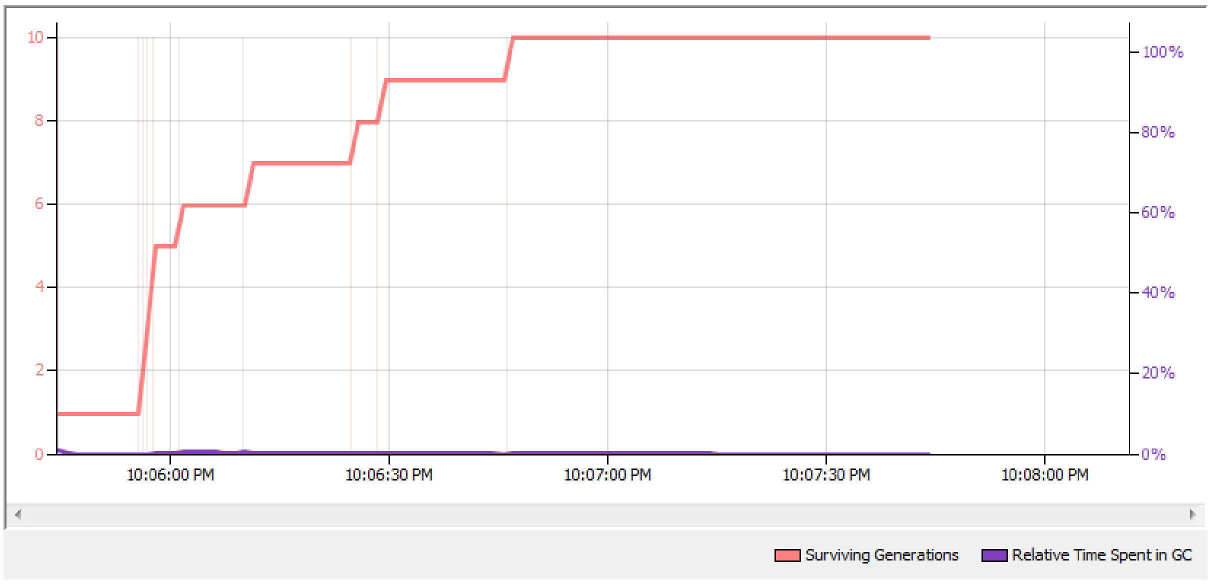
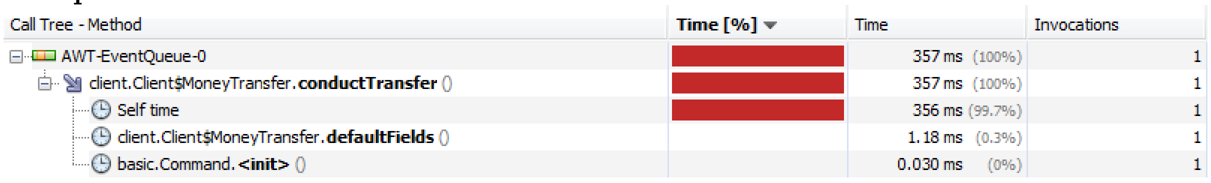


Figure 59 Garbage Collection stats for transfer value test

The application spent a maximum of 1.1% of the relative time in Garbage Collection with at most 10 surviving generations. The total CPU time for this part of the application was 357 milliseconds of which 1.38 milliseconds were spent resetting fields to clear of the details after completion. transaction.



Call Tree - Method	Time [%]	Time	Invocations
AWT-EventQueue-0	🟥	357 ms (100%)	1
client.Client$MoneyTransfer.**conductTransfer** ()	🟥	357 ms (100%)	1
Self time	🟥	356 ms (99.7%)	1
client.Client$MoneyTransfer.**defaultFields** ()		1.18 ms (0.3%)	1
basic.Command.**<init>** ()		0.030 ms (0%)	1

Figure 60 CPU timings for transfer test

Server Tests

Tests were conducted on the server to see how it would cope when multiple client requests were made. To begin with, the server was made to validate multiple login credentials in real time. The login credentials were floated to the server by a multiple client simulator programme in which several threads are initialised from atleast two machines and tasked with the responsibility of sending dummy credentials for the server to validate. The code below shows the class embodying these threads:

```
public class MultipleAccess {


  private int counter = 0;


  public static void main(String[] args){
    MultipleAccess multiple = new MultipleAccess();
    multiple.contact();
  }
  public void contact(){
    ExecutorService executor = Executors.newCachedThreadPool();
    try{
      //server is an example of a URL that resolves to the IP of the server
     Socket contactSocket = new Socket(InetAddress.getByName("Server"),5050);
     int threadCounter = 0;


    while(threadCounter < 100000){
      executor.execute(new Send(contactSocket));//send request
      threadCounter += 1;//increment
    }
    }catch(Exception exc){
      exc.printStackTrace();
    }
    }

    private class Send implements Runnable{

        private Socket socket;

        private ObjectOutputStream outStream;


        public Send(Socket skt){

            try{

             socket = skt;

             outStream = new ObjectOutputStream(socket.getOutputStream());

            }catch(Exception exc){

                exc.printStackTrace();

            }

        }

        @Override

        public void run(){

            try{

                Command command = new Command();//Command class is found in the package named “basic” 
containing utility classes. It specifies the command as an int and data as an Object.

                command.setCommand(Command.VALIDATE_CLIENT_LOGIN);//command here is 
VALIDATE_CLIENT_LOGIN

                String[] credentials = {"X","11223344","PC"};//X is the fake PIN, 11223344 is the fake application 
signature and PC is the fake device

                command.setCommandData(credentials);

                //send to server and close stream

                outStream.writeObject(command);

                outStream.flush();

                outStream.close();

                System.out.println("Contact #:"+counter);//increment thread counter

            }catch(Exception exc){

                exc.printStackTrace();

            }

        }

    }

}
```

The telemetric results for a hundred thousand request – service operations are shown here:

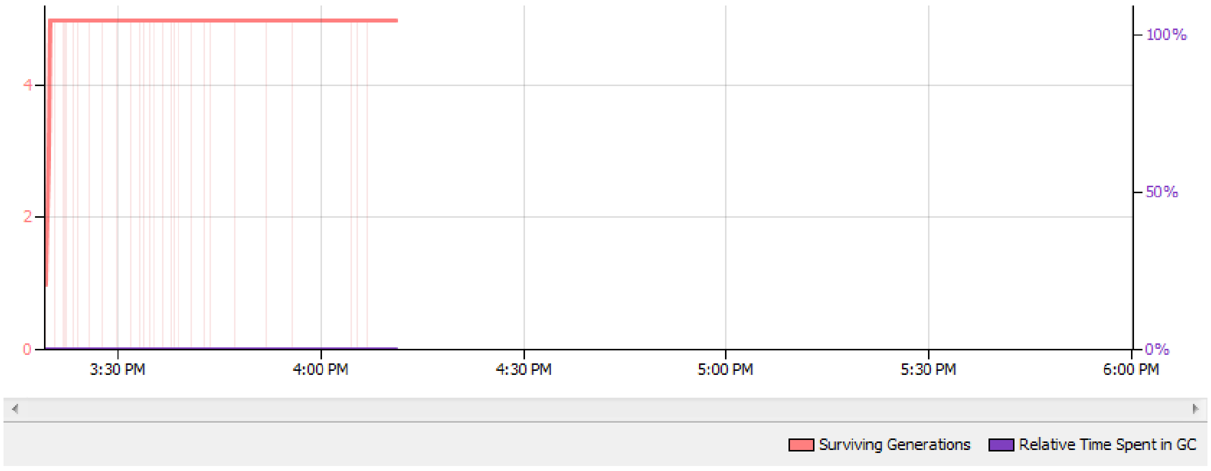


Figure 61 Garbage Collection stats for server test

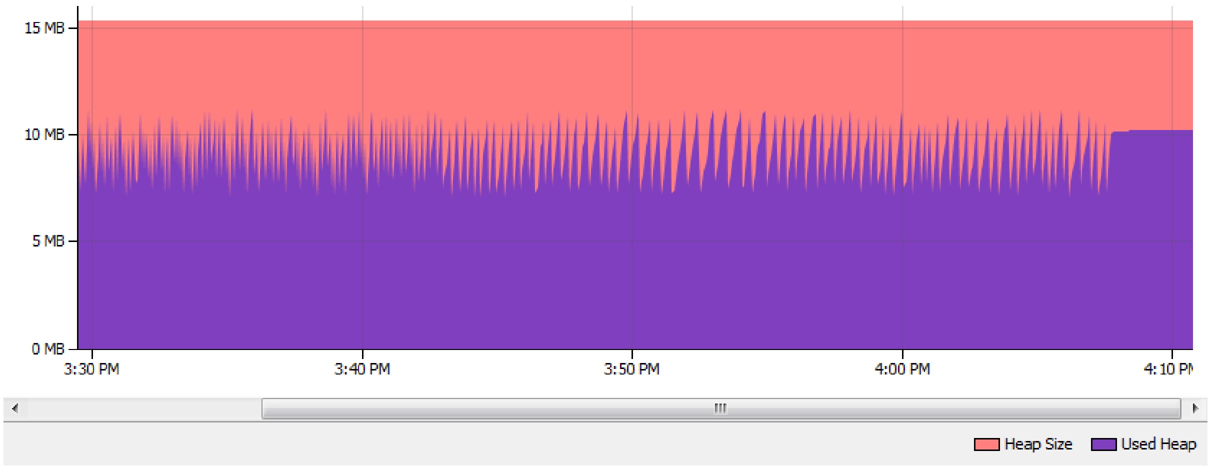


Figure 62 Memory usage stats for server test

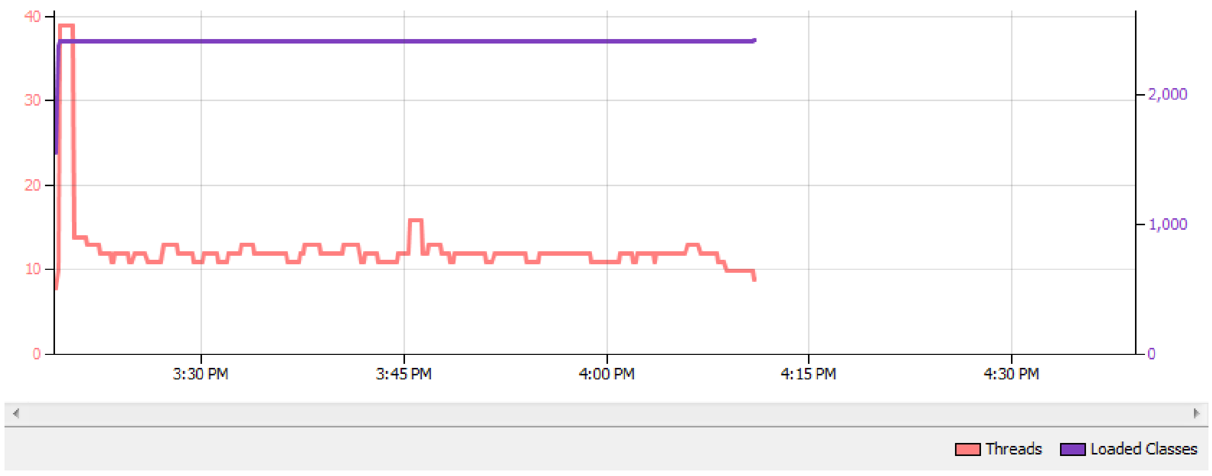
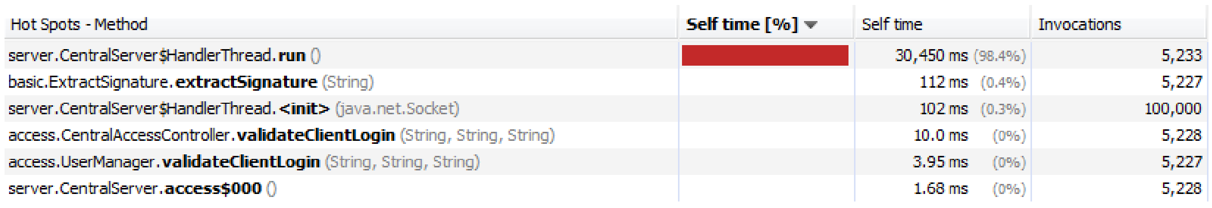


Figure 63 Threads and Loaded classes stats for server test

Despite a hundred thousand invocations, the thread re-use capability of Java allowed existing threads from previous invocations to be used and so instead of a hundred thousand equal handler (workder) threads on the server side, only 5,227 were used, representing a ratio of 1 server thread to ~ 20 client threads.



