

**Software Development
For Hospital Patient Management**

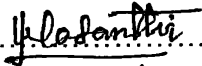
**By
W. K. Kasturiarachchi
99/ A/ AS/ 024**

**A research report submitted in fulfillment of the requirement
for the
Degree of Bachelor of Science
In
Physical Sciences
Faculty of Applied Sciences
Sabaragamuwa University of Sri Lanka
Buttala
2003.**

Declaration

I certify that dissertation does not incorporate without acknowledgment of any material previously submitted for degree or diploma in any university, to the best of my knowledge and belief this does not contain any material previously published, written or orally communicated by another person where due references made in text.

W. K. Kasturiarachchi




Signature

Date: 09/04/2003

To the best of my knowledge above particulars are correct.

Mr. E. Sithirasenan
Internal Supervisor,

The Director,
IDSystems (Pvt.) Ltd., 454/12A,
Piachaud Gardens,
Kandy.



Signature

Date: 29/4/2003

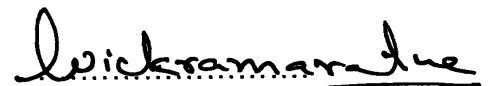
Mr. T. Venugopal
External Supervisor,
IDSystems (Pvt.) Ltd., 454/12A,
Piachaud Gardens,
Kandy.



Signature

Date: 29/04/03

Dr. N. Wickramaratne
Head / Department of Physical Sciences,
Faculty of Applied Sciences,
Sabaragamuwa University of Sri Lanka,
Buttala.



Signature

Date: 09-04-2003

DEDICATION

TO MY LOVING FATHER AND MOTHER

Acknowledgement

I express my sincere gratitude to internal supervisor, Mr. Sithirasenan, The Director, IDSystems (Pvt.) Ltd. for his assistance, encouragement, guidance and his valuable time to make this study a success and he gave many facilities for continue our project.

I express my sincere appreciation and deep sense of gratitude to external supervisor Mr. Venugopal, The Engineer, IDSystems (Pvt.) Ltd. who is kindly offered me industrial placement and he is explained big area of the Linux.

I am deeply grateful to Dr. D.B.M. Wickramaratne, The Dean, Faculty of Applied Sciences, Sabaragamuwa University of Sri Lanka for his guidance and advice.

I am heavily indebted to Dr. N. Wickramaratne, Head/ Department of Physical Sciences, Faculty of Applied Sciences, Sabaragamuwa University of Sri Lanka.

I am heavily indebted to Ms. D. Gunasekara, The Director Singapore Informatics in Kandy Branch, who gave me invaluable support throughout my training period. I would also like to extend my thanks to all the staff members including R. Prakache, C. Kodituwakku whose were always willing to give their support and assistance to me throughout the training period.

I would like to take this opportunity to extend my heartfelt thanks towards the lecturers for their incorporation throughout my study and my colleagues for their invaluable helps, guidance given to me at all time.

Abstract

This software is developed for the patient management in Hospitals. I was trained as a trainee developer for 5 months in partial fulfillment of the requirements for the degree of Bachelor of Science in Physical Sciences, Faculty of Applied Sciences, and Sabaragamuwa University of Sri Lanka. This project is a team effort and I took part in the project for the requirement analysis, designing phases, implementation phases, and testing part. At the time of preparing this document the project was completed only up to the system testing stage. The rest of the stages were documented. It is expected to complete soon.

The project main objective is to develop a user-friendly Hospital Patient Management system for the private channeling hospitals based on the functional specifications given by the Suwasewane Hospital in Kandy. To achieve this objective, Classical Life Cycle/ Waterfall model was used and the steps followed are:

- Defining the requirements of the system and get the approval for it from the Hospital.
- Designing the technical specifications (Table definitions, Table normalization)
- Implementation

Application of the software is following way.

After the patient is admitted in the Hospital, all the patient details are recorded, manually and automatically in the first interface. Then record treatments and charges details and give receipt to the patient. After this step all of data record in the ward, laboratory and etc. This all of the details is saved in the tables of database.

When patient discharge, first select the patient name. Then final Bill is automatically created. This is my main final result in the project. The other result, get the past (history) details of the patient. If some patient wants his or her past treatments and charges details, she or he can get past details. Since all of the details are stored in the tables of the database.

There is used Linux (Red Hat 7.2) operating system. For this development used PHP and MySQL languages mainly. Also used HTML, Java Script for create interfaces. For generate connections between interfaces and the database used MySQL, PHP.

The development process was carried out as an internal project of IDSystems (Pvt.) Ltd. There are three trainees including me to do this development. This System is making for any Hospital within changing simple operation and to use the system. It's a ready mate system the idea is E-Channeling. This system is developed as a **samba server** application.

Contents

Page No.

Abstract	I
Acknowledgment	III
List of figures	IV
List of tables	V
Contents	VI

CHAPTER-1

1.0 Introduction	1
1.1 The Organization	1
1.2 Services By IDSystems (Pvt.) Ltd.....	2
1.3 The Software System.....	2
1.4 Objectives	3

CHAPTER-2

2.0 Theoretical Background and Used Commands	4
2.1 Linux /Unix Operating System	4
2.1.1 What is Linux	4
2.1.2 How did Linux get started	4
2.1.3 Hardware Requirements	6
2.1.4 Important Parts of the kernel.....	7
2.1.5 Major services in a Unix system	7
2.2 Frequently Used Linux Commands.....	9
2.2.1 Basic Linux Commands	9
2.2.2 Advanced Linux Commands	13
2.2.3 Mounting and Unmounting	17
2.2.4 Checking filesystem integrity with fsck	17
2.2.5 Fighting fragmentation.....	18
2.2.6 Boots and Shutdowns	19
2.2.7 More about Shutdowns	19
2.2.8 Overview of the Directory Tree	20
2.2.9 Vi Status	22
2.2.10 Shell Commands	22

2.2.11	Interrupting, Canceling	23
2.2.12	File Manipulation	23
2.2.13	Corrections during inserts	24
2.2.14	Delete	24
2.2.15	Insert, Change	25
2.2.16	Copy and Paste	26
2.2.17	Operators	26
2.3	PHP Overview	27
2.3.1	What is PHP	27
2.3.2	Why use PHP	27
2.3.3	Basic Syntax of PHP	28
2.3.4	Instruction separation	30
2.3.5	Comments	30
2.3.6	Example for PHP	31
2.3.7	Permission for PHP files	32
2.4	MySQL Overview	32
2.4.1	Introduction	32
2.4.2	MySQL Commands	33
2.4.3	Mostly Usage MySQL Commands ...	34
2.5	Advantages and Disadvantages of Linux	37
2.5.1	Advantages of Linux	37
2.5.2	Disadvantage of Linux	38
2.6	Database Management System	38
2.6.1	The System Life Cycle	40
2.6.2	Data Flow Models	41
2.6.3	Data Dictionaries	42

CHAPTER-3

3.0	Methodology	44
3.1	System Project Plan	44
3.2	System DFDs	46
3.3	System Pseudo Codes	50
3.4	Resource Allocation	54

CHAPTER-4

4.0 Results and Discussion	55
4.1 Results	55
4.2 Discussion	56
5.0 References	58
Appendix	59
Appendix – A	59
Tables for Finalized database	
Appendix – B	61
Data Dictionaries of HPMS	
Appendix- C	63
System Interfaces	

List of Figures

	Page No.
2.1 Types of Linux	6
2.2 Linux Directory Tree	21
2.3 Example for PHP coding	27
2.4 Another Example for PHP coding	30
2.5 First Screen of PHP	31
2.6 Simple Example for PHP	31
2.7 Screen for MySQL Databases	34
2.8 Screen for MySQL Tables	35
2.9 System Life Cycle Structure	41
2.10 Symbols in Data Flow Diagram (DFD)	42
3.1 HPMS development life cycle	44
3.2 Context Level DFD for HPMS	46
3.3 Top Level DFD for HPMS	47
3.4 Top Level DFD for Pharmacy Section	48
3.5 Top Level DFD for Laboratory Section	49
4.1 Final Bill for HPMS	55

List of Tables

	Page No.
2.1 Screen for MySQL description Table	35
2.2 Inserted Data	35

CHAPTER-1

1.0 Introduction

Private or Government hospitals do not maintain patient's records in systematic way. Mostly this is done manually. So in these situations these records are either misplaced or lost.

The purpose of this project is to develop a Computer Software to keep track of patients in hospital and to maintain these records for future reference. Also the system provides in-ward and out-wards patient billing and receipting and channelling. The front-end of the software is developed using HTML and Java Script. The back-end used PHP. MySQL is used to maintain and administrate the database. Finally the development is done on the Linux platform. The system is developed keeping mind the future trends. It provides facilities for E-Commerce and intranet access. This system can run on Apache.

This is the software for patient management in the General Hospital. This is not suitable for specialised hospital. Because the requirements are consider for General Hospital. Specialised hospital has limited type of requirements.

Here after patient is admitted, enter the personal details and treatment details in the interfaces in the wards and labs. When patient discharges, can get Final Bill automatically after selecting the patient name. The data enter in the Reception place, Wards and Labs. Here used PHP and MySQL languages mainly. And also HTML is used. The system is web enable application.

1.1 The Organization

IDSystems (Pvt.) Ltd. is a professional organisation that provides an array of services including networking for business organisations, selling of personal computers, system development. The firm stated its business in 1993 and by today has been involved in many large projects and is one of the big names in Kandy. The founder of this prestige organisation who is currently the owner is Mr. E. Sithirasenan an expertise in the field of computer software development.

Here used Linux operating system. There are used Query languages such as PHP and MySQL. And also used Hyper Text Markup Language (HTML), Java Script. There are several interfaces. First enter the data. Then click enters button. All of the data goes to table and saved. After discharge the patient the data is not deleted. The Final Bill is automatically created after selecting the patient name.

1.4 Objectives

(Computerized the hospital)

- ❖ Develop computer software to keep tract of patients in hospitals.
- ❖ Maintain patients' records to future reference.
- ❖ Prepare Test Bill for inward patients.
- ❖ Prepare Final Bill for outward patients.

CHAPTER-2

2.0 Theoretical Background and Used Commands

2.1 Linux/Unix Operating System

2.1.1 What is Linux?

Linux is an operating system that can be downloaded free and "belongs" to an entire community of developers, not one corporate entity. In other words, anyone from professional software developers to hobbyist computer hackers can access and make changes to the Linux kernel all the information about Linux is open and available to everyone. That's why Linux is known as "open source" or "free software," because there is nothing secret about this system. This freedom also allows companies to sell and distribute Linux on CD-ROM or by other means, although those companies must keep their code open to the public.

With more and more people looking for an alternative to Windows, Linux has recently grown in popularity and is quickly becoming a favourite among major corporations and curious desktop users. Not only does it give users a choice of operating systems, it also proves itself valuable with its power, flexibility, and reliability (Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed).

2.1.2 How did Linux get started

The concept of open source programming has been around for many years its roots stem from universities that needed to be able to share information as well as allow students and developers to adapt programs to meet their needs. In 1984, Richard Stallman, a researcher at the MIT AI Lab, started a project he called GNU to counter the fast-moving trend toward proprietary, fee-based software. Stallman, who remains an open advocate of open source, believes that making source code available to anyone who wants it is integral to furthering computer science and innovation. This concept served as the basis of Linux development, the brainchild of Linus Torvalds. When Torvalds began developing Linux in 1991, he was a student at the University of Helsinki and originally targeted Linux at the Intel 386 (although it is now


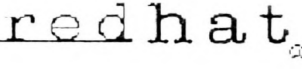



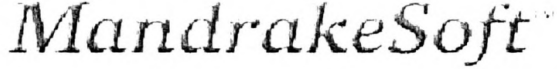

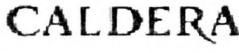



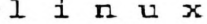
Linux distributions.	Website/Logo
Red Hat Linux: http://www.redhat.com	 
SuSE Linux: http://www.suse.com	 
Mandrake Linux: http://www.mandrakesoft.com	 
Caldera Linux: http://www.calderasystems.com	 
Debian GNU Linux: http://www.debian.org	 
Slackware Linux: http://www.slackware.com	 

Figure 2.1 Types of Linux

2.1.3 Hardware Requirements

To Install the Linux OS the following h/w are suggested:

- Intel x86 or Pentium II/333 MHz
- 32 MB RAM
- 2 GB HDD of either IDE OR SCSI TYPE.
- 1.44 MB HDD
- Network Interface Card (for networking)
- Mouse
- SVGA Colour Monitor (www.Howto.com).

2.1.4 Important Parts of the kernel

The Linux kernel consists of several important parts:

- process management
 - memory management
 - hardware device drivers
 - file system drivers
 - Network management, and various other bits and pieces.
- www.Howto.com

2.1.5 Major services in a UNIX system

- **init**

The single most important service in a UNIX system is provided by `init`. `init` is started as the first process of every UNIX system, as the last thing the kernel does when it boots. When `init` starts, it continues the boot process by doing various startup chores (checking and mounting file systems, starting daemons, etc).

- **Logins from terminals**

Logins from terminals (via serial lines) and the console (when not running X) are provided by the `getty` program. `init` starts a separate instance of `getty` for each terminal.

- **Syslog**

The kernel and many *system programs* produce error, warning, and other messages: It is often important that these messages can be viewed later, even much later, so they should be written to a file. The program doing this is **syslog**.

