User-friendly DHCP and DNS configuration GUI for Linux

Abstract

This project is for an SME (Small Medium Enterprise) called ACME, which was founded in Scotland, and currently operates four offices in Glasgow, Aberdeen, Dundee and Edinburgh. This project develops a user-friendly Graphical User Interface (GUI) to configure Domain Name Servers (DNS) and Dynamic Host Configuration Protocol (DHCP) servers in the Linux operating system using C++ language in QT framework.

Table of Contents

List of Figures

List of Tables

1.0. Introduction

Nowadays, information technology has become an integral part of all businesses and plays a major role in their existence and progress. Company websites are highly important in introducing a company's products and services to the general public. Emails are used as a medium to engage a company's employees and also to answer any external query. As the company expands and customer base grows, more people are hired by the company to run its business. There is also a demand for the expansion of their internal IT infrastructure. Any IT infrastructure expansion needs installation and maintenance of E-mail servers, web servers and print servers, hence, IP-addresses allocation (either manually or automated such as DHCP) is one important for these servers.

There are different types of operating systems supporting different servers. Commercial operating systems such as Windows and OS-X are expensive, and they incur a high license fee for their deployment. However, their installation process is made easier by providing a user-friendly graphical user interface (GUI). Hence, the network administrator can easily install and maintain the network of systems and servers using a Graphical User Interface (GUI). For open source operating systems, servers and workstations are usually configured using command line interface (CLI). As a result of a large number of configuration commands, it becomes difficult, especially for support staffs to install or troubleshoot servers using CLI. However, if a GUI is provided or designed to facilitate easy configuration of open source servers, it will help even an inexperienced staff to maintain the network.

This project develops a GUI application to configure open source server's settings such as Dynamic Host Configuration Protocol (DHCP) and Domain Name Servers (DNS) for a Small Medium Enterprise (SME) called ACME Ltd. Moreover, this project critically discusses different types of operating systems, e.g. Linux and Windows, Programming Languages like C (Cprogramming.com, 2013), C++ (Cprogramming.com, 2013), Fortran (Chem.ox.ac.uk, 2015), ADA, Pascal, COBOL and Lisp, and Application Development Framework like Visual Studio and QT in a process to decide the best choice of Language and framework to develop the application. From the critical evaluation, it was derived that Linux will be used as an operating system because it is free and more difficult to configure system settings for DNS, DHCP and so on. C++ is chosen because it has less verbose code structure, no runtime overhead as well as being more secure and closest to machine language amongst other high-level languages. QT application development framework for C++ was chosen because it is open source and it also has a vibrant community of users. Furthermore, the project discusses additional functionalities and securities which can be embedded within the system to enhance the efficiency and usability of the application.

1.1. Background or Case Study

This project is for an SME (Small Medium Enterprise) called ACME, which was founded in Scotland and currently runs four offices in Glasgow, Aberdeen, Dundee and Edinburgh. The company produces furniture for living rooms to the general public, and its branch offices also act as sales and support centres. The company has identified the importance of IT (Information Technology) in their business as a driving force to enhance sales. As a result, the company have installed Domain Name Servers (DNS) and Dynamic Host Configuration Protocol (DHCP) servers to enhance its company operations. Users who are not technology savvy have found the requirement of typing commands on terminals difficult. The company is requesting for the design and development of a user-friendly Graphical User Interface (GUI) to configure or start important service on the servers. However, they are not willing to spend much money on licenced software and other software development and implementation costs.

1.2. Report structure

The report structure is as follows:

- Section 2 contains the Aim and Objectives of the project
- Section 3 includes an options analysis which can also be called background study or literature review.
- Section 4 is the methodology section which discusses the approaches used for this project
- Section 5 is the implementation section of the application
- Section 6 is the testing section of the application
- Section 7 focuses on the issues and the solutions of the implementation and testing section
- Section 8 is the conclusion and future work

2.0. Aim and Objectives

Aim: The project aims at designing and implementing a user-friendly common GUI interface to configure DNS and DHCP servers.

Objectives:

- **Objective 1:** To design a user-friendly and easy to use GUI for company server administrators
- **Objective 2:** Identify a cost-effective operating system
- **Objective 3:** Identify a suitable language and application development framework to develop the system
- **Objective 4:** To develop, test and maintain the system

3.0. Option Analysis

3.1. Research on Operating Systems

An operating system is considered to be the most popular software that runs on the computer. The functionalities of an operating system are to process and manage computer memory, software and hardware (GCFLearnFree.org, 2015). Also, it serves as the medium of communication between the user and the computer; the user can use the computer without knowing computer language (low-level language) (GCFLearnFree.org, 2015). Currently, there are several operating systems available on the market. However, Windows, Linux and Mac are the leading operating systems in the market.

3.1.1. Linux Operating System

Linux operating system is considered as high quality and easy to use operating system (Thomas and Sicam. 2008, p3). It is a free software that has free licensing. It can also be used to compare other operating systems like Microsoft Windows and Apple Mac. There are different types of Linux operating systems, some of which are, Ubuntu (Ubuntu, 2014), SUSE (Opensuse, 2014), FEDORA (Fedora, 2014), Red-Hat (Red-Hat, 2014) and so on. Hence, Ubuntu and OpenSUSE are considered appropriate for the development of GUI application because they support C++ compilers. However, Linux OpenSUSE was selected because its SUSE server has recently improved with more advanced features supporting C++ development. Furthermore, it also has pre-installed DNS and DHCP servers. Therefore, Linux OpenSUSE is selected for the development for ACME Company.

3.1.1.1. Linux OpenSUSE

"The openSUSE is a PC operating system based on GNU and Linux" (Opensuse-guide.org, 2015). It is licence free and an alternative to Microsoft Windows and Mac; it possess several advantages. OpenSUSE can be used as an operating system in laptops, netbooks, desktops, and several centre personal computers (Opensuse-guide.org, 2015). OpenSUSE is the leading and oldest Linux distribution kernel. The remarkable aspect is that OpenSUSE has a new version released every eight months, which supports new languages as well as security updates for 18 months (Opensuse-guide.org, 2015).

The most important advantages of using LINUX OpenSUSE are (Opensuse-guide.org, 2015):

- **Security:** There are no issues about viruses and spyware.
- **Stability:** Linux OpenSUSE is considered as stable because its operating system rarely crashes. Only individual application might crash more frequently which does not affect the operating system.
- **Maintenance:** There is no need for scanning for viruses and spyware, frequent rebooting, cleaning registry database and defragmenting of disks.
- **Open standards:** “GNU/Linux openSUSE and its applications generally support open standards, making it possible for seamless interoperability with other platforms, helping to avoid vendor lock-in”.
- **Community:** OpenSUSE is described as a “world-wide team spot” because many community volunteers use Linux developers to develop Enterprise edition of Linux OpenSUSE.
- **Open Source:** There is no strict licence agreement. The User is free to do any modification in the source code, and can also share the modification.
- **Legality:** The free and loose licensing will reduce the malicious and unlawful use of software (piracy)
- **Economy:** Linux OpenSUSE does not require a hardware upgrade to take place frequently, and as such, reduces the cost spent on hardware.
- **Transparency:** The development of the operating system is done openly. A public mailing list does the communication, and it also has a public bug tracker.
- **Diversity:** There is a wide range of distributions available from different vendors for diverse purposes.
- **Trying something new:** Many people are inspired to try something new. OpenSUSE is a new concept with new features which attracts many people to use it.
- **Privacy:** OpenSUSE maintains protection for a user’s personal information and files.

Table 1 contains the list of things that should be considered before the installation of OpenSUSE.

Table 1: Pre-installation requirement for Linux OpenSUSE (Opensuse-guide.org, 2015).

System Minimum Requirements	CPU: Pentium III 500 MHz or advanced processor RAM: 1 GB physical RAM (2 GB recommended)

	Disk Space: 5,0 GB for a typical installation (more recommended) Sound and Graphics Card: Most modern cards are supported
Burn the ISOs in an external device	DVD USB stick
BIOS Setup	If the computer does not boot from the DVD or USB media, check that the computer BIOS is configured to boot from CD/DVD or USB.

3.2. Research on Programming Languages

Linux operating system environment supports several programming languages, some of which includes: C (Kernighan and Ritchie, 1988), C++ (Stroustrup, 1997), Fortran (Kremer and Rame, 1993), ADA (Ichbiah, 1991), Pascal (Holmes, 1990), COBOL (Shelly, Cashman and Foreman, 2000), Lisp (Sangal, 1991) amongst others. However, C++ was chosen because of its tendency to produce less verbose code, less or no runtime overhead, and more security than other languages. Moreover, Linux comprises of a built-in C++ compiler, it also includes all the mandatory editors and tools freely obtainable for installation, and they are pre-installed with it.

3.3. Research on Application Development Framework

Currently, there are several application development frameworks available for C++ programming language, which include visual studio IDE, QT and so on. Visual studio supports C++ language (Sysprogs, 2014) for designing, developing and implementing Linux applications. Novak (2011, p23) stated that visual studio is intuitive and has many tools that help to develop GUI. However, it also has some licensing issues as it is not freely obtainable. On the other hand, QT is considered as a UI (User Interface) framework and all the supporting tools are open source projects (Qt-project.org, 2015). Blanchette and Summerfield (2008) point to QT C++ as a framework application to develop and create a GUI application.

This framework is widely developed and deployed in different desktop as well as embedded applications for different hardware (Deepthi & Sankaraiah, 2011) (Gois & Batagelo, 2012). Blanchette and Summerfield (2008, p77-95) demonstrated ways to implement application functionality using QT C++ in Linux. Based on the analysis of these frameworks, QT C++ was chosen and will be used for the design of the application GUI. This will run on an Open SUSE server.

3.4. Research on GUI

Windows GUI has a unique look and feel. For example, the font used in window's GUI is Sogeo UI (User Interface) font (Microsoft.com, 2015). Moreover, in many cases, where a UI component displays a particular functionality, it is also accompanied by an icon. An example of such is seen in Figure 1.

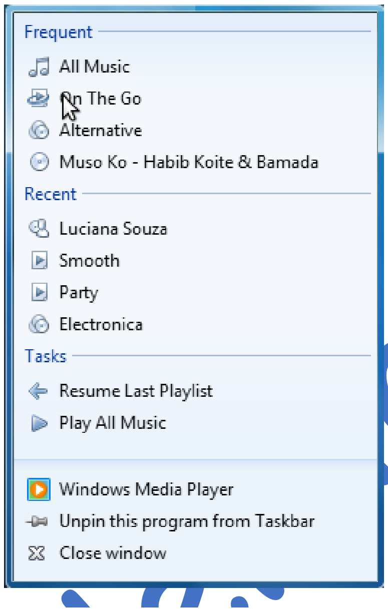


Figure 1: GUI sample for Windows

In Figure 1 above, a small icon is displayed next to each label. To have a windows look and feel in our GUI's, we need QStyle widget in Qt C++ application. A specific class that gives Microsoft window's look and feel to widgets or user interfaces is QWindowsStyle class (Qt Project,2013). We will use this class in the final version of our software when building the GUI.