Hot Spots - Method	Self time [%]	Self time	Invocations
server.CentralServer$HandlerThread.**run** ()		30,450 ms (98.4%)	5,233
basic.ExtractSignature.**extractSignature** (String)		112 ms (0.4%)	5,227
server.CentralServer$HandlerThread.**<init>** (java.net.Socket)		102 ms (0.3%)	100,000
access.CentralAccessController.**validateClientLogin** (String, String, String)		10.0 ms (0%)	5,228
access.UserManager.**validateClientLogin** (String, String, String)		3.95 ms (0%)	5,227
server.CentralServer.**access$000** ()		1.68 ms (0%)	5,228

Figure 64 CPU timings for server test

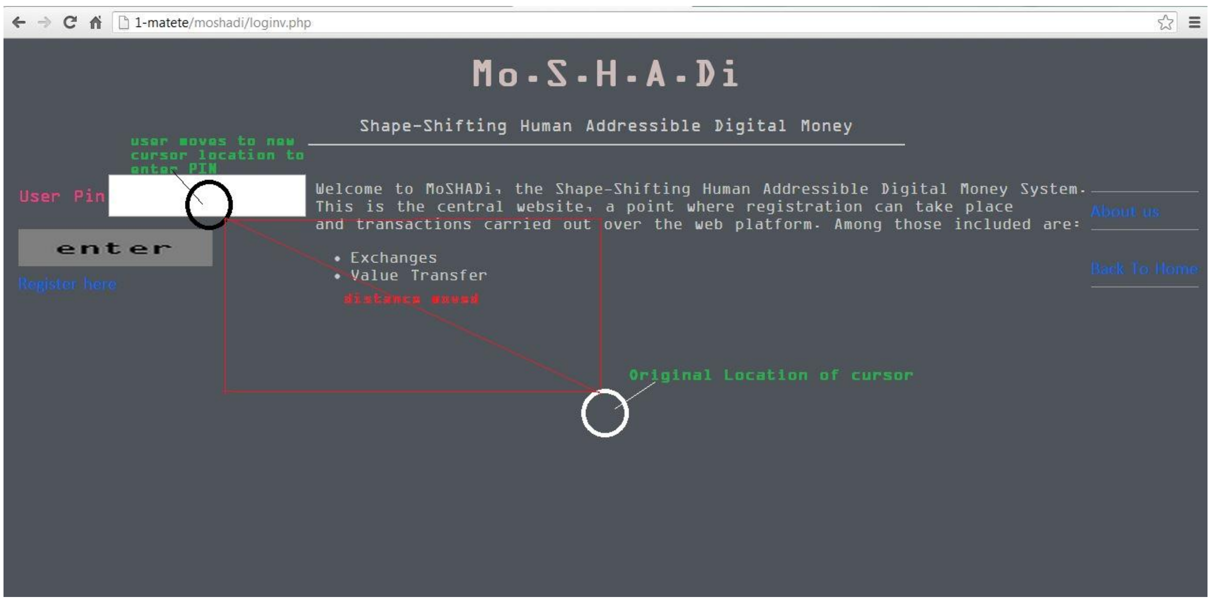
The information above is a summary of the timings.

Usability and Intuitive Tests

Since the system was designed to accommodate a variety of users so that all participants in an economy can take part by using it, a few features were incorporated such as voice guides for the visually impaired, voice control for the handicapped and pattern based control for the less literate.

These were designed to be simple and user friendly and in this section are tested to see how well they meet the requirement for usability. To begin with, the speed of the visual interface using the voice guides was tested. The approach taken was to use the system blind folded with the timings taken for each point of contact engaged on the interface. Metrics involved were the distance between interface components and the times to getting to them, which were then used to compute the average usability speed of the interface.

To graphically summarise:


1-matete/moshadi/loginv.php
Mo.S.H.A.Di
Shape-Shifting Human Addressible Digital Money
user moves to new cursor location to enter PIN
User Pin
enter
Register here
distance moved
Original Location of cursor
About us
Back To Home

Welcome to MoSHADi, the Shape-Shifting Human Addressible Digital Money System. This is the central website, a point where registration can take place and transactions carried out over the web platform. Among those included are:

- Exchanges
- Value Transfer

Figure 65 Distance tests to vital components on web login page

The image above shows how the voice guided web version of the client side application is arranged to minimise the distance a user has to move from the default central location on the browser window to one component, here the private PIN field. Blind fold on a user slightly familiar with the system test revealed that it took approximately 3.5 seconds for a user to move left and upwards from the centre of the screen to the components of interest – the PIN field and enter button. As the user arrives on the component, the voice guide mentions it to assure them they are on the right position on the screen.

Going further and into the main control panel:

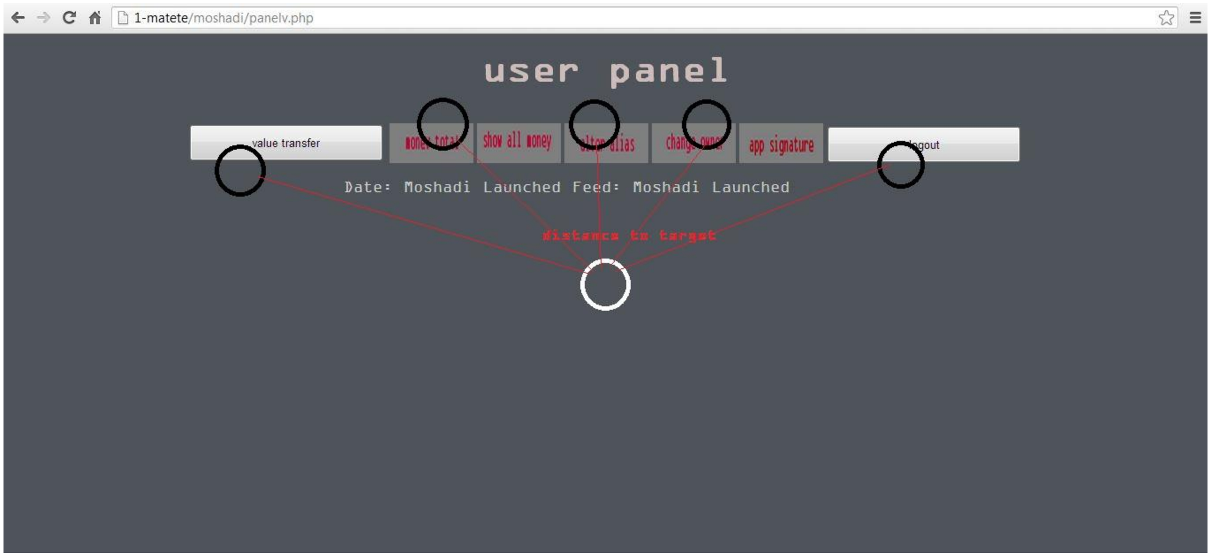


Figure 66 Voice Guided Control Web App Panel: Interface Components enlarged

The components here are arranged so that approximately equal distances can be covered from a central point on the screen for both the voice guided and non-voice guided interfaces. In the case of the voice guided interface, the components are larger to increase the probability of contact with a component but not compromising on the spacing between them to avoid selection of wrong components by the user.

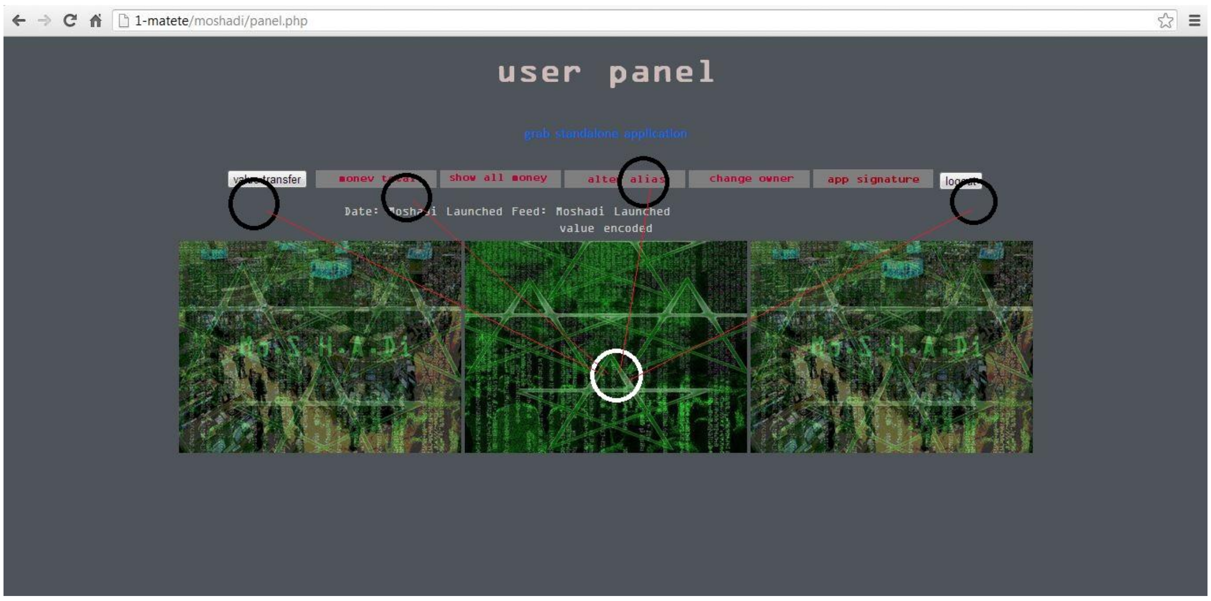
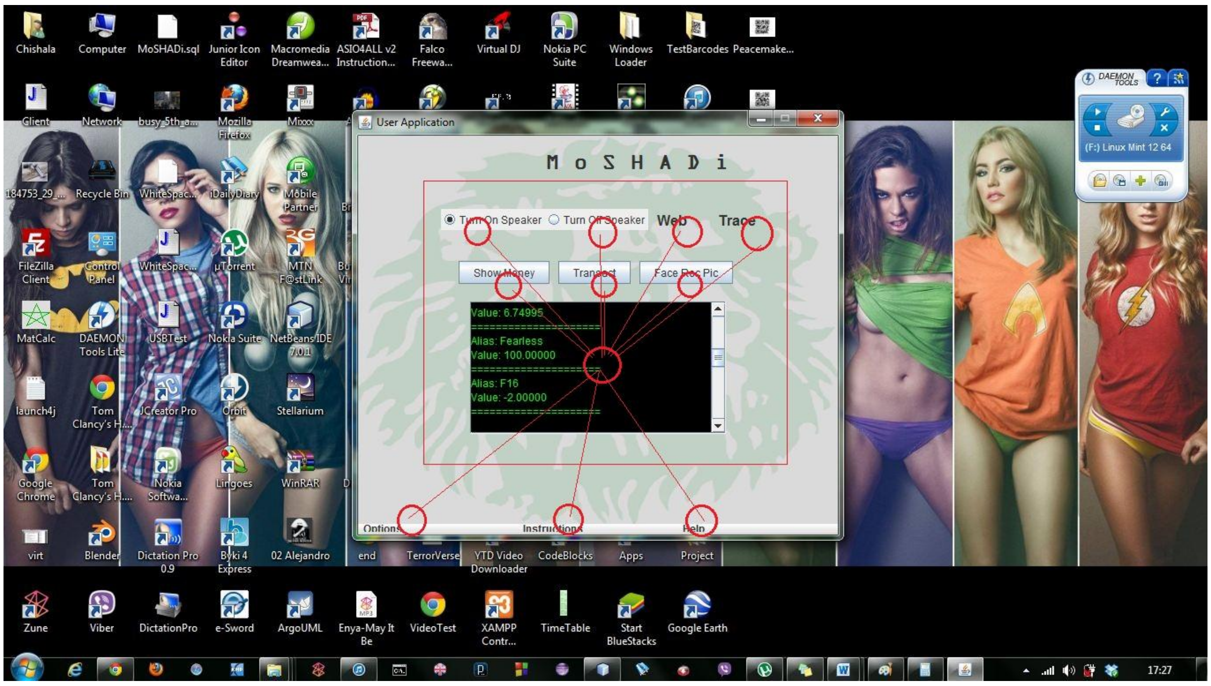


Figure 67 Non-Voice Guided Control Panel: Interface components spread out but quickly within reach

For the control panels tests illustrated that the distance from the centre of the interface to the components of interest for the voice guided interface was on average less than that of the non-voice guided interface. It took on average 5 seconds for a user slightly familiar with the system to get to the components from the centre of the interface in a blind folded test. This was ~1.5 seconds slower than the voice-guided system.

On the standalone client applications, the interface was deliberately narrowed with components restricted to accommodate both voice guided and non-voice guided functionality to the greater benefit of the user. The image below shows the arrangement:



```
Value: 6.74995
=====================
Alias: Fearless
Value: 100.00000
=====================
Alias: F16
Value: -2.00000
=====================
```

Figure 68 Standalone Application Distances to vital components from central area of the application window

The standalone application has a high level of symmetry and gives the user the option to use the voice guide or not, just as the web does. The applications are instructive, with each feature incorporated giving the user knowledge of how to go about using it. This approach coupled with the simple layout made for a more intuitive application interface.

