Mobile Payments

Alok Menghrajani (amenghra@andrew.cmu.edu)

Boris Danev (bdanev@andrew.cmu.edu)

Chuck Szeto (cszeto@andrew.cmu.edu)

Danny Lam (dlam@andrew.cmu.edu)

Kenan Hastor (khastor@andrew.cmu.edu)

November 2002

Abstract

This research project was conducted by five Carnegie Mellon University students, as part of the "15-494 Web Commerce, Security and Privacy", a one semester course taught by Professor Norman Sadeh.

In recent years, mobile phones have become very popular; more recently phone operators have started investing a lot of money in mobile Internet. However, this new technology is yet to be fully exploited.

In this project, we covered the technical issues related to developing a complete payment system over the mobile Internet. Our main challenges were to overcome the standardization issues related to this emerging technology and to provide end-to-end security. We tried to scale down the fixed Internet environment into these small devices which have limited screen size, bandwidth and computation power.

In this report, we covered the historical and business related issues to mobile Internet and mobile payments in particular, as well as the technical issues we were faced with.

A demonstration of our working implementation using an actual phone was also presented as part of this research.

Although one of the security aspects had to be disabled for the demonstration, the results obtained were promising. We were able to get a working solution that utilized end-to-end security. At times, we felt like pionneers working with primitive tools and faced with lack of support from the development community.

Contents

Listings

"Ericsson estimates that by 2004 there will be around one billion users of mobile telephony and some 600 million mobile Internet subscribers worldwide. The most important thing that is needed to get all these consumers to start using mobile e-commerce is a standard, which makes it safe and easy to use."

Ahrenbring, Ericsson Mobile Communications.

1 Introduction

Over the past decade, mobile phones are becoming increasingly popular. Today, these devices are equipped with color and higher resolution screens (typically 100x80 pixels with 256 colors). Along with the main manufacturer of these devices such as Sony-Ericsson, Nokia, Motorolla, new players like Microsoft are coming into the scene.

Mobile Internet has been initiated by phone service providers who invested in upgrading their existing network in order to support faster connection speeds and connection-less oriented protocols. Companies such as Yahoo! or CNN have tried to take advantage of their fixed internet presence to target the mobile Internet audience; but have usually failed as showed by the Nielsen Norman Group survey. Users have usually complained about the slow connections, quality of data and bad sign posting. They usually didn't complain much about the small display, which proves the mobile phone as a viable solution. We are going to develop a system where the user overcomes the disadvantages of the mobile internet by having extra convenience and security. Other advantages of the mobile internet are locality based services or highly tailored services according to individual needs.

The challenges with supporting trusted transactions over the mobile devices is the lack of end-to-end security in the earlier protocols. Also, the low computational power of these devices prevents crytographic algorithms with large inputs from being run. On the other hand, some mobile devices have smart card slots, and can provide very high security.

2 Overview

2.1 Technology

2.1.1 WAP

Wireless Application Protocal (WAP) is the standard published by the WAP Forum in order to provide information access protocols to mobile devices. The normal TCP/IP protocols couldn't be scaled for the mobile usage due the special needs involved with users moving from one base station to another.

WAP introduces a WAP Gateway, that connects users with mobile devices to the fixed Internet.

The first version of the WAP (WAP 1.0) defines a security layer called WTLS (Wireless Transport Layer Security). This protocol has vulnerability at the gateway level, because the data must be decrypted and recrypted.

However, the second version of the WAP (WAP 2.0) addresses this problem by using a wireless profiled TCP and TLS (Transport Security Layer).

2.1.2 XHTML & WML

Authoring for the mobile devices is very different from the fixed Internet. The designers must always keep in mind the limited size of the display. They must also either write their pages in WML (Wireless Markup Language) which is a stack based language, or in XHTML which is a stripped down version of HTML.

2.1.3 SSL

Secure Socket Layer (SSL) is a protocol used to ensure end-to-end security in a client-server communication. It provides data encryption (using public and private keys), message integrity (using a message digest), and optional client authentication.

The creation, managment and communication of the public keys require a PKI (public key infrastructure).

2.1.4 WAPPUSH

A Push message is an unsolicited message sent by a server to a client. The device will receive the message, even if offline. The message format basically is an XML document, where all the information for the message is held. This document is encapsulated with header field content (either WML or HTML depending on which device is obtaining the push message) which contains configuration information about how the message shall be received and the settings of the receiving party.

2.2 Goals

We decided to accomplish four major objectives:

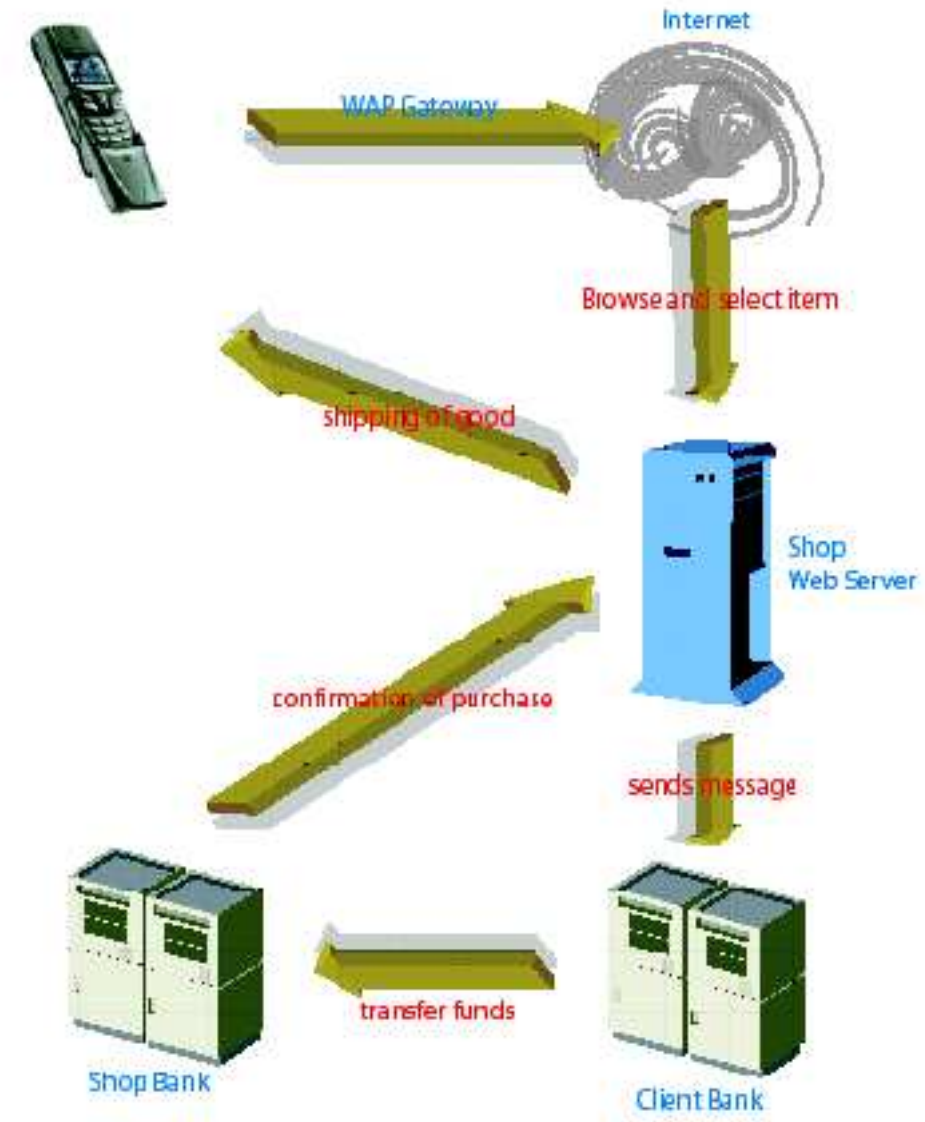
- Design a mobile payment solution
  The constraints were to design a fundamental system that could be implemented in a semester, yet have functionalities that are common with real life mobile payment solutions.

- Implement the mobile internet servers for our system and provide an easy user interface
  Getting each individual component of our system working independently.

- Handling security related issues
  Interconnecting all the components. Handling issues regarding key generation and server certification.

- Successfully demonstrate the working system on a real mobile phone

2.3 Design



Our design is inspired by the SET model for secure credit card transaction. We have improved the model by having the consumer acknowledge the transaction with his bank. This is the main reason why we can claim that our system is much more secure than the credit card. We have made the assumption that the bank and the vshop are going to be able to communicate with ease. In order to achieve this, we designed our own implementation for an InterServerMsg system.

These are our major components:

- Mobile phone
  We used a simulator at first, and then a real phone exclusively.
- WAP Gateway
  The mobile phone connects through this gateway. This gateway can also forward some extra information such as the caller-id.
- Vshop
  A virtual shop that can be browsed by a mobile phone.
- UserBank
  A virtual bank that stores the customer's account information, and that requests a confirmation for the transaction.
- VshopBank
  A virtual bank that stores the shop's account information, and tells the shop when the money is transferred.

A typical transaction scenario:

1. The user connects to the vshop using his mobile phone and browses the catalogue.
2. The user purchases an item and the vshop contacts the user's bank for payment.
3. The user confirms the transaction with his bank.
4. The UserBank transfers the funds to the VshopBank.
5. The VshopBank sends a clearence to the vshop.
6. The Vshop can proceed to the shipment of the item.

3 Mobile Payments: Standards

Currently, the mobile industry has several different standards. Each of them is trying to anticipate the mobile commerce future. European Committee for Banking Standards (ECBS) and the European Telecommunications Standards Institute (ETSI) have signed a co-operation agreement to increase their efforts with development of standards for the security of telecommunication and m-commerce. We can also see the Mobile Payment Forum, which is developed by the main credit cards (MasterCard, Visa, American Express), and by big mobile service providers (Vodafone and T-Mobile) and DoCoMo and Oracle. Their objectives are: "The Mobile Payment Forum is a global, cross-industry organization dedicated to developing a framework for standardized, secure, and authenticated mobile commerce using payment card accounts." (4)

We have also a MeT (Mobile electronic Transaction) forum which have the objectives: "to further strengthen the framework for secure mobile transactions - the ability to buy goods and services using a mobile device. MeT Limited Sponsors are Ericsson, NEC, Nokia, Panasonic, Siemens and Sony Ericsson and currently there are about 50 Associate." (2) This Forum whose members include the main players in M-Commerce for handset devices, show great efforts by trying to set universality for mobile transactions by creating a unique security standard.

Currently, MeT is co-working on mobile transaction standards with the Mobey Forum; a global organization with standardization efforts of mobile technology in financial services. These two organizations discuss banking and financial services industry issues, that will work with many other industry bodies to cover other aspects of mobile transaction.

Overall, we can conclude that there are many different standardization agencies and organizations that try to set global standards in mobile services. Every aspect has to be covered by establishing new standards, where security is the biggest priority issue.

4 Case Study

4.1 Dutch KPN and German E-Plus

In a joint venture, the Dutch KPN and German E-Plus mobile service provider agreed with the Japanese DoCoMo to introduce DoCoMo's *iMode* handset in Europe.

A month after the program was launched, German and Dutch customers were signing up to this European i-mode at a rate of over a thousand per day. Uwe Bergheim, the CEO of E-Plus, said that the number of users is within the expected range and that the company is happy with what it has achieved to date. (6)

Most i-mode handsets, such as the NEC N21i, are selling in Germany for less than the equivalent of USD 200, which is the standard price for phones with a value-added element. Bergheim believes that E-Plus' i-mode pricing is now pitched at such a level as to turn on customers with the basic charge priced to sell at USD 3 per month. (6)

But the real deal comes down to what customers themselves actually use, outside E-Plus direct charging. "The subscription prices are fixed by the content providers themselves" Bergheim explains. "Prices vary between 25 cents and USD 2 per month and anyone paying a content subscription has unlimited use of the service during that month." (6)

Some content is also free of charge, such as those offered by Deutsche Bahn, the German railway, Fleurop and UCI-World of Cinema. Sending an email costs a flat rate of 19 cents. "We are offering over 500 pages of information, services, entertainment and games, right down to complete timetable information from Deutsche Bahn railways. This is all provided in real time," Bergheim stands. Bergheim notes the number of content providers putting their faith in i-mode is growing rapidly. "Ultimately the simplicity of i-mode is winning almost everyone over." (6)

E-Plus' early market research shows that I-mode has convinced customers to sign up for the multimedia mobile offering in a way that WAP in its first generation never did. 80% say they are "happy" or "very happy" with the product, according to Bergheim, while 92% of i-mode subscribers are saying they would have no reservations to recommend the service to their friends. (6)

Many other market partners have also recognized the opportunities offered by mobile multimedia in combination with E-Plus i-mode. Content providers receive the lion's share of subscription revenue, a generous 86%, which Bergheim believes gives them a direct share in the financial success of I-mode. The remaining 14% of the subscription revenue is retained by E-Plus. The number of content services on the mobile portal has risen from 60 to 90 over recent weeks, with new providers including famous German brands such as ADAC, the Postbank and Aral. In the next few weeks, the one hundredth provider is expected to join, and the providers are expected to include popular German newspaper titles. (6)

The number of so-called free i-mode sites is also rising continually. Currently, there are some 700 private homepages registered with E-Plus' 'mobile-homes' service. These free pages are i-mode sites that anyone can write and publish on the internet, accessible from any i-mode handset by entering a normal internet web address. 'The straightforward business model of I-mode has great appeal for content providers says Bergheim, "because

they can address their customers quickly and under their familiar brand names. The content offerings are now easy to develop and to match exactly to mobile use." (6)

According to research by consultancy Jupiter/MMXI, already more money is spent on mobile content such as ring tones, logos, news and sport than content bought through the internet via personal computers. By 2006, Jupiter forecasts that European consumers will be spending in the region of USD 3 billion for content through their cellular phones, as compared with USD 1.7 billion spent on content via their laptop or desktop personal computers. (6)

Given such a steady start, Bergheim and his Dutch partners at both KPN and NTT DoCoMo could be stealing a lot of content from big wireless providers even before WAP 2.0 can take over the reins as a single unified wireless internet content standard. (6)

4.2 Sonera and Pizza Hut

The Finland based mobile service provider Sonera offers their mobile users through a Mobile Pay system purchasing goods from vending machines, parking meters and fast food restaurants. The users are then billed at the end on their mobile bill on the purchase they made. (7)

Currently in Helsinki City three Pizza Hut fast food restaurants are being tested and evaluated. Basically Pizza Hut offers the consumers to pay their meals in restaurants with their mobile cell phone. (7)

The process is straight forward where at the restaurant the customer makes a call or sends an SMS message to a service number that is given to him together with the menu to activate the mobile payment. The consumer than will immediately receive a confirmation code back by SMS. After finishing his meal the customer can use his confirmation code for payment. To settle the payment, the cashier checks the customers code and the amount to be charged. The customer will receive an SMS as a receipt. The customer will be charged for all his mobile purchases according to his choice by credit card or as a direct debit. (7)

In order to use this system, the customer has to register for the Sonera Mobile Pay service, and the customer data is maintained in Sonera Mobile Pay's payment server. The merchant uses a system to connect to an open interface to Mobile Pay's payment server. Then when the merchant's system sends a request for payment to the payment server, the customer data and the payment transaction data are checked, the transaction is either approved of or rejected, and the data is returned to the merchant's system. This system works according to the same principle as approving of a credit card payment in a credit card company. (7)

Overall, the Sonera Mobile Pay system offers payment solutions for many different kinds of services by mobile devices. All the interface solutions of Mobile Pay are open, so that different solutions of different suppliers could be implemented. For example merchant can implement the charged service or sale channel deployed by a mobile phone in such a way that Sonera charges a costumer for the products and transfers money to merchants account.