For a graphical interface to be user-friendly, it should have at least the following features:

- **Clarity of the design:** It accounts for the design. A clear GUI design conveys the whole message of GUI easily to the user (Vallerio *et al.*, 2006).
- **Responsive:** The GUI should be responsive to the users, i.e. it should not be slow and laggy while in use. Slow response of the GUI can make users frustrated and can cause them to reject the application (Vallerio *et al.*, 2006).

- **Familiar Design:** Themes and names of the UI components should be familiar to the users. I.e. names of the UI components to be used in GUI should have the same names as in any other similar application (Vallerio *et al.*, 2006).

3.5. DHCP Server

DHCP stands for Dynamic Host Configuration Protocol. The main functionality of a DHCP server is to control the network configuration of a host through a remote server (Functionspace.org, 2014). DHCP server is by default installed not only in Linux OpenSUSE but also in several other operating systems. DHCP server is considered as an alternative to the manual configuration of network setting on a network or host device which is usually time-consuming (Functionspace.org, 2014). The header information of the DHCP messages is provided in table 2.

Table 2: DHCP Message headers (Himanshu, 2015)

FIELD	OCTETS	DESCRIPTION
op	1	Type of message
htype	1	Type of hardware address
hlen	1	Length of the hardware address
hops	1	Used in case of relay agents. Clients set them to 0
xid	4	It is a transaction IF mainly used during the session by the client and server.
secs	2	This is a time which holds the time it has taken since the request is sent from the client. Usually it will be in seconds
flags	2	Flags
ciaddr	4	Client IP address
yiaddr	4	This is an IP address usually allocated by the server to the client
siaddr	4	Server IP address

giaddr	4	IP address of the relay agent
chaddr	16	This is an address used by the client hardware
sname	64	Hostname of the server
file	128	Boot file name
Options	var	Additional options

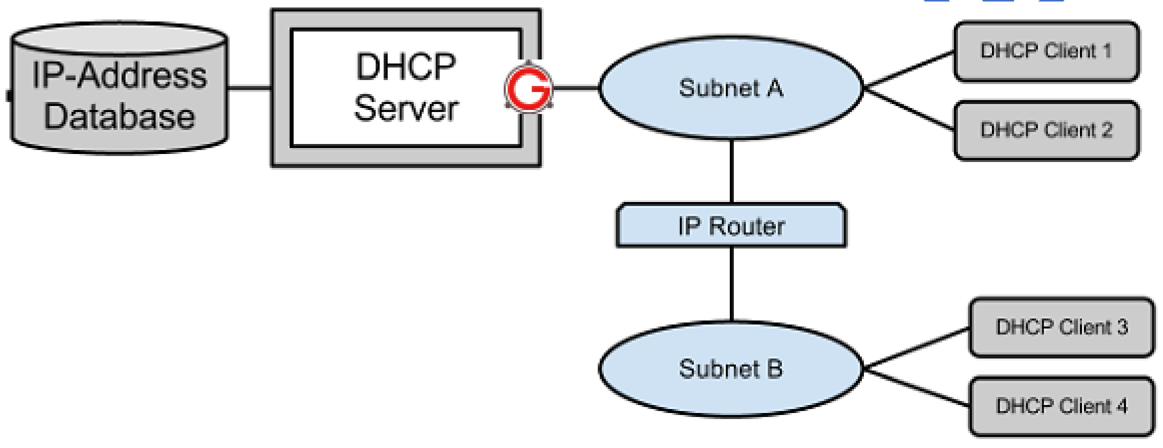


Figure 2: Workflow of the DHCP server (Himanshu, 2015)

Figure 2 shows the workflow of the DHCP server (Himanshu, 2015). There are several DHCP messages available which are DHCPDISCOVER, DHCPOFFER, DHCPREQUEST, DHCPACK, DHCPNAK, SHCPDECLINE, DHCPINFORM, and SHCPRELEASE. The description of the messages are provided in table 3.

Table 3: DHCP messages (Himanshu, 2015)

DHCP messages	Description
DHCPDISCOVER	This is the message which initiates the DHCP interaction between client and server. In general, this message is sent by the client who is connected to the local subnet. The source IP address of this message is 0.0.0.0. However, the broadcast message IP address is 255.255.255.255

DHCPOFFER	This is a message that is sent by the DHCP server to the DHCP client as a response to the message DHCPDISCOVER. SHCPOFFER message contains network configuration setting for the client that sends the DHCPDISCOVER message.
DHCPREQUEST	This is the message sent from the client to the server as a response to the message DHCPOFFER. DHCPREQUEST indicate that the network configuration is accepted.
DHCPACK	This message is sent as a response to the message DHCPREQUEST from the server to the client. DHCPACK is the response message to DHCPREQUEST. Moreover, DHCPACK is the acknowledgement by the server to the client.
DHCPNAK	This is a different method for SHCPNAK. DHCPNAK is the acknowledgement transferred from the server to the client whenever the request will not be able to process.
SHCPDECLINE	This is transferred from the client to the server when the client identifies that the IP address assigned by the server is already in use.
DHCPINFORM	When the client needs configuration or setting information for the IP address, this message will be sent to the server.
SHCPRELEASE	When a client wants to opt out from the IP address provided by the server, then it will send this message to the server.

3.6. DNS Server

The DNS performs in an identical manner to an internet phone book. When a user inserts the website address into the web browser, the DNS server uses the IP address to load the website into the user's browser (Gonyea, 2015). For example, if the user types the address http://dyn.com, then the DNS server will use the website IP address 204.13.248.115 to load the page. Should in case the DNS is not available, then the user can visit the website not by its name, but by the IP address (Gonyea, 2015).

Figure 3: DNS server (Gonyea, 2015)

When a user visits a website, the computer follows a series of steps to convert the humanly readable form website to the machine-readable IP address (Gonyea, 2015). This process is done when the user views the website or sends emails or even when the user watches online radios. There are seven steps used in this process which are (Gonyea, 2015):

- Step 1: Request for information
- Step 2: Inquire for the recursive DNS servers
- Step 3: Inquire for the root name server
- Step 4: Inquire for the Top-Level Domain (TLD) nameservers
- Step 5: Request for the confident DNS servers
- Step 6: Retrieve the essential record
- Step 7: Retrieve the required answer

The steps mentioned above will only take milliseconds to complete execution.

4.0. Specification

This project had different stages of research conducted in order to choose which operating system, programming language, and application development framework to use in solving the problem. Finally, it was concluded that the Linux operating system and C++ programming language with QT framework would be used for the development. Also, literature and web research were done to fully understand how a DNS and DHCP servers works, which help the author to configure the servers. Further research work was conducted to study similar software functionality to derive a proper system specification. Additionally, more research work was conducted to understand GUI design mechanisms; which will help in developing a user-friendly GUI for the system. The final functional, non-functional and system requirements list identified as follows:

Functional Requirements

- Server administrator login
- Successful administrator login should be able to configure different servers (DNS and DHCP)
- Exploring and identifying configuration files and methods used for manual configuration of the web server
- Connecting different components of GUI with its relevant configuration script

Non-functional Requirements

- GUI should be user-friendly
- The system should have help or tooltip to aid users in studying and understanding the GUI's functionalities
- The system should have installation documentation given to the company to aid in the installation or deployment of the software on their computers.

System Requirements

- Linux OpenSUSE Operating system
- QT Application Development Framework

- GCC C++ Compiler

5.0. Methodology

The Software Development Life Cycle (SDLC) model was used in this development project. SDLC is fundamentally sequences of steps, or phases, which offer a model for the software development (Roebuck, 2012). The main benefit of using SDLC model is to develop effective, high quality and error-free software. Moreover, it also helps the developer to track the progress of the project development and monitor the time frames in order to meet the deadlines (Roebuck, 2012). Furthermore, SDLC comprises of six stages which are Requirement gathering and analysis, Design, Implementation or coding, Testing, Documentation and Deployment and Maintenance (See Figure 4).

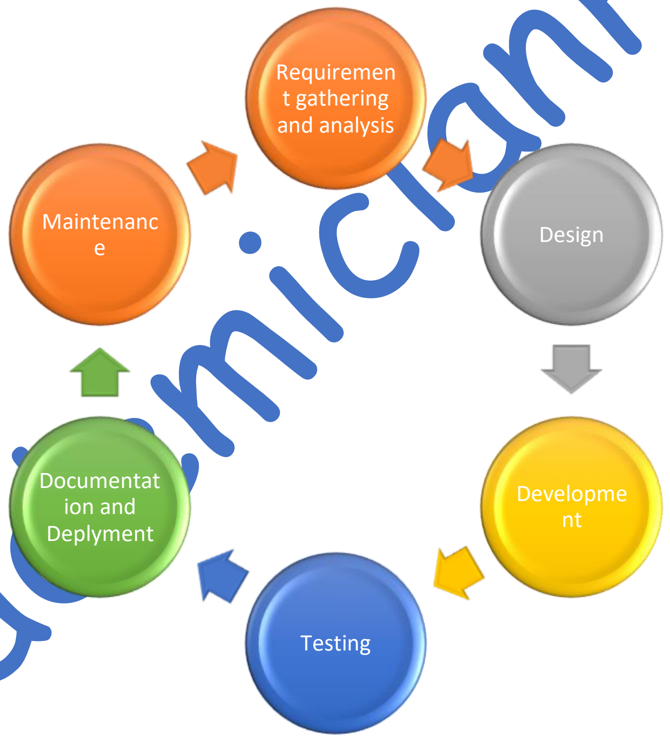

Requiremen t gathering and analysis
Design
Developme nt
Testing
Documentat ion and Deplyment
Maintenanc e

Figure 4: System Development Life Cycle (SDLC)

5.1. Stage 1: Requirement gathering and analysis

The requirement gathering was conducted by studying the case study of AMCE and identifying the implicit and explicit functional, non-functional and system requirements. The initial identified requirement list received feedback from ACME relevant officers. The final requirement list was created and sent to get final approval from the supervisor as well as ACME officers.

5.1.1. Initial requirement list

The basic requirement of ACME is to develop a general GUI interface to configure DHCP and DNS servers. The list of initial functional, and the non-functional requirement is provided below:

Functional Requirements

- The system should have GUI which will allow the user to configure DHCP and DNS servers by clicking a button.
- Exploring and identifying configuration files and methods used for manual configuration of the web server.
- Connecting different components of GUI with its relevant configuration script

Non-functional Requirements

- GUI should be user-friendly
- The system should have help or tooltip to aid users in studying and understanding the GUI's functionalities

5.1.2. Feedback received for the initial requirement list

ACME officers replied by saying that at the moment, they require a single Administrator login and they are willing to use the current development as a prototype to understand how the system works. Moreover, the officers requested that the GUI needs to be very simple in the first phase. Also, they also mentioned that in this first stage, they expect to see the additional advanced functionalities, which can be embedded into the system. Furthermore, they highly advised that system specification should have a basic idea of how much they are going to spend for licensing.

