MOBILE WORKFORCE – AMWAY

Suman Kumar Mittapally
MOBILE WORKFORCE - AMWAY

A graduate project submitted to Dakota State University in partial fulfillment of the requirements for the degree of

Master of Science

In

Information Systems

December, 2009

By

Suman Kumar Mittapally

Project Committee:

Dr. Ronghua Shan
Dr. Stephen Krebsbach
Dr. Sreekanth Malladi
We certify that we have read this project and that, in our opinion, it is satisfactory in scope and quality as a project for the degree of Master of Science in Information Systems.

Student Name: Suman Kumar Mittapally

Master’s Project Title: Mobile Workforce - AMWAY

Faculty supervisor: Ronghua Shan Date: 12/10/2009

Committee member: M. Friedman Date: 12/10/2009

Committee member: Stephen Kiley Date: 12/14/09
ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance has been a source of inspiration throughout the course of the project.

I would like to thank Dr. Ronghua Shan for his academic support and encouragement that he has given throughout the project. I express my profound sense of gratitude to Dr. Omar F. El-Gayar, Dean of Graduate Studies and Research for inspiring guidance and encouraging me throughout the project.

I specially thank Dr. Stephen Krebsbach and Dr. Sreekanth Malladi, my committee members for lending their unconditional support; help and cooperation in making this project a success.

Furthermore I would like to thank my family and friends for their moral support and encouragement.
ABSTRACT

More and more organizations are evaluating the best route to mobilize their workers, whether field service personnel, sales representatives or on the move executives.

Amway is the largest multi-level marketing (MLM) organization in the world. In Amway, one is recruited as an "independent" distributor of Amway products by buying a couple of hundred dollars' worth of the products from the one who recruits you, known as your "up line." Every distributor in turn tries to recruit more distributors. Income is generated by sales of products by the distributor plus "bonuses" from sales of his or her recruits and their recruit-descendents.

It goes like this:
If I buy $200 of stuff from Amway this month, I'll get a 3% bonus check (3% of $200 = $6). If I share the opportunity with nine others, and we each buy $200 of stuff from Amway this month, they each were responsible for $200 and will get $6, but I'm responsible for $2000, moving me to the 12% level. I get $240. However, I'm responsible for paying the bonuses of the people right below me - $54 - so I keep $186. I make more because I did more, I found nine people who wanted to buy at a discount and get a bonus for doing it. After I reach the 25% bonus level there are other bonuses that kick in, but they're all based on the volume of product flow, not on signing people up or having lots of people

APPLICATION:
It may not be possible for a distributor (part-time) to remember all the details of the product which he sells. So, mobile enables him to do so by having a catalogue with the product details, distributor information, personalized sales information and distributor application.

The project entitled Mobile Workforce-Amway is an application to assist mobile phone users who are the distributors of Amway organization to access the information regarding the details of Amway products, update the information about his or her personal sales. A distributor can access all the details of the product and make sales accordingly. At the single touch of a button the user is provided with an array of information.
Mobile workforce has the most advanced set of accessibility features for providing users with a flexible, user-friendly interface. Some of the exclusive features of this application include accessing the data at the server and retrieving the data from the database.

A user can show the catalogue on his mobile to the customer which contains the details of the products. When a user selects to launch Amway products on Mobile, he is logged on to a screen where he is asked to select which service he would like to access. Then based upon the users choice the user is presented with the selected service details.

The application uses the latest technology and software. This application is portable since it has been built using the Java language. Mobile workforce is a robust application. This application is free of errors since high quality testing has been applied on each and every module of the application. By building this application I have tried to achieve compatibility, portability and accessibility. Furthermore I would like to conclude by saying that this application can be enhanced to provide futuristic applications since it can accommodate enhancements without much inconvenience.
DECLARATION

I hereby certify that this project constitutes my own product, that where the language of others is set forth, quotation marks, so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I declare that the project describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

Suman Kumar Mittapally.
TABLE OF CONTENTS

ACKNOWLEDGEMENT ........................................................................... iii
ABSTRACT .............................................................................................. iv
DECLARATION ........................................................................................ vi
TABLE OF CONTENTS ........................................................................... vii
LIST OF TABLES ..................................................................................... ix
LIST OF FIGURES .................................................................................... x

1. Introduction ....................................................................................... 1
   1.1 Problem Statement ........................................................................ 1
   1.2 Current Industry Scenario ........................................................... 1
   1.3 Brief about the application ............................................................ 2

2. Literature survey ............................................................................... 3
   2.1 Technology .................................................................................. 3
   2.2 Language ..................................................................................... 27

3. System Overview ................................................................................ 31
   3.1 Introduction ................................................................................ 31
      3.1.1 Product Information: .............................................................. 31
      3.1.2 Sales Information: ................................................................. 31
      3.1.3 Introduction to the project: ..................................................... 31
         3.1.3.1 Distributor Information...................................................... 32
         3.1.3.2 Personalized Sales Information ....................................... 32
         3.1.3.3 Distributor Application .................................................... 32
   3.2 System Overview .......................................................................... 33
   3.3 System requirement section ......................................................... 33

4. Architecture block diagram .............................................................. 38

5. Detail design .................................................................................... 40
   5.1 Overview .................................................................................... 40
   5.2 Architecture Design .................................................................... 46
      5.2.1 Subsystems ........................................................................... 46
         5.2.1.1 User Interaction ............................................................... 47
         5.2.1.2 Application Logic ............................................................ 47
         5.2.1.3 Servlet Programming ..................................................... 47
      5.2.2 Data Flow Diagrams ............................................................. 48
      5.2.3 Persistence ............................................................................ 51
   5.3 UML Diagrams ............................................................................ 55
5.3.1 Use Case Diagrams ................................................................. 58
5.3.2 Class Diagrams ................................................................. 61
5.3.3 Sequence Diagrams ............................................................ 61

6. System testing ........................................................................... 63
   6.1 Testing .............................................................................. 63
   6.2 Test cases ........................................................................ 66

7. Conclusion ................................................................................ 70

WEBSITE REFERENCES .................................................................. 71

BIBLIOGRAPHY ............................................................................ 72
LIST OF TABLES

Table 1: Extensions description table .................................................. 16
Table 2: Abbreviation table ............................................................... 37
Table 3: Login table ........................................................................... 52
Table 4: Product table ......................................................................... 53
Table 5: Category table ....................................................................... 53
Table 6: Customer payment mode table .............................................. 54
Table 7: Distributor payment mode table ............................................. 54
Table 8: Distributor application table .................................................... 55
Table 9: Test case 1 .............................................................................. 66
Table 10: Test case 2 ............................................................................ 67
Table 11: Test case 3 ............................................................................ 67
Table 12: Test case 4 ............................................................................ 68
Table 13: Test case 5 ............................................................................ 69
LIST OF FIGURES

Figure 1: J2ME Architecture ................................................................. 4
Figure 2: Generic Connection Framework ........................................... 7
Figure 3: Development Workstation...................................................... 9
Figure 4: MIDlet Lifecycle .............................................................. 14
Figure 5: Architecture Block Diagram .................................................. 38
Figure 6: Screen 1(Login Screen) ............................................................ 40
Figure 7: Screen 2(List of services) ...................................................... 41
Figure 8: Screen 3(List of services and other links) ............................... 42
Figure 9: Screen 4(Payment screen) ....................................................... 43
Figure 10: Screen 5 contd................................................................. 44
Figure 11: Screen 6(Personalized Sales Information) ............................. 45
Figure 12: Screen 7(Distributor Application) .......................................... 46
Figure 13: Data Flow diagram (Level 0) ................................................ 48
Figure 14: Data Flow diagram (Level 1) ................................................ 48
Figure 15: Data Flow diagram (Level 1.1) .............................................. 49
Figure 16: Data Flow diagram (Level 1.2) .............................................. 49
Figure 17: Data Flow diagram (Level 1.3) .............................................. 50
Figure 18: Data Flow diagram (Level 1.4) .............................................. 50
Figure 19: Use Case diagram 1............................................................ 58
Figure 20: Use Case diagram 2............................................................ 58
Figure 21: Use Case diagram 3............................................................ 59
Figure 22: Use Case diagram 4............................................................ 59
Figure 23: Use Case diagram 5............................................................ 60
Figure 24: Use Case diagram 6............................................................ 60
Figure 25: Class diagram ................................................................. 61
Figure 26: Sequence diagram............................................................. 61
Figure 27: Sequence diagram (Authentication) ...................................... 62
Figure 28: Sequence diagram (Application process) ............................ 62
1. Introduction

1.1 Problem Statement

Now a day’s marketing field plays a leading role in each and every sector. Every distributor and salesperson is busy in their day to day planning. So it may not be possible for a distributor to remember all the details of the product which he sells and how much sales he has made in each day and what are the new products that were manufactured by the industry.

1.2 Current Industry Scenario

The distributor in the existing system uses brochures to give the details of the product to the customer which leads to wastage of money. The distributor has to carry the product with him compulsorily, because if the product which the customer asks is not there with him; he has to go back and get the product with him this may increase the transportation costs, he may also call up to the relevant person to know the details but in this case he has to wait for the other person to get in contact with him. There is also a case where in meanwhile the customer may lose interest in buying the product.

In today’s fast moving world, the promise of wireless computing is compelling. The urgency to access information “anytime, anywhere” is currently fuelling the growing demand from professionals for an engaging and powerful wireless computing experience, complete with new and improved devices, applications, and services. This demand has compelled us in developing this application.

The mobile workforce solution enables the mobile phone service customers to cost effectively upgrade an information technology (IT) infrastructure to take full advantage of mobile technology. This solution allows distributors to know the details about any new information, while connected as though they were in their office.

The Mobile Workforce Amway program provides these services while preserving:

• Information confidentiality
• Rapid and reliable access to information
• Information accuracy – the right information at the right moment
• Manageability
• Costs
Benefits of Mobile Amway Workforce technology:

- Improves customer service responsiveness.
- Increases the efficiency of business processes.
- Exchange information with the field force in real-time.
- Improves the speed and accuracy of data entry from the field.

1.3 Brief about the application

Mobile Workforce – Amway has the most advanced set of accessibility features for providing users with a flexible, user-friendly interface. Some of the exclusive features of this application include accessing the information at the server and providing very rich GUI features for easy user interaction. The Mobile Workforce -Amway application works in the following manner. The information that is requested from the mobile user is maintained in a database that is present at the server end. The user makes the information request by navigating through the GUI provided on the mobile. The http request is sent to the server where in the appropriate servlet is invoked and the response is sent back to the mobile phone.

