PERSONAL FILE SERVER AND MEDIA CENTER USING RASPBERRY PI

Vincent Meringgai Ak Birang

Bachelor of Computer Science with Honours
(Software Engineering)
2015

PERSONAL FILE SERVER AND MEDIA CENTER USING RASPBERRY PI

VINCENT MERINGGAI AK BIRANG

This project is submitted in partial fulfillment of the
requirements for the degree of
Bachelor of Computer Science and Information Technology

Faculty of Computer Science and Information Technology
UNIVERSITI MALAYSIA SARAWAK
2015

UNIVERSITI MALAYSIA SARAWAK

THESIS STATUS ENDORSEMENT FORM

TITLE PERSONAL FILE SERVER AND MEDIA CENTER USING RASPBERRY PI

ACADEMIC SESSION: 2 2014/2015

VINCENT MERINGGAI AK BIRANG
(CAPITAL LETTERS)

hereby agree that this Thesis* shall be kept at the Centre for Academic Information Services, Universiti Malaysia Sarawak, subject to the following terms and conditions:

1. The Thesis is solely owned by Universiti Malaysia Sarawak
2. The Centre for Academic Information Services is given full rights to produce copies for educational purposes only
3. The Centre for Academic Information Services is given full rights to do digitization in order ro develop local content database
4. The Centre for Academic Information Services is given full rights to produce copies of this Thesis as part of its exchange item program between Higher Learning Institutions [or for the purpose of interlibrary loan between HLI]
5. ** Please tick (√)

☐	CONFIDENTIAL	(Contains classified information bounded by the OFFICIAL SECRETS ACT 1972)
☐	RESTRICTED	(Contains restricted information as dictated by the body or organization where the research was conducted)
✓	UNRESTRICTED	

Validated by

[illegible]
(AUTHOR'S SIGNATURE)

[illegible]
(SUPERVISOR'S SIGNATURE)

Nurfauza Jali
Lecturer
Faculty of Computer Science and Information Technology
UNIVERSITI MALAYSIA SARAWAK

Permanent Address

NO 42, TAMAN PUTRAJAYA,
JALAN TUN HUSSEIN ONN,
97000 BINTULU, SARAWAK

Date: 01/07/2015

Date: 1/7/2015

Note * Thesis refers to PhD, Master, and Bachelor Degree
** For Confidential or Restricted materials, please attach relevant documents from relevant organizations / authorities

Acknowledgement

The author would like to express his utmost gratitude to his Final Year Project (FYP) supervisor, Mdm Nurfauza Jali for her guidance, knowledge and help throughout the period of FYP 1 and FYP 2 semesters. The author is very grateful for her supports, guidance and help in resource acquisitions during the FYP session as well as reviews and feedbacks given for each submission drafts that helped contribute to the completion of this FYP Final Report.

The author is also thankful to his FYP examiners, Mr Ahmad Hadinata Fauzi and Mdm Hazlini Borhan for their feedbacks. The author is grateful for their approval on his FYP Title, Project Description and Project Proposal, especially Mr Ahmad Hadinata for his generous assessment and kind feedbacks for FYP 1 and FYP 2 Final Report. Aside from this, the author is also thankful to Prof. Dr. Wang Yin Chai for his FYP briefings, supports and guidance throughout the FYP session.

The author would also like to thank his friends and fellow colleagues that have helped him directly and indirectly throughout FYP 1 and FYP 2 semesters in term of giving advice, material resources, feedbacks and support. The supports given have been useful and more or less contributed in helping the author to complete this FYP dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Description
1080p30	Term for screen resolution of 1920 x 1080 pixels, with 30 frames per second
AM/FM	Amplitude-Modulation/Frequency Modulation, radio transmission method
API	Application Programming Interface
ARM	Advanced RISC Machines, refers to ARM-architecture of computer CPU
ARM64	ARM-architecture, 64-bit type
BD	Blu-Ray Disc, solid-state storage medium (~25 GB-50 GB storage)
CD	Compact Disc, solid-state storage medium (~700 MB storage)
CIFS	Common Internet File system, a variation of SMB protocol
CPU	Central Processing Unit, the computer component that process instructions
DirectX	A collection of APIs for processing multimedia content
DVD	Digital Video Disc, solid-state storage medium (~4.7 GB storage)
DVR	Digital Video Recorder, a video-recording hardware
FAT32	File Allocation Table, 32-bit type
FreeBSD	Free Berkeley Software Distribution, open-source OS, UNIX-derivative
FRS	File Replication Service
FTP	File Transfer Protocol, data transfer protocol through the Internet
FYP	Final Year Project
GB	Giga Byte unit for data byte size, 1 GB is equivalent to 10^9 bytes
GFLOPS	Giga FLoating-point Operations Per Second, measure of computing power
GPIO	General Purpose Input/Output, refers to the pins on an integrated circuit
GPU	Graphics Processing Unit, the component that process display output
GUI	Graphical User Interface, refers to the user interface of a system/program
H.264	Advanced Video Coding (MPEG-4 AVC), a video compression format
HDD	Hard-Disk Drive
HDMI	High-Definition Multimedia Interface
HTML	Hyper-Text Markup Language, a scripting language for a webpage
HTPC	Home Theater Personal Computer
HTTP	Hyper-Text Transfer Protocol, data transfer protocol for the Web

I2c	Inter-Integrated Circuit, a serial bus for motherboard/embedded systems
IBM	International Business Machine Corporation, hardware/software manufacturer
LAMP	Linux-Apache-MySQL-PHP software bundle, for LAMP web server
LAN	Local Area Network
LCD	Liquid Crystal Display, a type of video display used on display screen
LED	Light-Emitting Diode
LTS	Long-Tenn Support, extra feature for Ubuntu distribution version
MB	Mega Byte, unit for data byte size, 1 MB is equivalent to 10^6 bytes
MHz	Mega Hertz, frequency measurement unit, 1 MHz is equivalent to 10^6 Hertz
MKV	The extension for Matroska Multimedia Container media format
MMC	MultiMedia Card, a type of solid-state storage medium
MP4	The extension for MPEG-4 media format
NAS	Network-Attached Storage
NTFS	New Technology File System
NVIDIA	A trademark name of a manufacturer of GPUs
OpenGL	Open Graphics Library, multi-platform API for rendering 2D/3D graphics
OpenVG	Open Video Graphics, cross-platform API, provides hardware acceleration
OpenVMS	Open Virtual Memory System, a type of server operating system
OS	Operating System
OS/2	Operating System/2, a type of computer OS developed by Microsoft and IBM
PC	Personal Computer
PHP	Hypertext Preprocessor, a scripting language like HTML
POWER	Refers to a type of architecture for computer processors, like x86 and ARM
RAID	Redundant Array of Independent Disks, a type of data storage technology
RAM	Random-Access Memory, main memory for computer system
RCA	Radio Corporation of America, electrical connector for audio/video signals
SD	Secure Digital, refers to memory card, a solid-state storage medium
SDIO	Secure Digital Input Output, extension of SD specification for I/O functions
SDRAM	Synchronous Dynamic Random Access Memory, a type of RAM
SMB	Server Message Block, a type of network protocol for shared access

SoC	System on Chip, integrated circuit of computer components and electronics
SPI	Serial Peripheral Interface
TV	Television
UART	Universal Asynchronous Receiver/Transmitter
UI	User Interface
UK	United Kingdom
UNIX	Originally UNICS, UNiplexed Information & Computing Service, type of OS
UPnP	Standard Universal Plug-and-Play
USB	Universal Serial Bus, a communication protocol of with electronic device
USD	United States Dollar
V	Voltage, measurement unit for electric current
WHS	Windows Home Server
Wi-Fi	A type of local area wireless technology
Win7	Windows 7, a type of x86-based computer OS by Microsoft
Win8	Windows 8, a type of x86-based computer OS by Microsoft
WinXP	Windows XP, a type of x86-based computer OS by Microsoft
WMC	Windows Media Center
x86	Term for x86-based architecture of Intel-based CPU
XBMC	Xbox Media Center, a type of media center software
XBOX	A type of gaming console owned by Microsoft
ZFS	Zettabyte File System, a type of file system like FAT32 or NTFS

ABSTRACT

Conventional file sharing and media viewing usually involves tedious file transfer. This project aims to provide access convenience for sharing data and direct-viewing of media files by having file server (FS) and media center (MC) capability using Raspberry Pi. The system consists of several functionalities that were developed through iterative and incremental development. The resulting system has its FS function catered by Samba program and its MC function catered by Kodi program. Direct MC output is on high-definition television (HDTV).

The web-based user interface (WebUI) provides administrative functions for the system, its FS and users management and indirect access to its MC function through web player for all registered users. The system has undergone several testing processes and it is a working prototype of an economical and feasible file server and media center using Raspberry Pi. The system can still be improved with other functions and features in the future.

ABSTRAK

Perkongsian fail secara konvensional dan tontonan media biasanya melibatkan kerja berlebihan semasa pemindahan fail tersebut. Projek ini bertujuan untuk menyediakan kemudahan akses untuk perkongsian data dan tontonan langsung bagi fail media dengan mempunyai pelayan fail dan pusat media dengan menggunakan Raspberry Pi. Sistem ini terdiri daripada beberapa fungsi yang telah dibangunkan melalui pembangunan lelaran dan tambahan. Sistem yang dihasilkan mempunyai fungsi pelayan fail yang disediakan oleh program Samba dan fungsi pusat media yang disediakan oleh program Kodi. Output pusat media disiarkan secara langsung di televisyen beresolusi tinggi (HDTV).

Manakala antaramuka untuk penggunanya yang berasaskan web (WebUI) menyediakan fungsi pentadbiran untuk sistem itu, pelayan failnya dan pengurusan penggunanya dan juga memberi akses tidak langsung kepada fungsi pusat media melalui pemain web untuk semua pengguna yang berdaftar. Sistem ini telah melalui beberapa proses ujian dan ia adalah satu prototaip yang berfungsi, terdiri daripada pelayan fail dan media pusat yang ekonomi dan boleh dilaksanakan dengan menggunakan Raspberry Pi. Sistem ini juga masih boleh ditambah dengan fungsi dan ciri-ciri lain pada masa akan datang.

1 INTRODUCTION

1.1 Introduction

In the modern world, data and media are mostly in digital form. Nowadays, data storing has become more crucial and significant. Data storage has also steadily evolved over time from the early usage of diskettes and compact-discs (CDs) and due to the constant increase of data size especially media files, there was always a constant need for bigger size storage. This is why the main storage later has evolved to Digital Video Discs (DVDs) and Blu-Ray Discs (BDs), and USB flash drives to external Hard-Disk Drives (HDDs). While storing data externally in bigger size storage eases the huge file size problem, it however incorporates a tedious way of accessing them.

The conventional way of storing data and media files usually include several distributed electronic devices for example documents are stored in personal computers (PCs) while huge size media files and backup data are stored in external HDDs. To obtain a desired file, one may have to connect their external storage devices to a PC and this may involve time consuming copy-and-paste process. To view or watch a media file, one may have to transfer it into another external storage device such as flash drive or SD card. Then, one need to connect it to media output devices such as TV, monitor, speakers and media player. All these can be quite annoying if one need those files on a frequent basis.

1.2 Problem Statement

The typical solution to solve the problem with access convenience is to use a commercial file server, which usually has very. large storage size to store all the files in one place. However, the current value of a commercial file server is quite expensive and thus many still prefer the conventional way of managing their files. Plus, a commercial file server may have limited usage and capability – as a server to host and share files readily to be accessed. It may lack on the feature to view them directly on media output devices due to the lack of media player or codec support.

This means that one still needs a PC that is connected to the file server in order to access the stored files. Moreover, the PC may need a proper program to open and view the content of the files, provided that the file server allows direct access. This can be even more problematic if certain media files needing certain codecs in order to play or view them. And while a PC may have all the proper program and codecs required to view any files stored, one may still want to view them on a bigger flat-screen TV. So, one still need to go through the time consuming process of copy-and-paste especially for huge file size like video files. Furthermore, one later need to plug them into a media player or TV in order to view them.

Thus, there is a need to have an alternative solution to bypass the tedious process and at the same time with much less cost if possible. And this project proposes to build a cheaper and working prototype of a file server that also has a media player capability. Furthermore, it will be built using Raspberry Pi, which gives it potential to be enhanced with additional useful features for more convenience.

1.3 Objectives

The objectives of this project are as listed below:

a) To create a file server which host both data (documents, executables, generic files) and media files (audio, video, image files) using Raspberry Pi.

b) To equip the file server with media center capability by installing a media center program and connect it with media output devices (monitor/TV/speakers).

c) To build a main system interface (web-based) for users to easily interact and use the functions.

1.4 Methodology

The methodology to be used to develop the file server and media center using Raspberry Pi will be based on the incremental development. This methodology is chosen mainly because the final product will have several functionalities that need to be delivered separately and added incrementally. It is proposed that it will start with very basic functions of a file server and later basic media player capability as they are the highest priority functions to be implemented. The development of the main system interface will also use this methodology as the functions on the web-based interface will be added incrementally.

Once these main two functions are working correctly, the system will be shown and used by user and based on the feedback, the functions will be modified if necessary and enhanced with several useful features. Other new functions with their own features will then be added incrementally based on priority. The addition of the functions and features will involve a series of concurrent activities.

Architecturally, the Raspberry Pi will be physically connected with several hardware, input and output devices. It will be the central unit that controls all the hardware and devices. The main operating system for the Raspberry Pi will be of Debian Linux distribution. As for the technical details in developing the functions and features, basic programs that are relevant to file server and media player functionalities will be used. These programs later will be modified wherever necessary and depending on requirements for other functions and features, new programs may need to be developed solely to accommodate them.

1.5 Scope

The file server and media center will have at least two main functions: file server capability to host and share stored files; and media player capability to access media files directly. Other functions and its features are to be determined later based on user request and feedback during requirement analysis phase and testing.

The intended user for the system is anyone with intention to use it as a file server and at the same time may use it to access, view content or play media files stored in the server directly on media output devices connected to it. Thus, the system may not have or has very limited capability (if any) in modifying or editing the content of stored data or media files without proper programs.

Other limitation is that the proposed file server and media center is intended for personal use. It will be initially connected only within a local private network and thus, the files stored may only be accessed and shared with those connected to the same network. This also means that the file server feature is intended to cater for small number of clients in a home network.

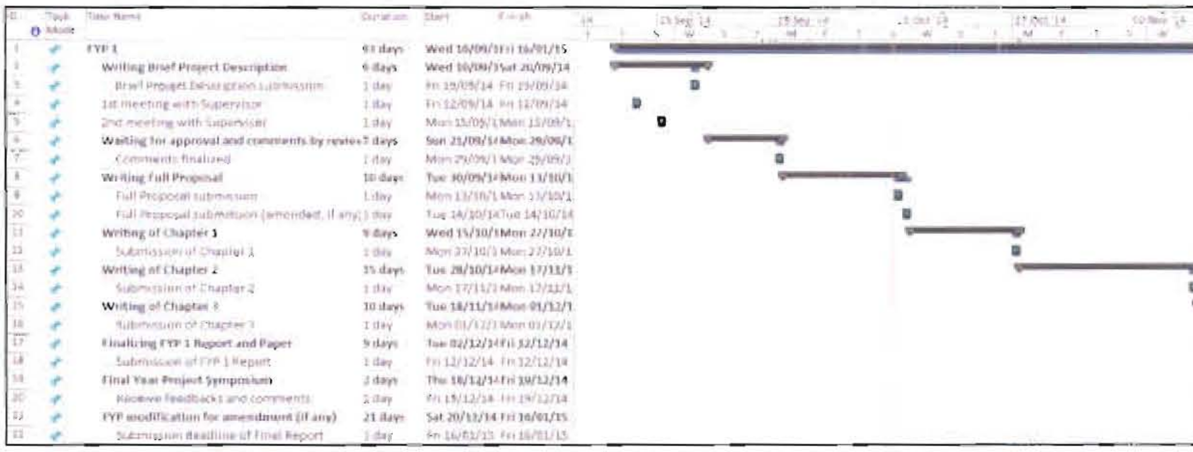
1.6 Significance of Project

The proposed personal file server and media center will be a working prototype of a personal file server equipped with media player capability. This means that the system will host data and share them. At the same time, user can also access and view the content or play stored media files directly on media output devices the system is connected to, without the need to transfer them to be played on other external media player. It will also be a model prototype of an affordable file server for personal use due to the relatively much cheaper Raspberry Pi.

The fact that it will use Raspberry Pi as a platform will further display the wide potential and versatility of the small piece of electronic hardware. This device can be configured as the central of innovative and creative products. While the proposed system is intended to be developed for personal use, its concept and architecture may be modified and used in small-scale business or even educational purposes in learning facilities such as school.

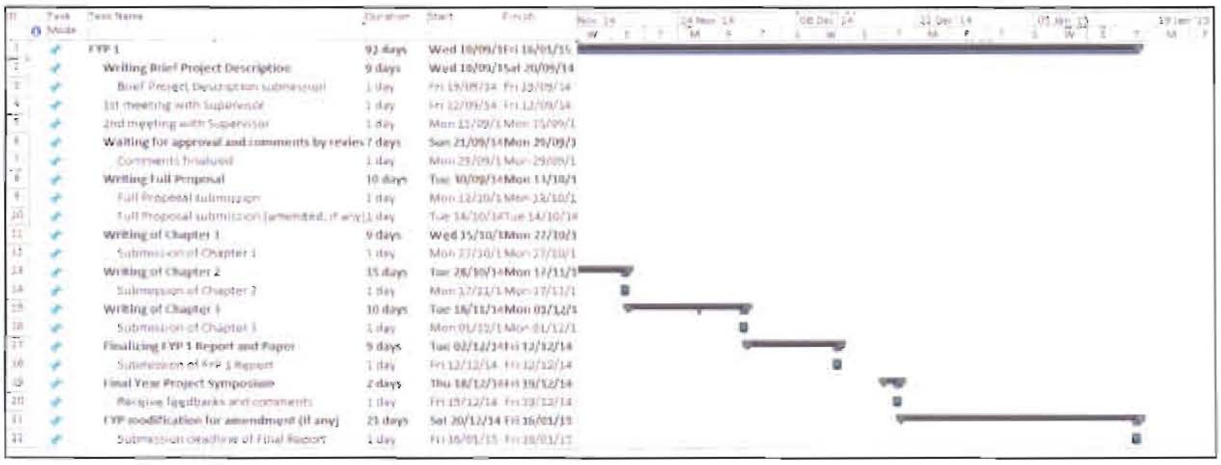
1.7 Project Schedule

Below are the project schedules for this proposed project during FYP 1 and FYP 2 semesters.



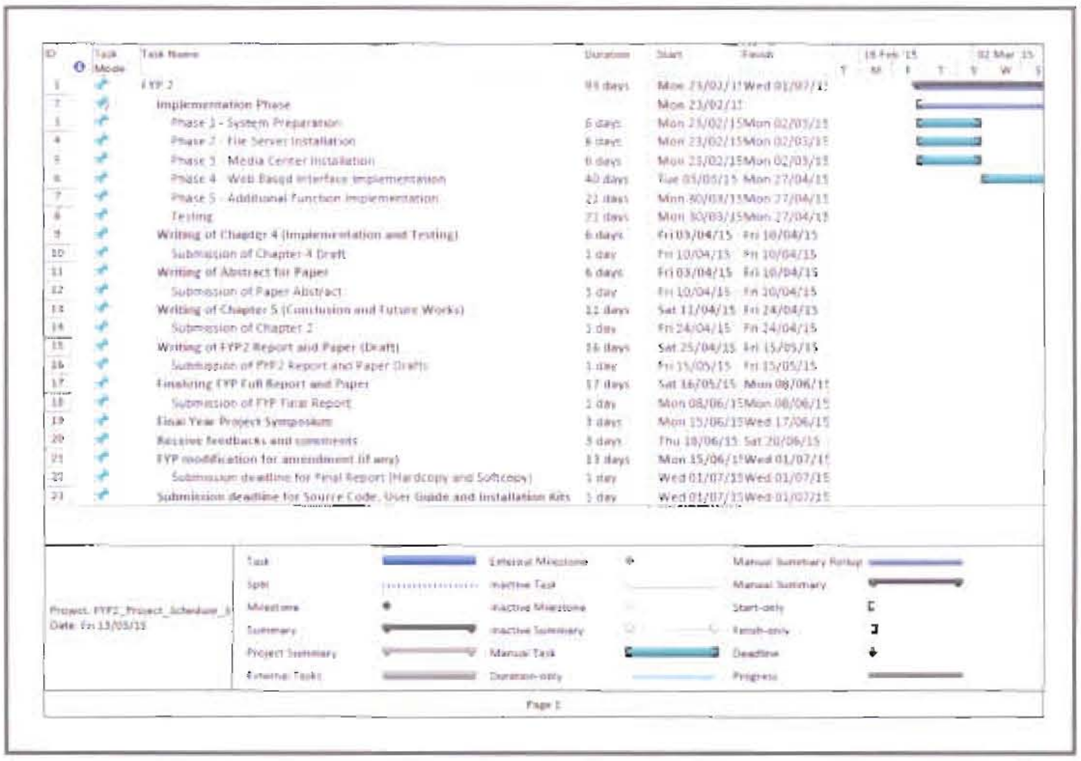
ID	Task Mode	Task Name	Duration	Start	Finish
1		FYP 1	93 days	Wed 10/09/14	Fri 16/01/15
2		Writing Brief Project Description	6 days	Wed 10/09/1	Sat 20/09/14
3		Brief Project Description submission	1 day	Fri 19/09/14	Fri 19/09/14
4		1st meeting with Supervisor	1 day	Fri 12/09/14	Fri 12/09/14
5		2nd meeting with Supervisor	1 day	Mon 15/09/1	Mon 15/09/1
6		Waiting for approval and comments by review	7 days	Sun 21/09/14	Mon 29/09/1
7		Comments finalized	1 day	Mon 29/09/1	Mon 29/09/1
8		Writing Full Proposal	10 days	Tue 30/09/14	Mon 13/10/1
9		Full Proposal submission	1 day	Mon 13/10/1	Mon 13/10/1
10		Full Proposal submission (amended, if any)	1 day	Tue 14/10/14	Tue 14/10/14
11		Writing of Chapter 1	9 days	Wed 15/10/1	Mon 27/10/1
12		Submission of Chapter 1	1 day	Mon 27/10/1	Mon 27/10/1
13		Writing of Chapter 2	15 days	Tue 28/10/14	Mon 17/11/1
14		Submission of Chapter 2	1 day	Mon 17/11/1	Mon 17/11/1
15		Writing of Chapter 3	10 days	Tue 18/11/14	Mon 01/12/1
16		Submission of Chapter 3	1 day	Mon 01/12/1	Mon 01/12/1
17		Finalizing FYP 1 Report and Paper	9 days	Tue 02/12/14	Fri 12/12/14
18		Submission of FYP 1 Report	1 day	Fri 12/12/14	Fri 12/12/14
19		Final Year Project Symposium	2 days	Thu 18/12/14	Fri 19/12/14
20		Receive feedbacks and comments	1 day	Fri 19/12/14	Fri 19/12/14
21		FYP modification for amendment (if any)	21 days	Sat 20/12/14	Fri 16/01/15
22		Submission deadline of Final Report	1 day	Fri 16/01/15	Fri 16/01/15

Figure 1.1: Project schedule for FYP 1 with Gantt Chart (part A).



ID	Task Mode	Task Name	Duration	Start	Finish
1	✓	FYP 1	93 days	Wed 10/09/1	Fri 16/01/15
2	✓	Writing Brief Project Description	9 days	Wed 10/09/1	Sat 20/09/14
3	✓	Brief Project Description submission	1 day	Fri 19/09/14	Fri 19/09/14
4	✓	1st meeting with Supervisor	1 day	Fri 12/09/14	Fri 12/09/14
5	✓	2nd meeting with Supervisor	1 day	Mon 15/09/1	Mon 15/09/1
6	✓	Waiting for approval and comments by review	7 days	Sun 21/09/14	Mon 29/09/1
7	✓	Comments finalised	1 day	Mon 29/09/1	Mon 29/09/1
8	✓	Writing Full Proposal	10 days	Tue 30/09/14	Mon 13/10/1
9	✓	Full Proposal submission	1 day	Mon 13/10/1	Mon 13/10/1
10	✓	Full Proposal submission (amended, if any)	1 day	Tue 14/10/14	Tue 14/10/14
11	✓	Writing of Chapter 1	9 days	Wed 15/10/1	Mon 27/10/1
12	✓	Submission of Chapter 1	1 day	Mon 27/10/1	Mon 27/10/1
13	✓	Writing of Chapter 2	15 days	Tue 28/10/14	Mon 17/11/1
14	✓	Submission of Chapter 2	1 day	Mon 17/11/1	Mon 17/11/1
15	✓	Writing of Chapter 3	10 days	Tue 18/11/14	Mon 01/12/1
16	✓	Submission of Chapter 3	1 day	Mon 01/12/1	Mon 01/12/1
17	✓	Finalizing FYP 1 Report and Paper	9 days	Tue 02/12/14	Fri 12/12/14
18	✓	Submission of FYP 1 Report	1 day	Fri 12/12/14	Fri 12/12/14
19	✓	Final Year Project Symposium	2 days	Thu 18/12/14	Fri 19/12/14
20	✓	Receive feedbacks and comments	1 day	Fri 19/12/14	Fri 19/12/14
21	✓	FYP modification for amendment (if any)	21 days	Sat 20/12/14	Fri 16/01/15
22	✓	Submission deadline of Final Report	1 day	Fri 16/01/15	Fri 16/01/15

Figure 1.2 : Project schedule for FYP 1 with Gantt Chart (part B).