Telemetric tests were performed on the voice guide and revealed the following results:

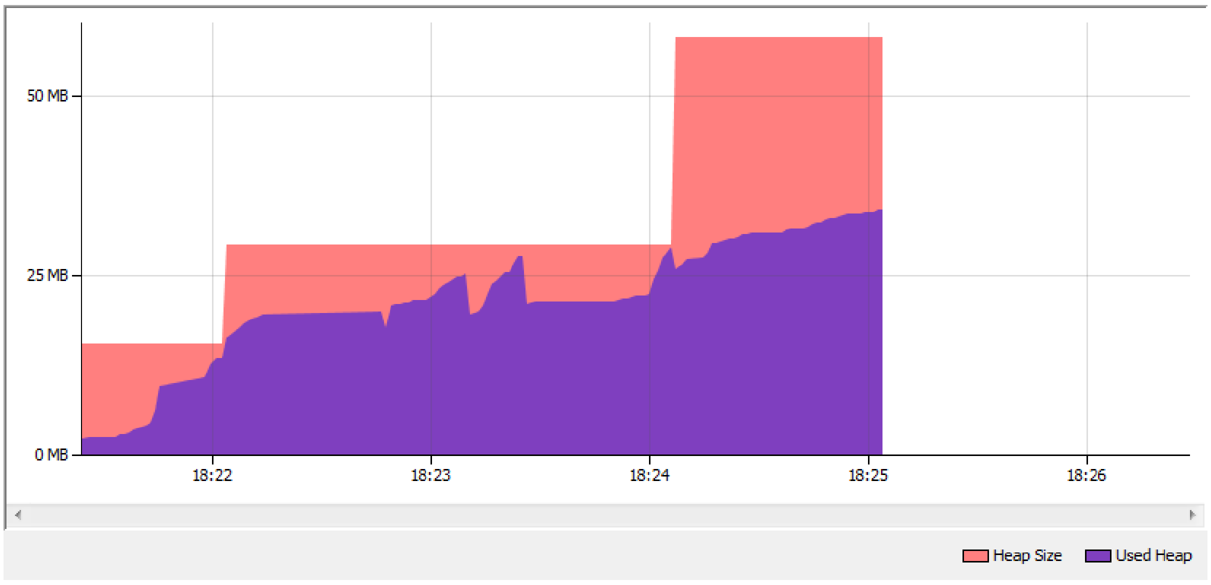


Figure 69 voice guide test memory usage stats graph

The application was assigned a maximum heap size of 61 megabytes and used up a maximum of 2.6 megabytes to do speech synthesis.

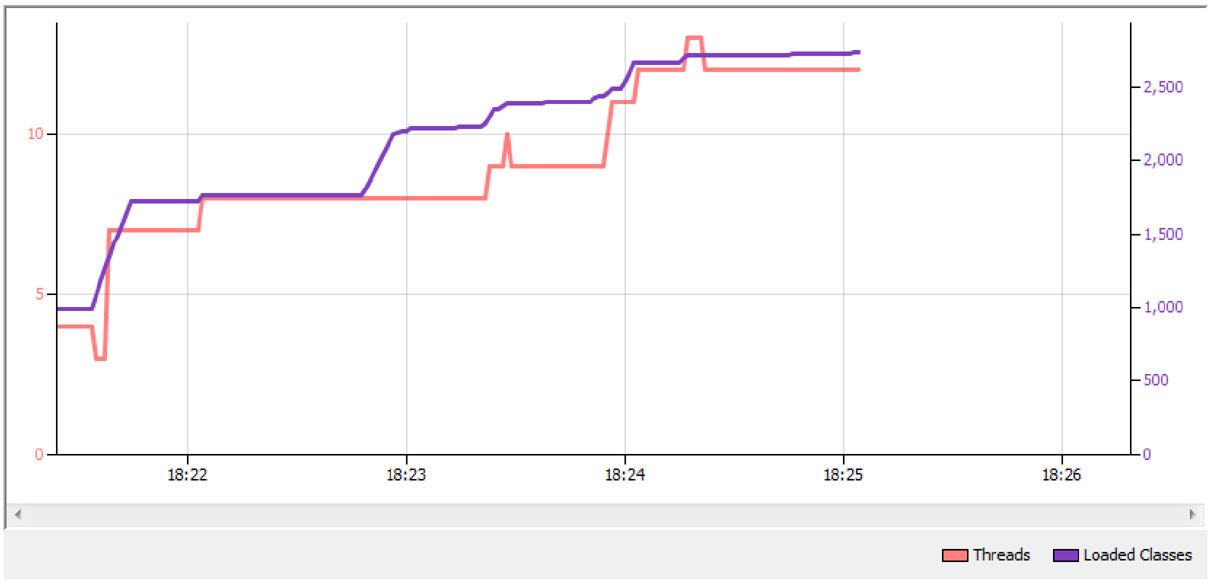
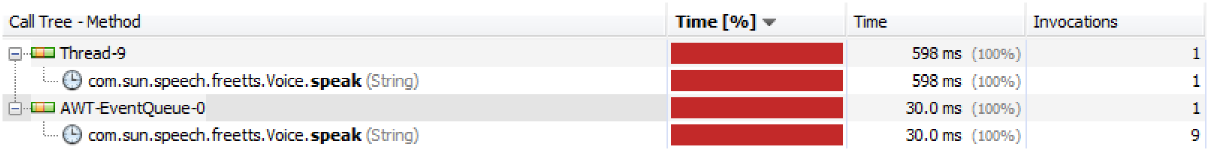


Figure 70 voice guide involved threads and classes stats

In terms of loaded classes and threads used, a maximum of 13 threads was required of which 4 threads were used for speech synthesis operations.



Call Tree - Method	Time [%]	Time	Invocations
Thread-9	🟥	598 ms (100%)	1
com.sun.speech.freetts.Voice.**speak** (String)	🟥	598 ms (100%)	1
AWT-EventQueue-0	🟥	30.0 ms (100%)	1
com.sun.speech.freetts.Voice.**speak** (String)	🟥	30.0 ms (100%)	9

Figure 71 CPU timings for Voice guide test

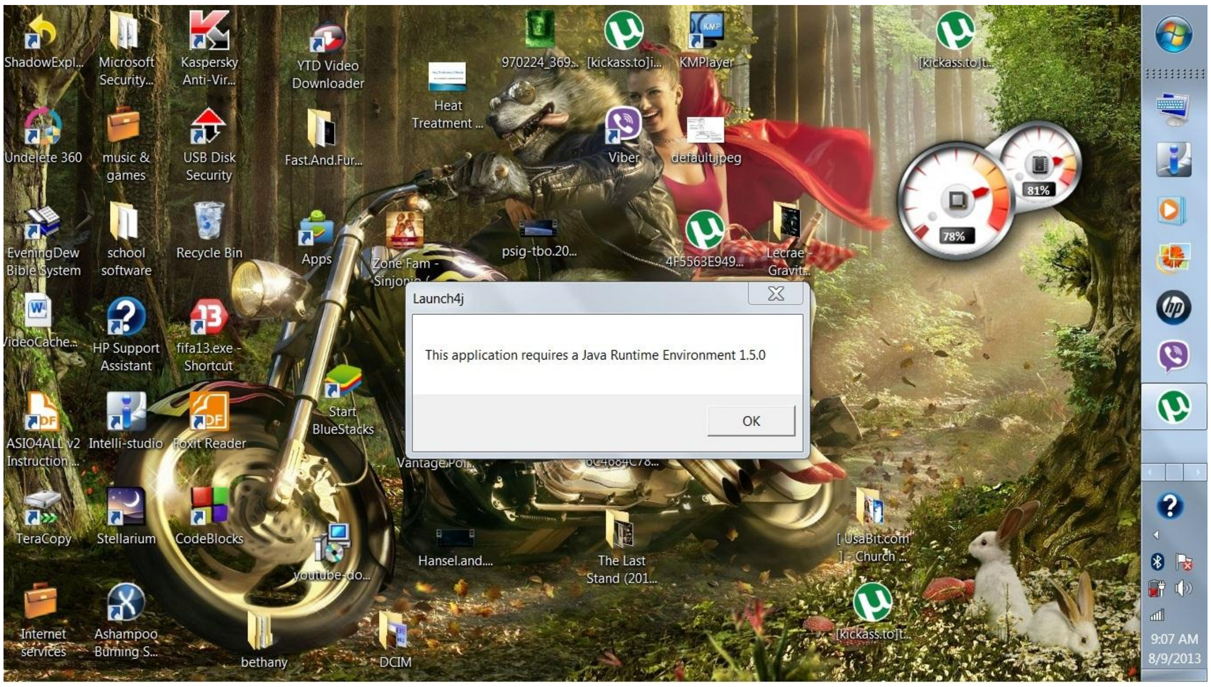
The telemetric summary above showed that 598 milliseconds were required to process textual information and produce audio output for it.

These tests revealed a very efficient voice guidance system relying on text to speech technology.

Portability Tests

Because of the fact that the web application runs in the context of a browser, its portability is guaranteed. Attention was paid to the standalone applications. To be achieve portability to other systems running Windows, java was used. For the portability tests, the applications were bundled up as executables and tried out on two different computers, one without java runtime environment installed and another with java runtime environment installed.

On the system without java, applications prompted the user to download and install the runtime environment by invoking the browser with the destination page set to the download section of the website oracle.com from where the link to the downloadable java runtime environment is available.


Launch4j

This application requires a Java Runtime Environment 1.5.0

Figure 72 Running application of computer without Java Runtime Environment installed

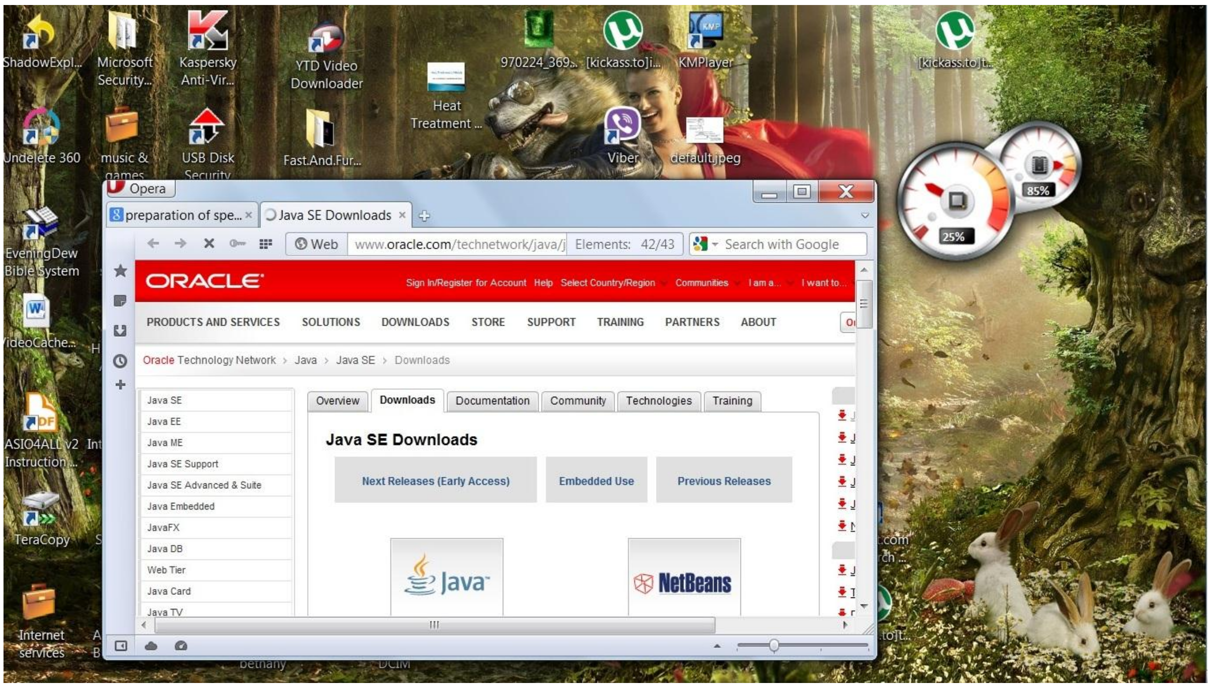


Figure 73 User is redirected to java download page on clicking OK on prompt message

On the other computer with an already installed runtime environment the applications launched successfully and the results for the main user application are shown:

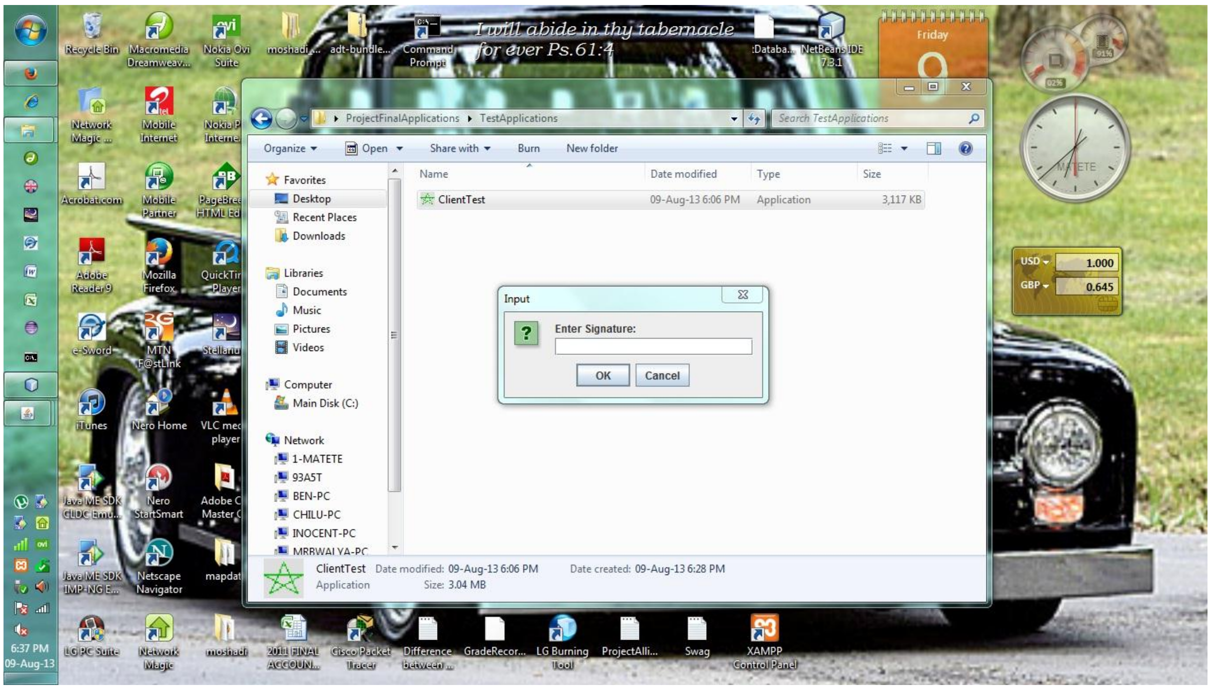


Figure 74 Trying out the application on a desktop with Java installed. It worked and prompted the setting of the signature needed to identify the device during log ins as shown

First the user is prompted to set the signature unique to the application and originally supplied via the web application after download. If entered correctly, the application starts up with a splash screen and then opens up the usual access login window.

