MINUTE MANAGEMENT SYSTEM

by

Akhona Mahangu

A thesis submitted in partial fulfilment of
the requirements for the degree of

Bsc Computer Science (Hons)

University of the Western Cape

2010

Date: November 5, 2010

University of the Western Cape

Abstract

MINUTE MANAGEMENT
SYSTEM

By: Akhona Mahangu

Abstract

Supervisor: Professor Venter
Co- supervisor: Mr. Connan
Department of Computer Science

The purpose of this project is to develop a system to manage the processes of meetings from the formation of the agenda, minutes taking to the follow-up tasks. The minute manager will be used to record minutes of a meeting and manage details connected to the meeting. The minute manager will send information to the meeting participants, give them feedback about the decisions made and inform them of what is expected from each of them.

TABLE OF CONTENTS

Table of Contents ..... ***i***

List of figures ..... ***iv***

List of Tables ..... ***v***

Acknowledgments ..... ***vi***

Glossary ..... ***vii***

Chapter1 ..... **1**

User requirements Document 1

Introduction 1

User's view of the problem 1

What is expected from a software solution? 2

What is not expected from a software solution? 2

Summary 3

Chapter 2 ..... **4**

Requirements analysis document 4

Introduction 4

Designer's interpretation of the problem and possible solution 4

Identify existing solutions 5

Modelling the solution 6

Technologies to be used for developing the system 6

Possible ways of testing the solution 6

Paper-based Prototyping 6

Usability testing 6

Complete System Testing 7

Summary 7

Chapter 3 ..... **8**

User interface specification 8

Introduction 8

Description of the complete user interfaces 8

What the interface looks like to the users 9

ADMINISTRATOR 10

USER 13

Home page 13

How the user interface behaves 14

User 14

Administrator 14

How the users interacts with the interface 15

Summary 19

Chapter 4 20
High level design 20
Introduction 20
Detailed breakdown of the technical solution 20
Class Diagram 21
Detailed interaction between subsystems 22
Summary 22

Chapter 5 23
Low level design 23
Introduction 23
Inner details of class atttributes and methods 23
Psuedo code 25
Login function 25
Create new agenda 26
Notification of task 26
Summary 27

Chapter 6 28
Code documentation 28
Introduction 28
Login script documentation 28
Agenda creation script documentation 31
Print.php script documentation 36
Display_info.php script documentation 38
Summary 39

Chapter7 40
SYSTEM TESTING 40
Introduction 40
Test plan 40
Functionality testing 40
Usability testing 41
Test results and discussion 41
Functionality testing 42
Usability 43
Recommendations 44
Summary 44

Chapter8 45
USER GUIDE 45
Introduction 45
Administrator user guide 45
Login 45
Create Agenda 46
Save agenda 47
Record minutes 49
Handling of saved minutes 49

General(committee member) user guide 50
Login 50
Menu 51
Summary 52

Chapter9 53
CONCLUSION 53

Appendices 54
Appendix A 54
Appendix B 55
Appendix C 58
Appendix D 59
Appendix E 60
Appendix F 61
Appendix G 62
Appendix H 63

Bibliography 64
Bibliography 64

Index 65

LIST OF FIGURES

NumberPage
Figure 1: Migration from paper based to web based system 5
Figure 2: Login page 9
Figure 3: Home 10
Figure 4: Agenda fields 10
Figure 5: Agenda created 11
Figure 6: Agenda request 12
Figure 7: Select committee 13
Figure 8: Administrator's home page 13
Figure 9: Activity diagram 15
Figure 10: Activity diagram for the administrator 16
Figure 11: Use case diagram for all users' interaction with the system 18
Figure 12: The class diagram 21
Figure 13: Interaction between subsystems 22
Figure 14: functionality 42
Figure 15: Usabilty 44
Figure 16 Login 46
Figure 17: Menu 46
Figure 18: Agenda fields 47
Figure 19: Add more fields 47
Figure 20 : Saving agenda template 48
Figure 21: Navigation page 48
Figure 22: Recording meeting minutes 49
Figure 23: handling saved minutes 50
Figure 24: User_Login 51
Figure 25: User menu 51

LIST OF TABLES

NumberPage
Table 1: User interface specification *14*
Table 2: Administrator interface interaction *14*
Table 3: Description of attributes for classes *24*
Table 4: Description of the inner details of each method's presence in the classes *24*
Table 5: Participant's details *41*

Akhona Mahangu (2010)

ACKNOWLEDGMENTS

Firstly, I send my sincere gratitude to the Almighty God who's given me the strength to carry out this project.

Secondly, to my wonderful supervisor Prof. IM Venter, thank you for your guidance, support and encouragement. Thank you very much for investing your skills on me and reminding me that there's always room for improvement.

A special thanks goes to Mr Connan for the advice and contributions to this project, and not forgetting Mr Kruger for the geek advice and working strategy.

My friends, classmates and colleagues who inspired and motivated me to do well and made it fun to sit and work. Thank you guys for your support and for helping me with my problems. A special thanks to Ryno and Ludwe for your selflessness and assistance.

Last but not least thank you to my family, for the support and making me believe that I can do anything and succeed.

GLOSSARY

Apache is a Web Server that is distributed under an "open source" license.

API Application Programming Interface enables a software program to interact with other software, much in the same way that a user interface facilitates interaction between humans and computers.

CLI Command Line Interface allows a user to interact with a computer operating system or with software by typing commands in the command line in order to perform tasks. CLI is a text-only interface and differs from a graphical user interface (GUI) in which a user clicks on options, or menus by means of a mouse device or touch pad, to select tasks.

GUI Graphic User Interface allows people interaction with programs in more ways than typing; it uses graphical icons, and visual indicators, rather than text-based interfaces.

HTTP Hypertext Transfer Protocol is an application-level protocol used for distributed, collaborative and hypermedia information systems.

JavaScript is a language that enables programmatic access to objects within other applications, primarily used in the client-side of the web development.

MySQL is an open source relational database management system that is based on the structure query language.

PHP Is a hypertext processor that allows web developers to create dynamic content that interacts with the database.

PHP MyAdmin is a free interface that facilitates the use of php and managing the MySQL database on the server.

UIS User Interface Specification captures the details of the software user interface and converts them in a written document. The specification covers all possible actions that an end user may perform and all visual, auditory and other interaction elements.

UML Unified Modeling Language, a modeling language is software engineering that includes a set of graphical notation techniques to create visual models of software systems.

Chapter 1

USER REQUIREMENTS DOCUMENT

Introduction

Minute management is an activity that has been carried out for many years; this activity involves drawing up an agenda and recording minutes of the meeting. The minute management is traditionally a paper-based system but with technology that can be changed to make it more efficient and effective; which is what this paper is about, creating a web-based system that helps with the generation of an agenda and management of the minutes.

In this chapter the problem from the user's point of view will be discussed and the scope of the problem defined.

User's view of the problem

To get the user's requirements four people were interviewed with a set of probes (See Appendix A). All the interviewees agreed on some of the problems that they experienced but a few more problems were added by each person (See Appendix B). Some of the problems that the interviewees pointed out were as follows:

- The paper-based minute managing system is time consuming (Abbott, 2010) and its functionalities are limiting to the user.

- The existing system being dependant on paper print which is also time consuming and waste of paper.

- Meeting participants always want to be reminded of their duties.

- Existing system does not allow user to put together documents of different formats and number them (Connan, 2010).

What is expected from a software solution?

According to the interviews conducted, these are the user's expectations from the system:

- The interface must be simple, easy to understand and use.
- Meeting minute and agenda must be accessible to all stakeholders
- The system must be secure, that is only registered users should be able to make changes.
- The system must be able to notify the stakeholders of their duties and send a regular reminder until the task has been completed.
- The system should be able to compile all attachments into a document for the meeting irrespective of their formats (i.e. word docs and pdf should be combined and pages numbered).

What is not expected from a software solution?

What is not expected from the system is for the system to automatically capture the minutes of the meeting or for the minutes to be uploaded as a voice recording. The minutes will need to be manually entered into the system.

Summary

In this chapter the user's requirements for the system were stated and problems with the existing system were analysed; which led to the discussion of the requirements and expectations of the system that needs to be developed. In the next chapter the designer interprets the user's requirements and proposes possible solutions to the problems.

Chapter 2

REQUIREMENTS ANALYSIS DOCUMENT

Introduction

In the previous chapter the user requirements were discussed and the problems with the existing system were identified. In this chapter the designer's view of the problem will be explained and a possible solution to the problem(s) will be analysed.

Designer's interpretation of the problem and possible solution

What the users want is a system that will automate some of the processes involved in the management of meeting minutes. The possible solution to the user's problem is to develop a web-based system (see Figure 1) that will assist with the compilation of an agenda for a meeting, store the minutes that will be manually entered in a database after every meeting. The system should send an email to participants that have been assigned specific tasks during a meeting and remind them of their duties at regular intervals until the task has been completed. If a task has been completed the participant will indicate via email and will not be reminded again.