ID	Task Mode	Task Name	Duration	Start	Finish
1	✓	FYP 2	93 days	Mon 23/02/1	Wed 01/07/1
2	✓	Implementation Phase		Mon 23/02/1	
3	✓	Phase 1 - System Preparation	6 days	Mon 23/02/15	Mon 02/03/15
4	✓	Phase 2 - File Server Installation	6 days	Mon 23/02/15	Mon 02/03/15
5	✓	Phase 3 - Media Center Installation	6 days	Mon 23/02/15	Mon 02/03/15
6	✓	Phase 4 - Web Based Interface Implementation	40 days	Tue 03/03/15	Mon 27/04/15
7	✓	Phase 5 - Additional Function Implementation	21 days	Mon 30/03/15	Mon 27/04/15
8	✓	Testing	21 days	Mon 30/03/15	Mon 27/04/15
9	✓	Writing of Chapter 4 (Implementation and Testing)	6 days	Fri 03/04/15	Fri 10/04/15
10	✓	Submission of Chapter 4 Draft	1 day	Fri 10/04/15	Fri 10/04/15
11	✓	Writing of Abstract for Paper	6 days	Fri 03/04/15	Fri 10/04/15
12	✓	Submission of Paper Abstract	1 day	Fri 10/04/15	Fri 10/04/15
13	✓	Writing of Chapter 5 (Conclusion and Future Works)	11 days	Sat 11/04/15	Fri 24/04/15
14	✓	Submission of Chapter 2	1 day	Fri 24/04/15	Fri 24/04/15
15	✓	Writing of FYP2 Report and Paper (Draft)	16 days	Sat 25/04/15	Fri 15/05/15
16	✓	Submission of FYP2 Report and Paper Drafts	1 day	Fri 15/05/15	Fri 15/05/15
17	✓	Finalizing FYP Full Report and Paper	17 days	Sat 16/05/15	Mon 08/06/15
18	✓	Submission of FYP Final Report	1 day	Mon 08/06/15	Mon 08/06/15
19	✓	Final Year Project Symposium	3 days	Mon 15/06/15	Wed 17/06/15
20	✓	Receive feedbacks and comments	3 days	Thu 18/06/15	Sat 20/06/15
21	✓	FYP modification for amendment (if any)	13 days	Mon 15/06/1	Wed 01/07/1
22	✓	Submission deadline for Final Report (Hardcopy and Softcopy)	1 day	Wed 01/07/15	Wed 01/07/15
23	✓	Submission deadline for Source Code, User Guide and Installation Kits	1 day	Wed 01/07/15	Wed 01/07/15

Project: FYP2_Project_Schedule_1
Date: Fri 13/05/15

Task		External Milestone	◆	Manual Summary Rollup	
Split		Inactive Task		Manual Summary	
Milestone	◆	Inactive Milestone		Start-only	[
Summary		Inactive Summary		Finish-only	]
Project Summary		Manual Task		Deadline	⬇
External Tasks		Duration-only		Progress	

Page 1

Figure 1.3: Project schedule for FYP 2 with Gantt Chart (part A).

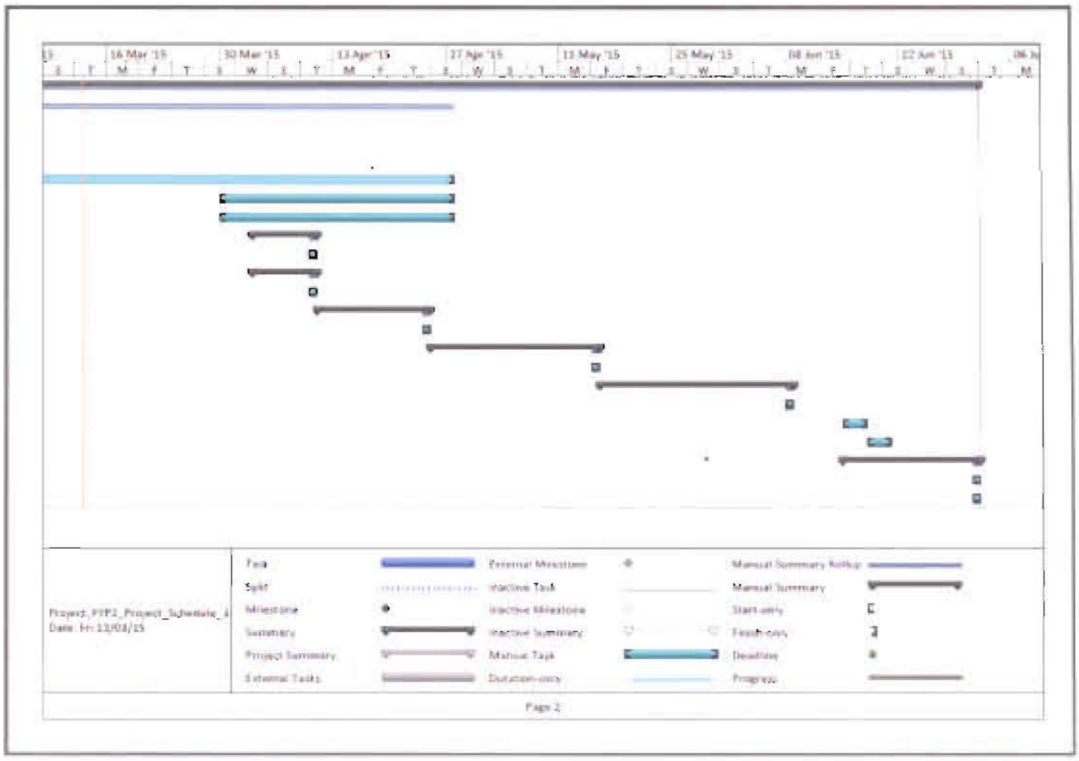


Figure 1.4: Project schedule for FYP 2 with Gantt Chart (part B).

1.8 Expected Outcome

Expected outcome of this project will be a working prototype of personal file server and media center that hosts both data (documents, executables, generic files) and media files (audio, video, image files) using Raspberry Pi. It will be connected to media output devices for direct file and media viewing. This system will be low cost and economical than typical commercial file server and may be improved with additional useful features. This file server and media center is intended for personal usage, targeted mainly for home users. But its concept and implementation can also be applied for small business and also for education purposes in learning facilities such as school.

This personal file server will also have file viewer and media player capability along with additional codecs for support of various media formats. This file server will also be configured to have direct connection with media output devices which means no more tedious transfer of media files using other external storage to view them.

Aside from that, a main system interface (web-based interface) will be developed for users to interact with the system functions, mainly the file server functionalities. Its lower power consumption may enable this system to run 24/7. Further modifications may be done to include useful additional features. An example of basic architecture for this file server and media center is as shown below:

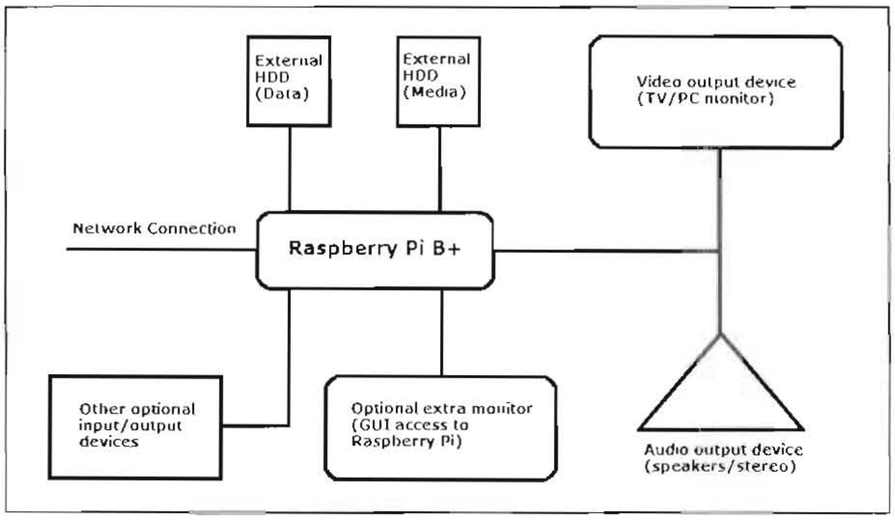

External HDD (Data)
External HDD (Media)
Video output device (TV/PC monitor)
Network Connection
Raspberry Pi B+
Other optional input/output devices
Optional extra monitor (GUI access to Raspberry Pi)
Audio output device (speakers/stereo)

Figure 1.5: Basic architecture of the proposed system using Raspberry Pi.

1.9 Summary

This chapter has introduced to reader this project to develop a *Personal File Server and Media Center using Raspberry Pi*. The system proposed for this project aims is to solve the problem with access convenience for sharing data and direct-viewing of media files by having both the file server capability and media center capability. The proposed methodology chosen to develop the system is incremental development since the system will have several functionalities and features that need to be added incrementally. The resulting system will cater for the need of personal use and within a private home network. However, its implementation concept and architecture may be modified and used in small business and educational facilities.

2 LITERATURE REVIEW

2.1 Introduction

This chapter contains the background and technical review of this project. It consists of several subsections that describe in details about this project background, the Raspberry Pi device, the file server technology and the media center technology. This chapter also contains reviews and analysis on the existing technologies that will be used to realize the proposed system. Each review subsection focus mainly on the concept, technology and implementation details of the existing technologies and also includes the comparisons between them.

2.2 Overview of Project Background

This idea of creating a file server and media center by using Raspberry Pi device emerged from the introduction towards the Raspberry Pi device itself. Upon learning of what it is; its relatively-new implementation, tiny physicality, portability, its purpose and its surprisingly cheap cost – it is discovered that the device has a wide range of potential usage and capability based on the large number of existing projects done by using it.

Among the existing projects, it is proven that the device is capable to be modified and configured to create a file server. Coincidentally, the author has a longing need for a cheap and economic file server mainly to use it to access media files and share them with other computers in a private network. And the idea to use Raspberry Pi device to create a file server emerged naturally given the advantages and viability. But it is later decided that it would be more convenient for the file server to also include the capability as a media center. After all, there are also other projects that use the Raspberry Pi device to create a media center. Finally, all these gave birth to the idea to create a file server and media center by using the Raspberry Pi device.

The details of the Raspberry Pi device and the underlying technology of file server and media center are discussed in the next three subsections. All subsections contain the review and analysis on the existing technologies for each subsection.

2.3 Overview of Raspberry Pi Device

2.3.1 Introduction to Raspberry Pi

As described in *FAQs* (n.d.) page on its official website, the www.raspberrypi.org – the Raspberry Pi device is essentially a low-cost, credit-card sized computer that plugs into a computer monitor or TV. It is basically a small single-board computer that also uses standard keyboard and mouse. Despite its tiny size, it has a wide range of potential usage in various kinds of electronic projects.

Like a normal computer, it has the basic capabilities of a desktop PC ranging from functionality to browse the Internet, making spreadsheets, word-processing, playing games and even capable to play high-definition video (*What is a Raspberry Pi*, n.d.). Due to its cheap cost and yet capable and portable, the device has become increasingly popular worldwide and being used in various electronic projects.

2.3.2 Brief History of Raspberry Pi

The Raspberry Pi device was developed in the UK under a registered educational charity organization – the Raspberry Pi Foundation, which main goal is to "advance the education of adults and children, particularly in the field of computers, computer science and related subjects" (*What is a Raspberry Pi*, n.d.). The idea to create a tiny and affordable computer for kids emerged in 2006 among the colleagues in University of Cambridge's Computer Laboratory and several prototypes were created between 2006 to 2008 (*About Us*, n.d.).

Following the announcement on the official Raspberry Pi website on February 29th in 2012, the Raspberry Pi model A and model B became the first models available on general sales with price of just £22 (35 USD) for model B, with a promise of a cheaper version (model A) of just £16 (25 USD) later that year (Cellan-Jones, 2012). After more than two years, on July 14th, 2014, Eben Upton – the Raspberry Pi founder, officially announced the release of the Raspberry Pi model B+, an improved version of its predecessor but still at the same price of 35 USD (Upton, 2014a). And later on November 10th, 2014, he announced the official release of the Raspberry Pi model A+, an improved version of model A at a low price of just 20 USD (Upton, 2014b).

2.3.3 Technical Specification

Physically, a Raspberry Pi device has the dimension of 85.60mm x 56mm x 21mm (or roughly 3.37" x 2.21" x 0.83") and weighs just 45g (*FAQs*, n.d.). It utilizes a System on a Chip (SoC) of Broadcom BCM2835, which contains RAM of 256MB (for model A), 512MB (for model B), 700 MHz ARM1176JZFS CPU and a VideoCore IV GPU capable of Blu-Ray quality playback with H.264 at 40Mb/s. The board of a Raspberry Pi device typically has "26 dedicated GPIO pins, including a UART, an i2c bus,, a SPI bus with two chip selects, i2s audio, 3v3, 5v, and ground" (*FAQs*, n.d.).

In term of performance, the GPU provides OpenGL ES 2.0 with 24 GFLOPS computing capability, hardware-accelerated OpenVG, and 1080p30 H.264 with high-profile encode and decode capability. By comparison, it has roughly equivalent graphics capabilities of an Xbox and computing power equivalent to 300 MHz Pentium 2 processor (*FAQs*, n.d.). The capabilities are quite outstanding for such a small, cheap and capable piece of an electronic device.

Like a typical computer, it has the typical peripherals, plugs and ports attached on it for connectivity with various other hardware and devices. Essential ones are the GPIO pins, RCA video plug, 3.5mm Audio jack, LEDs, ports of USB, LAN, HDMI, SD card slot and its power plug as illustrated by Raspberry Pi model B in below. The device also has a camera port for connectivity for a camera up to 5MP resolution. In term of playback, it can record and play media files of H.264 containers (MP4/MKV) by default and through the HDMI, it supports high quality video and sound for connectivity with display devices with supporting HDMI feature (*FAQs*, n.d.).

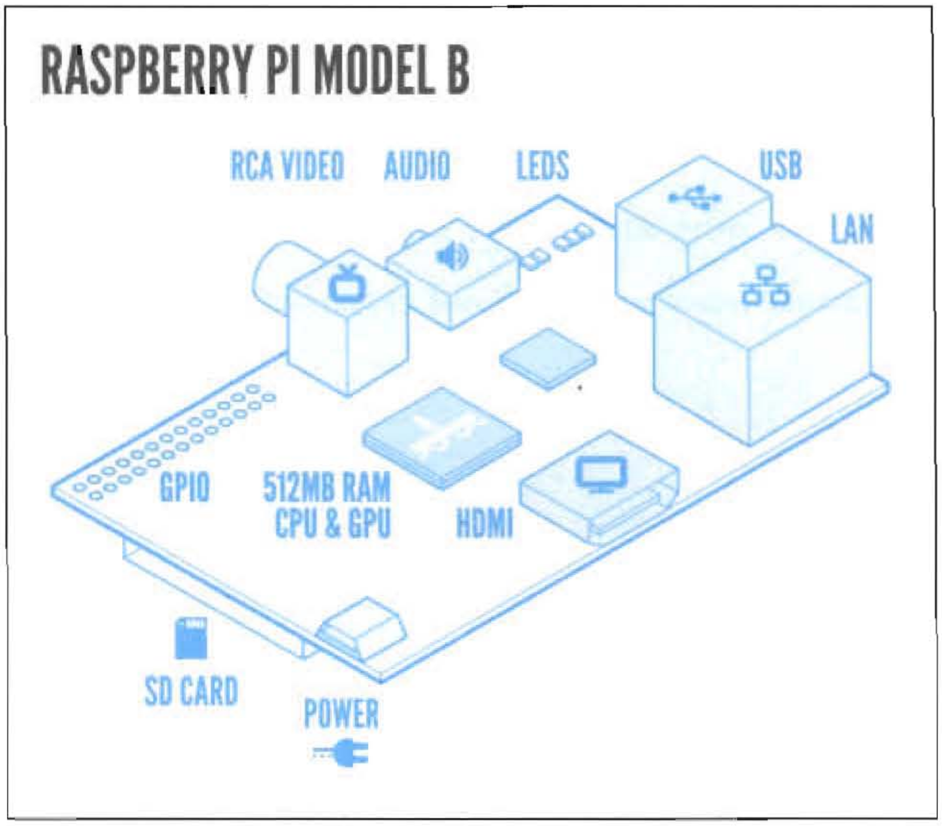


Figure 2.1: The general architecture of Raspberry Pi Model B (www.raspberrypi.org).

Since Raspberry Pi device uses ARM-based CPU, it only supports ARM-based operating systems (OS), mainly the ARM-Linux distribution and recommended distro is Raspbian, which specifically designed for the device. In other words, it does not support x86-based OS like Windows and cannot run any x86-based software (*FAQs*, n.d.).

2.3.4 Advantages of Raspberry Pi as Small File Server Platform

It is worth noting that for comparison purposes, a normal PC or laptop can be configured to work as a platform to serve as file server. However, they are not suitable or practical to be used in the long run. Raspberry Pi device is much better to be used in the context and functions of the proposed system – which is a small file server intended for home and personal usage. According to Clay (2014), Raspberry Pi has several advantages as a platform for a small file server such as:

a) Low power consumption.

   Compared to normal PC or laptop, Raspberry Pi device only use 5V using MicroUSB. This feature alone makes the device very capable and economical to run 24/7.

b) Small form factor.

   Raspberry Pi is very small and not bulky compared to normal PC or laptop. This makes it requires very small space to fit in a room.

c) Completely silent.

   Raspberry Pi use SD card for its main storage and thus has no moving parts compared to normal PC or laptop. Thus, the device itself makes no noise.

d) Affordable.

   For the purpose of building a small file server, it is very affordable alternative which cost just 35 USD. Compared to normal PC or laptop, couple of hundreds USD is just not worth it or even practical to be spent for a small file server platform.

2.4 Review on Existing Raspberry Pi Devices

As mentioned previously under Section 2.3, only four models of Raspberry Pi devices have been released. The first ones are the model A and B, commercially available since February 2012 and later model B+ in July 2014 and very recently the release of model A+ in early November 2014. The next subsections review the distinctive details of each model and also summarize the similarities and differences between the Raspberry Pi models.

2.4.1 Raspberry Pi model A

Raspberry Pi model A is basically the lower-spec variant of the Raspberry Pi, with 256MB of RAM and just one USB port and no Ethernet port (*Model A*, n.d.). This means that, this model has limited capability to connect with other hardware or devices and does not support networking. However, this also means that model A is lighter and consumes less power than its model B counterpart. This essentially makes this model ideal for embedded projects such as robotics where weight and power is of main concern (*Model A*, n.d.). And naturally it is quite cheap at a price of just 25 USD. Figure below shows the Raspberry Pi model A.

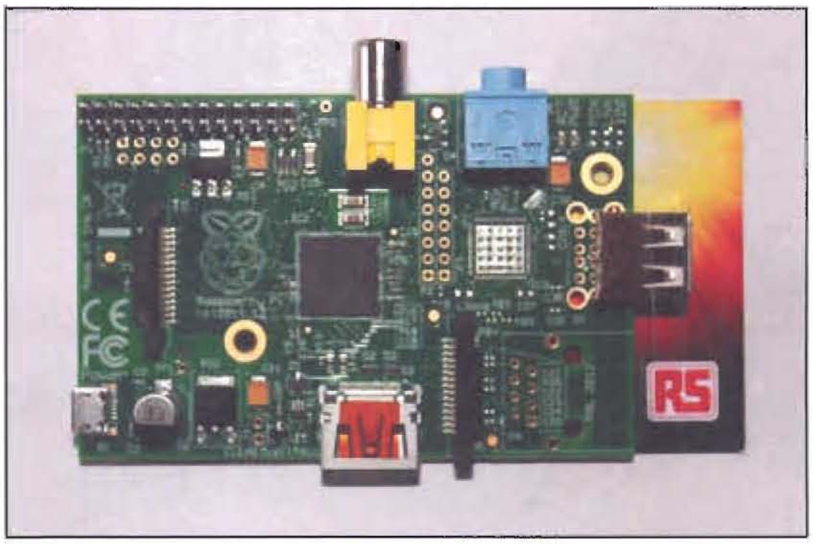


Figure 2.2: Raspberry Pi model A (www.rs-online.com).

2.4.2 Raspberry Pi model A+

Raspberry Pi model A+, like model A, is the lower-spec variant of Raspberry Pi and officially replaced its predecessor in November 2014. Compared to model A, this model has 40 GPIO pins (instead of 26), has Micro SD slot (instead of SD slot), lower power consumption, better audio and neater architectural design (*Model A+*, n.d.). These features make this model a properly improved version of model A despite still having limited USB port and no Ethernet port. And like its predecessor, it is suitable mainly for embedded projects, especially with more GPIO pins and even lower power consumption. In addition to these improvements, this model is even cheaper than its predecessor, costing just 20 USD. Figure below shows the Raspberry Pi model A+.

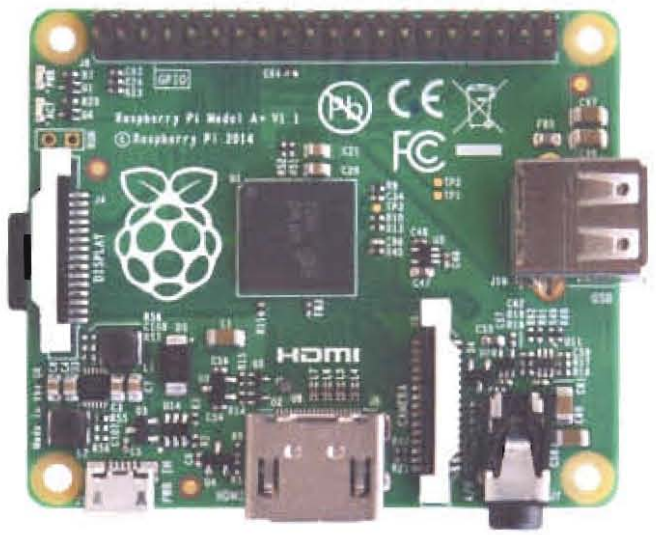


Figure 2.3: Raspberry Pi model A+ (www.raspberrypi.org).

2.4.3 Raspberry Pi model B

Raspberry Pi model B is basically the higher-spec variant of the Raspberry Pi, with 512MB of RAM, two USB ports and a 100MB Ethernet port (*Model B*, n.d.). This means that this model has more RAM, extra USB port and has Ethernet port for networking capability. This makes it the most popular model as it has all the typical features needed just like a desktop PC. It is basically the model with standard complete feature and this essentially makes it very suitable to power real-world projects as it is being used in various electronic projects (*Model B*, n.d.). And this model costs just 35 USD. Figure below shows the Raspberry Pi model B.

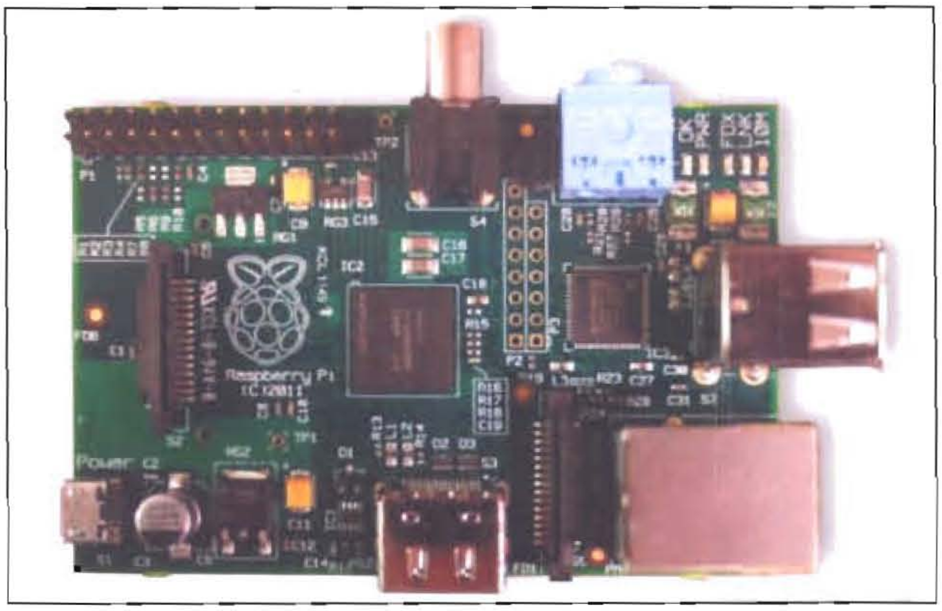


Figure 2.4: Raspberry Pi model B (www.geek.com).

2.4.4 Raspberry Pi model B+

Raspberry Pi model B+, like model B, is also the higher-spec variant of Raspberry Pi and officially replaced its predecessor in July 2014. Compared to model B, this model has 40 GPIO pins (instead of 26), 4 USB 2.0 ports (2 extra), has Micro SD slot (instead of SD slot), lower power consumption, better audio and neater architectural design (*Model B+*, n.d.). These features make this model a properly improved version of model B. The extra GPIO pins and USB ports make this model more flexible and suitable for various electronic projects as well as for educational purposes. In addition to that, this model still cost the same as its predecessor at 35 USD. Figure below shows the Raspberry Pi model B+.