- **cron and at**

Each user can have a crontab file, where she lists the commands she wishes to execute and the times they should be executed. The `cron` daemon takes care of starting the commands when specified.

The `at` service is similar to `cron`, but it is once only: the command is executed at the given time, but it is not repeated.

- **Graphical user interface**

The graphical environment primarily used with Linux is called the X Window System (X for short). X also does not implement a user interface; it only

implements a window system, i.e., tools with which a graphical user interface can be implemented. Some popular window managers are: *fvwm*, *icewm*, *blackbox* and *windowmaker*. There are also two popular desktop managers, *KDE* and *Gnome*.

- **Networking**

UNIX operating systems have many networking features. Most basic services (filesystems, printing, backups, etc) can be done over the network. This can make system administration easier, since it allows centralised administration, while still reaping in the benefits of microcomputing and distributed computing, such as lower costs and better fault tolerance.

- **Network logins**

Network logins work a little differently than normal logins. There is a separate physical serial line for each terminal via which it is possible to log in. For each person logging in via the network, there is a separate virtual network connection, and there can be any number of these.

- **Network file systems**

With a network file system any file operations done by a program on one machine are sent over the network to another computer. This fools the program to think that all the files on the other computer are actually on the computer the program is running on. This makes information sharing extremely simple, since it requires no modifications to programs.

- **Mail**

The mail system consists of many programs. The delivery of mail to local or remote mailboxes is done by one program (the *mail transfer agent* (MTA), e.g., *sendmail* or *smail*), while the programs users use are many and varied (*mail user agent* (MUA), e.g., *pine*, *mutt* or *elm*). The mailboxes are usually stored in */var/spool/mail*.

- **Printing**

The print queue software also *spools* the printouts on disk, i.e., the text is kept in a file while the job is in the queue. This allows an application program to spit out the print jobs quickly to the print queue software; the application does not have to wait until the job is actually printed to continue.

- **The filesystem layout**

The filesystem is divided into many parts; usually along the lines of a root filesystem with */bin*, */lib*, */etc*, */dev*, and a few others; a */usr* filesystem with programs and unchanging data; a */var* filesystem with changing data (such as

log files); and a /home filesystem for everyone's personal files. Depending on the hardware configuration and the decisions of the system administrator, the division can be different; it can even be all in one filesystem.

(Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed).

2.2 Frequently Used Linux Commands

2.2.1 Basic Linux Commands

All of UNIX is case sensitive. A command with even a single letter's capitalization altered is considered to be a completely different command. The same goes for files, directories, configuration file formats, and the syntax of all native programming languages.

- **ls**

List the information about the files.

- **ls -a**

The -a option that means to list *all* files as well as hidden files.

- **ls -l**

Which lists the contents in *long* format?

- **cp**

The command cp stands for *copy*. It duplicates one or more files.

The format is:

```
cp <file> <newfile>
```

- **cd**

The cd command is used to take you to different directories.

- **mkdir**

Creates a new directory.

E.g.:

```
mkdir foo
```

```
cd foo
```

- **pwd**

The command pwd stands for *present working directory* (also called the *current directory*) and tells what directory you are currently in.

- **rm**

To remove (i.e., erase or delete) a file, use the command

```
rm <filename>.
```

- **rmdir**
To remove a directory, use the command `rmdir <dir>`.
Note that you cannot remove a directory unless it is empty. To remove a directory as well as any contents it might contain, use the command `rm -R <dir>`.
- **mv**
The `mv` command is used to move files and directories. It really just renames a file to a different directory.
Example: `mv apple.txt foo/orange.txt`
- **man**
The command `man <command>` displays help on a particular topic and stands for *manual*.
- **cat**
`cat <filename> [<filename> ...]`
Write the contents of all the files listed to the screen. `cat` can join a lot of files together with `cat <filename> <filename> ... > <newfile>`. The file `<newfile>` will be an end-on-end *concatenation* of all the files specified.
- **clear**
Erase all the text in the current terminal.
- **df**
Stands for *disk free* and tell you how much free space is left on your system.
- **du**
`du <directory>`
Stands for *disk usage* and prints out the amount of space occupied by a directory.
- **head**
`head [-n <lines>] <filename>`
Prints the first `<lines>` lines of a file or 10 lines if the `-n` option is not given.
- **wc**
`wc [-c] [-w] [-l] <filename>`
Count the number of bytes (with `-c` for character), or words (with `-w`), or lines (with `-l`) in a file.
- **whoam**
Print your login name.
- **grep**
`grep [options] <pattern> <filename>`

`grep -n <pattern> <filename>`

show the line number in the file where the word was found.

`grep -<num> <pattern> <filename>`

prints out <num> of the lines that came before and after each of the lines in which the word was found.

`grep -A <num> <pattern> <filename>`

prints out <num> of the lines that came After each of the lines in which the word was found.

`grep -B <num> <pattern> <filename>`

prints out <num> of the lines that came Before each of the lines in which the word was found.

- **find**

The find command is used to search for files.

Example: `find ./ -name foo*.c -print`

`grep -<num> <pattern> <filename>`

prints out <num> of the lines that came before and after each of the lines in which the word was found.

`grep -A <num> <pattern> <filename>`

prints out <num> of the lines that came After each of the lines in which the word was found.

`grep -B <num> <pattern> <filename>`

prints out <num> of the lines that came Before each of the lines in which the word was found.

- **find**

the find command is used to search for files.

Eg: `.find ./ -name foo*.c -print`

- **chmod**

chmod command is used to change the permissions of a file. It's usually used as follows:

`chmod [-R] [u|g|o|a][+|-][r|w|x|s|t] <file>`

For example,

```
chmod                u+x                myfile
```

adds execute permissions for the user of myfile.

- **date**

Print out the current date and time.

- **ps**

List Running Processes.

- **stat**

To get a complete listing of the file's permissions.

- **free**

Print out available free memory. You will notice two listings: swap space and physical memory.

- **less**

With less, you can use the arrow keys to page up and down through the file.

- **tar**

When many files are packed together into one, this packed file is called an *archive*. Usually archives have the extension *.tar*, which stands for *tape archive*.

To *create* an archive of a directory, use the tar command:

```
tar -c -f <filename> <directory>
```

Create a directory with a few files in it, and run the tar command to back it up.

A file of <filename> will be created. You can also use the *verify* option (see the man page) of the tar command to check the integrity of <filename>. Now remove the directory, and then restore it with the *extract* option of the tar command:

```
tar -x -f <filename>
```

- **vi**

Edit to a text file.

- **exit**

To log out from the current log in.

(Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed).

2.2.2 Advanced Linux Commands

Device Files

The MAKEDEV Script

Most device files will already be created and will be there ready to use after you install your Linux system. If by some chance you need to create one which is not provided then you should first try to use the **MAKEDEV** script.

```
# /dev/MAKEDEV -v ttyS0  
create ttyS0 c 4 64 root:dialout 0660
```

This will create the device file `/dev/ttyS0` with major node 4 and minor node 64 as a character device with access permissions 0660 with owner root and group dialout. `ttyS0` is a serial port. The major and minor node numbers are numbers understood by the kernel. The kernel refers to hardware devices as numbers, this would be very difficult for us to remember, so we use filenames. Access permissions of 0660 means read and write permission for the owner (root in this case) and read and write permission for members of the group (dialout in this case) with no access for anyone else.

The mknod command

MAKEDEV is the preferred way of creating device files, which are not present. However sometimes the **MAKEDEV** script will not know about the device file you wish to create. This is where the **mknod** command comes in. In order to use **mknod** you need to know the major and minor node numbers for the device you wish to create. The `devices.txt` file in the kernel source documentation is the canonical source of this information.

To take an example, let us suppose that our version of the **MAKEDEV** script does not know how to create the `/dev/ttyS0` device file. We need to use **mknod** to create it. We know from looking at the `devices.txt` file that it should be a character device with major number 4 and minor number 64. So we now know all we need to create the file.

```
# mknod /dev/ttyS0 c 4 64
# chown root.dialout /dev/ttyS0
# chmod 0644 /dev/ttyS0
# ls -l /dev/ttyS0
crw-rw---- 1 root dialout 4, 64 Oct 23 18:23 /dev/ttyS0
```

Formatting

Formatting is the process of writing marks on the magnetic media that are used to mark tracks and sectors.

Floppies are formatted with **fdformat**. The floppy device file to use is given as the parameter. For example, the following command would format a high density, 3.5 inch floppy in the first floppy drive:

```
$ fdformat /dev/fd0H1440
```

Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.

The **badblocks** command can be used to search any disk or partition for bad blocks (including a floppy). It does not format the disk, so it can be used to check even existing filesystems. The example below checks a 3.5 inch floppy with two bad blocks.

```
$ badblocks /dev/fd0H1440 1440
```

Filesystem

A *filesystem* is the methods and data structures that an operating system uses to keep track of files on a disk or partition; that is, the way the files are organised on the disk.

Most UNIX filesystem types have a similar general structure, although the exact details vary quite a bit. The central concepts are *superblock*, *inode*, *data block*, *directory block*, and *indirection block*.

The superblock contains information about the filesystem as a whole, such as its size (the exact information here depends on the filesystem). An inode contains all information about a file, except its name.

The name is stored in the directory, together with the number of the inode. A directory entry consists of a filename and the number of the inode, which represents the file. The inode contains the numbers of several data blocks, which are used to store the data in the file.

Linux supports several types of filesystems. As of this writing the most important ones are:

Minix

The oldest, presumed to be the most reliable, but quite limited in features (some time stamps are missing, at most 30 character filenames) and restricted in capabilities (at most 64 MB per filesystem).

xia

A modified version of the minix filesystem that lifts the limits on the filenames and filesystem sizes but does not otherwise introduce new features. It is not very popular, but is reported to work very well.

ext2

The most featured of the native Linux file systems, currently also the most popular one. It is designed to be easily upward compatible, so those new versions of the file system code do not require re-making the existing file systems.

In addition, support for several foreign filesystem exists, to make it easier to exchange files with other operating systems.

dos, vfat, iso9660, nfs, smbfs

Creating a filesystem

Filesystems are created, i.e., initialised, with the **mkfs** command. There is actually a separate program for each filesystem type. **mkfs** is just a front end that runs the appropriate program depending on the desired filesystem type. The type is selected with the **-t fstype** option.

The programs called by **mkfs** have slightly different command line interfaces. The common and most important options are summarised below; see the manual pages for more.

-t *fstype*

Select the type of the filesystem.

-c

Search for bad blocks and initialise the bad block list accordingly.

-l *filename*

Read the initial bad block list from the name file.

To create an ext2 filesystem on a floppy, one would give the following commands:

```
$ fdformat -n /dev/fd0H1440
```

```
Double-sided, 80 tracks, 18 sec/track. Total capacity  
1440 kB.
```

```
Formatting ... done
```

```
$ badblocks /dev/fd0H1440 1440 >bad-blocks
```

```
$ mkfs -t ext2 -l bad-blocks /dev/fd0H1440
```

```
mke2fs 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10
```

```
360 inodes, 1440 blocks
```

```
72 blocks (5.00%) reserved for the super user
```

```
First data block=1
```

```
Block size=1024 (log=0)
```

```
Fragment size=1024 (log=0)
```

```
1 block group
```

```
8192 blocks per group, 8192 fragments per group
```

```
360 inodes per group
```

```
Writing inode tables: done
```

```
Writing superblocks and filesystem accounting information:
```

```
done
```

```
$
```

(Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed)

(www.Redhat.com).

2.2.3 Mounting and Unmounting

Before one can use a file system, it has to be *mounted*. The operating system then does various bookkeeping things to make sure that everything works. Since all files in UNIX are in a single directory tree, the mount operation will make it look like the contents of the new file system are the contents of an existing sub-directory in some already mounted file system.

The mounts could be done as in the following example:

```
$ mount /dev/hda2 /home
```

```
$ mount /dev/hda3 /usr
```

The **mount** command takes two arguments. The first one is the device file corresponding to the disk or partition containing the file system. The second one is the directory below which it will be mounted.

Linux supports many filesystem types. **mount** tries to guess the type of the filesystem. You can also use the **-t fstype** option to specify the type directly; this is sometimes necessary, since the heuristics **mount** uses do not always work. For example, to mount an MS-DOS floppy, you could use the following command:

```
$ mount -t msdos /dev/fd0 /floppy
```

When a filesystem no longer needs to be mounted, it can be unmounted with **umount**. **umount** takes one argument: either the device file or the mount point.

```
$ umount /dev/hda2
```

```
$ umount /usr
```

(Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed).

2.2.4 Checking filesystem integrity with fsck

Most systems are setup to run **fsck** automatically at boot time, so that any errors are detected (and hopefully corrected) before the system is used.

fsck must only be run on unmounted filesystems. never on mounted filesystems (with the exception of the read-only root during startup). This is because it accesses

the raw disk, and can therefore modify the filesystem without the operating system realising it. There *will* be trouble, if the operating system is confused.

It can be a good idea to periodically check for bad blocks. This is done with the **badblocks** command. It outputs a list of the numbers of all bad blocks it can find. This list can be fed to **fsck** to be recorded in the filesystem data structures so that the operating system won't try to use the bad blocks for storing data.