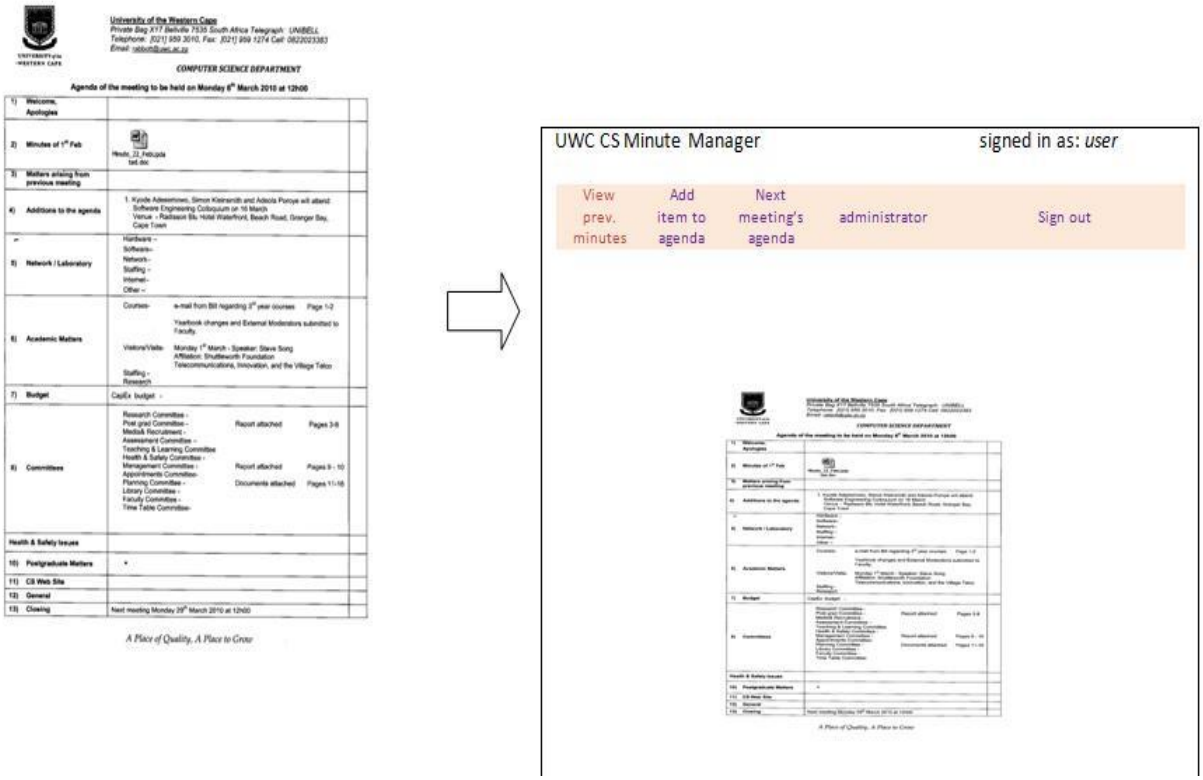

Figure 1: Migration from paper based to web based system

Identify existing solutions

At the moment the managing of tasks, agenda formation and minutes is performed by hand and paper. After every meeting the secretary of the meeting records minutes by hand, types it up and sends it to the participants for correction and to see which tasks were allocated to them. The documentation of this system's processes depend mainly on the secretary. Whether the participants have carried out their task will only be known at the next meeting. Systems that exist for managing minutes were identified while doing research, e.g. the Liteminutes system that records video clips of a meeting and are stored for later viewing [Chiu, P. (2001)].

Modelling the solution

For the analysis of the user requirements Unified Modelling Language (UML) will be used. It will help with modelling the activities of the system and also help in understanding how the system will work. Furthermore, UML will break down the project to make it easier to develop.

Technologies to be used for developing the system

Because the system is web-based MySQL will be used as a database with Apache as a web server. This technology is advantageous since it is easy to use, free (has no cost) and easy to manage and maintain. Some other technologies that could be used when developing the system are: Perl, Php, Java script, HTTP and Php MyAdmin (See Glossary).

Possible ways of testing the solution

PAPER-BASED PROTOTYPING

The paper-based/sketch board prototype will show the expected functionalities of the system on paper using a story board, use cases, etc. The paper-based prototype saves time and money since the testing is done before the actual system is developed; this prototype is advantageous because changes can be made and redone until the desired results are produced with minimal cost.

USABILITY TESTING

Usability testing is a technique used to perform an evaluation of a product's functionality by testing it on a group of user. For usability testing of this project the users that initially gave the requirements will be asked to evaluate the prototype/system. During usability testing selected users will be asked to interact with the system to determine whether the system produces the expected results or performs in the desired manner.

COMPLETE SYSTEM TESTING

The complete system will be tested by performing tasks on the system to check whether the system produces the desired results. When the system has been implemented properly the produced results will match the functional requirement specification. Stress testing will be conducted to make sure that the system is robust and reliable. For example the 4 users would be asked to add items to the agenda at the same time and the system will be evaluated on the performance i.e. whether the task will be successful for all as well as the system's response time.

Summary

In this chapter the designer's view of the problem and the possible solutions were explored. Ways in which the system will be tested were discussed. In the next chapter the user interface specification will be discussed as well as modelling techniques for the development of the prototype.

Chapter 3

USER INTERFACE SPECIFICATION

Introduction

In the previous chapter the user's requirements were analysed and the possible solution was identified. The designer also modelled ways of testing the system that will be designed. In this chapter the user interface will be specified; i.e. describe how the interface will look and behave.

Description of the complete user interfaces

The interface will consist of 3 main functions. Each will be represented by a different interface page: the login page, a page for the viewing the agenda and a minutes archive page; where one can view the minutes of a previous meeting and a page where a new agenda will be requested. The login page allows a registered user access to the system. A user can only view the agenda and minutes when they have logged in. Unregistered cannot perform any functions on the system.

All users will have the same login page but will have different interfaces past the login page. The previledges for the different users are as follows:

- The administrator will have different interfaces for the following functions:
    - Creation of a new agenda and editing of agenda including attaching documents to the agenda.
    - Creating a new user and deleting an existing user.
    - Recording meeting minutes

- The ordinary user will have different interfaces for the following functions:
  - Logging in
  - View minutes of a specific meeting

What the interface looks like to the users

The system will have two users; the registered user, registered administrator. The administrator will compile the agenda and all the notes required for the meeting. These different users will have different privileges on the system and will have different interfaces.

Login page

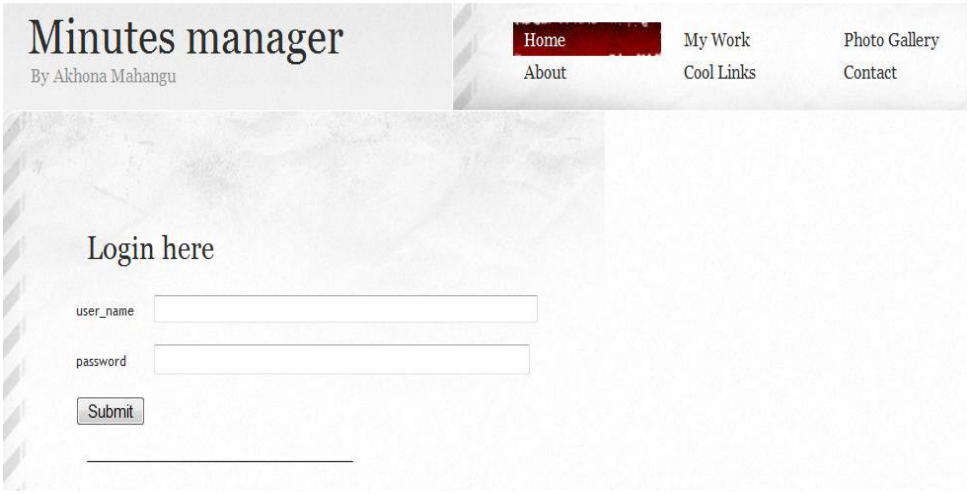

Figure 2: Login page

The user will have to enter a unique username and password, and the system will determine by the their login information whether they are administrator or user and will then show a home page of the user depending on the user type(user/admin) and also the department they belong to (See Figure 2).

ADMINISTRATOR

Home page

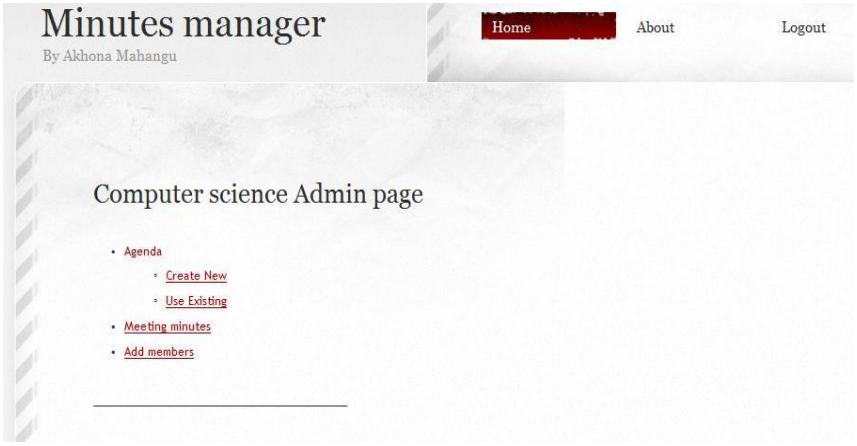

Figure 3: Home

When the administrator has successfully logged into the system then the home appear (see Figure3). The home page offers the administrator the option of choosing the functions that they would like to perform next on the system.

Meeting agenda

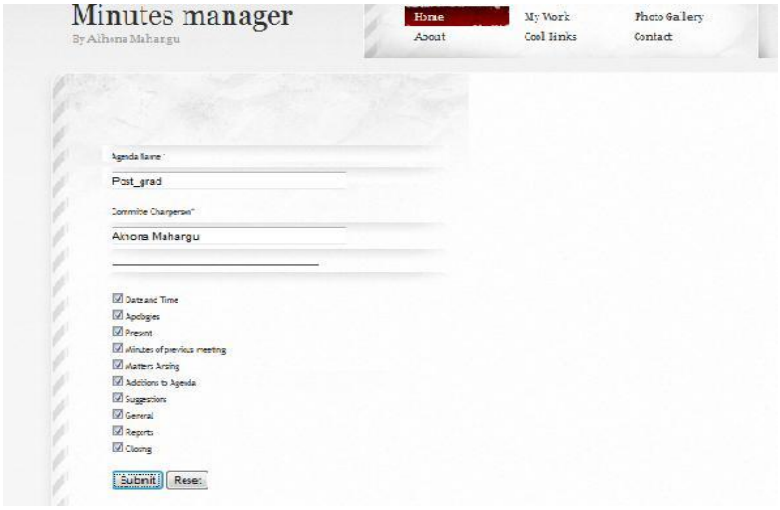

Figure 4: Agenda fields

Figure 4 is the next page that appears when the administrator is creating a new agenda. It allows the administrator to select the fields they want to add to the agenda they're creating. The administrator has to give the agenda a name and also add the chairperson of the committee for which the agenda is being created.