5.1.3. Final Requirement list

The final requirement list which was agreed by the ACME officers and the supervisor is given below:

Functional Requirements

- Server administrator login
- Successful administrator login should be able to configure different servers (DNS and DHCP)
- Exploring and identifying configuration files and methods used for manual

configuration of the web server

- Connecting different components of GUI with its relevant configuration script

Non-functional Requirements

- GUI should be user-friendly
- The system should have help or tooltip to aid users in studying and understanding the GUI's functionalities
- The system should have installation documentation given to the company to aid in the installation or deployment of the software on their computers.

System Requirements

- Linux OpenSUSE Operating system
- QT Application Development Framework
- GCC C++ Compiler

5.2. Stage 2: Design

During the design stage, the wireframe, class diagram and activity diagram were developed in order to have a clear understanding of the application.

5.2.1. Wireless diagram

There were initial and final wireframe diagram, which were drawn using Microsoft Visio. The initial wireframe design consists of 3 GUIs. The description of the GUIs are given below:

Wireframe diagram 1: The administrator interface diagram, comprises of the application form (container), dialogue form and panel, menu bar for Navigations (servers, administrators, describe problems, and so on), a dialogue box to close, maximise and minimise the dialogue or application. The menu bar for DHCP, DNS, WEB server works as navigations to open new window and label and textbox tools for search. (See APPENDIX B)

Wireframe diagram 2 for DHCP consists of dialogue form (container), panels, labels, textboxes button (generate, read, exit, help) and dialogue button to close, maximise and minimise the dialogue window. (SEE APPENDIX D)

Wireframe diagram 3 for DNS consists of dialogue form (container), Panels, Labels, Textboxes. The radio button to choose, Button (generate, exit, help) and dialogue button to close, maximise and minimise the dialogue window. (See APPENDIX C)

The modified final wireframe diagrams description was given below:

Wireframe diagram 1 for the general GUI where it will allow the administrator to go into the first stage of login. (See APPENDIX E)

Wireframe diagram 2 is for the login of the Administrator. (See APPENDIX F)

Wireframe diagram 3 is for administrator interface diagram, which comprises of the application form (container), dialogue form and panel, menu bar for Navigations (servers, administrators, describe problems, and so on), a dialogue box to close, maximise and minimise the dialogue or application. The menu bar for DHCP, DNS, WEB server works as navigations to open new window and label and textbox tools for search. (See APPENDIX G)

Wireframe diagram 4 is for DHCP, and it consists of dialogue form (container), panels, labels, textboxes button (generate, read, exit, help) and dialogue button to close, maximise and minimise the dialogue window. (SEE APPENDIX H)

Wireframe diagram 5 is for DNS, and it consists of a dialogue form (container), Panels, Labels, Textboxes. The radio button to choose, Button (generate, exit, help) and dialogue button to close, maximise and minimise the dialogue window. (See APPENDIX I)

5.3. Stage 3: Implementation or coding

The development of the configuration GUI for a Linux OpenSuse uses C++ programming language on QT Application Framework. The artefact will be a programme running on a Linux server that will be used to configure DNS and DHCP servers.

5.4. Stage 4: Testing

The testing of the prototype was conducted in two different stages. The first stage was conducted by the developer and the second stage testing was conducted by some selected people (testers).

Testing conducted by the developer

Several types of testing could be conducted by the developer, some of which are unit testing, regression testing, integration testing, system testing and performance testing. The different types of testing will be conducted at different stages of the development. Such testing includes the ones done during the unit development, during the integration of the units, after the development and so on. The explanation of the different types of testing conducted by the developer are provided below:

- **Regression testing:** This testing will help the developer to discover many new errors and also configuration errors. Regression testing result can lead to modifications of the prototype.
- **Performance testing:** This testing can create different sets of a large number of test cases for different transactions or functionalities in the prototype. This testing will help the developer to perfect all the possible ways of performing the specific transaction or functionality. Several unexpected errors could also be identified.
- **Unit testing:** Unit testing applies to this development because this development would be done in units. The developed units are Administrator Interface, DNS and DHCP. This testing should be easy because the units were not too complicated. Moreover, errors at the early stage of development that needs to be fixed before proceeding to the other stage can be identified.
- **Integration testing:** This testing was performed during the joining of the small units. It can detect errors or bugs in the integration codes.
- **System testing:** The only time that this testing will be performed is after the system is integrated and made as one single working unit. Moreover, this testing will check the overall functionalities and behaviours of the system

Testing conducted by the ACME officers or client

The testing and the application modification was conducted in a cycle until the ACME officers agreed with the final application. This testing was mainly on the functionality of the application.

5.5. Stage 5: Documentation and Deployment

During this stage, the tested system would be packaged with the installation guide, and users installation manual for the software are prepared.

5.6. Stage 6: Maintenance

Regular updates and any other installation issues will be handled in this stage.

NOTE: Stage 3 and Stage 4 will also use the prototyping model, where the development and testing will be a repetitive process.

6.0. Implementation/Development

This section comprises the implementation of a user-friendly Graphical User Interface (GUI) to configure or start important service on the servers in the Linux operating system using C++ in QT framework. The implementation has six GUI screens.

Screen 1

This screen is mainly known as a start-up screen which has two major buttons, which are: 'Press to launch' and 'Quit. Moreover, the main screen also has a time label which shows the current system time. Furthermore, the main screen consists of the Linux SUSE logo to indicate that this application will work in a Linux operating system. The screen printout of the main screen or start-up screen is provided in Figure 5.

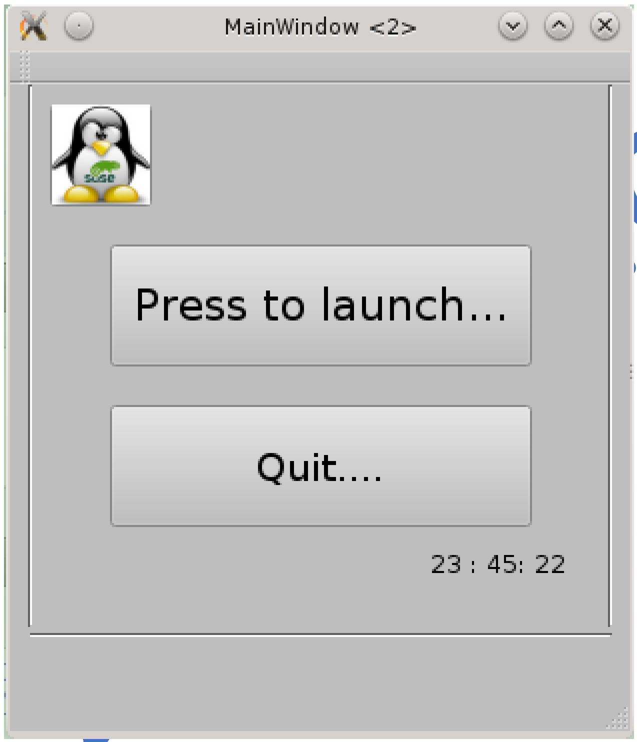


Figure 5: Screen printout of the main screen

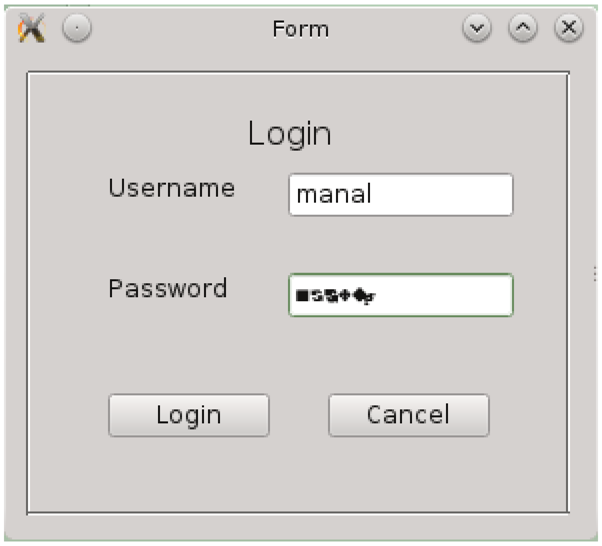
'Press to launch' button will bring out the login screen. The C++ code used for 'Press to launch' button is given below:

```
void MainWindow::on_pushButton_clicked()
{
   home2.show(); //show login page
   this -> hide(); // hide the main page
```

}

Screen 2

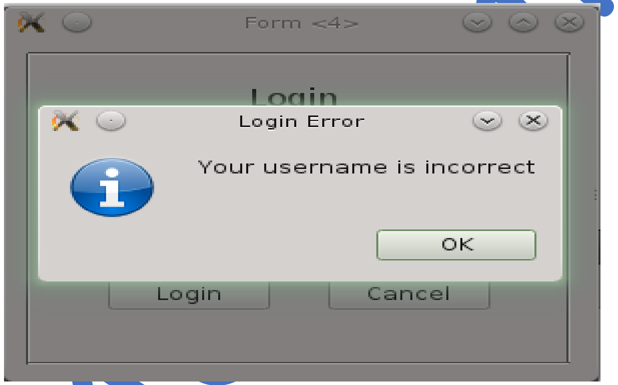
This is the login screen. In this screen, the user must give an accurate username and password in order to get access to the home page of the application. The actual username and password for the administrator are manual and 123456. The printout of the login screen is provided in figure 6.



Username: manal

Figure 6: Screen printout of Login Page

If the user enters an incorrect username, the application will then prompt a message box saying that the username is incorrect. See Figure 7 for the username is incorrect prompt.



Your username is incorrect

Figure 7: Screen printout of username is incorrect input

If the user enters an incorrect password, then the application will prompt a message box saying that the password is incorrect. See Figure 8 for the password is incorrect prompt.