The following example will show how this could be done.

```
$ badblocks /dev/fd0H1440 1440 >bad-blocks
$ fsck -t ext2 -l bad-blocks /dev/fd0H1440
Parallelising fsck version 0.5a (5-Apr-94)
e2fsck 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Check reference counts.
Pass 5: Checking group summary information.
/dev/fd0H1440: ***** FILE SYSTEM WAS MODIFIED *****
/dev/fd0H1440: 11/360 files, 63/1440 blocks
```

(Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed).

2.2.5 Fighting fragmentation

When a file is written to disk, it can't always be written in consecutive blocks. A file that is not stored in consecutive blocks is *fragmented*.

It takes longer to read a fragmented file, since the disk's read-write head will have to move more. It is desirable to avoid fragmentation, although it is less of a problem in a system with a good buffer cache with read-ahead.

Ext2 effectively always allocates the free block that is nearest to other blocks in a file. For ext2, it is therefore seldom necessary to worry about fragmentation. There is a program for defragmenting an ext2 filesystem called, strangely enough, **defrag** (Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed).

2.2.6 Boots And Shutdowns

During bootstrapping, the computer first loads a small piece of code called the *bootstrap loader*, which in turn loads and starts the operating system. The bootstrap loader is usually stored in a fixed location on a hard disk or a floppy.

The boot process

The boot sector contains a small program (small enough to fit into one sector) whose responsibility is to read the actual operating system from the disk and start it. On a Linux boot floppy, there is no filesystem, the kernel is just stored in consecutive sectors, since this simplifies the boot process. It is possible, however, to boot from a floppy with a filesystem, by using LILO, the Linux LOader.

When booting with LILO, it will normally go right ahead and read in and boot the default kernel. It is also possible to configure LILO to be able to boot one of several kernels, or even other operating systems than Linux, and it is possible for the user to choose which kernel or operating system is to be booted at boot time.

LILO can be configured so that if one holds down the **alt**, **shift**, or **ctrl** key at boot time (when LILO is loaded), LILO will ask what is to be booted and not boot the default right away. Alternatively, LILO can be configured so that it will always ask, with an optional timeout that will cause the default kernel to be booted.

With LILO, it is also possible to give a *kernel command line argument*, after the name of the kernel or operating system (Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed), (www.Redhat.com).

2.2.7 More about Shutdowns

It is important to follow the correct procedures when you shut down a Linux system. If you fail to do so, your filesystems probably will become trashed and the files probably will become scrambled. This is because Linux has a disk cache that won't write things to disk at once, but only at intervals. This greatly improves performance but also means that if you just turn off the power at a whim the cache may hold a lot of

data and that what is on the disk may not be a fully working filesystem (because only some things have been written to the disk).

Another reason against just flipping the power switch is that in a multi-tasking system there can be lots of things going on in the background, and shutting the power can be quite disastrous. By using the proper shutdown sequence, you ensure that all background processes can save their data.

If your system has many users, use the command **shutdown -h +time message**, where time is the time in minutes until the system is halted, and message is a short explanation of why the system is shutting down.

```
# shutdown -h +10 'We will install a newdisk. System should  
> be back on-line in three hours.'  
#
```

This will warn everybody that the system will shut down in ten minutes, and that they'd better get lost or lose data.

When the real shutting down starts after any delays, all filesystems (except the root one) are unmounted, user processes (if anybody is still logged in) are killed, daemons are shut down, all filesystem are unmounted, and generally everything settles down.

(Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed), (www.Redhat.com).

2.2.8 Overview of the Directory Tree

- The full directory tree is intended to be breakable into smaller parts, each capable of being on its own disk or partition.
- The major parts are the root (*/*), */usr*, */var*, and */home* filesystems. Each part has a different purpose. The directory tree has been designed so that it works well in a network of Linux machines which may share some parts of the filesystems over a read-only device (e.g., a CD-ROM), or over the network with NFS.

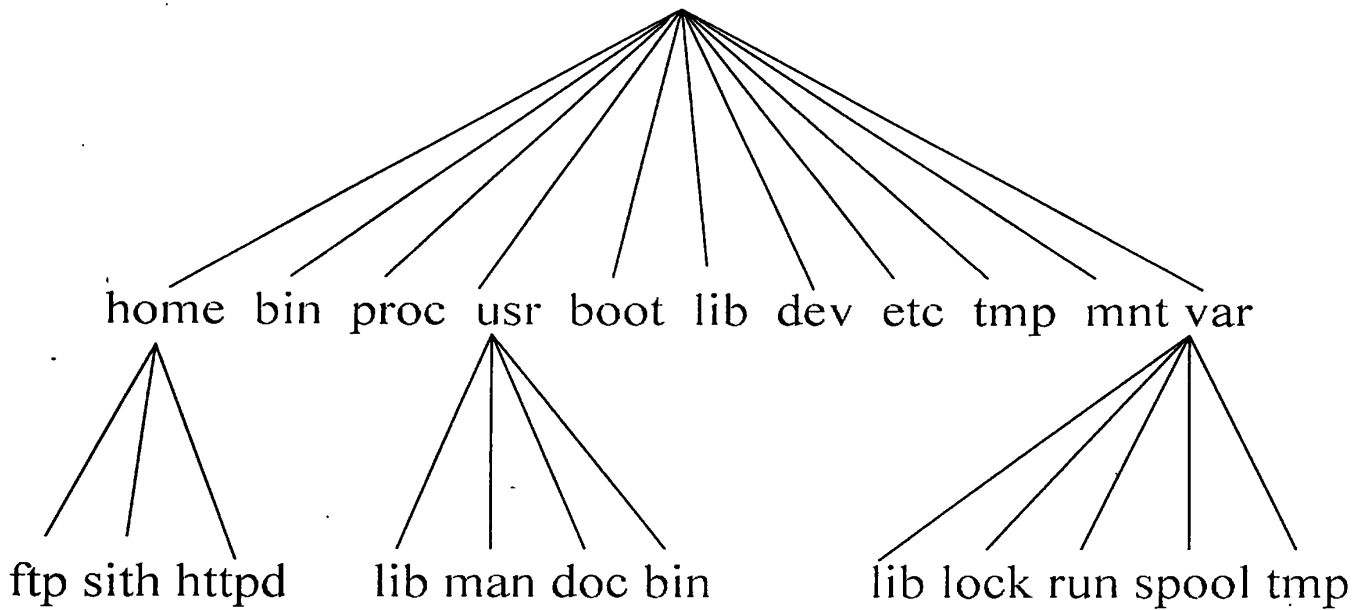


Figure 2.2 Linux Directory Tree

- The root file system is specific for each machine (it is generally stored on a local disk, although it could be a ramdisk or network drive as well). It contains the files those are necessary for booting the system up, and to bring it up to such a state those other file systems may be mounted.
- The `/usr` filesystem contains all commands, libraries, manual pages, and other unchanging files needed during normal operation. No files in `/usr`. It should be specific for any given machine, nor should they be modified during normal use.
- The `/var` filesystem contains files that change, such as spool directories (for mail, news, printers, etc), log files, formatted manual pages, and temporary files. Traditionally everything in `/var` has been somewhere below `/usr`, but that made it impossible to mount `/usr` read-only.
- The `/home` filesystem contains the users' home directories, i.e., all the real data on the system. Separating home directories to their own directory tree or filesystem makes backups easier; the other parts often do not have to be backed up, or at least not as often as they seldom change. A big `/home` might have to be broken across several file systems, which requires adding an extra naming level below `/home`, for example `/home/students` and `/home/staff` (Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed), (www.Redhat.com).

2.2.9 Vi states

Vi has 3 modes:

1. **command mode** - Normal and initial state; others return here (use **ESC** to abort a partially typed command)
2. **input mode** - entered by specific commands **a i A l o O c C s S R** and ended by **ESC** or abnormally with interrupt
3. **line mode** - i.e. waiting for input after a **:**, **/**, **?** or a **!** command (end with **CR**, abort with **CTRL-c**). **CTRL** is the control key: **CTRL-c** means "control c"
(Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed), (www.PHP.com).

2.2.10 Shell Commands

1. **TERM= code** Puts a code name for your terminal into the variable **TERM**
2. **export TERM** Conveys the value of **TERM** (the terminal code) to any UNIX system program that is terminal dependant.
3. **tput Init** Initializes the terminal so that it will function properly with various UNIX system programs.
4. **vi filename** Accesses the **vi** screen editor so that you can edit a specified file.
5. **vi file1 file2 file3** Enters three files into the **vi** buffer to be edited. Those files are **file1**, **file2**, and **file3**.
6. **view file** Invoke **vi** editor on **file** in read-only mode
7. **vi -R file** Invoke **vi** editor on **file** in read-only mode
8. **vi -r file** Recover **file** and recent edits after system crash
(Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed), (www.PHP.com).

2.2.11 Interrupting, Cancelling

1. **ESC** end insert or incomplete command.
2. **CTRL-?** **CTRL** is the control key: **CTRL-?** means "control ?" delete or rubout interrupts.
3. **CTRL-I** reprint/refresh screen if **CTRL-?** scrambles it.
(Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed), (www.Redhat.com).

2.2.12 File Manipulation

1. **ZZ** Save the file and exit vi
2. **:wq** Save the file and exit vi
3. **:w** Write the current file
4. **:w!** Force write the current file, if file is read-only
5. **:wname** Write to file *name*
6. **:q** Exit from vi
7. **:q!** Force exit from vi (discarding changes)
8. **:e name** Edit file *name*
9. **:e!** reedit, discard changes
10. **:e + name** edit file *name*, starting at end
11. **:e + n** edit starting at line *n*
12. **:e #** edit alternate file
13. **:n** edit next file in *arglist*
14. **:args list** files in current filelist
15. **:rew** rewind current filelist and edit first file
16. **:n args** specify new arglist

17. **:f** show current file and line

18. **CTRL-G** synonym for **:f** , show current file and line

19. **:ta tag** to tag file entry *tag*

20. **CTRL-J** :ta, following word is tag

(Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed), (www.PHP.com).

2.2.13 Corrections during insert

1. **CTRL-h** Erase last character

2. **CTRL-w** Erase last word

3. **erase** Press DELETE key, same as CTRL-h

4. **kill** Your kill key, erase input this line

5. **** Escapes CTRL-h, DELETE and kill

6. **ESC** Ends insertion, back to command

7. **CTRL-?** Interrupt, terminates insert

8. **CTRL-d** Backtab over *autoindent*

9. **CTRL-v** Quote non-printing character

(Castagnetto, J., et al., (1999) Professional PHP Programming), (Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed).

2.2.14 Delete

1. **x** Delete the character under the cursor

2. **X** Delete the character before the cursor

3. **D** Delete to the end of line

4. **d^** Delete back to start of line

5. **dd** Delete the current line
6. **ndd** Delete *n* lines starting with the current one
7. **dnw** Delete *n* words starting from cursor
(DuBois, P., (1999) MySQL), (Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed).

2.2.15 Insert, Change

1. **i** Enter input mode inserting before the cursor
 2. **I** Enter input mode inserting before the first non-blank character
 3. **a** Enter input mode inserting after the cursor
 4. **A** Enter input mode inserting after the end of the line
 5. **o** Open a new line below current line and enter input mode
 6. **O** Open a new line above current line and enter input mode
 7. **r** Replace the character under the cursor (does NOT enter input mode)
 8. **R** Enter input mode replacing characters
 9. **C** shift-c. Change rest of line
 10. **D** shift-d. Delete rest of line
 11. **s** Substitute chars
 12. **S** Substitute lines
 13. **J** Join lines
 14. **J** Join lines
- (DuBois, P., (1999) MySQL), (Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed).

2.2.16 Copy and Paste

Every delete command, or explicitly fills the "yank buffer" by **Y** and **yy**.

1. **Y** Copy the current line to the yank buffer
2. **nyy** Copy *n* lines starting from the current to the yank buffer
3. **p** Paste the yank buffer after the cursor (or below the current line)
4. **P** Paste the yank buffer before the cursor (or above the current line)
5. **"xp** Put from buffer *x*
6. **"xy** Yank to buffer *x*
7. **"xd** Delete into buffer *x*

(Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed).

2.2.17 Operators (use double to affect lines)

1. **d** delete
2. **c** change
3. **<** left shift
4. **>** right shift
5. **I** filter through command
6. **=** indent for LISP
7. **y** yank text to buffer

(Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed).

2.3 PHP Overview

2.3.1 What is PHP?

PHP (recursive acronym for "PHP: Hypertext Preprocessor") is a widely-used Open Source general-purpose scripting language that is especially suited for Web development and can be embedded into HTML.

```
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    <?php
      echo "Hi, I'm a PHP script!";
    ?>
  </body>
</html>
```

Figure 2.3 Example for PHP coding

2.3.2 Why we are use PHP

Working under the Linux platform, we can work with Python, Pearl, Java but PHP is simplest and powerful language. PHP is widely used as open source general purpose scripting language. That is specially suited for web development and can be embedded into HTML. Using HTML we can not create a dynamic pages, but using PHP can create dynamic pages. PHP is server-side scripting, so we can do anything any other CGI program can do. Such as collect from data, generate dynamic page content, send and receive cookies. PHP is also command line scripting and writing client-side GUI applications.

PHP is support for wide range of databases.