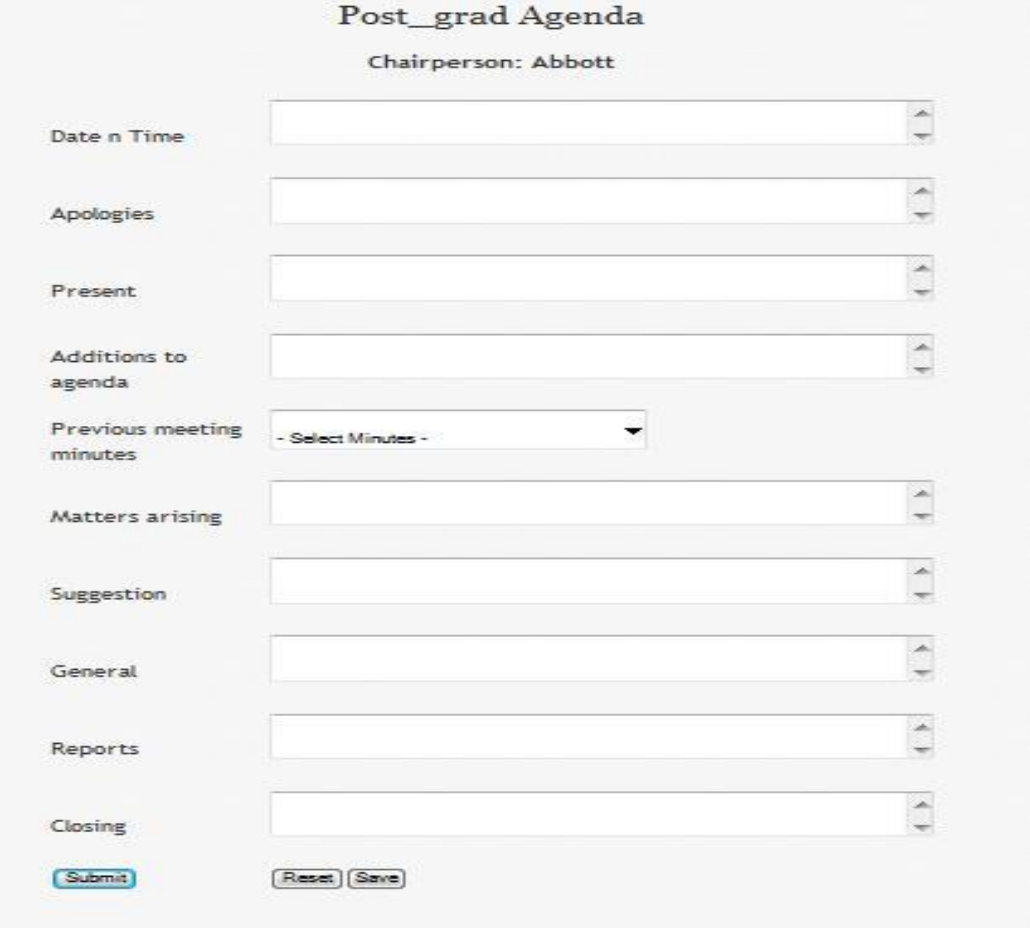

Figure 5: Agenda created

Figure 5 shows the new agenda that has been created. At this point the administrator can add minutes to the agenda if they are available or just save the agenda template for later use.

Creating meeting minutes

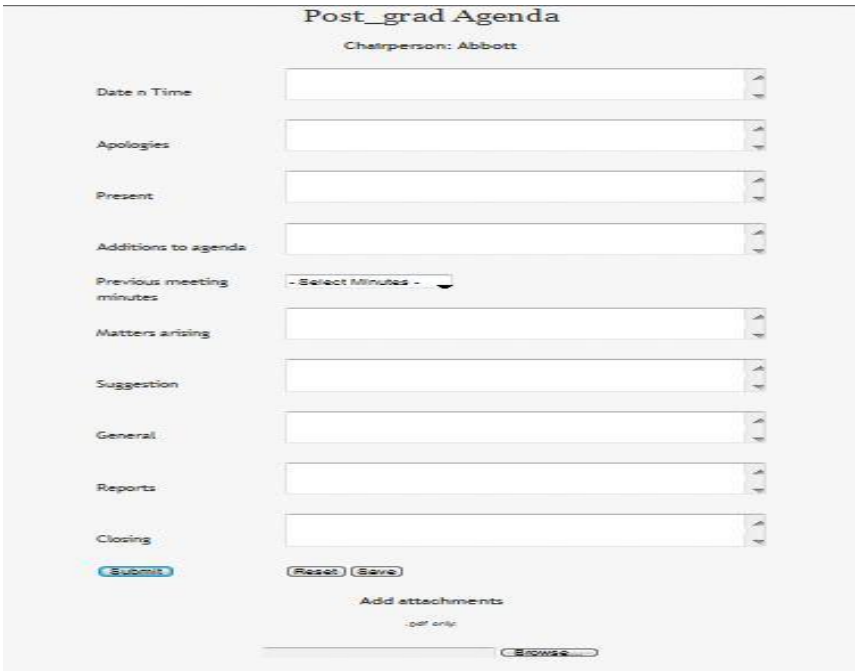

Figure 6: Agenda request

The administrator can record new minutes using the existing agenda that they had created which allows the administrator to attach pdf documents to the minutes (see Figure6). The adimistrator also has the option of adding the link to the previous minutes on the new minutes. The minutes are then store on the database as a link to the pdf file that is created and the attachment also stored together with the minutes on the server.

USER

Home page

Figure 7: Select committee

Figure 7 shows the menu of the user when logged in to the system. The user can view the latest minutes , register an agenda topic and also go to the archive page to view more of the previous minutes.

Minutes archive page

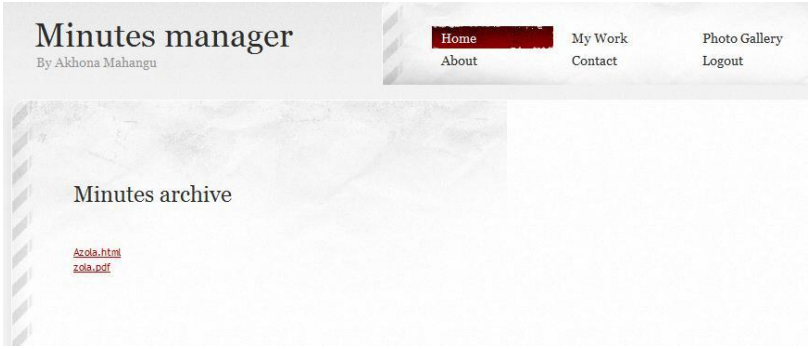

Figure 8: Administrator's home page

The minutes archive page will show a list of all the previous minutes stored on the system ordered by date of creation (see Figure 8).

How the user interface behaves

USER

Logging in	
Departmental Minutes	*Viewing previous minutes*
	Editing the agenda
	Adding documentation for the meeting
CLS Minutes	*Viewing previous minutes*
	Editing the agenda
	Adding documentation for the meeting
Applying for a meeting agenda	

Table 1: User interface specification

The interface allows the user to login with a unique username and password which must be authenticated before the user can be granted access to the system. In case of wrong input the user will be notified through the error message prompt for the correct login information. When login is successful, the home page will appear where the user can choose to view previous minutes, add items to the agenda for the next meeting, or request a new meeting agenda. The functions of the system are illustrated in the table above (Table 1).

ADMINISTRATOR

Logging in		
Committees	*Create new committee*	
	Edit committee	*Add/ delete members*
Agendas	*Create new agenda*	
	Edit/ delete agenda	*Attach items to agenda*
Add new user		

Table 2: Administrator interface interaction

Table 2 illustrates the main functions that can be performed by the administrator on the system. The administrator logs into the system with the use of a unique username and password. When successfully logged in the administrator can; view committees, create a new committee, edit committee or add/delete committee members; access agendas and create a new agenda or edit/delete an existing agenda and/or attach items to the agenda; or the administrator can add new user, creating a username and password for them.

How the users interacts with the interface

USER

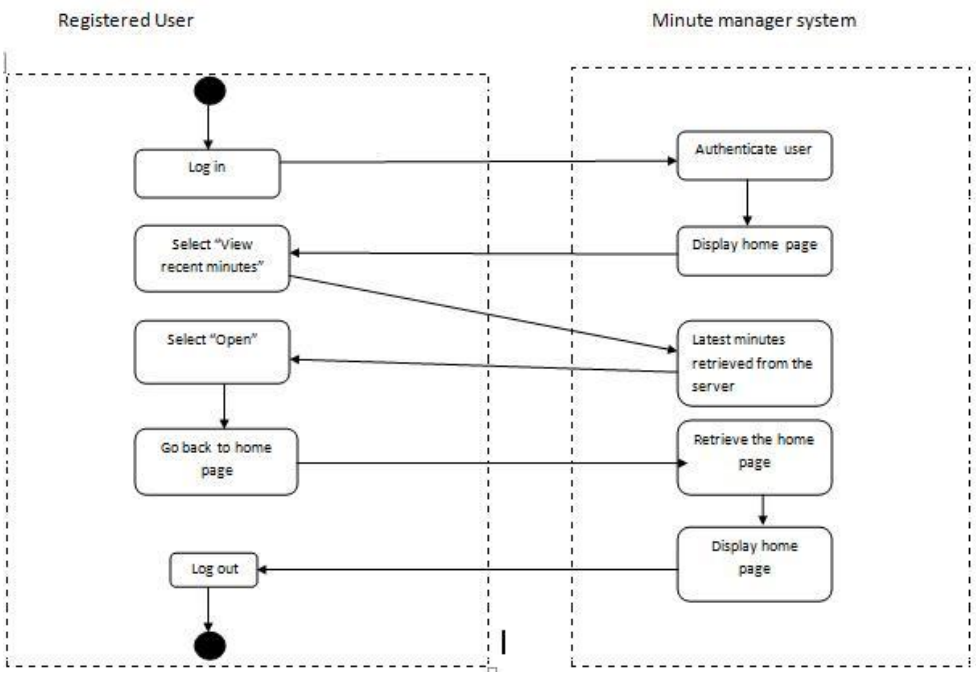

Figure 9: Activity diagram

Figure 9 shows the interaction between the system and the user through the interface. The interaction is as follows:

1. The user logs into the system

2. System authenticates user log information
3. Systen display home interface with the menu
4. User selects 'View recent minutes'
5. Sytsem retrieves latest minutes recorded saved on the server
6. When done user returns to home page
7. User logs out of the system

ADMIN

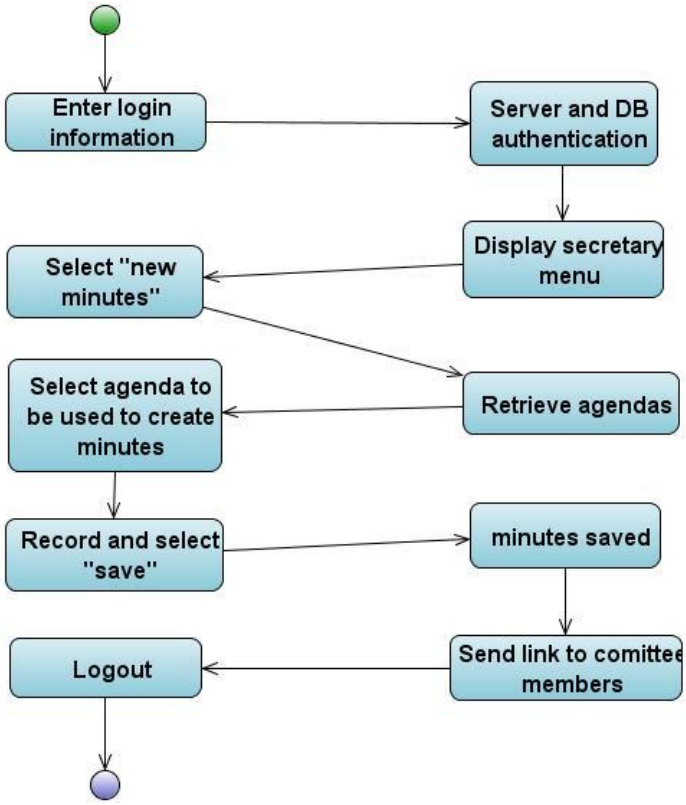

Figure 10: Activity diagram for the administrator

Figure 10 shows how the administrator/secretary interacts with the system. And it is as follows:

1. Administrator/secretary logs into the system
2. System authenticates user information
3. The system retrives user home/menu page and displays it
4. The administrator selects the option to create a new agenda
5. System retrieve existing agendas for recording of minutes
6. Administrator selects the agenda to be used
7. Records minutes and or attach documents and save the minutes
8. System saves minutes on the database
9. System send link to the minutes to all member of the committee via email
10. Administrator can now logout of the system.