The features of this application are explained in detail below:

1. One would always feel comfortable to view all the information about the products in his mobile phone instead of carrying brochures. When one selects to launch Mobile Workforce, one is logged on to a screen where one is asked to select the option for which he would like to access the information. After the user is authenticated he is allowed to view the services provided. One has to select a module from the following given modules: product information module, distributor information module, personalized sales information; distributor application form. Upon selecting the module, the user will be allowed to access the particular information.

2. The user can view the product information in that particular module. The user can show this information to his customer with image and description of the product. If the customer is willing to buy the product the user can directly order the product and sell the product through his mobile.

3. Another feature provided for the users is to view the commissions which the distributor (user of the mobile) gets from the sales he made. And he can also introduce new distributors and add them to the Amway tree of distributors through his mobile.
2. Literature survey

2.1 Technology

Architecture of J2ME:

The primary platform targeting the consumer and embedded market is **Java 2 Micro Edition (J2ME)**. J2ME standard is a means of writing java applications for devices such as mobile phones. J2ME consists of the Java virtual machine specification and API specifications. The API specifications are based on Java 2 Standard Edition (J2SE), but modified to meet the unique requirements of each product. In other words, J2ME defines an appropriate set of class libraries and virtual machine technology for a particular class of product.

Depending on the targeted computing platform, Sun has grouped the Java technologies into three editions: Java 2 Micro Edition (J2ME), Java 2 Standard Edition (J2SE) and Java 2 Enterprise Edition (J2EE). Each of these editions has been customized specifically for the platform it is targeting, whether it is a consumer device, a desktop computer or an enterprise network server.

The Java 2 Micro Edition consists of the technology, APIs, tools and standards needed to create applications for consumer devices that are developed by soliciting the i/p on requirements through the Java Community Process (JCP). J2ME specifically targets the consumer space, which covers the range of small commodities such as smart cards and pagers all the way up to the TV set-top boxes.

J2ME provides a complete solution for creating dynamically extensible, networked products and applications for the consumer and embedded appliances.
J2ME ARCHITECTURE

Figure 1: J2ME Architecture

Connected Limited Device Configuration (CLDC)

CLDC defines the base set of Application Programming Interfaces & virtual machine for resource-constrained devices like mobile, PDA & pagers. When coupled with Mobile Information Device Profile (MIDP), it provides a solid java platform for developing applications to run on devices with limited memory, processing power & graphical capabilities. At this time, there are two versions of the CLDC. Version 1.0, released in May of 2000, is known as Java Specification Request (JSR) 30. Version 1.1, currently in public review, is JSR 139. Version 1.0 is the one that is currently shipping in devices.

Objective

The goal of the CLDC specification is to standardize the highly portable, minimum-footprint Java application development platform for resource constrained,
network-connected devices, which is developed through the Java Community Process (JCP). CLDC is designed with the following goals in view:

- Reduce the footprint requirements to levels suitable for mass-market deployment.
- Facilitate application portability by abstracting native system operation into standardized APIs.
- Extend device functionality by allowing dynamic download of applications into device.

**Target Devices**

CLDC is designed to bring the advantages of the Java platform to network-connected devices that have limited processing power, memory such as cell phones, entry level PDA. Target devices have the following capabilities:

- A 16 bit or 32 bit processor with a clock speed of 16MHz or higher.
- At least 160 KB of non-volatile memory allocated for the CLDC libraries & virtual machine.
- At least 192 KB of total memory available for the Java platform.
- Low power consumption, often operating on battery power.
- Connectivity to some kind of network, often with a wireless, intermittent connection and limited bandwidth.

The CLDC specification defines three things:

- The capabilities of the Java virtual machine (VM), which is not a full-featured Java VM.
- A very small subset of the J2SE 1.3 classes. A new set of APIs (Application Programming Interfaces) for input/output called the Generic Connection Framework.

**Generic Connection Framework**

The CLDC does not define any APIs related to user interfaces. The CLDC does not define how applications are loaded onto a device or how they are activated or deactivated. These and other things are defined by the J2ME profiles that use the CLDC as their base. So while it's true that the CLDC does define a complete Java runtime environment, the additional APIs defined by a profile or supplied by the vendor are really necessary to build useful applications.
CLDC does not define the following:

- Application Life-cycle management (installation, launch & deletion of application)
- User interface functionality.
- Event handling.
- High-level application model i.e. high-level user interface.

These features are addressed by the profiles on top of CLDC.

Requirements

Hardware Requirements

CLDC is intended to run on a wide variety of small devices ranging from wireless communication devices like cell phones, organizers & home appliances too. The underlying hardware capabilities varies for these device, hence CLDC does not impose any specific hardware requirements other than memory requirements. The specification assumes that the configuration libraries, profile libraries & applications all fit in the total memory of 160-512 KB. More specifically:

- 128 KB of non-volatile memory is available for virtual machine & CLDC libraries.
- At least 32 KB of volatile memory is available for java runtime & object memory.

Software Requirements

Like hardware requirements software requirements varies considerably. Some of the devices may have full-featured operating systems that support multiple, concurrent operating system & hierarchical file system. Some of the devices do not support such a system. Hence CLDC makes minimal assumptions about the software requirements. The specification assumes a kernel is available to manage the underlying hardware. This OS must provide at least one schedulable entity to run the JVM. The OS need not support separate address spaces or processes, nor guarantee about the real time scheduling or latency behavior.

CLDC - High-level Architecture & security
At the heart of the CLDC implementation is the Java Virtual Machine (JVM). This JVM runs on top of the Host OS. On top of the VM are the libraries comprising CLDC & profiles.

![Diagram of Generic Connection Framework]

**Figure 2: Generic Connection Framework**

**The Virtual Machine**

The Java VM used in the CLDC is restricted in certain important ways when compared to a full-featured J2SE VM. These restrictions allow the VM to fit the memory and power constraints of the small devices that the CLDC target: the CLDC VM and classes can fit in 128K of memory.

The primary restrictions on the VM are:

- **No floating point types**: the floating point support was removed as majority of the CLDC target devices did not have hardware floating point support & the cost of supporting floating point s/w is very high.

- **No object finalization or weak references**: CLDC does not include the method Object finalize(), hence the JVM supporting the CLDC does not support finalization of class instances.

- **No JNI (Java Native Interface) or reflection (hence no object serializations)**: the way VM invokes native functionality is implementation-dependent. This support was eliminated due to the limitation of security & memory constraint.

- **No thread groups or daemon threads** (note that threads are supported, but not thread groups): the JVM supporting CLDC implements multithreading but not thread groups or daemon threads. The thread operations start, stop can be applied only to individual threads.
- **No application-defined class loaders**: the JVM supporting CLDC has built in class loader that cannot be overridden, replaced or re-configured by the user.

- **Error handling limitation**: JVM supporting CLDC supports exception handling, but only a set of error classes included.

Note that CLDC 1.1 relaxes some of these restrictions, in particular re-enabling support for floating-point types and weak references.

The **K virtual machine** is the virtual machine used in the J2ME Connected Limited Device Configuration (J2ME CLDC); Version 1.0. The K virtual machine is a runtime implementation of the Java virtual machine. It is an extremely lean runtime for use in devices with a small memory footprint such as cell phones, two-way pagers, and personal digital assistants (PDAs). The $K$ in K virtual machine stands for kilobyte to indicate this virtual machine works with a total memory of a few hundred kilobytes at most and sometimes fewer than 128 kilobytes.

**Application management**

Many small devices do not have any file system or standardized mechanism for storing the dynamically downloaded information. The CLDC does not require the Java classes downloaded from the external source are stored persistently on the device. Rather the VM might just load the class files & later delete them.

There is always a need to execute the java applications many times without having to download the same over the wireless network over & over again; the user incurs high download expenses.

Due to the significant variations in the capabilities among the CLDC devices the details of application management are device specific & implementation-dependent. The application management capabilities are often written in C or other low-level programming languages specific to the host OS.

**Security**

One of the greatest benefits of Java technology, in the small spaced devices is the dynamic & secure delivery interactive content & applications over different kinds of networks to small devices. As compared to the amount of code devoted to the security in
J2SE, is far less due to the memory availability. Hence some compromises are necessary while defining the security model for the CLDC. The security focuses on two areas:

- Low level VM security
- Application level security

Low level VM Security

Low level VM security implies that a java application executed by the VM does not harm the device in which it is running. The CLDC requires class verification to be done differently. This constraint is guaranteed by the java class file verifier, which ensures that the java byte code & other items stored in the java class files does not reference to invalid memory locations. An off-device class verifier, a process called Preverification, processes class files. At runtime, the VM uses information inserted into the class files by the preverifier to perform the final verification steps. Files that have not been processed by the preverifier are not loaded since they cannot be verified.

Application level Security

Even in the standard java environment, the security provided by the class file verifier is limited, wherein it can only verify if the program is valid java program. Access to the external sources like file system, printer or network etc is beyond the scope of the verifier. Hence an alternative solution is “Sandbox Model”. The sandbox, which runs each application in a tightly constrained environment separated from other applications as well as from underlying system resources. All interactions with native functionality occur
via Application Programming Interfaces (APIs) that are available within the sandbox, rather than directly. J2ME abandons safeguards inherent in other versions of Java in favor of this more limited sandbox approach. The trade off was necessary for J2ME in order to facilitate operation on limited devices such as mobile phones.

According to Sandbox model:

- Java class files have properly been verified & guarantee to be valid java applications.

- Only a predefined set of Java APIs are available to the application programmer, nor can the programmer can download any new libraries containing native functionality, or access any native functions that are not part of the java library or not defined by the CLDC, profiles & licensee open classes.

- The downloading & management of java applications on the device at native code level inside the VM, & no user-definable class loaders are provided, in order to prevent the programmer in overriding the standard classes.

The J2SE Subset

The subset of J2SE 1.3 included in the CLDC consists of classes from these three packages:

- java.lang
- java.io
- java.util

It also specifies a generalized wireless network API in javax.microedition.io. Only selected classes from each package are included: for example, the java.util.Vector and java.util. Hash table classes are included, but none of the collection classes are. The largest package is the java.lang package, which defines the classes that are fundamental to any java application, classes like java.lang.Object or java.lang.Integer. The java.io subset only includes abstract and memory-based classes and interfaces like java.io.DataInput or java.io.ByteArrayInputStream. The java.util subset only includes a few utility classes. Some of the classes are subsets of their J2SE equivalents. Configurations are allowed to remove unnecessary methods or fields, but they cannot add new public or protected methods or fields.
Mobile Information Device Profile (MIDP)

The Mobile Information Device Profile (MIDP) is a set of Java APIs, which, together with the Connected Limited Device Configuration (CLDC), provides a complete J2ME application runtime environment targeted at mobile information devices, such as cellular phones and two-way pagers. The MIDP defines the application architecture for these devices and addresses issues such as user interface, persistence storage and networking.