Such as,

- Adabas D
- Dbase
- Oracle
- Ingres
- MySQL
- Informix
- ODBC
- Velocis
- Unixdbm

2.3.3 Basic Syntax of PHP

Escaping from HTML

When PHP parses a file, it simply passes the text of the file through until it encounters one of the special tags, which tell it to start interpreting the text as PHP code. The parser then executes all the code it finds, up until it runs into a PHP closing tag, which tells the parser to just start passing the text through again. This is the mechanism, which allows you to embed PHP code inside HTML: everything outside the PHP tags is left utterly alone, while everything inside is parsed as code.

There are four sets of tags, which can be used to denote blocks of PHP code. Of these, only two (`<?php. .?>` and `<script language="php">. .</script>`) are always available; the others can be turned on or off from the `php.ini` configuration file. While the short-form tags and ASP-style tags may be convenient, they are not as portable as the longer versions. Also, if you intend to embed PHP code in XML or XHTML, you will need to use the `<?php. .?>` form to conform to the XML. The tags supported by PHP are:

Example: Ways of escaping from HTML

1. `<? Echo ("this is the simplest, an SGML processing instruction\n"); ?>`
`<?= expression ?>` This is a shortcut for "`<? echo expression ?>`"
2. `<?php echo("if you want to serve XHTML or XML documents, do like this\n"); ?>`
3. `<script language="php">`
 `echo ("some editors (like FrontPage) don't`
 `like processing instructions");`
`</script>`
4. `<% echo ("You may optionally use ASP-style tags"); %>`
`<%= $variable; # This is a shortcut for "<% echo . . ." %>`

The first way is only available if short tags have been enabled. This can be done via the `short_tags()` function (PHP 3 only), by enabling the short_open_tag configuration

setting in the PHP config file, or by compiling PHP with the `--enable-short-tags` option to **configure**.

Again, the second way is the generally preferred method, as it allows for the use of PHP in XML-conformant code such as XHTML.

The fourth way is only available if ASP-style tags have been enabled using the asp_tags configuration setting.

Note: Support for ASP-style tags was added in 3.0.4.

The closing tag for the block will include the immediately trailing newline if one is present. Also, the closing tag automatically implies a semicolon; you do not need to have a semicolon terminating the last line of a PHP block.

PHP allows you to use structures like this:

Example: Advanced escaping

```
<?php
if ($expression) {
    ?>
    <strong>This is true.</strong>
    <?php
} else {
    ?>
    <strong>This is false.</strong>
    <?php
}
?>
```

This works as expected, because when PHP hits the `?>` closing tags, it simply starts outputting whatever it finds until it hits another opening tag. The example given here is contrived, of course, but for outputting large blocks of text, dropping out of PHP parsing mode is generally more efficient than sending all of the text through `echo()` or `print()` or some such.

(Castagnetto, J., et al., (1999) Professional PHP Programming), (Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed).

2.3.4 Instruction separation

Instructions are separated the same as in C or Perl - terminate each statement with a semicolon.

The closing tag (?>) also implies the end of the statement, so the following are equivalent:

```
<?php
    echo "This is a test";
?>

<?php echo "This is a test" ?>
```

Figure 2.4 Another Example for PHP coding

2.3.5 Comments

PHP supports 'C', 'C++' and UNIX shell-style comments. For example:

```
<?php
    echo "This is a test"; // This is a one-line c++ style comment
    /* This is a multi line comment
       yet another line of comment */
    echo "This is yet another test";
    echo "One Final Test"; # This is shell-style style comment
?>
```

The "one-line" comment styles actually only comment to the end of the line or the current block of PHP code, whichever comes first.

```
<h1>This is an <?php # echo "simple";?> example.</h1>
```

```
<p>The header above will say 'This is an example'.
```

You should be careful not to nest 'C' style comments, which can happen when commenting out large blocks.

```
<?php
/*
    echo "This is a test"; /* This comment will cause a problem */
*/
?>
```

2.3.6 Example for PHP

Vim filename.php

This is the way of starting PHP file. As example, Vim First.php
Here extension is php and the file name is First. Then can see following type of screen.

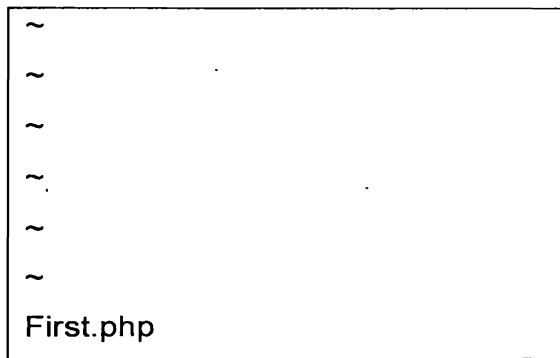


Figure 2.5 First Screen of PHP

In here can write the PHP coding.

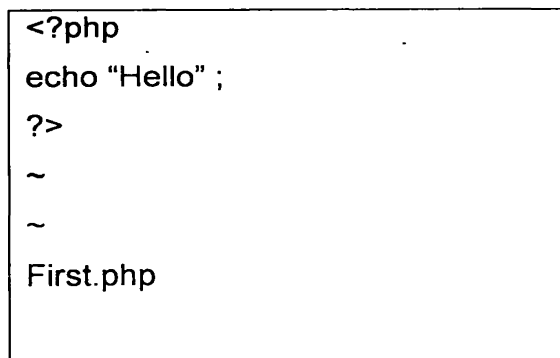


Figure 2.6 Simple Example for PHP

PHP code starts with <? Tag and end with ?> Tag. Echo means normal print command.

This file can save using writes (w) command. Can exit with quit (q) command. Both can use same time (wq). After quit file, can run on Mozella screen. Can get as **Hello**.

(Castagnetto, J., et al., (1999) Professional PHP Programming), (Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed).

2.3.7 Permission for PHP file

For the file can give several ways of permissions. They are owners, group, others. Each team has number. The computer identifies the permission by using the numbers.

For read the number is four. For write the number is two. For the exit the number is one.

Read = 4

Write = 2

Exit = 1

If give seven for one team, they can read, write, exit.

Permission can give as following way.

Chmod 700 filename

As example,

Chmod 777 First.php

This means read, write, exit permission have for owner, group, others for the First.php file.

(Castagnetto, J., et al., (1999) Professional PHP Programming).

2.4 MySQL Overview

2.4.1 Introduction

The database has become an integral part of almost every human's life. Without it, many things we do would become very tedious, perhaps impossible tasks.

Banks, universities, and libraries are three examples of organisations that depend heavily on some sort of database system. On the Internet, search engines, online shopping, and even the web site naming convention (<http://www...>) would be

impossible without the use of a database. A database that is implemented and interfaced on a computer is often termed a database server.

- One of the fastest SQL (Structured Query Language) database servers currently on the market is the MySQL server.
- The capabilities range across a number of topics, including the following:
 - Ability to handle an unlimited number of simultaneous users.
 - Capacity to handle 50,000,000+ records.
 - Very fast command execution, perhaps the fastest to be found on the market.
 - Easy and efficient user privilege system

(DuBois, P., (1999) MySQL), (Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed).

2.4.2 MySQL Commands

MySQL follows the relational rules of tables, Columns, and rows. Those rows correspond to records, columns to fields, and tables to file. Also relational DBMS doesn't use the master and detail data sets that characterize network or hierarchical DBMSs. MySQL uses its own Data Manipulation Language(DML) and Data Definition Language(DDL).

- **Connect to MySQL.**

This involves the insertion of the hostname, username and password given specifically for MySQL use.

Syntax: mysql -h hostname -u username -p
 or

mysql -u username --password

The user will then be prompted for a password If that works, you should see some introductory information followed by a mysql> prompt:

```
Shell> mysql -h host -u user -p
```

```
Enter password: *****
```

```
Welcome to the MySQL monitor.  Commands end with \q.
```

Your MySQL connection id is 459 to server version:

3.22.20a-log

Type 'help' for help.

Mysql>

The prompt tells you that mysql is ready for you to enter commands.

(DuBois, P., (1999) MySQL), (Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed).

2.4.3 Mostly Usage MySQL Commands

There are mostly important commands. Some of them are followed.

Can create database using,

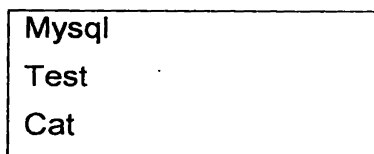
Mysql > create database db_name;

As example,

Mysql > create database Cat;

After create table can see using,

Mysql > show databases;



```
Mysql
Test
Cat
```

Figure 2.7 Screen for MySQL Databases.

Can create table using,

Mysql > create table tb_name (

First field type length,

Second field type length,

Final field type length

As example

```
Mysql > create table patient (  
    ID int (20) not null auto_increment primary key,  
    Name varchar (20) not null,  
    Address varchar (10) default null,  
    PhoneNo int(10) null  
);
```

The table structure can see using ,

```
Mysql > show tables;
```

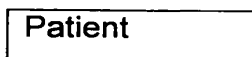


Figure 2.8 Screen for MySQL Tables

```
Mysql > show patient;
```

Table 2.1 Screen for MySQL description

Field	Size	Type	Extra
ID	Int (20)	Not null	Auto_incremant
Name	Varchar(20)	Not null	
Address	Varchar(10)	Default null	
PhoneNo	Int(10)	Null	

Can insert values to the above table using ,

```
Mysql > insert into patient(Name, Address, PhoneNo )  
values('Saman','Kandy',232444);
```

```
Mysql > select * from patient;
```

Can see table data using above select command. * means all of the data in the table included.

Table 2.2 Inserted Data

ID	Name	Address	PhoneNo
1	Saman	Kandy	232444

If want to delete whole table can use following command.

```
Mysql > drop table tb_name ;
```

If want to delete some of fields in the table , can use following command.

```
Mysql > alter table tb_name drop field_name;
```

If want to add some of fields to the table, can use following command.

```
Mysql > altar table tb_name add field_name;
```

As example,

```
Mysql > alter table patient add Age int(10) not null;
```

If want to delete fields some of data in the table, can use following command.

```
Mysql > delete from tb_name where field_name;
```

As example for the Test table,

```
Mysql > delete from Test where TestNo="0";
```

If want to update some record, can use following command.

```
Mysql > update tb_name set field_name where other_field_name ;
```

As example,

```
Mysql > update Test set Active='no' where TestNo=5;
```

If want to change column type some of the table, can use following command.

```
Mysql > altar table tb_name modify field_name;
```

If want to rename the table, can use following command.

```
Mysql > altar table tb_name rename as new_ tb_name;
```

Can exit from Mysql using,

```
Mysql > \q
```

After can go to PHP.

Using this method I have completed my whole of the project.

There are several types of Data Types.

As Example,

Int, Varchar, Large, Small, Date, Time so on.

(Castagnetto, J., et al., (1999) Professional PHP Programming), (DuBois, P., (1999) MySQL), (Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed).

2.5 Advantages and Disadvantages of Linux

2.5.1 Advantages of Linux

In this development used Red Hat Linux. Linux is very powerful and a real operating system and Linux is fairly small compared to other UNIX operating systems.

Many UNIX operating systems require 500MB or more, whereas Linux can be run on as little as 150MB of space and can run on as little as 2MB of RAM.

There are not spaces when writing table field in MySQL database.

The Linux operating system has several advantages and little bit of disadvantages. The several advantages are followed.

➤ Full Multitasking

Multiple tasks can be accomplished and multiple devices can be accessed at the same time.

➤ Virtual Memory

Linux can use a portion of hard drive as virtual memory, which increases the efficiency of the system by keeping active processes in RAM and placing less frequently used or inactive portions of memory on the disk. Virtual memory also utilizes all the system's memory and does not allow memory segmentation to occur.

IX machines. This powerful interface supports many applications and is a standard interface for the Industry.

➤ Built-in networking support

Linux users standard TCP/IP Protocols, including Network File System(NFS) and Network Information Service(NIS). Can access the Internet by connecting the system with an Ethernet card or over a modem to another system.

➤ Shared Libraries

Each application, instead of keeping its own copy of software, shares a common library of subroutines it can call at run time. This saves a lot of hard drive space on the system.

(Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed).

2.5.2 Disadvantages of Linux

The little bit of disadvantages is followed.

- There are two Query Languages mainly. They are PHP and MySQL. Both are command languages. So if use Linux, all of the commands must be remember. This is main problem.
- Linux has Graphical User Interface (GUI). But it is not many users friendly than Windows.
- The other problem is data entering speed is depend on persons to persons and experiences. This is decrease speed little bit.
- Then other important point is PHP, MySQL languages are case sensitive.

(Pitts, D., Ball, B., (1998) Red Hat Linux Unleashed).

2.6 Database Management Systems

Every computerized information system requires not only data, but the structure of that data. A database management system (DBMS) collects and structures related files so that many users can easily retrieve, manipulate, and store data.

A number of computer manufactures and software developers market database management systems. These software packages enable users to organize large, complex bodies of data in to useful, accessible and compact forms –thus avoiding the awkward and inefficient file management techniques of the past. Since database systems merge data in to one pool, any change to one file automatically affects all other relevant files:

Programmes to retrieve, update, add, or delete data in the database can involve batch, on-lines, or mixed processing.

A DBMS precludes the need for the programmer to worry about such time-consuming details as where or how the computer physically stores the data because the DBMS can retrieve the data at will. Then the analyst can focus on higher priorities. The system itself and users needs.

(Edwards, P., (2001) System Analysis and Design), Sommerville, I., (1999) Software Engineering).