USE CASE

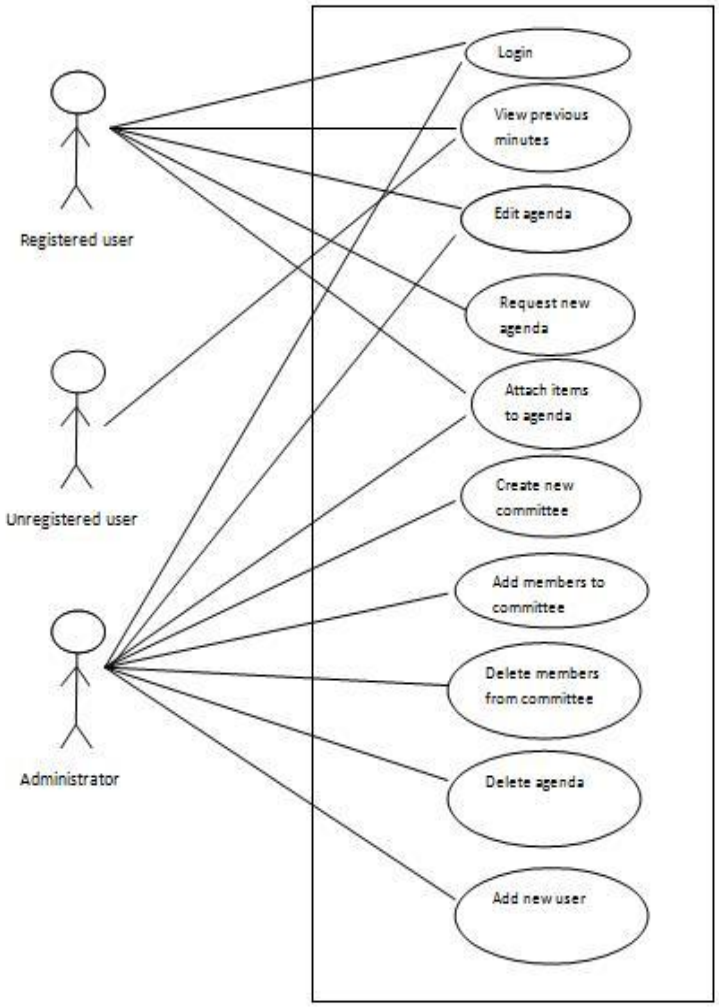

Figure 11: Use case diagram for all users' interaction with the system

The figure above illustrates the activicties peformed by the different users on the system. The different users can perform activities on the system based on the previledges they're granted when being registered on the system.

Summary

In this chapter the user interface was specified, how it will look like and also how it is expected to behave. This chapter also explained the interaction of the user with the system. In the next chapter the high level design will be specified.

Chapter 4

HIGH LEVEL DESIGN

Introduction

In the previous chapter we covered the user interface specification and how the user will interact with the system. In this chapter we will be looking at object oriented analysis which will describe the object-oriented view of the problem, where every object will be described and documented in the data dictionary. The relationship between the objects is shown while the class diagrams display the attributes and methods of each class.

Detailed breakdown of the technical solution

The minute manager system will consist of the following subsystems:

Interface object – this interface object will represent one main interface with links to represent each function. It will allow users to login, view previous minutes, view and edit agenda, allow the administrator to create a new agenda, add new minutes to the minutes archive, add a new committee and add/delete members of the committee. Users must link to particular pages be able to access them and their functionality.

Database Object – this object represent the database. Database contains tables that are used to store the previous minutes, agenda, committee and the users' details.

Web server Object – this object is responsible for storing data, display interfaces, connect/disconnect interfaces with databases, and contain databases. It is also support the interface object with different message functionalities, to make it possible for the administrator to send notification/reminder of task by email to the users and to allow the users to respond to the reminders.

Administrator object – this object contain the administrators information. It is responsible for uploading files, updating data, deleting data and to view database status.

User object – this object contains information about the users.

Class Diagram

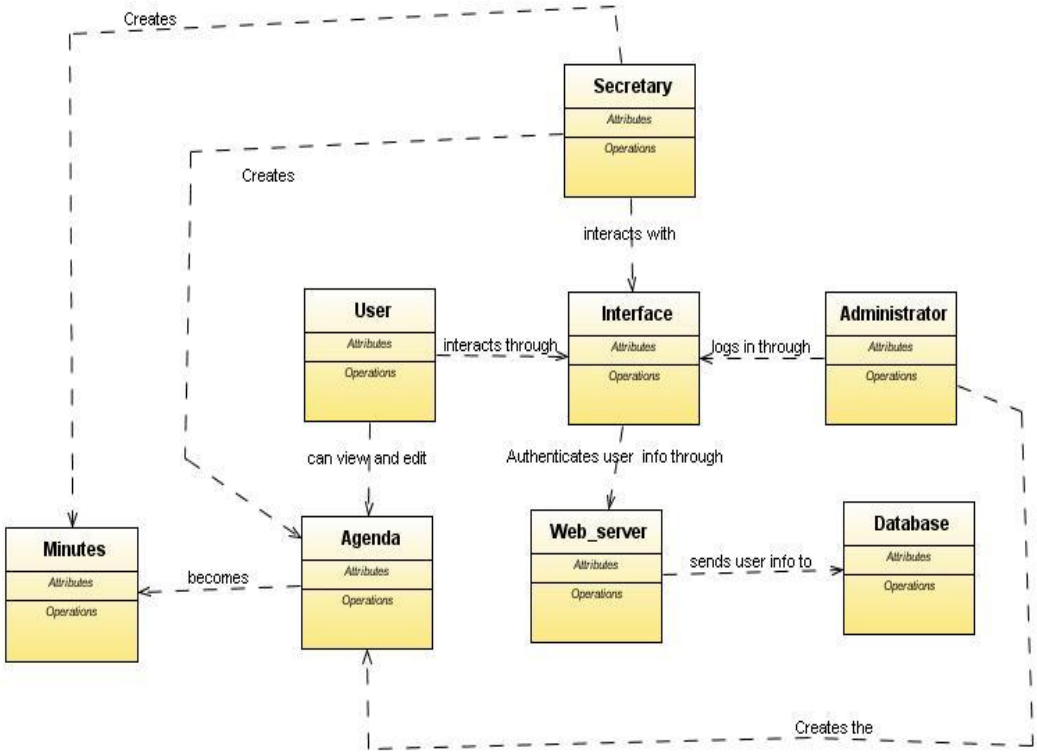

Figure 12: The class diagram

The class diagram shows the relationships between objects in the system (see Figure 11). The Administrator, the User and the Web server objects are the main objects in the class diagram. The purpose of each object is mentioned in the previous section.

Detailed interaction between subsystems

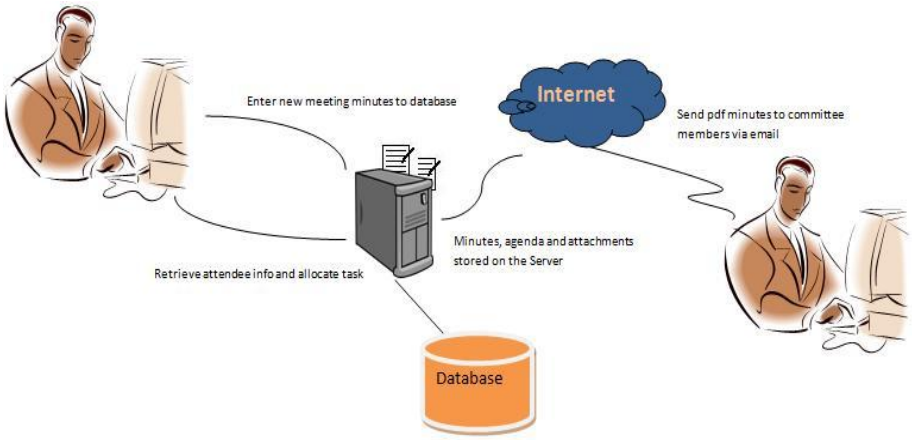

Figure 13: Interaction between subsystems

The user(admin or user) on one side logs into the system and their information is authenticated through the web server. The system shows the home page that satisfies the previledges of the user. When the user(admin) introduces new information to the system the new information is stored in the database. When new minutes are entered on the system some tasks allocated to user(s) are sent to the user through an email.

Summary

In this chapter the high level design was analysed. The various classes involved and how they are expected to interact with each other was described. In the next chapter the problem will be analysed further, and the system design will be analysed at a lower level.

Chapter 5

LOW LEVEL DESIGN

Introduction

In the previous chapter, the high level design was analysed and classes were used to analyse the solution showing their attributes and the relationship between them. In this chapter we will have a closer look at the classes, looking at the inner attributes and methods.

Inner details of class atttributes and methods

This section describes the inner details of each class attribute in terms of data types.