4.3 MobilPay

The Austrian German company mobilpay.com offers a payment system via mobile device that reduces the risk of payment via the Internet to near zero, they claim. The system started on June 30, 2000. (8)

The system is simple, because the registered merchants offer on their websites "mobilpay" as a payment option. If an (also registered) customer agrees to pay per "mobilpay," his or her mobile phone rings within seconds requesting him to confirm the order via a PIN. After the customer returns the SMS with the PIN, "mobilpay" gives the okay. Then the amount is deducted from the customer's account. The merchant is ready to deliver. (8)

At the onset, the fact that merchants and customers need to register with mobilpay.com seems to limit the company's potential for growth. Then again, PayPal, who has the same requirement, is used by over 30% of eBay's auction sellers, so there may a market for this type of service. (8)

MobilPay incomes comes from commissions, which indicates that more users would generate more revenue and secure MobilPay to survive the future. (8)

4.4 Conclusion

After carefully viewing our cases and our survey, we came up with a good business model for telecom companies that could enable them to get a large share of the market. Overall, they have the following features:

- serve simply as a carrier providing data transportation services for customers
- increase existing involvement in support services (billing, hosting, dispute resolution, customer support, etc.)
- move on to other services such as location-based services or financial services (including general payment services)
- produce content (such as news) themselves
- become simply a network service provider without own infrastructure

Since pure transport of data is likely to become a commodity business with low margins, this option does not look very promising. Moving further into customer services promises much more higher margins. Customer Relations Management (CRM) may be the key to profitability. However, competitors from other sectors are also well placed to take over part of the value chain:

- portals
- ISPs
- financial services industry

- content industry

Some of these players might even consider a go-it-alone strategy. In this respect, the success of NTT DoCoMo's i-mode looks encouraging for other telecom companies. I-mode has launched a very successful packet-switched mobile service. It has created its own portal and offers billing services to the merchants who use the portal. However, telecom companies are not the only organizations which are integrating vertically. Banks like Merita Nordbanken have already moved into hosting and building portals. Once a fully developed wholesale market for 3G capacity has evolved, they could even expand into mobile telephony (as Virtual Mobile Network Operators). However, given the particular strengths of each group of players, the formation of strategic alliances is still an option worth considering.

5 Characteristics

5.1 Security

We tried to create a self certified certificate, that would act as our CA (Certification Authority), but we weren't able to find any tool to sign a given certificate. We found a makeshift using *keytool*.

5.1.1 Keytool

Keytool is a tool distributed with Java. It allows the user to create keys and certificates, as well as a keystore (to store the keys) and truststore (to store certificates the user trust).

We created three certificates (one for the vshop, one for the VshopBank and one for the UserBank), and then put each others' certificate in the truststore. We used 1024 bits RSA encryption.

Here are the commands we used:

Example of key creation and certificate generation:

```
% keytool -genkey -alias vshop_cert -keyalg RSA -validity 30 -keystore vshop_keystore
Enter keystore password: webcom
What is your first and last name?
    [Unknown]: Shop owner
What is the name of your organizational unit?
    [Unknown]: Vshop
What is the name of your organization?
    [Unknown]: Vshop
What is the name of your City or Locality?
    [Unknown]: Pittsburgh
What is the name of your State or Province?
    [Unknown]: PA
What is the two-letter country code for this unit?
    [Unknown]: US
Is CN=Mobile Payment, OU=Carnegie Mellon, O=webcom, L=Pittsburgh, ST=PA,
C=US correct?
    [no]: yes
Enter key password for <vshop_cert>
        (RETURN if same as keystore password): <CR>

% keytool -export -alias vshop_cert -keystore vshop_keystore -rfc -file vshop_cert.cer
Enter keystore password: webcom
Certificate stored in file <vshop_cert.cer>
```

Example of truststore insertion:

```
% keytool -import -alias userbank_cert -file userbank_cert.cer -keystore vshop_truststore
Enter keystore password: webcom
Owner: CN=User Banker, OU=UserBank, O=UserBank, L=Pittsburgh, ST=PA, C=US
Issuer: CN=User Banker, OU=UserBank, O=UserBank, L=Pittsburgh, ST=PA, C=US
Serial number: 3de7f214
Valid from: Fri Nov 29 18:02:44 EST 2002 until: Sun Dec 29 18:02:44 EST 2002
Certificate fingerprints:
        MD5: 11:95:96:7F:B0:96:40:BC:AD:88:E9:75:84:97:2D:CF
        SHA1: 42:2D:BC:EF:F4:E1:10:05:11:6A:A7:46:45:28:F3:CD:D4:1F:C3:CD
Trust this certificate? [no]: yes
Certificate was added to keystore

% keytool -import -alias vshopbank_cert -file vshopbank_cert.cer -keystore vshop_truststore
Enter keystore password: webcom
Owner: CN=Vshop Banker, OU=VshopBank, O=VshopBank, L=Pittsburgh, ST=PA,
C=US
Issuer: CN=Vshop Banker, OU=VshopBank, O=VshopBank, L=Pittsburgh, ST=PA,
C=US
Serial number: 3de7f6b6
Valid from: Fri Nov 29 18:22:30 EST 2002 until: Sun Dec 29 18:22:30 EST 2002
Certificate fingerprints:
        MD5: 3B:8D:C8:71:4A:E1:4C:5D:7F:CE:9F:7D:5C:67:BE:E8
        SHA1: 95:B8:AB:39:4C:9B:52:E5:8F:FE:F7:FE:1B:65:F3:20:5B:50:C9:A8
Trust this certificate? [no]: yes
Certificate was added to keystore
```

The truststore can be viewed using the % keytool -list -keystore vshop_truststore command.

5.1.2 Openssl

Openssl is a tool we used to create a certificate signing request (CSR), that we sent to verisign, in order to get a free verisigned certificate. This unfortunatly didn't work. We couldn't get the free certificate to work with Apache, and a self-certified certificate is not recognized by the mobile device (see §5.3.3)

Creating the CSR:

```
% openssl req -config openssl.cnf -new -out my-server.csr
Using configuration from openssl.cnf
Generating a 1024 bit RSA private key
.....................................................................
++++++
.....................++++++
writing new private key to 'privkey.pem'
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
---
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
---
Country Name (2 letter code) :US
State or Province Name (full name) :Pennsylvania
Locality Name (eg, city) :Pittsburgh
Organization Name (eg, company) :Webcom
Organizational Unit Name (eg, section) :Webcom
Common Name (eg, your websites domain name) :alok
Email Address :amenghra@andrew.cmu.edu

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password :webcom
```

We then submitted this file to verisign. Since it didn't work, we then created a self-signed certificate.

```
% openssl rsa -in privkey.pem -out my-server.key
read RSA key
Enter PEM pass phrase:
writing RSA key

% openssl x509 -in my-server.csr -out my-server.cert -req -signkey my-server.key -
days 365
Signature ok
subject=/C=US/ST=Pennsylvania/L=Pittsburgh/O=Webcom/OU=Webcom/CN=alok
/Email=amenghra@andrew.cmu.edu
Getting Private key
```

5.2 Simulators

We tried several simulators, but all turned out to have bugs or problems in specific areas. We were pretty disapointed by these simulator, so we finally work with a real phone only.

5.2.1 Nokia

The Nokia toolkit is one of the best we found. Unfortunately, none of the new phones were available. The toolkit requires backgrading to Java 1.3, the wappush didn't work, the product seemed rushed, the documentation is poor, there are version conflicts and developer support is poor.

5.2.2 Openwave

Openwave's toolkit is very simple. There is no way to accept certificates. The only use we found for this tool is to create screenshots.

5.2.3 Ericsson

This simulator doesn't have any phone that supports xhtml at all. They have a wappush though, but it isn't free.

5.3 Installation: General Overview

For this mobile-payment project, a multi-platform/multi-tiered design utilizing a variety of technologies was decided in order to test the full functionality and scope that the project could go onto. With this in mind, a wide variety of software tools has been used and tried to make the mobile-payment project be a success. All these tools can be found on the demo CD. The demo simulation will be run under Windows XP. Also, a cvs root was installed in order to be able to share files and merge the team's efforts on this project report.

1.) MOBILE PHONE SDK's:

A variety of mobile phone software development kits (SDK's) were downloaded and used in order to figure out which SDK functionality would be the most beneficial for the project. Due to problems faced using these applications(*as discussed in the Problems section of this report*), the **Nokia Development Toolkit 3.1** was chosen due to ease of use and practicality with WAP functionality. This kit came with a "perfect" virtual phone (one that's doesn't have the same keyboard, memory and display constaints as a real phone). The download could be found in (**http://portals.devx.com/Nokia/WidgetContainer/6573**) and is self-explanatory.

Other mobile phone SDK's tried were the **Sony Ericsson WAP Emulator** in (**http://www.ericsson.com/mobilityworld/sub/open/technologies/wap/tools.htm** and the OpenWave SDK in

(**http://developer.openwave.com/download/index.html**). Both downloads are also self-explanatory to the user.

2.) SIMULATIONS:

A list of current and latest software that is being used for the virtual banks (*vbanks*) and virtual shop (*vshop*) of this project will later be mentioned in this section. For the virtual banks and shop, the SSL code was implemented in Java, using the classes found in **JDK 1.4** (with **JSSE 1.0.2** capability) API downloaded from **http://sun.java.com**. Keeping the multi-platform/multi-tiered design in mind, the *vbanks* and *vshop* are implemented differently from each other. The *vbanks* are using a combination of JSP and Java Servlet functionality on an Apache Tomcat Server. However, the *vshop* is using PHP within an Apache Server.

5.4 VShop

5.4.1 Description

The *Vshop* is an example of a merchant that is selling goods through a mobile device. Due to the constain of the small display, the vshop has a very simple catalogue that can be easily browsed.

The user enters the vshop by entering his phone number and password. In order to enter the password quickly, the user should use digits only (but is not obliged to). If the user's gateway is compatible with our scripts (which can be improved to support most real wap gateways), the phone number is automatically entered.

The user can then browse a catalogue of products (for an example, we filled the database with video games and music cds). The user then selects a product and is sent to the bank to confirm the purchase.

The vshop is powered by an Apache Secure webserver. The dynamic content is generated using PHP and a MySql database. The content is in xhtml format, which is very similar to html, but adds constrains such as having to close all the tags and write proper html header tags. The php files are present in the annexe.

We also installed phpMyAdmin, a set of php files to create and manage mySql databases. The vshop communicates with the vbanks using secure sockets and java objects. (PHP is an HTML-embedded scripting language, it stands for PHP: Hypertext

Preprocessor (Many people are confused by acronyms like this one that are recursive acronym)).

MySql is a free implementation of the Sql database query language.

5.4.2 Database structure

Here is a detailed description of the database structure and the way it is used in the vshop. For the complete list of SQL queries used to generate these tables, refer to the annexe.

User information

phoneNo	userPwd	gateway IP	gateway Port	name	autologin	bankIP	accountNo	address

The user information is stored in the *users* table. It is accessed during authentification and when sending a purchase message to the server. The gatewayIP and gatewapPort fields were designed for extra security, so that the phoneNo is bound with the gatewayPort. The autologin is a feature we designed but that represents potential security problems (such as in the case of tefth) and should be implented in a real world scenario with extreme care.

It is very very important that these fields are verified not to contain any potentialy dangerous values (such as shell pipes) when entered in the database. Some of these values are passed to the SSLUserPurchase program via a command line shell. If the name is a string such as b̈ob — rm -r
ëntire directories can be erased. Had we gotten more time, we would have implemented proper string checks in the php script (buy.php) that calls exec.

Catalogue

productID	category	name	price	description

category	father

The pages generated for the catalogue are based on two tables, *products* and *categories*. The first one gives the actual information about each product (the name, price, description, etc...), while the second table gives the tree structure of the catalogue, in order to make naviguation easier.

Transactions

transactionID	productID	price	userID	purchase (date)	shipment (date)

transactionID	transactionID

Once the User decides to buy an item, an entry in the *transactions* table is created. The productID and price as saved, since the price might change in the future. The purchase date is set to the current date. The shipment date is set to zeros (MySql allows a date to be all zeros, which means year 0000, month 0, day 0, time 00:00:00, which doesn't make sense but is often practicle). In case of an error, there will be an entry in the transaction table, but not in the waiting table. We haven't decided on any policy in case of error. Example of actions that could be taken are send email to owner of shop and vbank, save transaction in a special table and retry later, delete transaction and try creating a new one, etc... If the connection with the bank succeeds, the transactionID is also added to the *waiting* table, which is a queue of items waiting to be confirmed. Once the confirmation is received, the transactionID is removed from the *waiting* table and added to the toship table. No further action is taken, but we could imagine automating the shipment action or taking some other action.