The MID Profile runtime environment allows to dynamically deploy new applications and services on the end user devices. It is designed to work on top of CLDC and the software and hardware requirements of Mobile Information Devices are in addition to those for the broader range of Connected Limited Devices. The APIs defined by MIDP allows an open application development for Mobile Information Devices. The Mobile Information Device Profile (MIDP), combined with the Connected Limited Device Configuration (CLDC), is the Java runtime environment for today's mobile information devices (MIDs) such as phones and entry level PDAs.

Mobile user interface

MIDP features a high-level user interface API. This enables the developer’s easy to use, high graphical interface & portable applications. MIDP user interface functionality includes predefined screens for displaying edit boxes, lists etc.

Multimedia & Game Functionality

MIDP is ideal for games & multimedia functionality. A low-level user interface API complements the high level API, giving the developers greater control of graphics & inputs. Built-in audio provides support for tones, tone-sequences & WAV file. Multimedia API, an optional package for MIDP, to add video & multimedia rich content.

Extensive Connectivity

MIDP enables developers to fully leverage the native data n/w & messaging capabilities of mobile information devices. It also supports std connectivity like HTTP, HTTPS, datagram, sockets & server sockets. MIDP also supports SMS, Cell Broadcast Service (CBS) GSM & CDMA n/w capabilities though Wireless Messaging API (WMA) optional package.

Over the Air Provisioning
A major benefit of MIDP is its ability to dynamically deploy & update applications over the air (OTA). The MIDP specification defines how MIDP application is discovered, installed, updated & removed on mobile information device. MIDP also enables service provider to identify which MIDP application.

**End-to-End Security**

MIDP provides a robust security to the n/w, applications & mobile information devices. The use of HTTPS leverages existing standards such as SSL & WTLS to enable the transmission of encrypted data. Security domains protect against unauthorized access of data, application, network & other resources by MIDP applications on the device. By default MIDP applications are not trusted. To have privileged access, a MIDP application must be assigned to specific domains that are defined on mobile device, for which the application must be successfully authenticated.

**MIDP Applications (MIDlets)**

The applications written for mobile information devices such as cellular phones and pagers are called MIDlets. Like applets, the software that runs them controls MIDlets. In the case of an applet, the underlying software is a browser or the applet viewer tool and in the case of a MIDlet, the underlying software is the cell phone or two-way pager device implementation that supports the CLDC and MIDP. A MIDlet is a well-behaved MIDP application, which lives within the resource constraints, which runs and terminates when requested.

All the devices that support MIDP are supposed to have a device-specific Application Management Software (AMS) that takes care of installing, managing and removing MIDlets interactively.

**MIDlet Lifecycle**

MIDlets move through a well-defined lifecycle consisting of five phases. It is the task of the Application Management Software to move MIDlets through these phases:

**Retrieval**

The AMS retrieves the MIDlet from some source and reads the MIDlet into the device's memory. The medium through which the MIDlet is downloaded depends on the device. It could be through a serial cable, an IRDA port, or a wireless network.
Installation

Once the MIDlet is downloaded, the AMS installs the MIDlet on the device. During the installation process, the MIDP implementation verifies that the MIDlet does not violate the device’s security policies.

Launching

A MIDlet is launched when a user selects it using the interface provided in the device. At this point, the MIDlet enters the KVM and the lifecycle methods of the MIDlet are invoked.

Version Management

The AMS keeps track of all the MIDlets that are installed on the device including their version numbers. This information is used to upgrade a MIDlet to its new version.

Removal

The AMS removes a MIDlet and cleans up the related resources from the memory. A MIDlet can be in one of the three states after the Application Management Software launches it:

Paused

A MIDlet enters the Paused state once it is created and initialized by the AMS. It can also enter this state when it is Active.

Active

This state means the MIDlet is running normally. A MIDlet goes to the Active state from the paused state if there are no runtime exceptions during its initialization.

Destroyed

This state means the MIDlet has released all its resources and is terminated. A MIDlet can reach this state either from the paused state due to a runtime exception during its initialization or from the active state when the user has chosen to close the application.
Figure 4: MIDlet Lifecycle

Web Server

The Web Server Software is a continuously looping server waiting for requests by web client (browsers or other) for documents over the network. It will parse the request, classify the request and take the corresponding actions. The actions might be executing a script, returning a document, accessing a remote database, etc.

The web server understands the request from the clients i.e. the server & client use same protocol. And the protocol used by web server is Hypertext Transfer Protocol (HTTP). Web (with HTTP) running on top of the TCP/IP Internet network protocol requires TCP port. A port is analogous to a telephone extension. When a client program connects to the server on the network, it requests the server program's port. This port indicates exactly which server program the originating client wants to connect to. Data then is written to the outgoing port (on the client's computer) arrives at the server program's incoming port, where it is read. The port number of the server need not be specified explicitly by the client. Because web server has a default port, if the client does not specific which port to connect to, it will default to port 80. Another word for default port is well known port. Thus, the well-known port of web server is 80. Information Content is the information being served by a web server. It could be data from a local disk, data from database or remote location transparent to the clients.

Traditional Web Server

Traditionally, web servers understand how to receive and reply to HTTP requests, serve up local file information and execute scripts as necessary. A traditional web server
does not understand the content of the documents. In other words, all documents are served as stream of data from the server to the client. In particular, web server does not know about hyper links within the documents, the links are just part of the document from the point of view of a web server. When user clicks on a hyperlink, the browser requests another document from the hyperlink. Thus, the browser "manages" the hyperlink instead of the web server.

Also, the web server does not know about inline images, movie or voice pieces. All these MIME pieces are requested separately from the browser. These requests are just like any other request as far as web server is concerned.

In a way, traditional web server does not have the ability to understand what the content of the documents are. XML can help transforming a traditional web server to a truly interactive one and serve up meaningful contents to different clients.

**Document Tree**

The documents, images, and other information that the web server is to serve as organized into a tree or some form of hierarchy structure. The root of the tree is the starting point with child (node) following it. The web-organized tree is usually different than the actual local file system tree.

**Information Type**

Attempting to show an audio file, as an image will not work, nor treating an ASCII text as image, And the browser has to know ahead of time that a document is a HTML document, so that it can decode the tag into the correct format to display to a user.

First, the web server will send out a header of information telling the client what kind of information is coming before the information arrives.

Content-type: Application/Postscript
Content-encoding: Gzip

Before the web server sends out those metadata of data, it has to find out what kind of data is being sent. This is kind by a convention as follows:

<table>
<thead>
<tr>
<th>Extension</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>html, .htm</td>
<td>HTML document</td>
</tr>
<tr>
<td>.txt</td>
<td>Unformatted ASCII</td>
</tr>
<tr>
<td>.ps</td>
<td>PostScript</td>
</tr>
<tr>
<td>.gif</td>
<td>GIF image</td>
</tr>
<tr>
<td>.mpeg</td>
<td>MPEG image</td>
</tr>
<tr>
<td>.wrl, .vrml</td>
<td>VRML scene description</td>
</tr>
<tr>
<td>.class</td>
<td>Java applet</td>
</tr>
<tr>
<td>.... Etc ...</td>
<td>... Etc ...</td>
</tr>
</tbody>
</table>

Table 1: Extensions description table

**Tomcat**

Tomcat is an open-source application server based on Java technology. In a nutshell, it serves Java Servlets and Java Server Pages (JSP) technology on a Web site. A Tomcat worker is a Tomcat instance that is waiting to execute servlets on behalf of some web server. For example, we can have a web server such as Apache forwarding servlet requests to a Tomcat process (the worker) running behind it.

**Tomcat Security Scope**

As stated in the servlet specification, Tomcat security is primarily concerned with:

- **Authentication** - The means by which communicating entities prove their identities to one another.
- **Access control** - The means by which requests for resources are limited to users or programs.
- **Integrity** - The means used to prove that information has not been modified while in transit.
- **Confidentiality** - The means used to ensure that information is understandable only by authorized users.

Tomcat security is user role-based and web container (somewhat web application) centric.

**Authentication**

By default, the user need not authenticate to access Tomcat resources. Authentication is needed only when specified in the deployment descriptor with the auth-
constrain element. You use a web client (typically a web browser) to authenticate with Tomcat using one of the following mechanisms:

HTTP Basic Authentication
HTTP Digest Authentication
Form Based Authentication
HTTPS Client Authentication

Tomcat is the servlet container used in the official reference implementation for Sun's Java servlet and JSP technologies

Tomcat is a servlet container with a JSP environment. A servlet container is a runtime shell that manages and invokes servlets on behalf of users. Servlet containers can be partitioned into:

Stand-alone servlet containers: These are an integral part of the web server. This is the case when using a Java-based web server, for example the servlet container that is part of the JavaWebServer. Stand-alone is the default mode used by tomcat. Most web servers, however, are not Java-based, which leads us to the next two container types.

**In-process servlet containers:** The servlet container is a combination of a web server plug in and a Java container implementation. The web server plug in: opens a JVM inside the web server's address space and lets the Java container run in it. If a certain request should execute a Servlet, the plug in takes control over the request and passes it (using JNI) to the Java container. An in-process container is suitable for multi-threaded single-process servers and provides good performance but is limited in scalability. This needs a web server adapter to be installed.

**Out-of-process servlet containers:** The servlet container is a combination of a web server plug in and a Java container implementation that runs in a JVM outside the web server. The web server plug in and the Java container JVM communicate using some IPC mechanism (usually TCP/IP sockets). If a certain request should execute a Servlet the plug in takes control over the request and passes it (using the IPCs) to the Java container. The response time of an out-of-process engine is not as good as in the in-process one but the out-of-process engine performs better in many measurable ways (scalability, stability, etc.). This needs a web server adapter to be installed.
Tomcat can be used as either a stand-alone container (mainly for development and debugging) or as an add-on to an existing web server (currently Apache, IIS and Netscape servers are supported). Tomcat is a Java program, and therefore it is possible to execute it from the command line, after setting several environment variables.