Class	**Attributes**
Administrator	**Username:** administrator number to identify the administrator. **Data type:** number **Example:** 2613609
	Password: administrator password. **Data type:** varchar **Example:** mazola
User	**Username:** user's username is needed for authentication. **Data type:** string **Example:** Myname
	Password: user's password needed when for authentication. **Data type:** characters and numbers **Example:** mypass55
Interface	**interfaceName:** are use to store the names of the interfaces needed when displaying interfaces. **Data type:** String **Example:** Login interface
	WebserverName: Used to store the server address needed when sending

	emails. **Data type:** numbers/ characters **Example:** 172.16.36.0/localhost
Databases	**String database_name:** are used to store the database name. **String table_name:** are used to store database table names.
Database_table	**Data type:** string **Example:** user_db **Varchar description:** are used to store the user information.

Table 3: Description of attributes for classes

Inner details of class methods

Class name	**Method**
Webserver	**Lookup (search_word):** This method connects and disconnects the interfaces to the server. It is used mostly when storing or retrieving data to and from the server
Database	**storedata():** This method stores the data to the database; minutes, agenda and attachments. It also stores the users' details
Interface	**displayInterfaces():** this method displays the different interfaces for different purposes.

Table 4: Description of the inner details of each method's presence in the classes

Psuedo code

Login function

The login function is for both users and administrators. Display login page: the user or the administrator enters username and password in textbox.

```
If (click on the login button) {
   Search the user's database for username and password
   If (details match) {
     Close the login page
     If (status = user) {
        Display the message page
       } else {
          Display the administrator page
          }
    } else {
         Clean username textbox and password textbox Display
         "User not found" Start the login procedure again
         }
}
If (click on the reset button) {
    Clean username textbox and password textbox Start to fill
    in the username textbox and password textbox
}
```

Create new agenda

The pseudocode for an administrator creating a new agenda is as as follows:

```
getFields(){

Enter name of agenda

Select agenda fields

}

createAgenda{

if selected field

show input tag for the field on the form

submit minutes button on the form

save agenda as template

}
```

Notification of task

If the user was allocated a task at the meeting then the administrator has to notify the user via an email of their tasks.

```
Notify()

{

    If (tasked_member = "user"){

            Subject = "task"

            Send email to "email_user"
```

```
Else
    {
        Connection to "user" was not established.
    }
```

Summary

This chapter featured pseudo-codes showing an outline of programs, written in a form that can easily be converted into real programming statements. It also featured sequence diagram showing the student interactions with the system as lifelines run down the page. The next chapter will present a detailed documentation of testing.

Chapter 6

CODE DOCUMENTATION

Introduction

In the previous chapter the data types for the class attributes, operations and algorithms were defined. The implementation details of classes were also defined together with the diagrams that explain the system impementation details. In this chapter the entire system is sketched viewing all the activities performed on and with the system. The source code of the programs is fully documented. For each PHP file, a MySql query is defined to ensure that information is retrieved from the database according to user requirements. The source code is commented to ensure that the code is understandable and modifiable.

Login script documentation

```
<?
/**
*****************************************************************************
*
*login2.php
*****************************************************************************
*
*Author: Akhona Mahangu
*Email: 2613609@uwc.ac.za
*****************************************************************************
*
This page(login2.php) helps the two users to login we can login as admin and
user.
*****************************************************************************
```

This code contains following pages:

connect.php : which helps to connect to the database

adminComp.html: represents the administrator page for the computer science department . In this code it is used to display the Admin Interface if the user that is logging in is registered as admin and under the computer science department.

adminStats.html: represents the administrator page for the computer science department . In this code it is used to display the Admin Interface if the user that is logging in is registered as admin and under the computer science department.

userComp.html: represents the user home page for the computer science department users.

userStats.html: represent the user home page for the stats department users.

****

In this page every query has its comment. The code also includes some other important comments.

**/

```
$query = "SELECT * FROM loggers WHERE

user_name ='".$_POST["user"]."' AND password='".$_POST["pass"]."'";

include("connect.php");

$result = mysql_query($query) or die("Unable to verify the user because :" .mysql_error());

$row = mysql_fetch_array($result);

if ($row) {

session_start();

$_SESSION["user_name"] = $row['user_name'];
```

```
// Check user type
$user = $row['type'];
$priv = $row['dept'];
if($user == "admin"){
if($priv == "computer") {
header("Location: adminComp.html");
}
else if($priv == "stats"){
header("Location: adminStats.html");
}else{
exit;
}
}else if($user == "user") {
header("Location: user_log.php");
exit;
}
else if($user == "guest") {
header("Location: booking.php");
exit;
}
// Kill the session
```

```
unset($_SESSION["user_name"]);

}

else {

//echo "attempt unsuccessful";

header("Location: trylog.html");

exit;

}

mysql_free_result($result);

mysql_close($conn);

?>
```

Agenda creation script documentation

```
<?php

/**

*****************************************************************************
*

*build_tabs.php

*****************************************************************************
*

*Author: Akhona Mahangu

*Email: 2613609@uwc.ac.za

*****************************************************************************
```

Description:

This page is used when agenda fields are selected and it is used to process the checkbox information from the page newCreate2.php.

This page is generates an agenda form based on submition of the checkbox information from the script mentioned above.

It later create a file with a file name given on the script newCreate.php and saves the form generated on this page on it.

The new form that is generated by this script calls another php script called agenda.php that is used on submit when the minutes are record.

This script also contains a form that allows the user to add custom fields

****

Caveats:

The modularity of the functionality can be improved

****
*

In this page every query has its comment. The code also includes some other important comments.

```
$con = mysql_connect("localhost","root","");
 if (!$con)
 {
   die('Could not connect: ' . mysql_error());
 }
mysql_select_db("minutes", $con);
$result = mysql_query("SELECT tabs FROM tab_fields");
//$pat=~'\^<HOST> -.*"(GET|POST).*\?.*\=http\:\/\/.* HTTP\/.*$\i';
$agenda = $_POST['temp_name'];
if(empty($agenda)){
```

```
Header("Location:newCreate2.php");
}
else{
//set background
echo '<body background="./images/bg01.jpg">';
//echo '<select name="minutes" onChange="top.location.href =
this.form.minutes.options[this.form.minutes.selectedIndex].value"><option
value="">-Tabs -</option>';
//$t==2;
//echo'ess';
//echo $t;
$agenda = $_POST['temp_name'];
//write the agenda name to a file
$test = "new.txt";
$open = fopen($test,'w');
$new = $_POST['temp_name']."\n";
fwrite($open,$new);
fclose($open);
//end here
$name = $_POST['temp_name'].".html";
//echo $agenda;
$File = "./new_agenda/$name";
$Handle = fopen($File, 'w');
//chmod($Handle, 0777);
$Data =  $agenda."\n";
fwrite($Handle, $Data);
$Data = '<body background="../images/bg01.jpg">';
fwrite($Handle, $Data);
echo '<form action="display_info.php" method="post">';
```

```
$Data = '<form action="../print.php" method="post">';
fwrite($Handle, $Data);
echo "<font align =center><center><h2>".$agenda." Agenda
</font></h2>";
$Data = "<font align =center><center><h2>".$agenda." Agenda
</font></h2>" ;
fwrite($Handle, $Data);
while($row = mysql_fetch_array($result))
{
$i==1;
$database = $row['tabs'];
//echo '$POST["check'+$i+'"] <br>';
if(isset($_POST['check'.$i]) && $_POST['check'.$i] == 'Yes')
{
   echo "<tr>";
   $Data = "<tr>" ;
   fwrite($Handle, $Data);
   echo '<td width="191">';
  $Data =  '<td width="191">';
   fwrite($Handle, $Data);
   echo $database;
   $Data =  $database;
   fwrite($Handle, $Data);
  echo '</td>';
$Data = "</td>";
fwrite($Handle, $Data);
echo'<td
width="286">       <textarea
name="';
```

```
$Data='<td
width="286">       <textarea
name="';
fwrite($Handle, $Data);
echo $database;
$Data =  $database;
fwrite($Handle, $Data);
echo '" cols="50" rows = 2></textarea><br><br></td>';
$Data = '" cols="50" rows = 2></textarea><br><br></td>';
fwrite($Handle, $Data);
echo "</tr>";
$Data = "</tr>";
}
 $i++;
}
$Data = "<br>" ;
fwrite($Handle, $Data);
$Data = '<input type="Submit" name="formSubmit" value="Submit" />' ;
fwrite($Handle, $Data);
$Data = '<input type="reset" name="formReset" value="Reset" /><br>
<br>' ;
fwrite($Handle, $Data);
$Data = "</form></center>";
fwrite($Handle, $Data);
}
?>
<br>
<input type="Submit" name="formSubmit" value="Save" />
<input type="reset" name="formReset" value="Reset" /><br><br>
```

```
$Data = '<h4>Add attachments</h4>';
fwrite($Handle,$Data);
$Data = '<p>.pdf only:</p><br>';
fwrite($Handle,$Data);
$Data = '<input type = "file" name = "datafile" size = "30">';
fwrite($Handle,$Data);
</form></center>
?>
```

Print.php script documentation

```
<?php
/**
*****************************************************************************
*print.php
*****************************************************************************
*Author: Akhona Mahangu
*Email: 2613609@uwc.ac.za
*****************************************************************************
Description:
This script is used to store meeting minutes to the minutes table on the database
It is called by the checbox.php script and it takes the form information from the
form generated by build_tabs.php and stores it on the databe
and also displays it to confirm the information captured.
*****************************************************************************
In this page every query has its comment. The code also includes some other
important comments.

$con = mysql_connect("localhost","root","");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
mysql_select_db("minutes", $con);
$result = mysql_query("SELECT tabs FROM tab_fields");
//set background
```

```
echo '<body background="./images/bg01.jpg">';
echo "<br><h2><font color = 'blue'>Your information has been captured as follows: </font></h2><br>";
echo '<h2>'.$theDatag.'Committee Minutes <br></h3>';
echo '<p>Created on '.date("d/m/y").'<br><br>';
echo"<font color = 'blue'>_____</font><br><br>";
$agenda = $_POST['temp_name'];
echo $agenda;
while($row = mysql_fetch_array($result))
{
$i==1;
$database = $row['tabs'];
//echo $database."<br>";
if(isset($_POST[$database]))
{
echo "<tr>";
echo '<td width="191">';
echo $database.":";
echo '</td>';
echo $_POST[$database];
echo '<br><br></td>';
echo "</tr>";
}
$i++;
}
echo "Attachment :".$_POST["datafile];
echo '<table><tr>';
echo'<form action = createPdf.php >';
```

```
echo '<td><input type="submit" name="formSubmit" value="Create Pdf" />';
echo '</form>';
echo'<form action = adminComp.html >';
echo '<td><input type="submit" name="formSubmit" value="Go home" />';
echo '</form>';
echo '</tr></table>';
?>
```

Display_info.php script documentation

```
<?php
/**
**********************************************************************
*display_info.php
**********************************************************************
*Author: Akhona Mahangu
*Email: 2613609@uwc.ac.za
**********************************************************************
```

Description:

This page is used when agenda is saved, it gives the user an option of opening it immediately or to go back to the home page.

```
**********************************************************************
```

Caveats:

None identified

```
**************************************************************************

$new = "new.txt";
$fh = fopen($new, 'r');
$agenda = fgets($fh);
fclose($fh);
echo "< xcenter>The agenda has been saved as ";
echo '<body background="./images/bg01.jpg">';
echo"<center><a href = './new_agenda/".$agenda.".html'>".$agenda.".html
</a><br><br><br>";
echo'<form action = adminComp.html >';
echo '<td><input type="submit" name="formSubmit" value="Go home"
/></td>';
echo '</form></center>';
//include("get_info.php");
//include("build_tabs.php");
?>
```

Summary

This chapter provided the code documentation for the processes performed to create an agenda and record minutes. The PHP scripts consisted of a name, Mysql queries and comments. The next chapter describes how the system will be tested to check whether the functionality is correct.