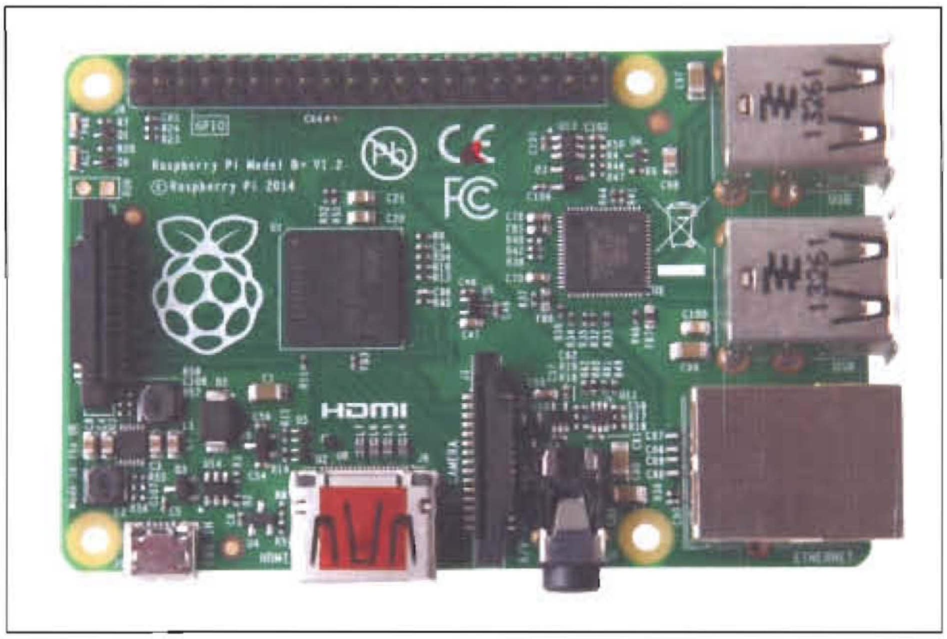


Figure 2.5: Raspberry Pi model B+ (www.raspberrypi.org).

2.4.5 Comparisons between the Raspberry Pi models

This section summarizes the similarities and differences between the Raspberry Pi models, as shown in table below. The data and values described in the table below are as stated in *FAQs* (n.d.), *Model A* (n.d.), *Model A+* (n.d.), *Model B* (n.d.) and *Model B+* (n.d.).

Table 2.1: The comparisons of Raspberry Pi models.

Criteria	Model A	Model A+	Model B	Model B+
SoC	Broadcom BCM2835			
CPU	700 MHz ARM1176JZFS			
GPU	Broadcom VideoCore IV GPU with OpenGL ES 2.0 (24 GFLOPS), 1080p30 H.264			
Memory	256 MB (SDRAM)		512 MB (SDRAM)	
USB Ports	1	1	2	4
Video	HDMI, RCA composite jack	HDMI, 3.5mm combined jack	HDMI, RCA composite jack	HDMI, 3.5mm combined jack
Audio	Analog 3.5mm audio jack, high-definition sound via HDMI			
Storage	SD/MMC/SDIO	Micro SD	SD/MMC/SDIO	Micro SD
Network	N/A	N/A	10/100 Mb/s Ethernet	
Power	5 V via MicroUSB			
Dimensions	85.60mm x 56mm	65 mm x 56.5 mm	85.60mm x 56mm	85.60mm x 56mm
Weight	45g	23g	45g	45g
Cost	35 USD	20 USD	35 USD	35 USD

Upon analyzing the information shown in table above, several points can be derived as listed below:

a) All Raspberry Pi models are based on the same chip type and thus making them equivalent in term of CPU/GPU processing and performance.

b) All models also support HDMI aside from having standard 3.5 mm audio jack and all models are powered with 5 V via MicroUSB.

c) Model B and B+ is obviously the higher end of Raspberry Pi as they are both having 512 MB of RAM instead of just 256 for model A and A+.

d) Only model B and B+ have networking capability due to its Ethernet peripheral.

e) Model B+ has the highest number of USB ports, which is 4, twice that of model B while model A and A+ both having just 1 USB port.

f) In term of storage, model A+ and model B+ are both equipped with the new MicroSD storage capability as they are the latest versions released in 2014.

g) Dimension-wise and weight consideration, model A+ is the lightest and the smallest of the four models.

Overall, model B+ seems to be the best model to be used for this project as it has 4 USB ports that is particularly useful for file server connectivity with devices. It also has network capability which is a must for a file server and has more RAM than model A or A+. Additionally, it has better audio quality, consume less power than model B and also neater in architectural design.

2.5 Overview of File Server Technology

2.5.1 Introduction to File Server

There are various definitions that constitutes to what is a file server. According to Beal (n.d.), a file server is a computer or device that is connected within a network, dedicated to storing files and any user connected on the network can store files on the server. While Rouse (2005) described file server as a computer that acts as a central storage and manage data so that they are accessible to other computers on the same network. And Hanson (2014) defines file server as computer in a LAN network that serves as centralized data storage for other machines as part of a client-server model of computer networking.

File servers are commonly used in business and educational fields where sharing of data is important and beneficial to all users in an organization. They enable users to share information over a network without the need to physically transfer them into other external storage devices. Any modern computer can be configured to act as a file server. And based on the number of hardware and software configurations, file servers can serve various purposes ranging from simple data storage and sharing, as backup for critical data and distribution of data over the Internet through FTP (File Transfer Protocol) and HTTP (Hyper-Text Transfer Protocol) (Hanson, 2014).

There are two main type of file server – dedicated and non-dedicated. A dedicated file server only serves as a file server and does not do any other tasks such as processing (Beal, n.d.). This means that it serves solely as central storage and hosts the data for other computers in its network. In a more sophisticated network, a file server can also be a dedicated NAS (Network-Attached Storage), which acts as a remote hard disk drive for other computers in the network (Rouse, 2005). On the other hand, a non-dedicated file server is essentially "a file server that can also be used simultaneously as a workstation" (*Non-dedicated file server*, n.d.). This means that it serves both as a file server and a workstation – stores and hosts data and at the same time do processing tasks on the data.

2.5.2 How a File Server Works

According to Evans (n.d.), there are three important points to describe how a file server works: first – it may run server software; second – it is equipped with plenty of storage space; and third – there are users connected to the file server. The server software are the operating system (OS) platforms like Windows Server, Mac OS Server, UNIX and Linux, which specifically designed to enhance file server functionality, making them ideal software platforms for file server setting and configuration (Evans, n.d.).

As for storage space, a file server typically equipped with one or more internal hard drives totaling with terabytes of storage space. These drives are configured through the server OS to host the files stored in them and once proper connection is established, users' computer will detect and treats the hard drives in the file server as additional drives (Evans, n.d.). Users can then access the files and manipulate them based on the configured permission settings. The file server may also enable the sharing of storage space to any users connected in the network.

File server in itself is a computer, complete with its own motherboard and typical peripheral and electronic devices connected to it such as processor (CPU), memory (RAM) and network card and storage devices. Unlike a normal PC, the hardware configurations and specifications vary since a file server is usually a dedicated one and has one main task – to serve as file server. Thus, it does not need other unnecessary hardware devices such as multimedia speakers and graphics processing unit (GPU). And also as stated by Evans (n.d.) earlier, it normally runs file server software, which acts as an interface program and controls the file server capability. A simple file server in a private network where the clients are connected whether through LAN or Wi-Fi has a general architecture as shown in the next figure.

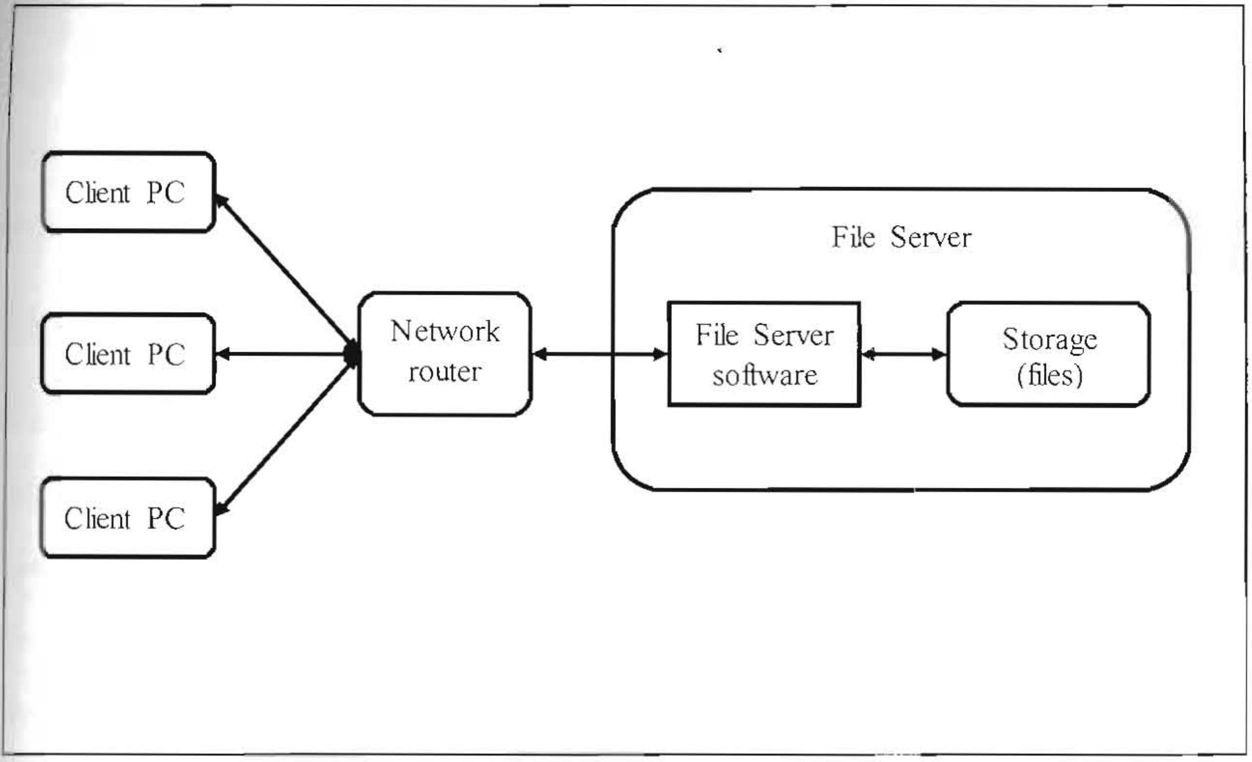


Figure 2.6: The general architecture of a file server.

Based on the figure above, a simple file server basically consists of file server software and its own storage to store the files that it hosts or share. Client PCs that are connected within the network and has proper access permission can then access the stored files in the file server storage. The network router is the main electronic device that provides interconnection for the whole network. The file server software is the central program that controls the file server and provides user interface for server administrator for system interaction and configuration. This general architecture of a simple file server is relevant and applies to all the file server software as discussed later in section 2.6.

2.6 Review on Existing File Server Software

There are several file server programs available whether commercially or for free. The popular ones are the Windows Home Server 2011, FreeNAS, Ubuntu Server Edition and Samba. The next subsections describe them in terms of their technology and implementation details.

2.6.1 Windows Home Server 2011

Windows Home Server 2011, code-named "Vail", is home server OS developed by Microsoft, designed for small office and homes, which was released in 2011 (Foley, 2011). It is built based on its predecessor Windows Home Server and it is the latest version of Windows Home Server (Foley, 2012). Due to the familiarity with Windows OS and given its popularity among most home users, WHS 2011 naturally is the easiest server software to set up, compared to Unix or Linux OS (Throckmorton, 2011). Despite of this, WHS 2011 is commercial type of software and thus not free, costing around 50 USD.

WHS 2011 only comes in 64-bit to support the use of more than 4 GB of RAM and it comes with media server capability with its robust transcoding and streaming capabilities including support for wide range of media codecs (Case, 2011). Aside from that, Case (2011) also stated that the sharing of files and printers is easier since WHS 2011 box can be listed under Windows 7 HomeGroup, though with limited flexibility on access permissions. It also has easier login management by using the same login account for PC to login on the server (Case, 2011).

Figure 2.7: Windows Home Server 2011 logo.

2.6.2 Ubuntu Server Edition

Ubuntu Server Edition is a variant of Ubuntu Desktop Edition, an open-source Debian-based Linux operating system which developed mainly to serve as file server software. Unlike the Desktop Edition, Ubuntu Server does not have GUI (Graphical User Interface), which makes it less user-friendly towards users that are more familiar with GUI-based OS like Windows (*ServerFaq*, 2012). And being a variant of Ubuntu, Server Edition is also relatively easy to install, maintain and upgrade for those familiar with Linux command-line interface.

Ubuntu Server Edition, like other Linux distribution OS, is also a flexible and can be configured to run a low-power and headless server (Fitzpatrick, 2009). According to Ubuntu (2014), the latest version of Ubuntu Server Edition, the 14.04 LTS version has all the improvements in the Linux kernel to Ubuntu and it is capable to run on x86, ARM64 and

POWER architectures. And being LTS (Long-Term Support) version, it has 5-years support in maintenance and security updates (Ubuntu, 2014).

Figure 2.8: Ubuntu Server Edition logo.

2.6.3 FreeNAS

FreeNAS is an open-source network-attached storage (NAS) operating system built based on FreeBSD. a Unix operating system (Throckmorton, 2011). Unlike FreeBSD which uses only old text-based system and command-line interface, FreeNAS offers an intuitive and user-friendly web interface along with command-line interface for ease of use (Throckmorton, 2011). However, it only runs on 32 and 64-bit x86 architecture hardware. It is also an extremely minimal distribution of FreeBSD, designed to minimize the resources devoted for storage (Fitzpatrick, 2009).

Like FreeBSD, FreeNAS has ZFS file system (much like NTFS and FAT32 in Windows) which offers similar functionalities of NTFS such as limitless file and partition size caps and auto repair (Throckmorton, 2011). And despite its minimal size, FreeNAS has various features - replication backup, data protection through backup services, data encryption, file system snapshots, file sharing, web interface and support for various third party plugins (*FreeNas Features*, n.d.) It also supports both hardware and software based RAID, disk encryption, and management of groups and users via local authentication or Microsoft Domains (Fitzpatrick, 2009).

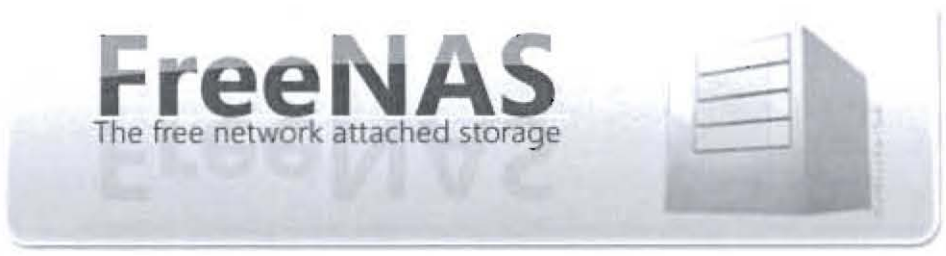


Figure 2.9: FreeNAS logo.

2.6.4 Samba

Samba is free file server software specifically designed to easily configure Ubuntu and Windows computer into a file server (Throckmorton, 2011). It provides secure, stable and fast file and print services for all clients using the SMB/CIFS protocol, which is used by Windows, OS/2, Linux and many others (Samba, n.d.).

Samba can run on various different platforms other than Windows, such as Unix/Linux, IBM System 390 and OpenVMS and it also allows "interoperability between Linux/Unix servers and Windows-based clients" (Samba, n.d.). This makes Samba particularly useful to run file server that serves mixed clients of different platforms. In addition, it gives network administrators flexibility and freedom in terms of setup and server configuration, making it grown in popularity since its release in 1992 (Samba, n.d.).

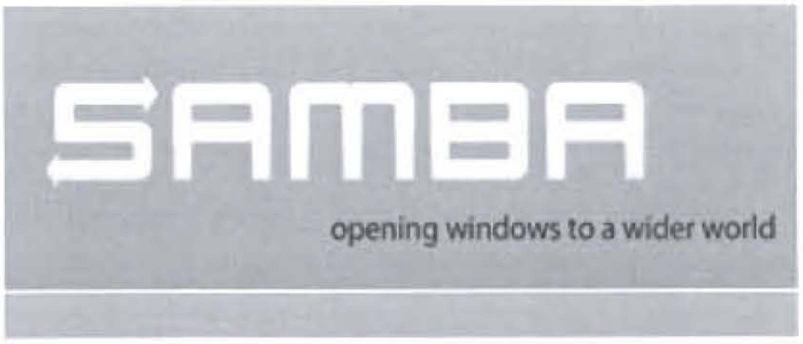


Figure 2.10: Samba logo.

2.6.5 Comparisons Between the File Server Software

This section describes the comparisons between the file server software by analyzing their similarities and distinctive features as described previously. They are as shown in table below:

Table 2.2: The comparisons between WHS 2011, Ubuntu SE, FreeNAS and Samba.

Criteria	WHS 2011	Ubuntu SE	FreeNAS	Samba
Platform	Windows	Linux/Unix	Windows	Multiplatform
Program Interface	Windows GUI (Windows 7)	Mainly command-line	Web Interface	Mainly command-line
Software type	Operating System	Operating System	Operating System	Application program
Ease of usage	Easy	Hard	Easy	Easy
Usage Flexibility	Limited	Flexible	Flexible	Flexible
Features	Various	Various	Various	Various
Updates	Yes	Yes	Yes	Yes
Media Center Support	Yes	Yes	Yes	Yes
Cost	50 USD	Free	Free	Free

Upon analyzing the information shown in table above, several points can be derived as listed below:

a) WHS 2011, Ubuntu SE and FreeNAS are all operating system-type of software while only Samba is an application program.

b) WHS 2011 and FreeNAS can only run on Windows platform while Ubuntu SE on Linux/Unix platform and only Samba is capable to run in at least both platforms.

c) WHS 2011 and FreeNAS are easy to use due to their GUI-based interface.

d) All of them have various features and flexible in term of its usage except for WHS 2011.

e) All file server programs have updates support and can be configured into media center.

f) All file server programs are free except for WHS 2011 which costs 50 USD.

Since the Raspberry Pi device only supports ARM-based software like Linux, we have only Ubuntu SE and Samba as primary choices. And since the Raspberry Pi device will be installed with Raspbian OS, this automatically makes Samba as the only choice to be used as file server programs among the four. Besides, Samba is very suitable for the proposed system mainly because it can run on multiplatform and at the same time capable to serve clients with multiple platforms.

Even though Samba has mainly command-line interface, it is relatively easy to be configured. And a web-based interface for the main system interface will be developed to help users conveniently interact with Samba functionalities. Besides it has updates support, flexible with support for various file server features, easy to use, supports configuration to be used as media center and it is free.

2.7 Overview of Media Center Technology

2.7.1 Introduction to Media Center

Media center can be described as an electronic device that has the capability to support the playback of media files such as video, audio and image on external video and audio output hardware. In the context of this project, media center refers to the Home Theater Personal Computer, or HTPC. By definition, a home theater PC is basically a "personal computer that is used to store and play music and movies as well as display photos" (*HTPC*, n.d.).

A HTPC is normally physically connected to several other output devices such as television, PC monitor and speakers. It may include a remote control and a GUI interface for users to easily navigate between the media files stored in the HTPC storage. Alternatively, users can simply use standard keyboard and mouse. In short, users can use the HTPC to access the media files and conveniently view them on large screen TV with superior sound from large speakers.

2.7.2 How a Media Center Works

At its simplest form, a media center or Home Theater PC in this context – is a home PC that is connected to audio and video receiver and the PC acts as the central device in a home theater environment (Layton, 2006). And as a typical computer, the PC has all the basic functions ranging from computing functions, video files streaming, editing and playback, audio files recording and playback and also serves as media storage (Layton, 2006). The HTPC is normally connected to television, typically a large screen TV through various wiring options.

And HTPC is usually equipped with huge storage size and sometimes requires extra storage devices such as external hard disk drives (Layton, 2006). And as described previously, users can use the TV to navigate through the stored content by using standard input devices and remote control. For better audio playback quality, users may install a sound card on the HTPC to support and utilize the capability of the connected speakers. Additionally, the HTPC may be equipped with a TV Tuner to allow it to receive TV signals and record programs (if the tuner has DVR capability) and may also be equipped with AM/FM radio receiver (Layton, 2006).

A simple media center or HTPC has a general architecture as shown below:

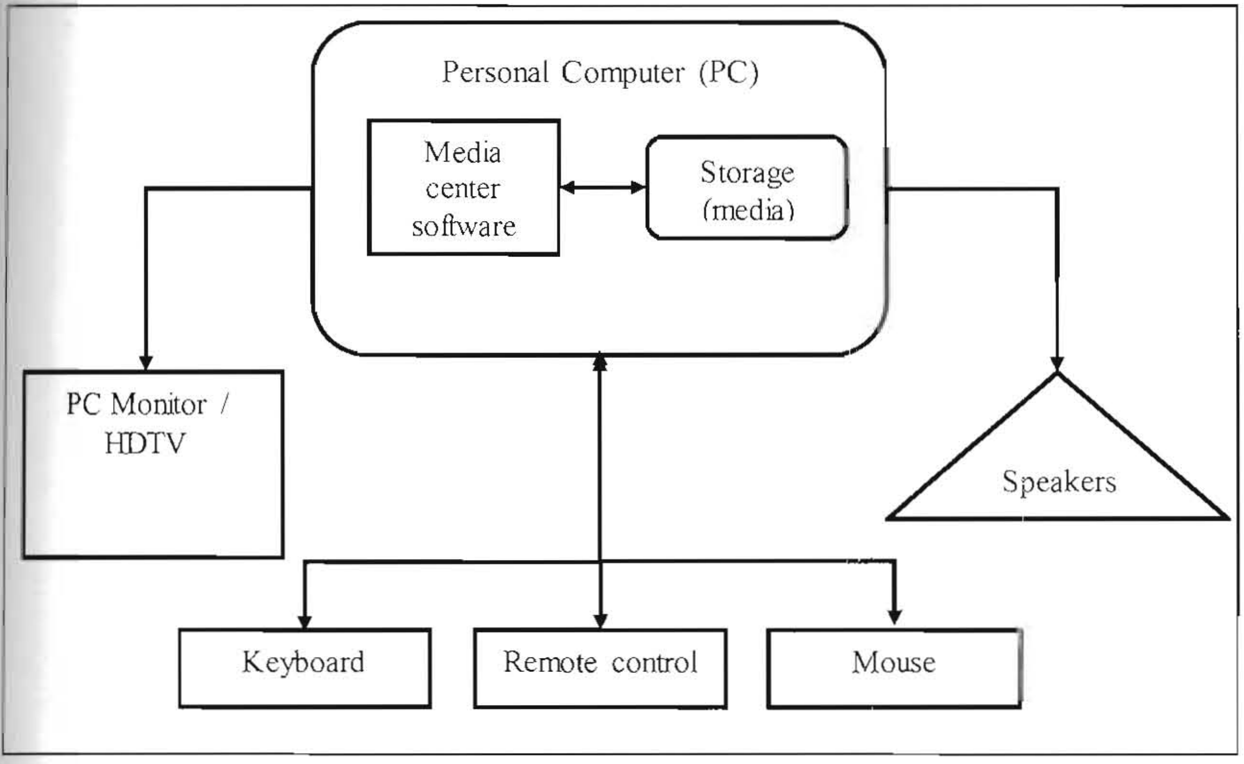


Figure 2.11: The general architecture of a media center (HTPC).

Based on the figure shown above, a media center basically consists of a PC installed with a media center software and its own storage to store the media files containing video, audio and image files. To complete its function, the PC must be connected with basic video and audio output – in this case, a PC monitor or a TV and standard speakers. Aside from that, a normal PC would then connected with standard input devices such as keyboard and mouse – either connected through USB ports or wirelessly. The media center software is the central program that accesses the media files and provides user interface for users to browse through the storage directories. A remote control is an optional device for user convenience while viewing and listening to the media files.

2.8 Review on Existing Media Center Software

There are several media center programs available that can provide GUI navigation and access to the media files on a HTPC. The most popular ones are the Windows Media Center, XBMC Media Center and Boxee.

2.8.1 Windows Media Center

Windows Media Center (WMC) is a free media center program that is usually pre-included in Windows Vista and Windows 7. WMC is the most mainstream media server and apparently very good for streaming music and music videos (Tuukka, 2011). WMC by default provides access to all media library in Windows OS and can play several media formats. According to Fitzpatrick and Purdy (2010), WMC has its own strengths and weaknesses as shown in the table below:

Table 2.3: The strengths and weaknesses of WMC.

Strength	Weakness
Nice and easy DVR without having to pay a monthly fee.	**Limited codec support** for the diverse range of video and audio formats.
Good interface that has obvious sections and fairly intuitive directional layouts.	**Platform limitation** as it can only run on Windows.
Large range of compatible remotes that available in most electronic stores.	**Complex remotes** that sometimes confusing and not simple to use.
Generally **easy uetworking** across Windows systems through the use of shared folders.	**Locked-down DVR files** where the recorded video can only be played on authorized Windows machines, Zunes, and XBox devices.