**Web Container:**

In World Wide Web technology contexts, a web container comprises essentially the component of a web server that interacts with servlets. The web container has the functionality of mapping a URL to a particular servlet and of ensuring that the process requesting the URL has the correct access rights.

**Java Servlets**

Servlets are modules that run inside request/response-oriented servers, such as Java-enabled web servers, and extend them in some manner. For example, a servlet might be responsible for taking data in an HTML order-entry form and applying the business logic used to update a company's order database. Servlets are to servers what applets are to browsers.

The Servlet API allows servlets to be embedded in many different web servers. Building Web pages on the fly is useful (and commonly done) for a number of reasons:

1. The Web page is based on data submitted by the user. For example the result pages from search engines are generated this way, and programs that process orders for e-commerce sites do this as well.

2. The data changes frequently. For example, a weather-report or news headlines page might build the page dynamically, perhaps returning a previously built page if it is still up to date. The Web page uses information from corporate databases or other such sources. For example, for making a Web page at an on-line store that lists current prices and number of items in stock.

**Servlet Architecture Overview**

The central abstraction in the Servlet API is the Servlet interface. All servlets implement this interface, either directly or, more commonly, by extending a class that implements it such as HttpServlet. The Servlet interface provides for methods that manage the servlet and its communications with clients. Servlet writers provide some or all of these methods when developing a servlet.
When a servlet accepts a call from a client, it receives two objects: one is a 
ServletRequest and the other is a ServletResponse. The ServletRequest class encapsulates the 
communication from the client to the server, while the ServletResponse class encapsulates 
the communication from the servlet back to the client.

The ServletRequest interface allows the servlet access to information such as the 
names of the parameters passed in by the client, the protocol (scheme) being used by the 
client, and the names of the remote host that made the request and the server that received 
it. It also provides the servlet with access to the input stream, ServletInputStream, through 
which the servlet gets data from clients that are using application protocols such as the 
HTTP POST and PUT methods. Subclasses of ServletRequest allow the servlet to retrieve 
more protocol-specific data. For example, HttpServletRequest contains methods for 
accessing HTTP-specific header information.

The ServletResponse interface gives the servlet methods for replying to the client. It 
allows the servlet to set the content length and mime type of the reply, and provides an 
output stream, ServletOutputStream, and a Writer through which the servlet can send the 
reply data. Subclasses of ServletResponse give the servlet more protocol-specific 
capabilities. For example, HttpServletResponse contains methods that allow the servlet to 
manipulate HTTP-specific header information. The classes and interfaces described 
above make up a basic Servlet. HTTP servlets have some additional objects that provide 
session-tracking capabilities. The servlet writer can use these APIs to maintain state 
between the servlet and the client that persists across multiple connections during some 
time period.

**Servlet Lifecycle**

Servers load and run servlets, which then accept zero or more requests from 
clients and return data to them. They can also remove servlets. These are the steps of a 
servlets lifecycle.

When a server loads a servlet, it runs the servlet's init method. Even though most 
servlets are run in multi-threaded servers, there are no concurrency issues during servlet 
initialization. This is because the server calls the init method once, when it loads the 
servlet, and will not call it again unless it is reloading the servlet. The server cannot 
reload a servlet until after it has removed the servlet by calling the destroy method.
Initialization is allowed to complete before client requests are handled (that is, before the service method is called) or the servlet is destroyed.

After the server loads and initializes the servlet, the servlet is able to handle client requests. It processes them in its service method. Each client's request has its call to the service method run in its own servlet thread: the method receives the client's request, and sends the client its response.

Servlets can run multiple service methods at a time. It is important, therefore, that service methods be written in a thread-safe manner. For example, if a service method might update a field in the servlet object, that access should be synchronized. If, for some reason, a server should not run multiple service methods concurrently, the servlet should implement the SingleThreadModel interface. This interface guarantees that no two threads will execute the servlet's service methods concurrently.

Servlets run until they are removed from the service, for example, at the request of a system administrator. When a server removes a servlet, it runs the servlet's destroy method. The method is run once; the server will not run it again until after it reloads and reinitializes the servlet. When the destroy method runs, however, other threads might be running service requests. If, in cleaning up, it is necessary to access shared resources (such as network connections to be closed), that access should be synchronized. During a servlet's lifecycle, it is important to write thread-safe code for destroying the servlet and, unless the servlet implements the SingleThreadModel interface, servicing client requests.

http:
A connection is a path between two devices that utilizes a hard-wired (cable) technology or wireless technology to transmit and receive data over a network. These devices can be a small device and a back-end processing system or two devices. Three pieces of information is required to establish a connection between two devices. They are network address, communication protocol, and communication parameters.

The network address uniquely identifies each device on the network. An Internet Protocol (IP) address is used on many networks, although other addressing schemes exist. A communication protocol is a set of rules that describe how data is transmitted between two devices. Both devices must agree on the communication protocol before transmitting data. There are three widely used communication protocols: Hypertext Transfer Protocol
(HTTP), *File Transfer Protocol (FTP)* and socket.

*Hypertext Transfer Protocol* defines how messages are formatted and transmitted and what actions servers and mobile client should take in response to various commands. HTTP is an application protocol primarily intended for client-server communication. A client is an application program that sets up a connection so that it can send queries. A server is an application program that accepts requests for the set-up of connections over which queries can be answered.

**HTTP Connection in J2ME:**

J2ME enables us to build an application that overcomes the disadvantages of both the exchanging data between a small device and the back-end processing system. The exchange occurs in the real time over a connection created by J2ME application between the small device and the remote computer. This connection is created using the Generic Connection Framework (GCF).

The communication protocol HTTP is supported by MIDP 1.0/2.0. HTTP requires that a client initiate a request for information from a remote computer. The remote computer is typically a server, although any device can respond to a client’s request if the computing device supports HTTP.

The network operating system routes the request to the remote computer, where the request is reviewed and the response is returned to the client. Typically, the content of the response is the requested information or a notification message indicating that the information is unavailable or the client in unauthorized to receive the information, depending on the nature of the application.

GCF optimizes mobile devices by providing a level of network services enabling the device profile to select network protocols and the network services to support. This means devices support only network protocols that they require. However, not all implementations support only network protocol. MIDP implementations are required to support HTTP; therefore we can count on that support.

**Connection and Streams**

Calling the Connector opens a connection `open()` method. The `Connector.open()` method requires one parameter, which is String containing the communication protocol, network address and a communication parameter. These components are separated within the String by a colon. We can disregard the last colon if no parameters are used to open
the connection

    Connection connection = Connector.Open (""");

The Class.forName() method is automatically called at the run time to determine
the class that implements the protocol being used for the connection. The
Connector.Open() method returns an instance of the Connection Interface, which is used
within our J2ME application to transfer information between two computing devices.
Figure below contains the Connection Interface class hierarchy.

The Connector class described in the GCF is used to establish network
connections. The Connector class is used to access one of the seven GCF connection
interfaces provide a basic architecture for the network operations and network protocol
independence for writing network code. GCF connection interfaces are used the same
way regardless of the underlying network protocol.

<table>
<thead>
<tr>
<th>CONNECTION INTERFACE CLASS HIERARCHY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection Interface</td>
</tr>
<tr>
<td>Input Connection interface</td>
</tr>
<tr>
<td>Stream Connection interface</td>
</tr>
<tr>
<td>Content Connection interface</td>
</tr>
<tr>
<td>HTTP Connection</td>
</tr>
<tr>
<td>Output Connection interface</td>
</tr>
<tr>
<td>Stream Connection interface</td>
</tr>
<tr>
<td>Content Connection interface</td>
</tr>
<tr>
<td>HTTP Connection</td>
</tr>
<tr>
<td>Datagram Connection interface</td>
</tr>
<tr>
<td>Stream Connection Notifier interface</td>
</tr>
</tbody>
</table>

These interfaces are Connection, Content Connection, Datagram Connection,
Input Connection, Output Connection, Stream Connection, and Stream Connection
Notifier. All of these are located in the javafx.microedition.io package.

The Connection interface is the basic connection that can either open or close a
connection. The Content Connection interface is used in the streaming connection that
access information located on the web server. The Datagram Connection interface is
used for packet transmission through the use of data gram connection. The Input
Connection interface and Output Connection interface are used to receive and send data
from and to a communications device. The Stream Connection interface is used for two-way transmission using a communication device. The Stream Connection Notifier interface is used when a stream connection is established.

Input/output classes defined in the java.io.package are used to manage transmission over an open connection. It is the input/output classes that write data to an open connection and read data from the open connection. The Input Stream class and Output Stream class are the base classes for all input and output classes.

**J2ME API FOR HTTP**

Interface HTTP Connection:

This interface defines the necessary methods and constants for an HTTP connection. HTTP is a request-response protocol in which the parameters of request must be set before the request is sent. The connection exists in one of three states:

- Setup, in which the request parameters can be set.
- Connected in which request parameters have been sent and the response is expected.
- Closed, the final state, in which the HTTP connection as been terminated.

The following methods may be invoked only in the Setup state:

- setRequestMethod
- setRequestProperty

The transition from Setup to Connected is caused by any method that requires data to be sent to or received from the server.

The following methods cause the transition to the connected state when the connection is in Setup state.

- openInputStream
- openDataInputStream
- getLength
- getType
- getEncoding
- getHeaderField
- getResponseCode
• getResponseMessage
• getHeaderFieldInt
• getHeaderFieldDate
• getExpiration
• getDate
• getLastModified
• getHeaderField
• getHeaderFieldKey

The following methods may be invoked while the connection is in Setup or Connected state.

• close
• getRequestMethod
• getRequestProperty
• getURL
• getProtocol
• getHost
• getFile
• getRef
• getPort
• getQuery

After an output stream has been opened by the openOutputStream or openDataOutputStream methods, attempts to change the request parameters via setRequestMethod or the setRequestProperty are ignored. Once the request parameters have been sent, these methods will throw an IOException. When an output stream is closed via the OutputStream.close or DataOutputStream.close methods, the connection enters the connected state. When the output stream is flushed via the OutputStream.flush or DataOutputStream.flush methods, the request parameters MUST be sent along with any data written to the stream.

The transition to Closed state from any other state is caused by the close method and the closing all of the streams that were opened from the connection.
**Interface Connection**

This is the most basic type of generic connection. Only the close method is defined. The open method is not defined here because opening is always done using the Connector.open() methods.

**Interface Content Connection**

This interface defines the stream connection over which content is passed.