C h a p t e r 7

SYSTEM TESTING

Introduction

The previous chapter provided full code documentation for each process for the processes of agenda creation and minutes recording. In this chapter system testing will be carried out to verify and ensure that the system meets its design specifications and other requirements. This chapter specifies the strategies used for system testing, specifies the testers, and documents the test results.

Test plan

FUNCTIONALITY TESTING

System testing involved different strategies for evaluating the functionality of a piece of software:

- Unit testing: this strategy is used to test whether the individual units of the system are fit for use. For example testing the command buttons, checkbox etc.

- Integrated testing: this testing strategy combines individual software modules to be tested as a group. For example creating an agenda, recording minutes and adding members to a committee, modules are tested together to ensure that the system is fully functional.

- Black box testing: this technique uses valid and invalid input to test the output and system behaviour.

USABILITY TESTING

Usability testing measures the usability, or ease of use, of a specific object or set of objects and to perform this test the user were asked to evaluate the usability of the system by giving a rating between 1 and 5, with 1 being ineffecient and 5 being very effecient.

While measuring functionality and usability, the system was tested on different users. The tests helped to discover errors and areas that required improvement. The Participants details are recorded on the table below (See Table 5).

Name (Participant)	Department	Modules Tested	Time Taken
Fatima	Computer Sc.	Create agenda, record minutes	10 mins
Leslie	Stastics	Create agenda, record minutes	15 mins
Lydia	Health Sc.	Log, view minutes	<5 mins

Table 5: Participant's details

Test results and discussion

In order to obtain the feedback from users, a participation consent form had to be issued to the user, a scenario was drawn to facilitate and guide the perfomance of some tasks on the system, and some evaluation questions were drawn to evaluate the functionality and usability of the system. This document will show an anlysis of the user(refering to all users) test feedback.

Functionality testing

Functionality testing involved testing interface pages, links, buttons and the database. During this test, users tested the core functions of the system (generating an agenda template and recording minutes). Two-thirds of the users said that the syatem functions are easy toperform and 1 in 3 found the system functionality very easy to follow (see Figure 14). All the users agreed that they would love to use a system like this one.

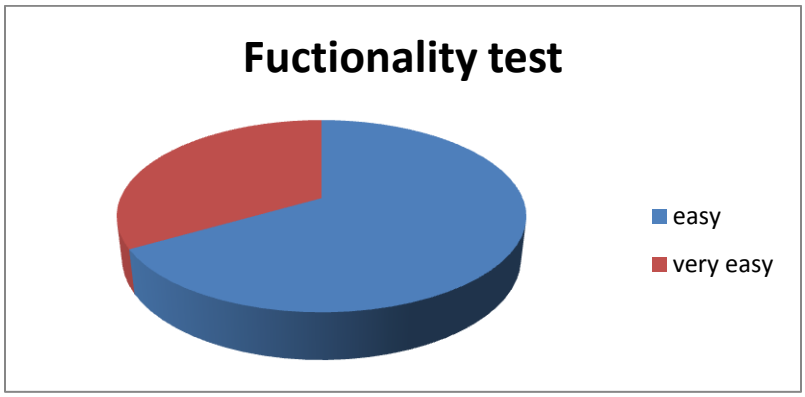

Figure 14: functionality

To test the functaionality of the system the users had to follow a scenario they had been given that comprised of the following tasks for the administrator:

1. Login as administrator
2. Choose desired action page
3. Select required fields for the new agenda
4. Create agenda and attach documents
5. Save agenda

All that the general user had to test was logging in to the system and viewing the minutes archive. The graph above indicates the final outcomes of all the aspects tested by the user in terms of the functionality of the system which included:

- The ease of navigation (use of links, buttons, etc)
- Successful execution of tasks

Usability

Usability testing focused mainly to the general appearance and the user friendliness of the system. According to results obtained, users had the same judgement to the interfaces (administrator and general user). Hence the information obtained represents the general appearance and user friendliness of the whole system interface.

General appearance

Users showed positive attitude to the appearance and design of the interface. On the first test one of the test participants 1 in three users suggested that some aspects the system be changed, like the visibility of the "back" button. After that was changed they were satisfied with the change.

User-friendliness

All users reported that the system was easy to understand and use. It was much-admired for its simplicity and the fact that no many mouse clicks were required to complete an entire process. Links worked accurately making it easy to navigate among pages. The following bar-graph (see Figure 15) represents users‟ responses to user friendliness of the system.

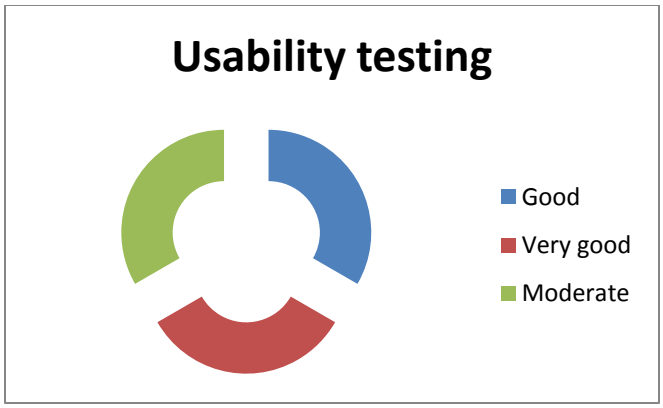

Figure 15: Usabilty

Recommendations

During system testing users made significant comments and recommendations about possible ways of improving this system. Some corrections were made to rectify errors. Because of time constraints some functionalities were not successfully implemented on the system and therefore will be recommended for future work. These functionalities include:

- Applications to committee with the use of SR forms
- Converting the meeting minutes to a pdf file
- Sending minutes as an email attachment to committee members

Summary

This chapter discussed the techniques used to test the system and results obtained. The following chapter provides a detailed user guide documentation that will help the users to use the system.

C h a p t e r 8

USER GUIDE

Introduction

The previous chapter discussed the system testing process, highlighting the strategies used, the test results acquired and their discussion, and also listed some recommendation for improvement from the testers. This chapter provides a detailed system user guide that is aimed at helping the users to understand how the system works. Because this system is a desktop application the user guide is compiled for computer users only. The User Guide include: administrator user guide and the regular user(committee member) guide in which committee and administrator are provided with the information relevant to their respective tasks. This information points out steps undertaken in performing particular task.

Administrator user guide

This user guide is designed for administrators/secretary to provide them with information so that they can carryout the agenda formulation and minutes recording process effectively. The user guide comprise of the following components: ways of navigating around the system and how to perform tasks.

Login

The first step for the user to be able to use the system is to get authentication through the login page. The login page requires the user to enter a username and password (see Figure 16).

- Enter username and password
- Click on "Submit" button

- If login information is incorrect then access will be denied

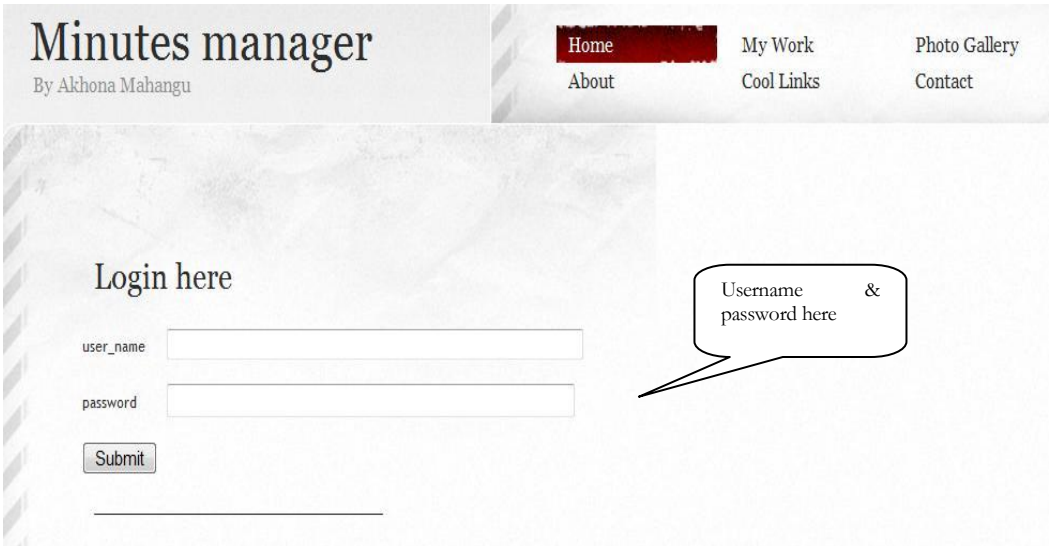

Figure 16 Login

Create Agenda

The administrator selects agenda -> create new from the menu page (see Figure 17).

Figure 17: Menu

The next step is to select fields that will be part of the agenda(see Figure 18).