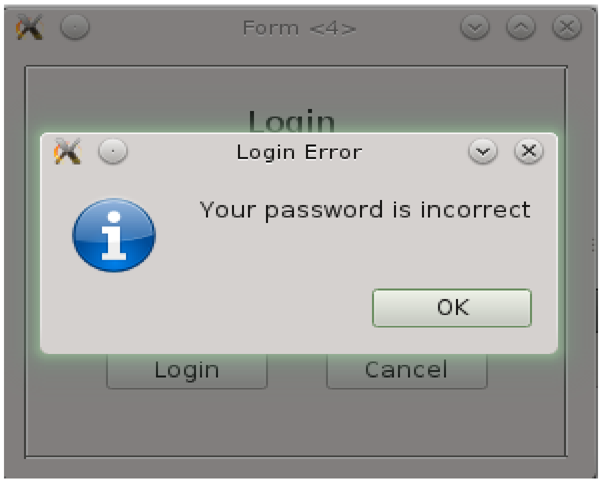


Figure 8: Screen printout for password is incorrect prompt

If the username and password are correct, then it will lead to the home page of the application. The C++ code for the login button is provided below:

```
void login::on_pushButton_clicked()
{
  if (ui->lineEdit->text() == "manal") // Check whether the username is equal to manal
  {
    if(ui->lineEdit_2->text() == "123456") // check whether the password is equal to 123456
    {
      home6.show(); // show the home page
      this -> hide(); // Hide login page
    }
    else
    {
      QMessageBox ::information(this, tr("Login Error"),tr("Your password is incorrect")); // Display message box when the password is wrong
      ui->lineEdit->setText(""); // Clear Username field
      ui->lineEdit_2->setText(""); // Clear password field
      ui->lineEdit->setFocus(); // Set the focus of the cursor to username field
    }
  }
  else
  {
    QMessageBox ::information(this, tr("Login Error"),tr("Your username is incorrect")); // Display message box when the username is wrong
    ui->lineEdit->setText(""); // Clear Username field
    ui->lineEdit_2->setText("");// Clear password field
    ui->lineEdit->setFocus();// Set the focus of the cursor to username field

  }
```

}

Screen 3

This is a homepage. The homepage has eight menu button such as Servers, Administrator, Describe Problems, Comments, Solutions, Keep Trace of Changes, Attachment and Checklist. The brief explanation of each section is provided below:

Server button: it is to get the servers which can be configured using this application

Administrator button: It is the button which will give the list of privileges an administrator has in the system

Describe Problems button: This button will help the administrator log the problem faced and the solution adopted for future reference.

Comments: This section allows the user to write the comment and experience concerning this application.

Solutions: This section displays the additional solutions available for the application to enhance its functionalities such as error detecting software and so on.

Keep Track of Changes: This will keep the record of restructuring and maintenance conducted in the application.

Attachment: This section will have an external attached document, which will help the user to enhance his/her knowledge about Linux servers.

Check List: This is the checklist which provides the current version functionalities of the application.

The screen printout of the home page is provided in figure 9. The homepage also has a search option which will allow the user to search for his interest. The current application has only server button working, and the other buttons are dummy entities which will be considered in future development.

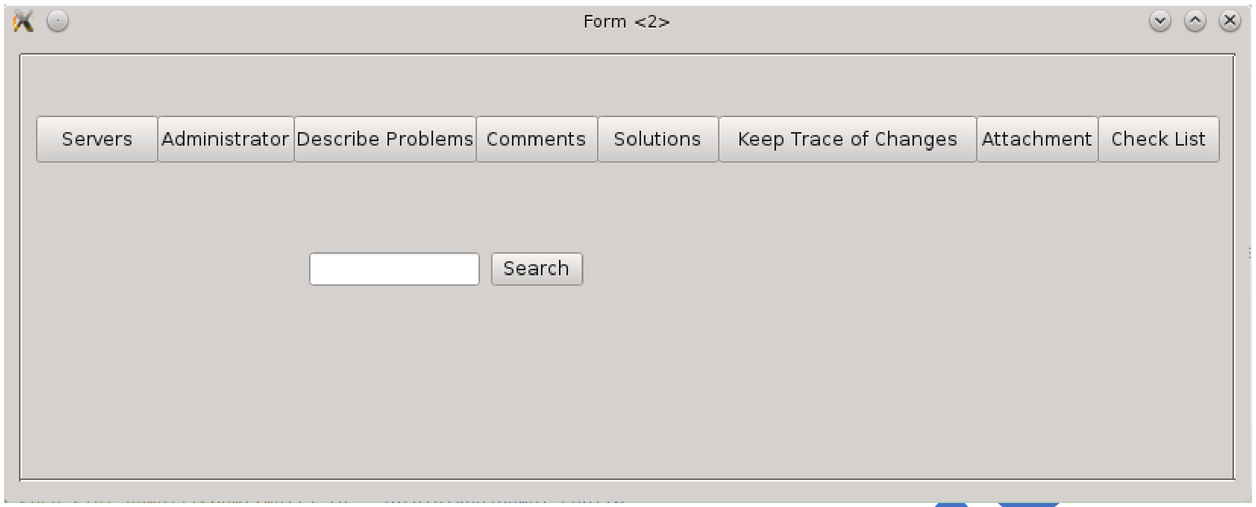


Figure 9: Screen printout of the homepage

The server option button will trigger configuration page.

Screen 4

This is the configuration homepage which shows the servers that can be configured by pressing the button. This page has two command buttons to configure DHCP and DNS. The Screen printout of the configuration home page is provided in figure 10.

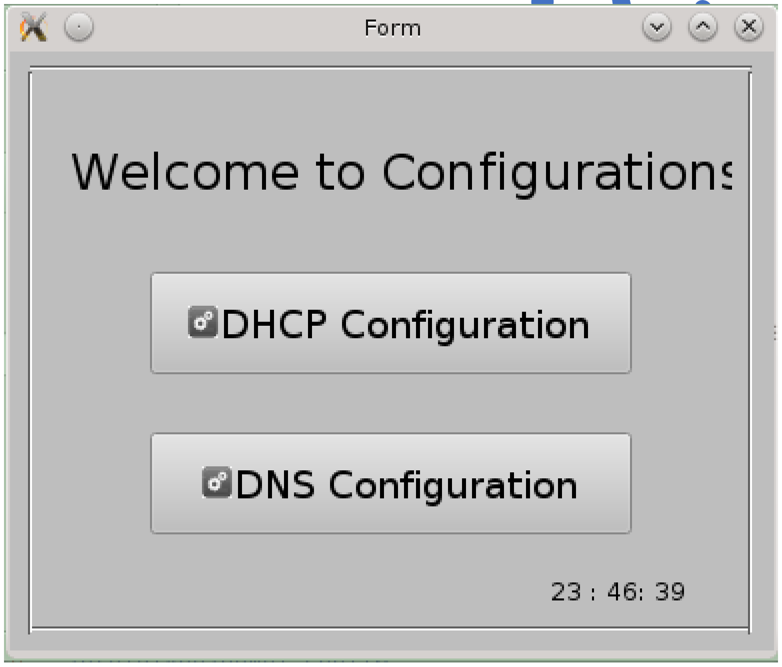
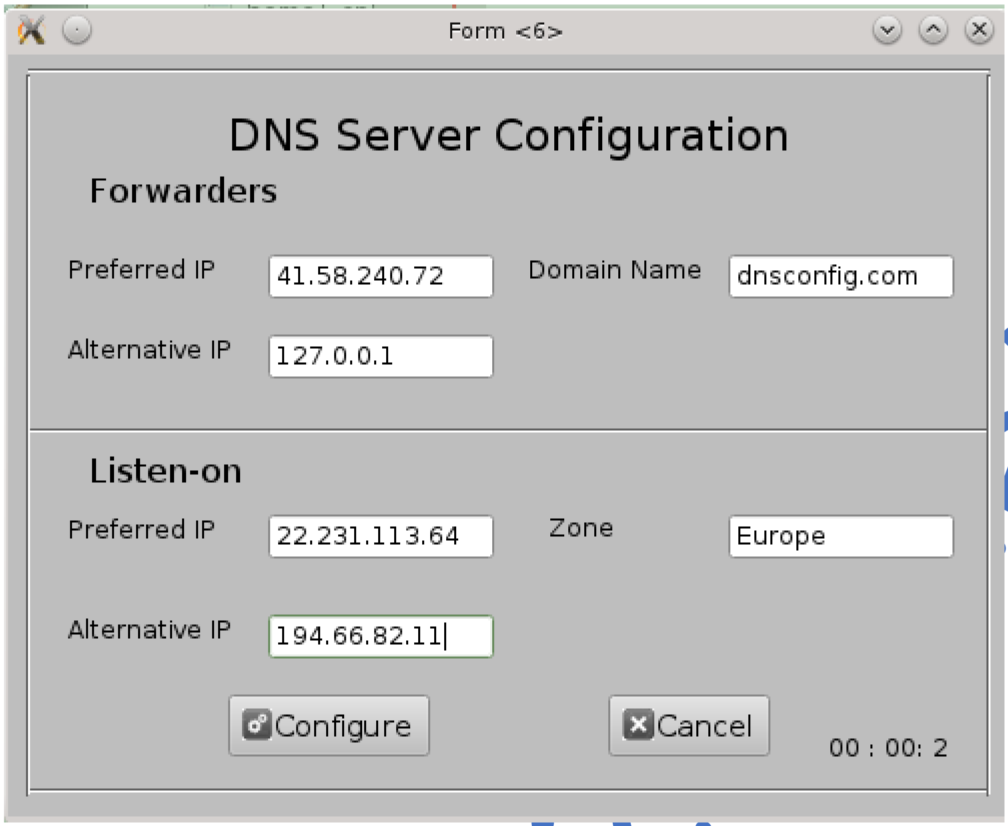


Figure 10: Screen printout of the home screen for configuration

Screen 5

This is a DNS Server configuration page which requests the user to insert preferred IP and alternative IP for Forwarders, Domain Name, Preferred IP and Alternative IP for Listen-on and Zone. The screen printout for DNS Server Configuration is provided in figure 11.



Form <6>

DNS Server Configuration

Forwarders

Preferred IP: 41.58.240.72

Domain Name: dnsconfig.com

Alternative IP: 127.0.0.1

Listen-on

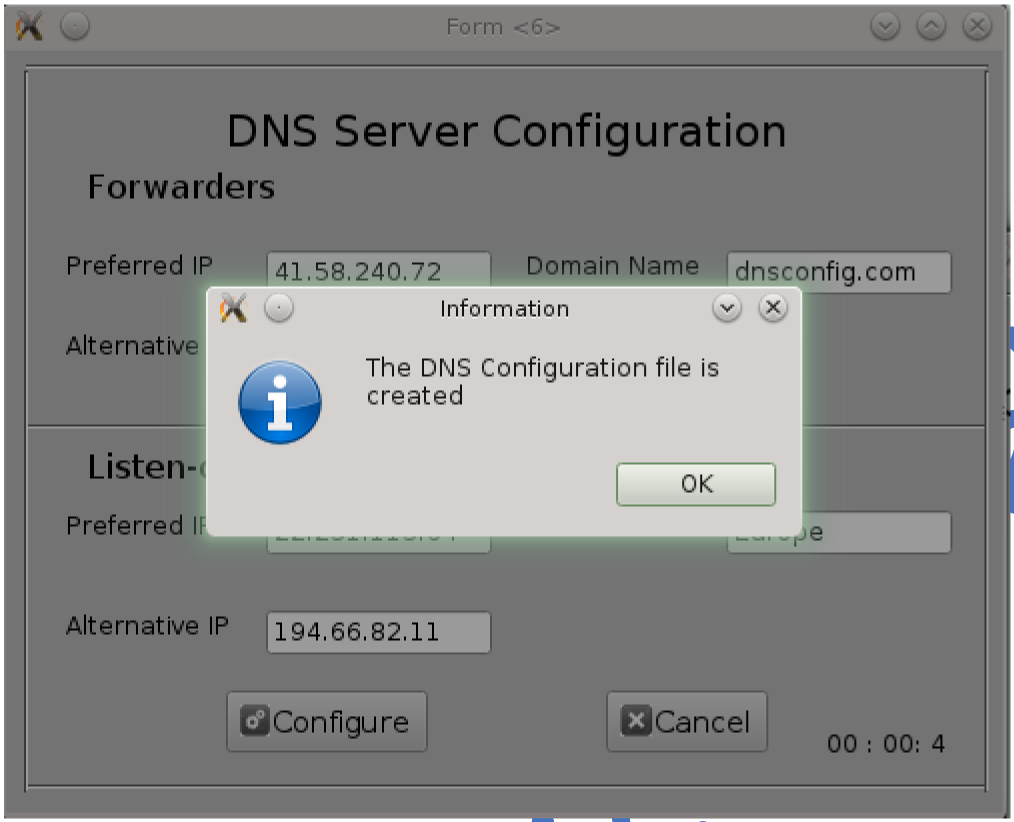
Preferred IP: 22.231.113.64

Zone: Europe

Alternative IP: 194.66.82.11

Figure 11: Screen printout for DNS configuration page

When the configuration file is created, the system will indicate the successful configuration file created in a message box. The message box notification is provided in figure 12.