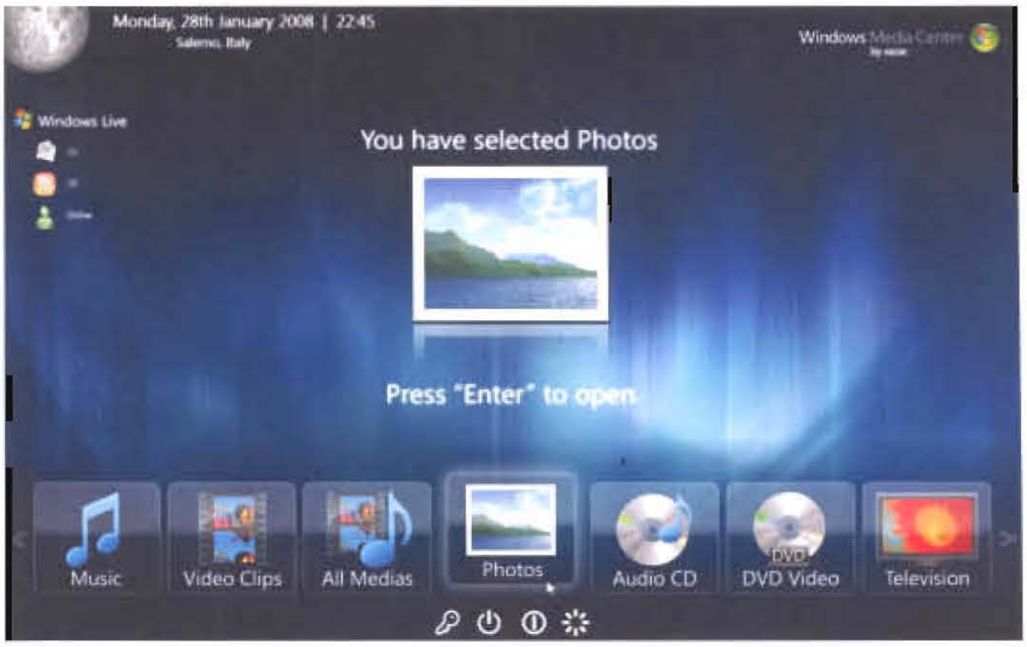


Figure 2.12: The screenshot of Windows Media Center interface.

2.8.2 XBMC Media Center

XBMC is an open-source media center software that was originally designed as a media center interface for the XBOX game console and over the years has evolved into a powerful multiplatform media center capable of playing most available media formats (Tuukka, 2011). He also described XBMC as well-established and robust software, which makes it the foundation for other alternative media centers. Note that as of November 2014, XMBC version 14 and onwards were rebranded as Kodi media center. According to Fitzpatrick and Purdy (2010), XBMC has its own strengths and weaknesses as shown in the table below:

Table 2.4: The strengths and weaknesses of XBMC.

Strength	Weakness
Open source and **open nature** which makes it always open to improvements/modifications.	**Lack of streaming support** for TV and video streaming sites like Netflix and Hulu.
Good meta-data and file recognition that efficiently tracks files properties.	**Over-stuffed**, sometimes with complicated menus that can be rather confusing to new users.
Light and agile because it is focused on smooth media files playback.	
Slick UI design that by default is already futuristic-looking and customizable UI further adds to XBMC flexibility in interface design.	
Excellent media format support as XBMC can play literally any media format available.	

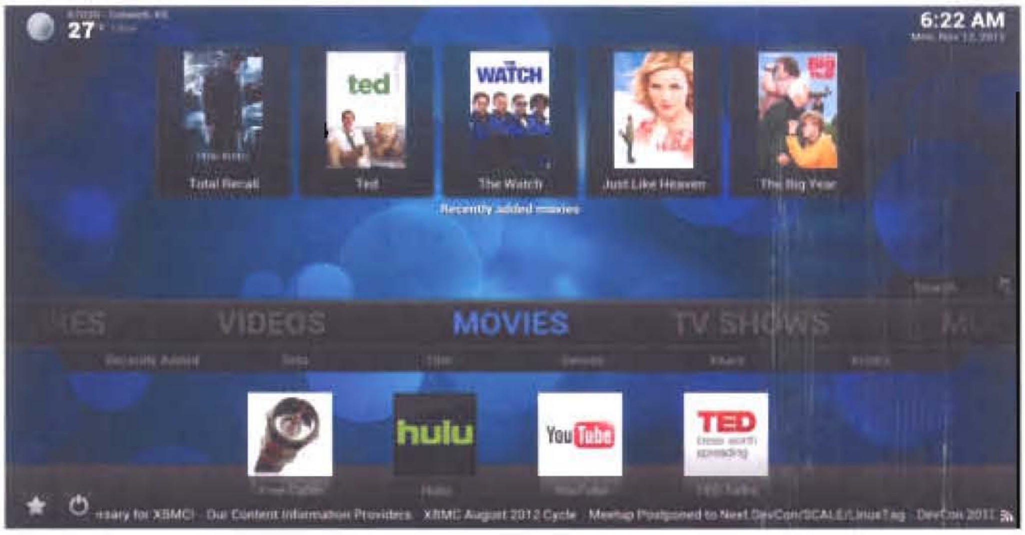


Figure 2.13: The screenshot of XBMC interface.

2.8.3 Boxee Media Center

Boxee media center is based on the same internal code as XBMC but its media is focused mainly on web content from video sites, blog streams and social apps (Fitzpatrick & Purdy, 2010). Besides act as a typical media center, Boxee is also capable of integrating online video content with user recommendation system (Tuukka, 2011). According to Fitzpatrick and Purdy (2010), Boxee has its own strengths and weaknesses as shown in the table below:

Table 2.5: The strengths and weaknesses of Boxee.

Strength	Weakness
Built-in support with Hulu and Netflix which enable users to stream content from the video streaming sites.	**Mediocre interface design** that is rather weird and hard to get used to and even disliked by some users.
Growing directory of **web content apps from popular social sites** like FailBlog and Vimeo and has app "store" for customized streaming content.	**Lack focus on handling local file** where Boxee is not effective in recognizing and updating local file stores.
Capable to play any web content media as it uses a reworked Firefox browser to view Hulu aside from its own browser.	

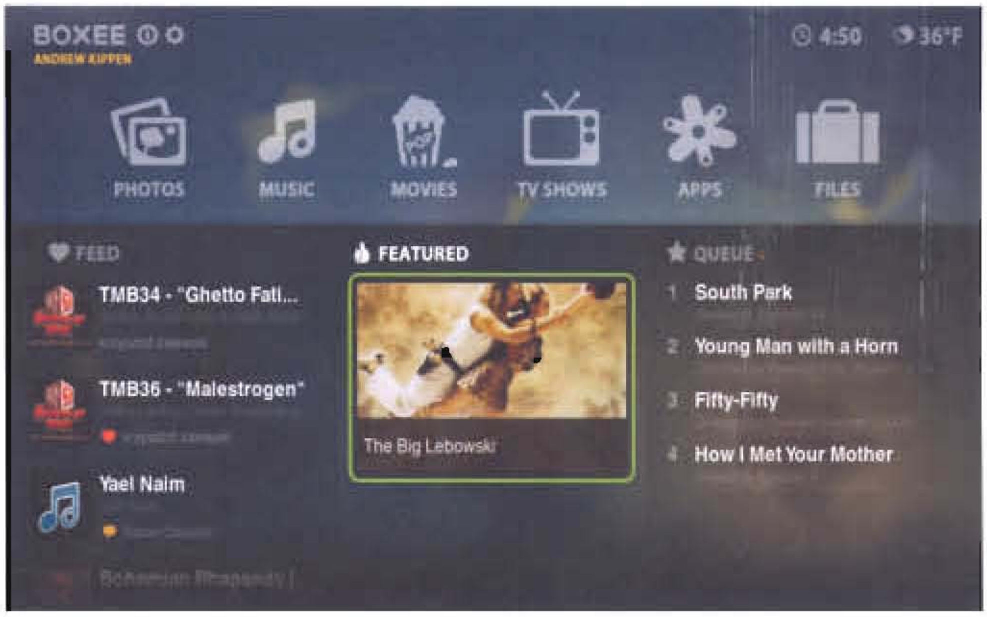


Figure 2.14: The screenshot of Boxee interface.

2.8.4 Comparison between the Media Center Software

This section describes the comparisons between the media center software by analyzing their similarities and distinctive features as described previously. Extra information also being used from Fitzpatrick and Purdy (2010) analysis. They are as shown in Table 4 below:

Table 2.6: The comparisons between Windows Media Center, XBMC and Boxee.

Criteria	WMC	XBMC	Boxee
Supported platforms	Windows	Windows, Mac, Linux, Xbox, Apple TV	Mac, Linux, Apple TV
Media format support	Medium	Very high	High
TV Tuner support	Yes	Yes	No
Hardware acceleration	Yes (through DirectX)	Yes (through NVIDIA)	Yes (through NVIDIA)
Hulu and Netflix streaming support	Netflix only	No	Yes
GUI design quality	High	Very High	Low
Customizable GUI	Yes	Yes	No
Remote control support	High	High	Low

Upon analyzing the information shown in table above, several points can be derived as listed below:

a) Only XBMC and Boxee can run on multi-platforms and WMC naturally runs only on Windows.

b) All 3 media center programs are capable of media playback but XBMC has the highest playback support for wide range of media format.

c) All 3 media center programs support hardware acceleration but only WMC and XBMC has TV tuner support.

d) Only Boxee has both Hulu and Netflix streaming support, however it does not have support for customizable GUI like WMC and XBMC.

e) XBMC has very good design quality and like WMC, it also has high support for remote control.

Overall, the XBMC is the best media center among the three simply because it has the highest supports for media format which is very important as key feature to be had for the proposed file server in this project. Besides, it can run on various other platforms and has great customizable GUI design aside from having support for wide range of remote control.

2.9 Overview of Proposed System

Based on the analysis of comparisons done in Section 2.4.5, Section 2.6.5 and Section 2.8.4, it is proposed that the system to be developed is based on the best selection of Raspberry Pi model, file server software and media center program. Thus, the proposed system will be built on Raspberry Pi model B+, using Samba as its file server software and XBMC program to provide it with media center capability.

These choices are much preferable due the reasons described in the comparison analysis. And since the proposed system will include media center capability, this directly makes it a non-dedicated file server. The system will also be improved with a web-based interface for users to interact with the system and utilize all the system functions. The requirement details of the interface and its proposed design are described in Section 3.3.4 and Section 3.4.3 respectively.

As for the architectural design and schematics, it will be built based on the proposed architecture shown in Section 3.4.1 and schematics in Section 3.4.2.

2.10 Summary

Conclusively, this main chapter has provided the relevant literature reviews on the main components that are going to be used in this project. It has presented the overviews of Raspberry Pi models, described in details on file server and media center technology and the popular choice for each. This main section also compared and analyzed each component in terms of their variations and available programs to choose the best for each. Analysis result concluded that the proposed system will be built on Raspberry Pi model B+, using Samba as its file server software and XBMC program for its media center capability.

3 REQUIREMENT ANALYSIS AND DESIGN

3.1 Introduction

This chapter contains the development methodology and the requirement analysis and design for the system implementation of this project. It consists of several subsections that describe in details about the chosen development methodology, the analysis on the hardware, software and user requirement and the design details of the system.

3.2 Development Methodology

This section describes the development methodology for this project. The methodology chosen for the implementation of the personal file server and media center using Raspberry Pi will be based on the incremental development. As mentioned previously in Introduction section, incremental development is chosen because the system will have several functionalities that need to be delivered separately and added incrementally. Initially, the system will have very basic functions of a file server and then basic media center capability. Once these two features are implemented, several other functions will be added to improve the system as a whole in terms of functionality and features.

3.2.1 Incremental Development

According to Sommerville (2011), incremental development is basically the process of developing an initial implementation, expose it for user feedback and improve it through several versions until an adequate system is finally developed. He further stated that specification, development and validation activities are interleaved as concurrent activities rather than separate, with rapid feedback across activities. Figure below shows the architecture of incremental development.

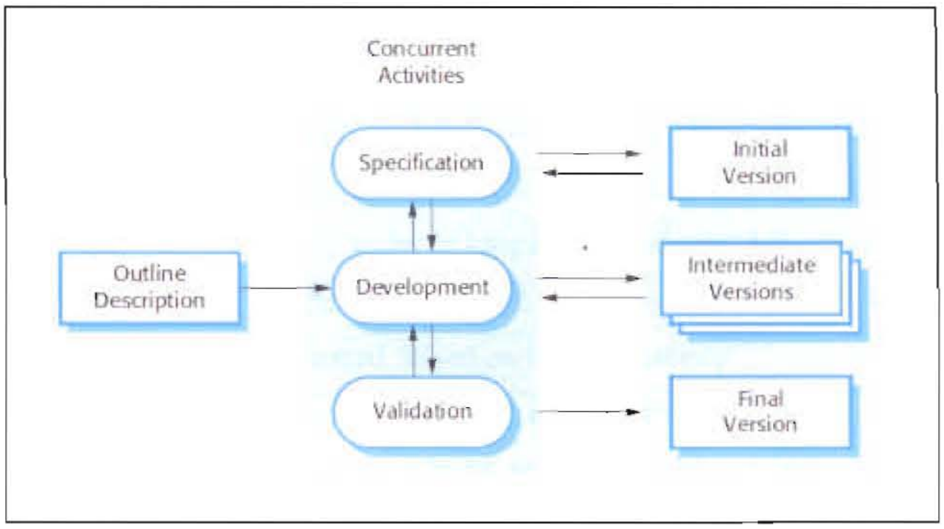


Figure 3.1: The architecture of incremental development (Sommerville, 2011).

Based on the figure above, incremental development consists of several elements: outline description, concurrent activities (consist of specification, development and validation) and the system versions (consist of initial, intermediate and final version). The details for each are as described in Table 3.1 below:

Table 3.1: The concurrent activities in incremental development.

Elements	Description
Outline Description	The basic outline that describes the system requirement (the system functionalities to be developed).
Concurrent Activities	The interleaved activities of main implementation process. They are to be performed iteratively to implement each of the system functionalities based on the outline description from end-user.
a) Specification	The specification activity is where system functionality is specified in details terms of technicality and expected outcome.
b) Development	The development activity is where system functionality is being implemented based on its specification.
c) Validation	The validation activity is where the implemented system functionality is being tested and validated with its specification.
System Versions	The variation of system versions as it is being developed throughout the implementation phases.
a) Initial version	The initial version is where the system has its most basic function and feature implemented.
b) Intermediate version	The intermediate version is where the system has some additional functions and features being added incrementally through a series of iterative implementation process.
c) Final version	The final version is where the system has all its functions and features fully implemented and ready for delivery to end-user.

The resulting system functionalities and features will be implemented through a series of specification, development and validation as described previously. In order to properly implement the system, the development part of the incremental development is to be divided into several phases as briefly described in table below. The descriptions of each development phase are described later in details under Chapter 4 (Implementation and Testing).

Table 3.2: The planned development phases for development part.

Phase	Main activities
Development phase 1	Preparation of the Raspberry Pi device with its operating system, essential programs and drivers.
Development phase 2	Implementation of file server capability. Will be using Samba file server software.
Development phase 3	Implementation of media center capability. Will be using XBMC media center software.
Development phase 4	Implementation of webserver capability. Will be using LAMP webserver component.
Development phase 5	Implementation of system interface (web-based interface) mainly for the file server function and secondary media center function.
Development phase 6	System integration and validation. Involves the process of integrating all the components and configuring their functionalities.

3.3 Requirement Analysis

This section describes the requirement analysis for this project. It covers the hardware and software requirements that are needed for the implementation and also user requirements that describe all the functions and features to be implemented.

3.3.1 Hardware Requirements

All the hardware and devices needed for this project are as described in table below:

Table 3.3: List of hardware needed with their specification and description.

Hardware	Specification	Description
Raspberry Pi	Model B+	The central electronic device for this project
SD Card	MicroSD Card, at least 8 GB, preferably of class 10	The primary storage for Raspberry Pi (OS and system files)
PC Monitor	Standard PC monitor, LCD/LED, with HDMI support	Standard display output device
Television	High-definition flat screen TV, LCD/LED, with HDMI support	High-definition display output device
Speakers	Standard PC speakers, at least 2-channels (stereo)	Standard audio output device
Standard PC keyboard	Standard Universal Plug-and-Play (UPnP) keyboard, USB or wireless	Standard input device to interact with the system
Standard PC mouse	Standard Universal Plug-and-Play (UPnP) mouse, USB or wireless	Standard input device to interact with the system
External Hard-Disk Drive	Standard 3.5" external HDD, at least 500 GB, with USB 2.0 support	Main storage device for file server purposes
Powered USB hub	Standard powered USB hub, providing at least 4 extra USB slots	External USB hub to sufficiently power several hardware devices
Network router	Standard network router, with LAN and Wi-Fi capability	Standard router for networking – for file server purposes

PC or laptop	Standard PC or laptop, running WinXP/Win7/Win8	Standard client hardware, for access within the network
Wirings	LAN cables, power cables, 3.5mm audio jack cable, USB cables, HDMI to USB converter	For network, input/output and interconnection among the hardware and external devices
Extension cord	Standard extension cord with at least 4 extra power plugs	To provide enough power plugs to power all the hardware and devices

3.3.2 Software Requirements

All the software and extra programs needed for this project are as described in table below:

Table 3.4: List of software needed with their specification and description.

Software	**Specification**	**Description**
Rasbian OS	Raspbian version Sept 2014 (Debian Wheezy), kernel 3.12	The Linux OS for the Raspberry Pi device, also part of LAMP server
Samba file server	Samba version 4.1.13, stable release	The standard program to provide file server capability
XBMC program*	XBMC version 13.2 (Gotham) or Kodi version 14+	The standard program to provide media center capability
Apache program	Apache HTTP Server 2.4.10 (httpd), stable release	Part of LAMP server to cater for the System Interface
MySQL server	MySQL Community Server version 5.6.22	Part of LAMP server to cater for the System Interface
PHP 5	PHP version 5.6.4	Part of LAMP server to cater for the System Interface
Drivers	Depending on hardware and devices driver requirement specifications	Related drivers that may be needed to support the installation of external hardware and devices

(Specifications based on data from *Raspbian* (n.d.), *Samba.org* (n.d.) and *kodi.tv* (n.d.).)

*Note that for XBMC program, version 14.1 was initially used during implementation (and later upgraded to version 14.2) and it was rebranded as Kodi for version 14+.

3.3.3 User Requirements

User requirements are the requirements that end-user expects in the final system to be implemented. For this project, the user requirements focus mainly on the features of the final system ranging from basic functions to additional features.

The user requirements for this project are as described below:

a) **File server capability**

The system shall have file server capability, provided in a private network, accessible to client users through LAN or wireless connection. The file server shall be able to host and share files (both data and media files) to other authorized clients within same network. Clients shall have at least read permission to the shared data stored in the file server.

b) **Media center capability**

The system shall have media center capability, provided in a private network, accessible to client users through LAN or wireless connection. The media center shall be able to access all media files stored in the file server storage (external HDD) and be able to open and play the media files. The media center shall support the playback of various popular audio and video files format. Clients shall be able to view and listen to the media playback through the system standard video output devices (PC monitor/TV) and standard audio output device (stereo speakers).

c) **Main system interface**

The system shall be added with a web-based system interface. The interface will be used by to interact with the file server function, specifically Samba file server functionalities. The interface shall be extended to integrate and manages all the functions and features of the system for user convenience. The requirement specification for the system interface is described in details in Section 3.3.4.

3.3.4 Web-based User Interface Requirements

The primary purpose of the Web-based User Interface (WebUI) will be to act as the GUI to configure the file server capability by interacting with the Samba file server, and in this case it will be a web-based administration tool. Other than that, the system interface will also act as a homepage for all the functions of the whole system. The system interface will be a web-based interface, to be built using PHP (Hypertext Preprocessor) and HTML (Hyper-Text Markup Language) scripting language.

Through PHP, the system interface will have capability to accept user input through web form and use the input for several functions including user login and validation, file sharing management and also the file server configuration. The interaction between the web-based interface and the Samba file server will be possible by executing external commands using PHP scripting language and the output or results will be displayed on the web-interface tool.

3.3.4.1 WebUI file server configuration requirements

The requirement details for file server functions include:

a) Login feature to validate user before they can start using the file server function. Once validated, they can use the web-interface to manage file sharing functions and configure the file server.

b) File sharing management functions includes adding new share, modifying an existing share and deleting an existing share.

c) File server configuration functions includes modifying server settings in term of security settings and file sharing access.

Due to the nature of this direct accessibility to modify the internal system, this function is limited only to the admin of the system.

3.4 Design

This section describes the design details for the system of this project. It covers the system architectures for the system and its main interface that will govern all the functions and features of the system.

3.4.1 System Architecture

The architecture for this system varies as it is being developed. This is because the functions and features are to be implemented incrementally. Thus, the system physical and hardware configurations for system initial version will be different with its intermediate/final version.

3.4.1.1 Initial Version Architecture

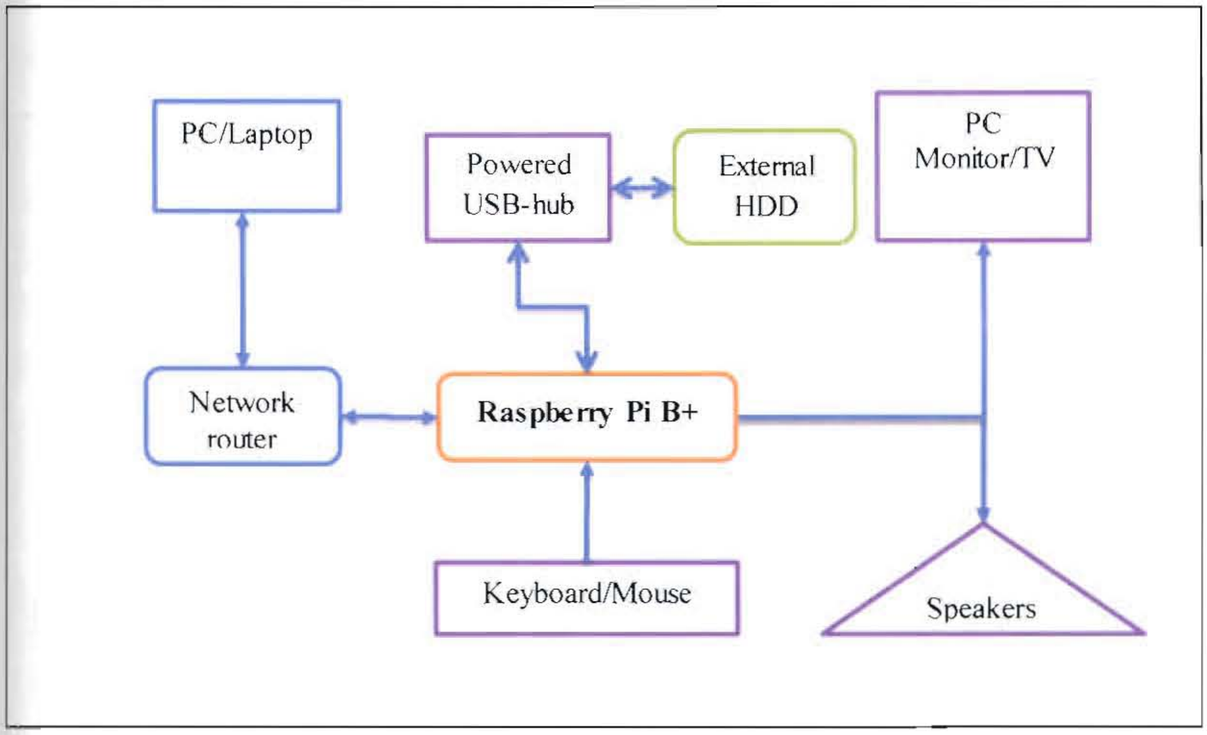


Figure 3.2: The architecture of initial system version.

Based on the figure above, the initial version architecture is based on the system having its basic functions and features. In this version, the system has the following functions:

a) File server function

b) Media center function

In this architecture, the file server uses external hard-disk drives (HDDs) as its main storage and a network router for its network connection. The file server hosts both data and media files and it is accessible to users through client computers connected within the same network through LAN or Wi-Fi. The media center function uses standard video output devices (PC monitor or TV), standard audio output device (speakers) and standard input devices (keyboard and mouse). Through the media center program, user can then access the media files stored in the system and view them directly.

3.4.1.2 Intermediate/Final Version Architecture

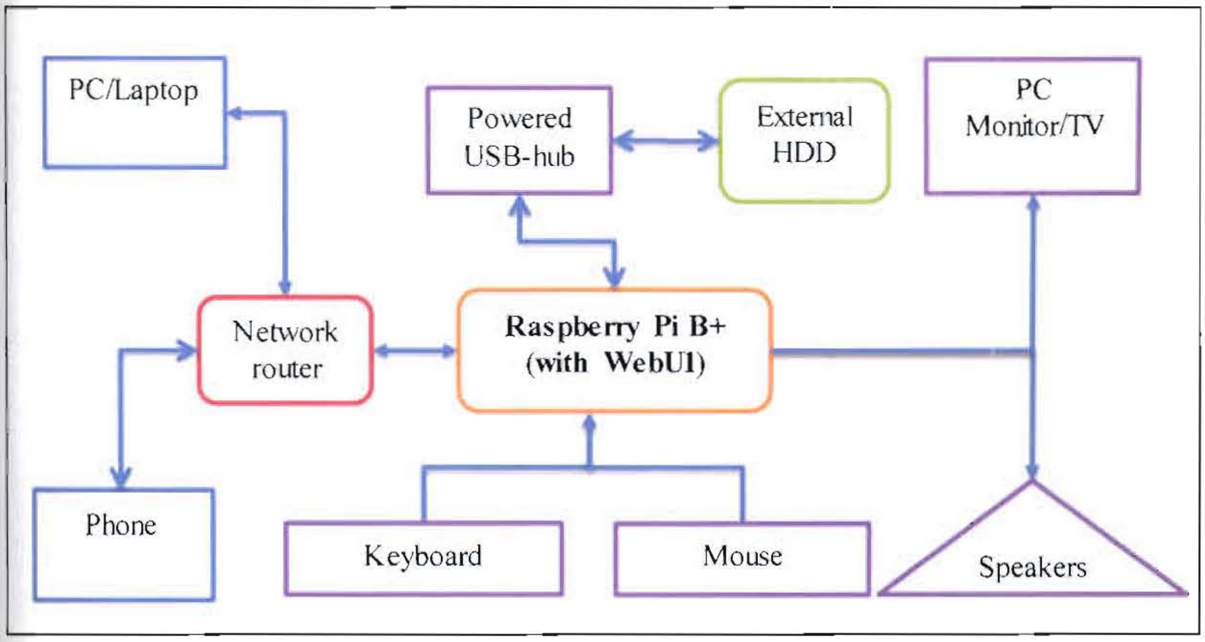


Figure 3.3: The architecture of intermediate/final system version.