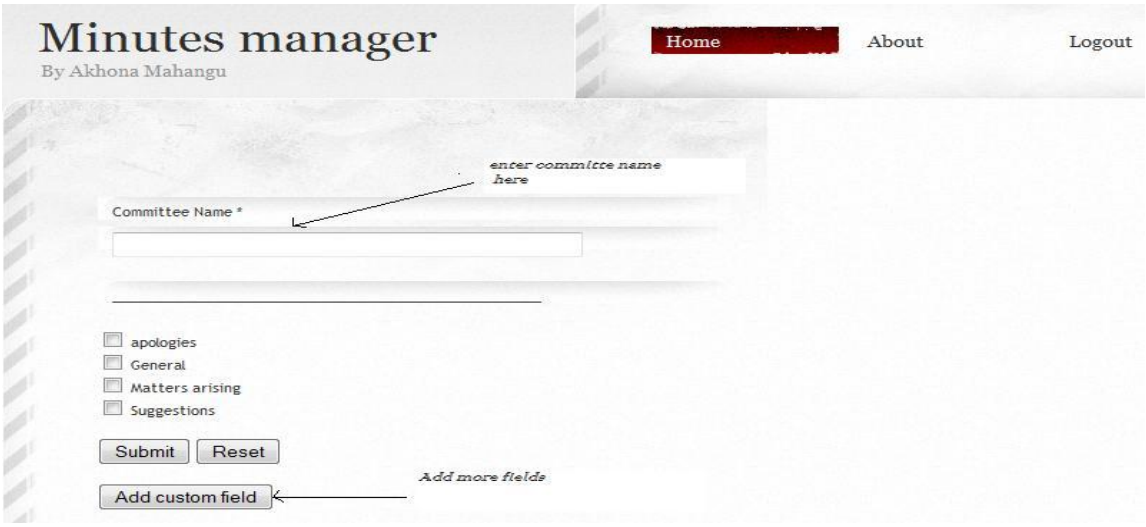

Figure 18: Agenda fields

The administrator can use the "Add custom field" button to add more fields to the list. When the button is clicked the administrator can then add the field they want (see Figure 19).

Figure 19: Add more fields

Save agenda

When agenda has been created it can then be saved (see Figure 20).

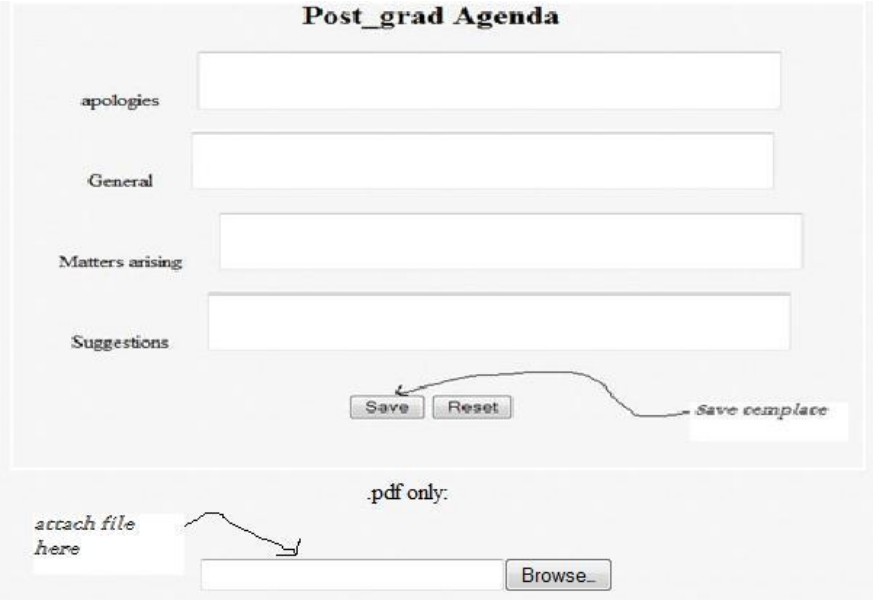

Figure 20 : Saving agenda template

When the agenda template has been created it is then saved and the adminstrator will then have an option of opening it then to record minutes or go back to the home page (see Figure 21).

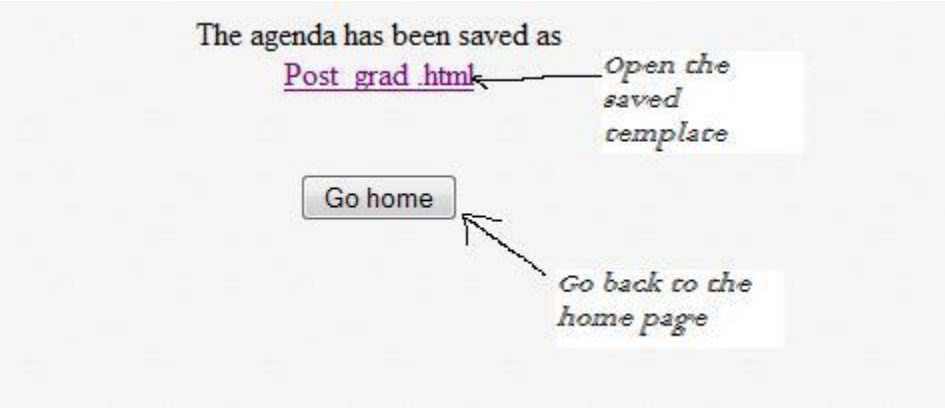

Figure 21: Navigation page

Record minutes

To record minutes the administrator has to select "new minutes" from the administrator menu (see Figure 22), then select the agenda that will be used for recording the minutes.

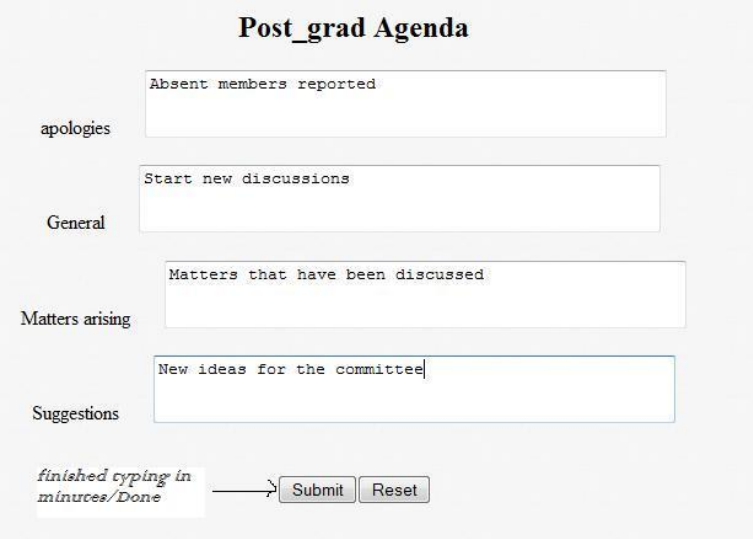

Figure 22: Recording meeting minutes

Handling of saved minutes

When the minutes have been successfully created then select "create pdf"to create a pdf version of the minutes (see Figure 23).

Figure 23: handling saved minutes

General(committee member) user guide

This user guide is designed for regular user (committee member) to provide them with information so that they can carryout their processes effectively.

Login

The first step for the user to be able to use the system is to get authentication through the login page. The login page requires the user to enter a username and password (see Figure 24).

- Enter username and password
- Click on "Submit" button
- If login information is incorrect then access will be denied

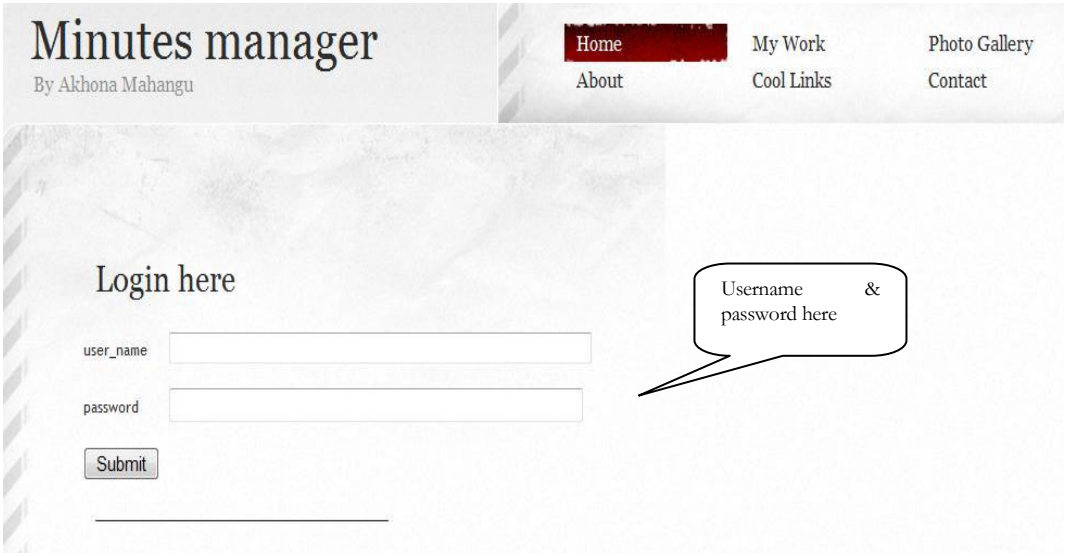

Figure 24: User_Login

Menu

The user can click on any option on the menu to navigate to the next desired page (see Figure 25).

Figure 25: User menu

Summary

This chapter discussed the user guide documentation. The following chapter provides a conclusions about the system and the research carried out through this project.

Chapter 9

CONCLUSION

This thesis aimed at documenting the Minutes Manager System project, which was proposed and carried out with a view to address problems experienced when managing meeting minutes the the old school way (by keeping big printout books of the minutes).

Generally, the chapters in this thesis discuss the basic processes followed to achieve the completion of this project, and these are requirements gathering, analysis, design, coding, testing and maintenance. All the objectives that have not been met while implementing the system for the purpose of this project are considered for future work.

To end off this thesis a big thank you goes to Prof. Venter for investing her exceptional skills on this project and also to everyone who helped with planning, implemrntation and testing of this project.

APPENDICES

APPENDIX A

Probes

Project title: Minute Manager

Description

This project is about developing software that record and manages minutes of a meeting.

What must the application do?

The application will allow an individual to record minutes for each meeting; act as a reminder to tasks allocated to the individuals concerned. For each task that needs to be carried out before the next meeting the system will sent a notification to the person who is allocated the task and send reminders at allocated times before the next meeting. The system should allow all the users' concerned to view the agenda for the next meeting and make changes to it when necessary.

The following questions/probes will be used to interview people that deal with the compilation and management of agendas and minutes.

1. What problems do you have with the existing system?
2. Do you think a system that can automate some of your duties would be helpful to your minute recording and management?
3. If yes, how? And if not, why not?
4. What functionalities would you regard as important/ would like the system to have?