The signature, together with system details of the user, operating system and device name are recognised in the MoSHADi database system. This approach helps to provide security in the sense that a device whose details do not match that of the original user's is not allowed to open up the main application, protecting the details of the real user. Even though a marriage exists between the user machine on which the application is originally run and the application, details can be reset via the web interface, with appropriate credentials identifying the user as the real one. This allows the application to be usable on other Windows machines with java runtime environment installed.

Summary

Tests carried out on system components revealed how it utilises computer system resources, operates to provide functionality to users and how the system as a whole met some of the non-functional requirements.

Chapter 5

Discussion and Conclusion

With the technology available, it is not only possible to transfer money value electronically from one place to another or between entities but also to render it in various, more convenient forms and means recognizable by an automated system operating at a level as close to the people as possible but retaining control to bodies of authority. The miniature system developed in this project came to show how this can be realised with conventional technology and techniques allowing the creation of value, using it and moving it between individuals and entities.

If such a system was implemented in communities, it would indirectly encourage investment in digital communications infrastructure, electronics and electronics research, sensitisation programmes educating people about finance and making them more technologically literate, thrust a greater level of control into the hands of financial regulators and have them work closer with communications companies and regulatory bodies, the ability to analyse value usage and spending characteristics of individuals and institutions with greater accuracy would be possible. It would see money value moved swiftly and more securely across the world without bearing the cost of providing physical security in the form of armed escorts. It also would mean the end of manufacturing and recycling of paper and coin money - contributing positively to the environment and the end of money counterfeiting but problems would arise when people try to deceive value identification systems which would need to be delt with. Ultimately, such a system on a wider scale would see the transition from a money system relying on distinct physical paper, metallic and plastic as with bank notes, electronic cards and coins and into a completely digital form shifting version with an interface defined by means of interaction between autonomous systems and people. It would make an environmental friendly form of money suitable for an information age.

Despite the rosy ends of things, such a system would face a number of challenges in the real world as did the miniature system developed. An area of much concern is security, carrying value over the air and on the line makes it susceptible to interception by third parties. Cryptography when incorporated induces performance drag, especially if millions, perhaps hundreds of millions of value transfers and transactions are being handled per unit time. Another element would be the level of trust people have in the system of money. People want to touch their money somehow, sometimes and making it literally intangible presents a new challenge in this respect.

Challenges Faced

One of the biggest challenges when it came to developing the system to embody what has been described and was proposed was determining the kinds of representations the system would identify and recognise, within the confines of the technology available. Initially, it was thought to integrate vision, sound, air gestures and textual representation of money value. In terms of vision, faces of people were planned to be identified dynamically or in real time through analysis of live video feed from cameras and from there the system would gather an array or collection of all money value associated with a person from their face and have them access any of their money value once appropriate credentials such as ID's are supplied. Face detection and subsequent recognition were the technique required to realise this feature, however, the magnitude of the task and the resources available could only allow static identification of faces from pictures for which the system accommodated. To compliment the vision feature, a Quick Response code (QR code) identification system relying on conventional webcams was integrated allowing QR codes masking value to be identified. For static code recognition, the web application was built with a JPEG format QR-code image scanner. With respect to sound, the system was designed to accept voice input and generate internal representations that the system could understand, store and associate with money value. Using the Sphinx library for Java speech API from Carnegie Mellon University and writing code to render representations of voice commands operating on a privately developed dictionary consisting of only a few legal commands a user can make, a voice command driven feature was developed and integrated into some modules of the miniature system. Problems came about when bundling the application as executable Java Archive Files and Windows Executables since the path to the defined dictionary would resolve to a location the application resource finder did not know of. Eventually, with the help of Eclipse IDE's executable wrapper, all externally referenced resources and classes were extracted and bundled together for some components, solving the resource location resolution problem but widescale integration into user applications was still facing the problem of non-responsiveness.

After some research, blob detection was thought of as one way of tracking actions or gestures captured through live video feed and mapping those actions to represent a user's value of money detectable across the system. Development began on a gesture control feature relying on this technique but due to time constraints work only reached as far as blob detection via luminance changes. Here, all values of light above a threshold of 0.2f were picked up and filtration conducted to pick out pixel groupings of interest. However, the extraction couldn't be clean cut and as background light regularly kept dropping in and detected as part of the pixel blob. This presented the problem of failing to track distinct user actions accurately without having to include unwanted visual information from the surroundings. Despite this drawback, an alternative feature in form of pattern setting was developed. This relied on mouse control to allow the user to make patterns to represent their value of money and have it recognised system wide.

Another problem was stability because the system would become unresponsive after a few weeks, approximately every 7 to 8 weeks from October of 2012 til August of 2013. It prompted updates or changes and became a matter of devising new and solid code for the system in order to buy time into the future.

Eventually stable code was written and formed the basis of the final application submitted.

It must be stated that complex components such as face detection on pictures and voice guided control were not 100% accurate. On a scale of 1 to 10, the former had a success rate of 3/10 and the latter 6/10. It was tremendously disappointing and frustrating that despite efforts to improve these statistics, performance penalties and non-responsiveness were the result. Ommission was far from the mind as it would ultimately defeat the goal of proving value can be represented in a host of ways, faces and voice patterns not being exceptions.

Evolution?

For a system whose basis is the diverse means of interaction and representation of money value, it can be hoped that more complex features would be incorporated in the quest for a wider array of representations and interaction as well as integration into everyday life and the environment. One way would be to have biometric data in form of details such as eye patterns, blood types, DNA, natural smell and facial composition of all people, known across the system and effectively push initiative from the user to the system, leaving only authentication to the user for security purposes. Biochips would be connected to a network functioning not only on conventional means of connectivity but on extended mesh networks ensuring service at all times.

Once integrated, people themselves will effectively become high profile member components of the system, participating at both high and low levels of detail. Payment systems would evolve to accommodate this and people would easily transfer value between one another without ever making contact or showing visible intentions of transferring value. Those in the habit of stealing would leave more obvious trails directly linked to them and if possible, a mechanism to ‘arrest’ or ‘stop’ a value from use placed as a safeguard measure.

Also, integrated chips relying on the conversion of thermal energy of people into electrical energy and using it to power circuits handling value and communicating wirelessly with proximity devices linking to servers could be incorporated by implanting them in people making what would be called P.a.M. standing for People are Money.

Conclusion:

For as long as there is collective trust in a given value of money and the properties associated with money are preserved, the representation of that money value can assume various forms. In the context of an information society, this representation can be digitally realised with integrity preserved and enforced across handling systems.

Appendix

Some of the most important functionality can be isolated on a few Java classes in the case of the standalone applications, a few libraries and a few scripts in terms of web pages. In this section, some of the critical code is shown, together with code for the failed motion detection module.

Among the libraries used was Sphinx java library for speech recognition, Sun Microsystems free text to speech was used for speech synthesis with MBROLA voices, as mentioned in the discussion and ZXing (Zebra Crossing) java library for QR-Code generation and static reading.

Here are some of the classes and scripts identified as being at the core of the system:

Command.java

This class embodies the current set of commands understood by the server and sent by the client. It also provides a wrapper for data carried over networks through serialization. It is used by all applications developed.

```
package basic;


/**
 *
 * @author Chishala Matete Mpundu
 */
public class Command implements java.io.Serializable{
    public static final String MAIN_SERVER = "192.168.1.14";
    public Object data;
    public int command;
    public Command(int cmd){
        command = cmd;
        data = null;
    }
    public Command(int cmd,Object command_data){
        command = cmd;
        data = command_data;
    }
    public Command(){
        command = -1;
        data = null;
    }
    /**
     * Sets the command
     * @param cmd
     */
    public void setCommand(int cmd){
        command = cmd;
    }
    /**
     * Gets the actual command
     * @return String
     */
    public int getCommand(){
        return command;
    }
    /**
     * Sets the command data
     * @param command_data
     */
    public void setCommandData(Object command_data){
        data = command_data;
    }
    /**
     * Gets the data associated with the command.
     * @return Object
     */
    public Object getCommandData(){
        return data;
    }
    public static final int A_Y_T = 1;
    public static final int STORE_NUMBER = 2;
    public static final int GET_NUMBER = 3;
    public static final int DELETE_NUMBER = 4;
    public static final int STORE_USER = 5;
    public static final int DELETE_USER = 6;
    public static final int GET_ALL_BY_ALIAS = 7;
    public static final int GET_BY_ALIAS_AND_PIN = 8;
    public static final int UPDATE_VALUE = 9;
    public static final int GET_USERS_MONEY = 10;
    public static final int GET_TOTAL_VALUE_OF_USER_MONEY = 11;
    public static final int MODIFY_USER_PIN = 12;
    public static final int GET_USER = 13;
    public static final int GET_SYSTEM_USERS_AND_COUNT = 14;
    public static final int GET_MONEY_CIRCULATING = 15;
    public static final int VALIDATE_LOGIN = 16;
    public static final int EXPORT_SYSTEM_DATABASE = 18;
    public static final int GRAB_KEY_DATABASE = 19;
    public static final int GRAB_MAIN_DATABASE = 20;
    public static final int GET_LOG = 21;
    public static final int BACKUP_ALL_TABLES = 22;
    public static final int GET_USER_ID = 23;
    public static final int CHANGE_ALIAS = 24;
    public static final int CHANGE_OWNER = 25;
    public static final int VALIDATE_CLIENT_LOGIN = 26;
    public static final int TRANSFER_VALUE = 27;
    public static final int CHANGE_USER_PIN = 28;
    public static final int RETAILER_TRANSACTION = 29;
    public static final int STORE_PATTERN_MONEY = 30;
    public static final int GET_BARCODE_EQUIVALENT = 31;
    public static final int GET_USER_AND_MONEY = 32;
    public static final int IS_THIS_YOUR_ACCOUNT = 33;
    public static final int ASSIMILATE_VALUE = 34;
    public static final int IS_BANK_REGISTERED = 35;
    public static final int DEDUCT_FROM_BALANCE = 36;
    public static final int ADD_TO_BALANCE = 37;
    public static final int WITHDRAW_FROM_ACCOUNT = 38;
    public static final int DEPOSIT_INTO_ACCOUNT = 39;
    public static final int GET_PATTERN_MONEY =40;
}
```