**Class ConnectionNotFoundException**

```java
java.lang.Object
  
  +-- java.lang.Throwable
  
    +-- java.lang.Exception
  
      +-- java.io.IOException
  
        +-- javax.microedition.io.ConnectionNotFoundException
```

This class is used to signal that a connection target cannot be found.

**Class Connector**

```java
java.lang.Object
  
  +-- javax.microedition.io.Connector
```

This class is a factory for creating new Connection objects. The creation of Connections is performed dynamically by looking up a protocol implementation class whose name is formed from the platform name (read from a system property) and the protocol name of the requested connection (extracted from the parameter string supplied by the application programmer.) The parameter string that describes the target should conform to the URL format as described in RFC 2396. This takes the general form:

```
{scheme}: [{target}][{parms}]
```

- where `{scheme}` is the name of a protocol such as *http*.
- The `{target}` is normally some kind of network address.

Any `{parms}` are formed as a series of equates of the form "; x=y". Example: "; type=a". An optional second parameter may be specified to the open function. This is a
mode flag that indicates to the protocol handler the intentions of the calling code. The options here specify if the connection is going to be read (READ), written (WRITE), or both (READ_WRITE). The validity of these flag settings is protocol dependent. For instance, a connection for a printer would not allow read access, and would throw an IllegalArgumentException. If the mode parameter is not specified, READ_WRITE is used by default.

An optional third parameter is a boolean flag that indicates if the calling code can handle timeout exceptions. If this flag is set, the protocol implementation may throw an InterruptedException when it detects a timeout condition. This flag is only a hint to the protocol handler, and it does not guarantee that such exceptions will actually be thrown. If this parameter is not set, no timeout exceptions will be thrown.

**Simplified Stream Methods on Connector**

The Connector class defines the following convenience methods for retrieving an input or output stream directly for a specified URL:

- `InputStream openInputStream (String url)`
- `DataInputStream openDataInputStream (String url)`
- `OutputStream openOutputStream (String url)`
- `DataOutputStream openDataOutputStream (String url)`

Please be aware that using these methods implies certain restrictions. You will not get a reference to the actual connection, but rather just references to the input or output stream of the connection. Not having a reference to the connection means that you will not be able to manipulate or query the connection directly. This in turn means that you will not be able to call any of the following methods:

- `getRequestMethod ()`
- `setRequestMethod()`
- `getRequestProperty()`
- `setRequestProperty()`
- `getLength()`
- `getType()`
- `getEncoding()`
• getHeaderField()
• getResponseCode()
• getMessage()
• getHeaderFieldInt
• getHeaderFieldDate
• getExpiration
• getDate
• getLastModified
• getHeaderField

2.2 Language
The JAVA Programming Language and JDBC:

A JAVA program, written properly and according to specification, can run on any JAVA technology enabled platform without recompilation. The JAVA programming Language is completely specified and, by definition, a JAVA technology enabled platform must support a known core of libraries. One such library is java.sql package or JDBC, which you can think of as a portable version of ODBC, and is itself a major standard. Using the JAVA programming language in conjunction with JDBC provides truly portable solution to writing database applications.

JDBC DRIVER

JDBC DRIVER is a class that implements the JDBC Driver interface and understands how to convert program (and typically SQL) requests for a particular database. Clearly, the driver is what makes it all work.

JDBC Application Programming Interface

The JDBC API is broken into two parts: The Core API and The JDBC Optional Package. In general the JDBC core API adds a few more classes to its previous version, but is primarily concerned with performance, class enhancements and functionality, and the new SQL3 (also known as SQL99) data types.

The functionality in the core API includes scrollable results sets, batch updates, programmatic inserts, deletes and updates, performance hints, character streams for streams of internationalized Unicode characters, full precision for java.Math decimal values and support for time zone in date, time and timestamp values.
Connecting to the Database

There are always two steps to making a database connection using the driver manager:

a. **Load the JDBC Driver**

   You must load a driver that enables the jdbc classes to communicate with the data source. Here is a standard method for dynamically loading a driver:
   
   ```java
   Class.forName(DriverClassName);
   ```

   A standard jdbc compliant driver should also create a new instance of the driver class with this code. Unfortunately, in practice this doesn’t work for all the cases. For this reason the following code is used
   
   ```java
   Class.forName(DriverClassName).newInstance();
   ```

b. **Connect to a data source**

   The driver supplies methods to make a connection, but requires a specific type of URL, which uses the jdbc protocol. The generalized form is
   
   ```java
   Jdbc: <subprotocol>:<subname>
   ```

   Using the driver manager class, you request a connection using the passed URL and the driver manager selects the appropriate driver.

   Here is the standard form of the connection request:
   
   ```java
   Connection con = DriverManager.getConnection(URL, Username, Password);
   ```

   This form is best for portability even in cases where username and password are empty string (""") due to a database default or say text files acting as ODBC data sources, which cannot make use of such attributes.

Creating a Table:

While the connection class has a number of capabilities in order to use DDL or Data manipulation language (DML) SQL statements, a statement object is required so the next step is to ask the connection for a statement object:

```java
Statement stmt = con.createStatement();
```

**Connecting a JAVA program to a database**

A connection object represents and controls a connection to a database while everything in JDBC depends on the capabilities of the database and the JDBC driver, in general you can have multiple connections to a same database and/or connections to the
multiple databases. The driver manager class handles driver registration and provides methods for obtaining a connection.

One of the first steps in obtaining a connection is often the most important: how to set up the database URL?

Jdbc: <subprotocol> <subname>, with the <subprotocol>: identifying the machine or server and the <subname> essentially identifying the database. In practice, the content depends on the specific driver and can be bewildering, ranking along with the classpath problems in producing "No suitable drivers" errors. This is fairly straightforward, primarily because the client and the server run on the same machine.

Most DBMS engines that support remote (and even local) connections do so using a TCP/IP (Transmission Control Protocol / Internet Protocol) port. Like any other socket program, the DBMS engine is free to decide what port it wants to use. While TCP/IP is generally the norm, other communication protocols may be used. DB2, for example, can also use APPC (Advanced Program to Program Communication) on several platforms.

When applications attempt to connect to a network or internet server, identification / location information must be provided. The general JDBC way is to use // host: port / sub name, where host is an IP address or DNS (Domain Name Service) or other locatable name.

**Statements, Result set and interacting with a Database**

A statement object is a container or transport mechanism to send / execute (normally) SQL statements and retrieve any results via its associated connection. In areas controlled by the connection interface there are three types of statements, including prepared statements and Callable statements, both of which are sub interfaces of statements. You don’t need to create a new instance of statement, but instead, request the associate connection to create one:

```
Statement stmt = con.createStatement();
```

The execute series are the most often used of statement’s methods:

- `executeQuery()` is used to execute SQL statements that return a single `ResultSet`.
- `executeQuery()` is used for statements that return a `ResultSet`, basically a SELECT statement.
executeUpdate() is used to execute SQL statements that modify a table or values of columns in a table and return the number of rows affected (which is zero in the case of DDL statements).

executeUpdate() returns an int containing the affected row count for INSERT, UPDATE or DELETE statements, or zero for SQL statements that do not return anything, like DDL statements.

execute() can be used to execute any type of SQL statement, but is intend for those that can return multiple results or values.

To allow the most flexibility to work with various databases and datasources, JDBC places no restriction on the kinds of SQL statements that a statement can send.

A statement only keeps the one ResultSet open at a time and often reuses the same ResultSet for new data. You should be sure to get all the data required from the ResultSet before executing another query via its associated statement. A statement should automatically close() the ResultSet on execution and on Statement.close(), but you may want to close the ResultSet yourself as soon as its data is no longer needed.

Data Navigation

ResultSet.next() returns a Boolean: true if there is a next row and false if not (meaning the end of the data/set has been reached). Conceptually, a pointer or cursor is positioned just before the first row when the ResultSet is obtained. Invoking next() moves to the first row, then the second row and so on.

If(result.next())
The if-statement collects the data. After that, a loop while(result.next()) is used, to allow the program to continue to the end of the data.

Data Extraction

Once positioned at a row, the application can get the data on a column-by-column basis using the appropriate ResultSet.getXXX method.
3. System Overview

3.1 Introduction

What is Mobile Workforce - Amway (MWA)?

Mobile Workforce - Amway is a J2ME based wireless mobile application which when installed in a Java enabled mobile phone lets a user access accurate and reliable information from his/her mobile phone with the click of a button.

It provides the user with the updated information about the products. It helps him to do sales quick and easily.

Potential of Mobile Workforce - Amway (MWA)

In this era of telecommunication, mobile network operators are finding innovative ways to increase their profits. One innovative way is providing personalized services (services according to individual demand). One of the most attractive ways of providing personalized service is through Mobile Workforce - Amway. It caters mainly to the distributors of Amway. The distributor can make sales easily with the use of this mobile handy-device.

3.1.1 Product Information:

According to the service selected by the user, the MWA application would provide the required information to the user. The MWA application would allow the distributor (user) to make sales by showing the details of the product to his customer. The customer can view and select the product which he wants to buy from the distributors mobile.

3.1.2 Sales Information:

The distributors can also update the information in the server through the mobile. This also involves viewing how many subordinates he has and how much commission he gets. This helps in knowing about his personalized sales information in ease.

3.1.3 Introduction to the project:

- Purpose of this section.

This SRS describes the functional and performance requirements allocated to the project Mobile Workforce-Amway. This SRS describes the scope of the project.

- Intended Audience.
This document intended for project developers of Mobile Workforce-Amway and for review by the technical guide of the project. It would also be used by the QA group for facilitating the development and testing process.

- **Scope of the development project.**

The project City on Mobile is being developed to provide the user (customer) with personalized services on his mobile phone. So, one of the most exciting ways for providing this service is Mobile Workforce. Services will be offered to the user based upon the customer’s request.

The product information will be displayed on the user’s screen whenever the user wants the customer to view. The product information will be displayed from the server database. The retrieving of information will be done through servlets. When the payment is made the transaction will be updated into the database of the server. This feature helps in maintaining the database consistently. The services offered by the Mobile Workforce are:

3.1.3.1 **Distributor Information**

This Module helps in knowing the details of the down line distributors.

3.1.3.2 **Personalized Sales Information**

In this module the data can be inserted and at the same time can be retrieved. Servlets are used in getting and inserting the data.

3.1.3.3 **Distributor Application**

This module helps a customer to join the Amway organization as a distributor. The distributor can later submit the details of the new distributor to Amway organization.