Form <6>

DNS Server Configuration

Forwarders

Preferred IP 41.58.240.72

Domain Name dnsconfig.com

Alternative

Information

The DNS Configuration file is created

OK

Listen-

Preferred IP

Alternative IP 194.66.82.11

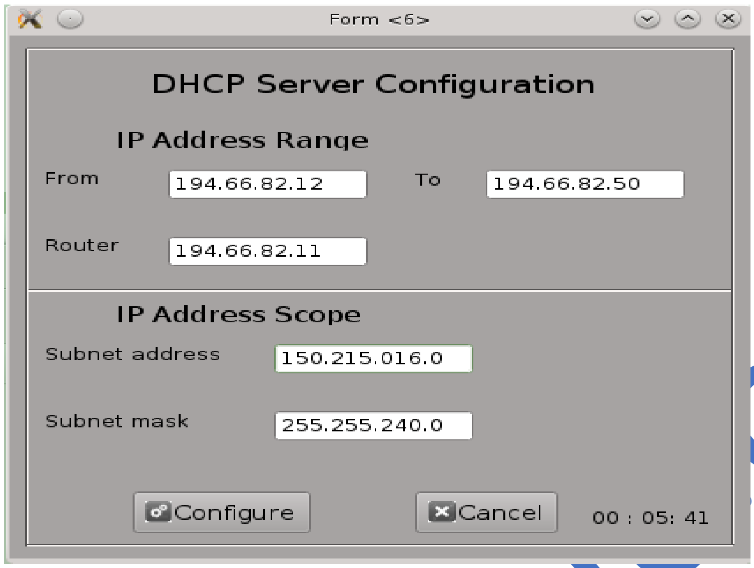
Figure 12: Screen printout of DNS config file creation

The C++ code for the Configure button is provided below:

```
void DNS::on_pushButton_2_clicked()
{
  QFile file("named.conf"); //create the configuration file
  if (!file.open(QIODevice::WriteOnly | QIODevice:: Text)) // Check whether the file is open
     return;
  QTextStream ts(&file);
  ts << "forwarders { " + ui -> lineEdit->text()+ "; " + ui->lineEdit_4->text()+";};\n";
  ts << "listen-on { " + ui-> lineEdit_2 -> text()+"; "+ui->lineEdit_3->text()+";};\n";
  ts << "Domain-name { " + ui->lineEdit_5 -> text()+";};\n";
  ts << "Zone { " + ui->lineEdit_7-> text()+";};\n";
  ts << "notify no;";
  ts << "};\n";
  file.close(); // close the file
  QMessageBox ::information(this, tr("Information"),tr("The DNS Configuration file is created"));
  // message box to display that the configuration file is created
}
```

Screen 6

This is a DHCP Server Configuration page which requests users to insert IP Address Range from and To, Router address, Subnet address and subnet mask. The configuration button will create the DHCP configuration file. The screen printout of DHCP Server configuration page is provided in figure 13.


Form <6>

DHCP Server Configuration

IP Address Range

From 194.66.82.12 To 194.66.82.50

Router 194.66.82.11

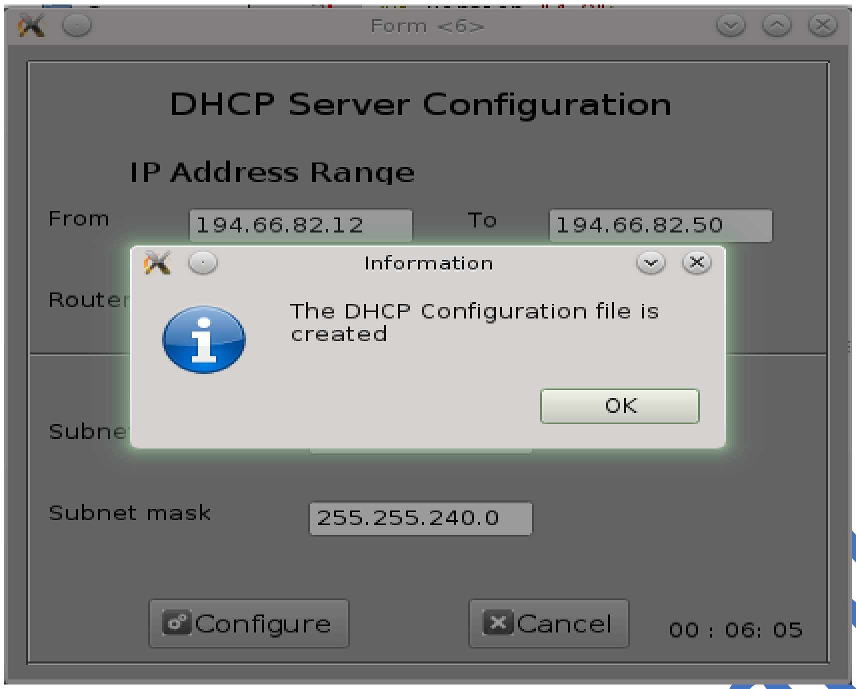
IP Address Scope

Subnet address 150.215.016.0

Subnet mask 255.255.240.0

Figure 13: Screen printout of DHCP server page

Clicking on the configure button will create a DHCP configuration file. The success of creating a configuration file will be communicated to the user by a message box. The message box display is provided in figure 14.


Form <6>

DHCP Server Configuration

IP Address Range

From 194.66.82.12 To 194.66.82.50

Router

Subne

Subnet mask 255.255.240.0

Information

The DHCP Configuration file is created

Figure 14: Screen printout of DHCP config file created

The C++ code for DHCP config button is provided below:

```
void dhcp::on_pushButton_clicked()
{
  QFile file("dhcpd.conf"); //create the configuration file
  if (!file.open(QIODevice::WriteOnly | QIODevice:: Text)) // Check whether the file is open
    return;
  QTextStream ts(&file);
  ts << "subnet " + ui -> lineEdit->text()+ " netmask " + ui->lineEdit_2->text()+" {\n";
  ts << "range " + ui-> lineEdit_3 -> text()+" "+ui->lineEdit_4->text()+";\n";
  ts << "option routers " + ui->lineEdit_5->text()+";\n";
  ts << "}\n";
  file.close();
  QMessageBox ::information(this, tr("Information"),tr("The DHCP Configuration file is created"));
  // message box to display that the configuration file is created
}
```

General Code

Display current system time

The system has a label which will display the current system time in a specific format. The C++ code code to display this is provided below:

```
void MainWindow::showTime()
{
  // This is the method which is used to show the current system time
```

```
   QTime time =QTime::currentTime();
   QString time_text = time.toString("hh : mm: ss");// The time will be shown in the given format
   ui ->label->setText(time_text); // The time will be shown in the label
}
```

Display the form in the centre of the computer screen

The C++ code which is used to display the form in the centre of the computer screen is provided below:

```
   ui(new Ui::MainWindow)
{
   ui->setupUi(this);
   //positioning the running application screen to center of the display screen
   QRect position =frameGeometry();
   position.moveCenter(QDesktopWidget().availableGeometry().center());
   move(position.topLeft());
   // timer
   QTimer *timer=new QTimer(this);
   connect(timer , SIGNAL(timeout()), this, SLOT(showTime()));
   timer->start();
}
```

Cancel button code

The cancel button is in many forms. The code for the cancel form is provided below:

```
void dhcp::on_pushButton_2_clicked()
{
    this ->close(); //cancel the application
}
```

The complete application have headers (.h) and C++ (.cpp) files such as dhcp, dns, main, mainwindow, login, and home. (See Appendix XX)

Output

Configuration file created for DHCP server

The configuration file for DHCP server is named as 'dhcpd.conf'. The content of the file is given below:

```
subnet 150.215.016.0 netmask 255.255.240.0 {
range 194.66.82.12  194.66.82.50;
option routers 194.66.82.11;
}
```

Configuration file created for DNS server

The configuration file for DNS server is named as 'named.conf'. The content of the file is given below:

```
forwarders { 41.58.240.72; 127.0.0.1;};
listen-on { 22.231.113.64; 194.66.82.11;};
Domain-name { dnsconfig.com;};
notify no;};
```

7.0. Testing

The testing was conducted in two stages. The first stage is mainly by a developer who does the testing during and after the development of the application. The type of testings used by the developer are regression testing, performance testing, unit testing, integration testing and system testing. The second stage of the development is to get a continuous feedback after the first development. This includes adopting the extra functionalities to the application.

7.1. Testing conducted by the developer

The development for each server was developed in separate unit, therefore, the independent unit testing was conducted. The unit testing details are given in table 4.

Table 4: Unit testing summary

Testing type	Name	No of test data	No of Errors found	Error Description	Developer comment
Unit testing	DHCP unit	5	2	The configuration file was not creating, and Message box was not displayed	All errors were rectified
Unit testing	DNS unit	5	1	The screen is not displayed in the middle	The error was rectified
Unit testing	Login unit	5	3	No verification for username or password is wrong	All errors were rectified.

When the units were created, the developer integrated it with additional forms. The testing was conducted during the integration stage, to track any errors that occurred during the integration testing. The integration testing details are given in table 5.

Table 5: Summary of integration testing results

Testing Type	Name	No of test data	No of errors found	Error Description	Developer comment
Integration testing	Integration of DHCP and DNS units with a configuration home page	2	0		No errors identified.
Integration testing	Integration of configuration with the homepage	4	1	The homepage Server button was not working	The error was rectified.
Integration testing	Integration of home page and log in	6	1	The login page is not redirecting to home page	The error was rectified

The developer conducted the complete system testing, regression testing and performance testing and the details are provided in table 6.