Based on the figure above, the intermediate/final version architecture is based on the system equipped with its basic and additional functions and features. In this version, the system has the following function and features:

a) File server function
b) Media center function
c) Main system interface (the WebUI)

In this architecture, the basic configurations and usability for file server and media center are slightly different than described in previous sub-section. It is proposed that the system to have two external HDDs to separate the data and media files for proper storage management purposes. This final system version may also be complete with additional functions. Finally, the system will also have a main system interface that will act as an administrative tool to manage all the basic and additional functions and features.

3.4.2 System Schematics

This subsection discusses the schematics of the system architecture shown in previous section.

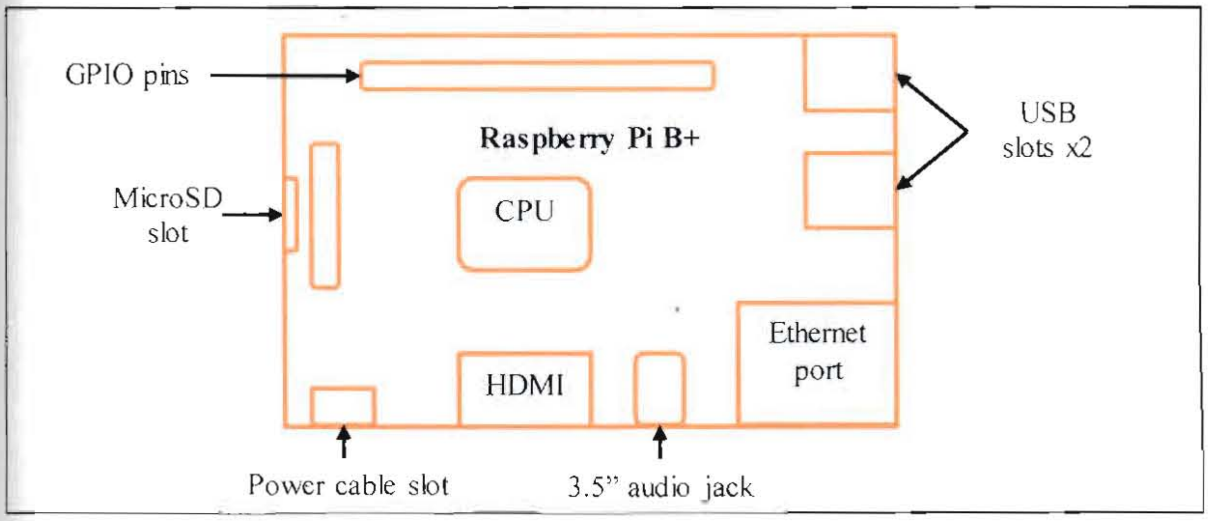


Figure 3.4: The schematic of Raspberry Pi model B+.

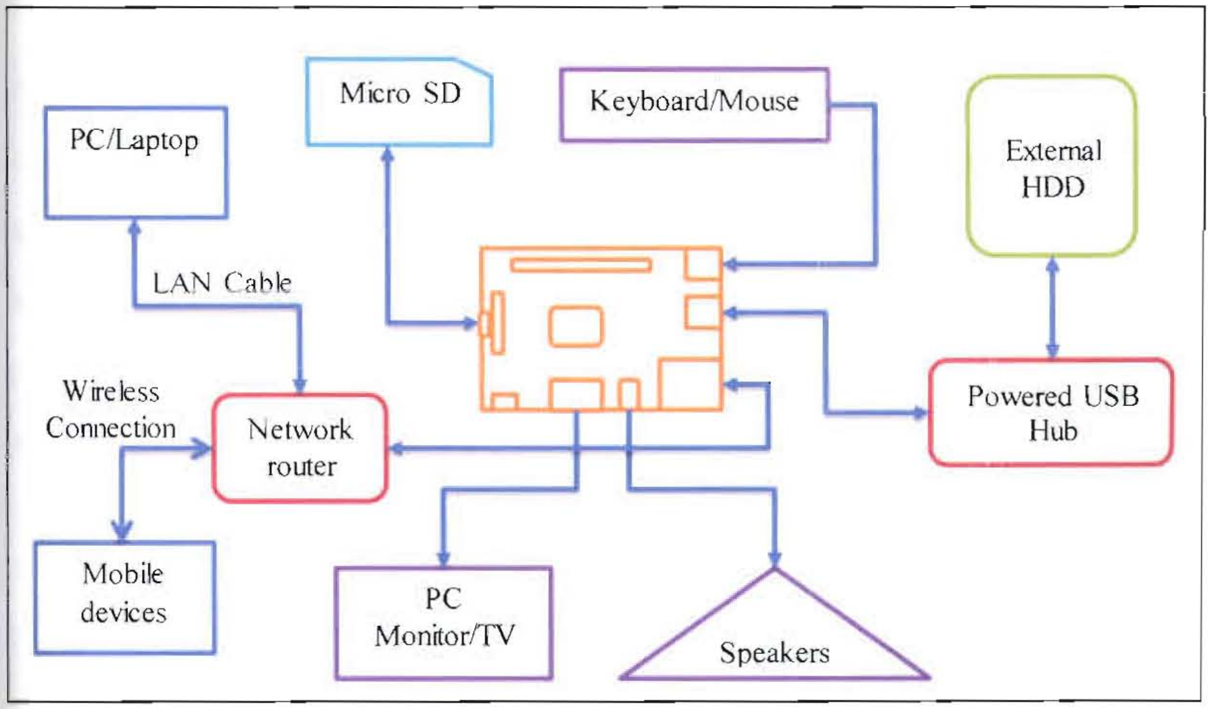


Figure 3.5: The schematic of intermediate/final system version.

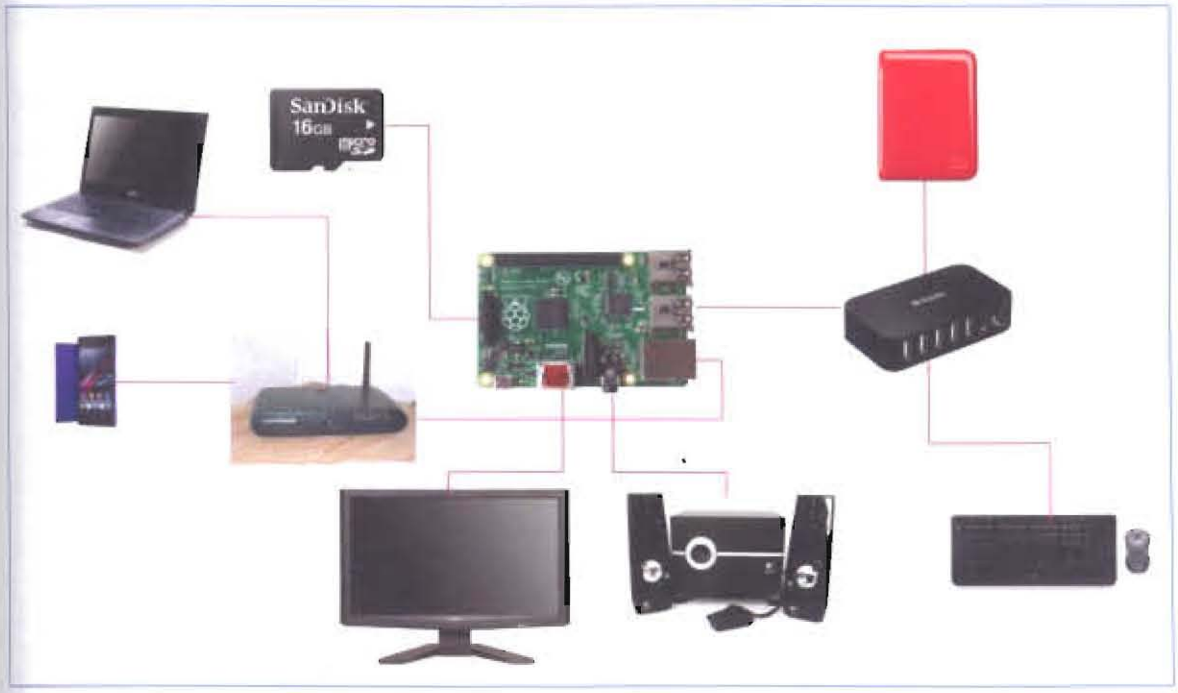


Figure 3.6: The schematic of intermediate/final system version (picture).

3.4.3 Web-based User Interface

The Web-based User interface is intended to have one primary function: the file server configuration function which enable user to configure file server shares and server settings. Alternatively, if possible, the Web UI will be extended to include secondary media center function, capable to play media files through web player, aside from Kodi being primary. Other functions may be added later based on the WebUI functionality. Next subsections illustrate the proposed interface for the WebUI file server function and media center function.

3.4.3.1 Interface for File Server Function

For file server function, it includes login feature to validate the user.

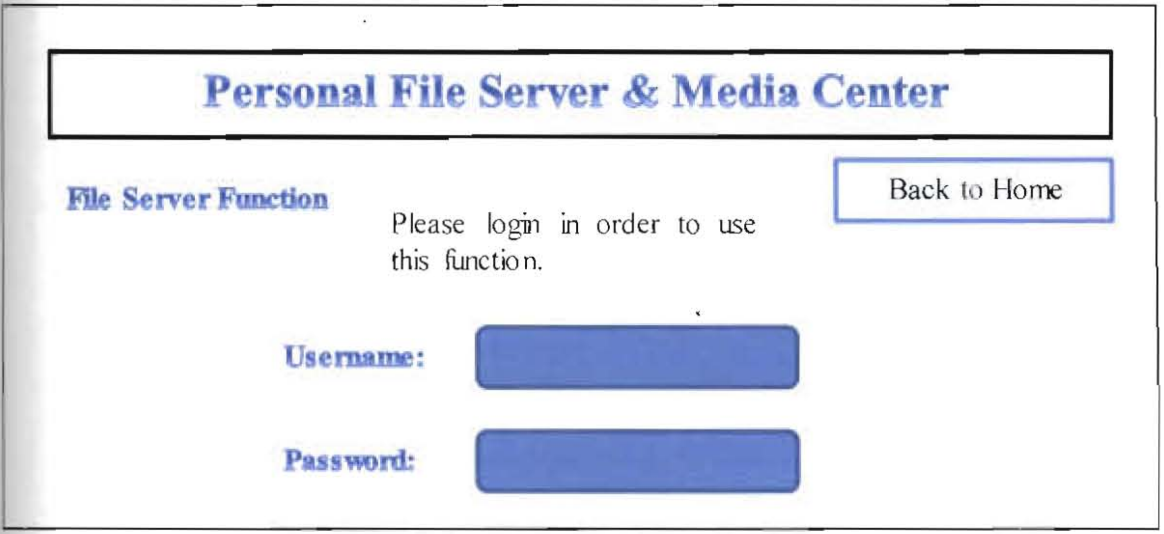
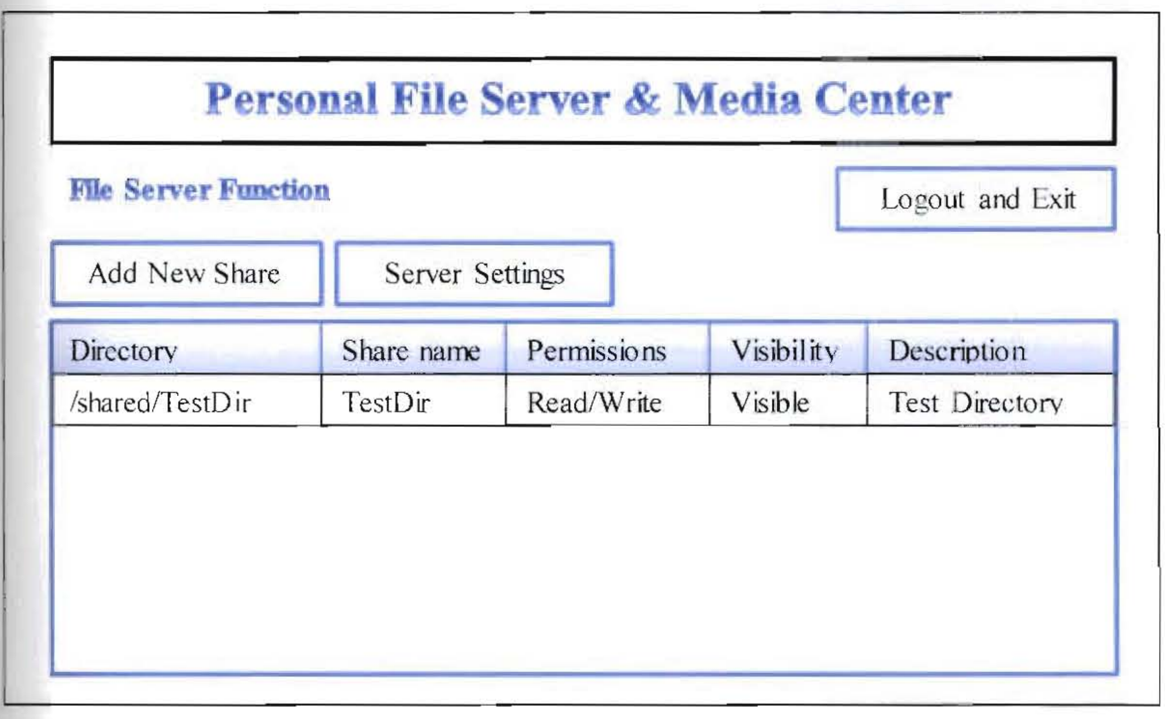

Personal File Server & Media Center
File Server Function
Back to Home
Please login in order to use this function.
Username:
Password:

Figure 3.7: The login interface for file server function.

Once validated, user will be directed to the default interface for file server function.



Directory	Share name	Permissions	Visibility	Description
/shared/TestDir	TestDir	Read/Write	Visible	Test Directory

Figure 3.8: The default interface for the file server function.

User can then add new share by clicking the "Add New Share" button.

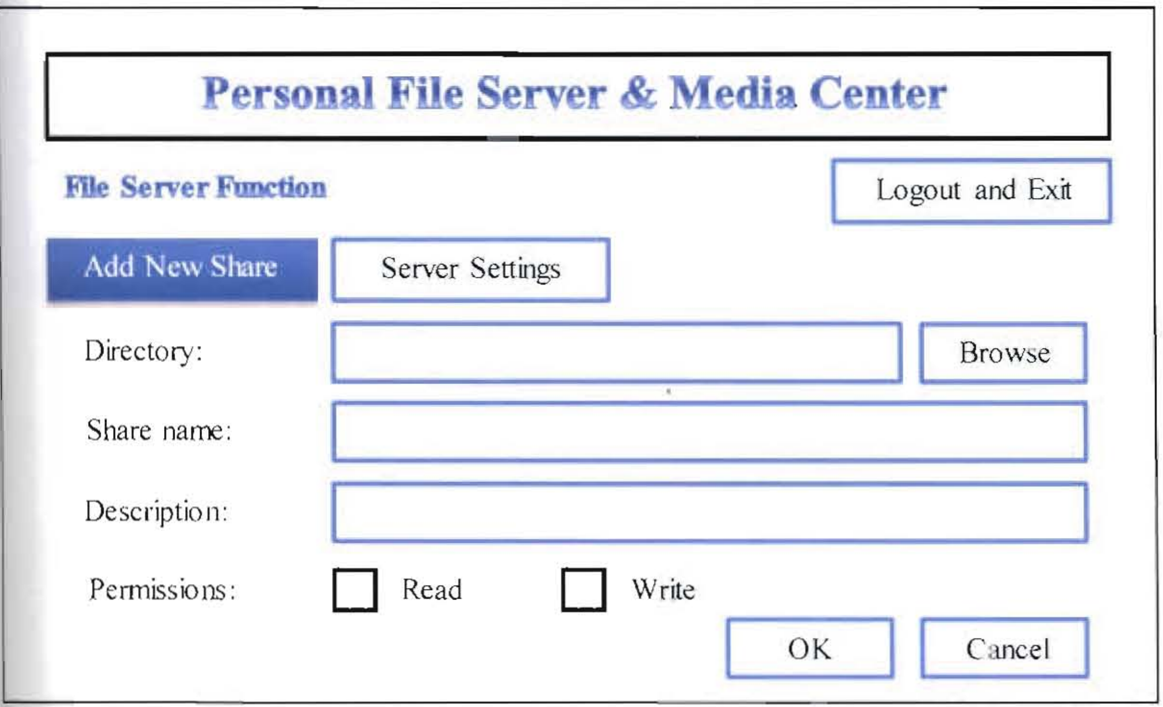
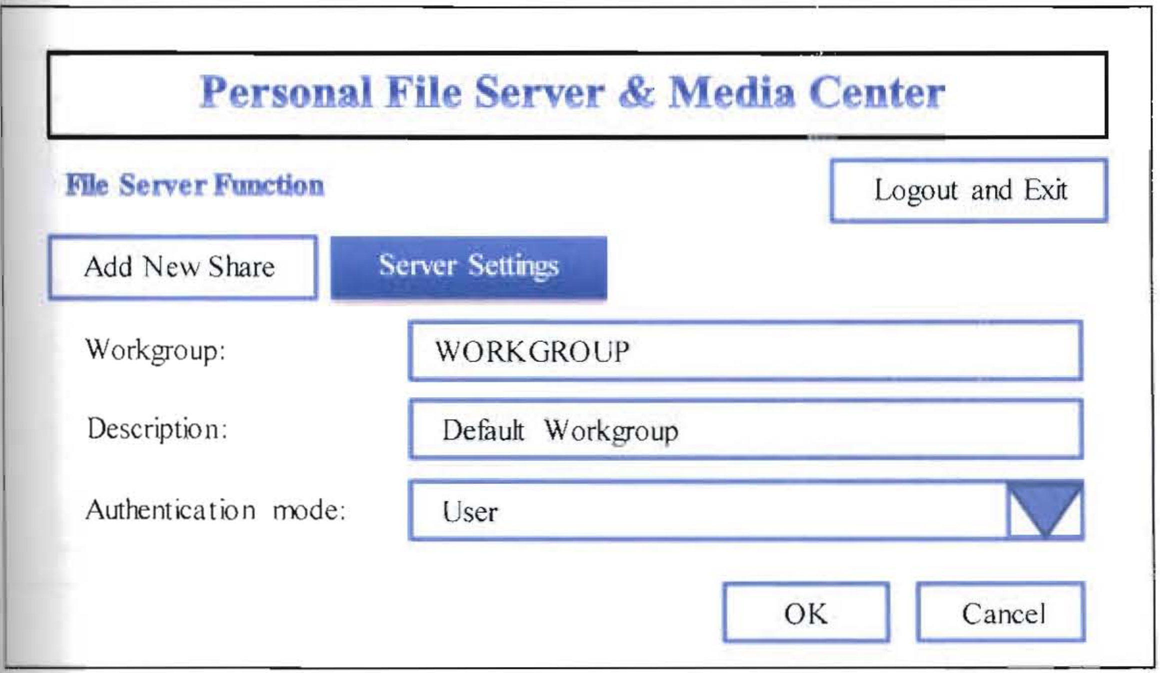


Figure 3.9: The interface for add new share for file server function.

User can then configure server settings by clicking the "Server Settings" button.



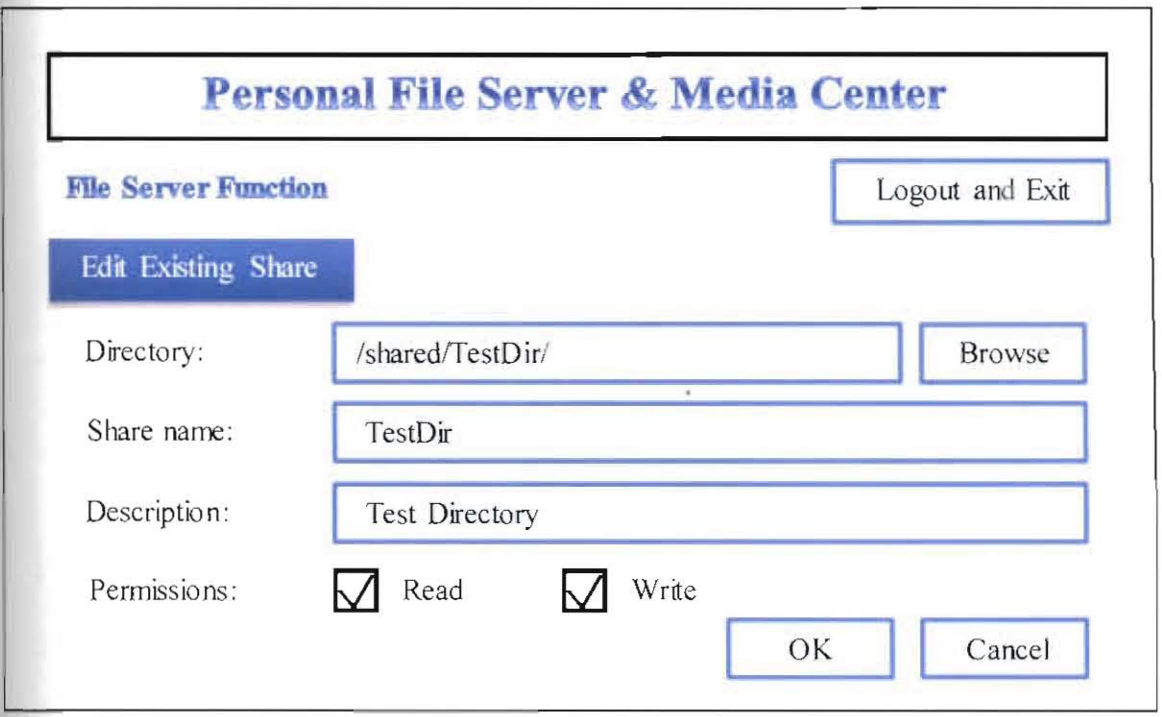
Workgroup: WORKGROUP

Description: Default Workgroup

Authentication mode: User

Figure 3.10: The interface for server settings in file server function.

Double-clicking any existing share on the list will bring user to existing share edit page.



Directory: /shared/TestDir/

Share name: TestDir

Description: Test Directory

Permissions: ☑ Read ☑ Write

Figure 3.11: The interface for editing existing share for file server function.

3.4.3.2 Interface for Media Center Function

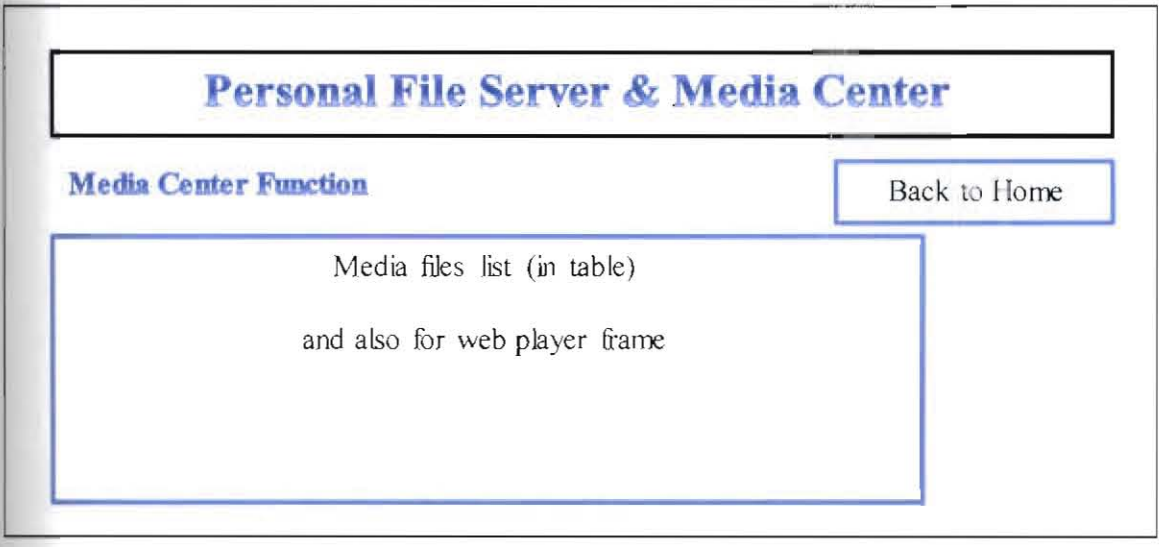

Personal File Server & Media Center
Media Center Function
Back to Home
Media files list (in table)
and also for web player frame

Figure 3.12: The default interface for the media center function.