- **Organization of the document.**

It is organized as the following:

Section 1: Introduction - high-level description of the document

Section 2: System Overview

Section 3: MWA SR section: GUI Flow and System Requirements

- **Conventions**

As a means of tracking each requirement, the following naming convention has been used:

< MWA1.0-FC-SB-NNN >

Where MWA 1.0 signifies Release 1.00 of the Mobile Workforce Amway application.

FC is a unique 2 to 3 character designation of the functional component, as follows:
GEN for General requirements that will be applicable for all components

DIS for Discovery Component

COM for Communication Component

PRO for Profile Management Component

GUI for Graphical User Interface

SB is an optional, unique, 2 to 3 character designation of a subcomponent of the functional component.

NNN is the requirement number (may be as many digits as needed for uniqueness).

3.2 System Overview

J2ME System Architecture:

The primary platform targeting the consumer and embedded market is Java 2 Micro Edition (J2ME). J2ME standard is a means of writing java applications for devices such as mobile phones. J2ME consists of the Java virtual machine specification and API specifications. The API specifications are based on Java 2 Standard Edition (J2SE), but modified to meet the unique requirements of each product. In other words, J2ME defines an appropriate set of class libraries and virtual machine technology for a particular class of product. Depending on the targeted computing platform, Sun has grouped the Java technologies into three editions: Java 2 Micro Edition (J2ME), Java 2 Standard Edition (J2SE) and Java 2 Enterprise Edition (J2EE). Each of these editions has been customized specifically for the platform it is targeting, whether it is a consumer device, a desktop computer or an enterprise network server.

3.3 System requirement section

MWA General Requirements

☐ MWA Software Requirements

☐ MWA1.0-GEN-1001 Language

MWA application would be using Java language.

<END>
☐ MWA1.0-GEN-1002 Platform

MWA application would be based on Windows 9x.

<END>

☐ MWA1.0-GEN-1003 API

MWA application would be using the J2ME API provided by Sun Microsystems.

<END>

MWA1.0-GEN-1004 Unlimited Usage

MWA application would be offered to the user as long as he is a valid distributor of the Amway Company.

<END>

MWA1.0-GEN-1005 Transferable

MWA application is transferable from one mobile device to another.

<END>

MWA Hardware Requirements

MWA1.0-GEN-1006 Client End

MWA application’s client end is a Java enabled mobile phone supporting MIDP 2.0 applications.

<END>

General Screen Layout

MWA1.00-GUI-1007Screen Components

All screens shall have the following in common:

1. Screen name (title) on upper left corner.
2. Content display area.
3. Select button between the OK and BACK buttons. This screen helps the user in choosing and selecting the option.
4. OK button on the lower right corner. This button allows the user to carry on with the application.
5. BACK button on lower left corner. This button allows the user to go back to the previous screen.

<END>

User Interface Screens Description
This project is being developed for all the distributors of Amway provided with a mobile phone. The mobile phone would host the client interface in the form of screens. The distributor would navigate the screen with the help of keypad on his phone. There are altogether 15 screens in this application. A brief explanation is as follows.

**MWA1.00-GUI-1008 Welcome Screen**

This is the Welcome Screen. This screen welcomes the user to AMWAY. The distributor can start using the application by using the ENTER button.

<END>

**MWA1.00-GUI-1009 Login Screen**

This is the login screen where in the distributor enters his loginid and password. Based upon the valid or invalid details the distributor is connected to the server or rejected respectively. If the distributor details are entered wrong, a message saying invalid user and password is given. The user can go back and enter once again or exit by clicking EXIT.

<END>

**MWA1.00-GUI-1010 Main Screen**

This is the Main Screen. This screen lets the distributor select any one module from the list of modules. From here the distributor can either exit or select any one module. The modules are:

- Product information
- Distributor information
- Personalized sales
- Distributor application

From here the user can either go back to the main screen or select the option he wishes to carry on with the application.

<END>

**MWA1.00-GUI-1011 Module Selection Screen**

This screen is Module selection screen of product information. Based upon his choice in the main screen the user would be presented with this screen. This screen interface provides the user of retrieving product information by giving a list of products with the images. From this screen the user can either go back to the main screen or select
any one product. On selecting the product the associated description are shown. The products provided are:

- Carwash
- Dishdrop
- Leather
- Seespray

MWA1.00-GUI-1012 Module Selection Screen

This screen is the Module selection screen of distributor information. If the user selects the option as distributor information in the main screen then he comes to this screen. Here also he can go back to the main screen or can select any one option by using BACK or OK respectively. The screen provides the user with the below:
1. List of distributors
2. Distributor information

On selecting a distributor from the list the information of the selected distributor is given.

MWA1.00-GUI-1013 Module Selection Screen

This is the Module selection screen of personalized sales. If the user selects the option as personalized sales in the main screen then he comes to this screen. Here also he can go back to the main screen or can select any one option by using BACK or OK respectively. The screen provides the distributor with the form where he has to insert the details about the sales he has done.

MWA1.00-GUI-1014 Module Selection Screen

Screen 6: This screen is the distributor application selection screen. If the user selects the option as distributor application in the main screen then he comes to this screen. Here also he can go back to the main screen or can select any one option by using BACK or OK respectively. The screen provides the distributor with a form wherein he has to fill the details of the customer who is interested in becoming a distributor. This form is submitted to the Amway server.

<END>
Abbreviations

MWAMobile Workforce Amway GUI:Graphical User Interface

<table>
<thead>
<tr>
<th>MWA</th>
<th>Mobile Workforce Amway</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>SRS</td>
<td>Software Requirement Specification</td>
</tr>
<tr>
<td>J2ME</td>
<td>Java 2 Micro Edition</td>
</tr>
</tbody>
</table>

Table 2: Abbreviation table

References

- www.forum.nokia.com
- www.sun.java.com
- www.mobilein.com
- www.google.com
4. Architecture block diagram

A REQUEST RESPONSE SCENARIO

HTTP REQUEST

HTTP RESPONSE

Figure 5: Architecture Block Diagram

Http Server: Handles the http requests and returns relevant information in form of http responses by fetching data from a database.

J2ME Enabled Mobile Phone: Serves as the Client.

The Client application, which is J2ME, sends a HTTP request to the server.

The server will send HTTP response to the client.

The architecture of Mobile Workforce-Amway is described below:

- User launches the application Mobile Workforce-Amway to access information about the products, his personalized sales information through his mobile.

- When a user selects to launch the application, the web browser builds a HTTP request and sends it to the web server through the communication network. HTTP is the protocol used for communication with the server. The HTTP contains a request and a response. The request is a string that contains a URL that is used for identifying a web component.

- The Web server waits for requests from the clients, processes them and sends the reply.

- The web server receives the request, converts it into a request object and sends it to the appropriate web component. A web server contains a web container, which is the component that interacts with the servlets. The main function of web
container is to map the _URL_ from the client to a particular server and ensure that the receiving _URL_ has the correct access rights.

- The servlets interact with the database for retrieving information based upon the chosen service that is retrieved from the database.

- The web container fills in the response object and sends it to the web server which converts it to a http response and sends it back to the client, including a MIME-like Content-Type i.e. header that tells the client what kind of data follows the header. If the content type is recognized, the client browser will display the content as appropriate.
5. Detail design

5.1 Overview

When the user launches the application—Mobile Worforce-Amway, an Amway screen is displayed. If the user selects OK then he is presented with the login form, where the user has to enter his/her loginid and password which is mandatory. Upon giving the loginid and password the MIDlet sends a request to the server to verify whether the user is a valid user or not.

![Image of Amway screen]

**Figure 6: Screen 1 (Login Screen)**

The next screen presents the user with a list of services being offered based upon the choice in the main screen. When the user gets authenticated he will be entered into the next screen. The next screen presents the user with a list of modules.
Product information:
When the user selects the first module i.e. product information module, the category of products is displayed. Upon selecting the category, the products under that category are displayed. When the user selects the product, the MIDlet sends a request to the server to get the data. The server refers to the servlet, which is the server-side programming and the servlet contains the JDBC connection by which it is going to retrieve the data from the database according to the servlet programming and after getting request from the MIDlet, the servlet is processed and corresponding value is retrieved from the database and servlet is going to send back the response to the MIDlet. With the help of servlet, the data is dynamically retrieved from the database. Depending upon the service and the product selected by the user the required information is dynamically generated from the database and displayed on the screen.
Figure 8: Screen 3 (List of services and other links)

The user will be entered into the payment mode with the customer request. The payment mode consists of two modes, distributor mode and customer mode.
Figure 9: Screen 4(Payment screen)

Distributor Information:
When the user selects the distributor information module, MIDlet sends a request to the server to get the data. The information displayed will be about the user (distributor’s information) i.e the commission he gets.
Figure 10: Screen 5 contd...

**Personalized Sales Information:**

The Distributor Personalized sales which include how many products he has sold etc.
Figure 11: Screen 6 (Personalized Sales Information)

When the user chooses an option, he is provided with the relevant information of that particular category. For the user’s convenience there are two options- BACK and OK. Choosing the BACK option will take the user to the previous screen and choosing the OK option will take the user to get the details of the particular category selected.

**Distributor Application:**

When the user selects the module, distributor application, he can allow a customer to become a distributor. The user can also delete a distributor from the Amway distributor list with the given distributor’s id.
Figure 12: Screen 7(Distributor Application)

The user can select the category which is given as a set of ImageItems using itemCommandListener interface. And can activate the commands using CommandListener interface. The actions done by the ImageItems are written in CommandAction. CommandAction used by the ImageItems take Command and Item as parameters. The application which has to be submitted to the server is done through a servlet.

5.2 Architecture Design

5.2.1 Subsystems

Every project is divided into individual blocks or modules each performing an individual task with each block behaving as a building block of the application and contributing to the working of the application. Each of these blocks or modules is termed as a subsystem of a project.
5.2.1.1 User Interaction

The technology used at the front end is J2ME. The client end deals with rich GUI, navigation of the user and the display of the requested information. The information may vary from text files, pictures or contact information from the phone book located at the server. The navigation at the client end involves selection of the file from a list of files.

5.2.1.2 Application Logic

The connectivity deals with establishing a connection with the server, sending and receiving the requests to and from the server. The connectivity is done using HTTP, which is a very secured networking protocol for connecting to a server. All the requests and the responses are carried on the network using HTTP.