Advantages of DBMS;

➤ File consolidation

Pooling data reduces redundancy and inconsistency and promotes co-operation among different users since databases link records together logically, a data change in one system will cascade through all the other system using that.

➤ Program and file independence

This feature separates the definition of the file from their programmes, allowing a programmer to concentrate on the logic of the programme instead of on precisely how to store and retrieve data.

➤ Access versatility

Users can retrieve data in many ways. They enjoy the best of both worlds sequential access for reporting data in a prescribed order and random access for rapid retrieval of a specific record.

➤ Data security

A DBMS usually includes a password that controls access to sensitive data. By limiting access to read only/ write only specified records or even fields with in records, passwords can prevent certain users from retrieving or altering data.

➤ Program development

Programmers must use standard name for the data items rather than inventing their own from program to program. This lets the programmer focus on the desired function.

➤ Program maintenance

Changes and repairs to a system are relatively easy to accomplish.

➤ Special information

Special purpose report generators can produce reports with minimum effort. (Edwards, P., (2001) System Analysis and Design), (Sommerville, I., (1999) Software Engineering).

Disadvantages of DBMS;

➤ Additional hardware

Using a DBMS MAY REQUIRE the purchase of additional memory and/ or disk drives. Memory is used to hold the DBMS software while extra disk drives keep the special files that the DBMS requires. However, this is offset by saving due to less redundancy in the data.

➤ **Staff Retraining**

Programmers who are unfamiliar with DBMS concepts and terminology will need special training to orient them to the new DBMS environment.

➤ **Software Cost**

Software vendors charge for their DBMS software and in some cases, the DBMS software cost can exceed \$100000.

➤ **Special Staff**

The organisation may need to hire a DBMS expert to supervise and manage the DBMS. Usually called a database administrator, this person establishes rules regarding who is permitted to use the data, monitors the data security against unauthorized intrusion, and advises analysts and programmers about the best ways to use the DBMS.

Despite these disadvantages, many organisations are switching from flat file system to a DBMS. They see advantages as outweighing the disadvantages, specially, over the long term.

(Edwards, P., (2001) System Analysis and Design).

2.6.1 The System Life Cycle

System projects involve a sequence of phases called a system life cycle, which consists of four distinct phases. Analyse, Design, Implementation, and Maintenance.

System Analysis involves studying the ways an organization currently retrieves and processes data to produce information with the goal of determining how to make it work better. To do so, the systems analysts may develop alternative systems and evaluate each in terms of cost /benefit and feasibility.

System design requires the analyst to decide on the formats of the reports that the system will be produce, define data-storage methods, plan how to collect and input data, and define the necessary programs. Design is the second phase in the life cycle.

System implementations include actual programming testing, training, and use new system: Upon completion of the system, the analysts, users, and management

evaluate the system to ensure that it fulfils all its goals. Implementation is the third step of the systems process.

System maintenance is the fourth and final phase in the systems life cycle. System maintenance includes repairing the system when errors are detected, enhancing the system when the user needs new functions, or changing the system to meet new laws or changes in the organizations propose or goals.

(Sommerville, I., (1999) Software Engineering).

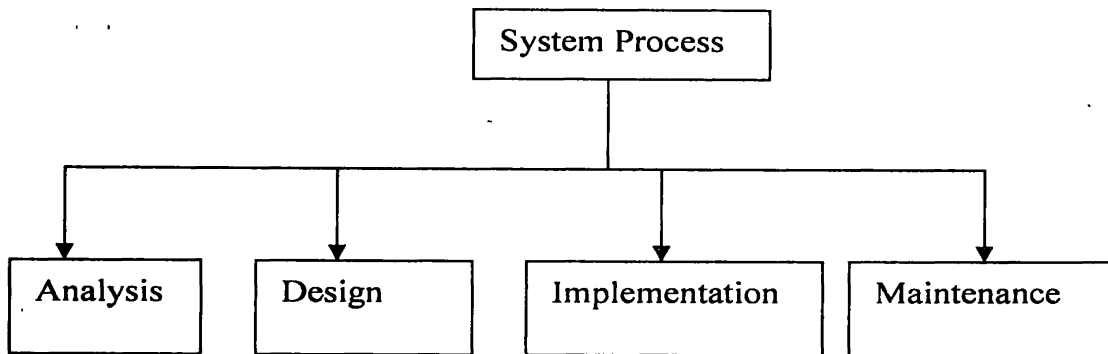


Figure 2.9 System Life Cycle Structure

2.6.2 Data Flow Models

Data Flow models are an intuitive way of showing how data is processed by a system. At the analysis level, they should be used to model the way in which data is processed in the existing system. The notation used in these models represents functional processing, data stores and data movements between functions. Data flow models are used to show how data flows through a sequence of processing steps. The data is transformed at each step before moving on to the next stage.

These processing steps or transformations are program functions when data flow diagrams are used to document a software design. There are various notations used for data flow diagrams. Rounded circles represent processing steps, arrows annotated with the data name represent flows and rectangles represent data entity.

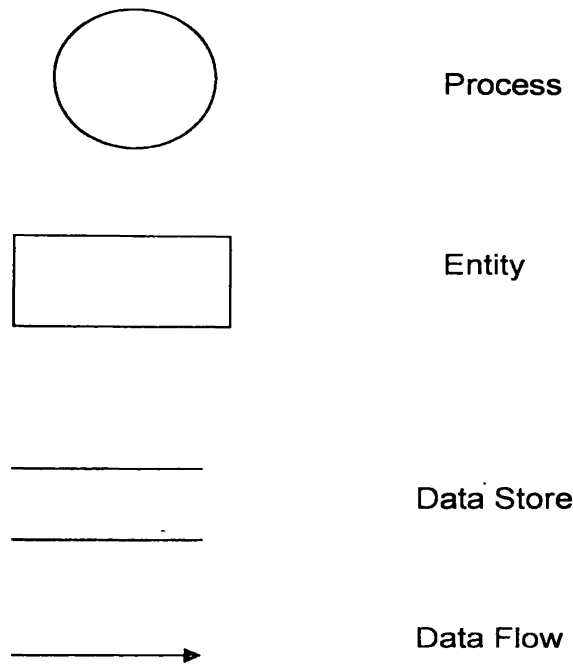


Figure 2.10 Symbols in Data Flow Diagrams (DFD)

Data Flow Models are valuable. Because tracking and documenting how the data associated with a particular process moves through the system helps analysts understand what is going on.

Data Flow Diagrams have the advantage that, unlike some other modelling notations, they are simple and intuitive.

(Sommerville, I., (1999) Software Engineering).

2.6.3 Data Dictionaries

As a system data model is derived, many named entities, relationships and forth will be identified. The names given to the entities should be chosen to give the reader some clues to their meaning.

However, further description of the named entities is usually needed to make the model understandable. This description can be informal or formal.

Whatever approach is used, it is always worth collecting all descriptions in a single repository or data dictionary.

A data dictionary is, simplistically, a list of names used by the system, arranged alphabetically. As well as the name, the dictionary should include a description of the named entity and, if the name represents a composite object, there may be description of the composition. Other information such as the date of creation, the creator, and the representation of the entity may also be included depending on the type of model which is being developed.

The advantages of using a data dictionary are;

- It is a mechanism for name management.
A large system model may be developed by many different people who have to invent names for entities and relationships. These names should be used consistently and should not clash. The data dictionary software can check for name uniqueness and tell requirements analysts of name duplications.
- It serves as a store of organizational information which can link analysis, design, implementation and evaluation. As the system is developed, information is taken to inform the development. New information is added to it. All information about an entity is in one place.

All system names, whether they are names of entities, types, relations, attributes or services, should be entered in the dictionary. Support software should be available to create, maintain and interrogate the dictionary. This software might be integrated with other tools so that dictionary creation is partially automated.

(Edwards, P., (2001) System Analysis and Design), (Sommerville, I., (1999) Software Engineering).

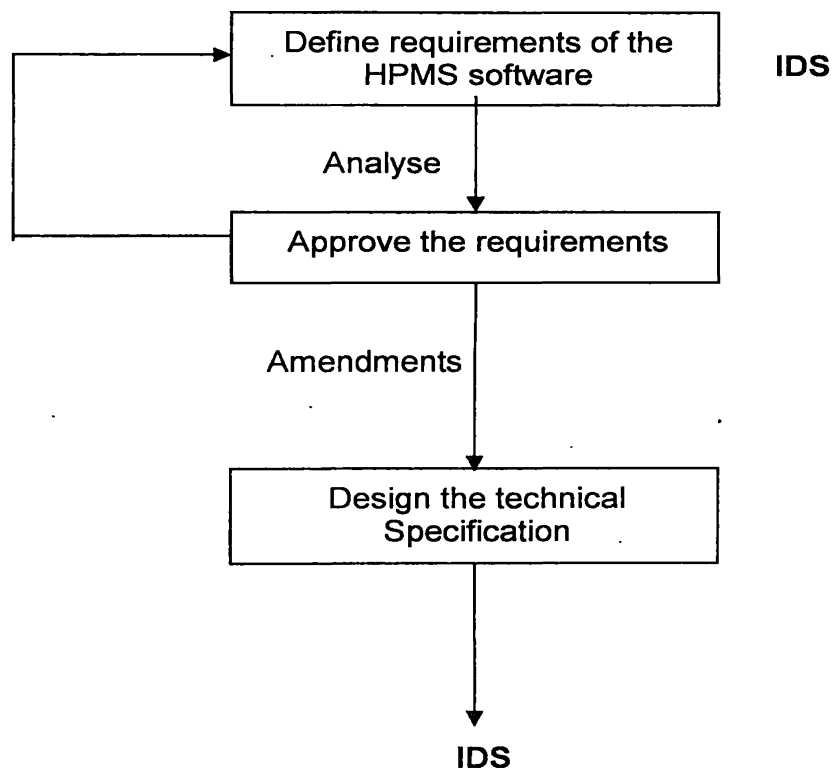
CHAPTER-3

3.0 Methodology

3.1 Project Plan

System Development Life Cycle

The tasks involved in the development of the Hospital Patient Management System (HPMS) are given in the diagram below. It defines the steps involved in the project, and IDSystems, deliverables and approval points during the project life cycle. Each of the tasks is described in details in the following chapters.