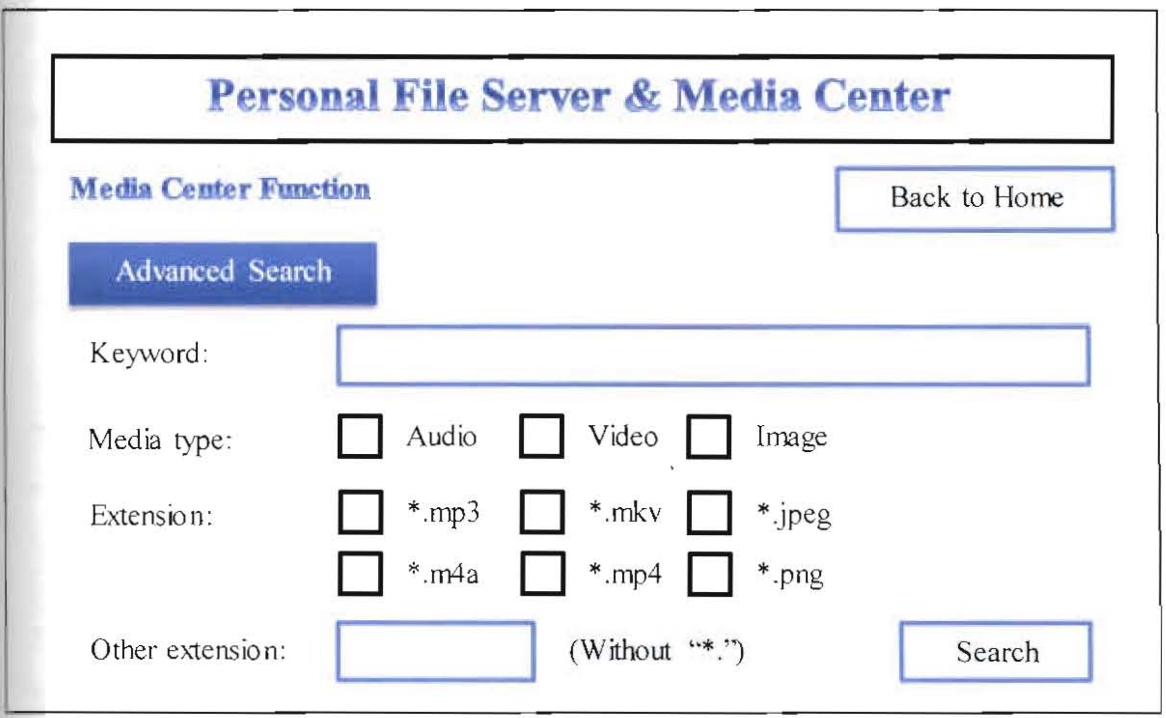


Figure 3.13: The interface for advanced search for media center function.

3.4.4 User-System Interaction

This subsection shows the interaction between the user and the system for each function.

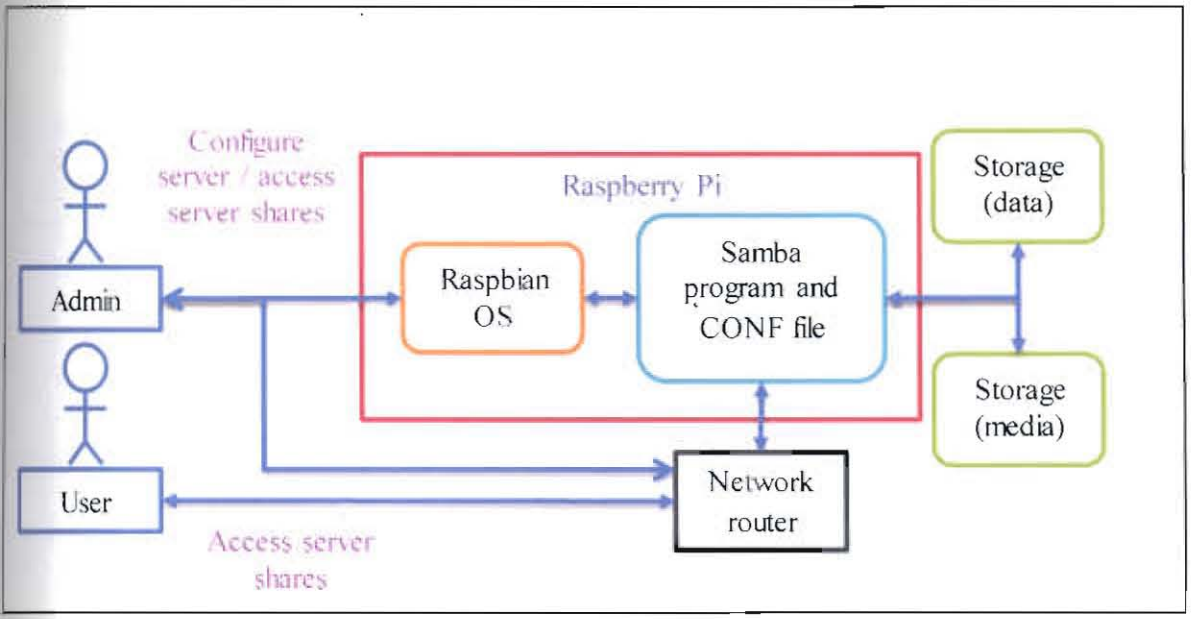


Figure 3.14: The nser-system interaction for file server function (direct).

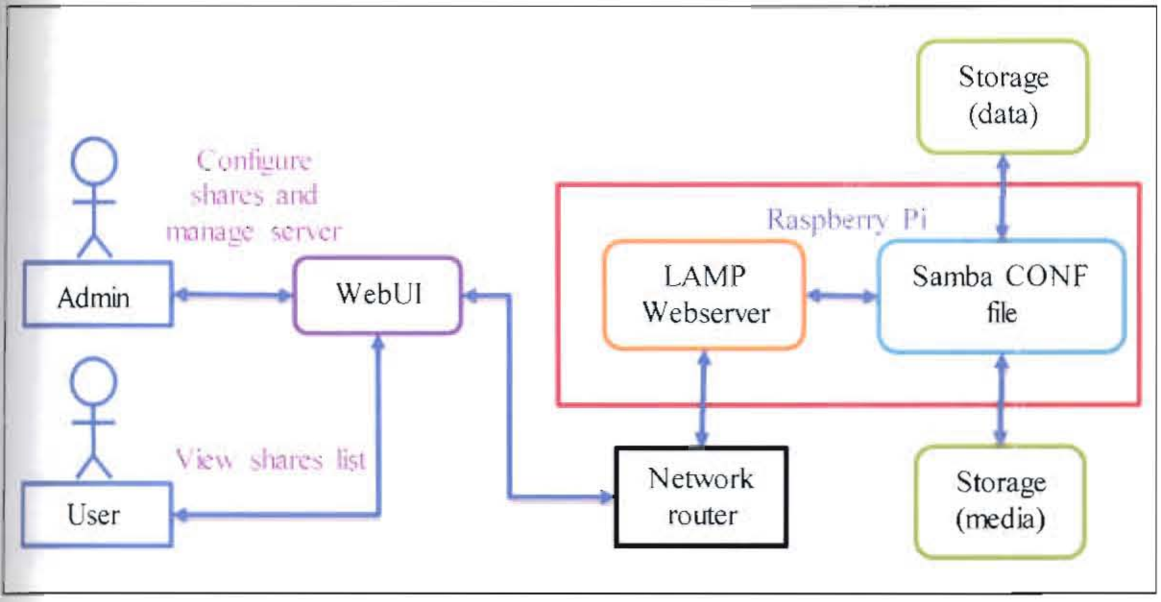


Figure 3.15: The user-system interaction for file server function (using WebUI).

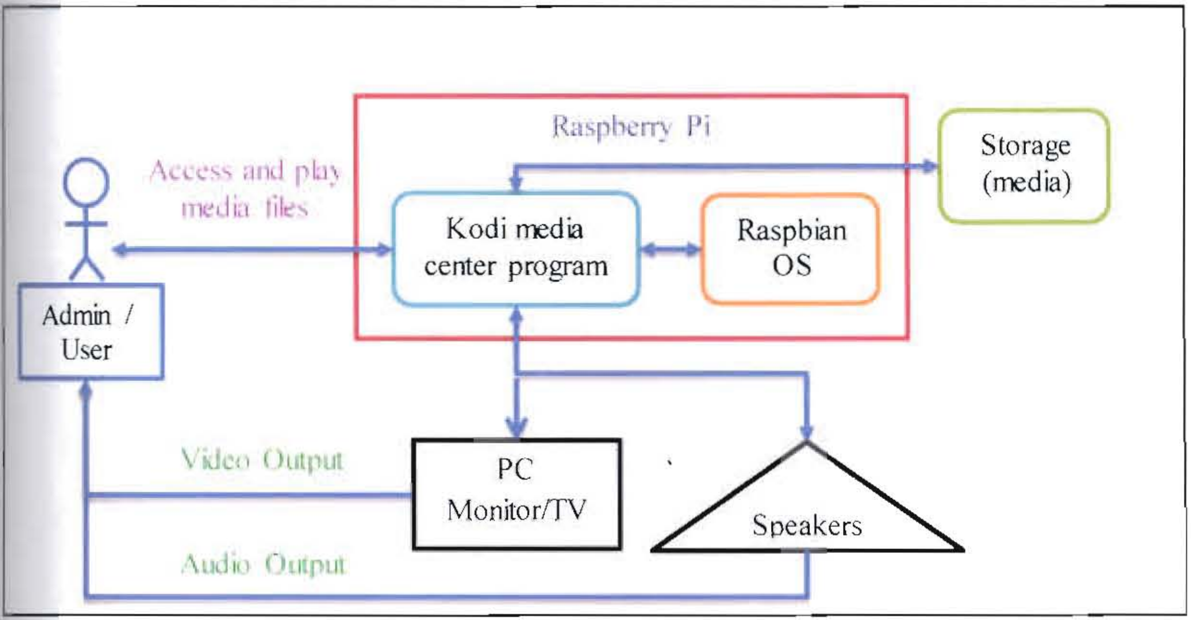


Figure 3.16: The user-system interaction for media center function (using Kodi).

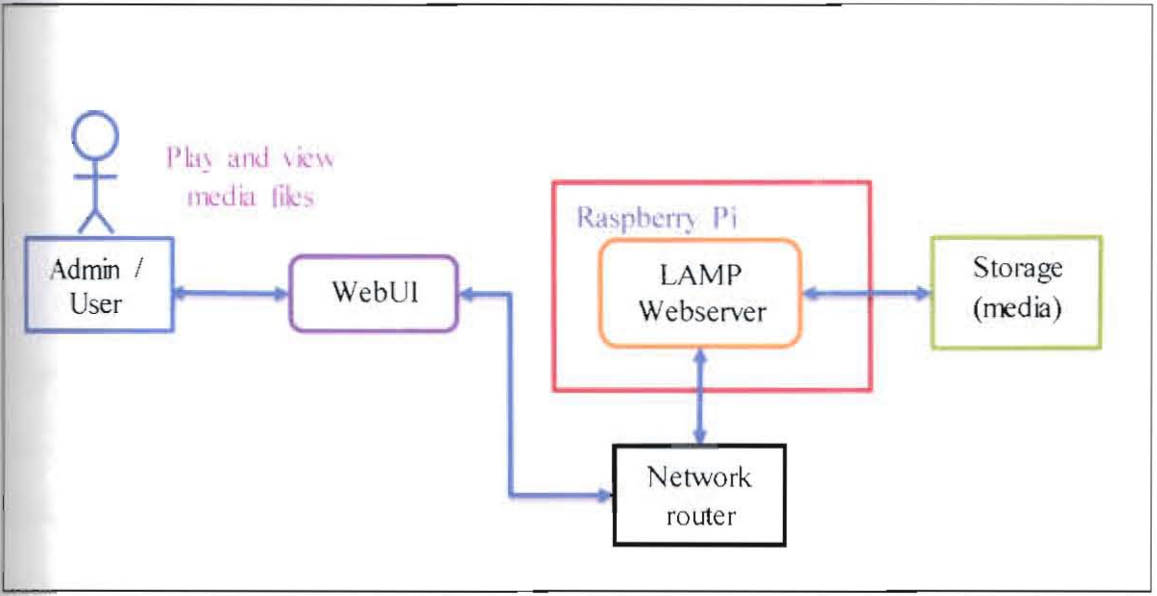


Figure 3.17: The user-system interaction for media center function (using WebUI).

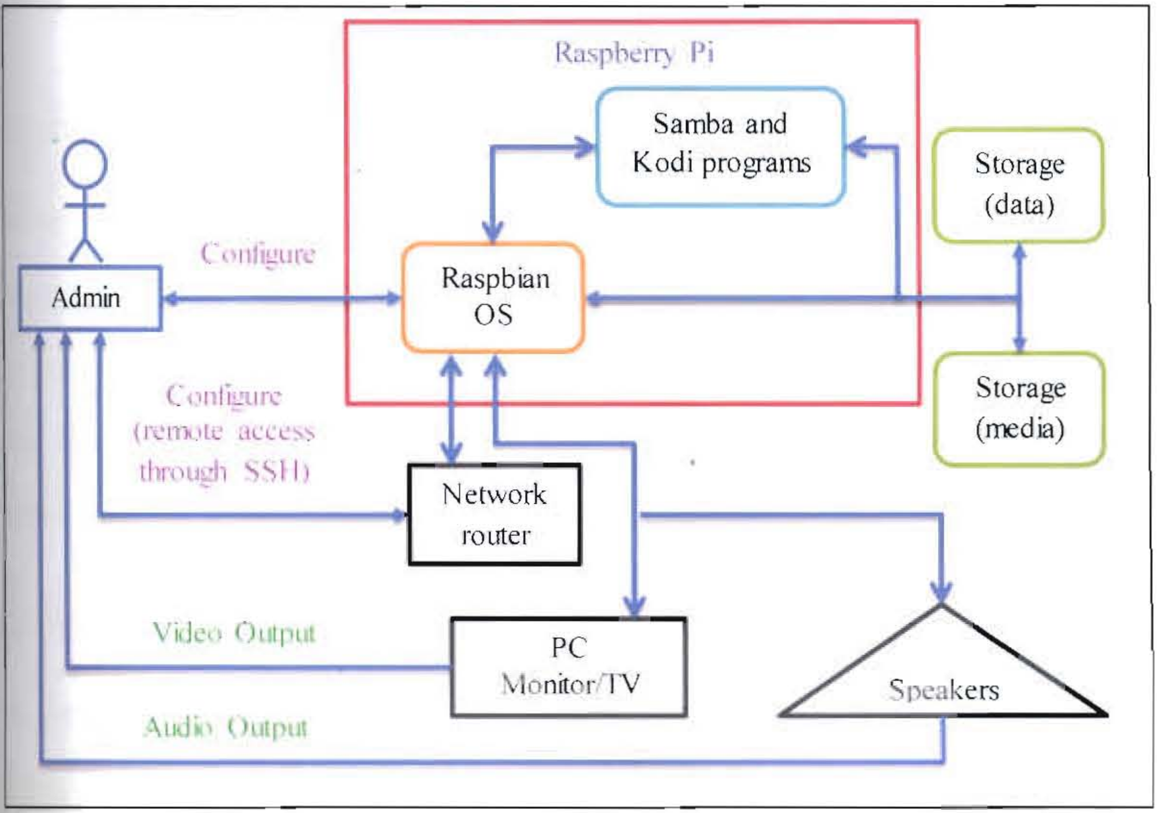


Figure 3.18: The user-system interaction for system configuration and management.

3.5 Summary

This chapter has described in details the proposed methodology to develop the system for this project – the incremental development. It also briefly outlined the planned development for to deliver the functions and features of the system incrementally. This chapter also describes the hardware, software, user and system interface requirements of the proposed system. Aside from that, it contains the proposed system architecture and its schematics. This chapter also presented the designs for the web-based interface to integrate all the functions and features. This chapter also shows the user interaction with the system functions.

4 Chapter 4 – Implementation

4.1 Introduction

This section describes in details the implementation process for the proposed system in this project. The details include the development process of the system as a whole. This development process is divided into several phases as described earlier in Section 3.2. Each phase describes how the development process for that particular phase was done. Also included is the list of tools and devices used for implementation.

4.2 Development Phase 1: System Preparation

This phase is mainly to prepare the Raspberry Pi device ready for installation procedures in the next subsequent phases. Main objectives are to validate whether the Raspberry Pi device is working, update its operating system components with the latest version and also to connect the device with hardware and component devices. Once these are completed, the interconnection between the Raspberry Pi with the hardware devices and components are then verified before proceeding to the next development phase.

4.2.1 Installation Procedures

The installation procedures for this development phase are as described step-by-step below:

1) The Raspberry Pi device is inserted with MicroSD card containing Raspbian OS and encased in its plastic case to hold and cover the device.
2) Note that the MicroSD card being used already contains pre-installed Raspbian v3.12.18+ along with NOOBS (New-Out-Of-the-Box-Software) file. For details on how to install Raspbian OS from empty MicroSD card, please refer to setup guide section in system User Manual (included in System CD).

3) Next, the Pi device is connected with PC monitor through HDMI cable with HDMI to DVI adapter, keyboard and wireless mouse.

4) Next, the Pi device is plugged in and switched on. Then its operating system is loaded and the author logged in using default credential. Note that the default password is the same for all new Pi devices. The password is then later changed (highly recommended). The default login credential for the Pi device is:

```
//username = pi
//password = raspberry.
```

5) Once the desktop is loaded and displayed on PC monitor screen, the keyboard and wireless mouse are then tested and verified to be working correctly.

6) The Pi device is then connected with router through LAN cable and later wireless connection. Its connection is then verified by using *ifconfig* and *ping* command.

```
ifconfig
ping 192.168.1.254       //router gateway IP
```

7) Access to the Pi device from other device was also tested by using *ipconfig* and *ping* command from a laptop that is connected under the same network.

```
ipconfig
ping 192.168.1.65 //Raspberry Pi device default IP
```

8) The Pi device is then connected wirelessly to phone Wi-Fi hotspot. Internet access was tested and verified by *ping*-ing Google domain.

```
ping www.google.com
```

9) Once Internet access is verified, the Pi device is then updated using system command.

```
sudo apt-get update
```

4.2.2 Hardware and Components Setup

The initial setup between the Raspberry Pi and other devices are as shown in figure below:

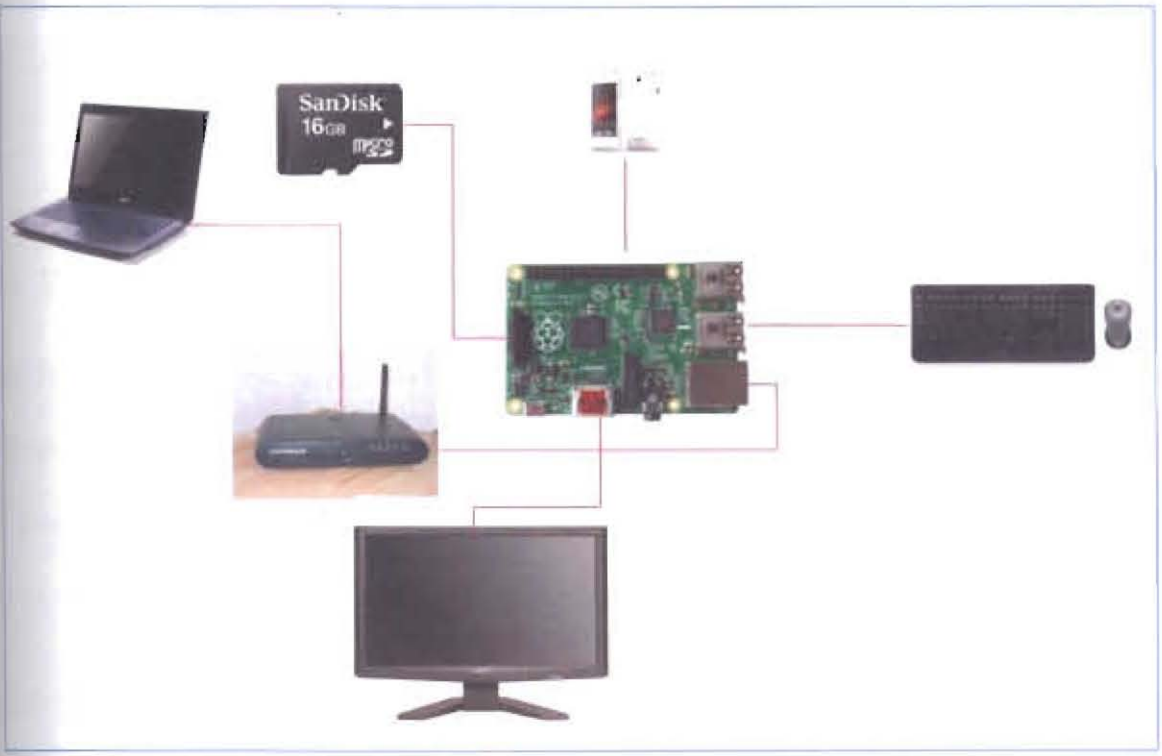


Figure 4.1: Initial setup of the system (Phase 1).

4.2.3 System Specification

The specification of the system at this phase is as described in table below:

Table 4.1: System components and their specification.

Name	Type	Model/Version	Description
Raspberry Pi	Device	Model B+	Central controller device
Modem/Router	Device	Speedtouch585 v6	Main router for interconnection
PC Monitor	Device	Acer x193HQ	Default output display for Rasp Pi
Laptop	Device	Acer Aspire 4750G	Default PC for access/programming
Keyboard	Device	Logitech K120	Default input device
Mouse	Device	Logitech M185	Default input device, wireless

Laptop	Device	Acer 4750G	Default PC for access/programming
Smartphone	Device	Sony Xperia E	Default Wi-Fi hotspot for Internet
Raspbian	Software	3.12.22+	Default OS for Rasp Pi
HDMI cable	Cable	Standard HDMI	For video output to monitor/screen
LAN cables	Cable	Standard LAN	For physical connection with router
HDMI to DVI	Adapter	F-M adapter	For connection with PC monitor

4.3 Development Phase 2: File Server Installation

This phase is mainly to equip the Raspberry Pi device with file server capability. Main objective are to install Samba program for file server function and to install GAdmin-Samba program to manage and configure the Samba program. Once these two programs are installed, Samba will then be configured and its file server capability will be verified and tested.

4.3.1 Installation Procedures

The installation procedures for this development phase are as described step-by-step below:

1) Samba program and Samba essential components are being installed by using terminal commands; the installation packages are downloaded from the Internet through Wi-Fi hotspot connection provided by smartphone.

```
sudo apt-get install samba          //file size = 8179 KB
sudo apt-get install samba-common
```

2) The same procedure as in (1) is repeated for installation of GAdmin-Samba program.

```
sudo apt-get install gadmin-samba
```

3) Once they are both installed, a backup of Samba config file (*smb.conf* at directory */etc/samba/*) is created (file name: *smb_BACKUP-ORI.conf*). The original, default content of *smb.conf* is as shown in Appendix section 8.1.

4) The router default IP range, gateway and nameserver are checked and noted using *ifconfig* command and Pi also by reading *resolv.conf* file (*/etc/resolv.conf*).

```
sudo nano /etc/resolv.conf
```

5) The Pi device default IP address is then checked using *ifconfig* command and is shown to be 192.168.1.65.

6) The Pi device default LAN config is then modified into static configuration by editing its interfaces system file (*/etc/network/interfaces*).

```
sudo nano /etc/network/interfaces
//added these lines under 'iface eth0 inet static'
//address 192.168.1.65
//netmask 255.255.255.0
//gateway 192.168.1.254
```

7) Once done, the Samba config file (*smb.conf*) is then opened and configured both by using GAdmin-Samba program and manually using Leafpad, a text-editor program. Note that by using GAdmin-Samba, most key and values created by the program are default configurations that work for most system and do not need to be changed.

8) The server configurations under *global* section are then modified to enable the Samba file server to work with the router.

```
//configuration on smb.conf, under global section
//server string = Samba
//workgroup = MSHOME
//hosts allow = 127. 192.168.1.
//interfaces = 127.0.0.1/8 192.168.1.65
//remote announce = 192.168.1.255
```

9) Once done, a test share folder is created (*/test/share-public*) to test whether the Samba file server is working. Using GAdmin-Samba program, the test folder is then configured to be a public share with read and write permissions. Once everything is done, Samba file server is restarted.

10) To verify that the folder sharing is working, a laptop is connected to the same router using LAN cable, its IP is 192.168.1.66. The interconnection between the laptop and the Pi device is then verified using *ping* command from both devices.

```
//ping laptop from Pi device
ping 192.168.1.66
//ping Pi device from laptop
ping 192.168.1.65
```

11) Once the connection between them is verified, the shared folder is then checked and appeared under laptop (Win7) *Networks* list, under Samba. Inside Samba is the *test-public* folder that can be accessed without password (public share). The read and write capability is then verified when a new folder was able to be created inside it.

12) The sharing capabilities are then further tested with different access and permissions such as read-only, non-public, and visibility. Their access and permissions variant are all verified using laptop and also smartphone (through *ES File Explorer* app).

4.3.2 File Server Specification

The specification of the file server function at this phase is as described in table below:

Table 4.2: File server components and their specification.