5.2.1.3 Servlet Programming

The server end deals with receiving the requests from the mobile phone, processing them, and sending back the responses. The responses of the server include sending back a list of filenames, contents of a file or the contact information of a person. The major function of the server is to forward the selected file from the VD to the mailbox of the recipient selected by the user. The server end is maintained using an Apache Tomcat server. It is an open source application server based on java technology.
5.2.2 Data Flow Diagrams

Figure 13: Data Flow diagram (Level 0)

Figure 14: Data Flow diagram (Level 1)
Figure 15: Data Flow diagram (Level 1.1)

Figure 16: Data Flow diagram (Level 1.2)
Figure 17: Data Flow diagram (Level 1.3)

Figure 18: Data Flow diagram (Level 1.4)
5.2.3 Persistence

Oracle and SQL

What is SQL?

*SQL*, the *Structured Query Language* is an ANSI (American National Standards Institute) Standard for accessing database systems. SQL statements are used to retrieve and update data in the database. SQL works with database programs like Access, DB2, Informix, Microsoft SQL Server, Oracle, Sybase and many others.

**SQL Queries**

With SQL, the user can query a database and have a result returned in a tabular form.

**SQL Data Manipulation:**

As the name indicates, SQL is syntax for executing queries. But the SQL language also includes syntax to update records, insert new records and delete existing records. These query and update commands together form the Data Manipulation Language (DML) part of SQL

- SELECT - extracts data from a database.
- UPDATE - updates data in a database.
- DELETE - deletes data from a database.
- INSERT - inserts new data into a database.

**SQL Data Definition:**

The Data Definition Language (DDL) part of SQL permits database tables to be created or deleted. Also indexes can be defined, and links can be specified between tables.

The most important DDL statements in SQL are:

- CREATE TABLE - Creates a new database table.
- ALTER TABLE - Alters (changes) a database table.
• DROP TABLE - Deletes a database table.
• CREATE INDEX - Creates an index (search key)
• DROP INDEX - Deletes an index.

Application of mobile workforce requires following database tables:

1. Login
2. Product
3. Category
4. Pemode
5. Pdmode
6. Dappl

**Login table:**

The table Login is used to get the information of the present distributors in the application.

<table>
<thead>
<tr>
<th>Name</th>
<th>Null</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGIN ID</td>
<td>NOT NULL</td>
<td>VARCHAR2 (20)</td>
</tr>
<tr>
<td>PASSWORD</td>
<td>NOT NULL</td>
<td>VARCHAR2 (20)</td>
</tr>
<tr>
<td>PHNO</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>ADDRESS</td>
<td></td>
<td>VARCHAR2 (20)</td>
</tr>
</tbody>
</table>

**Table 3: Login table**

*The table contains the attributes:*

LOGIN ID: Unique id which is given to a distributor.

PASSWORD: Password of a particular distributor.

PHNO: Phone number of a distributor.

ADDRESS Address of a distributor.

**Product table:**

The product table contains the details of the products. Description of this table is shown below.

<table>
<thead>
<tr>
<th>Name</th>
<th>Null</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCT ID</td>
<td>NOT NULL</td>
<td>VARCHAR2 (25)</td>
</tr>
<tr>
<td>PRODUCT NAME</td>
<td></td>
<td>VARCHAR2 (25)</td>
</tr>
<tr>
<td>CATEGORY ID</td>
<td>NOT NULL</td>
<td>NUMBER(20)</td>
</tr>
<tr>
<td>Name</td>
<td>Null</td>
<td>Type</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>PRICE</td>
<td></td>
<td>NUMBER(20)</td>
</tr>
<tr>
<td>QU</td>
<td></td>
<td>VARCHAR(4)</td>
</tr>
<tr>
<td>QUANTITY</td>
<td></td>
<td>VARCHAR(10)</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td></td>
<td>VARCHAR(400)</td>
</tr>
</tbody>
</table>

**Table 4: Product table**

The table contains the attributes:

- **PRODUCT ID**: Id given to a product.
- **PRODUCT NAME**: Name of the product.
- **CATEGORY ID**: This is the id given for each category.
- **PRICE**: Price of a product.
- **QU**: Quality of the product.
- **QUANTITY**: Quantity of a product i.e. 500ml or 1000ml.
- **DESCRIPTION**: Detail description of the product.

**Category table:**

Category table consists of the category of products. Here the category is homecare, cosmetics, nutrition, and personal care.

Description of this table is shown below.

<table>
<thead>
<tr>
<th>Name</th>
<th>Null</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATEGORY ID</td>
<td>NOT NULL</td>
<td>NUMBER(10)</td>
</tr>
<tr>
<td>CATEGORY NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2 (10)</td>
</tr>
</tbody>
</table>

**Table 5: Category table**

The table contains the attributes:

- **CATEGORY ID**: This is the id given for each category.
- **CATEGORY NAME**: Gives the name of a category.

There are two categories in payment mode. There are customer mode and distributor mode.

**Pcmode:**

This table gives the details of the customer payment mode. Description of this table is shown below.
<table>
<thead>
<tr>
<th>Name</th>
<th>Null</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOMERID</td>
<td>NOT NULL</td>
<td>VARCHAR2 (20)</td>
</tr>
<tr>
<td>CUSTOMER NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2 (30)</td>
</tr>
<tr>
<td>PRODUCT ID</td>
<td>NOT NULL</td>
<td>VARCHAR2 (20)</td>
</tr>
<tr>
<td>NO_OF_PROD</td>
<td>NOT NULL</td>
<td>NUMBER (10)</td>
</tr>
<tr>
<td>PMODE</td>
<td>NOT NULL</td>
<td>VARCHAR (20)</td>
</tr>
</tbody>
</table>

**Table 6: Customer payment mode table**

*The table contains the attributes:*

- CUSTOMER ID: The id given to each customer
- CUSTOMER NAME: Name of a customer.
- PRODUCT ID: Id given to a product.
- NO_OF_PROD: The number of products the customer wants to buy.
- PMODE: The mode the customer can select. There are three modes which the customer can select: cash, credit, cheque.

<table>
<thead>
<tr>
<th>Name</th>
<th>Null</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGIN ID</td>
<td>NOT NULL</td>
<td>NUMBER (20)</td>
</tr>
<tr>
<td>PRODUCT ID</td>
<td>NOT NULL</td>
<td>NUMBER (20)</td>
</tr>
<tr>
<td>NO_OF_PROD</td>
<td>NOT NULL</td>
<td>NUMBER (30)</td>
</tr>
<tr>
<td>PMODE</td>
<td>NOT NULL</td>
<td>VARCHAR (10)</td>
</tr>
</tbody>
</table>

**Table 7: Distributor payment mode table**

*The table contains the following attributes:*

- LOGIN ID: Unique id which is given to a distributor.
- PRODUCT ID: Id given to a product.
- NO_OF_PROD: The number of products the customer wants to buy.
- PMODE: The mode the customer can select. There are three modes which the customer can select cash, credit or cheque.
Dappl:

The Dappl table contains the details of a distributor:

<table>
<thead>
<tr>
<th>Name</th>
<th>Null</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRSTNAME</td>
<td>NOT NULL</td>
<td>VARCHAR2 (20)</td>
</tr>
<tr>
<td>SUP_ID</td>
<td>NOT NULL</td>
<td>VARCHAR2 (30)</td>
</tr>
<tr>
<td>ADDRESS</td>
<td>NOT NULL</td>
<td>NUMBER(10)</td>
</tr>
<tr>
<td>CONTACT NO</td>
<td>NOT NULL</td>
<td>VARCHAR2 (25)</td>
</tr>
<tr>
<td>PACKAGEDGIVEN</td>
<td>NOT NULL</td>
<td>NUMBER(10)</td>
</tr>
<tr>
<td>JOINDATE</td>
<td>NOT NULL</td>
<td>DATE</td>
</tr>
<tr>
<td>LOGINID</td>
<td>NOT NULL</td>
<td>NUMBER(10)</td>
</tr>
</tbody>
</table>

Table 8: Distributor application table

The table contains the following attributes:

FIRSTNAME: Name of a distributor.
SUP_ID: The Id of a superior distributor.
ADDRESS: Exact address of a customer who wants to be a distributor.
CONTACTNO: Phone number of a Distributor.
PACKAGEDGIVEN: The package which was given to the distributor.
-The package can be 50,100 etc products package.
JOINDATE: The date when the customer wants to join the organization.
LOGINID: It is the Id given to the new distributor.

5.3 UML Diagrams

UML Approach

UML stands for Unified Modeling Language. UML is a language for specifying, visualizing and documenting the system. This is the step while developing any product after analysis. The goal from this is to produce a model of the entities involved in the project which later need to be built. The representations of the entities that are to be used in the product being developed need to be designed.

Software design is a process that gradually changes as various new, better and more complete methods with a broader understanding of the whole problem in general come
into existence. There are various kinds of methods in software design. They are as follows:

✓ Use case Diagram
✓ Sequence Diagram
✓ Collaboration Diagram
✓ State Chart Diagram
✓ Object Diagram
✓ Class Diagram
✓ Component Diagram
✓ Deployment Diagram

**Use case Diagrams:**

Use case Diagram consists of use cases and actors and shows the interaction between the use cases and actors.

❖ The purpose is to show the interactions between use cases and actor.
❖ To represent the system requirement from user’s perspective.
❖ It must be remembered that the use-cases are the functions that are to be performed in the module.
❖ An actor could be the end-user of the system or an external system.

**Sequence Diagram:**

The purpose is to show the flow of functionality through a use case. In other words, we can call it mapping processes in terms of data transfers from the actor through corresponding objects.

❖ To represent the logical flow of data with respect to a process.
❖ It must be remembered that the sequence Diagram display objects and not the classes.

**Collaboration Diagram:**

The diagrammatic representation of process is shown. It is the same as sequence diagram. However, the representation is generic and not logical step by step. The purpose of Collaboration Diagram is to assist the Quality Engineers and System Architects to have a view of the distribution of processing between objects. The diagram represents object dependencies and helps the architects to redesign the system to loosely couple objects.
Class Diagram:
  ❖ This is one of the most important of the diagrams in development.
  ❖ The diagrams break the class into three layers. One has the name, the second
    describes its attributes and the third its methods. The private attributes are
    represented by a padlock to the left of the name.
  ❖ The relationships are drawn between the classes.
  ❖ Developers use the Class Diagram to develop the classes.
  ❖ Analyses use it to show the details of the system.
Architects look at class diagrams to see if any class has too many functions and see if
they are required to be split.