Table 6: Summary of Regression System and Performance Testing Results

Testing Type	Name	No of test data	No of errors found	Error description	Developer comment
System testing	Complete system	3	0		No errors were identified
Regression testing	Complete system	1	1	The DNS and DHCP files are crashing during regression testing	Error identified was left for future works
Performance testing	Complete system	3	1	The application is slowing down when other files are open.	Error identified was left for future work.

7.2. Testing conducted by the ACME officers or client

The testing was conducted twice. The comments received after the first testing is given below:

- In DHCP configuration, the first line should be IP address rather than subnet.
- In the DNS configuration, additional fields such as domain name and zone need to be included
- The application should use correct IP address and subnet configurations
- The GUI looks interesting and easy to use
- The application should preferably run in the middle of the screen

The comments received from the second testing is given below:

- The configuration of DNS and DHCP forms are good
- The application must have a login. The user should provide a valid username and password to access the application functionalities.
- It is highly advised to include system time in the application.

Comments received from the first testing, enhanced the configuration of DNS and DHCP servers and the comments received from the second testing added new features such as login, and system time into the application.

8.0. Issues arising from Implementation and Test

The issues faced and the solutions taken for the implementation and testing are listed below:

- **Issue 1: Time-consuming when developing in QT C++.** The developer has to completely read the C++ manual for QT in order to develop.
- **Issue 2: It was challenging to install the dual operating system in the laptop**. The developer has to use Virtual Box to install Linux.
- **Issue 3: Downloading QT was a challenge because the file is huge.** The developer has to approach the library to get a stable internet to complete the download.
- **Issue 4: During testing, the errors were many, and it took the developer a long time to correct the errors.** The solution adopted was to record the errors and then look for a solution for the errors on the internet.

9.0. Conclusion and Recommendations

This project provides a common GUI for configuring servers on Linux platform using QT C++.

Recommendation

- The GUI could integrate other servers such as web server, proxy server, FTP server, Samba server, NFS server, Database server, LDAP server, time server, print server, mail server, Linux virtual server, Linux internet gateway, NIS authentication server, SSL server, and application server.
- The system should be able to stop the currently running server.
- The administrator should be able to view the currently running servers
- The administrator should have an option of scheduling the start and stopping the server in the future.
- The administrator should be able to view the servers installed.
- The administrator should be able to restart the servers.
- The administrator should be able to reset the password
- The administrator should be able to create a new user account
- The system should have a pointer tip description of the icons and buttons

10.0. References

Bjørner, D. (2006). Software engineering. Berlin: Springer-Verlag.

Blanchette, J., M, Summerfield.2008. *C++ GUI programming with Qt 4*. 2nd ed. Upper Saddle River, N.J. London: Prentice Hall.

Deepthi, R.S.; Sankaraiah, S., "Implementation of mobile platform using Qt and OpenCV for image processing applications," *Open Systems (ICOS), 2011 IEEE Conference on* , vol., no., pp.284,289, 25-28 Sept. 2011.

Fedora, 2014, [Online] [viewed on 7th November 2014] Available at: http://fedoraproject.org

Functionspace.org, (2014). DHCP and its compatibility. [online] Available at: http://functionspace.org/topic/3354/DHCP-and-its-compatibility [Accessed 1 May 2015].

GCFLearnFree.org, (2015). Computer Basics: Understanding Operating Systems. [online] Available at: http://www.gcflearnfree.org/computerbasics/2 [Accessed 5 Jan. 2015].

Gois, J.P.; Batagelo, H.C., "Interactive Graphics Applications with OpenGL Shading Language and Qt," *Graphics, Patterns and Images Tutorials (SIBGRAPI-T), 2012 25th SIBGRAPI Conference on* , vol., no., pp.1,20, 22-25 Aug. 2012.

Gonyea, C. (2015). DNS: Why It's Important & How It Works | Dyn Blog. [online] Dyn.com. Available at: http://dyn.com/blog/dns-why-its-important-how-it-works/ [Accessed 1 May 2015].

Hamed, A.M.M.; Abushama, H., "Popular agile approaches in software development: Review and analysis," *Computing, Electrical and Electronics Engineering (ICCEEE), 2013 International Conference on* , vol., no., pp.160,166, 26-28 Aug. 2013.

Hans van Vliel, *Software Engineering: Principles and Practice,* JOHN WILEY & Sons, 1995. ISBN 0471936111.

Himanshu A, W. (2015). What is DHCP and How DHCP Works? (DHCP Fundamentals Explained). [online] Thegeekstuff.com. Available at: http://www.thegeekstuff.com/2013/03/dhcp-basics/ [Accessed 1 May 2015].

Holmes, B. (1990). Pascal programming. London: DP Publications.

Ichbiah, J. (1991). Rationale for the design of the Ada programming language. Cambridge [Cambridgeshire]: Cambridge University Press.

Jovanovic, D.; Dogsa, T., "Comparison of software development models and their usage in computer-telephony systems," *Telecommunications, 2003. ConTEL 2003. Proceedings of the 7th International Conference on* , vol.2, no., pp.587,592 vol.2, 11-13 June 2003.

Kernighan, B. and Ritchie, D. (1988). The C programming language. Englewood Cliffs, N.J.: Prentice Hall.

Kremer, U. and Rame, M. (1993). Compositional oil reservoir simulation in Fortan D. Houston, Tex.: Rice University, Dept. of Computer Science.

Learn access, 2011. *The Waterfall Development Methodology*. [Online] [viewed 7th May 2014] Available at: http://learnaccessvba.com/application_development/waterfall_method.htm .

Microsoft.com, (2015). Segoe UI. [online] Available at: https://www.microsoft.com/typography/fonts/family.aspx?FID=331 [Accessed 1 May 2015].

Novak, I.2011.*Beginning Visual studio LightSwitch development*. Indianapolis, Ind: Wiley.

OpenSuse, 2014, [Online] [viewed on 8th November 2014] Available at: http://www.opensuse.org/en/

Opensuse-guide.org, (2015). 4. Installation - Howto Install openSUSE on Your Computer. [online] Available at: http://opensuse-guide.org/installation.php [Accessed 1 May 2015].

Poortvliet, J. 2013. *GTK* [Online] [viewed 4th May 2014] Available at: http://en.opensuse.org/GTK

Poortvliet, J. 2013. *OpenSUSE 13.1: Ready For Action!* [Online] [viewed 4th May 2014] Available at: https://news.opensuse.org/2013/11/19/opensuse-13-1-ready-for-action/

Qt-project.org, (2015). Qt Project. [online] Available at: http://qt-project.org/ [Accessed 6 Jan. 2015].

Red Hat, 2014, [Online] [viewed on 7th November 2014] Available at: http://www.redhad.com/en/

Roebuck, K. (2012). Systems Development Life Cycle (SDLC). Dayboro: Emereo Publishing.

Sangal, R. (1991). Programming paradigms in LISP. New York: McGraw-Hill.

SearchITchannel, 2010. Linux DHCP server and client: Configuration and deployment. [Online] [viewed 2nd May 2014] Available at: http://searchitchannel.techtarget.com/feature/Linux-DHCP-server-and-client-Configuration-and-deployment

Shelly, G., Cashman, T. and Foreman, R. (2000). Structured COBOL programming. Cambridge, MA: Course Technology.

Shneiderman, B., C, Plaisant.2005. *Designing the user interface: strategies for effective human-computer interaction. 4th ed.* 4th ed. New York: Addison Wesley.

Stroustrup, B. (1997). The C++ programming language. Reading, Mass.: Addison-Wesley.

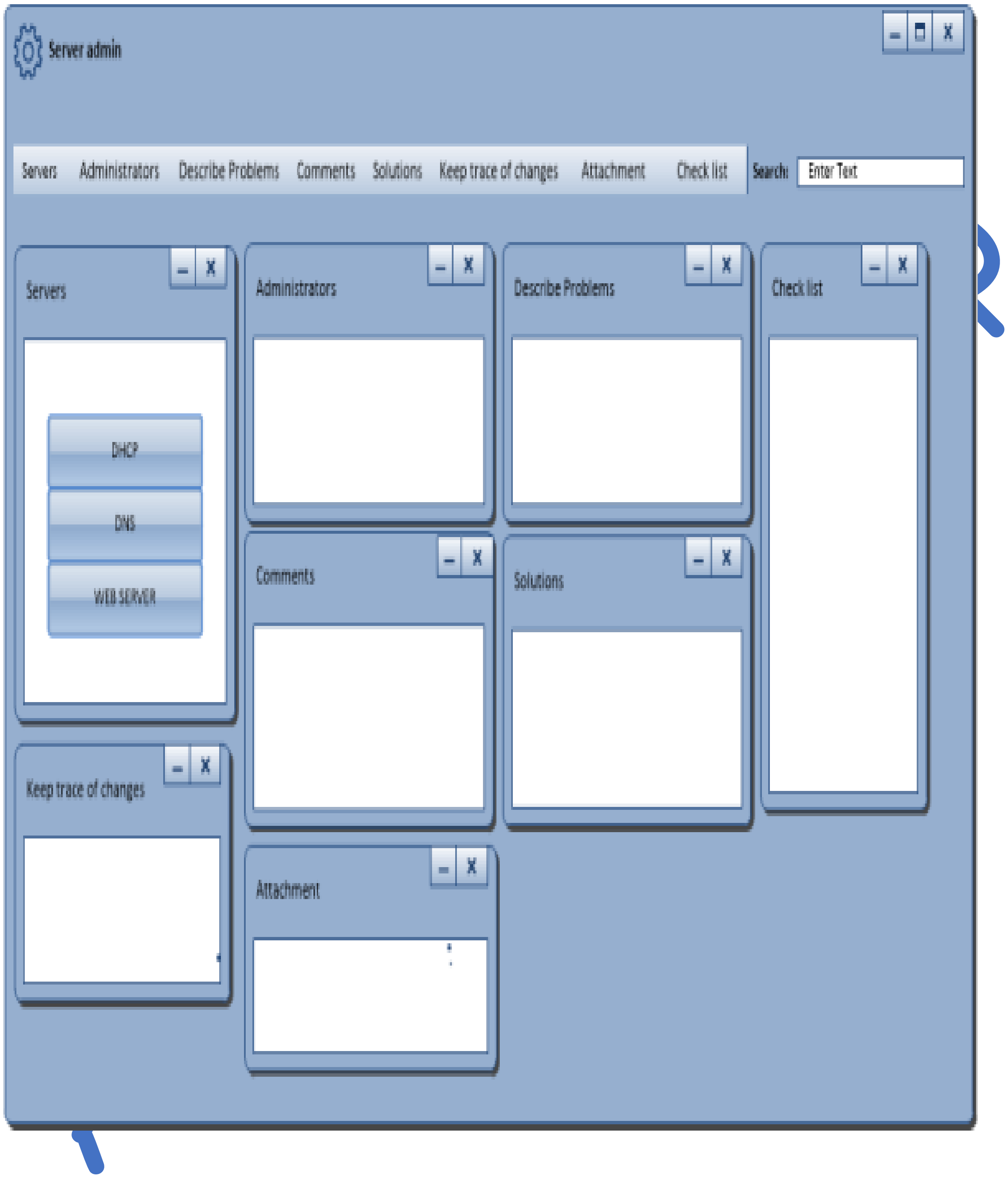
Thomas, K., J, Sicam. 2008. *Beginning Ubuntu Linux*. 3rd ed. Berkeley, Calif: Apress.