MoneyNumber.java

This class is the embodiment of value. It associates the user with a value as a double and an alias for textual representation.

```java
import java.security.*;
import java.util.*;
import javax.crypto.*;

public class MoneyNumber implements java.io.Serializable{
    private String number;
    private double value;
    private NumberSecurity security_manager;
    private String date_issued;
    private String userPIN;
    private Key key;
    private byte[] numberBytes;
    private String alias_name;
    /**
     * Constructor foe raw MoneyNumber object with nullified properties.
     */
    public MoneyNumber(){
        number = null;
        value = 0.00;
        security_manager = new NumberSecurity();
        date_issued = security_manager.dated();
        userPIN = "";
        alias_name = null;
        key = null;
    }
    /**
     * MoneyNumber object constructor focused on the key id as main property set.
     * @param id
     */
    public MoneyNumber(String pin){
        number = null;
        value = 0.00;
        security_manager = null;
        userPIN = pin;
        alias_name = null;
        key = null;
    }
    /**
     * Constructor for familiar MoneyNumber object where the number string and id
are set.
     * @param num
     * @param id
     */
    public MoneyNumber(String num,String pin,String alias){
        number = num;
        value = 0.00;
        String userPIN = pin;
        security_manager = new NumberSecurity();
        alias_name = alias;
        numberBytes = number.getBytes();
        try{
            KeyGenerator keyGen = KeyGenerator.getInstance("DES");
            keyGen.init(56);
            key = keyGen.generateKey();
            Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
            cipher.init(Cipher.ENCRYPT_MODE,key);
            numberBytes = cipher.doFinal(numberBytes);
            number = new String(numberBytes,"UTF8");
        }catch(Exception exc){
        }
    }
    /**
     * Constructor for new MoneyNumber objects where the value is attached.
     * @param val
     */
   public MoneyNumber(double val,String pin){
       number = generateNumberString();
       value = val;
       security_manager = new NumberSecurity();
       date_issued = security_manager.dated();
       userPIN = pin;
       alias_name = null;
       numberBytes = number.getBytes();
       try{
            KeyGenerator keyGen = KeyGenerator.getInstance("DES");
            keyGen.init(56);
            key = keyGen.generateKey();
      }catch(Exception exc){
      }
   }
   /**
    * Constructor for new objects with money number string and value known.
    * @param num
    * @param val
    */
    public MoneyNumber(String num,double val,String pin){
        number = num;
        value = val;
        security_manager = new NumberSecurity();
        Key key = security_manager.encryptionKey();
        date_issued = security_manager.generateKeyID();
        userPIN = pin;
        alias_name = null;
        numberBytes = number.getBytes();
        try{
            KeyGenerator keyGen = KeyGenerator.getInstance("DES");
            keyGen.init(56);
           key = keyGen.generateKey();
      }catch(Exception exc){
    }
    }

    public void setAlias(String aliasName){
        alias_name = aliasName;
    }
    public String getAlias(){
        return alias_name;
    }
    public void decrypt(){
        try{
        Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
        cipher.init(Cipher.DECRYPT_MODE,key);

        byte[] cypherText = numberBytes;
        byte[] plainText = cipher.doFinal(cypherText);
        number = new String(plainText,"UTF8");
        }catch(Exception exc){
            exc.printStackTrace();
        }

    }
    public Key getKey(){
        return key;
    }
    public void setKey(){
        try{
            KeyGenerator keyGen = KeyGenerator.getInstance("DES");
            keyGen.init(56);
            key = keyGen.generateKey();
        }catch(Exception exc){

        }
    }
    public String getPIN(){
        return userPIN;
    }
    public void setPIN(String pin){
        userPIN = pin;
    }
    public void encrypt(){
        try{
            Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
            cipher.init(Cipher.ENCRYPT_MODE,key);

            numberBytes = cipher.doFinal(numberBytes);
            number = new String(numberBytes,"UTF8");
        }catch(Exception exc){
            exc.printStackTrace();
        }
    }
    public void setDateOfIssue(String issueDate){
        date_issued = issueDate;
    }
    public String getDateOfIssue(){
        return date_issued;
    }
    public void setValue(double val){
        value = val;
    }
    public double getValue(){
        return value;
    }
    public void setNumber(String num){
        number = num;
        numberBytes = number.getBytes();
    }
    public String getNumber(){
        return number;
    }
    public static String generateNumberString(){
        String numGenerated = "";
        Random randomSelector = new Random();
        int limit = 7;
        char[] letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ".toCharArray();
        int counter = 0;
        while(counter < limit){
            int selectedNumber = randomSelector.nextInt(10);
            char letter = letters[randomSelector.nextInt(26)];
            if(counter == 4){
                numGenerated = numGenerated+"-"+letter+""+selectedNumber;
            }else{ numGenerated = numGenerated+""+letter+""+selectedNumber;}
            counter++;
        }
        return numGenerated;
    }
    @Override
    public String toString(){
        return number;
    }

    @Override
    public boolean equals(Object object){
        if((object != null) & (object instanceof MoneyNumber) ){
            return
(((MoneyNumber)object).getNumber().equalsIgnoreCase(this.getNumber())?
true:false);
        }else return false;
    }

    @Override
    public int hashCode(){
        int hash = 5;
        hash = 81 * hash + (number != null ? number.hashCode():0);
        hash = 83 * hash + (date_issued != null ? date_issued.hashCode():(new
Random().nextInt(10)));
        return hash;
    }
}
```

IdentifiedUser.java

This class identifies users in the system.

```
import java.util.Random;

public class IdentifiedUser implements java.io.Serializable{
    private String userName;
    private String nationalID;
    private String userPIN;
    private NumberSecurity security;

    public IdentifiedUser(String name,String id,String assignedPIN){
        userName = name;
        nationalID = id;
        userPIN = assignedPIN;
        security = new NumberSecurity();
    }
    public IdentifiedUser(String name,String id){
        userName = name;
        nationalID = id;
        security = new NumberSecurity();
        userPIN = security.generatePIN();
    }
    public IdentifiedUser(){
        userName = null;
        nationalID = null;
}
    public void setUserName(String name){
        userName = name;
    }
    public String getUserName(){
        return userName;
    }
    public void setNationalID(String id){
        nationalID = id;
    }
    public String getNationalID(){
        return nationalID;
    }
    public void setUserPIN(String pin){
        userPIN = pin;
    }
    public String getUserPIN(){
        return userPIN;
    }
    @Override
    public String toString(){
        return userName+":"+nationalID;
    }
    @Override
    public boolean equals(Object object){
        if((object != null) & (object instanceof IdentifiedUser)){
            return
(((IdentifiedUser)object).getNationalID().equalsIgnoreCase(this.getNationalID()) ?
true:false);
        }else return false;
    }
    @Override
    public int hashCode(){
        int hash = 7;
        hash = 89 * hash + (nationalID != null ?nationalID.hashCode():((new
Random()).nextInt(10)));
        return hash;
    }
}
```

ExtractSignature.java

Responsible for extracting the signature number wrapped inside a number with each downloaded standalone client applications:

```
public class ExtractSignature{

public static String extractSignature(String sigWrapper){
String signatureWrapper = sigWrapper;
int qIndex = Integer.parseInt(signatureWrapper.substring(0,1));
int cIndex = Integer.parseInt(signatureWrapper.substring(1,2));
char[] digits = signatureWrapper.toCharArray();
digits[qIndex] = '?';
digits[cIndex] = ':';
int counter = 0;
String extractedSignature = "";
char firstDigit = digits[0];
char lastDigit = digits[digits.length-1];
digits[0] = digits[qIndex];
digits[qIndex] = firstDigit;
digits[digits.length-1] = digits[cIndex];
digits[cIndex] = lastDigit;

while(counter < digits.length){
    extractedSignature = extractedSignature+""+digits[counter];
    counter++;
}
    return extractedSignature.substring(1,9);
}
}
```

BarCodeGenerator.java

This class generates the QR-Codes encoding the textual representation of the value. It is located on the Web Server component.

```
import com.google.zxing.common.BitMatrix;
import com.google.zxing.qrcode.QRCodeWriter;
import com.google.zxing.BarcodeFormat;
import com.google.zxing.client.j2se.MatrixToImageWriter;
import java.io.*;
import java.util.*;

public class BarCodeGenerator {
    public static void main(String[] args){
            try{
             String errorOutcome = "Not Encoded";
                String moneyAlias = args[0];
 String customerName = args[1];
                String currentDate = logDate();
                currentDate = currentDate.trim();
                String barcodeFileName =
customerName+"_"+moneyAlias+"_"+currentDate+".jpg";;
                String barcodeFilePath = "./barcodes/";
                int width = 400;
                int height = 300;
                String imageFormat = "png";
                File barcodeFile = new File(barcodeFilePath,barcodeFileName);
                barcodeFile.setReadOnly();
                BitMatrix bitMatrix = new QRCodeWriter().encode(moneyAlias,
BarcodeFormat.QR_CODE, width, height);
                MatrixToImageWriter.writeToStream(bitMatrix, imageFormat, new
FileOutputStream(barcodeFile));
                if(barcodeFile.exists()){
                String fileName = barcodeFileName;
                System.out.println(fileName);
                }else{
                    System.out.println(errorOutcome);
                }
        }catch(Exception exc){
            exc.printStackTrace();
        }

    }
    public static String dated(){
        Calendar calendar = Calendar.getInstance();
        Date date = calendar.getTime();
        String currentTime = date.toString();
```

```java
        String month = currentTime.substring(4,7);
        String day = currentTime.substring(8,10);
        String year = currentTime.substring(24);
        String time = currentTime.substring(11,19);
        String dateIssued = day+"-"+month+"-"+year+" "+time+"";
        return dateIssued;
    }

    public static String logDate(){
        String date = dated();
        String day = date.substring(0,2);
        String month = date.substring(3,6);
        String year = date.substring(7,12);
        String hour = date.substring(12,14);
        String minute = date.substring(15,17);
        String seconds = date.substring(18);
        date = hour+minute+seconds+"_"+day+"_"+month+"_"+year;
        return date;
    }

    public static String assembleName(String fullName){
        String result = "";
        StringTokenizer tokenizer = new StringTokenizer(fullName);
        ArrayList<String> names = new ArrayList<String>();
           while(tokenizer.hasMoreTokens()){
                String oneOfTheNames = tokenizer.nextToken();
                names.add(oneOfTheNames);
           }
        Iterator<String> iterator = names.iterator();
        result = names.get(0);
        int index = 1;
        while(index <= names.size()){
             result = result+"_"+names.get(index);
             index++;
        }
        return result;
    }

}
```
BarCodeReader.java

This class provides functionality for identifying the money value textual representation from the QR-Code provided.It is also housed under the Web Server component:

```java
import com.google.zxing.*;
import com.google.zxing.client.j2se.*;
import com.google.zxing.common.HybridBinarizer;
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.*;

public class BarCodeReader {
   public static void main(String[] args){
         if(args.length != 0){
       try{
       String barcodeFile = args[0];
       InputStream barCodeInputStream = new FileInputStream(barcodeFile);
       BufferedImage barCodeBufferedImage = ImageIO.read(barCodeInputStream);

        LuminanceSource source = new
BufferedImageLuminanceSource(barCodeBufferedImage);
```

```
        BinaryBitmap bitmap = new BinaryBitmap(new HybridBinarizer(source));
        com.google.zxing.Reader reader = new MultiFormatReader();
        Result result = reader.decode(bitmap);
            System.out.println(""+ result.getText());
       }catch(Exception exc){
       }}else{
         System.out.println("No Alias Available: Reason: You many have not
specified one or an error occurred during translation.");
       }
   }
}
```