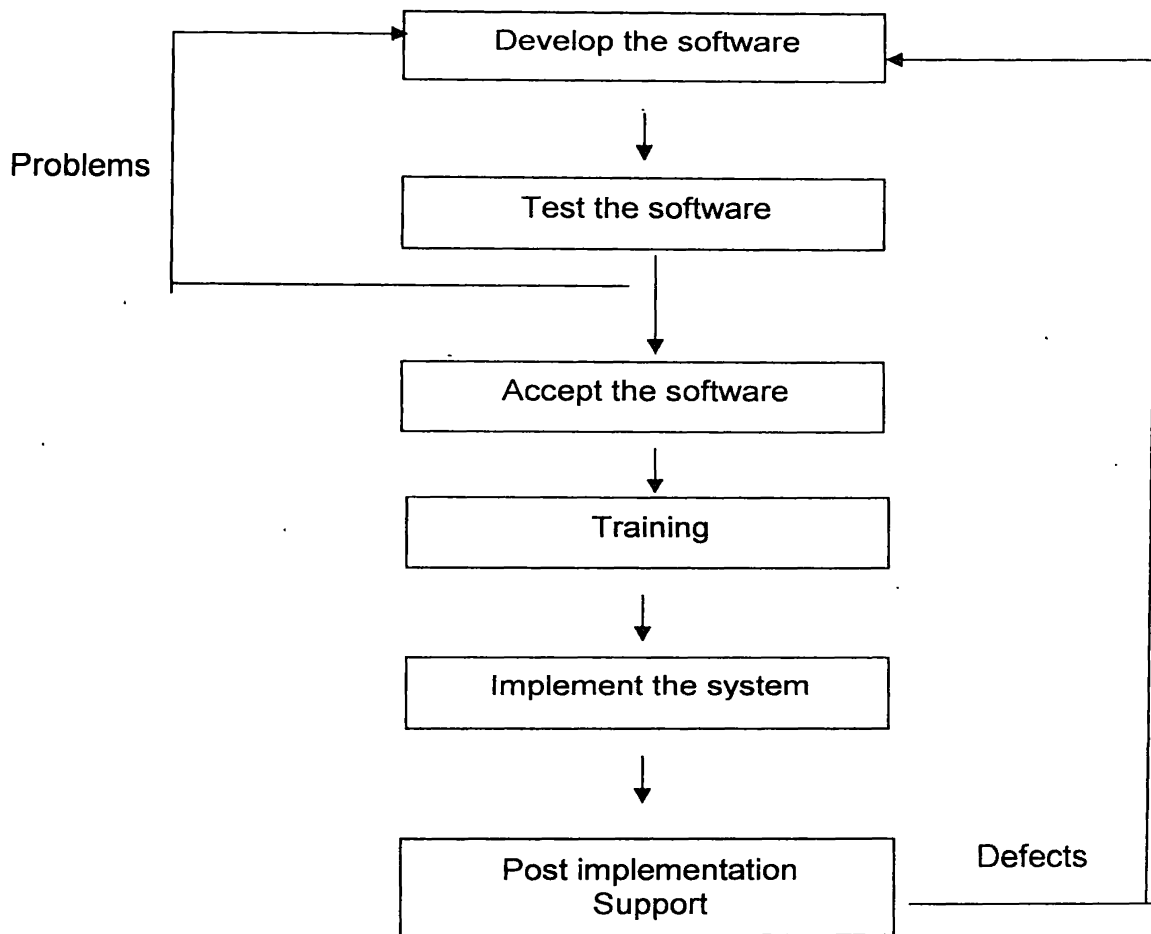


Figure 3.1 HPMS development life cycle

3.2 System DFDs

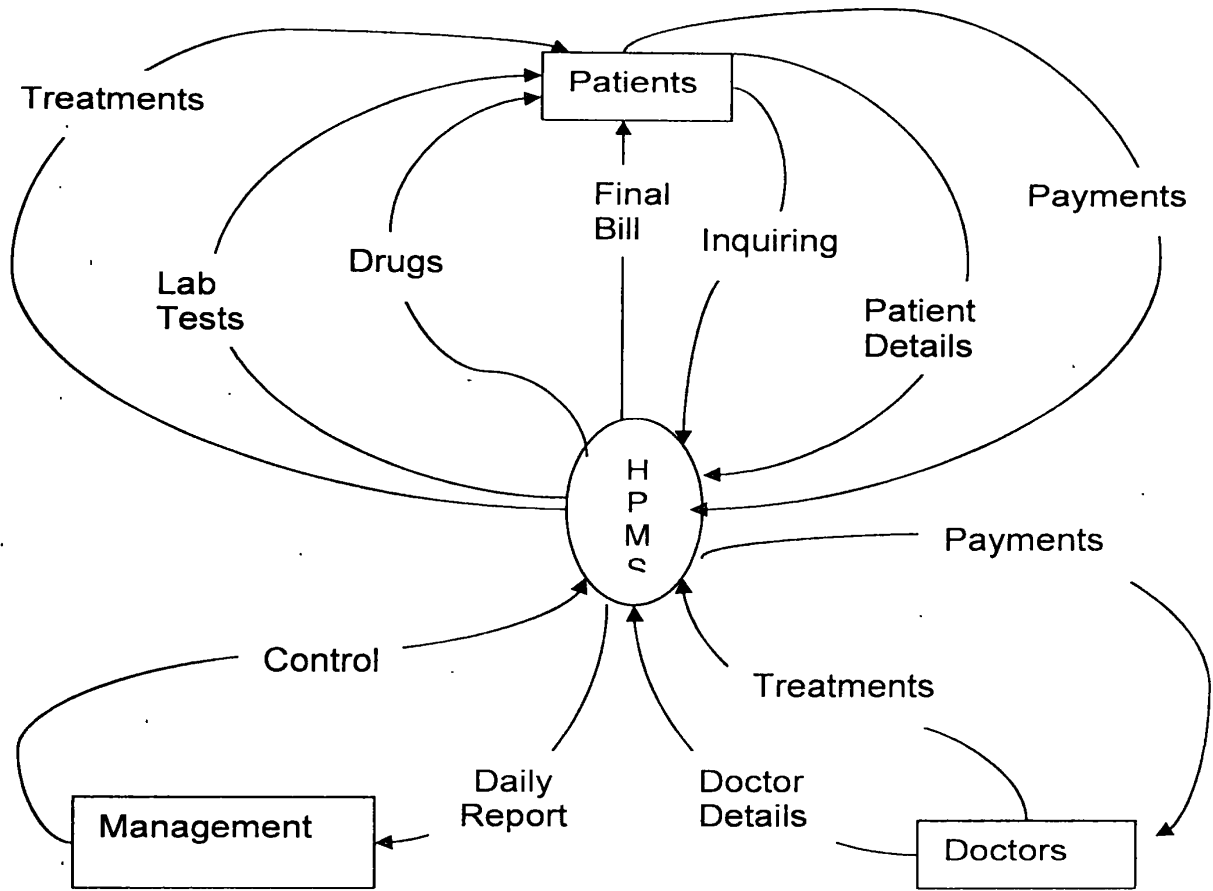
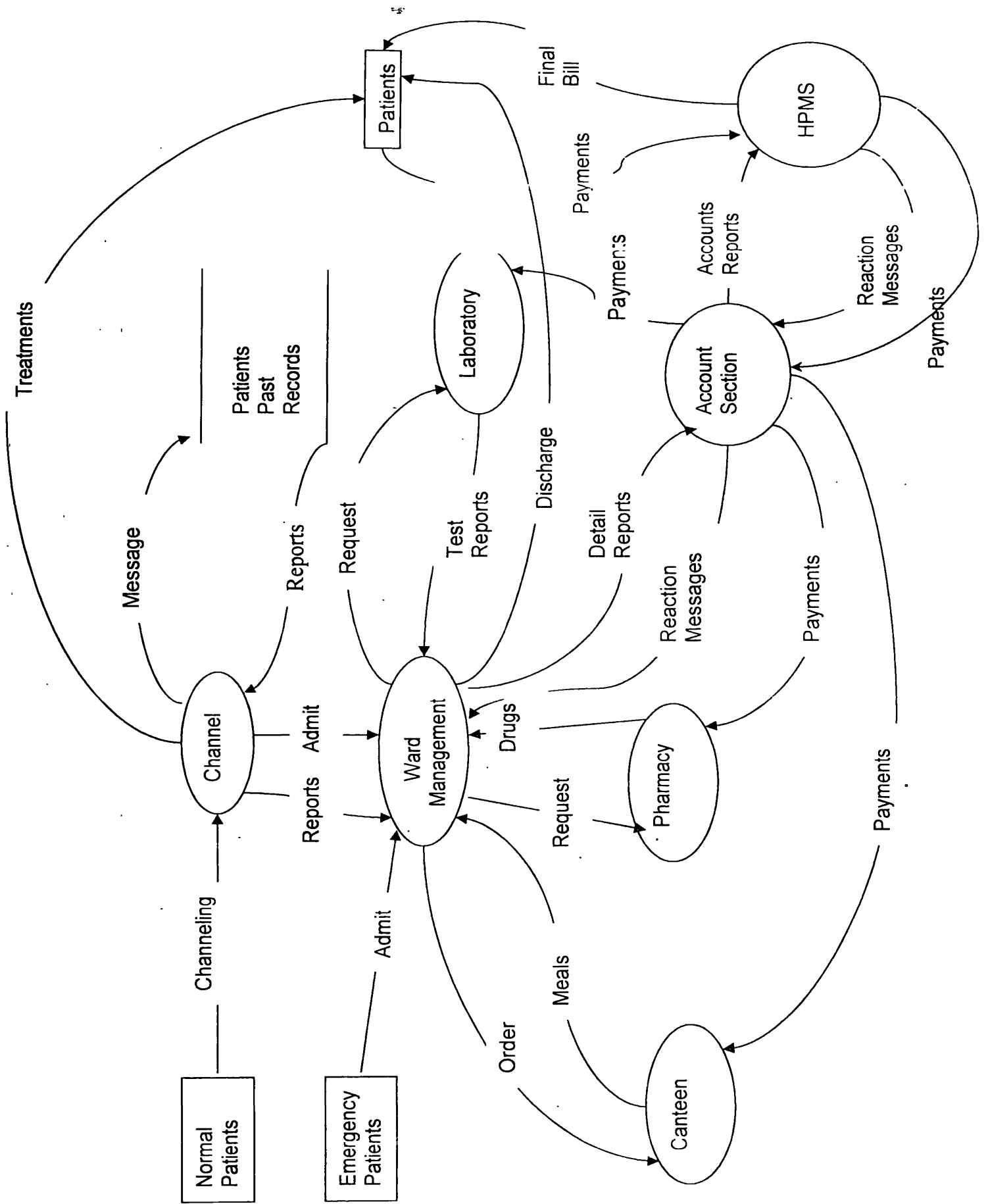