5.4.3 Installation

I. mySQL 3.23.53
(Ref: http://www.mysql.com)

MySQL is anOpen Source Database, which is designed for speed, power and precision in heavy-load and critical usage. This sql database is used by the vshop to generate the catalogue, track what items have been sold, and manage user registration and personnal information. The download for the binary distribution of this program can be found at **www.mysql.com** (*mysql_3.23.53-win.zip.*)

1. Extract the zip file to directory of choice (***C:/mySQL*** for example)

2. Click on the **setup.exe** executable and follow the installation instructions.
   (**Note:** All *Vshop* code was moved to **"mysql/bin"** directory)

II. Apache 1.3.27
(Ref: http://httpd.apache.org/docs/windows.html,

http://httpd.apache.org/docs/conf.html)

Apache is an HTTP server, originally designed for Unix systems. This version of Apache was designed for use with Microsoft Windows XP, 2000, NT, 98, and 95 systems (Win32). It includes many frequently requested new features, and has an API that allows it to be extended to meet users' needs more easily. The download for the binary distribution of this program can be found at **www.apache.org** (*apache_1.3.27-win32-x86-no_src.zip*).

Installation for Windows:

Once the Apache Program is unzipped, run the install executable and the setup process will prompt for:

1. Whether or not to run Apache for all users (installing Apache as a service), or if in a console window when you choose the Start Apache shortcut.
2. User's Server name, Domain name and administrative email account.
3. The directory to install Apache into (the default is "**C:\Apache Group\Apache**", although this can be changed to any other directory desired)
4. The installation type. The "Complete" option installs everything while the "Custom" install gives the option to choose not to install the documentation, or the source code.

During the installation, Apache will configure the files in the **conf** directory for the chosen installation directory. However if any of the files in this directory already exist they will ***not*** be overwritten. Instead the new copy of the corresponding file will be left with the extension *.default.conf*.

Also, if you already have a file called "***htdocs\index.html***" then it will not be overwritten (no "***index.html.default***" file will be installed either). This should mean it is safe to install Apache over an existing installation (but the existing server running needs to be stopped before doing the installation, and the new one will need to be started after the installation is finished).

After installing Apache, edit the configuration files in the **conf** directory as required. These files will be configured during the install ready for Apache to be run from the directory where it was installed, with the documents served from the subdirectory **htdocs**. There are lots of other options which should be set before you start really using Apache. However to get started quickly, the files should work as installed.

If the user wants to uninstall Apache, the configuration and log files will not be removed. The installation directory needs to be manually deleted if the value of the configuration and other web files in it are not important.

Configuration for Windows:
(Ref: http://www.modssl.org/contrib/)

For the scope of this project, SSL functionality will need to be installed in the Apache server for the interconnections between the *vshop* and the *vbank*. For ***Mod_ssl***

1. Go to website "**http://www.modssl.org/contrib**", and download *Apache_X-mod_ssl_Y-openssl_Z-WIN32[-i386].zip*
2. Unzip to a new directory. Copy the files "*ssleay32.dll*" and "*libeay32.dll*" from the **Apache/modssl** distribution directory to the Window's System32 directory.
3. Additionally, a config file for "*OpenSSL.exe*" will be needed. This can be found in "***http://tud.at/programm/openssl.cnf***". Copy it to the directory "*openssl.exe*" is in and overwrite the file.

The main configuration file for the Apache server, and for the scope of this project, is "*httpd.conf*" (found in the **conf** directory in the directory of your Apache installation folder). The "*httpd.conf*" file needs to be modified to support php with xhtml files. The lines for this file that ***NEED*** to be changed for this project, in sequential order, are:

"Section 1: Global Environment":

1. **ServerRoot** ***location*** where ***location*** is the location of the Apache installation directory where all the files are. (**eg:** ServerRoot "**C:\Apache_Group\Apache**")
2. Port 80 → #Port 80 (Comment it out; *Port* is not necessary). Add in *Listen 80*.
3. Add in *Listen 443* (443 is the socket that SSL connections are used).
4. Add in the line: *LoadModule ssl_module modules/mod_ssl.so* after all the other *LoadModule* lines in the "*httpd.conf*" file.
5. Add in the line: *AddModule mod_ssl.c* after all the other *AddModule* lines in the "*httpd.conf*" file.

"Section 2: Main server configuration":

1. ServerAdmin * → where * is the default email address, otherwise leave blank. (**eg:** ServerAdmin **dlam@andrew.cmu.edu**)
2. ServerName * → where * is the domain name or IP address of the localhost machine. (**eg**: ServerName **127.0.0.1**) (**Note:** For the *vshop*, the machine name host is ***alok.wv.cc.cmu.edu***. For the *vbank*, the machine name host is***urlin.wv.cc.cmu.edu***. Both machines are registered at the CMU wireless network which can be done at ***"www.cmu.edu/myandrew/netreg***").

3. DocumentRoot "*"→ where * is the directory where the Apache server is serving the user's documents. (**eg:** DocumentRoot "**C:\Apache_Group\Apache\htdocs**")

Add the following to the end of "*httpd.conf*":

```
 # see http://www.modssl.org/docs/2.4/ssl_reference.html for more info
SSLMutex sem
SSLRandomSeed startup builtin
SSLSessionCache none

SSLLog logs/SSL.log
SSLLogLevel info
# You can later change "info" to ''warn'' if everything is OK

<VirtualHost www.my-server.dom:443>
SSLEngine On
SSLCertificateFile conf/ssl/my-server.cert
SSLCertificateKeyFile conf/ssl/my-server.key
</VirtualHost>
```

where "**www.my-server.dom**" is the name of the machine host for Apache.

Once these primary configurations are set, restart the Apache server again and the program should reset these new settings.

Testing for Windows:

If there is trouble starting the Apache server, either as a service or a console application, use these methods to isolate the problem.

Run the "*Command Prompt*" from the Start Menu on the Programs list. Change to the folder to which you installed Apache and type the command ***apache***. Read the error message and review the "***error.log***" file for configuration mistakes. If the defaults were accepted during installation, the commands would be:

```
 >c:
> cd ''\Apache_group\Apache''
> apache
Wait for Apache to exit, or press Ctrl+C
> more <logs\error.log
```

The **error.log** would give a good indication of where and what the configuration mistakes are in order to be fixed.

After the Apache server is started and running (either in a console window or as a service), the default port that it will be set to is 80 (the port address can be changed in the Apache configuration files). To connect to the server and access the default page, launch a browser and enter this URL: *http://localhost/* or *http://127.0.0.1/*.

This should respond with a welcome page, and a link to the Apache manual. If nothing happens or you get an error, look in the "**error.log**" file in logs the directory. Once your basic installation is working, you should configure it properly by editing the files in the **conf** directory. Because Apache *can't* share the same port with another TCP/IP application, certain services may need to be stopped or uninstalled first.

III. PHP 4.2.3
(Ref: www.php.net)

PHP is a widely-used general-purpose scripting language that is suited for Web development and can be embedded into HTML.

This is a layer that comes on top of apache web server that allows server side scripting. From the PHP homepage (**www.php.net**), download the PHP binary distribution (*php-4.2.3-Win32.zip*). Extract the distribution files to a directory (**C:\PHP** for example). For the purpose of the project, the installation was done not using an executable, but instead configuring the installation files manually in order to guaranteed the full functionality of Apache server with PHP. Refer to the "*install.txt*" document from the PHP installation for further details.

PHP Installation:

Copy the "*php.ini-dist*" file from the installation to the **%SYSTEMROOT%** directory under Windows NT, Windows 2000 or Windows XP and rename it to "*php.ini*". The **%SYSTEMROOT%** directory is typically:
`C:\winnt` or `C:\winnt40` for NT/2000/XP servers

Editing "*php.ini*" for the Apache Server:

1. In the "*php.ini*" file, set the **doc_root** to point to the default Apache document directory (**eg:** doc_root = **C:/Apache_Group/Apache/htdocs**).

2. Add these lines to the "*httpd.conf*"file in the Apache server to their respective place:

```
LoadModule php4_module c:/php/sapi/php4apache.dllAddModule
mod_php4.cAddType application/x-httpd-php .php
```

   (a) Note: Add *EACH* line in sequential order in their respective place. **eg:** (* indicates that these lines are already found in the "*httpd.conf*" file during

Apache installation) In the actual "*httpd.conf*" file, (*Section 1: Global Environment*), the user will find lines like this:

*** LoadModule ...**
LoadModule ...* → end of already set Apache server settings

The user will add: **LoadModule php4_module c:/php/sapi/php4apache.dll** right after the already set Apache server settings. The same concept applies for the other two respective lines (**AddModule** and **AddType**).

(b) "*c:/php/sapi/php4apache.dll*" in the *LoadModule* line should be the default directory of where the "*php4apache.dll*" file is kept.

3. For ***saving sessions*** in php with Apache,

   (a) Set the "**session.save_path**" line in "*php.ini* to a default temp directory (**C:/temp** for example). **eg:** session.save_path = "**C:/temp**"

   (b) Set the "**register_globals**" variable in "*php.ini*" to **ON eg:** register_globals = **ON**

Testing PHP on Apache Server:

1. Once editing is done, restart the Apache server. In the **htdocs** directory of your Apache installation directory, open a text file and name it *phpinfo.php*.

2. In this file, put in: **<? Phpinfo() ?>** and save it.

3. Open a Web browser and type **http://localhost/phpinfo.php**

The output of the "*phpinfo.php*" script should be a long page full of system and environment information. (**Note:** For the Apache server to detect and run php script pages, they all need to be placed in the **htdocs** directory of the Apache installation directory.)

IV. phpMyAdmin 2.3.3
(Ref: www.php.net)

phpMyAdmin is intended to handle the administration of MySQL over the web, using built in php scripts found in the installation zip file (*phpMyAdmin-2.3.3-rc1-php.zip*).

1. Extract files to the **htdocs** directory of the Apache installation directory (files should be a series of php pages)

2. Modify "*config.inc.php*"Set on line 73:
   `$cfg['Servers'][$i]['password'] = '';` → `\cfg['Servers'][$i]['password'] = 'root';`
   This will detect that the password will be "**root**", or whatever the user wishes the password to be for this installation.

3. Set the following environment variable **CLASSPATH**:(In windows, go to → **My Computer-(right click)-properties-Advanced-Environment Variables-(add)**) Set to:
   `CLASSPATH = .;c:\j2sdk1.4.1_01\jre\lib` (**Note:** the . is very important!)

V. mySQL Connector/J 2.0.14
(Ref: http://www.mysql.com)

A JDBC driver for Java to function along with mySQL database. The download for the binary distribution of this program can be found at *www.mysql.com* (*mysql-connector-java-2.0.14.zip).*

1. Extract zip file to directory of choice (**C:/jconnector** for example)

2. Set the following environment variable **CLASSPATH**:(In windows, go to → **My Computer-(right click)-properties-Advanced-Environment Variables-(add)**) Set to:
   `CLASSPATH = .;c:\j2sdk1.4.1_01\jre\com` (**Note:** the . is very important!)

5.4.4 Testing efforts

Since the vshop was the first working entity we had, it has been intensivly tested. The vshop was also used to make sure that the interconnections code was working.

We simulated network latency (by adding sleeps in the java server code), and besides php that requires a timeout for the script (currently set to 30 seconds) there are no problems with the risk of forwarding the user to the bank's page before the purchase message arrives.

Our tests also focused on data transmission, connection concurrency, database queries. Due to the number of such tests and the evolution of our system over time, it is impossible for us to write a detailed report. We didn't encounter any major problems, and we fixed all minor problems.

A few test files are available in the vshop/tests directory.

5.4.5 Limitations and Possible Improvements

The vshop is very simple and can be improved in many places. For example, a stock tracking system could be included.

Users are currently directly registered in the database. User registration and user profile pages must be created.

Currently transactions that are not confirmed are left in the waiting forever. An improvment would be to support the broadcast and handling of a reject message. A timeout should perhaps also be used for messages not confirmed after a certain number of days. Here the action to take might depend on what the show owner wants to do.

5.5 Virtual Banks (Using Red Hat Linux 8.0)

5.5.1 Description

In order to make a fully realistic mobile payment system, we developped two virtual banks:

- User Bank
- Vshop Bank

The User Bank is the the bank of the client who will browse the Vshop Web Site with his personal trusted device.It offers two special services:

- Secure Web Portal
- Secure Mobile Payment Confirmation Portal

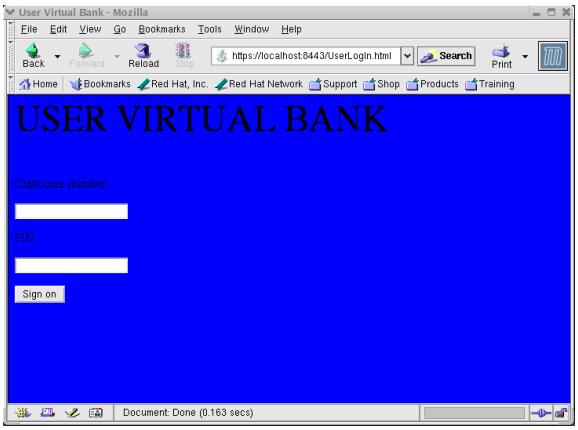
In order to make a fully realistic mobile payment system, we developped two virtual banks:

- User Bank (represents the client side)
- Vshop Bank (represents the vshop side)

The User Bank is the the bank of the client who will browse the Vshop Web Site with his personal trusted device.It offers two special services:

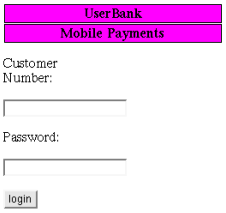
- Secure Web Portal
- Secure Mobile Payment Confirmation Portal

The Secure Web Portal was done with the only purpose to test the correctness of the mobile payment system (client side).It provides a web portal where the client can log in and check all his transactions.Once he finished , he can securely log out.The client can navigate through the portal web pages thanks to sessions.The portal uses SSL connection to the User Bank Web server in order to provide maximum security.



The Secure Mobile Payment Confirmation Portal operates only on the personal trusted

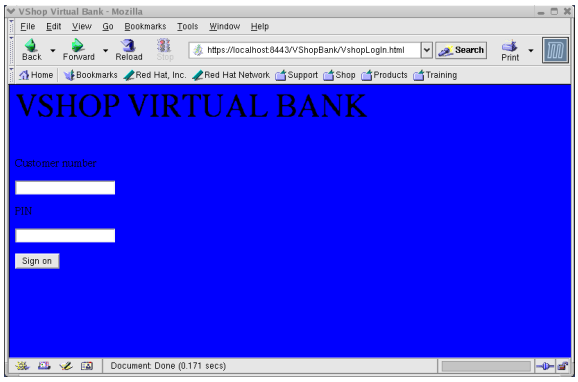
device(PTD) of the client. It provides a small user friendly interface written in XHTML , where the client after having made a purchase on the vshop , can confirm it or not. If he confirms his transaction, the Mobile Departement will take care to withdraw the amount of the purchase from the client's account and then send it to the Vshop Bank using secure SSL connection.