Name	Type	Model/Version	Description
Raspberry Pi	Device	Model B+	Central controller device
Raspbian	Software	3.12.22+	Default OS for Rasp Pi
Samba	Software	3.6.6	Default file server program
GAdmin-Samba	Software	0.2.9	For Samba server configuration

As for full Samba configuration settings, the full content of *smb.conf* is in Appendix 8.2.

4.4 Development Phase 3: Media Center Installation

This phase is mainly to equip the Raspberry Pi device with media center capability. Main objective is to install Kodi program for media center function. Once Kodi is installed, it will then be configured, its compatibility with Raspberry Pi device and its operating system will be tested and Kodi functions and features will be explored.

4.4.1 Installation Procedures

The installation procedures for this development phase are as described step-by-step below:

1) The first step is to install Kodi program. However, Kodi did not yet have official repositories under Rasbian list that it cannot simply be installed using normal *sudo apt-get install* command. Note: as of May 2015, Kodi now has official repositories under Rapsbian list.

```
sudo apt-get install kodi      //error: "kodi" not found
```

2) To install it, several other commands had to be entered and Kodi was later able to be downloaded through unofficial source and then installed successfully.

```
sudo apt-get install python-software-properties
sudo add-apt-repository ppa:team-xbmc/ppa
sudo apt-get update
sudo nano /etc/apt/sources/list.d/mene.list
//added this line at the end of the list:
//"deb http://archive.mene.za.net/raspbian wheezy contrib"
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-key 5243CDBD
sudo apt-get install kodi      //working, file size = 51.0 MB
```

3) Once Kodi is installed, the program is then started from system menu. Kodi managed to be executed and run normally. This in a way proved that Kodi is compatible with the Pi device and its Raspbian OS.

4) Kodi functions and features are then explored and tested in terms of media files playback (audio, video and images) and its default settings are also modified slightly.
5) Note that there is an unresolved bug/error upon exiting Kodi which cause the desktop screen to turn black and input from keyboard and mouse seems to be unresponsive. This is apparently due to the Raspbian GUI failed to reload again after exiting a full-screen Kodi program. The only way to get access back is either by clicking *Ctrl + Alt + F1* combination keys to enter console-only GUI or simply restart the Pi device.

4.4.2 Media Center Specification

The specification of the media center function at this phase is as described in table below:

Table 4.3: Media center components and their specification.

Name	Type	Model/Version	Description
Raspberry Pi	Device	Model B+	Central controller device
PC Monitor	Device	Acer x193HQ	Default output display for Rasp Pi
Stereo speakers	Device	Logitech Z103	Default audio output
Raspbian	Software	3.12.22+	Default OS for Rasp Pi
Kodi	Software	14.1	Default media center software
HDMI cable	Cable	Standard HDMI	For video output to monitor/screen
HDMI to DVI	Adapter	F-M adapter	For connection with PC monitor

4.5 Development Phase 4: Webserver Components Installation

This phase is mainly to equip the system with webserver components to make the system capable to act as a web server. Main objective is to install LAMP Webserver, which consists of Apache, MySQL and PHP components. Once the web server essential components are installed, it will be then tested and verified.

4.5.1 Installation Procedures

The installation procedures for this development phase are as described step-by-step below:

1) The LAMP webserver components are installed one by one through terminal commands. All components are downloaded through Internet access provided by smartphone Wi-Fi hotspot and installed automatically once download is finished.

```
sudo apt-get update     //update system first
sudo apt-get install apache2
sudo apt-get install mysql-server
sudo apt-get install php5
sudo apt-get install php5-mysql
sudo apt-get install phpmyadmin
```

2) Once a component is downloaded, no further configuration is needed to complete installation except for *phpMyAdmin* component. During installation, *phpMyAdmin* is set to be automatically run by *Apache2* program, password for access is prompted and defined here and new database was installed and configured automatically using *dbconfig-common* configuration option.

3) After that, *Apache2* config file is then modified *(/etc/apache2/apache2.conf)*. Once done, the system is then rebooted.

```
sudo nano /etc/apache2/apache2.conf
//added this to the last line :
//"Include etc/phpmyadmin/apache.conf"
```

4) Once *phpMyAdmin* is successfully installed, it is then accessed through a web browser and configured.

```
//phpmyadmin login:
//username: root
//password: <the password defined during installation>
//phpmyadmin URL access: http://raspberrypi/phpmyadmin/
```

5) Once access to *phpMyAdmin* through web browser is verified, its webserver function is then tested. This is done by accessing its default webserver *index.html* page (in /var/www/) through a web browser. The default webpage was verified to be accessible from the Pi device itself (through localhost), and both from laptop and smartphone access (through accessing default Pi device IP address).

```
//URL to access index.html from Pi device:
http://localhost/
//URL to access index.html from laptop/phone
http://192.168.1.65      //default Pi device IP
```

4.5.2 Webserver Specification

The specification of the webserver components at this phase is as described in table below:

Table 4.4: Webserver components and their specification.

Name	Type	Model/Version	Description
Raspberry Pi	Device	Model B+	Central controller device
Raspbian	Software	3.12.22+	Default OS for Rasp Pi
Apache	Software	2.2.22 (Debian)	Part of LAMP webserver
MySQL	Software	14.14, dist 5.5.41	Part of LAMP webserver
PHP5	Software	5.4.39	Part of LAMP webserver

4.6 Development Phase 5: Web-based UI Implementation

This phase is mainly to equip the system with a Web-based interface, as an intermediate between the Pi device and user. Main objective is to develop a Web-based interface for user interaction with the file server function. The webserver capability is realized by LAMP webserver which was installed previously in Phase 4. The Web-based UI main purpose is to provide access to file server configuration without direct access to the system. Aside from that, the Web-based interface was extended to include .secondary media center function, where users can view and play media files through web player and web image viewer.

4.6.1 Implementation Analysis and Planning

The process of configuring Samba server mainly involves the modification and configuration of *smb.conf* file, the Samba configuration file which is located inside */etc/samba/* folder. And thus, the nature of the configuration needs a system that can read and write or modify the the content of the file. The main idea is to do this through a Web-based interface, and in this context, it is by using a web page to do the configuration. This is possible to be done through the utilization of HTML, JavaScript, CSS and PHP technologies. In a way, the Web-based interface can be seen as a Web-app application that configures the system file server function.

Before the development of the Web-based interface can be initiated, it has to have a proper analysis and planning. During analysis process, extensive research has been done focusing on specific technical details of how the configuration will be achieved or whether it is even feasible. It was then found that PHP has various built-in functions to support read and write operations to external file but lack proper parser function that can parse configuration file like

Samba's CONF file. PHP does however have a built-in parser called *parse_ini_file()* to parse INI file, another type of configuration file.

What this parser function does is parses the content of a configuration file and put them in level 1 multi-associative array, and return the associative array. For more details, see its documentation as referenced by *php_ini_file* (n.d.). The structure of INI file and CONF file is the same since both consist of one or more sections that are enclosed in square brackets, e.g.: *[sectionName]* and have corresponding *key* and *value* under each section. The section names are unique but the key and values are usually not. And due to this, *parse_ini_file()* function can be used to parse Samba's CONF file too. The generic structure is as shown below:

```
[section1]
key1 = value1
key2 = value2

[section2]
key3 = value3
key4 = value4

[section3]
key5 = value5
```

Figure 4.2: The generic structure of INI and CONF file.

However, the built-in function has certain limitations when reading certain key values that contain reserved words: where values like `null`, `off`, no and `false` result in " " (blank), while values like `yes` and `true` result in "1" (*php_ini_file*, n.d.). This becomes a problem when parsing Samba's CONF file since it contains `yes` and no values and the content of the array being passed does not reflect the file actual content. However, due to performance consideration and time constraint, it is decided that *parse_ini_file()* function will be used, rather than creating a new parser function from scratch that may not even be parsing correctly.

And thus, it was mandatory to create PHP functions that can manipulate multi-associative array and change the values inside the array, regardless whether parsing Samba CONF file will be using built-in function or not. As for the displaying and modifying the content of the CONF file through web-browser, PHP, HTML, CSS and JavaScript technologies will be utilized to realize that.

4.6.2 Implementation Procedures

4.6.2.1 Webpages for Web-based UI

Table 4.5: Summary for the webpages for Web-based UI.

Criteria	Information
Total number of web pages	24
File types	23 PHP files, 1 HTML
Lines of code (LOC)	Range: 137 (min) – 754 (max)
Location	<webserver-folder>/SambaKodiPi/WebUI/

The implementation procedures for the development of the base webpages are as described step-by-step below:

1) Several bootstrap-based website templates were browsed, compared and downloaded.
2) It was then decided that SB-Admin-2.1.0.6 will be used as the website template mainly because of its sleek and attractive design along with various examples that showcasing its UI elements for future reference during coding process.
3) The template was then uploaded into Raspberry Pi default webserver folder (*/var/www/*) and the template was tested and verified to be displayed and working correctly and can be accessed from Pi localhost and from outside (laptop and phone).

4) The SB-Admin-2 template is then modified and several html and php webpages are created one by one by mainly by using Notepad++. The webpages were created based on the file server functionality. The full list are as listed in table below:

Table 4.6: The list of web pages for WebUI component.

No	Page name	Description
1	index.php	The default, index page for this Web-based UI site.
2	login.php	The login page to log in to the system.
3	register.php	The page for new users to register for new account.
4	about.php	The page to briefly explain what the website is all about.
5	contact.php	The page to display contact information and contact form.
6	home.php	The home page for registered user after they are logged in.
7	system.php	The page to display the whole system information as a whole.
8	system-config.php	The page to configure the system as a whole.
9	server.php	The page to display Samba server information.
10	server-shares.php	The page to display and add, edit and delete Samba shares.
11	server-config.php	The page to display and edit Samba server configuration.
12	media.php	The page to display media library information.
13	media-music.php	The page to play music files stored in the system storage.
14	media-video.php	The page to play video files stored in the system storage.
15	media-pictures.php	The page to view images stored in the system storage.
16	users-new.php	The page to manage new user application.
17	users-manage.php	The page to manage existing user accounts.
18	comments.php	The page to give comments and feedbacks for this system.
19	comments-view.php	The page to view comments and feedbacks on this system.
20	profile.php	The page to manage user profile information.
21	account.php	The page to manage user login information.
22	error-access.php	The page to display access error (user not logged in).
23	error-permission.php	The page to display permission error (user is not an admin).
24	index.html	The page to automatically redirect user to index.php.

5) An official logo for the WebUI was also created using Adobe Photoshop CS5, and it was during this step that the system decided to be named *SambaKodiPi*, in honor to Raspberry Pi and the two main programs that power the system main functionalities.
6) Over time, the webpages are being modified one by one as file server functions are being developed iteratively and incrementally by utilizing PHP, JavaScript, HTML and CSS technologies. Due to the high number of total pages and long lines of (some are redundant) code for each page, the resulting final source code for the WebUI pages are included as softcopy. The screenshots for each page however, are as shown in Appendix Section 8.3.

4.6.2.2 The PHP functions for WebUI base operations

Table 4.7: Summary for the file handling WebUI base operations.

Criteria	Information
File name	SKP-Base.php
Location	<webserver-folder>/SambaKodiPi/functions/
Description	For WebUI session initialization, login, user authentication, dbase query, access restriction, and for all other basic WebUI functions
Content details	28 global functions
Lines of code (LOC)	567

The implementation procedures for the development of the PHP functions for WebUI base operations are as described step-by-step below:

1) Before the WebUI operations start to be implemented, a new database called *SambaKodiPi* is created through *phpMyAdmin* page (*http://localhost/phpmyadmin/*). Inside it is the new user table with manually inserted value.

2) Once done, a new privilege is created for user called *"fyp"* and its password and granted specific access to the *SambaKodiPi* database created previously. This user credential will be used as login connection to connect to the database later.
3) The initial base PHP function was then created, specifically the login code section to verify user login into the system. The code section that works together with MySQL was implemented in *login.php* page by using Dreamweaver CS5. The login code is then tested by logging in into the system through the WebUI page and verified once the page manage to redirect user to *home.php* page upon successful login.
4) Next is the code section for logout action, also created using Dreamweaver CS5, to logout user from the system. Once done, the logout code is then tested and verified once user is redirected back to *login.php* page upon successful logout.
5) Next is the code section to extract user information (name and account type), in order to display them on the webpage under user icon section, also created using Dreamweaver CS5. The code section is then verified when it displays the user information correctly.
6) That concludes the basic base operations for the WebUI pages. However, following the capability of PHP functions to be working inside a single file, it was decided that the WebUI base operations are to be placed inside a single file too, called *SKP-Base.php*. The main reason for this change is for readability and maintenance purposes for the source code section inside WebUI pages. Doing this also heavily reduce code redundancy of similar source code inside each page.
7) In order to put the PHP base operations inside external file, the logic behind the code generated by Dreamweaver need to be fully understood as it is structural. And thus,

putting them inside an external file then required modification and creation of new functions so that the code section can be called from each WebUI page.

8) All the base operations are then placed inside their own function, which now accepts parameter arguments so that they can be called and used by each WebUI page. These functions are then later extended to include specific database query for user authentication and also access restriction. Due to the high number of LOCs, the final source code for the *SKP-Base.php* file that handles the WebUI base operations is not included in this report, but included as softcopy.

4.6.2.3 The PHP functions for File Server operations

Table 4.8: Summary for the file handling WebUI file server operations.

Criteria	Information
File name	SKP-FS.php
Location	<webserver-folder>/SambaKodiPi/functions/
Description	For manipulation of Samba configuration file (smb.conf) and its content and other operations used by WebUI file server function
Content details	22 global functions, 19 class functions, total of 41
Line of codes (LOCs)	891

The implementation procedures for the development of the PHP functions for Samba CONF manipulation are as described step-by-step below:

1) The initial PHP functions created were the ones that access values in normal array. They later extended to value modification in normal array. All functions were created based mainly from examples from PHP documentation (*php.net*) and other typical sources such as *stackoverflow.com* website.
2) Once the basic logic was implemented, new functions were then created to handle normal associative array. Once they were working fine, a discontinuous and very time consuming effort was given to extend the functions in order to support manipulation of multi-associative array as well, especially level 1 multi-associative array, since it is of that structure of CONF file.
3) Once the functions are implemented, they were being tested and modified accordingly by processing dummy INI and CONF file, working along with *parse_ini_file()* function to parse the content. Once the functions are verified to be accessing the content properly, it is then tested by parsing the real Samba CONF file.

4) Extra functions then created to modify values of yes and no inside Samba CONF file that are being displayed as *"1"* and "" (empty) inside array that is being passed by *parse_ini_file()* function. This in turn ensures that the values inside the array match that of actual Samba CONF values. At this point, the whole parsing process involves parsing using *parse_ini_file()* function and manual editing of array values.

5) Once the parsing process are verified to be working correctly, only then other functions are created and WebUI development focus back on "integrating" and using these PHP functions to display and provide a way to configure the Samba CONF content through the WebUI.

6) As the implementation for WebUI progressed, more PHP functions were created whenever necessary to accommodate the functions needed to implement file server configuration. For proper utilization, some of the functions were extended again into a class called *SambaCONF*, so that it is easier to be called and use its functions once it is initialized as object. In a way, the implementation of the *SambaCONF* class also utilized the concept of Object-Oriented Programming.

7) The PHP functions are then grouped together into one PHP file and put under functions folder so that all WebUI pages can utilize them. Note that the source code for the WebUI pages kept changing over time as more features are being implemented. Due to the high number of lines of codes (LOC), the final source code for *SKP-FS.php* file that handles the WebUI file server operations is not included in this report, but included as softcopy.

4.6.2.4 The PHP functions for Media Center operations

Table 4.9: Summary for the file handling WebUI media center operations.

Criteria	Information
File name	SKP-MC.php
Location	<webserver-folder>/SambaKodiPi/functions/
Description	For manipulation and data retrieval from media directory and other operations used by WebUI media center function
Total number of functions	13
Line of codes (LOCs)	327

The implementation procedures for the development of the PHP functions for WebUI media center operations are as described step-by-step below:

1) The initial PHP functions created was the one that scans directory and return the information in that directory. They later extended to recursive directory scanning based on passed path parameter and return the directory information in an array. Some of the concepts utilized in the functions were created based on examples from PHP documentation (*php.net*) and other typical sources such as *stackoverflow.com* website.
2) Once the recursive directory scanning operations are working, new function was then created to filter the directory for specific file type based on a specified extension value. Once this function is tested and working, it is then extended to filter for several file extensions at once, where the extensions are stored in an array, passed as a parameter.
3) Once the functions are tested to be working together, new function was then created that combines the concepts of both recursive directory scanning and file extensions filtration. This was done so that all of the operations for directory scanning for specific file are stored and handled by one function. This function was then tested and proved to be working as intended.

4) After that, other functions were created to help with debugging. As the implementation for WebUI media center function progressed, more PHP functions were created to accommodate the functions needed to implement the WebUI media center function.
5) Aside from that, other functions were also added to retrieve summary of media files information. Due to the high number of LOCs, the final source code for *SKP-MC.php* file that handles the WebUI media center operations is not included in this report, but included as softcopy.

4.6.3 Web-based UI Specification

The specification for the WebUI components at this phase is as described in table below:

Table 4.10: Web-based UI components and their specification.

Name	Type	Version	Description
Boostrap CSS	CSS	v3.3.2	Base CSS for WebUI
Boostrap JS	JavaScript	v3.3.2	Base JS for WebUI
Font Awesome	Font	v4.3.0	Custom icon font for WebUI
Font Awesome.css	CSS	v4.3.0	Custom CSS font for WebUI
Glyphicon Regular	Font	v1.009	Custom icon font for WebUI
JQuery.js	JavaScript	v2.1.3	Base JQuery JS for WebUI
SambaKodiPi.css	CSS	v1.0	Custom CSS for WebUI
SambaKodiPi.js	JavaScript	v1.0	Custom JS for WebUI
SambaKodiPi.js	JavaScript	v1.0	Custom JS for WebUI
SambaKodiPi.php	PHP	v1.0	Base PHP connection to database
SB-Admin-2	Template	v2.1.0.6	Base website template for WebUI
SKP-Base.php	PHP	v1.0	PHP file, for WebUI base operations
SKP-FS.php	PHP	v1.0	PHP file, for WebUI file server operations
SKP-MC.php	PHP	v1.0	PHP file, for WebUI media center operations
Other CSS/JS files	CSS/JS	vary	CSS/JS for SB-Admin-2 bootstrap template

4.7 Development Phase 6: System Integration and Validation

This phase is mainly to integrate the file server and media center components in the Raspberry Pi with the WebUI component. Main objectives are:

a) To validate that the WebUI configuration function is working and capable to modify the file server shares and server settings as intended

b) To validate that the WebUI media center function is working and capable to access, read and play the media files stored in the media. folder

c) To validate that the file server and media center components are compatible with the WebUI component and between each other.

The validation processes stated above were done by testing the WebUI functions on the real LAMP server in the Raspberry Pi. Most of the testing processes were done in conjunction with the System Testing activities. During the testing process, several errors and bugs were discovered and then fixed by modifying the source code of the WebUI component and modifying certain file systems and permissions in Raspberry Pi system. In the end, the WebUI component was successfully integrated with the Raspberry Pi and compatible with the file server component and media center component.

The WebUI configuration function was validated to be working and capable to read Samba configuration file (*smb.conf*) and display its content correctly on the WebUI page. It was also validated to be capable to modify the file server shares and server settings correctly. The WebUI media center function was also validated to be able to access and read media files stored in media folder and capable to play and display them correctly on the WebUI page.

4.8 Other Tools and Devices Used for Implementation

Below is the list of other tools and devices that were used to develop the system.

Table 4.11: Details of other tools and devices used during implementation process.

Name	Type	Model/Version	Description
PC Monitor	Device	Acer x193HQ	Used for coding WebUI pages
Laptop	Device	Aspire 4750G	Default PC for coding WebUI
Windows 7	OS	Home, SP1	Default OS for laptop PC
Keyboard	Device	Logitech K120	Used for coding WebUI
Mouse	Device	AVF AGM-X4	Used for coding WebUI
Smartphone	Device	Sony Xperia E	Default Wi-Fi hotspot for Internet
USB drive	Device	DT G2 8GB	Default USB drive for data transfer
Dreamweaver	Software	CS5.5 v11.5	Used for coding WebUI
Notepad++ (Win)	Software	6.5.5	Used for coding WebUI
Geany (Linux)	Software	1.22	Used for coding WebUI
DiffMerge	Software	4.2.0	Used for comparing WebUI codes
WinRAR	Software	3.5.1	Used for archiving WebUI backups
Mozilla Firefox	Software	36.0	Used for viewing and testing WebUI
Google Chrome	Software	41.0.2272.101 m	Used for viewing and testing WebUI
Smartphone	Device	Sony Xperia Z1	Used for viewing and testing WebUI
LAN cables	Cable	Standard LAN	For physical connection with router
VGA to VGA	Adapter	M-M adapter	For Pi connection with PC monitor
External HDD	Device	WD 500GB	External storage for backup purposes

5 Chapter 5 – Testing

5.1 Introduction

This section describes in details the testing activities for the system that is being implemented in this project. The testing activities were done based on ISTQB's testing principles, using concept from the ISO 9126 standards as reference (*Software Life Cycle*, 2011). And thus, the system testing for this system covers four main test levels:

a) Component testing

b) Integration testing

c) System testing

d) Acceptance testing

Each test level may consist of several sub-testing levels. Each level consists of description about the procedures on how the testing activities were done along with their results. Testing activities involves both static and dynamic testing for the validation and verification of the final system. Overall testing process focuses on measuring the quality of the system in terms of both its functional and non-functional requirements.

5.2 Component Testing

Component testing is essentially the first testing level, focuses on testing the smallest units of a software product. Main objective is to ensure that the system has the correct and complete functionality based on its requirements. Component testing for this system involves unit testing for the units of each hardware/software or device that makes up file server component, media center component, WebUI component and the system as a whole.

Unit testing, in general, concerns with the testing of individual software units or object classes that make up a software program and testing is focused on their functionality. In the context of this project, unit testing focused on three main areas. Unit testing activities were performed on each of the areas by using both static and dynamic testing methods. Due to this, unit testing was essentially done using white-box test techniques. The testing areas are:

1) The functionality of each hardware/software component and electronic device
2) The functionality of each WebUI page
3) The functionality of individual PHP functions that handle base operations, file server operations and media center operations

5.2.1 Unit testing for hardware/software components

The hardware/software components and electronic devices in this context are those that were used during implementation phase as described in Chapter 4. Hardware components were inspected physically while software components were inspected by checking their installation condition, both for any defects and later their functionality was tested and verified one by one during assembling process throughout the implementation process. After the testing, all components and devices were verified to be in good condition and working correctly. The result for this unit testing is as shown in Appendix Section 8.4.

5.2.2 Unit testing for WebUI web pages

For WebUI web pages, unit testing was performed by inspecting the base source code for each page for their structure and indentations. The functionality and correctness of each page then later tested by viewing them using web browsers (Mozilla Firefox and Google Chrome). The syntax and correctness of each page source code were then verified by inspecting the page source by using web browsers *"View page source"* built-in function. This function is capable to highlight incorrect or missing HTML tags as well as syntax error of HTML codes or improper use of HTML attributes and elements. After the testing, all WebUI pages were verified to have proper structure and indentations, have no syntax error in tags or improper use of HTML elements and displayed correctly on web browser. The result for this unit testing is as shown in Appendix Section 8.4.

5.2.3 Unit testing for PHP functions

For PHP functions, unit testing was performed by inspecting the source code of each of the PHP functions in three separate files namely: *SKP-Base.php*, *SKP-FS.php* and *SKP-MC.php*. The functionality and correctness of each function were tested by inspecting the code line by line for correct structure and indentations and by executing the functions and class functions using test parameters and dummy arrays. All the dynamic testing activities were done by using *test.php* page, an additional page created just for testing various HTML, JavaScript and PHP functions before they were "transferred" and used in the actual WebUI page.

During the testing, some functions were found to be incorrect and they were fixed and re-tested again until they are working as intended. After the testing, all PHP functions for base, file server and media center operations were verified to be coded correctly in terms of structure, parameters, and function calling and thus verified to be working correctly. The results for this unit testing are as shown in Appendix Section 8.4.

5.3 Integration Testing

Integration testing is essentially the second test level after component testing. At this level, it is assumed that the test objects for this level are already tested in component testing and their bugs and errors have already been fixed. The main objective of integration testing is to verify the interaction and interconnection between system devices and hardware components and the functionalities of the integrated components: the Raspberry Pi controller device, the file server, the media center and the Web-based UI.

In the context of this project, component testing focuses on three main areas:

1) The functionality of each of the integrated components (the Raspberry Pi component, File Server component, Media Center component and LAMP webserver component)
2) The functionality of WebUI integrated component as a whole (WebUI pages and the integration with its CSS files, JavaScript files and PHP function files
3) The interaction between all the integrated components that make up the SambaKodiPi system as a whole

5.3.1 Functionality testing of integrated components inside Raspberry Pi

For (1), the Raspberry Pi integrated components were inspected physically for any physical defects. For other components, the sub-components that make up the component were tested and verified based on certain test criteria during their integration process throughout the implementation phase. After the testing, all integrated components were verified to be in good condition and working correctly. Full results for this testing are described in details in Table 8.6 in Appendix Section 8.5.

5.3.2 Functionality testing of WebUI integrated components

For (2), the WebUI integrated components were tested based on certain criteria determined by the author. The criteria include inspecting all webpage files for completeness and existence, reviewing the extended source code that use or call any CSS, JavaScript, or PHP code or function. The integration between HTML, CSS, JavaScript and PHP was also verified by using Firefox web browser *"View page source"* function. The functionality of it as a whole was then verified by how each page displays the end results on web browsers, as intended and without any error. Full results for this testing are described in details in Table 8.6 in Appendix Section 8.5.