Figure 3.2 Context Level DFD for HPMS



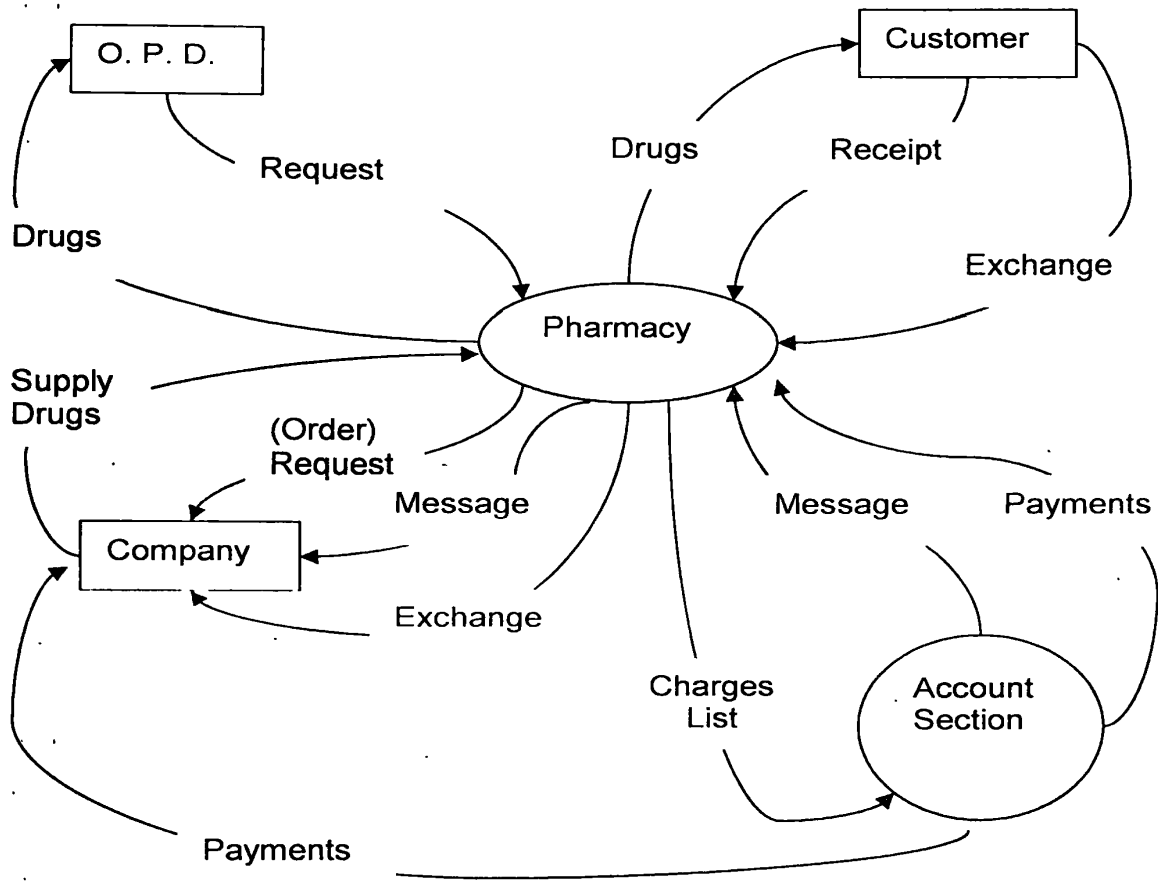


Figure 3.4 Top Levels DFD for Pharmacy Section

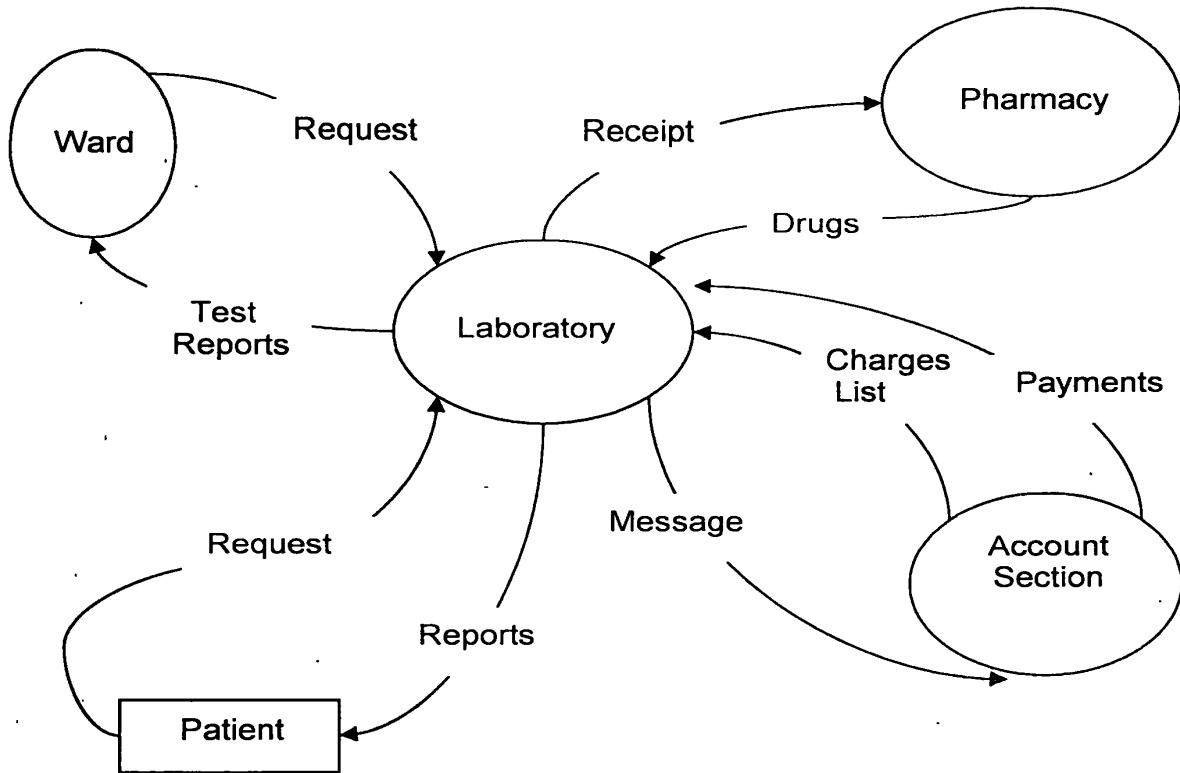


Figure 3.5 Top Levels DFD for Laboratory Section

3.3 System Pseudo codes

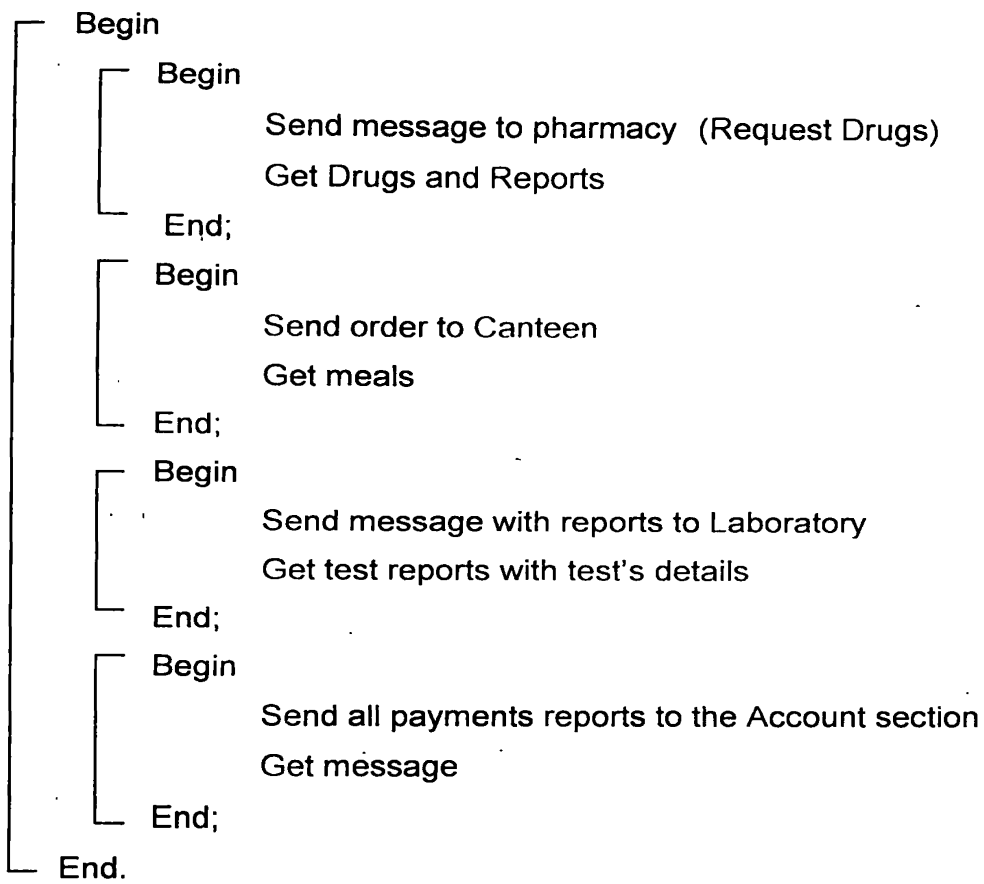
Pseudo code for Channelling

```
Begin
  If (Normal Patient)
    Channel
    Admit in ward
    Then get treatments and leave
  Else (Emergency Patient)
    Admit in ward
  End if;
  Begin
    Send message to the patient's past data stores
    Get reports from data stores
  End;
End.
```

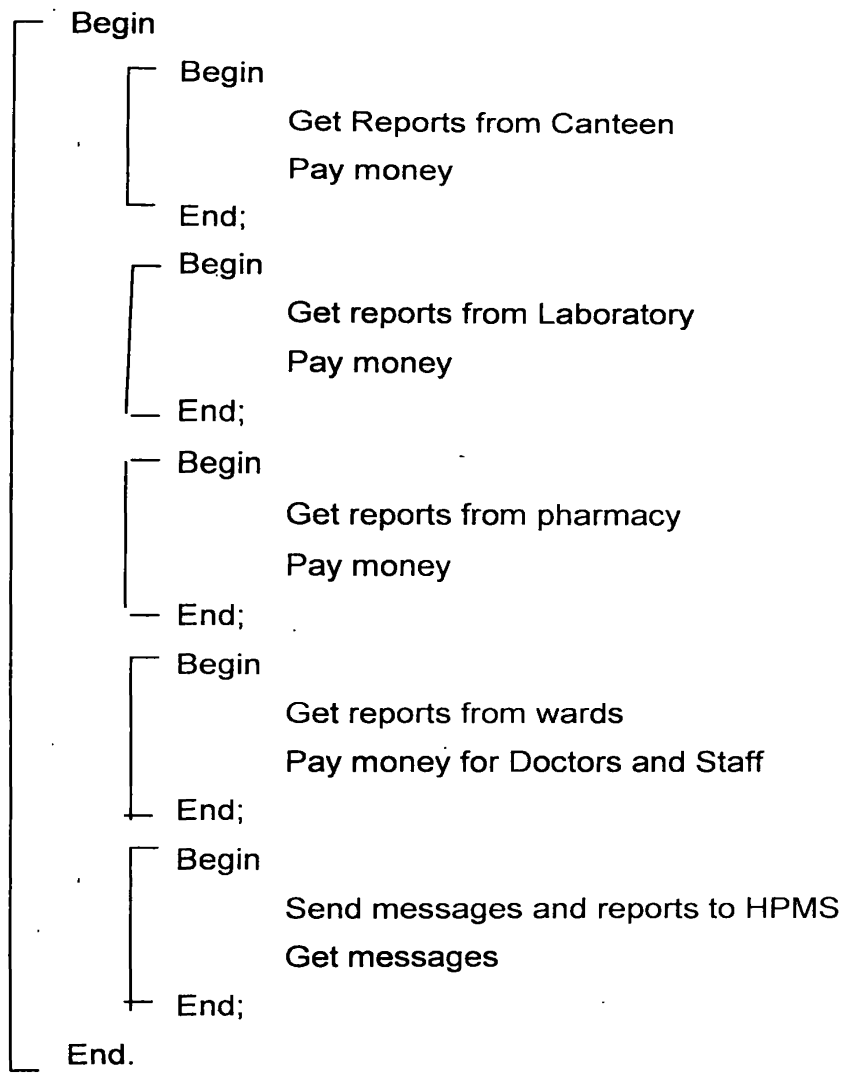
Pseudo code for HPMS

```
Begin
  Get Accounts reports from Account Section
  Send messages to Account Section
  Prepare and send Final Bill to Discharged Patients
  Get money from patients
  Pay money to the Account section
End.
```

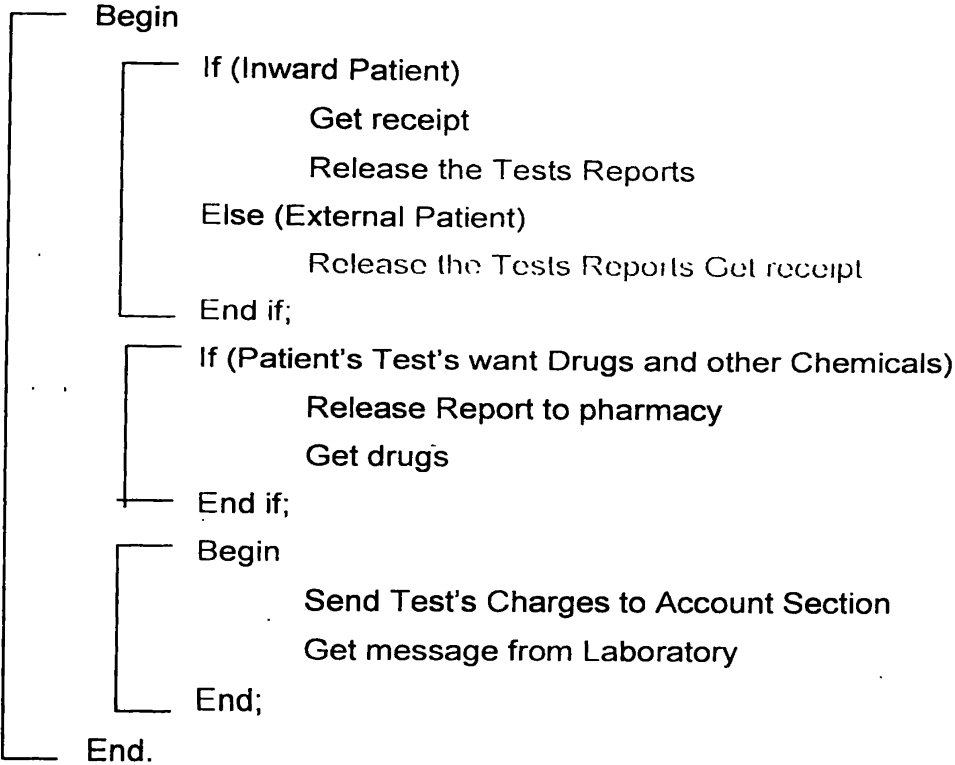
Pseudo code for Ward Management



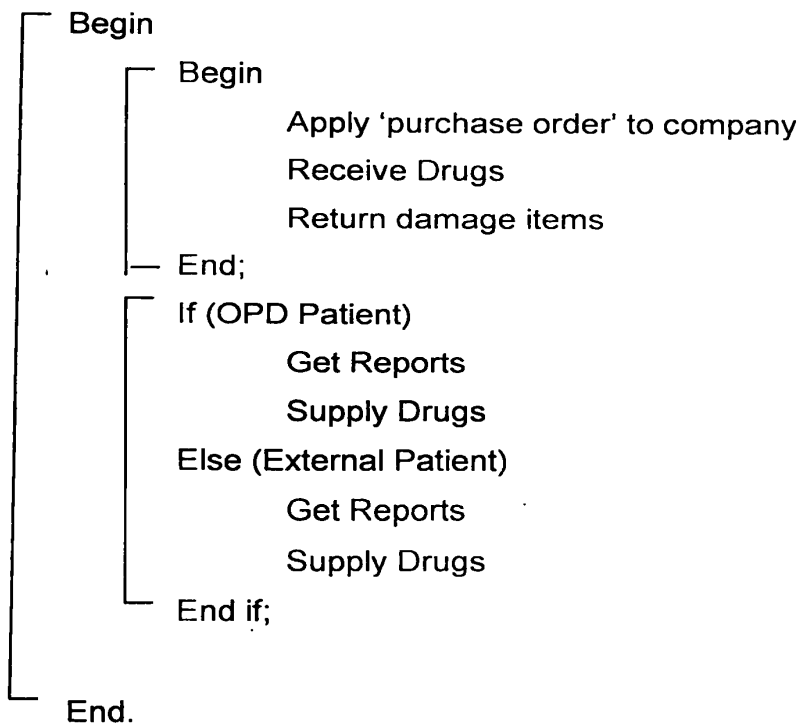
Pseudo code for Account Section



Pseudo code for Laboratory Section



Pseudo code for Pharmacy Section



3.4 Resource Allocation

The project team consists of the following personal

- Project manager/ System Designer
- Technical Designer
- Senior programmer
- Three trainee programmers
- Crystal Report trainee
- Quality Assurance Team

CHAPTER-4

4.0 Results and Discussion

4.1 Results

Select the Patient	
Name of the Patient	<input type="text" value="Miss Wasanthi"/> <input type="button" value="v"/>
<input type="button" value="Submit"/>	
Enter Data	
Date of Discharge:	<input type="text" value="2003-02-11"/>
Time of Discharge:	<input type="text" value="16:38:58"/>
Final Bill	
BHT No:	<input type="text"/>
Date of Discharge:	<input type="text"/>
Time of Discharge:	<input type="text"/>
Name of the Patient:	<input type="text"/>
Date of Admit:	<input type="text"/>
Time of Admit:	<input type="text"/>
Room No:	<input type="text"/>
Bed No:	<input type="text"/>
Consultant Dr.	<input type="text"/>

Rent days:	<input type="text"/>
Room Charges:	<input type="text"/>
Blood Charges:	<input type="text"/>
Urine Charges:	<input type="text"/>
Hormone Charges:	<input type="text"/>
X-Ray Charges:	<input type="text"/>
Scanning Charges:	<input type="text"/>
Pharmacy Drugs Charges:	<input type="text"/>
Net Charges:	<input type="text"/>
Thanks Patients	

Structure of Final Bill for HPMS

4.2 Discussion

Before developing the system, I went to see the existing system in Suwasewana (Pvt.) Hospital. I am discussed with the director and staff about the system. There was not proper system. But I am taken rough idea of their system. They are given some of document of their system. I am taken some of requirements from these documents.

The provided software system works following ways. The first step is, enter data to the first interface. Such as, Date of admit ,Time of admit , Status , First Name , Last Name ,Sex ,Address, Phone No, Ward No, Consultant Dr. And so on. Here Date and

Time are automatically generated. It is easy and correct. All of the data is saved in Patient Table in Database.