The Vshop Bank provides only two special services:

- Secure Web Portal
- Confirmation Notification

The Secure Web Portal was done with the only purpose to test the correctness of the mobile payment system (vshop side).The Vshop Secure Web Portal provides a simple interface where the an authorized representative of the Vshop can check its account and transactions. The Vshop Bank has also a Confirmation Notification service which consists in sending a confirmation message to the Vshop that it received the amount of money for a given transaction. Transactions have unique ids and the Vshop knows exacly which client has made a purchase. After having received the Confirmation Notification, the Vshop can proceed shipping the product.



5.5.2 Design

The User Bank has two major design components along with a database system:
User Bank Server and User Bank Web Server. The User Bank Server deals with receiving transaction messages from outside, distinguish between different types of transactions and stores the information in a particular place of the its database system , depending on the

type of message received. In our case , the User Bank can recognize only one type of message and it is the SSLUserPurchaseMsg Object , which is sent when a client of the User Bank has made a purchase with his personal trusted device. All other messages are ignored. Important point about receiving the SSLUserPurchaseMsg is that the User Bank has in its certificate truststore the certificate of Vshop , which enables the User Bank to perform authentication of the message (see Section Security for information about the recognized certificates of the User Bank)
The User Bank Web Server deals with all the web based part of the User Bank.It gives a web access to client's account and possibilty to confirm a purchase. One important role of the User Bank Web Server is that it is in charge of sending the SSLBankPaymentMsg upon confirmation. The SSLBankPaymentMsg Object contains all the information needed by the Vshop Bank in order to update the Vshop account. An SSLBankPaymentMsg is sent to the Vshop Bank if and only one the client who has made a purchase on the Vshop confirms it.
The Database system of the User Bank contains the followng SQL tables:

- table USERUSERS (special purpose : authentification when log in)
- table USERACCOUNT (purpose: all account information of a given client)
- table USERTRANSACTIONS(purpose: all the transactions of clients)
- table USERMOBILEPAYMENTS(special purpose: all mobile transactions are stored here waiting for confirmation)

The Vshop Bank has similar two major design components:Vshop Bank Server and Vshop Bank Web Server along with its database system.
The Vshop Bank Web Server has no special services a part from the standart service to retrieve account information upon log in. The Vshop Bank Server deals with receiving transaction messages from outside. In our case it can receive only one special message called the SSLBankPaymentMsg. all other messages are ignored. Once it receives an SSLBankPaymentMsg, the server proceeds in authentication checking its own certificate truststore (see Section Security for information about the recognized certificates of the Vshop Bank)If the authentication succeed the Vshop Bank Server updates the Vshop account using the information provided by the SSLBankPaymentMsg and then executes his special service Confirmation Notification, sending a SSLConfirmationMsg to the Vshop Server The Database system of the Vshop Bank is simpler and contains almost the same structure:

- table VSHOPUSERS (special purpose : authentification when log in)
- table VSHOPACCOUNT (purpose: all account information of a given vshop)
- table VSHOPRANSACTIONS(purpose: all the transactions of vshops)

For more details about the implementation of the User Bank and the Vshop Bank, please refer to the Appendix Section A.2 where we present a commented JSP/JAVA code of the User Bank and Vshop Bank and their SQL Database Systems

5.5.3 Installation

I. Tomcat 4.1.12
(Ref: http://jakarta.apache.org/tomcat/index.html)

Tomcat is the *servlet container* that is used in the official Reference Implementation for the Java Servlet 2.3 and JavaServer Pages 1.2 (JSP) technologies. This application is an extension/modification of the Apache server. The Java Servlet and JavaServer Pages specifications are developed by Sun under the Java Community Process.

1. **Setup in Linux**:
   (Ref: http://jakarta.apache.org/tomcat/tomcat-4.1-doc/RUNNING.txt)

   (a) Download a binary distribution of Tomcat from:
   http://jakarta.apache.org/builds/jakarta-tomcat-4.0/nightly/

   (b) On a Unix platform, the following distribution file will be needed:
   jakarta-tomcat-4.0-YYYYMMDD.zip

   (c) Unpack the binary distribution into a convenient location so that the distribution resides in its own directory ("**jakarta-tomcat-4.0**" for example).

   (d) Execute the following shell commands to start the Tomcat server:
   `>cd $HOME\bin (Unix) > ./startup.sh (Unix)`

   (e) Execute the following shell commands to shut the Tomcat server:
   `>cd $HOME/bin (Unix) > ./shutdown.sh (Unix)`
   where **$HOME** is the directory where the Tomcat server is installed.

2. **Testing:**

   (a) After startup, the default web applications included with this server will be available by browsing: *http://localhost:8080/*

3. **Configuring SSL connection:**

   (a) Create a certificate keystore by executing the following command:
   In Unix: `> $JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA`
   and specify a password.

   (b) Uncomment the "*SSL HTTP/1.1 Connector*" entry in:
   `$HOME/conf/server.xml` and tweak as necessary.
   (**Note:** $HOME is the directory of where the Tomcat installation is)

   (c) Include the *keystorePass* tag in the connector entry, because a different keystore and password is being used instead of the default one. **eg:** (In

server.xml)

```
 <-- Define an SSL HTTP/1.1 Connector on port 8443 --> <!-- <Connector
className="org.apache.catalina.connector.http.HttpConnector"
port="8443" minProcessors="5" maxProcessors="75" enableLookups="true"
keystorePass acceptCount="10" debug="0" scheme="https" secure="true">
<Factory className="org.apache.catalina.net.SSLServerSocketFactory"
clientAuth="false" protocol="TLS"/> </Connector> -->
```

4. **Disabling the firewall on Linux:**

   (a) There is a default firewall built in and activated with Red Hat Linux 8.0, which prohibits communication between applications from different mediums. In one scenario, the *vshop* could not communicate from one laptop to the *vbanks* on another laptop due to the firewall. To disengage the default firewall: *On Linux command prompt*: `>cd /etc/init.d/ >./iptables stop`

II. Oracle 8i Database
(Ref: http://www.oracle.com/ip/deploy/database/8i/index.html?content.html)

The Oracle8i database is used due to the power and ease for internet development and deployment. It is also compatible with many different platforms and applications, including Linux and Tomcat Apache server. All setup was done from the Oracle 8i installation CD (typical setup).

5.5.4 Security

Trusted certificates for the User Bank and the Vshop Bank

```
% keytool -list -keystore userbank_truststore
Enter keystore password: webcom

Keystore type: jks
Keystore provider: SUN

Your keystore contains 2 entries

vshopbank_cert, Nov 29, 2002, trustedCertEntry,
Certificate fingerprint (MD5): 3B:8D:C8:71:4A:E1:4C:5D:7F:CE:9F:7D:5C:67:BE:E8
vshop_cert, Nov 29, 2002, trustedCertEntry,
Certificate fingerprint (MD5): 0A:2D:27:96:CC:7D:9D:25:36:07:DB:6D:34:9E:77:6A
```

```
% keytool -list -keystore vshopbank_truststore
Enter keystore password: webcom

Keystore type: jks
Keystore provider: SUN

Your keystore contains 2 entries

userbank_cert, Nov 29, 2002, trustedCertEntry,
Certificate fingerprint (MD5): 11:95:96:7F:B0:96:40:BC:AD:88:E9:75:84:97:2D:CF
vshop_cert, Nov 29, 2002, trustedCertEntry,
Certificate fingerprint (MD5): 0A:2D:27:96:CC:7D:9D:25:36:07:DB:6D:34:9E:77:6A
```

For more information of how these certificates were generated, please refer to Section 5.4 Interconnections.

5.5.5 Testing efforts

In order to check the functionality of the virtual banks we proceeded in testing each design component separately and then put them together.
The testing efforts on the Web Servers aimed the folowing key points:

- *Correct establishment of a SSL connection*
- *Correct creation of a session for each particular user at log in time*
- *Correct invalidation of a session after log off*
- *Correct handling of database requests from Java Server Pages*
- *Correct execution of special tasks (eg. The User Bank Web Server has to send SSLBankPaymentMsg correctly to the Vshop Bank Server, if it is not the case it should signal error)*
- *Correct searching for java classes (eg. with Tomcat 4.1.12 all java classes used by Web Server needs to be in a folder WEB-INF/classes)*
- *Correct error handling*

The testing efforts on the User Bank and Vshop Bank Host Servers aimed the following key points:

- *Correct establishment of a listening port (15494 for User Bank Host Server and 15496 for the Vshop Bank Host Server)*
- *Correct creation of threads in order to be able to handle multiple requests from outside*

- *Correct sending of authentification certificate to other servers*

- *Correct receiving of authentication certificates and checking for trusted certificates in its truststore*

- *Correct execution of special tasks (eg. updating database system, sending special messages : Confirmation Notification with SSLConfirmationMsg for the Vshop Bank Host Server and SSLBankPaymentMsg for the User Bank Host Server)*

- *Correct error handling.(eg. The servers must never crash due to received incorrect messages)*

5.5.6 Limitations and Possible Improvments

The design of the above virtual banks is fully complete in terms of presenting our mobile payments system. However it presents some limitations especially in terms of handling errors and performing the special tasks already mentionned. Here are the main points:

1. The User Bank Mobile Payment System is able to handle only one mobile transaction for each customer. A problem will come if the customer has performed multiple purchases on the same vshop and the vshop redirect the customer to the Mobile Portal of the User Bank after the client finishes purchasing and quits to shop. This problem can be easily solved if we make a separate Java Server Page which tests which of the multiple Confirm buttons on the client's mobile payment account web page has been activated.

2. Our error handling techniques are pretty simple and there are some possibilties that the customer payes for a merchandise but never receives the product.This situation can occur if one of the special tasks described before fails to preform due to some external reason.This problem can be solved if there is more complicated error handling and collaboration between banks and vshop.

6 Interconnections

6.1 Overview

Interconnections entail the method of communication used by each component in the project enviroment. A component represents any endpoint of the connection such as the vshop or the vbank. For simplicity, interconnections can be divided up into the catagories of implementation and protocol. Implementation deals with the actual code structore and the decisions that went into the design of the java socket code. The protocol encompasses what different security standards are commonly used for mobile payment systems such as SSL or WTLS. Protocol also deals with the actually message formats that are used for communication such as WAPPUSH.

6.2 Protocol Detail

NOTES: Assuming reader has be informed about the general SSL procedure, certificates, and what keystores and truststores are also defn of ciphers.

6.2.1 SSL

In java, the SSL protocol is encapsulated in the Java Secure Socket Extension package(JSSE). Encapsulation allows maximum flexibility in what developers are able to do with the many encryption ciphers available in SSL. Also, this was done to maintain stability by standarizing socket property manipulation. For example, the SSLSocketFactory and SSLServerSocketFactory are sockect wrapper classes that manage all the properties of secure sockets. Although a developer could set any desired encyrption property though the SSLcontext class such as change the security provider, all socket manipulation is still handled by the respective SocketFactory classes.

In this project, the default encryption properties were used. This was achieved though the call of *SSLSocketFactory.GetDefault()* method. The predefined default uses the SunJSSE Security Provider. This provider follows the X.509 standard for certification creation and uses the **SSL_RSA_WITH_RC4_128_SHA** encyrption cipher suite for key encryption.

As for key management, each component was given a sample keystore and truststore. These stores were created through the use of a program called keytool that allows the creation of self signed certificates. Self signed certificates were used instead of an actual CA for the sake of simplifying of the use case. Since using an actually CA to sign certificates would require setting up certification paths and possibly the purchase of an atually certificate, it was decided that implementation of CA interaction was beyond the scope of the use case that was modeled. Instead, the project makes the assumption that all the banks and vshop trust the certificates of each respective part. For details on the keytool program and the actual certificates made, please see the appendix.

6.2.2 WTLS

WTLS is based on the Transport Layer Security protocol (TLS), which was designed from the Secure Sockets Layer protocol (SSL). WTLS provides privacy and reliability for client/server communications over a network, and is more suited for the limited memory and processing capabilities of WAP enabled mobile phones.

From the WAP Framework Architecture, when a gateway receives a request from a WAP client, it's translated into HTTP to communicate with the appropriate content server. In the implementation of a secure WAP system, the communication between the WAP client and gateway is encrypted with WTLS. The gateway will then decrypt WTLS and then re-encrypt using SSL to connect to the content server. The SSL connection is similar to what one would find in a traditional secure Internet application.

For the final implementation for this project, WTLS was going to be used to connect from the Nokia Toolkit to the vshop. However, there was no use of WTLS protocols due to various difficulties.

6.2.3 WAPPUSH

The project follows the convention, as described in the Introduction, of using "*push*" protocols to send messages from one device/system to another, in the WAP architecture framework. However, due to many factors hindering progress, this method of sending messages was abandoned for the project.

WAP Framework Architecture (Figure):

{WAP Terminal}—(OTA Protocol)—{Push Proxy Gateway (PPG)}—(PAP)—{WAP Server}

(**Note:** Due to difficulties regarding the PPG (WAP Gateway), as discussed later in the report, this part of the Architecture was completely ignored project-wise for future reference. However, the PPG (or Legacy Protocol stack if using WAP 2.0) is supposed to be important part of the architecture since it parses the WML formatted message sent by the terminal to a HTTP formatted message for the server. Vice versa, the PPG sends the terminal the WML/WMLScript message that the server sends.)

A Push message is an unsolicited message sent by a server to a client. The message format basically is an XML document, where all the information for the message is held. This document is encapsulated with header field content (either WML or HTML depending on which device is obtaining the push message) which contains configuration information

about how the message shall be received and the settings of the receiving party.

Here is the prospectus of how the WAP Push functionality should have worked forthis project:

Simulating the PAP from the PushInitiator to the PPG, a Pushinitiator Java program (Pushinitiator.java) was written with the JDK 1.4 API libraries (downloaded from sun.java.com website). This was written using socket layer network code (TCP) with the xml SI message document encapsulated in the program. Execution of the program follows entering the port and IP address of the client program (this will later be discussed in a future section) on a DOS command line prompt, where the XML message will be sent to the client.