Appendices

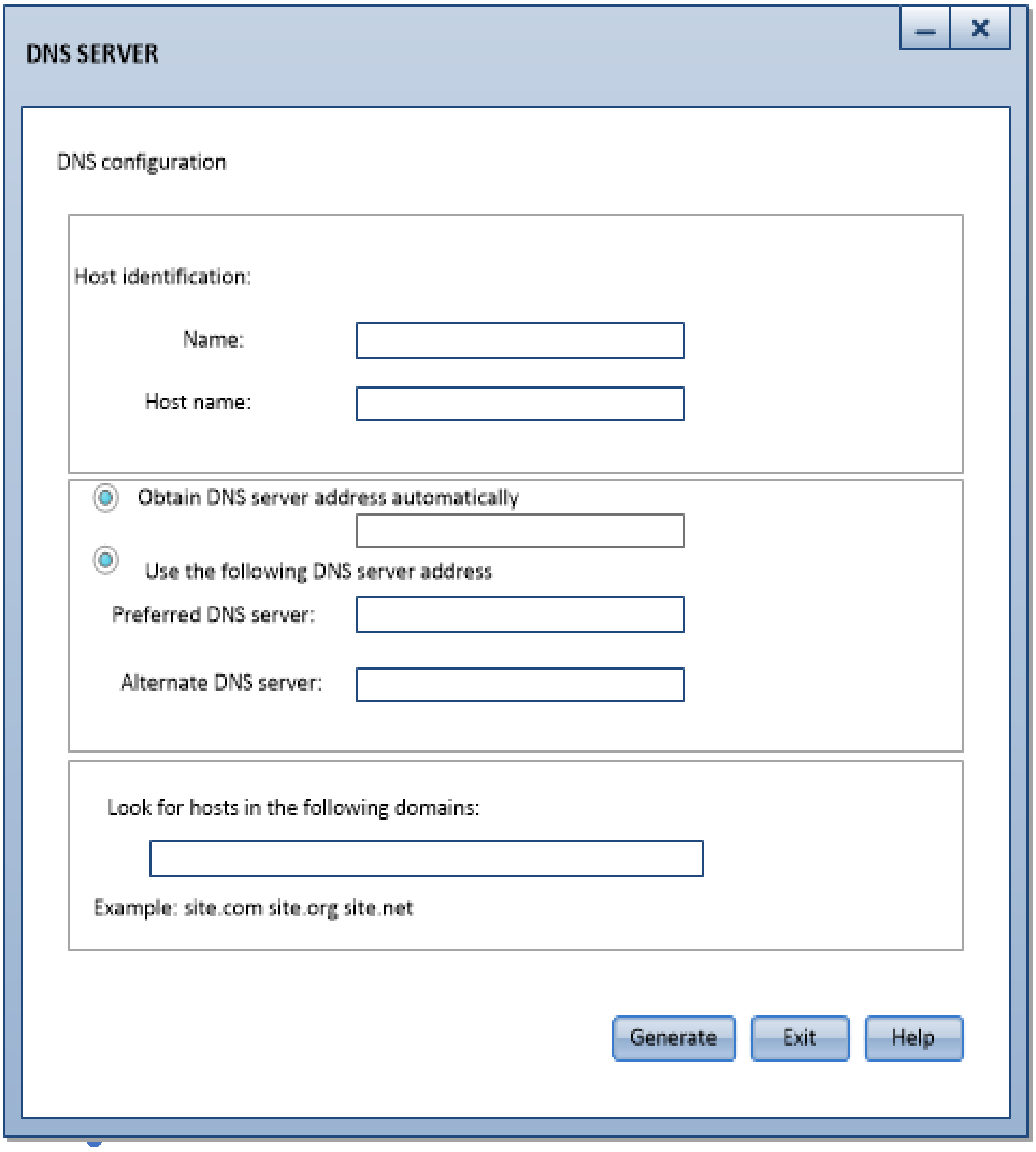
APPENDIX A: Project Plan time and Actual time

Actual Work Plan	**Plan time**	**Actual Time**
Development		
GUI designing	4	6
DNS Server configuration	5	4
DHCP Server configuration	5	3
Login and other sequences	3	5
Testing		
Developer testing	10	11
ACME testing	3	2
Report writing		
Introduction	1	1
Literature Review	10	9
Methodology	5	4
Implementation	5	5
Testing	3	4
Conclusion and Future work	1	1
Proofreading	2	8
Formatting the report	1	2
Printing the report	1	2

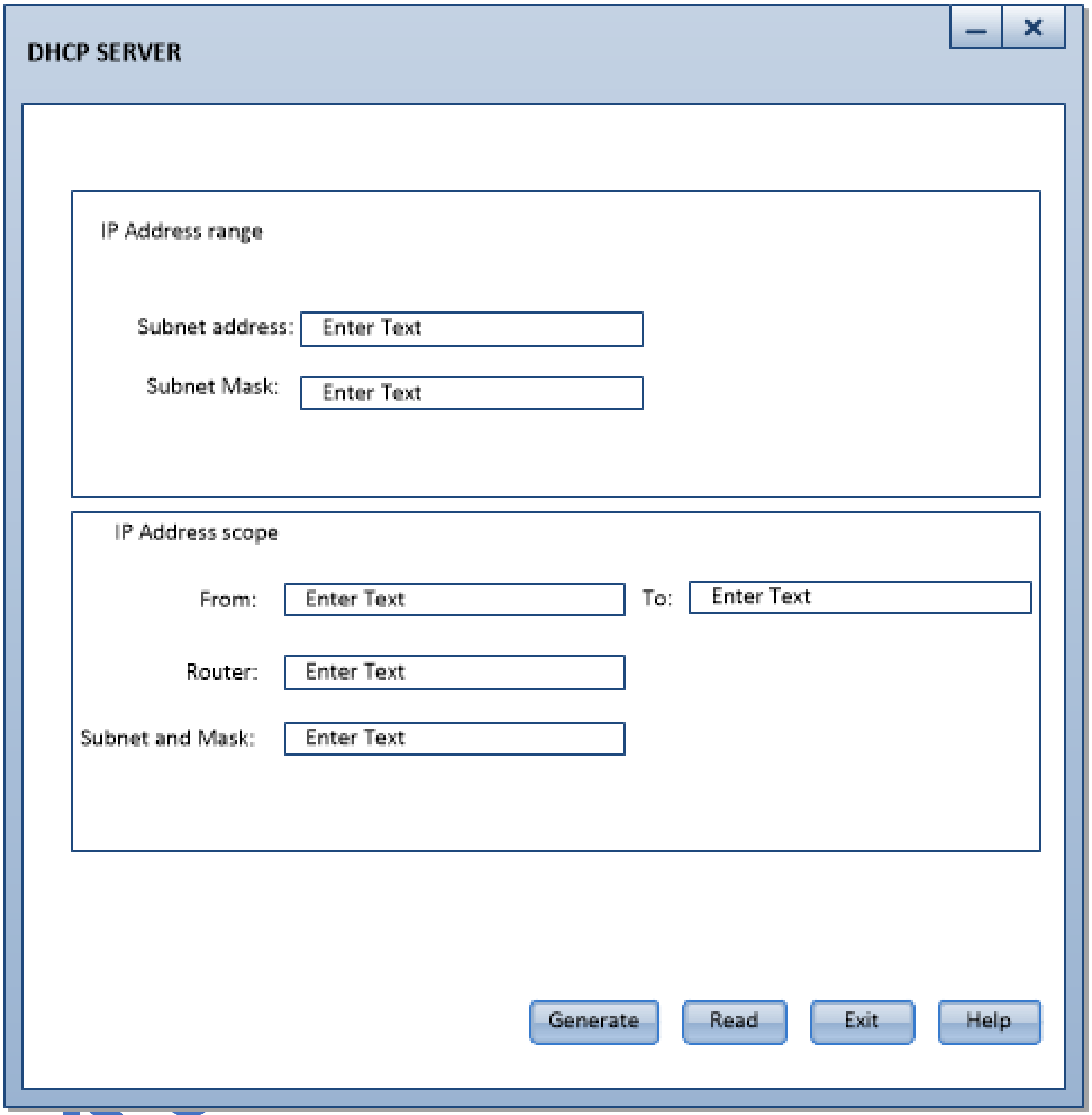
APPENDIX B: Wireframe for Administrator Interface Diagram



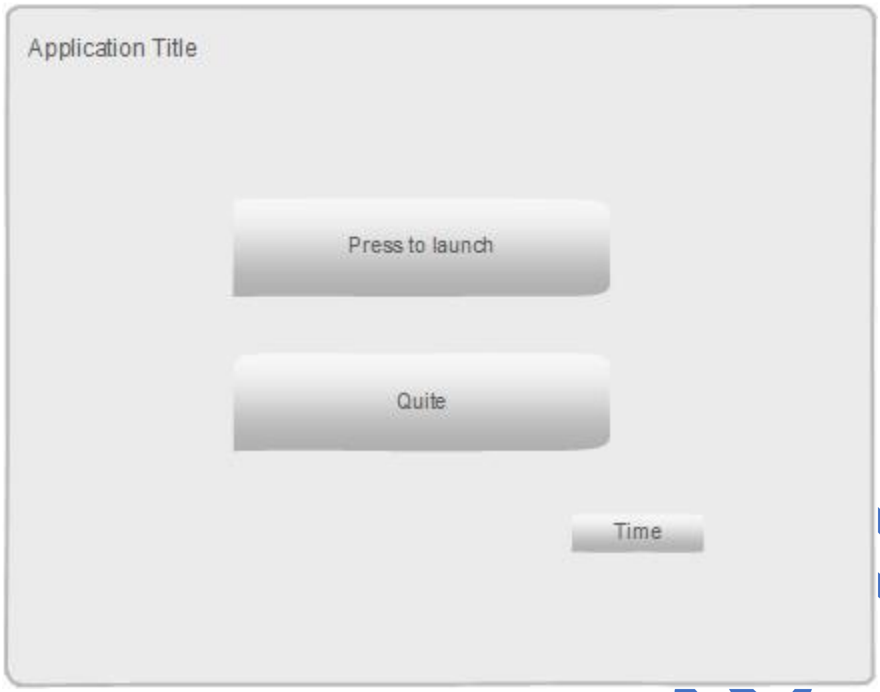
APPENDIX C: Wireframe for DNS


DNS SERVER
DNS configuration
Host identification:
Name:
Host name:
Obtain DNS server address automatically
Use the following DNS server address
Preferred DNS server:
Alternate DNS server:
Look for hosts in the following domains:
Example: site.com site.org site.net
Generate
Exit
Help