LaudHousing.java

Class responsible for voice guided pattern generation. Part of a simulator retail terminal:

```
import edu.cmu.sphinx.frontend.util.Microphone;
import edu.cmu.sphinx.recognizer.Recognizer;
import edu.cmu.sphinx.result.Result;
import edu.cmu.sphinx.util.props.ConfigurationManager;
import java.awt.*;
import javax.swing.*;
import java.util.*;
import java.util.concurrent.*;
import java.awt.event.*;
import com.sun.speech.freetts.*;
import java.net.*;
import java.io.*;
import basic.*;

public class LaudHousing extends JFrame implements MouseListener {

    private static final long serialVersionUID = 505l;
    private Box
arranger,amountBox,masterLayout,aliasField,retailerAliasField,masterAliasBox,pinBo
x;
    private Panel choicePanel, shapersPanel;
    private JLabel
aliasLabel,amountLabel,help,retailerAliasLabel,privatePINLabel,saveStatus,
confirmLabel, diamondLabel, triangleLabel, listeningStatus, starLabel,
circleLabel, crossLabel, squareLabel;
    private Random colorSelector;
    private Color[] colors = {Color.BLUE, Color.YELLOW, Color.PINK, Color.GREEN,
Color.ORANGE, Color.RED, Color.BLACK};
    private ExecutorService executor;
    private Shaper[] shapers = new Shaper[6];
    private GridLayout choiceGrid, shapersGrid;
    private String figure,moneyIDYielded,moneyToAlter,pin,moneyCode,retailerMoney;
    private Voice speaker;
    private VoiceManager voiceManager;
    private JTextField alias,retailerAlias,amountField;
    private JPasswordField privatePINField;
    private JButton unset,enter;
    private MoneyNumber money;
    private double val;
    private int period_occurrence,control;
    private ImageIcon icon;

    public LaudHousing() {
        super("Acoustic Money >> Talk To Me");
        masterLayout = Box.createHorizontalBox();
        aliasField = Box.createHorizontalBox();
        retailerAliasField = Box.createHorizontalBox();
        masterAliasBox = Box.createVerticalBox();
        pinBox = Box.createHorizontalBox();
        pinBox.setMaximumSize(new Dimension(800,60));
        amountBox = Box.createHorizontalBox();
        amountBox.setMaximumSize(new Dimension(800,60));
        control = 0;
        figure = "";
        val = 0.00;
        period_occurrence = 0;

        help = new JLabel("Help");
        help.addMouseListener(
                    new MouseAdapter(){
                          @Override
                          public void mouseClicked(MouseEvent me){
                                JOptionPane.showMessageDialog(null,"First Enter
Your PIN in the 'Customer PIN' field\nNext, click on the microphone to invoke the
command listener.\nA limited set of commands are understood"
                                              + "yielding shapes on the lion
panels under your authority, among them:\n"
                                              + "\t1. Wake Up: Wakes up the
listener\n\t2. Circle or Round: as the label is shown, draws a circle as part of
the pattern you are making\n\t3."
                                              + " Star, Six Pointed Star or Israel
Star: Draws A star\n\t4.Square or Box: Draws a square\n\t5. Cross: Draws a
cross\n\t6. Triangle or Pyramid: Draws a triangle\n\t7. Kite or Diamond: Draws a
Rhombus\n\t8."
                                              + "Clear: Undoes everything in the
pattern\n\t9. Undo: Undoes the recently drawn shape or figure\n\t10. Value: If
your pin is entered, Confirm asks the listener to query the value of the pattern
you just drew via voice\n\t11."
                                              + "Confirm: confirms the pattern
just made and ready for setting against a given value\n\t12. Hide: Hides the
listener\n\t13. Die or Kill: Closes the listener.","How To
use",JOptionPane.INFORMATION_MESSAGE);
                          }
                    });
        help.setBackground(Color.GRAY);
        help.setForeground(Color.GREEN.darker());
        amountLabel = new JLabel("amount");
        amountField = new JTextField(20);
        amountBox.add(amountLabel);
        amountBox.add(Box.createHorizontalStrut(20));
        amountBox.add(amountField);

        privatePINLabel = new JLabel("customer pin:");
        privatePINField = new JPasswordField(20);
        pinBox.add(privatePINLabel);
        pinBox.add(Box.createHorizontalStrut(20));
        pinBox.add(privatePINField);


        arranger = Box.createVerticalBox();
        pin = "";
        money = new MoneyNumber();

        executor = Executors.newCachedThreadPool();

        moneyToAlter = "";
        moneyIDYielded = "";
        moneyCode = "";
        retailerMoney = "";
        voiceManager = VoiceManager.getInstance();
        speaker = voiceManager.getVoice("kevin16");
        speaker.allocate();
        choiceGrid = new GridLayout(4, 4, 5, 5);
        shapersGrid = new GridLayout(2, 3, 10, 10);
        choicePanel = new Panel();
        choicePanel.setVisible(true);
        choicePanel.setSize(200, 500);
        choicePanel.setLayout(choiceGrid);
        choicePanel.setMaximumSize(new Dimension(300, 500));
        saveStatus = new JLabel("!");
        choicePanel.add(saveStatus);
        Class metaObject = this.getClass();
        URL url = metaObject.getResource("mic.jpg");
        icon = new ImageIcon(url);
        listeningStatus = new JLabel("click",icon, SwingConstants.CENTER);
        listeningStatus.addMouseListener(this);
        choicePanel.add(listeningStatus);
        aliasLabel = new JLabel("money:");
        alias = new JTextField(20);
        alias.setEditable(false);
        retailerAlias = new JTextField(20);
        retailerAliasLabel = new JLabel("retailer's money:");
        retailerAliasField.add(retailerAliasLabel);
        retailerAliasField.add(Box.createHorizontalStrut(20));
        retailerAliasField.add(retailerAlias);
        retailerAliasField.setMaximumSize(new Dimension(800,60));
        unset = new JButton("unset");
        unset.addActionListener(
                    new ActionListener(){
                          @Override
                          public void actionPerformed(ActionEvent ae){
                          moneyToAlter = "";
                          alias.setText("");
                          }});
        enter = new JButton("set");
        enter.addActionListener(
                    new ActionListener(){
                          @Override
                          public void actionPerformed(ActionEvent ae){
                                String aka = alias.getText();
                                String customer_pin = new
String(privatePINField.getPassword());
                                if(aka.equals("")  == true | aka == null){
                                      JOptionPane.showMessageDialog(null,"You
have to specify the money to use", "Invalid Money
Specified",JOptionPane.ERROR_MESSAGE);
                                }else if(customer_pin.equalsIgnoreCase("")){
                                      JOptionPane.showMessageDialog(null,"Please
Enter PIN","No PIN Entered", JOptionPane.ERROR_MESSAGE);
                                }else{
```

```
                                setCustomerPIN(customer_pin);
                                String sVal = amountField.getText();
                                double v = Double.parseDouble(sVal.equals("") ?
"0.00":sVal);

                                setVal(v);
                                pin = getCustomerPIN();
                                setAliasToUse(aka);
                                resetPeriodOccurrence();
                                executor.execute(new SeekMoney());

                                }
                        }});
        aliasField.add(aliasLabel);
        aliasField.add(Box.createHorizontalStrut(20));
        aliasField.add(alias);
        aliasField.add(Box.createHorizontalStrut(20));
        aliasField.add(enter);
        aliasField.add(Box.createHorizontalStrut(20));
        aliasField.add(unset);
        aliasField.setMaximumSize(new Dimension(800,60));
        masterAliasBox.add(pinBox);
        masterAliasBox.add(aliasField);
        masterAliasBox.add(amountBox);
        masterAliasBox.add(retailerAliasField);
        circleLabel = new JLabel("circle");
        choicePanel.add(circleLabel);
        starLabel = new JLabel("star");
        choicePanel.add(starLabel);
        crossLabel = new JLabel("cross");
        choicePanel.add(crossLabel);
        squareLabel = new JLabel("square");
        choicePanel.add(squareLabel);
        diamondLabel = new JLabel("diamond");
        choicePanel.add(diamondLabel);
        triangleLabel = new JLabel("triangle");
        choicePanel.add(triangleLabel);
        confirmLabel = new JLabel("confirm");
        confirmLabel.addMouseListener(this);
        choicePanel.add(confirmLabel);
        choicePanel.add(help);
        shapersPanel = new Panel();
        shapersPanel.setVisible(true);
        shapersPanel.setSize(750, 500);
        shapersPanel.setLayout(shapersGrid);
        colorSelector = new Random();
        Shaper shaper0 = new Shaper(120,250);
        shaper0.setLineColor(colors[colorSelector.nextInt(colors.length)]);
        shaper0.addMouseListener(shaper0);
        shapersPanel.add(shaper0);
        shapers[0] = shaper0;
        Shaper shaper1 = new Shaper(120,250);
        shaper1.setLineColor(colors[colorSelector.nextInt(colors.length)]);
        shaper1.addMouseListener(shaper1);
        shapersPanel.add(shaper1);
        shapers[1] = shaper1;
        Shaper shaper2 = new Shaper(120,250);
        shaper2.setLineColor(colors[colorSelector.nextInt(colors.length)]);
        shaper2.addMouseListener(shaper2);
        shapersPanel.add(shaper2);
        shapers[2] = shaper2;
        Shaper shaper3 = new Shaper(120,250);
        shaper3.setLineColor(colors[colorSelector.nextInt(colors.length)]);
        shaper3.addMouseListener(shaper3);
        shapersPanel.add(shaper3);
        shapers[3] = shaper3;
        Shaper shaper4 = new Shaper(120,250);
        shaper4.setLineColor(colors[colorSelector.nextInt(colors.length)]);
        shaper4.addMouseListener(shaper4);
        shapersPanel.add(shaper4);
        shapers[4] = shaper4;
        Shaper shaper5 = new Shaper(120,250);
        shaper5.setLineColor(colors[colorSelector.nextInt(colors.length)]);
        shaper5.addMouseListener(shaper5);
        shapersPanel.add(shaper5);
        shapers[5] = shaper5;
        masterLayout.add(shapersPanel);
        masterLayout.add(Box.createHorizontalStrut(10));
        masterLayout.add(choicePanel);
        arranger.add(masterAliasBox);
        arranger.add(masterLayout);
        add(arranger);
        setVisible(false);
        setSize(950, 500);
    }
    public void setVal(double v){
       val = v;
    }
    public double getVal(){
       return val;
    }
    public void setFigure(String s){
       figure = s;
    }
    public String getFigure(){
       return figure;
    }
    public void resetFigure(){
       figure = "";
    }
    public void resetPeriodOccurrence(){
       period_occurrence = 0;
    }
    public void setRetailerMoney(){
       retailerMoney = retailerAlias.getText();
    }
    public void setRetailerMoney(String m){
       retailerMoney = m;
    }
    public String getRetailerMoney(){
       return retailerMoney;
    }
    public void setAliasToUse(String money){
       moneyIDYielded = money;
    }
    public String getAliasToUse(){
       return moneyIDYielded;
    }
    public void appear(){
        this.setVisible(true);
    }
    public void setCustomerPIN(String s){
        pin = s;
    }
    public String getCustomerPIN(){
        return pin;
    }
    public void disappear(){
        this.setVisible(false);
        kill(this);
    }
    @Override
    public void mouseClicked(MouseEvent me) {
        if (((JLabel) me.getSource()).equals(listeningStatus)) {
            listeningStatus.setForeground(Color.BLUE);
            executor.execute(new Laudhauzing());
        } else if (((JLabel) me.getSource()).equals(confirmLabel)) {
            speaker.speak(moneyIDYielded);
        }
    }
    @Override
    public void mouseEntered(MouseEvent me) {
    }
    @Override
    public void mouseExited(MouseEvent me) {
    }
    @Override
    public void mousePressed(MouseEvent me) {
    }
    @Override
    public void mouseReleased(MouseEvent me) {
    }
    private class Laudhauzing implements Runnable {
        @Override
        public void run() {
            ConfigurationManager cm = new
ConfigurationManager(Laudhauzing.class.getResource("masterconfig.xml"));
            Recognizer recognizer = (Recognizer) cm.lookup("recognizer");
            recognizer.allocate();
            Microphone microphone = (Microphone) cm.lookup("microphone");
            if (!microphone.startRecording()) {
             recognizer.deallocate();
            }
            listeningStatus.setText("speak");
            int index = -1;
            do {
                Result result = recognizer.recognize();
                if (result != null) {
                    String resultText = result.getBestFinalResultNoFiller();
                    if (resultText.equalsIgnoreCase("star") |
resultText.equalsIgnoreCase("israel star") | resultText.equalsIgnoreCase("star of
israel") | resultText.equalsIgnoreCase("six pointed star")) {
                        index = (index + 1)%shapers.length;
                            shapers[index].setShape("star");
                            moneyIDYielded = moneyIDYielded + "star ";
                            alias.setText("");
                            alias.setText(moneyIDYielded);
                    } else if (resultText.equalsIgnoreCase("circle") |
resultText.equalsIgnoreCase("round")) {
                        index = (index + 1)%shapers.length;
                            shapers[index].setShape("circle");
                            moneyIDYielded = moneyIDYielded + "circle ";
                            alias.setText("");
                            alias.setText(moneyIDYielded);
                    }else if (resultText.equalsIgnoreCase("diamond") |
resultText.equalsIgnoreCase("kite") | resultText.equalsIgnoreCase("rhombus")) {
                        index = (index + 1)%shapers.length;
                             shapers[index].setShape("diamond");
                            moneyIDYielded = moneyIDYielded + "diamond ";
                            alias.setText("");
                            alias.setText(moneyIDYielded);
                    } else if (resultText.equalsIgnoreCase("cross") |
resultText.equalsIgnoreCase("holy cross")) {
                        index = (index + 1)%shapers.length;
shapers[index].setShape("cross");
                            moneyIDYielded = moneyIDYielded + "cross ";
                            alias.setText("");
                            alias.setText(moneyIDYielded);
                    } else if (resultText.equalsIgnoreCase("triangle") |
resultText.equalsIgnoreCase("Upward Arrow")) {
                        index = (index + 1)%shapers.length;
                            shapers[index].setShape("triangle");
                            moneyIDYielded = moneyIDYielded + "triangle ";
                            alias.setText("");
                            alias.setText(moneyIDYielded);
                    } else if (resultText.equalsIgnoreCase("value")) {
                          moneyIDYielded = moneyIDYielded.trim();
                          executor.execute(new SeekMoney());
                          double value_of_money = money.getValue();
                          speaker.speak("Value: "+value_of_money);
                    } else if (resultText.equalsIgnoreCase("undo")) {
                          if(index == -1){
                                 index += 1;
                          }
                        shapers[index].setShape("nothing");
                        StringTokenizer tokenizer = new
StringTokenizer(moneyIDYielded);
                        String temp = "";
                        int count = 0;
                        while (tokenizer.hasMoreTokens()) {
                            String choice = tokenizer.nextToken();
                            if (count == index) {
                                break;
                            } else {
                                temp += choice + " ";
                            }
                            count++;
                        }
                        moneyIDYielded = temp;
                        alias.setText(moneyIDYielded);
                        if (index == 0) {
                            index = shapers.length - 1;
                        } else if(index > 0){
                            index = index - 1;
                        }
                    }else if(resultText.equalsIgnoreCase("Are You There")){
```

```
                              speaker.speak("Laud Hauzing, Always By Your Side");
                        }else if(resultText.equalsIgnoreCase("Wake Up")){
                              speaker.speak("Laud Hauzing Is Wide Awake");
                        }else if (resultText.equalsIgnoreCase("transact")) {
                              setRetailerMoney();
                              if(getRetailerMoney().equalsIgnoreCase("")){
                                    JOptionPane.showMessageDialog(null,"Transaction
cannot be completed until the cashier sets the retailer's money to debit","
Retailer Money Not Set",JOptionPane.ERROR_MESSAGE);
                              }else{
                            executor.execute(new
PerformTransaction(pin,getAliasToUse(),val,getRetailerMoney()));
                              }
                        } else if (resultText.equalsIgnoreCase("confirm")) {
                              if(moneyIDYielded.equals("")){
                                    speaker.speak("Nothing Specified");
                              }else{
                            speaker.speak(moneyIDYielded);
                            alias.setText(moneyIDYielded);
                              }
                        } else if (resultText.equalsIgnoreCase("die") |
resultText.equalsIgnoreCase("kill")) {
                            kill(this);
                        } else if (resultText.equalsIgnoreCase("clear")) {
                            index = -1;
                            int counter = 0;
                            while (counter < shapers.length) {
                                shapers[counter].setShape("nothing");
                                counter++;
                            }
                            moneyIDYielded = "";
                            alias.setText("");
                        }

                    } else {
                        continue;
                    }
                } while (true);
            }

            public void kill(Runnable rnbl) {
                rnbl = null;
            }
        }
        public String getMoneyCode(){
           return moneyCode;
        }
        public MoneyNumber getMoney(){
           return money;
        }
        private class PerformTransaction implements Runnable{
           private String customerMoney,customerPin,retMoney;
           private double moneyVal;
           public PerformTransaction(String p,String clientMoney,double val,String
retailermoney){
                 customerMoney = clientMoney;
                 customerPin = p;
                 retMoney = retailermoney;
                 moneyVal = val; }
           @Override
           public void run(){
                 try{
                    Object[] transactionDetails =
{customerPin,customerMoney,moneyVal,retMoney};
                    Command command = new Command();
                    command.setCommand(Command.RETAILER_TRANSACTION);
                    command.setCommandData(transactionDetails);
                    try{
                        Socket socket = new Socket(Command.MAIN_SERVER,5050);
                        ObjectOutputStream outputStream = new
ObjectOutputStream(socket.getOutputStream());
                        outputStream.writeObject(command);
                        outputStream.flush();
                        ObjectInputStream inStream = new
ObjectInputStream(socket.getInputStream());
                        boolean operational_result = inStream.readBoolean();
                    if(operational_result == true){
                        JOptionPane.showMessageDialog(null,"Transaction
Completed.","Operation Successful",JOptionPane.INFORMATION_MESSAGE);
                    }else{
                       JOptionPane.showMessageDialog(null,"Transaction
Failed.","Failed Transaction",JOptionPane.ERROR_MESSAGE);
                    }
                    inStream.close();
                    outputStream.close();
                    socket.close();
                    }catch(Exception exc){
                    }finally{
                    alias.setText("");
                    retailerAlias.setText("");
                    privatePINField.setText("");
                    setCustomerPIN("");
                    setRetailerMoney("");
                    setAliasToUse("");
                    pin = "";
                    }
                 }catch(Exception exc){
                 }
           }
           public void kill(Runnable rnbl){
                 rnbl = null;
           }
        }
        private class SeekMoney implements Runnable{
           @Override
           public void run(){
                 try{
                       Socket socket = new Socket(Command.MAIN_SERVER,5050);
                       ObjectOutputStream outStream = new
ObjectOutputStream(socket.getOutputStream());
                       Command command = new Command();
                       command.setCommand(Command.GET_USER_AND_MONEY);
                       command.setCommandData(pin);
                       outStream.writeObject(command);
                       outStream.flush();
```

```
                    ObjectInputStream inStream = new
ObjectInputStream(socket.getInputStream());
                    Object[] result = (Object[])inStream.readObject();
                    UserMoneys stack = (UserMoneys)result[1];
                    ArrayList<MoneyNumber> stash = stack.getUserMoneys();
                    Iterator<MoneyNumber> iterator = stash.iterator();
                    MoneyNumber cmprtr = null;
                                        while(iterator.hasNext()){
                           cmprtr = iterator.next();
       if(cmprtr.getAlias().equalsIgnoreCase(getAliasToUse())){
                                  setMoney(cmprtr);
                                  setMoneyCode(cmprtr.getNumber());
                                  break;
                           }
                    }
                    inStream.close();
                    outStream.close();
                    socket.close();
             }catch(Exception exc){
             }
      }
      public void kill(Runnable rnbl){
             rnbl = null;
      }
   }
   public void setMoney(MoneyNumber nm){
      money = nm;
   }
   public void setMoneyCode(String c){
      moneyCode = c;
   }
   public void kill(LaudHousing lh){
      lh = null;
   }
}
```