Pushinitiator command prompt:

```
>PushInitiator http://127.0.0.1:8800
```

where *http://127.0.0.1:8800*, is the ipaddress (127.0.0.1) and port (8800) of the receiving client.

The PushInitator code was supposed to be incorporated with the functionalities of the vshop and vbanks.

Simulating the Push OTA (Over The Air) Protocol from the WAP terminal (the virtual phones from the Nokia/Ericsson SDK's) to the receiving party, the virtual phone would receive the message from the vshop (which would have the PushInitiator code) firstly. Since the virtual phone's simulated the behavior of an actual mobile cell phone, the SDK's would display the message content received, and the user would hit a reply on one of the keys (**eg:** yes/no).

This Push OTA behavior would have been similar for the vshop and vbank interconnections as well.

6.2.4 Installation and Problems encountered

1. **SSL:**

   As mention above, the JSSE package hides most of the security implementation details from the code in order to offer the developer a portable interface to manipulate the sockets. This led to issues in terms of actual file path configurations. For example, a typical SSLClient needs to know what certificates it can trust by looking in its truststore. However, since the security implementation details are hidden, it was quite difficult finding the default location of this truststore. Although unsucessful

in finding the default locations for the respective keystores and truststores, a flag command was found that could overide the default location paths. This command specified a java.lang.system property upon code execution. This led to sublteties in implementation that are discussed in detail in the Imlementation section. For details on the syntax of this flag please see the appendix.

2. **WAPPUSH:**

Factors that influenced not utilizing WAP Push/WTLS for the project were mainly due to software difficulties and lack of appropriate resources.

Software:

For both the Nokia and Ericcson development kits, problems were faced with not being able to figure out how to use their functionalites appropriately. For the Nokia virtual phone, the message content would not display for the PushInitator program despite references from the Nokia website and help documents that this method should supposedly work. Similarly, the WAP Gateway that is built into the Nokia Toolkit wouldn't function correctly as well. The Ericsson virtual phone would not work for the applications of this project without the Ericsson PPG, which would have cost a great amount of money.

Also, time was wasted in finding a WAP gateway that would compile and run in the Windows environment. Many possible ones offered, such as OpenWave's WAP Gateway, required a large monetary compensation for download use. Nokia Activ Server 2.1 and Nokia Activ Alert 2.1 were found to support the Nokia SDK, where the former is the WAP gateway and the latter is a Push cluster component for the Nokia Activ server. The PushInitiator program could connect to the Nokia Active Server 2.1, but a method to configure the server to foward this information to the Nokia Toolkit was not found.

Other problems/limitations that were encountered:

- Constant change in the Nokia website
- Needed to downgrade JDK from newest version (1.4 we believe) to 1.3 for installing Nokia Toolkit.
- No service support for toolkit developers

- Expected software behavior not found. Wasted time on fixing/researching these issues (ex: Pushinitiator code for nokia toolkit)
- No relevant info/answers to our questions on the web (ex: previous posts)
- Proper documentation describing flaws in toolkits not found

As a result, there was no time to invest in researching possible intrusion detection solutions for this project such as implementing firewall Java code for the PushInitiator or other security protocols such as digital signatures, certificates, or WTLS encryption/decryption algorithms.

6.2.5 Limitations and Possible Improvements

Since the SSL protocol is encapsulated in JSSE and this package has gone under many revisions over the different versions of java, the SSL code will only run properly under the latest version of java (currently 1.4.). During the project implementation, this was a great hinderance since a majority of the cluster computers on campus didn't have the latest version (example: wean linux cluster had java 1.3.1 and wean windows cluster had java 1.2). A possible improvement to the interconnection system is to create code that offers backward compatability across older versions of the java JSSE package.

6.3 Implementation

6.3.1 Description

Through the use of serializable objects, each component is able to create specific messages without worrying about stream implementation details. Objects extending Serializable have the characteristic of being convertable to and from byte streams. This characteristic allows abstraction on what is sent when sockets are open for input and output streams.

As for socket management, creating and opening sockets for message communication is encapsulated by the SSLserver and SSLclient objects. These classes deal with securing an input and output stream through the network by following SSL protocol. Each vbank and vshop will contain a SSLserver object to process incoming messages but will create SSLclient objects as needed per message sent to other components. To handle multiple incoming messages, SSLserver extends ServerTread, a class that creates new threads per each client that connects to a server.

The reason for this interconnection architecture is to maximize coupling between communication code and component webserver code. This design choice allows maximum portablity since each individual component does not rely on the implemenation of

interconnections in order to run properly. This also allows the ability to import other interconnections models such as WAP into the project environment.

6.3.2 Installation and Problems encountered

Although this code architecture allows separation of component and interconnection code, certain subtleties exist in component code regarding the creation of interconnection objects. Since SSL is used as the protocol to ensure security and authentication of messages, certificate information is need during the creation of SSLserver/SSLclient objects. To achieve this the exec method is used. The exec method creates a sub-process that executes user defined instructions. This gives rise to a security risk since the user has the ability to tag other instructions (example: pipe command) after the creation of the SSLobject. Although the exec command has the reputation of being unsafe, the components take this into consideration by using pre-defined parameters and using user input type checking.

Another subtlety is the object encapsulation of the message itself. Serializable objects where implemented in java with the focus of simplicity in code and not security of material. Although difficult, with enough study of the bytestreams, a hacker can reconstruct the object from it. Once the message object is known, that user can duplicate a message and possibly send malicious messages to servers. Currently, components in this project handle this security risk by using SSL to encapsulate the object streams and also by using hash codes to create message digests.

6.3.3 Testing efforts

In this project, abstract class InterServerMsg models the basic form of a typical message sent between any two components. Deriving specific messages from this class allowed stability of socket code and also a method of debuging since each derived class inherited InterServerMsg's function to print out it's own contents.

For the socket code, testServer.java and testClient.java were created. These driver programs were used to check the stability of the multi-threaded server code and the server certificate authentication needed by the client. We then used programs such as netstat and ping to watch and debug packet information.

6.3.4 Limitations and Possible Improvements

This interconnection architecture also limits the level of privacy offered. For example, In this use case, since a message is sent from vshop to user bank to verify the payment, the user bank has the possibility to monitor user purchasing trends. A possible improvement to the interconnection system is to add, multiply layers of encryption so that only certain recepients will be able to veiw certain information. For example, there could be a different encryption used for the item bought at the vshop so that only the user and

vshop could decrypt this information but when the order is fowarded to the user bank, it will only be able to decrypt the price of this object.

Finally, an improvement for scalability is to handle the case where socket timeouts occur during client connections to the server. Although the server can create new threads per client, this can become quite expensive if it is waiting for clients that don't close the socket properly or has prematurely terminated the connection without notifying the server. By default, the linux has a 20 minute timeout countdown but this can be adjusted to further tweak server performance.

7 Conclusion

7.1 System Strength and Weaknesses

In terms of goals, we were able to satisfy most of our terms. For instance, the system that we currently implemented has the essential key process concepts necessary for a secure, convenient, and robust mobile payment system (as seen in our demonstration and the quality of our source code). However, there are still some concerns that should be addressed. We go into detail in the following sections.

1. **Security:** The system is able to handle multiple secure connections. Unfortunately, the system is not able to handle WTLS. This is due to self-signed certificate complications with respect to the ericsson WAP gateway.

2. **Privacy:** In the current implementation, the userbank has more information than is needed. This leads to a compromise in privacy because there is the potential that the bank will use this information to track user purchasing patterns.

3. **Scalability:** Although this model is robust for the purpose of the demonstration, the system can be modified for large scale industry usage. These modifications entail, more user input checks and possible optimization of message size. For example, the php session tags can be shorter and the html files can be stored as one letter names. Another possible improvement can be implementation of security measures that handle Denial of Service (DoS) and socket timeout appropriately.

References

[1] Electronic Payment Systems Observatory
http://epso.jrc.es

[2] Mobile Electronic Transaction Initative
http://www.mobiletransaction.org
http://www.cellular.co.za/met.htm
http://www.cellular.co.za/mcommerce.htm

[3] Mobey Forum Initiative
http://www.mobeyforum.org

[4] Mobile Payment Forum Initiative
http://www.mobilepaymentforum.org

[5] W3C
http://www.w3c.org

[6] I-Mode
http://www.eurotechnology.com/imode/

[7] Sonera *www.sonera.se*

[8] MobilPay *www.mobilpay.com*

[9] Nokia Development Kit and Wap Emulators
http://www.nokia.com

[10] Ericsson
http://www.ericsson.com *http://www.sonyericsson.com*

[11] Openwave
http://www.openwave.com

[12] Apache Web Servers
http://www.apache.org
http://jakarta.apache.org

[13] Java API documentation
http://www.java.sun.com

[14] Databases
http://www.mysql.com
http://www.oracle.com

A Appendix

A.1 Survey (based on 20 people)

This survey was conducted in Oakland asking people from different backgrounds about the WAP possibilities on a SonyEricsson T68i mobile device. The following survey results have been recorded from different people conducted. You can see first the number of a surveyed person, after this it follows the University and the primarily major. The surveyed persons were offered to rate their experience by browsing different websites as for example wap.voicestream.com or www.sonyericsson.com on a scale between 1 and 3, based upon the screen size, connection speed and navigation.

Screen size: 1 (good); 2 (average); 3 (bad)
Connection speed: 1 (fast); 2 (normal); 3 (slow)
Navigation: 1 (simple); 2 (average); 3 (complicated)

A.1.1 Detailed results

1—UPitt History of art architecture
Screen size: 1
Connection speed: 2
Navigation: 2

2—UPitt Industrial Design
Screen size: 1
Connection speed: 2
Navigation: 1

3—not a student
Screen size: 1
Connection speed: 1
Navigation: 1

4—CMU CS
Screen size: 1
Connection speed: 3
Navigation: 2

5—CMU CS
Screen size: 1
Connection speed: 2
Navigation: 2

6—CMU CS
Screen size: 2
Connection speed: 3
Navigation: 2

7—CMU ECE
Screen size: 2
Connection speed: 2
Navigation: 1

8—CMU CS
Screen size: 3
Connection speed: 3
Navigation: 3

9—UPitt Medical School
Screen size: 1
Connection speed: 2
Navigation: 1

10—CS Master CivilEng.
Screen size: 1
Connection speed: 2
Navigation: 1

11—Starbucks worker
Screen size: 1
Connection speed: 1
Navigation: 1

12—Starbucks worker
Screen size: 1
Connection speed: 1
Navigation: 3

13—UPitt Mechanical Eng.
Screen size: 2
Connection speed: 3
Navigation: 2

14—CMU Business Adm
Screen size: 1
Connection speed: 1
Navigation: 1

15—CMU IS
Screen size: 2
Connection speed: 2
Navigation: 2

16—CMU Business Adm
Screen size: 1
Connection speed: 1
Navigation: 1

17—CMU Economics
Screen size: 2
Connection speed: 2
Navigation: 1

18—CMU MBA
Screen size: 2
Connection speed: 2
Navigation: 2

19—Kiva Han worker
Screen size: 1
Connection speed: 1
Navigation: 2

20—Kiva Han worker
Screen size: 1
Connection speed: 1
Navigation: 1

From this survey we can conclude that there is no general opinion on the cellular phone characteristics. Perhaps this indicates that people are still not really familiar with browsing the internet through a mobile device. We can see that some like the small size and others not. I would say the connection speed has the lowest rating which indicates that the connection used by conducting this survey was not really good. It depends on different locations which connection rate we were able to achieve. My assumption is that we didn't have most of the time GPRS connection rate, which goes up to four time more than the GSM rate which is around (9600).

The conclusion from our small survey would be that most of the people are still not familiar with mobile web and that the connection available right now is not the good one.

A.2 CD

The CD included with this report contains the following files:

- An electronic version of this report
- All the source code and compiled classes needed to run the demo
- Complete Nokia toolkit with 6590 phone, docs, and the extra nokia servers
- Ericsson toolkit
- Openwave toolkit
- MySql server
- Apache webserver
- JConnector (MySql driver for Java) located in the java folder
- Java development toolkit version 1.4.1
- Java runtime version 1.3 in case a downgrade is needed
- Php for apache, docs and phpMyAdmin
- Winzip
- Acrobat Reader

A.3 Source code for the vshop

A.3.1 Webpages

Listing 1: Index.php

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>VShop</title>
<link rel="stylesheet" href="style.css" type="text/css"/>
</head>
<body>
<p class='box'><b>Carnegie Mellon University</b></p>
<p>15-494 Web commerce, security and privacy project.<br/></p>
<p><center><a href="login.php">enter vshop</a></center></p>
</body>
</html>
```

Listing 2: Login.php

```
<?
/* This is the login page.
   The user is requested to enter his phone number
   and password. If the user's wap gateway forwards the phone
   number, than the field will be automatically filled.
```

```
   A second option would be to fill it using client side script.

   XHTML should have perhaps provided support for this ?
*/

// Read the http header.
/* We could have increased the amount of headers in wish we
   look for the phone number.
*/
$headers = getallheaders();
$cid = $headers['X-Nokia-CHARGING_ID'];

// Connect to database.
mysql_connect("localhost", "root", "root");
mysql_select_db("vshop");

if (!is_null($cid)) {
  // We found the phone number in the header.
  $result=mysql_query("select userPwd,autoLogin from users where phoneNo='$cid'");
  if ($row=mysql_fetch_row($result)) {
    if ($row[1]==1) {
      // Check to see if user has enabled autologin.
      Header("Location: engine.php?user=$cid&amp;pw=$row[0]");
      exit;
    }
    // Save the phone number for the input field.
    $userid=$cid;
  }
}
?>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>VShop</title>
<link rel="stylesheet" href="style.css" type="text/css"/>
</head>
<body>
<p class='box'><b>Login</b></p>
<form method="get" action="engine.php"><table>
<tr><td style="width: 30%">Phone No:</td><td><input type="text" name="user" value="
    <?=$userid?>"/></td></tr>
<tr><td>Password:</td><td><input type="password" name="pw"/></td></tr>
<tr><td colspan="2" style="text-align: center"><input type="submit" value="login"/>
    </td></tr>
</table></form>
</body>
</html>
```

Listing 3: Engine.php

```
<?
/* This is the main vshop engine.
   Sessions are handled here.
   The user is authenticated and then can start browsing
   the catalogue.
*/

// Disable trans_sid because of bug in sonyericsson phone !
ini_set("session.use_trans_sid", "0");
// Disable cookies, as we prefer to use url parameters (more widly supported).
ini_set("session.use_cookies", "0");
session_start();
```

```
// Connect to database.
mysql_connect("localhost", "root", "root");
mysql_select_db("vshop"); ?>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>VShop</title>
<link rel="stylesheet" href="style.css" type="text/css"/>
</head>
<body>
<?
/* Authentication page.