The second step is, select the test type and enter amount of charges. In here can do update also. In this stage patient should not pay. All of charges should pay when patient discharge. But receipt can give to the patient before going wards. This data is saved in charges Table.

The next step is, do experiments and enter testing details such as Blood details, Urine Details, Hormone Details. Before enter the data the patient name must be selected.

This data also saved in Blood, Urine, Hormone tables. Pharmacy Details also can add by creating the Pharmacy tables. This all of the data can view, update and delete.

After selecting patient name the Final Bill is automatically created, when patient is discharged. This is final step of the project. Final system can run on Apache.

If have not computerized system, all of data enter in manually. In manually recording some of data can miss and very difficult. Management is difficult and complex too. Patient details are not stored safely. Since patient treatment details is very important at patient's future treatments. But the computerized system, most of the data records automatically. Those decrease the mistakes.

Database is created using MySQL in the Linux platform. So the database has higher security. Here database administration is very important. MySQL is different from SQL. But it is not big difference. SQL is running on the Win NT system (Windows Environment). MySQL is running on the Linux Network (Linux Environment).

This is provided for the General Hospital. This software can update for the specialised Hospital too. I had to work very hard to archive that. This project has been very good experience for me.

5.0 References

- David Pitts, Bill Ball, (1998) Red Hat Linux Unleashed. Sams Publishing, Third Edition, PP. 5-500.
- Frank Boumphrey, Cassandra Greer, Dave Raggett, Jenny Raggett, Sebastian Schnitzenbaumer, Ted Wugofski, (2000) Beginning XHTML. Shroff Publishers, First Edition, PP. 1-200.
- Ion Sommerville, (1999) Software Engineering. UK, Fifth Edition, PP. 20-400.
- Jesus Castagnetto, Harish Rawat, Sascha Schumann, Chris Scollo, Deepak veliath, (1999) Professional PHP Programming. Published by Wrox Press Ltd., UK, PP.5-705.
- Paul DuBois, (1999) MySQL. BPB Publications Techmedia, Indian Edition, PP: 5-700.
- Perry Edwards, (2001) System Analysis and Design. Singapore, PP. 25-200.

Web tutorials/on line courses

- www.Howto.com
- www.Mysql.com
- www.PHP.com
- www.Redhat.com

Appendix

Appendix -A

Tables: Tables for finalised database

Table: Finalised Patient Table

Fields	Type	Length
<u>ID</u>	int , auto_increment, primary key	20
BHTNo	int	10
EntryTime	time	
EntryDate	date	
Status	enum('Mr.', 'Mrs.', 'Miss.', 'Ms.')	
FirstName	varchar	20
LastName	varchar	20
FullName	varchar	40
Sex	enum('Male', 'Female')	
WardNo	int	10
Age	int	10
Address	varchar	20
PhoneNo	int	10
RoomNo	int	10
BedNo	int	10
Consultant	varchar	30

Table: Finalised Blood Table

Fields	Type	Length
<u>ID</u>	int, auto_increment, primary key	20
PaID	int	20
Date	date	
Bgroup	enum ('A', 'B', 'AB', 'O')	
Density	varchar	10
Volume	varchar	10

Table: Finalised Urine Table

Fields	Type	Length
<u>ID</u>	int, auto_increment, primary key	20
PaID	int	20
Date	date	
Sugar	varchar	11
Water	varchar	10
Cells	varchar	15

Table: Finalised Hormone Table

Fields	Type	Length
<u>ID</u>	int, auto_increment, primary key	20
PaID	int	20
Date	date	
HormoneType	varchar	20
Others	varchar	10

Table: Finalised Test Table

Fields	Type	Length
TestNo	int	10
TestName	varchar	20
Active	enum('Yes','No')	

Table: Finalised Charges Table

Fields	Type	Length
<u>ID</u>	int, auto_increment, primary key	20
TestNo	int	10
PaID	int	11
Amount	float	10,2

Table: Finalised Final Table

Fields	Types	Length
PaID	int	20
DischargeDate	date	
DischargeTime	time	

Appendix -B

Tables: Attributes for finalised database

Amount	Amount of the each tests. This is float type. The length is ten integer and two decimals. (10,2)
Bed No	What is bed number of the patient, This is integer type. Maximum length is ten.
Bgroup	What is the patient blood group, It means A, B, AB or O.
Cells	What types of cells are in the Urine, It means red blood cell, white blood cell so on. This is varchar type. Maximum length is fifteen characters.
Consultant	Who is the consultant doctor for each patient. This is varchar type and maximum length is thirteen Characters.
Date	This is applied in Blood, Urine, Hormone tables. It means at what day patient did test. This is data type.
Db_name	Database Name. All of the tables are within this.
Density	This means how much density of the blood. varchar type and maximum characters are ten.
EntryDate	That means, at what day the patient is admitted. This is data type.

EntryTime	That means, at what time the patient is admitted. This is time type.
FirstName	First name of the patient. This is varchar type and maximum length is twenty characters.
HPMS	Hospital Patient Management System. This is developed system. This system is computerized all of hospital details.
Others	What are the other changes of the hormone. It means are there virus, bacteria so on. This is varchar type and maximum length is ten characters.
PaID	This is same as ID of the Patient. The patient ID is generated automatically. This is integer type.
Sex	This is sex of the patient (Male or Female)
Status	This is status of the patient. It means , is the patient Mr. , Mrs. , Miss., Ms.
Sugar	How much sugar is included in Urine
Tb_name	Table Name. All of tables have table name. This is case sensitive. It means one table has only one name. Can not use another name. Table name should remember using lower and upper case.
TestName	Test Name of the test. As example Blood test, Urine test. This is character type and maximum length is twenty characters.
Volume	How much volume of the blood. This is character type and maximum length is ten characters.
WardNo	For what ward patient is admitted. This is integer type and maximum length is ten.
Water	What volume of the water in Urine. This is character type and maximum length is ten characters.

Appendix-C

System Interfaces

The screenshot shows a Microsoft Internet Explorer browser window with the address bar containing `http://sing.ids.lk/~wasathi/hospital/`. The browser's menu bar includes File, Edit, View, Favorites, Tools, and Help. The address bar also shows the current page title and a 'Go' button. Below the address bar, there are several icons for navigation and search, including Back, Forward, Stop, Search, Favorites, and Media. The main content area of the browser displays a web page with a sidebar on the left and a main content area on the right. The sidebar contains several underlined links: [Insert PERSONAL Data](#), [Insert BLOOD Data](#), [Insert URINE Data](#), [Insert HOMONE Data](#), [Tests & Charges Data](#), [Final Bill](#), and [History Details](#). The main content area features a large heading 'Insert Data' and a form titled 'Enter Data'. The form contains the following fields and values:

Enter Data	
Entry Date	2003-02-11
Entry Time	16:37:24
Status	Mr. ▾
First Name	
Last Name	
Age	
Address	
Sex	Male ▾
Phone No	
Ward No	
Room No	
Bed No	

The browser's status bar at the bottom shows 'Done' on the left, 'Internet' on the right, and a taskbar with several open applications: 'wasathi@sin...', 'wasathi@sin...', 'http://sing.d...', 'Document1 - ...', and 'Address'. The system clock in the bottom right corner displays '4:49 PM'.

[Insert PERSONAL Data](#)

[Insert BLOOD Data](#)

[Insert URINE Data](#)

[Insert HOMONE Data](#)

[Tests & Charges Data](#)

[Final Bill](#)

[History Details](#)

Select Data

Patient Name Miss.wasanthi kasturi

Insert Blood Data

Enter Data	
Blood Group	A
Blood Density	
Blood Volume	
Date	2003-02-11
<input type="button" value="submit"/>	<input type="button" value="Cancel"/>

[View Data](#)

[Delete Data](#)

- [Insert PERSONAL Data](#)
- [Insert BLOOD Data](#)
- [Insert URINE Data](#)
- [Insert HOMONE Data](#)
- [Tests & Charges Data](#)
- [Final Bill](#)
- [History Details](#)

Select Data

Patient Name Miss.wasanthi kasturi

Insert Urine Data

Enter Data	
Amount Of Sugar	<input type="text"/>
Have Any Cells , What are them?	<input type="text"/>
Amount Of Water	<input type="text"/>
Date	2003-02-11
<input type="button" value="Submit"/>	<input type="button" value="Cancel"/>

[View Data](#)

http://sing.ids.lk/~wasathi/hospital/ - Microsoft Internet Explorer

File Edit View Favorites Tools Help



Back



Search

Favorites



Media



Address http://sing.ids.lk/~wasathi/hospital/



Go

Links Customize Links Free Hotmail Windows Windows Media

[Insert PERSONAL Data](#)

[Insert BLOOD Data](#)

[Insert URINE Data](#)

[Insert HOMONE Data](#)

[Tests & Charges Data](#)

[Final Bill](#)

[History Details](#)

Select Name

Patient Name Miss wasanthi kasturi

Insert Homone Data

Enter Data	
Change Homone	
Other Changes	
Date	2003-02-11
<input type="button" value="submit"/>	<input type="button" value="Cancel"/>

[View Data](#)

[Delete Data](#)

Done

Internet

Start

wasathi@sin...

wasathi@sin...

http://sing.id...

Document1 - ...

Address



4:50 PM

[Insert PERSONAL Data](#)

[Insert BLOOD Data](#)

[Insert URINE Data](#)

[Insert HOMONE Data](#)

[Tests & Charges Data](#)

[Final Bill](#)

[History Details](#)

Select Tests' Amounts

Select Data

Name Miss.wasanthi.kasturi ▾

Enter Data

BloodTest

UrineTest

HomoneTest

XRayTest

ScanningTest

ECGTest

[View Data](#)

[Delete Data](#)

- [Insert PERSONAL Data](#)
- [Insert BLOOD Data](#)
- [Insert URINE Data](#)
- [Insert HOMONE Data](#)
- [Tests & Charges Data](#)
- [Final Bill](#)
- [History Details](#)

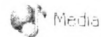
Select a Patient

Name of the Patient Miss.wasanthi kasturi



Search

Favorites



Address <http://sing.ids.lk/~wasathi/hospital/>



Links [Customize Links](#) [Free Hotmail](#) [Windows](#) [Windows Media](#)

[Insert PERSONAL Data](#)

[Insert BLOOD Data](#)

[Insert URINE Data](#)

[Insert HORMONE Data](#)

[Tests & Charges Data](#)

[Final Bill](#)

[History Details](#)

Select The Patient

Name Of The Patient Miss wasanthi kasturi

Enter Data

Date of Discharge: 2003-02-11

Time of Discharge: 16:38:58

Submit

Cancel



Start

wasathi@sin...

wasathi@sin...

http://sing.i...

Document1

Address

Internet



4:51 PM

F:\Documents and Settings\IDS\Desktop\pro\bin\final.html - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address F:\Documents and Settings\IDS\Desktop\pro\bin\final.html

Final bill

BHT No :	120
Date of Discharge :	2003-03-03
Time of Discharge :	14:00:13
Name of Patient :	Mr.Dimuth Sepelage
Date of Admit:	2003-01-23
Time of Admit:	13:31:14
Room No :	123
Bed No :	1200
Consultant Dr. :	Nalini Wikramasingha

Rent Days	20
Room Charges	-2297
Blood Charges	2000.00
Urine Charges	
Homone Charges	6000.00
X-Ray Charges	
Scanning Charges	200.00
E.C.G. Charges	

Done My Computer

start 3 1/2 Floppy (A:) NEW F:\Documents and ... Interfaces - Micros ... 9:14 AM

[Insert PERSONAL Data](#)

[Insert BLOOD Data](#)

[Insert URINE Data](#)

[Insert HOMONE Data](#)

[Tests & Charges Data](#)

[Final Bill](#)

[History Details](#)

Blood Details

Patient Name	Blood Group	Blood Density	Blood Volume	Date
--------------	-------------	---------------	--------------	------

Urine Details

Patient Name	Sugar Amount	Have Cells ?	Water Amount	Date
--------------	--------------	--------------	--------------	------

Homone Details

Patient Name	Homone Type	Other Details	Date
--------------	-------------	---------------	------

Previous

National Digitization Project

National Science Foundation

Institute : Sabaragamuwa University of Sri Lanka

1. Place of Scanning : Sabaragamuwa University of Sri Lanka, Belihuloya

2. Date Scanned : ..2017..09..25.....

3. Name of Digitizing Company : Sanje (Private) Ltd, No 435/16, Kottawa Rd,
Hokandara North, Arangala, Hokandara

4. Scanning Officer

Name : ..B.A.C. Badarayanan.....

Signature : ..Cul.....

Certification of Scanning

I hereby certify that the scanning of this document was carried out under my supervision, according to the norms and standards of digital scanning accurately, also keeping with the originality of the original document to be accepted in a court of law.

Certifying Officer

Designation : ..Librarian.....

Name : ..T. N. Neighsoorei.....

Signature : .......

Date : ..2017..09..25.....

Mrs. T. N. NEIGHSOOREI
(MSSc, PhD, ASLA, BA)
Librarian
Sabaragamuwa University of Sri Lanka
P.O. Box 02 Belihuloya, Sri Lanka
Telephone 94 45 2280045
Fax 994 45 2280045

“This document/publication was digitized under National Digitization Project of the National Science Foundation, Sri Lanka”