**CHAPTER 4**

**DETAILED DESIGN**

**4.1 PURPOSE**

The purpose of this chapter is to provide a detail idea about the various modules present in this project and also provides details about the functionality of each module .It provides details about the input given to each module, the expected output and a brief description about each module. It also provides details about the different methods used. Once the high level design is completed the next stage is to perform detailed design of the software and hardware. While the high level design focuses on the tasks to be performed, the detailed design concentrates on how these can be performed. It describes the modules in terms of the data structure used and the algorithms, which explain how the modules are implemented.

Detailed Design of a system gives more in picture of most components described in the System Architecture. Other lower-level components and subcomponents also described as well. Each subsection of this section will refer to or contain a detailed description of system software and system hardware components. The Detailed Design specifies the structured charts, flow charts, functionality of each module, etc.

Most components Designed in the System Architecture section will require a more detailed discussion. Other lower-level components and subcomponents may need to be described as well. Each subsection of this section will refer to or contain a detailed description of system software components. Here we cover the following software component attribute.

**4.2 USER INTERFACE DESIGN**

The design of the system includes the proper display of the Administrator GUI and the Client GUI. It involves the interface architecture. The interface design comprises of the design of the user interface with WUI standards where the administrator can add the users into the system for whom the printing is to be tracked and managed.

After all the users are added, all jobs can be tracked by the admin and quota can be allocated. Even the bill and report generation option is also provided. The Admin can add both the users as well as the printers to different departments to which they belong.

**4.2.1 Hardware Interfaces**

The interfacing between printer and our software is done with the help of various managed and unmanaged windows library. These libraries are provided by Visual Studio to talk to printers. Our software mostly concerns with network printers which have snmp facility to communicate with printers.

**4.2.2 Software Graphical User Interfaces**

The entire printer management system consists of four modules which are the administrator GUI, Client GUI, and 2 windows services running in the background. One of the windows service is responsible for tracking when a print job is arrived and allowing the print based on quota of a specific user. Other service is used to update the records in the admin database after some fixed period of time. The different modules are discussed below in detail.

**4.3 MODULE 1: ADMINISTRATOR GUI**

The administrator GUI is used by the administrator for checking the print jobs and various other functionalities. The design algorithm for the same is :

* Step1: Start
* Step2: Displays the Admin Login for the UserName and Password
* Step3: Start Up design gives details of the number of printers connected in the LAN as well in Database, Pages Printed till date, Users in LAN, System UpTime and Printers status.
* Step3: Users design, it provides all the neccessary information about the users.
* Step4: Printers design, it gives the information about the printers.
* Step5: Jobs design, it involves providing all usefull information about users job as well as printers job.
* Step6: Billing design, its a financial page, it invloves computing the amount for pages printed by the particular user as well as printer.
* Step7: Settings design, it consists of many sub designs such as Users,Printers,Groups and Quota.
  + Users design, in which it allow the Admin to add the users into the database,once after the system successfully retrived the user in LAN.
  + Printers design, in which it allow the Admin to add the printers into the database after the system successfully retrived the printers in LAN.
  + Groups design, which it permit the Admin to add the user to the particular group and upadting the user quota at one shot as well as updating existing user group.
  + Quota design, the area in which it consists of all user quota requests and printer quota requests and Admin responds to those requests.
* Step8:End.

**4.4 MODULE 2: CLIENT GUI**

The client GUI is used by all the users to login and check their quota or send quota request to administrator. The design algorithm for the same is :

* Step 1: Start
* Step 2: Start this application as startup and minimized to tray
* Step 3: Place buttons for QuotaLeft and to SendQuotaRequests in design view
* Step 4: Define what happens when CheckQuota is clicked

if file exists then

read quota from file and display it to user

else display unable to retrieve quota

* Step 5: Define what happens when SendRequests is Clicked

Create a Mail Message which includes the name of the user who requested the quota and also the requested quota.