   Uses php session to log user. The session id is passed in the url,
   and is handled "automatically" with php.

   If the current_user is set, browse the catalogue (if the page isn't set, show the
       home page)
   Else authenticate the user and if correct, set the current_user to $user
*/

// check if current_user is set
if (!session_is_registered('current_user')) {
  // We must check the user/password
  $result=mysql_query("select * from users where phoneNo='$user' and userPwd='$pw'")
      ;
  $num=mysql_numrows($result);
  if ($num==0) {
    // access denied
    print("Access denied, sorry.<br/>Phone No: $user<br/>Password: $pw</body></html
        >");
    exit;
  } else {
    // Register session and
    // set page to display to home.
    $current_user=$user;
    session_register("current_user");
    $page="home";
  }
}

// default page is home
if (is_null($page))
  $page="home";

if ($page=="home") {
  // Handle special case of home page
  $result=mysql_query("select name,address from users where phoneNo='$current_user
      '");
  $row=mysql_fetch_row($result);
  $address=$row[1];
  print("<p>Welcome back $row[0].</p>");
  $result=mysql_query("select category from categories where father='home' order by
      category");
  $num=mysql_numrows($result);
  print("<p class='box'><b>Select category</b></p>");
  print("<table style='text-align: center'>");
  while($row=mysql_fetch_row($result)) {
    if ($row2=mysql_fetch_row($result)) {
      print("<tr><td><a href='engine.php?page=".urlencode($row[0])."&amp;".SID."'>
          $row[0]</a></td>");
```

```
      print("<td><a href='engine.php?page=".urlencode($row2[0])."&amp;".SID."'>$row2
          [0]</a></td></tr>");
    } else {
      print("<tr><td><a href='engine.php?page=".urlencode($row[0])."&amp;".SID."'>
          $row[0]</a></td><td></td></tr>");
    }
  }
  print("</table><hr/>");
  print("<small><p>Is this address right:</p><p>$address ?</p>");
  print("<p style='text-align: right'>user profile</p></small>");
} else {
  // Normal catalogue pages
  $result=mysql_query("select category from categories where father='$page' order by
       category");
  $num=mysql_numrows($result);

  // Header.
  if ($num>0) {
    print ("<p class='box'><small>");
    $i=0;
    while($row=mysql_fetch_row($result)) {
      $i++;
      print("<a href='engine.php?page=".urlencode($row[0])."&amp;".SID."'>$row[0]</a
          >");
      if ($i<$num)
        print(" | ");
    }
    print ("</small></p>");
  }

  // Main content.
  $result=mysql_query("select productID,name,price from products where category='
      $page' order by price");
  $num=mysql_numrows($result);
  if ($num>0) {
    print ("<table><tr><th style='width:65%'>Items</th><th style='width:35%'>Price</
        th></tr>");
    while ($row=mysql_fetch_row($result)) {
      print("<tr><td><a href='detail.php?item=$row[0]&amp;".SID."'>$row[1]</a></td><
          td>\$" . number_format($row[2], 2) . "</td></tr>");
    }
    print ("</table>");
  }

  // Footer.
  function path($rec) {
    $result=mysql_query("select father from categories where category='$rec'");
    $row=mysql_fetch_row($result);
    if ($row[0]!="home") {
      path($row[0]);
      print (" / <a href='engine.php?page=".urlencode($row[0])."&amp;".SID."'>$row
          [0]</a>");
    } else {
      print ("<a href='engine.php?page=".urlencode($row[0])."&amp;".SID."'>$row[0]</
          a>");
    }
  }
  print ("<p class='box'><small>Quick browse: ");
  path($page);
  print ("</small></p>");
}
?>
```

```
</body>
</html>
```

Listing 4: Detail.php

```
<?
/* This page simply displays the details of an item.
   Nothing very special about it.
   The case of inexistant product doesn't need to be handled...
*/

// Disable trans_sid because of bug in sonyericsson phone !
ini_set("session.use_trans_sid", "0");
// Disable cookies.
ini_set("session.use_cookies", "0");
session_start();
if (!session_is_registered('current_user')) {
  // Make sure user didn't just type the url.
  Header("Location: index.php");
  exit;
}

// Connect to database.
mysql_connect("localhost", "root", "root");
mysql_select_db("vshop");
?>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>VShop</title>
<link rel="stylesheet" href="style.css" type="text/css"/>
</head>
<body>
<?
// Get product info.
$result=mysql_query("select name,price,description from products where productID='
    $item'");
$row=mysql_fetch_row($result);

print("<p class='box'><b>$row[0]</b></p>");
if ($row[3]!="")
  print("<p><img src='1.gif' style='float: left'/>$row[2]</p>");
else
  print("<p>$row[2]</p>");
print("<p><center><a href='buy.php?item=$item&amp;".SID."'>Buy for \$" .
    number_format($row[1], 2) . " </a></center></p>");
?>
</body>
</html>
```

Listing 5: Buy.php

```
<?
/* This page connects to the bank and sends a UserPurchaseMsg java object.
   It then redirects the user to the bank's website.
*/

// Disable trans_sid because of bug in sonyericsson phone !
ini_set("session.use_trans_sid", "0");
// Disable cookies.
ini_set("session.use_cookies", "0");
session_start();
```

```
if (!session_is_registered('current_user')) {
  // If the user got here by typing the url (and not actually selecting an item)
  // then go back to homepage and request a login.
  Header("Location: index.php");
  exit;
}

// Connect to database.
mysql_connect("localhost", "root", "root");
mysql_select_db("vshop");

/* Get the user's info */
$result=mysql_query("select bankIP,accountNo,name from users where phoneNo='
    $current_user'");
$row=mysql_fetch_row($result);
$dest=$row[0];
$port=15494;
$username=$row[2];
$accountno=$row[1];
$vshopaccountno="1235";
$vshopbankip="urlin1.wv.cc.cmu.edu";

/* Get product info */
$result=mysql_query("select name,price from products where productID='$item'");
$row=mysql_fetch_row($result);
$article=$row[0];
$price=$row[1];

/* Add to transactions tables */
$result=mysql_query("insert into transactions (productID, price, userID, shipment)
    values ('$item', '$price', '$current_user', '00000000000000')");
$trans=mysql_insert_id();

/* Connect to bank servlet */
$e=exec("java -Djavax.net.ssl.trustStore=vshop_truststore -Djavax.net.ssl.
    trustStorePassword=webcom SSLUserPurchase $dest $port \"$username\" $accountno
    $vshopbankip $vshopaccountno \"$article\" $price $trans \"alok.wv.cc.cmu.edu\"")
    ;

if ($e!="ok") {
  // An error occured with cient java code.
  print("<html xmlns=\"http://www.w3.org/1999/xhtml\">\n<head>\n<title>Error</title
      ></head><body>");
  print("<p>An error occured while trying to connect to bank.<br/>");
  print $e;
  $result = mysql_query("update transactions set shipment='20000101000000' where
      transactionID=$trans");
  print("</p></body></html>");
  exit;
}

/* Add to waiting table */
mysql_query("insert into waiting (transactionID) values ('$trans')");

/* Go to bank to confirm */
Header("Location: http://$dest:8080/conf.jsp?accno=$accountno");
?>
```

A.3.2 SQL Queries

These are the queries needed to create the vshop's database.

```
create table users (
    phoneNo char(11) not null,
    userPwd char(11),
    gatewayIP char(15),
    gatewayPort int,
    name varchar(20),
    autoLogin bool,
    bankIP char(15),
    bankPort int,
    accountNo char(15),
    address varchar(60),
    primary key(phoneNo)
);

create table transactions (
    transactionID int not null auto_increment,
    productID int not null,
    price float,
    userID char(11) not null,
    purchase timestamp,
    shipment timestamp,
    primary key(transactionID),
    foreign key (productID) references products,
    foreign key (userID) references users
);

create table waiting (
    transactionID int not null,
    primary key(transactionID),
    foreign key (transactionID) references transactions
);

create table toship (
    transactionID int not null,
    primary key(transactionID),
    foreign key (transactionID) references transactions
);

create table categories (
    category char(20) not null,
    father char(20),
    primary key (category),
    foreign key (father) references categories(category)
);
```

```
create table products (
    productID int not null auto_increment,
    category char(20),
    name varchar(20),
    price float,
    description varchar(200),
    primary key(productID),
    foreign key(category) references categories
);
```

A.4 Source code for the virtual banks

A.4.1 Source code of the UserBank WebServer

Listing 6: UserLogIn.html

```
<HTML>
<HEAD><TITLE>User Virtual Bank</TITLE>
</HEAD>
<BODY bgcolor = "blue">
<P>
<font size = "20" face= "Verdana">USER  VIRTUAL BANK
 </font></P>
<P>
<P>
<br>
<FORM name="form1" method="post" action="UserTestLogIn.jsp">
<P>Customer number </P>
<INPUT type="text" name="accountid" maxlength="10">
<P>PIN</P>
<INPUT type="password" name="pinid" maxlength="4">
<P>
<INPUT type="submit" name="Sign on" value="Sign on">
</FORM>
</BODY>
</HTML>
```

Listing 7: UserTestLogIn.jsp

```
<%@ page import="java.util.*" %>
<%@ page import="java.sql.*" %>
<%!
String OraDriver= "oracle.jdbc.driver.OracleDriver";
String dbUrl= "jdbc:oracle:thin:@dbclass.intro.cs.cmu.edu:1521:dbintro";
String user;
String password;
String sqlStmtLogIn;
%>

<%
try {

user=request.getParameter("accountid");
session.setAttribute("session",user);
password= request.getParameter("pinid");
boolean entrance = false;
```

```
Class.forName(OraDriver);

Connection con = DriverManager.getConnection(dbUrl,"bdanev","danev");
Statement stmt = con.createStatement();


sqlStmtLogIn = "SELECT * FROM USERUSERS WHERE ACCOUNTID='"+user+"' AND PINID='"+
    password+"'";


ResultSet rs = stmt.executeQuery(sqlStmtLogIn);

while(rs.next()){
String dbUser = rs.getString("accountid");
String dbPassword= rs.getString("pinid");
if ((user.equals(dbUser)) && (password.equals(dbPassword)))
   {
   entrance=true;
   }

}

if(entrance == true){
rs.close();
stmt.close();
con.close();
%>
<jsp:forward page="UserUserAccount.jsp"/>
<%}
else
{
rs.close();
stmt.close();
con.close();
%>
<jsp:forward page="UserLogIn.html"/>
<%}
}

catch (Exception e) {
out.print("Something is wrong" + e.toString());
}
%>
```

Listing 8: UserUserAccount.jsp

```
<html>
<body>
<%@ page import="java.util.*" %>
<%@ page import="java.sql.*" %>
<%!
String OraDriver= "oracle.jdbc.driver.OracleDriver";
String dbUrl= "jdbc:oracle:thin:@dbclass.intro.cs.cmu.edu:1521:dbintro";
String user;
String dbName;
String dbBalance;
%>

<%
try {

Class.forName(OraDriver);
```

```
Connection con = DriverManager.getConnection(dbUrl,"bdanev","danev");
Statement stmt = con.createStatement();

user=(String)session.getAttribute("session");


ResultSet rs = stmt.executeQuery("SELECT * FROM USERACCOUNT  WHERE ACCOUNTID='"+user
    +"'");

while(rs.next()){
dbName = rs.getString("customername");
dbBalance= rs.getString("currentbalance");

}
//out.print("<tr><td>"+ dbName + "</td><td>" + dbBalance+ " $</td></tr>");
out.println("Hello " + dbName); %>
<P>
<%
out.println("Your current balance is :" + dbBalance + " $");
%>
<P> All your transactions are: </P>
<%
ResultSet ns = stmt.executeQuery("SELECT * FROM USERTRANSACTIONS WHERE ACCOUNTID='"+
    user+"'");
while (ns.next()) {
String dbTransactionDate = ns.getString("transaction_date");
String dbTransactionDesc = ns.getString("description");
String dbTransactionAmount = ns.getString("amount");
out.print("<tr><td>"+ dbTransactionDate + "    </td><td>" +
dbTransactionDesc+ "  </td><td>"+ dbTransactionAmount + " $</td></tr><BR>");
}

rs.close();
stmt.close();
con.close();
} catch(Exception e) {
out.print("Entrance wrong" + e.toString());
}
%>
<P>
<P>
<a href = "UserLogOut.jsp">
sign off
</a>

</body>
</html>
```

Listing 9: UserLogOut.jsp

```
<%@ page import="java.util.*" %>
<%@ page import="java.sql.*" %>
<%
session.invalidate();
%>
<jsp:forward page ="UserLogIn.html"/>
```

Listing 10: conf.jsp

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
<title>User Bank</title>
<link rel="stylesheet" href="style.css" type="text/css"/>
</head>
<%!
String defaultuser = null;
%><%
defaultuser = request.getParameter("accno");
if (defaultuser == null)
    defaultuser = "";
%><body>
<p class='box'><b>UserBank</b></p>
<form method="get" action="MobileTestLogIn.jsp">
<p>Customer Number:</p>
<p>
<% out.print("<input type=\"text\" name=\"customer_number\" value=\""+defaultuser
    +"\"/>");
%>
</p>
<p>Password:</p>
<p><input type="password" name="password"/></p>
<p><input type="submit" value="login"/></p>
</form>
</body>
</html>
```

Listing 11: UserMobileAccount.jsp

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Mobile Confirmation</title>
</head>
<body>
<%@ page import="java.util.*" %>
<%@ page import="java.sql.*" %>
<%!
String OraDriver= "oracle.jdbc.driver.OracleDriver";
String dbUrl= "jdbc:oracle:thin:@dbclass.intro.cs.cmu.edu:1521:dbintro";
String user;
String name = null;
%>

<%
try {

Class.forName(OraDriver);

Connection con = DriverManager.getConnection(dbUrl,"bdanev","danev");
Statement stmt = con.createStatement();

user=(String)session.getAttribute("session");