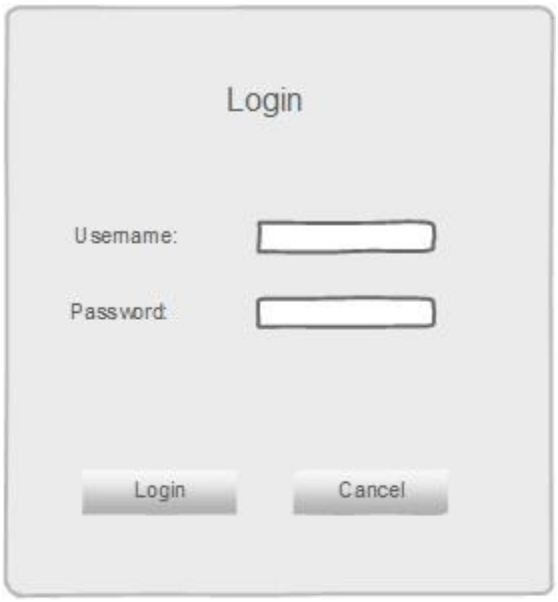
APPENDIX D: Wireframe for DHCP server



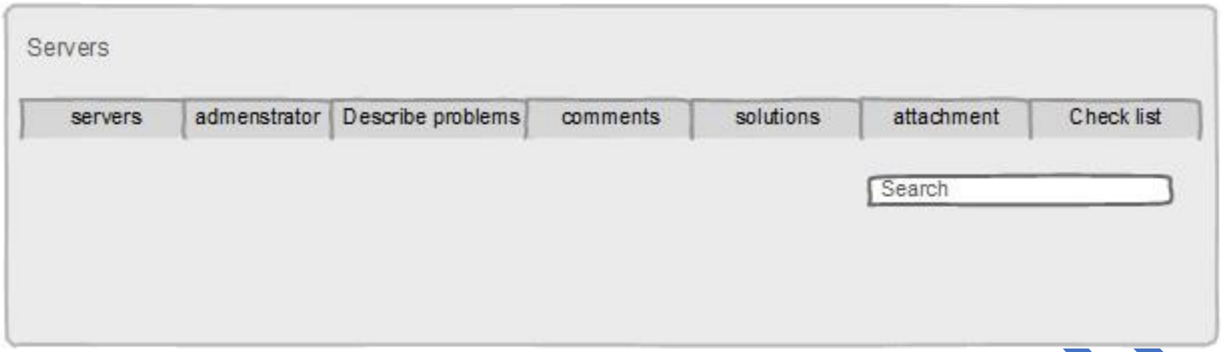
APPENDIX E: Start page



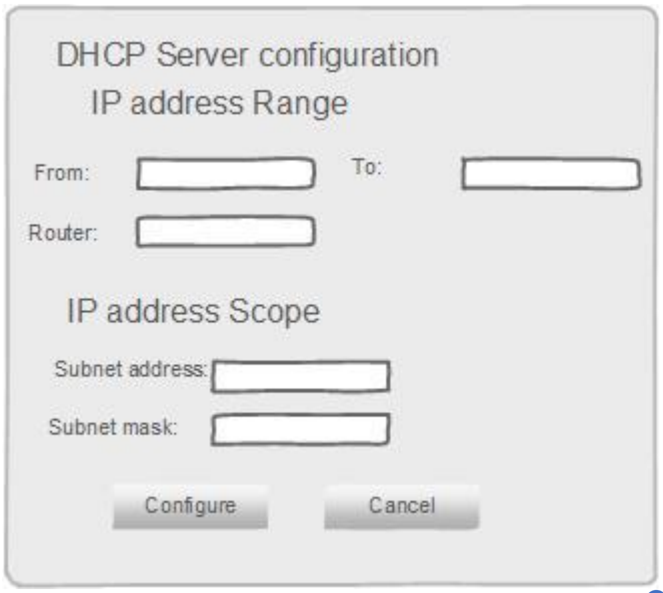
APPENDIX F: Login



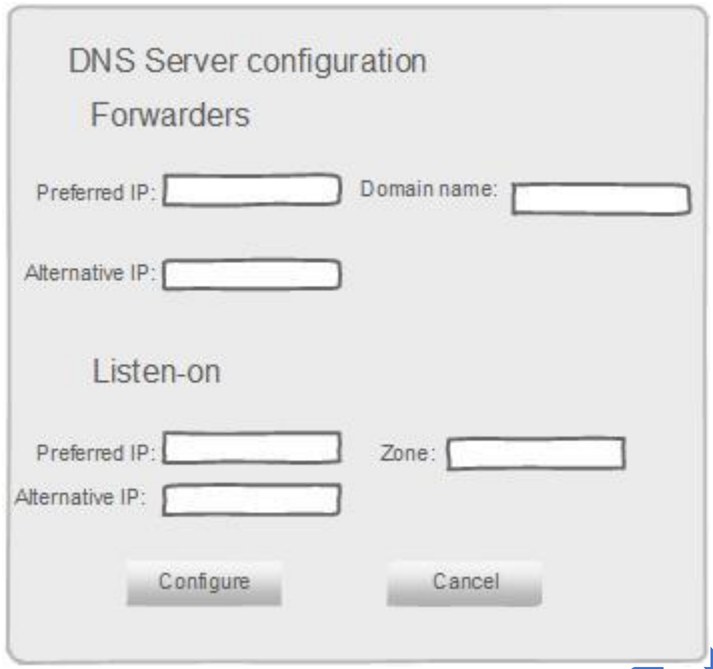
APPENDIX G: Homepage



APPENDIX H: DHCP server configuration page



APPENDIX I: DNS Server configuration page


DNS Server configuration

Forwarders

Preferred IP:		Domain name:	
Alternative IP:			

Listen-on

Preferred IP:		Zone:	
Alternative IP:			

APPENDIX J: Old Gantt chart

APPENDIX K: New Gantt chart

APPENDIX L: dhcp.h file source code

```
#ifndef DHCP_H
#define DHCP_H

#include <QWidget>

namespace Ui {
class dhcp;
}

class dhcp : public QWidget
{
   Q_OBJECT

public:
   explicit dhcp(QWidget *parent = 0);
   ~dhcp();

private slots:
   void on_pushButton_clicked();

   void on_pushButton_2_clicked();
   void showTime3();

private:
   Ui::dhcp *ui;
};

#endif // DHCP_H
```

APPENDIX M: dhcp.cpp file source code

```
#include "dhcp.h"
#include "ui_dhcp.h"
#include <qfile.h>
#include <qtextstream.h>
#include <QMessageBox>
#include "QRect"
#include "QDesktopWidget"
#include <QTimer>
#include <QDateTime>

dhcp::dhcp(QWidget *parent):
    QWidget(parent),
    ui(new Ui::dhcp)
{
    ui->setupUi(this);
    ui ->lineEdit->setFocus();
    QRect position =frameGeometry();
    position.moveCenter(QDesktopWidget().availableGeometry().center());
    move(position.topLeft());
    QTimer *timer3=new QTimer(this);
    connect(timer3 , SIGNAL(timeout()), this, SLOT(showTime3()));
    timer3->start();
}

dhcp::~dhcp()
{
    delete ui;
}

void dhcp::showTime3()
{
    QTime time3 =QTime::currentTime();
    QString time_text3 = time3.toString("hh : mm: ss");
    ui ->label_9->setText(time_text3);
}
void dhcp::on_pushButton_clicked()
{
    QFile file("dhcpd.conf");
    if (!file.open(QIODevice::WriteOnly | QIODevice:: Text))
        return;
    QTextStream ts(&file);
    ts << "subnet " + ui -> lineEdit->text()+ " netmask " + ui->lineEdit_2->text()+" {\n";
    ts << "range " + ui-> lineEdit_3 -> text()+" "+ui->lineEdit_4->text()+";\n";
    ts << "option routers "  + ui->lineEdit_5->text()+";\n";
    ts << "}\n";
    file.close();

    QMessageBox ::information(this, tr("Information"),tr("The DHCP Configuration file is created"));

}

void dhcp::on_pushButton_2_clicked()
{
     this ->close();
}
```

APPENDIX N: dhs.h file source code

```
#ifndef DNS_H
#define DNS_H

#include <QWidget>

namespace Ui {
class DNS;
}

class DNS : public QWidget
{
    Q_OBJECT

public:
    explicit DNS(QWidget *parent = 0);
    ~DNS();

private slots:

    void showTime2();
    void on_pushButton_2_clicked();

    void on_pushButton_3_clicked();

private:
    Ui::DNS *ui;
};

#endif // DNS_H
```

APPENDIX O: dns.cpp file source code

```
#include "dns.h"
#include "ui_dns.h"
#include <qfile.h>
#include <qtextstream.h>
#include <QMessageBox>
#include "QRect"
#include "QDesktopWidget"
#include <QTimer>
#include <QDateTime>
DNS::DNS(QWidget *parent) :
   QWidget(parent),
   ui(new Ui::DNS)
{
   ui->setupUi(this);
   QRect position =frameGeometry();
   position.moveCenter(QDesktopWidget().availableGeometry().center());
   move(position.topLeft());
   QTimer *timer2=new QTimer(this);
   connect(timer2 , SIGNAL(timeout()), this, SLOT(showTime2()));
   timer2->start();
}

DNS::~DNS()
{
   delete ui;
}

void DNS::showTime2()
{
   QTime time2 =QTime::currentTime();
   QString time_text2 = time2.toString("hh : mm: ss");
   ui ->label_8->setText(time_text2);
}

void DNS::on_pushButton_2_clicked()
{
   QFile file("named.conf");
   if (!file.open(QIODevice::WriteOnly | QIODevice:: Text))
      return;
   QTextStream ts(&file);
   ts << "forwarders { " + ui -> lineEdit->text()+ "; " + ui->lineEdit_4->text()+";};\n";
   ts << "listen-on { " + ui-> lineEdit_2 -> text()+"; "+ui->lineEdit_3->text()+";};\n";
   ts << "Domain-name { " + ui->lineEdit_5 -> text()+";};\n";
```

```
    ts << "Zone { " + ui->lineEdit_7-> text()+";};\n";
    ts << "notify no;";
    ts << "};\n";
    file.close();
    QMessageBox ::information(this, tr("Information"),tr("The DNS Configuration file is created"));
}

void DNS::on_pushButton_3_clicked()
{
    this ->close();
}
```