Users to be interviewed

- Leslie Selbourne(Secretary Stats department)
- Ms Abbott(Secretary Computer Science department)
- Mrs. Connan(Computer Science)
- Mr. Leendarts(Daniel)

APPENDIX B

Interviews

- Mr. L Selbourne
    - Problem(s) with the existing Paper-based system
        - People don't always take their duties seriously or seem to forget tasks that are allocated to them.
        - Only find out at next meeting if individuals haven't carried out their tasks.
        - People don't always read their emails or even forget about their tasks.
    - What functionality would you like the system to have?
        - Simple design, clear and user-friendly
        - All individuals concerned have access to the agenda/minutes before and after the meeting so they can make additions to the agenda if they wish to do so. Provided that the system can keep track of who made which changes and that the additions are authorized by the HOD.
    - What do you like about the proposed system?
        - Minute management becomes less time consuming because the workload is shared, since everything is done on the net.
- Ms R Abbott
    - Problem(s) with the existing Paper-based system
        - People don't always take their duties seriously or seem to forget tasks that are allocated to them.

        - Only find out at next meeting if individuals haven't carried out their tasks.
        - Too much paper work goes into managing minutes.
        - People don't always read their emails or even forget about their tasks.
    - What functionality would you like the system to have?
        - Simple design, clear and user-friendly
        - All individuals concerned have access to the agenda/minutes before and after the meeting so they can make additions to the agenda if they wish to do so. Provided that the system can keep track of who made which changes and that the additions are authorized by the HOD/ Chairperson of the meeting.
        - All stakeholders should be able to view and confirm minutes before the next meeting.
        - Stakeholders should be able to attach documents on the agenda.
    - What do you like about the proposed system?
        - Minute management becomes less time consuming because the workload is shared, since everything is done on the net.
        - That the stakeholders are notified of their duties and constantly reminded.
        - System would be the best solution.

- Mrs. V Connan
    - Problem(s) with the existing Paper-based system

        - Too much paper work goes is produced
        - Page numbers of the compiled meeting minutes and its attachments have to be hand written, because there's no program to do the compilation.
        - People don't always read their emails or even forget about their tasks.
    - What functionality would you like the system to have?
        - Error detection program for the forms that have to be added to the agenda.
        - Compile page numbers for different file types into one agenda/minutes file.
        - Categorize information and tasks by department
        - Keep a constant database of the minutes and attachments.
    - What do you like about the proposed system?
        - Minute management becomes less time consuming because the workload is shared, since everything is done on the net.

- Mr. Leendarts – after having mentioned that he has to knowledge of what goes into minutes taking and management in the department, this is what Daniel had to say in response to some of the probes that were posed to him:
    - What do you think of the proposed system?
        - It would make life easier for everyone involved
    - What would you like the system to have?
        - It should be easy to use
        - Secure and restrict access to foreign users.

APPENDIX C

Term 1 planning

Meeting dates & times /Tasks	Wed 3rd Feb 10h00	Wed 10th Feb 12h00	Wed 17th Feb 10h00	Wed 24th Feb 10h00	Wed 3rd March 10h00	Wed 10thMarch 10h00	Wed 17h March 10h00	Mon 23rd March	Wed 24th 10h00
Comments	**Identify users to interview**			*Combined meeting (IM Venter & WD Tucker) Identify someone (Writing Centre) to read your doc.*	*Separate meetings*	*Combined*	*Separate meetings*	*Combined*	**Presentation**
Thesis Document	Create document using Honours Project Guidelines from the website as well as Thesis doc from Word	See previous week! Complete the thesis outline (using Honours Project Guidelines)	See that you understand how to use the Styles and how to compile an Index, the Table of Contents, List of Figures etc.	and start with RAD.	o Check Index and add indexes o Bibliography – at least 5 entries.	Finalise write-up. Let someone proof read your document!		Hand in final document on Tuesday (23rd March) before 12h00	
URD	*Fill in headings* *Look at the questionnaire*	Start write-up of URD.	Continue with URD write-up. Interview stakeholders.	Complete URD					
RAD	n/a	Fill in headings		Start write-up of RAD.	Write-up RAD	Complete RAD			
Literature Survey	Familiarising yourself with your topic, and how it's implemented. Read and explore. literature on your topic.	Read and explore. some more – use Google Scholar..	Add all literature found to your bibliography – use the Harvard Notation	Keep on reading					
Presentation/deliverable	Write one paragraph that describes what you want to do, and why you want to do it This will be used as the abstract					Use thesis to Prepare slides for mock presentation	Mock presentation		24th March Present
Website	Ask Eoeslaar intelligent.networking@gmail.com	See previous week! Create website	Put plan onto website		Ask Eoeslaar about the server. Add URD to website	Add RAD to website			Put presentation on website

APPENDIX D

Term 2 planning

Tasks	**24th March**	**31st March**	**7th April**	**14th April**	**21st April**	**28th April**	**5th May**	**12th May**	**19th May**
Comments		*Combined meeting*			*Combined meeting*		*Combined meeting*		*Combined meeting*
Thesis Document	Presentation	Create 3 new chapters with subheadings for UIS, OOA and OOD – see p3	Complete editing. Start with the write up of the UIS	Start write up of the OOA & complete the UIS write-up	Complete write-up of OOA.	Complete write up of OOD	Make appoint-ment with writing Centrre	Complete write up. **15th May**: Hand in completed document to supervisor	
OOA		Read through the docu-mentation of this carefully and see that you understand it!		Start with analysing the RAD to create OOA	Complete OOA				
OOD					Start write up of OOD	Complete OOD			
UIS			Start with User Interface Specification	Complete UIS			Update changes to UIS		If needed - update changes to UIS
GUI & prototype				Start with the planning of the prototype	Program GUI/ prototype	Program GUI/ prototype	Program GUI/ prototype	Complete GUI/ prototype	Finalise GUI/ prototype
Other		Look at previous projects	Look at previous projects						
Presentation								*Prepare slides for mock presentation – 12th May – **for the 19h May***	Present on the 19th!
Web -site		Update web site					Check and update		**Put new plan, thesis & presen-tation on web site**

APPENDIX E

Term 3 planning

Planning Term 3 (2010)

Tasks	12th July	19th July	26th July	2nd Aug	9th Aug	16th Aug	23rd Aug	30th August – 5th Sept	6th Sept
Thesis Document	Finalise the editing of the documentation - & editing	Finalise the editing of the documentation - & editing Update any changes to the design – e.g. objects	Make changes to object's pseudo code as you develop the software, document all changes etc. in the code & **start** on the User's guide (User's Guide a deliverable for the next term only!)					**Finalise Documentation and hand in on the 27th**	
Re-visit the GUI and make changes or redesign	Check the GUI and see if you are happy that it deals with all the options	Re-design parts of the GUI or the whole GUI					Replace screenshots with screenshots of the current program (it will have changed)	Finalise GUI	
Create & populate database		Create & populate (add a few data references to) the MySQL database or put together files to be used in programme							
Programming task		Plan the approach by breaking task into objects or modules to program	Program 1st task/module /object	Program 2nd task/module /object	Programming	Program 4th task/module /object	Finalise programming & testing		
Testing and refining with a basic data set	Read about MySQL database and decide on its structure or if you use files how it will be used.	Read about software & tools you wish to implement	Read about software & tools you wish to implement		Decide on a subset of testing data	Testing and refining			
Presentation								*Prepare for presentation*	*Presentation 9th September*
Website	**Update NB**	**Update NB**					**Update NB**	**Update NB**	

Holiday Complete Still needs to be done

APPENDIX F

Term 4 planning

Tasks for the week of the	20th Sept	27st Sept	4th October	11th October	18th Oct	25th Oct	1st Nov	8th Nov Presentation on the 10th
Thesis Document	Finalise the editing of the documentation	Complete all editing of 3rd term's documentation.	Start by identifying all the tasks that the program must be able to do - write the User's guide			Finalise user's guide & Thesis documentation. - Ask colleague or someone to edit/proofread it!!		Hand in documentation on the 2nd – November
Programming tasks not completed	Revise programme & add what is still outstanding And make changes to thesis doc to reflect this!					Revise programme	Revise programme & create installation disc.	Finalise Installation disc.
Design Test suites	Read up about evaluation of program – get ideas of how you want to do evaluation. - Add to documentation. - Edit the chapters where you referred to it initially.	Add the theoretical part of evaluation to your documentation. - Edit the chapters where you referred to it initially.	Design test suites Choose your "users" Questionnaires etc. Ethical aspects					
Execute tests, revise code or even project design					Execute tests & keep record of it by writing it up in Thesis doc. - Create graphs that can be included in Thesis doc.			
Presentation								*Prepare for presentation and install on relevant computers. - Test if it works!!*
Website	**Update NB**	**Update NB**				**Final Update NB**		

APPENDIX G

Participant Consent Form

Project topic: Minute manager system
Name of student: Akhona Mahangu

Name of participant: _____
Department: _____

I (Participant) agree to participate in this research on the basis that:

1. This agreement is of my own free will
2. I have had the opportunity to ask any questions about the study
3. I realise that I may withdraw from the study at any time, without giving a reason and without any effect on my education
4. I have been given full information regarding the aims of the research and have been given information with the Researcher's names on and a contact number and address if I require further information.
5. All personal information provided by myself will remain confidential and no information that identifies me will be made publically available.

Signed: ..... Date: .....
(*Participant*)

Signed: ..... Date: .....
(*Student*)

APPENDIX H

Test results table

Functionality testing

Rating ► / Task ▼	(1) Very easy	(2) Easy	(3) Moderate	(4) Difficult	(5) Very difficult

Usability Testing

Rating ► / Task ▼	(1) Very good	(2) good	(3) Moderate	(4) Bad	(5) Very bad

Bibliography

Abbott, R. (2010, 02). Minute Management. (A. Mahangu, Interviewer)
Chiu, P. (2001). *LiteMinutes: An Internet-Based System.* Hong Kong: WWW10.
Leendarts, D. (2010, February). Minutes manager. (A. Mahangu, Interviewer)
Orenstein, D. (2000). *Quick Study: Application Programming Interface* .
Selbourne, L. (2010, 02). Minute Management. (A. Mahangu, Interviewer)
Sharp H, P. J. (2007). *Interaction design: Beyond human-computer interaction.*

INDEX

A

Administrator · 8
Apache · vi
API · vi

C

CLI · vi

G

GUI · vi

I

Interface · 21

M

Meeting minutes · 8
Minute management · 1

P

PHP MyAdmin · vii
Prototype · 7

S

Stress testing · 7

U

UIS · vii
UML · *See* Glossary