ResultSet rs = stmt.executeQuery("SELECT CUSTOMERNAME FROM USERACCOUNT WHERE
    ACCOUNTID='"+user+"'");
if (rs.next()) name = rs.getString(1);
out.println("<p><b>Hello " + name +"</b></p>");
rs.close();
%>
<p>
<font size="4" color="red">
Mobile transaction to confirm:
</font>
```

```
</p>
<%
ResultSet ns = stmt.executeQuery("SELECT * FROM USERMOBILEPAYMENTS WHERE
    CLIENTACCOUNTID='"+user+"'");
if (ns.next()) {
String dbTransactionDate = ns.getString("transaction_date");
String dbTransactionDesc = ns.getString("description");
String dbTransactionAmount = ns.getString("amount");
out.print("<p><b>Description</b></p>");
out.print("<p>"+dbTransactionDesc+"</p>");
out.print("<p><b>Date</b></p>");
out.print("<p>"+dbTransactionDate+"</p>");
out.print("<p><b>Amount</b></p>");
out.print("<p>"+dbTransactionAmount+" $</p>");
}
%>
<p><a href="UserConfirmMobileTransaction.jsp">
<font size="3" color="green">Confirm</font></a></p>
<%

ns.close();
stmt.close();
con.close();
} catch(Exception e) {
out.print("Entrance wrong" + e.toString());
}
%>
<p>
<a href = "UserMobileLogOut.jsp">
<font size="3" color="red">Reject</font>
</a>
</p>
</body>
</html>
```

Listing 12: MobileTestLogIn

```
<%@ page import="java.util.*" %>
<%@ page import="java.sql.*" %>
<%!
String OraDriver= "oracle.jdbc.driver.OracleDriver";
String dbUrl= "jdbc:oracle:thin:@dbclass.intro.cs.cmu.edu:1521:dbintro";
String user;
String password;
String sqlStmtLogIn;
%>

<%
try {

user=request.getParameter("customer_number");
session.setAttribute("session",user);
password= request.getParameter("password");
boolean entrance = false;

Class.forName(OraDriver);

Connection con = DriverManager.getConnection(dbUrl,"bdanev","danev");
Statement stmt = con.createStatement();
```

```
sqlStmtLogIn = "SELECT * FROM USERUSERS WHERE ACCOUNTID='"+user+"' AND PINID='"+
    password+"'";


ResultSet rs = stmt.executeQuery(sqlStmtLogIn);

while(rs.next()){
String dbUser = rs.getString("accountid");
String dbPassword= rs.getString("pinid");
if ((user.equals(dbUser)) && (password.equals(dbPassword)))
   {
   entrance=true;
   }

}

if(entrance == true){
rs.close();
stmt.close();
con.close();
%>
<jsp:forward page="UserMobileAccount.jsp"/>
<%}
else
{
rs.close();
stmt.close();
con.close();
session.invalidate();
%>

<jsp:forward page="conf.html"/>
<%}
}

catch (Exception e) {
out.print("Something is wrong" + e.toString());
}
%>
```

Listing 13: UserMobileLogOut.jsp

```
<%@ page import="java.util.*" %>
<%@ page import="java.sql.*" %>
<%
session.invalidate();
%>
<jsp:forward page ="conf.html"/>
```

Listing 14: UserConfirmationMobileTransaction.jsp

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Proceed Transaction</title>
</head>
<body>
<%@ page import="java.util.*" %>
<%@ page import="java.sql.*" %>
<%@ page import="java.lang.*" %>
<%@ page import="ch.cmu.webproject.*" %>
<%!
```

```
String OraDriver="oracle.jdbc.driver.OracleDriver";
String dbUrl="jdbc:oracle:thin:@dbclass.intro.cs.cmu.edu:1521:dbintro";
String user;
String vshopbankport = "15496";
String vshopport = "15494";
Connection con = null;
Statement stmt = null;
ResultSet ns = null;
%>

<%
user =(String)session.getAttribute("session");
try {
Class.forName(OraDriver);
con = DriverManager.getConnection(dbUrl,"bdanev","danev");
stmt = con.createStatement();
ns = stmt.executeQuery("SELECT * FROM USERMOBILEPAYMENTS WHERE CLIENTACCOUNTID='"+
    user+"'");
if (ns.next()) {
String dbMobileID = ns.getString("mobile_id");
String dbClientAccountNo = ns.getString("clientaccountid");
String dbClientName = ns.getString("clientname");
String dbVshopBankIP = ns.getString("vshopbankip");
String dbVshopIP = ns.getString("vshopip");
String dbVshopAccountID = ns.getString("vshopaccountid");
String dbDescription = ns.getString("description");
String dbAmount = ns.getString("amount");
String dbTransactionID = ns.getString("transaction_id");
String dbTransactionDate = ns.getString("transaction_date");


CurrentDate newdate = new CurrentDate();
String date = newdate.getCurrentDate();

/* Open a socket and send the money to the vshop bank */
String newdesc = dbDescription.replace(' ','_');
Process bankpayment = null;
bankpayment = Runtime.getRuntime().exec("/usr/java/j2sdk1.4.1_01/bin/java -cp /var/
    tomcat4/webapps/ROOT/WEB-INF/classes/ -Djavax.net.ssl.trustStore=/var/tomcat4/
    webapps/ROOT/WEB-INF/classes/ch/cmu/webproject/userbank_truststore -Djavax.net.
    ssl.trustStorePassword=webcom SSLBankPayment "+dbVshopBankIP+" "+"15496"+" "+
    dbVshopAccountID+" "+newdesc +" "+dbAmount+" "+dbTransactionID+" "+ dbVshopIP);


if (bankpayment == null)
   throw new Exception("Bankpayment failed to be sent!");


/* Insert the transaction in the database and update the amount if payment
    successfully sent*/
String sqlStmtInsert = "INSERT INTO USERTRANSACTIONS VALUES('"+user+"','"+date
    +"','"+dbDescription+"',"+dbAmount+")";
int SQLCheck = stmt.executeUpdate(sqlStmtInsert);
if(SQLCheck == 0)
   throw new Exception("Problem in Insertion");
String sqlStmtUpdate= "UPDATE USERACCOUNT SET CURRENTBALANCE = CURRENTBALANCE-"+
    dbAmount + "WHERE ACCOUNTID='"+dbClientAccountNo+"'";
int SQLUpdate = stmt.executeUpdate(sqlStmtUpdate);
if(SQLUpdate == 0)
   throw new Exception("Problem in Update");

/* Delete the mobile transaction from usermobilepayments */
```

```
String del = "DELETE FROM USERMOBILEPAYMENTS WHERE MOBILE_ID ="+dbMobileID;
int SQLDelete = stmt.executeUpdate(del);
if(SQLDelete == 0)
   throw new Exception("Problem in Delete");

}
session.invalidate();
ns.close();
stmt.close();
con.close();
%>
<p>
<font size="1" color="red">
Confirmation successful <br/>
Thank you
</font>
</p>
<%
} catch(Exception e)
{
ns.close();
con.rollback();
con.close();
session.invalidate();
out.print("Exception raised! Problem!" + e.toString());
}
%>
</body>
</html>
```

Listing 15: Confirmation_OK.jsp

```
<%@ page import="java.util.*" %>
<%@ page import="java.sql.*" %>
<%
session.invalidate();
%>
<jsp:forward page ="conf.html">
```

Listing 16: UserBankDB.txt

```
/*************************************************/
Here we present the database of the User Bank
Currently there are five users , which we used for testing
/*************************************************/

bdanev
danev
drop table ONLY_ONE_ROW
drop sequence usermobilepayments_seq
drop table USERUSERS
drop table USERACCOUNT
drop table USERTRANSACTIONS
drop table USERMOBILEPAYMENTS

create table ONLY_ONE_ROW(val integer)
insert into ONLY_ONE_ROW values (1)

create sequence usermobilepayments_seq increment by 1
create table USERUSERS(
    accountid varchar(10),
```

```
    pinid varchar(4)
   )
create table USERACCOUNT(
   accountid varchar(10),
   customername varchar(50),
   currentbalance number(38,2),
   primary key(accountid)
   )
create table USERTRANSACTIONS (
   accountid varchar(10),
   transaction_date date,
   description varchar(50),
   amount number(38,2)
   )
create table USERMOBILEPAYMENTS (
   mobile_id integer,
   clientname varchar(30),
   clientaccountid varchar(10),
   vshopbankip varchar(30),
   vshopaccountid varchar(10),
   transaction_date date,
   description varchar(50),
   amount number(38,2),
   transaction_id integer,
   vshopip varchar(20)
   )
insert into USERMOBILEPAYMENTS
   values (
      1,'Alok M','1234','urlin1.wv.cc.cmu.edu',
      '1235','15-DEC-2002','Coucouc',200.32,2,'21.212.21.21')
insert into USERUSERS
   values ('1234','1234')
insert into USERUSERS
   values ('1111111111','1111')
insert into USERUSERS
   values ('2222222222','2222')
insert into USERUSERS
   values ('3333333333','3333')
insert into USERUSERS
   values ('4444444444','4444')
insert into USERACCOUNT
   values ('1234','Alok M',1000)
insert into USERACCOUNT
   values ('1111111111','Kenan H',2000)
insert into USERACCOUNT
   values ('2222222222','Boris D',3000)
insert into USERACCOUNT
   values ('3333333333','Danny L',4000)
insert into USERACCOUNT
   values ('4444444444','Chuck C',5000)
insert into USERTRANSACTIONS
   values ('1234','05-NOV-02','Achat BestBuy',200.23)
insert into USERTRANSACTIONS
   values ('1111111111','04-NOV-02','Achat GeantEagle',20.45)
insert into USERTRANSACTIONS
   values ('2222222222','03-NOV-02','Achat Kauffmann',20.56)
insert into USERTRANSACTIONS
   values ('3333333333','05-NOV-02','Achat OfficeDepot',50.99)
insert into USERTRANSACTIONS
   values ('4444444444','05-NOV-02','Achat Target',30.89)
```

A.4.2 Source code of the VShop WebServer

Listing 17: VshopLogIn.html

```
<HTML>
<HEAD><TITLE>VShop  Virtual Bank</TITLE>
</HEAD>
<BODY bgcolor = "blue">
<P>
<font size = "20" face= "Verdana">VSHOP  VIRTUAL BANK
 </font></P>
<P>
<P>
<br>
<FORM name="form1" method="post" action="VshopTestLogIn.jsp">
<P>Customer number </P>
<INPUT type="text" name="accountid" maxlength="10">
<P>PIN</P>
<INPUT type="password" name="pinid" maxlength="4">
<P>
<INPUT type="submit" name="Sign on" value="Sign on">
</FORM>
</BODY>
</HTML>
```

Listing 18: VshopTestLogIn.jsp

```
<%@ page import="java.util.*" %>
<%@ page import="java.sql.*" %>
<%!
String OraDriver= "oracle.jdbc.driver.OracleDriver";
String dbUrl= "jdbc:oracle:thin:@dbclass.intro.cs.cmu.edu:1521:dbintro";
String user;
String password;
String sqlStmtLogIn;
%>

<%
try {

user=request.getParameter("accountid");
session.setAttribute("session",user);
password= request.getParameter("pinid");
boolean entrance = false;

Class.forName(OraDriver);

Connection con = DriverManager.getConnection(dbUrl,"bdanev","danev");
Statement stmt = con.createStatement();


sqlStmtLogIn = "SELECT * FROM VSHOPUSERS WHERE ACCOUNTID='"+user+"' AND PINID='"+
    password+"'";


ResultSet rs = stmt.executeQuery(sqlStmtLogIn);

while(rs.next()){
String dbUser = rs.getString("accountid");
String dbPassword= rs.getString("pinid");
if ((user.equals(dbUser)) && (password.equals(dbPassword)))
```

```
    {
    entrance=true;
    }

}

if(entrance == true){
rs.close();
stmt.close();
con.close();
%>
<jsp:forward page="VshopAccount.jsp"/>
<%}
else
{
rs.close();
stmt.close();
con.close();
%>
<jsp:forward page="VShopLogIn.html"/>
<%}
}

catch (Exception e) {
out.print("Something is wrong" + e.toString());
}
%>
```

Listing 19: VshopAccount.jsp

```
<html>
<body>
<%@ page import="java.util.*" %>
<%@ page import="java.sql.*" %>
<%!
String OraDriver= "oracle.jdbc.driver.OracleDriver";
String dbUrl= "jdbc:oracle:thin:@dbclass.intro.cs.cmu.edu:1521:dbintro";
String user;
String dbName;
String dbBalance;
%>

<%
try {

Class.forName(OraDriver);

Connection con = DriverManager.getConnection(dbUrl,"bdanev","danev");
Statement stmt = con.createStatement();

user=(String)session.getAttribute("session");


ResultSet rs = stmt.executeQuery("SELECT * FROM VSHOPACCOUNT WHERE ACCOUNTID='"+
    user+"'");

while(rs.next()){
dbName = rs.getString("customername");
dbBalance= rs.getString("currentbalance");

}
//out.print("<tr><td>"+ dbName + "</td><td>" + dbBalance+ " $</td></tr>");
```

```
out.println("Hello " + dbName); %>
<P>
<%
out.println("Your current balance is : " + dbBalance + " $");
%>
<P> All your transactions are: </P>
<%
ResultSet ns = stmt.executeQuery("SELECT * FROM TRANSACTIONS WHERE ACCOUNTID='"+user
    +"'");
while (ns.next()) {
String dbTransactionDate = ns.getString("transaction_date");
String dbTransactionDesc = ns.getString("description");
String dbTransactionAmount = ns.getString("amount");
out.print("<tr><td>"+ dbTransactionDate + "    </td><td>" +
dbTransactionDesc+ "  </td><td>"+ dbTransactionAmount + " $</td></tr><BR>");
}

rs.close();
stmt.close();
con.close();
} catch(Exception e) {
out.print("Entrance wrong" + e.toString());
}
%>
<P>
<P>
<a href = "VshopLogOut.jsp">
sign off
</a>

</body>
</html>
```

Listing 20: VshopLogOut.jsp

```
<%@ page import="java.util.*" %>
<%@ page import="java.sql.*" %>
<%
session.invalidate();
%>
<jsp:forward page ="VshopLogIn.html"/>
```