State Transition Diagrams:
  ❖ The Purpose is to model various states in which an object can exist.
  ❖ Most of the objects in a complex system maintain states, depending upon which
    modifies its behavior.
  ❖ Implementation takes start state and stop state.
  ❖ It is method for describing the system’s architecture in detail on various behaviors
    of the component.
  ❖ The details are not used for generating the code by Rational Rose, they are merely
    for documentation.
5.3.1 Use Case Diagrams

Figure 19: Use Case diagram 1

Figure 20: Use Case diagram 2
Figure 21: Use Case diagram 3

Figure 22: Use Case diagram 4
Figure 23: Use Case diagram 5

Figure 24: Use Case diagram 6
5.3.2 Class Diagrams

![Class Diagram]

Figure 25: Class diagram

5.3.3 Sequence Diagrams

![Sequence Diagram]

Figure 26: Sequence diagram
Figure 27: Sequence diagram (Authentication)

Figure 28: Sequence diagram (Application process)
6. System testing

6.1 Testing

Software Testing

As the coding is completed according to the requirement we have to test the quality of the software. Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. Although testing is to uncover the errors in the software but it also demonstrates that software functions appear to be working as per the specification, those performance requirements appear top have been met. In addition, data collected as testing is conducted provide a good indication of software reliability and some indications of software quality as a whole.

To assure the software quality we conduct both White Box Testing and Black Box Testing.

White Box Testing:

White Box Testing is a test case design method that uses the control structure of the procedural designs to derive test cases. As we are using a non-procedural language, there is very small scope for the White Box Testing. Whenever it is necessary, there the control structures are tested and successfully passed all the control structures with a very minimum error.

Black Box Testing:

Black Box Testing focuses on the functional requirement of the software. It enables to derive sets of input conditions that will fully exercise all functional requirements for a program.

The Black Box Testing finds almost all errors. If finds some interface errors and errors in accessing the database and some performance errors. In Black Box Testing we use mainly two techniques Equivalence Partitioning the Boundary Volume Analysis Technique.

Equivalence Partitioning:

In this method we divide input domain of a program into classes of data from which test cases are derived. An Equivalence class represents a set of valid or invalid or a set of related values or a Boolean condition. The equivalence for these is:
- Input condition requires specific value-specific or non-specific two classes.
- Input condition requires a range-in the range or out of range two classes.
- Input condition specifies a member of a set-belongs to a set or not belongs to the set two classes.
- Input condition is Boolean-valid or invalid Boolean condition two classes.

By these types of equivalent classes, we can test for many cases.

**Boundary Values Analysis:**

Number of errors usually occurs at the boundaries of the input domain generally. In this technique a selection of test cases is exercised using boundary values i.e., around boundaries.

By the above two techniques, we eliminated almost all errors from the software and checked for numerous test values for each and every input value. The results were satisfactory.

**System Testing:**

System testing is designated to uncover weakness that was not detected in the earlier tests. The total system is tested for recovery and fallback after various major failures to ensure that no data are lost. An acceptance test is done to validity and reliability of the system. The philosophy behind the testing is to find error in project. There are many test cases designed with this in mind. The flow of testing is as follows:

**Code Testing:**

Specification testing is done to check if the program does with it should do and how it should behave under various condition or combinations and submitted for processing in the system and it s checked if any overlaps occur during the processing.

This strategy examines the logic of the program. Here only syntax of the code is tested. In code testing syntax errors are corrected, to ensure that the code is perfect.

**Unit Testing:**

The first level of testing is called unit testing. Here different modules are tested against the specifications produced during the design of the modules. Unit testing is done to test the working of individual modules with test oracles.
Unit testing comprises a set of tests performed by an individual programmer prior to integration of the units into a large system. A program unit is usually small enough that the programmer who developed it can test it in great detail. Unit testing focuses first on the modules to locate errors. These errors are verified and corrected so that the unit perfectly fits to the project.

**System Testing:**

The next level of testing is system testing and acceptance testing. This testing is done to check if the system has met its requirements and to find the external behavior of the system.

System testing involves two kinds of activities:

- Integration testing
- Acceptance testing

**Integration Testing:**

The next level of testing is called the Integration Testing. In this many tested modules are combined into subsystems, which were then tested. Test case data is prepared to check the control flow of all the modules and to exhaust all possible inputs to the program. Situations like treating the modules when there is no data entered in the text box is also tested.

This testing strategy dictates the order in which modules must be available, and exerts strong influence on the order in which the modules must be written, debugged and unit tested. In integration testing, all the modules / units on which unit testing is performed are integrated together and tested.

**Acceptance Testing:**

This testing is performed finally by user to demonstrate that the implemented system satisfies its requirements. The user gives various inputs to get required outputs.

**Specification Testing:**

Specification testing is done to check if the program does what it should do and how it should behave under various conditions or combination and submitted for processing in the system and it is checked if any overlaps occur during the processing.
Performance Time Testing:

Performance time testing is done to determine how long it takes to accept and respond i.e., the total time for processing when it has to handle quite a large number of records. It is essential to check the exception speed of the system, which runs well with only a handful of test transactions. Such systems might be slow when fully loaded. So testing is done by providing large number of data for processing. A system testing is designed to uncover weaknesses that were not detected in the earlier tests.

The total system is tested for recovery and fallback after various major failures to ensure that no data are lost during an emergency. An acceptance test is done to ensure the user about the validity and reliability of the system.

6.2 Test cases

NOTE: Majority of SI & SM functional test cases defined in this document will be reused to prepare for SM and SC end-to-end call flow functional testing and performance testing. The objective is to have one common set of SI and SM test data so that the SM, SC, Performance, and High Availability test teams can maximize their time to verify the key CCE features as advertised to the customer.

<table>
<thead>
<tr>
<th>Table 9: Test case 1</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Test Case #: Authentication of the user</th>
<th>Priority (H. L): High</th>
</tr>
</thead>
</table>

**Test Objective:** To Authenticate the user.

**Test Description:** With the username and password as the input, validating the user.

**Requirements Verified:**

**Test Environment:** Tomcat Server, Database

**Test Setup/Pre-Conditions:**

<table>
<thead>
<tr>
<th>Actions</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct user id and Password entered</td>
<td>Connected to the server.</td>
</tr>
<tr>
<td>Correct user id and wrong password entered</td>
<td>Not connected to server, Relogin</td>
</tr>
<tr>
<td>Correct user id and wrong password for the 3rd time</td>
<td>Blocked from using the service</td>
</tr>
<tr>
<td>Pass: Yes</td>
<td>Conditional Pass:</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------</td>
</tr>
</tbody>
</table>

Problems/Issues: None

Notes: Successfully Tested and Executed

### Table 10: Test case 2

<table>
<thead>
<tr>
<th>Test Case #: product module verification</th>
<th>Priority (H, L): High</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Test Objective:</strong> verification of product module</td>
<td></td>
</tr>
</tbody>
</table>

Test Description: selecting a product module and verifying if its showing the required screen which consists of the module details

Requirements Verified:

Test Setup/Pre-Conditions:

<table>
<thead>
<tr>
<th>Actions</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selecting a product module</td>
<td>Showing the required module details</td>
</tr>
<tr>
<td>Selection of a product</td>
<td>Getting the description of the product</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pass: Yes</th>
<th>Conditional Pass:</th>
<th>Fail:</th>
</tr>
</thead>
</table>

Problems/Issues: None

Notes: Successfully Tested and Executed

### Table 11: Test case 3

<table>
<thead>
<tr>
<th>Test Case #: distributor module verification</th>
<th>Priority (H, L): High</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Test Objective:</strong> verifying the distributor module</td>
<td></td>
</tr>
</tbody>
</table>

Test Description: getting the distributor list on selecting the distributor module

Requirements Verified:

Test Environment: Tomcat Server, Database
Table 12: Test case 4

Test Case #: verification of the personalized sales module | Priority (H, L): High
--- | ---

**Test Objective:** To verify the personalized sales module

**Test Description:** Getting the required form on selecting sales module

**Requirements Verified:**

**Test Environment:** Tomcat Server, Database

**Test Setup/Pre-Conditions:**

<table>
<thead>
<tr>
<th>Actions</th>
<th>Expected Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selecting the sales module</td>
<td>Getting the associated form</td>
</tr>
<tr>
<td>Filling in the details</td>
<td>Insertions done in the data base</td>
</tr>
</tbody>
</table>

**Pass:** Yes  **Conditional Pass:**   **Fail:**

**Problems/Issues:** None

**Notes:** Successfully Tested and Executed
### Table 13: Test case 5

<table>
<thead>
<tr>
<th>Test Case #: verification of the distributor application module</th>
<th>Priority (H, L): High</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Test Objective:</strong> To verify the distributor application module</td>
<td></td>
</tr>
<tr>
<td><strong>Test Description:</strong> Getting the required form on selecting distributor application module</td>
<td></td>
</tr>
<tr>
<td><strong>Requirements Verified:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Test Environment:</strong> Tomcat Server, Database</td>
<td></td>
</tr>
<tr>
<td><strong>Test Setup/Pre-Conditions:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Actions</strong></td>
<td><strong>Expected Results</strong></td>
</tr>
<tr>
<td>Selecting the distributor application module</td>
<td>Getting the associated form</td>
</tr>
<tr>
<td>Filling in the details</td>
<td>Insertions done in the data base</td>
</tr>
<tr>
<td>Customer details are validated</td>
<td>Accept as a distributor</td>
</tr>
<tr>
<td>Customer details are invalidated</td>
<td>Reject the customer as a distributor</td>
</tr>
<tr>
<td><strong>Pass:</strong> Yes</td>
<td><strong>Conditional Pass:</strong></td>
</tr>
<tr>
<td>Notes: Successfully Tested and Executed</td>
<td></td>
</tr>
</tbody>
</table>
7. Conclusion

This project is successfully implemented with all the features mentioned in the System Requirement Specification. The project to a large extent simplifies the process of obtaining information whenever a mobile phone user requests information. This application provides instant reliable information to users when they need.

➢ The planned project is developed keeping in view the day-to-day problems faced by a common Amway distributor.
➢ The deployment of our application will certainly help the users to reduce unnecessary wastage of time in getting the details of his personalized sales and avoids the unnecessary wastage of money used in manufacturing brochures.

LIMITATIONS

However the project has some limitation. This application works only for one level of subordinates i.e., the user can know only his subordinates in the Amway distributor tree.

FUTURE ENHANCEMENTS

This application can be upgraded to accommodate more levels of Amway Distributor tree.
WEBSITE REFERENCES

➢ www.java.sun.com

➢ www.Wap Forum.com

➢ www.nokia.com

➢ www.mobilein.com

➢ www.amway.com
BIBLIOGRAPHY