MailMessage mail = new MailMessage();

mail.IsBodyHtml = true;

mail.From = new MailAddress("pesitpms@gmail.com");

mail.To.Add("bhutani92@yahoo.com");

mail.Subject = "Request for Quota by User " + UserName;

mail.Body = "User " + UserName + " has requested for " + Quota\_Requested + " Quota Allocation. Please do the needful.";

SmtpClient SmtpServer = new SmtpClient("smtp.gmail.com", 587);

SmtpServer.EnableSsl = true;

SmtpServer.Send(mail);

if Success then

Show Message Sent to Admin DialogBox

else Show Cannot requests for Quota... try Again Later...

* Step 6: End

**4.5 MODULE 3: PRINTER STATUS SERVICE**

This is a windows service which tracks all the print jobs and stores to the local database. This service is event based. When a job arrives in the print queue, it generates an event and the necessary steps are taken. The algorithm for the same is :

* Step 1: Start
* Step 2: Initialize private variables
* Step 3: Import all relevant Dll Functions of WinAPI
* Step 4: Define the Constructor
* Step 5: Create a new thread to call a method ThreadProc within OnStart procedure

Thread t1 <- new Thread(new ThreadStart(ThreadProc))

t1.Start()

* Step 6: Define the ThreadProc method

Implement a lock over this method

Update information such as Username MacAddress IPAddress to the database using a flag variable so that it gets updated only once

Define LocalPrintServer

myServer <- new LocalPrintServer()

Define PrintQueueCollection

myQueueCollection <- myServer.GetPrintQueues()

for each printer in myQueueCollection do

\_mrEvent[i] <- new ManualResetEvent(false)

\_waitHandle[i] <- null

for each PrintQueue in myQueueCollection do

if the PrintQueue name is not added to dictionary then

add PrintQueue Name to Dictionary

Make a connection to Printer using OpenPrinter method

if a valid handle is found then

Register the printer for a change notification

Capture the handle

Wait for change notification from the printer queue

* Step 6: Define the CallBack function which is called when a change notification occurs

ManualResetEvent \_pnwcEvent <-(ManualResetEvent)state

pDictCount <-\_handleDict[\_pnwcEvent.Handle

PrintQueue myPrintQueue <- new PrintQueue(myServer, CountToPrinterNameDict[pDictCount], PrintSystemDesiredAccess.AdministratePrinter)

myPrintQueue.Pause()

ManagePrinter(myPrintQueue, myServer)

Reset the Event and wait for next Event

* Step 7: Define ManagePrinter method

Use Win32\_Printer WMI Query to get Port Name of associated Printer

Use SNMP to derive NumberOfPages and Cartridge level from printer

for each job in PrintQueue do

Pause the job

Check for Valid Quota for that user

if Quota Available then

Print the Job

else Remove the Job from PrintQueue

Copy all Job Information to a separate XML file

* Step 8: Define the OnStop Procedure

Free all variables from memory

* Step 9: End

**4.6 MODULE 4: READXML SERVICE**

This is another windows service which reads the local database and updates into the main database on the server computer. This is timer based and checks for new information to be updated after every 5 minutes. Once the information is updated, the records are deleted.

The algorithm for the same is :

* Step 1: Start
* Step 2: Use System.Timers to write the job information at regular intervals
* Step 3: Iinitialize the private variables
* Step 4: Define the constructor
* Step 5: Define OnStart Procedure

Thread t1 <- new Thread(new ThreadStart(ThreadProc))

t1.Start()

* Step 6: Define ThreadProc Procedure

timer.Elapsed +=: new ElapsedEventHandler(OnElapsedTime)

timer.Interval <- 60000;

timer.Enabled <- true;

* Step 7: Define EventHandler

if File Exists then

Copy all job information to database

else Create the file and goto step 7

* Step 8: Define the OnStop Procedure

time.Enabled <- false

* Step 9 :End

**4.7 SUMMARY**

This chapter revealed the various details about the design. We saw the different modules which are the core of the entire project. All the design features are implemented in the next chapter.