Listing 21: vshopBanqueDB.txt

```
/*************************************************/
Here we present the database of the Vshop Bank.
Currently it contains only one vshop user
/*************************************************/
bdanev
danev
drop table VSHOPUSERS
drop table VSHOPACCOUNT
drop table VSHOPTRANSACTIONS

create table VSHOPUSERS(
   accountid varchar(10),
   pinid varchar(4)
   )
create table VSHOPACCOUNT(
   accountid varchar(10),
   customername varchar(10),
```

```
    currentbalance number(38,2),
    primary key(accountid)
    )
create table VSHOPTRANSACTIONS (
    accountid varchar(10),
    transaction_date date,
    description varchar(50),
    amount number(38,2)
    )
insert into VSHOPUSERS
    values ('1235','1235')
insert into VSHOPACCOUNT
    values ('1235','VShop',200.45)
insert into VSHOPTRANSACTIONS
    values ('1235','24-NOV-02','Achat BestBuy',200.34)
```

A.5 Source code for the Interconnections

A.5.1 InterServerMsg and subclasses

Listing 22: InterServerMsg.java

```
import java.lang.*;
import java.io.*;

public abstract class InterServerMsg implements Serializable {
    public InterServerMsg() {
    }

    public void printDebug() {
   System.out.println("Unknown InterserverMsg.");
    }
}
```

Listing 23: UserPurchaseMsg.java

```
import java.lang.*;
import java.io.*;

public class UserPurchaseMsg extends InterServerMsg implements Serializable {
    private String clientFullName;
    private String clientAccountNo;
    private String vShopBankIP;
    private String vShopAccountNo;
    private String article;
    private float  price;
    private int transactionID;
    private String shopIP;

    public UserPurchaseMsg(String name, String accNo, String vshopbankIp, String
        vshopaccno, String art, float p,int trans, String shop) {
   clientFullName = name;
   clientAccountNo = accNo;
   vShopBankIP = vshopbankIp;
   vShopAccountNo = vshopaccno;
   article = art;
   price = p;
   transactionID = trans;
   shopIP = shop;
```

```
    }

    public void printDebug() {
   System.out.println("UserPurchase: ");
   System.out.println("  Client Name: " + clientFullName);
   System.out.println("  Client Acc: " + clientAccountNo);
   System.out.println("  Shop BankIp: " + vShopBankIP);
   System.out.println("  Shop AccNo: " + vShopAccountNo);
   System.out.println("  Article: " + article);
   System.out.println("  Price: " + String.valueOf(price));
   System.out.println("  TransID: " + String.valueOf(transactionID));
   System.out.println("  ShopIP: " + shopIP);
    }

    public String getClientName(){
   return clientFullName;
    }

    public String getClientAccountNo(){
   return clientAccountNo;
    }

    public String getVshopBankIP(){
   return vShopBankIP;
    }

    public String getVshopAccountNo(){
   return vShopAccountNo;
    }

    public String getArticle(){
   return article;
    }

    public float getPrice(){
   return price;
    }
    public int getTransactionID(){
   return transactionID;
    }

    public String getShopIP() {
   return shopIP;
    }
}
```

Listing 24: ConfirmationMsg.java

```
import java.lang.*;
import java.io.*;

public class ConfirmationMsg extends InterServerMsg {
    private int transactionID;

    public ConfirmationMsg(int id) {
   transactionID = id;
    }

    public int getTransactionID() {
   return transactionID;
    }
```

```
    public void printDebug() {
   System.out.println("TransID: " + String.valueOf(transactionID));
    }
}
```

Listing 25: BankPaymentMsg.java

```
/*********************************************************************
file name: BankPaymentMsg.java
purpose: Object that is exchanged between the User Bank and the
Vshop Bank
*********************************************************************/
import java.lang.*;
import java.io.*;
public class BankPaymentMsg extends InterServerMsg {
    private String vShopAccountNo;
    private String article;
    private float  price;
    private int vshopTransactionID;
    private String vshopIP;
    public BankPaymentMsg(String vshopAccountNo, String art, float p, int id,String
        v) {
   vShopAccountNo = vshopAccountNo;
   article = art;
   price = p;
   vshopTransactionID = id;
   vshopIP = v;
    }

    public void printDebug() {
   System.out.println("BankPayment: ");
   System.out.println("  Vshop Acc: " + vShopAccountNo);
         System.out.println("  Article: " + article);
   System.out.println("  Price: " + String.valueOf(price));
   System.out.println("  VshopTransactionID : "+ String.valueOf(vshopTransactionID));
   System.out.println("  Vshop IP: " + vshopIP);
    }

    public String getVshopAccountNo() {
   return vShopAccountNo;
    }

    public String getArticle() {
   return article;
    }

    public float getPrice() {
   return price;
    }

    public int getVshopTransactionID() {
   return vshopTransactionID;
    }

    public String getVshopIP() {
   return vshopIP;
    }

}
```

A.5.2 SSLClient and classes using it

Listing 26: SSLClient.java

```
import java.io.*;
import java.net.*;
import javax.net.ssl.*;
import javax.net.*;

public class SSLClient {
    public SSLClient(String dest, int port, InterServerMsg msg) {
        SSLSocket sock = null;
        try {
       /* Connect to server */
       SSLSocketFactory sslFact = (SSLSocketFactory)SSLSocketFactory.getDefault();
       sock = (SSLSocket)sslFact.createSocket(dest, port);

       /* Create output and input streams */
       ObjectOutputStream oos = new ObjectOutputStream(sock.getOutputStream());
       BufferedReader in = new BufferedReader(new InputStreamReader(sock.
           getInputStream()));

       /* Send a message */
       oos.writeObject(msg);

       /* Get back acknoledge */
       System.out.println(in.readLine());

       /* Close connection */
       in.close();
       sock.close();
        } catch (EOFException e) {
       System.out.println("EOFException ?");
   } catch (Exception e) {
       System.out.println(e);
   }
    }
}
```

Listing 27: testClient.java

```
import java.lang.*;

public class testClient {
    public static void main(String[] args) throws Exception {
   if (args.length != 2) {
       System.out.println("Usage: java testClient host port");
   } else {
       String dest = args[0];
       int port = Integer.valueOf(args[1]).intValue();

       System.out.println("Trying to connect to " + dest + " on port " + String.
           valueOf(port));
       ConfirmationMsg c = new ConfirmationMsg(1);
       SSLClient s = new SSLClient(dest, port, c);
       System.out.println("MSG SENT !!!");
   }
    }
}
```

Listing 28: SSLUserPurchase.java

```
import java.io.*;
import java.net.*;
import javax.net.ssl.*;
import javax.net.*;

/* Sends a message to a server. used from vshop to bank */
/* usage: java SSLUserPurchase destIp port UserName AccountNo VShopAccountNo
    VShopBankIP article price */

public class SSLUserPurchase {
    public static void main(String[] args) throws Exception {
   String dest = args[0];
   int port = Integer.valueOf(args[1]).intValue();
   float price = Float.valueOf(args[7]).floatValue();
   int transID = Integer.valueOf(args[8]).intValue();

   /* send message */
   UserPurchaseMsg msg = new UserPurchaseMsg(args[2], args[3], args[4], args[5],
       args[6], price, transID, args[9]);
   SSLClient c = new SSLClient(dest, port, msg);
    }
}
```

Listing 29: SSLBankPayment.java

```
/*****************************************************************
file name: SSLBankPayment
purpose: Object that is exchanged between the User Bank and the
Vshop Bank in SSL mode
*****************************************************************/
package ch.cmu.webproject;
import java.io.*;
import java.net.*;
import javax.net.ssl.*;
import javax.net.*;

public class SSLBankPayment {
    public static void main(String[] args) throws Exception {
   String dest = args[0];
   int port = Integer.valueOf(args[1]).intValue();
   String vshopAccountNo = args[2];
   String article = args[3];
   float amount = Float.valueOf(args[4]).floatValue();
   int vshoptransid = Integer.valueOf(args[5]).intValue();
   String vshopIP = args[6];

   /* send message */
   BankPaymentMsg msg = new BankPaymentMsg(vshopAccountNo,article,amount,
       vshoptransid,vshopIP);
   SSLClient c = new SSLClient(dest, port, msg);
    }
}
```

Listing 30: SSLConfirmation.java

```
/*****************************************************************
file name: SSLConfirmation
purpose: Secure confirmation sent from the client bank to the VShop
After this message received in the Vshop, the Vshop can proceed shipping
the product
```

```
*****************************************************************/
import java.io.*;
import java.net.*;
import javax.net.ssl.*;
import javax.net.*;

public class SSLConfirmation {
    public static void main(String[] args) throws Exception {
   String dest = args[0];
   int port = Integer.valueOf(args[1]).intValue();
   int transactionID = Integer.valueOf(args[2]).intValue();

   /* send message */
   ConfirmationMsg msg = new ConfirmationMsg(transactionID);
   SSLClient c = new SSLClient(dest, port, msg);
    }
}
```

A.5.3 SSLServer, ServerThread and subclasses

Listing 31: SSLServer.java

```
/*****************************************************************
file name: SSLServer.java
purpose:   create a SSL on server side to communicate with client
*****************************************************************/
import java.io.*;
import java.lang.*;
import java.net.*;
import java.security.KeyStore;
import javax.net.*;
import javax.net.ssl.*;
import javax.security.cert.X509Certificate;


public class SSLServer {
    private SSLServerSocket listenSocket;
    private int port;
    private Class serverthread;

    public SSLServer(int p, Class s) {
   port = p;
   serverthread = s;
   try {
       SSLServerSocketFactory sslSrvFact = (SSLServerSocketFactory)
           SSLServerSocketFactory.getDefault();
       listenSocket = (SSLServerSocket)sslSrvFact.createServerSocket(port);
   } catch (Exception e) {
       System.out.println(e);
       return;
   }
   while (true) {
       try {
      SSLSocket clientSock = null;
      System.out.println("Server is ready and listening on port " + String.valueOf(
          port));
      clientSock = (SSLSocket) listenSocket.accept();
      ServerThread ns = (ServerThread) serverthread.newInstance();
      ns.setSock(clientSock);
      ns.start();
```

```
        } catch (Exception e) {
       System.out.println(e);
       continue;
        }
    }
     }
}
```

Listing 32: ServerThread.java

```
import java.lang.*;
import java.net.*;
import java.io.*;
import javax.net.ssl.*;

public abstract class ServerThread extends Thread {
    SSLSocket clientSock;
    InterServerMsg msg;

    public void setSock(SSLSocket s) {
   clientSock = s;
    }

    public void run() {
   /* get message */
   try {
       System.out.print("Received a connection from " + clientSock.getInetAddress().
           getHostName());
       System.out.println(", the client port is " + String.valueOf(clientSock.
           getPort()));
       /* Create Streams */
       ObjectInputStream ois = new ObjectInputStream(clientSock.getInputStream());
       PrintStream os = new PrintStream(clientSock.getOutputStream());
       /* get a message (as a java object) */
       Object interServerMsg = ois.readObject();
       Class c = Class.forName("InterServerMsg");
       if (!c.isInstance(interServerMsg)) {
      System.out.println("Received non InterServerMsg !");
      os.println("Received non InterServerMsg !");
      os.close();
      clientSock.close();
       } else {
      ((InterServerMsg) interServerMsg).printDebug();
      /* send ok */
      os.println("ok");
      os.close();
      clientSock.close();
      msg = (InterServerMsg) interServerMsg;
      handleMsg();
       }
   } catch (EOFException e) {
       System.out.println("Connection closed (EOF detected)");
   } catch (Exception e) {
       System.out.println("Connection closed due to errors.");
       System.out.println(e);
   }

    }

    public void handleMsg() {
    }
```

```
}
```

Listing 33: testServer.java

```
import java.sql.*;
import java.lang.*;
import java.io.*;

public class testServer extends ServerThread {
    public static void main(String[] args) throws Exception {
   if (args.length != 1) {
       System.out.println("Usage: java testServer port");
   } else {
       int port = Integer.valueOf(args[0]).intValue();
       /* Open server */
       try {
      SSLServer serv = new SSLServer(port, Class.forName("testServer"));
       } catch (Exception e) {
      System.out.println(e);
       }
   }
    }

    public void handleMsg() {
   msg.printDebug();
    }
}
```

Listing 34: VshopServlet.java

```
import java.sql.*;
import java.lang.*;
import java.io.*;

public class VshopServlet extends ServerThread {
    static Connection con = null;
    public static void main(String[] args) throws Exception {
   /* Open database */
   try {
       Class.forName("com.mysql.jdbc.Driver").newInstance();
   }catch (ClassNotFoundException e){
       System.err.print("ClassNotFoundException: ") ;
       System.err.println(e.getMessage()) ;
       return;
   }catch(IllegalAccessException e){
       System.out.println(e);
       return;
   }catch(InstantiationException e){
       System.out.println(e);
       return;
   }

   /* Start server */
   SSLServer serv = new SSLServer(15494, Class.forName("VshopServlet"));
    }

    /* code called each time a message arrives */
    public void handleMsg() {
   try {
       /* Process it... */
       Class c = Class.forName("ConfirmationMsg");
```

```
        if (!c.isInstance(msg)) {
       System.out.println("Received non ConfirmationMsg !");
       return;
        }
        int trId = ((ConfirmationMsg) msg).getTransactionID();
        /* update database: remove from waiting list and add to shipment list */
        con = DriverManager.getConnection("jdbc:mysql://localhost/mysql?user=root&
            password=root");
        Statement s = con.createStatement();
        String msql = "DELETE FROM VSHOP.WAITING WHERE transactionID=" + String.
            valueOf(trId);
        s.executeUpdate(msql);
        /* I'm not checking if trID exists in the vshop... Perhaps a check should be
            done
           at this level too. */
        msql = "INSERT INTO VSHOP.TOSHIP (transactionID) VALUES (" + String.valueOf(
            trId) + ")";
        s.executeUpdate(msql);
    } catch (SQLException e){
        System.err.println("Error in database" + e);
        return;
    } catch (Exception e) {
        System.out.println(e);
        return;
    }
    System.out.println("OK");
     }
}
```

A.5.4 Other classes used

Listing 35: CurrentDate.java

```
import java.util.*;
public class CurrentDate {
    private String currentDate;

    public CurrentDate() {
   Calendar cal = Calendar.getInstance(TimeZone.getDefault());
   String DATE_FORMAT = "dd-MMM-yyyy";
   java.text.SimpleDateFormat sdf =
       new java.text.SimpleDateFormat(DATE_FORMAT);
        sdf.setTimeZone(TimeZone.getDefault());

   currentDate = sdf.format(cal.getTime());
    }

    public String  getCurrentDate(){

   return currentDate;
    }

}
```