dict.gram

The grammar file containing the recognised terms for the voice guided pattern generator:

```
#JSGF V1.0;
grammar hello;
public <word>= (Circle | Wake Up | Are You There | Round | Triangle | Upward Arrow
| Cross | Holy Cross | Diamond | Star | Six Pointed Star | Star of Israel | Israel
Star | Kite | A | B | C | D | E | F);
public <decree>= (Value | Undo | Transact | Clear | Die | Kill | Confirm | Hide);
```

masterconfig.xml

The XML configuration file for the voice guided pattern generator.

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
    <property name="logLevel" value="WARNING"/>
    <property name="absoluteBeamWidth"  value="-1"/>
    <property name="relativeBeamWidth"  value="1E-80"/>
    <property name="wordInsertionProbability" value="1E-36"/>
    <property name="languageWeight"     value="8"/>
    <property name="frontend" value="epFrontEnd"/>
    <property name="recognizer" value="recognizer"/>
    <property name="showCreations" value="false"/>
    <component name="recognizer" type="edu.cmu.sphinx.recognizer.Recognizer">
```

```
        <property name="decoder" value="decoder"/>
        <propertylist name="monitors">
            <item>accuracyTracker </item>
            <item>speedTracker </item>
            <item>memoryTracker </item>
        </propertylist>
    </component>
    <component name="decoder" type="edu.cmu.sphinx.decoder.Decoder">
        <property name="searchManager" value="searchManager"/>
    </component>
    <component name="searchManager"
        type="edu.cmu.sphinx.decoder.search.SimpleBreadthFirstSearchManager">
        <property name="logMath" value="logMath"/>
        <property name="linguist" value="flatLinguist"/>
        <property name="pruner" value="trivialPruner"/>
        <property name="scorer" value="threadedScorer"/>
        <property name="activeListFactory" value="activeList"/>
    </component>
    <component name="activeList"
             type="edu.cmu.sphinx.decoder.search.PartitionActiveListFactory">
        <property name="logMath" value="logMath"/>
        <property name="absoluteBeamWidth" value="${absoluteBeamWidth}"/>
        <property name="relativeBeamWidth" value="${relativeBeamWidth}"/>
    </component>
    <component name="trivialPruner"
                type="edu.cmu.sphinx.decoder.pruner.SimplePruner"/>
    <component name="threadedScorer"
                type="edu.cmu.sphinx.decoder.scorer.ThreadedAcousticScorer">
        <property name="frontend" value="${frontend}"/>
    </component>
    <component name="flatLinguist"
                type="edu.cmu.sphinx.linguist.flat.FlatLinguist">
        <property name="logMath" value="logMath"/>
        <property name="grammar" value="jsgfGrammar"/>
        <property name="acousticModel" value="wsj"/>
        <property name="wordInsertionProbability"
                value="${wordInsertionProbability}"/>
        <property name="languageWeight" value="${languageWeight}"/>
        <property name="unitManager" value="unitManager"/>
    </component>
    <component name="jsgfGrammar" type="edu.cmu.sphinx.jsgf.JSGFGrammar">
        <property name="dictionary" value="dictionary"/>
        <property name="grammarLocation"
             value="resource:/client/"/>
        <property name="grammarName" value="dict"/>
       <property name="logMath" value="logMath"/>
    </component>
    <component name="dictionary"
        type="edu.cmu.sphinx.linguist.dictionary.FastDictionary">
        <property name="dictionaryPath"
value="resource:/WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz/dict/cmudict.0.6d"/>
        <property name="fillerPath"
        value="resource:/WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz/noisedict"/>
        <property name="addSilEndingPronunciation" value="false"/>
        <property name="allowMissingWords" value="false"/>
        <property name="unitManager" value="unitManager"/>
    </component>
    <component name="wsj"
type="edu.cmu.sphinx.linguist.acoustic.tiedstate.TiedStateAcousticModel">
```

```
        <property name="loader" value="wsjLoader"/>
        <property name="unitManager" value="unitManager"/>
    </component>
    <component name="wsjLoader"
type="edu.cmu.sphinx.linguist.acoustic.tiedstate.Sphinx3Loader">
        <property name="logMath" value="logMath"/>
        <property name="unitManager" value="unitManager"/>
        <property name="location"
value="resource:/WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz"/>
    </component>
    <component name="unitManager"
        type="edu.cmu.sphinx.linguist.acoustic.UnitManager"/>
    <component name="frontEnd" type="edu.cmu.sphinx.frontend.FrontEnd">
        <propertylist name="pipeline">
            <item>microphone </item>
            <item>preemphasizer </item>
            <item>windower </item>
            <item>fft </item>
            <item>melFilterBank </item>
            <item>dct </item>
            <item>liveCMN </item>
            <item>featureExtraction </item>
        </propertylist>
    </component>
    <component name="epFrontEnd" type="edu.cmu.sphinx.frontend.FrontEnd">
        <propertylist name="pipeline">
            <item>microphone </item>
            <item>dataBlocker </item>
            <item>speechClassifier </item>
            <item>speechMarker </item>
            <item>nonSpeechDataFilter </item>
            <item>preemphasizer </item>
            <item>windower </item>
            <item>fft </item>
            <item>melFilterBank </item>
            <item>dct </item>
            <item>liveCMN </item>
            <item>featureExtraction </item>
        </propertylist>
    </component>
    <component name="dataBlocker" type="edu.cmu.sphinx.frontend.DataBlocker">
    </component>
    <component name="speechClassifier"
           type="edu.cmu.sphinx.frontend.endpoint.SpeechClassifier">
        <property name="threshold" value="13"/>
    </component>
    <component name="nonSpeechDataFilter"
           type="edu.cmu.sphinx.frontend.endpoint.NonSpeechDataFilter"/>
    <component name="speechMarker"
           type="edu.cmu.sphinx.frontend.endpoint.SpeechMarker" >
        <property name="speechTrailer" value="50"/>
    </component>
    <component name="preemphasizer"
           type="edu.cmu.sphinx.frontend.filter.Preemphasizer"/>
    <component name="windower"
           type="edu.cmu.sphinx.frontend.window.RaisedCosineWindower">
    </component>
    <component name="fft"
          type="edu.cmu.sphinx.frontend.transform.DiscreteFourierTransform">
    </component>
    <component name="melFilterBank"
        type="edu.cmu.sphinx.frontend.frequencywarp.MelFrequencyFilterBank">
    </component>
    <component name="dct"
            type="edu.cmu.sphinx.frontend.transform.DiscreteCosineTransform"/>
    <component name="liveCMN"
              type="edu.cmu.sphinx.frontend.feature.LiveCMN"/>
    <component name="featureExtraction"
              type="edu.cmu.sphinx.frontend.feature.DeltasFeatureExtractor"/>
    <component name="microphone"
              type="edu.cmu.sphinx.frontend.util.Microphone">
        <property name="closeBetweenUtterances" value="false"/>
    </component>
    <component name="accuracyTracker"
               type="edu.cmu.sphinx.instrumentation.BestPathAccuracyTracker">
        <property name="recognizer" value="${recognizer}"/>
        <property name="showAlignedResults" value="false"/>
        <property name="showRawResults" value="false"/>
    </component>
    <component name="memoryTracker"
               type="edu.cmu.sphinx.instrumentation.MemoryTracker">
        <property name="recognizer" value="${recognizer}"/>
       <property name="showSummary" value="false"/>
       <property name="showDetails" value="false"/>
    </component>
    <component name="speedTracker"
               type="edu.cmu.sphinx.instrumentation.SpeedTracker">
        <property name="recognizer" value="${recognizer}"/>
        <property name="frontend" value="${frontend}"/>
       <property name="showSummary" value="true"/>
       <property name="showDetails" value="false"/>
    </component>
    <component name="logMath" type="edu.cmu.sphinx.util.LogMath">
        <property name="logBase" value="1.0001"/>
        <property name="useAddTable" value="true"/>
    </component>
</config>
```

Inside **Face_Money.php**

Server side Script that associates a face with value:

```
function processDetails($db){

        include("FaceDetector.php");

//

$image_field = "pic"; $dir =
$_SERVER['DOCUMENT_ROOT'].str_replace(basename($_SERVER['PHP_SELF']),'',$_SERVER['PHP_SELF'])
."";

$upload_form =
"http://".$_SERVER['HTTP_HOST'].str_replace(basename($_SERVER['PHP_SELF']),'',$_SERVER['PHP_S
ELF'])."face_money.php";

$panel =
"http://".$_SERVER['HTTP_HOST'].str_replace(basename($_SERVER['PHP_SELF']),'',$_SERVER['PHP_S
ELF'])."panel.php";

$upload_dir =
$_SERVER['DOCUMENT_ROOT'].str_replace(basename($_SERVER['PHP_SELF']),'',$_SERVER['PHP_SELF'])
."faces/";
```

```
$errors = array(1 => 'php.ini max file size exceeded',

                2 => 'html form max file size exceeded',

                3 => 'file upload was only partial',

                4 => 'no file was attached');

@is_uploaded_file($_FILES[$image_field]['tmp_name']) or error('Not an http
upload',$upload_form);

@getimagesize($_FILES[$image_field]['tmp_name'])or error('Only Image Files
allowed.',$upload_form);//getimagesize() is true if file is an image and false otherwise.

$fname = $_FILES[$image_field]['name'];

$fext = '.jpg';

$random_number_one = rand(1000,5000);$random_number_two = rand(100,999);$random_number_three
= rand(22,69);$random_number_four = rand(0,99);$random_number_five = rand(6000,7450);

$file_number =
$random_number_one."_".$random_number_two."_".$random_number_three."_".$random_number_four."_
".$random_number_five;

while(file_exists($upload_file_name =
$upload_dir.$file_number.$fext)){//$dir.$file_number.$fext

break;

}

//move file to image uploads folder

@move_uploaded_file($_FILES[$image_field]['tmp_name'],$upload_file_name) or error('Error
occured in loading file to recepient directory',$upload_form);

$fileName = $upload_dir.$file_number.$fext;//$dir.$file_number.$fext;

//third party GNU Licence class responsible for face detection

 $detector = new FaceDetector();

 $detector->scan("$fileName");

 $faces = $detector->getFaces();

 $result = "";

 foreach($faces as $face)

 {

       $result = "{$face['x']}{$face['y']}{$face['width']}{$face['height']}";

       $result = str_replace(".","",$result);

       if(strcasecmp('',$result) == 0){

               echo "<div align=\"center\">Sorry, could not detect face.<br/>Please try again
later</div>";

       }else{

               //echo "<div align=\"center\">$result</div>";

               //store details

               $user_pin = $_SESSION['userPIN'];

               $money = $_POST['money_alias'];

               $query = "SELECT money_number FROM money_table WHERE user_pin =
'".$user_pin."' AND alias_name = '".$money."'";

               $queryResult = mysqli_query($db,$query);

               $money_code = "";
               while($records = mysqli_fetch_row($queryResult)){

                       $money_code = $records[0];

               }

               $query = "INSERT INTO face_money VALUES
('".$result."','".$money_code."','".$user_pin."','".$fileName."')";

               mysqli_query($db,$query);

               if(mysqli_affected_rows($db) == 1){

                       echo "<div align=\"center\">Face In Picture Successfully Aligned With
Money Identified By $money<br/><a href=\"$panel\">back to panel</a></div>";

               }else{

                       echo "<div align=\"center\">Failed To Align Face In Picture With Your
Money Identified As $money<br/><br/><a href=\"$panel\">back to panel</a></div>";

}

       }

 }

}
```

<u>**GatherGesture.pde**</u>

Developed with guidance from Mario Klingemann's [40] algorithm on blob detection based on the language, Processing, the GatherGesture module detects blobs generated from user movement and writes binary values representing the trace to a text file.

```
import processing.video.*;

import blobDetection.*;

import javax.swing.*;

import java.util.ArrayList;

import java.net.*;

import java.io.*;

import java.awt.*;

Capture cam;

BlobDetection theBlobDetection;

PImage img;

boolean newFrame=false;

int clickCount;

ArrayList<UserGestureCoordinates> gestureCoordinates;

String fileName;

void setup()

{

  gestureCoordinates = new ArrayList<UserGestureCoordinates>();

size(640, 480);

cam = new Capture(this, 640, 480);

cam.start();

  img = new PImage(80,60);

  theBlobDetection = new BlobDetection(img.width, img.height);
  theBlobDetection.setPosDiscrimination(true);

  theBlobDetection.setThreshold(0.2f); // will detect bright areas whose luminosity > 0.2f;

}

void captureEvent(Capture cam)

{

  cam.read();

  newFrame = true;

}

void mousePressed(){

  String binary = "";

  for(int i = 0;i < gestureCoordinates.size();i++){

    binary = binary+""+gestureCoordinates.get(i).getX()+""+gestureCoordinates.get(i).getX();

  }

  final String instr = new java.lang.String("STORE_ACTION_MONEY");

  final String pin = JOptionPane.showInputDialog("Enter Customer PIN:");

  fileName = JOptionPane.showInputDialog("Customer Full Name\n(Separate with '_'):");

  fileName = fileName.concat(".txt");

  try{

    File file = new File("C:/MoSHADi/ComparisonPictures/",fileName);

  java.io.FileOutputStream fileWriter = new java.io.FileOutputStream(file);

  char[] numbers = binary.toCharArray();

  int index = 0;

  while(index < numbers.length){

    fileWriter.write(numbers[index]);

    index++;

  }

  int totalActionPoints = 0;

  fileWriter.close();

      if(file.exists()){

      JOptionPane.showMessageDialog(null,"Customer File Logged. Please transfer to
Server.","Operation Successful",JOptionPane.INFORMATION_MESSAGE);

      System.exit(0);

      }else{

      JOptionPane.showMessageDialog(null,"Failed to log Customer File. Please try
again.","Operation Failed",JOptionPane.ERROR_MESSAGE);

      }

}catch(java.lang.Exception exc){

println(exc.getMessage());

}

}
```

```
void draw(){
  if (newFrame)
  {
    newFrame=false;
    image(cam,0,0,width,height);
    img.copy(cam, 0, 0, cam.width, cam.height,
        0, 0, img.width, img.height);
    fastblur(img, 2);
    theBlobDetection.computeBlobs(img.pixels);
    drawBlobsAndEdges(true,true);
  }
}
void drawBlobsAndEdges(boolean drawBlobs, boolean drawEdges)
{
  noFill();
  Blob b;
  EdgeVertex eA,eB;
    b=theBlobDetection.getBlob(1);//n);
    if (b!=null)
    {
      if (drawBlobs)
      {
        strokeWeight(5);
        stroke(0,0,255);
        rect(b.xMin*width,b.yMin*height,
          b.w*width,b.h*height
          );
          gestureCoordinates.add(new UserGestureCoordinates(1,1));
        }
    }else{
      gestureCoordinates.add(new UserGestureCoordinates(0,0));
    }
}
void fastblur(PImage img,int radius)
{
 if (radius<1){
    return;
  }
  int w=img.width;
  int h=img.height;
  int wm=w-1;
  int hm=h-1;
  int wh=w*h;
  int div=radius+radius+1;
  int r[]=new int[wh];
  int g[]=new int[wh];
  int b[]=new int[wh];
  int rsum,gsum,bsum,x,y,i,p,p1,p2,yp,yi,yw;
  int vmin[] = new int[max(w,h)];
  int vmax[] = new int[max(w,h)];
  int[] pix=img.pixels;
  int dv[]=new int[256*div];
  for (i=0;i<256*div;i++){
    dv[i]=(i/div);
  }
  yw=yi=0;
  for (y=0;y<h;y++){
    rsum=gsum=bsum=0;
    for(i=-radius;i<=radius;i++){
      p=pix[yi+min(wm,max(i,0))];
      rsum+=(p & 0xff0000)>>16;
      gsum+=(p & 0x00ff00)>>8;
      bsum+= p & 0x0000ff;
    }
    for (x=0;x<w;x++){
      r[yi]=dv[rsum];
      g[yi]=dv[gsum];
      b[yi]=dv[bsum];
      if(y==0){
        vmin[x]=min(x+radius+1,wm);
        vmax[x]=max(x-radius,0);
      }
      p1=pix[yw+vmin[x]];
      p2=pix[yw+vmax[x]];
      rsum+=((p1 & 0xff0000)-(p2 & 0xff0000))>>16;
      gsum+=((p1 & 0x00ff00)-(p2 & 0x00ff00))>>8;
      bsum+= (p1 & 0x0000ff)-(p2 & 0x0000ff);
      yi++;
```

```
      }
      yw+=w;
    }
    for (x=0;x<w;x++){
      rsum=gsum=bsum=0;
      yp=-radius*w;
      for(i=-radius;i<=radius;i++){
        yi=max(0,yp)+x;
        rsum+=r[yi];
        gsum+=g[yi];
        bsum+=b[yi];
        yp+=w;
      }
      yi=x;
      for (y=0;y<h;y++){
        pix[yi]=0xff000000 | (dv[rsum]<<16) | (dv[gsum]<<8) | dv[bsum];
        if(x==0){
          vmin[y]=min(y+radius+1,hm)*w;
          vmax[y]=max(y-radius,0)*w;
        }
        p1=x+vmin[y];
        p2=x+vmax[y];
        rsum+=r[p1]-r[p2];
        gsum+=g[p1]-g[p2];
        bsum+=b[p1]-b[p2];
        yi+=w;
      }
    }
  }
  public class UserGestureCoordinates implements java.io.Serializable{
   private int xCoordinate;
   private int yCoordinate;
    public UserGestureCoordinates(int userX,int userY){
      xCoordinate = userX;
      yCoordinate = userY;
    }
    public int getX(){
      return xCoordinate;
    }
    public void setx(int userX){
      xCoordinate = userX;
    }
    public int getY(int userY){
      yCoordinate = userY;
      return yCoordinate;
    }
  }
```

Mini Glossary

C:

Cryptography: Discipline or techniques employed in protecting integrity or secrecy of data through mechanisms of confusion and diffusion.

D:

Double Spending: Where electronic money is spent more than once.

H:

Hyperinflation: Rapid, uncontrollable kind of inflation.

G:

Gold Standard: The gold standard was a commitment by participating countries to fix the prices of their domestic currencies in terms of a specified amount of gold.

N:

NFC Chips: Stands for Near-Field Communication Chips; Devices equipped with low powered radio allowing them to actively engage in conversations in the presence of devices with the same technology.

R:

RFID: Stands for Radio Frequency Identification; Radio frequency identification is a wireless communication technology that lets computers read the identity of inexpensive electronic tags from a distance, without requiring a battery in the tags.

References

[1] J. Robertson, *History of Money from Its Origins to Our Time*, Autrement, 2007.

[2] History of Money, http://www.investopedia.com/articles/07/roots_of_money.asp
[3] M.D. Bordo, "The Concise Encyclopaedia of Economics, Gold Standard" http://www.econlib.org/library/Enc/GoldStandard.html

[4] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, 2009

[5] S. Inenaga, K. Oyama & H. Yasuura "*Towards Modeling Stored-value Electronic Money Systems"* in IPSJ Transactions on Mathematical Modelling and Its Applications Vol. 3, p. 107-116, 2010.

[6] M. Zahrës, *E-Money: Niche Market That Might Be Expanding,* Deutsche Bank AG, May 2011.

[7]] M. Al-Laham, H. Al-Tarawneh, N. Abdallat, "*Development of Electronic Money and Its Impact on the Central Bank Role and Monetary Policy*", *Issues in Informing Science and Information Technology,* pp. 339-349 Volume 6, 2009.

[8] C. Bassey,"*Digital Money in a Digitally Divided World: Nature Challenges and Prospects of ePayment Systems in Africa", *2008.

[9] C. George, A. E. Haxthausen, "*Specification, Proof, and Model Checking of the Mondex Electronic Purse using RAISE",* UNU-IIST Report No. 352, February, 2007.

[10] "Mondex International: Re-engineering Money" http://wings.buffalo.edu/academic/department/som/isinterface/is_syllabus/mondex/mondex.html

[11] I. Osipkov, E. Y. Vasserman, N. Hopper, Y. Kim, "*Combatting Double Spending Using Cooperative P2P Systems",* 27th International Conference on Distributed Computing Systems, 2007.

[12] L.M. Muriira, "*Near Field Communication (NFC) Technology: The Future Mobile Money Service for Kenya*", International Journal of Computing and ICT Research, Vol. 6, Issue 1, June 2012.

[13] S.D. Mainwaring, W. March, B. Maurer, *Lessons for Digital Money Design from Japan,* 2007.

[14] M. Li, "*Octopus: Making Everyday Life Easier*", October 2008.

[15] *Virtual Currency Schemes*, European Central Bank, 2012.

[16] J. William, T. Suri, R. Townsend, "*Monetary Theory and Electronic Money: Reflections on the Kenyan Experience*", Economic Quarterly 96.1, p.83-122, 2010.

[17] J. Davis, "*The Crypto-Currency: Bitcoin and its Mysterious Inventor*",The New Yorker, October 10, 2011.

[18] "Digital Currency: Opportunities for the Postal Service", U.S. Postal Service Office of Inspector General, Report No. RARC-WP-12-001, of October 3, 2011.

[19] "The Future of Money in a Mobi-Digital World", http://www.wfs.org/blogs/michael-lee/ , October 7, 2012.

[20] http://www.google.com/wallet

[21] "History of Automatic Teller Machines or ATM-Luther Simijan by Marie bellis" http://inventors.about.com/od/astartinventions/a/atm.htm

[22] J.A. Kroll, I. C. Davey, E.W. Felten, "*The Economics of Bitcoin Mining, or Bitcoin in the Presence of Adversaries*", Princeton University, 2013

[23] F. Karray, M. Alemzadeh, J. A. Saleh, M. N. Arab, "Human-Computer Interaction: Overview on State of the Art", *International Journal on Smart Sensing and Intelligent Systems*, Vol. 1, No. 1, March 2008

[24] V. Hinze-Hoare, "*Review and Analysis of Human-Computer Interaction (HCI) principles", *Southampton University, 2007

[25] A. Dix, J.Finlay, G.D. Abowd, R. Beale, *Human-Computer Interaction 3rd Edition,* Prentice-Hall: Europe, 2004

[26] A. Anuar, K.M. Saipullah, N.A. Ismail, Y. Soo, "OpenCV Based Real-Time Video Processing Using Android Smart Phone",*International Journal of Computer Technology and Electronics Engineering (IJCTEE)*, Vol. 1, Issue 3

[27] P. Viola, M. Jones, *Robust Real-Time Object Detection,* Second International Workshop on Statistical and Computational Theories of Vision – Modelling, Learning, Computing and Sampling, Vancouver Canada, July 13, 2001.

[28] J. Cho, S.Mirzaei, J. Oberg, R.Kastner, *FPGA-Based Face Detection Systems Using Haar Classifiers*, 2008.

[29] S.K. Gaikwad, B.W. Gawali, P. Yannawar, "A Review on Speech Recognition Technique", *International Journal of Computer Applications (0975-887)*, Vol. 10 No.3, November 2010.

[30] K.R. Aide-Zade, C. Ardil, A.M. Sharifova, "The Main Principles of Text-To-Speech Synthesis System",*International Journal of Information and Communication Engineering*, June,6 2010.

[31] http://tcts.fpms.ac.be/synthesis/

[32] Information Technology-Automatic Identification and Data Capture Techniques-Bar code symbology-QR Code, ISO/IEC 18004, June 15, 2000.

[33] N. Sharma, L. Perniu, R.F. Chong, A. Iyer, C. Nandan, A. Mitea, M. Nonvinkere, M. Danubianu, *Database Fundamentals*, November 2010.

[34] http://research.lumeta.com/ches/map

[35] A.V. Aho, M.S. Lam, R. Sethi, J.D. Ullman, *Compilers, Principles, Tools and Techniques*, Second Edition, Pearson Education, 1996.

[36] CMU Sphinx – Speech Recognition Toolkit http://cmusphinx.sourceforge.net/

[37] LiteCoin- Open Source Peer-to-Peer Digital Currency, https://litecoin.org/

[38] J. Feigenbaum, C. Papadimitriou, R. Sami, S. Shenker, *A BGP-Based Mechanism for Lowest-Cost Routing*, 2002.

[39] E. Hillebrand,"*The Global Distribution of Income in 2050*", World Development Vol. 36, No. 5, pp. 727-740, 2008.

[40] Mario Klingemann, Processing Algorithm: Super Fast Blur v1.1 http://incubator.quasimondo.com