5.3.3 Integration testing between all integrated components

For (3), the interaction between the integrated components was tested once they are all integrated together to make up the resulting SambaKodiPi system in this project. The pre-condition for this is when the WebUI component is successfully installed and configured in the LAMP webserver folder. The interactions between them were tested based on certain criteria determined by the author. Full results for this testing are described in details in Table 8.7 in Appendix Section 8.5.

5.4 System Testing

System testing is the third test level after integration testing. Main objective is to verify that the SambaKodiPi system implemented meets its specified requirements. At this level, it is assumed that the system is now fully assembled and has passed integration testing as a whole. In the context of this project, system testing test object is the SambaKodiPi system as a whole. System testing can verify that an implemented system fulfills its requirements based on two separate classes of requirements:

a) Functional requirements

b) Non-functional requirements

In order to cover for both functional and non-functional requirements for the SambaKodiPi system, it is tested based on software quality characteristics as stated by ISO 9126 standard. Thus, system testing covers the following three main characteristics:

1) Functionality

   Covers functional testing, security testing, accuracy testing and interoperability testing

2) Reliability

   Covers reliability testing

3) Efficiency

   Covers performance testing

5.4.1 Testing Functionality

5.4.1.1 Functional testing

Description:

Functional testing focuses on testing the functions of a system. In this testing, system requirements and use cases are used as test basis to test the functionality of the final system.

Procedures:

Testing activities involved testing and verifying that the functionality of each of the system requirements and use cases are fully implemented.

Results:

The final system passes all functional test cases as it has all base functions based on its specified requirements and use cases. The final system also includes additional (secondary) media center function which was not originally specified in its base requirements. Full results for functional testing are described in details in Table 8.8 in Appendix Section 8.6.

5.4.1.2 Security testing

Description:

Security testing focuses on testing the access authorizations of a system. In this testing, the accessibility restriction of the system is used as test basis to test the security of the system.

Procedures:

Testing activities involved testing the access restriction of the WebUI pages if user is not logged in and access restriction if user tries to access restricted page/content specifically intended for admin only. Also includes the testing of access restriction for WebUI subfolders that store sensitive information.

Results:

The final system passes all security test cases. Access to certain WebUI subfolders is properly restricted, validating that the Apache configuration is proper and *.htaccess* file is working. Full results for security testing are described in details in Table 8.9 in Appendix Section 8.6.

5.4.1.3 Accuracy testing

Description:

Accuracy testing focuses on testing a system capability to perform its functions to provide the right results or effects. In this testing, the accuracy of the Kodi program and WebUI are tested based on the user interaction and input against WebUI response and results being displayed.

Procedures:

Testing activities involved testing Kodi program and its accuracy to display and play a media file, testing the WebUI structure in terms of correct web page for each of its URL link and testing the user input for various forms in the WebUI. The forms include the ones used for registration, comment, changing user profile change and password, adding new server share, editing existing server share and editing server settings. Accuracy testing also covers the accuracy of WebUI actions performing its functions related to user input and the forms, admin functions to accept or reject user application, disable or enable existing user account and resetting user password and delete an existing server share. The accuracy of WebUI media center function was also tested, verifying that it can display list of media files correctly and play the correct media file that is being clicked by user.

Results:

The final system passes all accuracy test cases. This further verified that the system was properly fixed for its defects/errors during component and integration test level. Full results for accuracy testing are described in details in Table 8.10 in Appendix Section 8.6.

5.4.1.4 Interoperability testing

Description:

According to ISO 9126, interoperability testing focuses on a system capability to interact with one or more specified components or systems. In this testing, the test cases focus on the SambaKodiPi file server and WebUI components.

Procedures:

Testing activities involved testing the interoperability of the file server capability to share files to other systems, which in this case the Windows platform and Android platform. Interoperability testing also covers the WebUI capability to be accessed by different browsers and on different platforms.

Results:

The final system passes all interoperability test cases. SambaKodiPi file server function is capable to host and share files to both laptop running Windows 7 and smartphone running Android KitKat, through ES File Explorer app. The WebUI website is proved to be accessible on both platforms, using both Google Chrome and Mozilla Firefox browsers. Full results for interoperability testing are described in details in Table 8.11 in Appendix Section 8.6.

5.4.2 Testing Reliability

5.4.2.1 Reliability testing

Description:

According to ISO 9126, reliability of a system is its capability to maintain a specified level of performance under certain conditions and for a specific period of time. In this testing, the test cases focus on SambaKodiPi functions to run normally for a specific period of time.

Procedures:

Testing activities involved testing the Kodi media center program and WebUI media center function to play several media files for a certain time period. It also involved testing the Samba file server shares accessibility over a 24 hours period.

Results:

The final system passes all reliability test cases. SambaKodiPi media center function seemed to be working smoothly when playing media files, for both Kodi program and WebUI media center function. Note that for WebUI media center function, its reliability also relies on the system that powers the web browser. As for Samba file server shares, they seemed to be still accessible normally after the Raspberry Pi system been switched on for over 24 hours period. Note that the file server function also relies on router stability that powers the network. Full results for reliability testing are described in details in Table 8.12 in Appendix Section 8.6.

5.4.3 Testing Efficiency

5.4.3.1 Performance testing

Description:

Performance testing measures how efficient a system performs its functionalities. In the context of this system, performance testing focus on the file transfer speed across file server function of SambaKodiPi system.

Procedures:

Testing activities to test the performance of file transfer speed involved the copying of a big size file and measuring the time take to copy the file. In this case, a movie file of size 1 GB was used. The details of the constant variables in the performance test are as described below:

Table 5.1: The constant variables in performance testing for file transfer speed.

Criteria	Constant variables
Performance test procedure	Copying a big file onto Desktop
File being transferred	Movie file of size 1 GB (1000 MB)
External HDD used	Seagate 500GB (with USB 3.0 support)
Powered USB-hub used	10-slots Powered USB-hub (support only up to USB 2.0)
Laptop PC being used	Acer Aspire 4750G
Raspberry Pi storage	Kingston Micro SD 8GB class 4

To test the performance of file transfer speed, the time taken to copy the 1 GB file was tested on various setup and condition as well as copying the file itself as server share. The file was copied for several times through the external HDD, with and without powered USB hub as intermediary and also using Raspberry Pi to test real-time file transfer speed when the file is accessed through server shares. The list of the different setups to test the file transfer speed is as listed below:

a) External HDD - Powered USB Hub - Raspberry Pi (using File Manager)
b) External HDD - Powered USB Hub - Raspberry Pi (using LXTerminal)
c) External HDD - Powered USB Hub - Raspberry Pi (using default command-line, no desktop GUI)
d) External HDD - Powered USB Hub - Raspberry Pi - Router - Laptop (the file as server share – the operational setup)
e) External HDD - Laptop USB 2.0 slot
f) External HDD - Powered USB Hub - Laptop USB 2.0 slot
g) External HDD - Laptop USB 3.0 slot (the ideal setup)
h) External HDD - Powered USB Hub - Laptop USB 3.0 slot

Results:

Through the test, it is found that the speed of file transfer through server shares is almost the same as normal copying speed from external HDD into the Raspberry Pi itself, which is around 5 MB per second. This file transfer speed is already quite good to ensure smooth media streaming. As an ideal comparison, the file transfer speed is at optimum speed of around 55 MB per second when tested through the ideal setup.

Note that the file transfer speed was restricted by the limitation of the powered USB-hub that was being used only supported up to USB 2.0 transfer speed. At the time of the testing, a powered USB hub that supports USB 3.0 was not available. It is believed that from the test results, the file transfer speed of a file as server share should be much higher or could be close to 50 MB per second if the storage has a powered USB hub that can support USB 3.0. The results of the performance of file transfer speed are as shown in Table 8.13 in Appendix Section 8.6.

5.5 Acceptance Testing

Acceptance testing is the fourth and final test level after system testing. Main objective is for intended users of a system to actually use the system and decide whether the system meets their specified requirements. In the context of this system, the actual end-user for the system is the author himself. Thus, alpha testing and beta testing are a moot point considering the end user is the developer himself and he already more than familiar with the system functions, operations and limitations. However, these testing can still be done in the future shall the system is intended for other group of users. Due to this, only acceptance testing is done on the system, to test user acceptance of the system.

5.5.1 Testing user acceptance

Description:

User acceptance testing is performed by the intended user of a system by actually using the final system and deciding whether the system is ready to be accepted and deployed on a real environment. Test cases were based on certain criteria determined by the author.

Procedures:

Testing activities involved testing user acceptance to the system functionalities as a whole. Criteria focus on whether the final system is complete with sufficient enough functions and features. It also test on whether the user is satisfied with the system as a whole and accept it as what they really wanted.

Results:

The final system passes all user acceptance test cases. The intended user, which is the developer himself, is fully satisfied with the file server and media center functions. The intended user also fully satisfied with the Web-based user interface (WebUI) file server configuration function and the secondary WebUI media center function as they are both working correctly. Full results for acceptance testing are described in details in Table 8.14 in Appendix Section 8.7.

5.6 Summary

This chapter has described in details how the system was tested through four test levels namely component testing, integration testing, system testing and acceptance testing. Each test level contains sub-testing activities. In each testing activities, the test descriptions, procedures and results are stated and described accordingly along with full testing results, included in Appendix section. After the system underwent those testing activities, the system as a whole has passed all the test cases. Finally it was accepted by the intended user as a complete system with sufficient functionalities, fulfilling what the user intended.

6 Chapter 6 – Conclusion and Future Works

6.1 Conclusion

In this FYP Final Report, all chapters seemed to have described and explained in details about this resulting final system for this FYP project. Chapter 1 has introduced this system on what was all about and Chapter 2 has provided more than sufficient literature review for this system and its background information. Meanwhile, Chapter 3 described in details the system analysis and design for this system along with its architecture and concept. Then Chapter 4 outlines its implementation details through its six development phases in realizing this final system. Finally, Chapter 5 displayed in details the testing activities that were done on the final system along with their results which are included in Appendix section.

Despite the overwhelming odds of technical difficulties and time constraint, in the end, the final system was successfully developed. The final system, which nicknamed as *SambaKodiPi* is essentially an integrated system of 5 main integrated components, namely: the Raspberry Pi component, the Samba file server component, the Kodi media center component, the LAMP webserver component and its Web-based User Interface (WebUI) component. All of these components have passed component and integration testing and proved to be working together. The final system also passed system testing, which tested its functionality and reliability and finally acceptance testing by its intended user. Granted that the system may still be quite simple but there will always be rooms and time for improvements and addition of more functions in the future. And it is now properly equipped with all the required capability to realize all its original intent and purposes. Thus, the final year project aimed to develop this system, as a whole, is definitely a success.

6.1.1 System limitations

Note that the final system has the following limitations:

a) Its file server and file sharing function is intended for personal usage, in a small home LAN network environment with limited number of users (around 5). Thus, its file server and file sharing is not tested to cater for high number of users. Note that doing this may overload the Raspberry Pi device and crash if it cannot handle high number of users connecting and accessing its server shares concurrently.

b) Its primary media center function was tested to work with one video output, either PC monitor or HDTV. Its HDMI video capability was not tested on multiple monitors or screens and Raspberry Pi may lack the power to support multiple monitors output.

c) Its Web-based User Interface (WebUI) component was tested only on Google Chrome v42 and Mozilla Firefox v36, on both Windows and Android platforms. And thus, it was not tested on other web browser, running on other platform. Also note that its display and functionality may not work on web browsers with low HTML5 support.

d) Its secondary media center function, which provides audio and video files playback using its WebUI component is limited in terms of format support. The audio player for the WebUI was implemented using HTML5 audio element and thus its audio playback support is limited to MP3, WAV and OGG file extensions. The video player for the WebUI was implemented using DivX web player plugin and thus its video playback support is limited to MKV, DIVX, MP4, M4V and MOV file extensions.

e) Also note that since the secondary media function for video playback uses 3rd party DivX web player plugin, the WebUI video playback will not work on web browsers on mobile devices since they lack support for 3rd party plugins. Audio and image viewer function however should be working fine as they are using purely HTML5 elements.

6.2 Future Works

There are many other functions that the author would have liked to add into the system but was unable to do so due to overwhelming time constraint as well as due to technical knowledge and implementation limitations. Other original functions intended for the system were also not implemented due to their impossible nature or out of the original system scope. They are now essentially categorized by the author as possible future works that can be done to improve the system, where some will implemented by the author himself. The list of future works for this system is as listed below:

1) Extend the system for access outside of local network/through the internet, but doing this will also need the system to be equipped with proper security implementation (along with proper session timeout and dual-login prevention)
2) Add TV-viewing function (was not implemented due to lack of hardware component and time constraint)
3) Develop a custom-made parser to parse Samba CONF file and avoid the limitations of *php_ini_file()* with certain keywords
4) Develop WebUI system config, if possible, to have function to enable/disable file server, remote restart/shutdown of Kodi program and remote restart of Raspberry Pi.
5) Develop an intermediary GUI for direct system management for the Rasp Pi (same concept with WebUI) that shall separate direct access to system OS
6) Extend Kodi and WebUI media center function files to include user-uploaded content
7) Extend WebUI system info, if possible, to display server health, performance and logs
8) Extend WebUI to have a web-based terminal access using SSH to manage system
9) Improve WebUI media function to have playlist function (with queue, like Youtube)
10) Improve WebUI to have localization for other languages

7 REFERENCES

Beal, V. (n.d.). Server. Retrieved November 9, 2014, from http://www.webopedia.com/TERM/S/server.html.

Case, L. (2011). *Windows Home Server 2011: What It Is and How to Use It*. Retrieved from http://www.pcworld.com/article/228598/windows_home_server_2011_what_it_is_and_how_to_use_it.html.

Cellan-Jones, R. (2012). *The Raspberry Pi computer goes on general sale*. Retrieved November 9, 2014, from http://www.bbc.co.uk/news/technology-17190918.

Clay, C. C. (2014). *Raspberry Pi: 11 reasons why it's the perfect small server*. Retrieved December 23, 2014, from http://www.zdnet.com/article/raspberry-pi-11-reasons-why-its-the-perfect-small-server/.

Evans, K. (n.d.). *How Does a File Server Work?* Retrieved November 12, 2014, from http://www.ehow.com/how-does_4761418_file-server-work.html.

FAQs (n.d.). Retrieved November 9, 2014, from http://www.raspberrypi.org/help/faqs/.

Fitzpatrick, J. (2009). *Best Home Server Software*. Retrieved from http://lifehacker.com/5162026/best-home-server-software.

Fitzpatrick, J., & Purdy, K. (2010). *Which Media Center Is Right for You: Boxee, XBMC, and Windows Media Center Compared*. Retrieved from http://lifehacker.com/5462275/which-_media-center-is-right-for-you-boxee-xbmc-and-windows-media-center-compared.

Foley, M. J. (2011). *Microsoft releases Windows 'Vail' server to manufacturing*. Retrieved from http://www.zdnet.com/blog/microsoft/microsoft-releases-windows-vail-server-to-manufacturing/9050.

Foley, M. J. (2012). *Microsoft confirms enthusiasts' fears: No more versions of Windows Home Server*. Retrieved from http://www.zdnet.com/microsoft-confirms-enthusiasts-fears-no-_more-versions-of-windows-home-server-7000000348/.

FreeNAS Features. (n.d.). Retrieved November 12, 2014, from http://www.freenas.org/about/features.html.

Hanson, G. (2014). *What is a file server?* Retrieved from http://www.wisegeek.com/what-is-a-file-server.htm.

HTPC. (n.d.). Retrieved November 9, 2014, from http://www.pcmag.com/encyclopedia/term/44500/htpc.

kodi.tv (n.d.).). Retrieved November 23, 2014, from http://kodi.tv/download/.

Layton, J. (2006). *How Media-center PCs Work*. Retrieved from http://computer.howstuffworks.com/media-center-pc.htm.

Model A. (n.d.). Retrieved November 11, 2014, from http://www.raspberrypi.org/products/model-a/.

Model A+. (n.d.). Retrieved November 11, 2014, from http://www.raspberrypi.org/products/model-a-plus/.

Model B. (n.d.). Retrieved November 11, 2014, from http://www.raspberrypi.org/products/model-b/.

Model B+. (n.d.). Retrieved November 11, 2014, from http://www.raspberrypi.org/products/model-b-plus/.

Non-dedicated file server. (n.d.). WordNet 3.0, Farlex clipart collection. (2003-2008. Retrieved November 12, 2014, from http://www.thefreedictionary.com/non-dedicated+file+server.

php_ini_file. (n.d.). parse_ini_file — Parse a configuration file. Retrieved April 18, 2015, from http://php.net/manual/en/function.parse-ini-file.php.

Raspbian (n.d.). Retrieved November 23, 2014, from http://www.raspberrypi.org/downloads/.

Rouse, M. (2005). File server. Retrieved from http://searchnetworking.techtarget.com/definition/file-server.

Samba. (n.d.). Retrieved November 9, 2014, from http://www.samba.org/samba/.

Samba.org (n.d.). Retrieved November 23, 2014, from http://www.samba.org/.

ServerFaq. (2012). Retrieved from https://help.ubuntu.com/community/ServerFaq.

Software Life Cycle. (2011). Chapter 2, Testing throughout the Software Life Cycle. Software Testing Foundations Certified Tester slide v1.0, MSTB/GTB.

Sommerville, I. (2011). *Software Engineering – 9th Edition*. Boston, US: Addison-Wesley.

Throckmorton, Z. (2011). *File Server Builder's Guide*. Retrieved from http://www.anandtech.com/show/4666/file-server-builders-guide.

Tuukka. (2011). Top 10 Best HTPC Software for HDTV. Retrieved from http://mymediaexperience.com/top-10-most-popular-media-center-software-for-your-hdtv/.

Ubuntu. (2014). *What's new in Ubuntu Server 14.04 LTS?* Retrieved from https://insights.ubuntu.com/2014/04/17/whats-new-in-ubuntu-server-14-04-lts/.

Upton, E. (2014a). *Introducing Raspberry Pi Model B+*. Retrieved from http://www.raspberrypi.org/introducing-raspberry-pi-model-b-plus/.

Upton, E. (2014b). Raspberry Pi Model A+ on sale now at $20. Retrieved from http://www.raspberrypi.org/raspberry-pi-model-a-plus-on-sale/.

What is Raspberry Pi. (n.d.). Retrieved November 9, 2014, from http://www.raspberrypi.org/help/what-is-a-raspberry-pi/.

8 APPENDICES

8.1 Appendix A – Default content of Samba configuration file

```
# This is the main Samba configuration file. You should read the
# smb.conf(5) manual page in order to understand the options listed
# here. Samba has a huge number of configurable options (perhaps too
# many!) most of which are not shown in this example
#
# Any line which starts with a ; (semi-colon) or a # (hash)
# is a comment and is ignored. In this example we will use a #
# for commentry and a ; for parts of the config file that you
# may wish to enable
#
# NOTE: Whenever you modify this file you should run the command "testparm"
# to check that you have not many any basic syntactic errors.
#
#======================                    Global                    Settings
=====================================
[global]

# workgroup = NT-Domain-Name or Workgroup-Name, eg: REDHAT4
   workgroup = MYGROUP

# server string is the equivalent of the NT Description field
   server string = Samba Server

# This option is important for security. It allows you to restrict
# connections to machines which are on your local network. The
# following example restricts access to two C class networks and
# the "loopback" interface. For more examples of the syntax see
# the smb.conf man page
;   hosts allow = 192.168.1. 192.168.2. 127.

# If you want to automatically load your printer list rather
# than setting them up individually then you'll need this
   load printers = yes

# you may wish to override the location of the printcap file
;   printcap name = /etc/printcap

# on SystemV system setting printcap name to lpstat should allow
# you to automatically obtain a printer list from the SystemV spool
# system
;   printcap name = lpstat

# It should not be necessary to specify the print system type unless
# it is non-standard. Currently supported print systems include:
# bsd, sysv, plp, lprng, aix, hpux, qnx
```

```
;   printing = bsd

# Uncomment this if you want a guest account, you must add this to
/etc/passwd
# otherwise the user "nobody" is used
;  guest account = pcguest

# this tells Samba to use a separate log file for each machine
# that connects
   log file = /usr/local/samba/var/log.%m

# Put a capping on the size of the log files (in Kb).
   max log size = 50

# Security mode. Most people will want user level security. See
# security_level.txt for details.
   security = user

# Use password server option only with security = server
# The argument list may include:
#   password server = My_PDC_Name [My_BDC_Name] [My_Next_BDC_Name]
# or to auto-locate the domain controller/s
#   password server = *
;   password server = <NT-Server-Name>

# Note: Do NOT use the now deprecated option of "domain controller"
# This option is no longer implemented.

# You may wish to use password encryption. Please read
# ENCRYPTION.txt, Win95.txt and WinNT.txt in the Samba documentation.
# Do not enable this option unless you have read those documents
encrypt passwords = yes

# Using the following line enables you to customise your configuration
# on a per machine basis. The %m gets replaced with the netbios name
# of the machine that is connecting
;   include = /usr/local/samba/lib/smb.conf.%m

# Most people will find that this option gives better performance.
# See speed.txt and the manual pages for details
# You may want to add the following on a Linux system:
#         SO_RCVBUF=8192 SO_SNDBUF=8192
   socket options = TCP_NODELAY

# Configure Samba to use multiple interfaces
# If you have multiple network interfaces then you must list them
# here. See the man page for details.
;   interfaces = 192.168.12.2/24 192.168.13.2/24

# Browser Control Options:
# set local master to no if you don't want Samba to become a master
# browser on your network. Otherwise the normal election rules apply
```

```
;   local master = no

# OS Level determines the precedence of this server in master browser
# elections. The default value should be reasonable
;   os level = 33

# Domain Master specifies Samba to be the Domain Master Browser. This
# allows Samba to collate browse lists between subnets. Don't use this
# if you already have a Windows NT domain controller doing this job
;   domain master = yes

# Preferred Master causes Samba to force a local browser election on
startup
# and gives it a slightly higher chance of winning the election
;   preferred master = yes

# Enable this if you want Samba to be a domain logon server for
# Windows95 workstations.
;   domain logons = yes

# if you enable domain logons then you may want a per-machine or
# per user logon script
# run a specific logon batch file per workstation (machine)
;   logon script = %m.bat
# run a specific logon batch file per username
;   logon script = %U.bat

# Where to store roving profiles (only for Win95 and WinNT)
#        %L substitutes for this servers netbios name, %U is username
#        You must uncomment the [Profiles] share below
;   logon path = \\%L\Profiles\%U

# Windows Internet Name Serving Support Section:
# WINS Support - Tells the NMBD component of Samba to enable it's WINS
Server
;   wins support = yes

# WINS Server - Tells the NMBD components of Samba to be a WINS Client
#    Note: Samba can be either a WINS Server, or a WINS Client, but NOT
both
;   wins server = w.x.y.z

# WINS Proxy - Tells Samba to answer name resolution queries on
# behalf of a non WINS capable client, for this to work there must be
# at least one    WINS Server on the network. The default is NO.
;   wins proxy = yes

# DNS Proxy - tells Samba whether or not to try to resolve NetBIOS names
# via DNS nslookups. The built-in default for versions 1.9.17 is yes,
# this has been changed in version 1.9.18 to no.
   dns proxy = no
```

```
#============================                    Share                   Definitions
==============================
[homes]
   comment = Home Directories
   browseable = no
   writable = yes

# Un-comment  the  following  and  create  the  netlogon  directory  for  Domain
Logons
; [netlogon]
;   comment = Network Logon Service
;   path = /usr/local/samba/lib/netlogon
;   guest ok = yes
;   writable = no
;   share modes = no


# Un-comment the following to provide a specific roving profile share
# the default is to use the user's home directory
;[Profiles]
;    path = /usr/local/samba/profiles
;    browseable = no
;    guest ok = yes


# NOTE: If you have a BSD-style print system there is no need to
# specifically define each individual printer
[printers]
   comment = All Printers
   path = /usr/spool/samba
   browseable = no
# Set public = yes to allow user 'guest account' to print
   guest ok = no
   writable = no
   printable = yes

# This one is useful for people to share files
;[tmp]
;   comment = Temporary file space
;   path = /tmp
;   read only = no
;   public = yes

# A publicly accessible directory, but read only, except for people in
# the "staff" group
;[public]
;   comment = Public Stuff
;   path = /home/samba
;   public = yes
;   writable = yes
;   printable = no
;   write list = @staff
```

```
# Other examples.
#
# A private printer, usable only by fred. Spool data will be placed in
fred's
# home directory. Note that fred must have write access to the spool
directory,
# wherever it is.
;[fredsprn]
;   comment = Fred's Printer
;   valid users = fred
;   path = /homes/fred
;   printer = freds_printer
;   public = no
;   writable = no
;   printable = yes

# A private directory, usable only by fred. Note that fred requires write
# access to the directory.
;[fredsdir]
;   comment = Fred's Service
;   path = /usr/somewhere/private
;   valid users = fred
;   public = no
;   writable = yes
;   printable = no

# a service which has a different directory for each machine that connects
# this allows you to tailor configurations to incoming machines. You could
# also use the %U option to tailor it by user name.
# The %m gets replaced with the machine name that is connecting.
;[pchome]
;  comment = PC Directories
;  path = /usr/pc/%m
;  public = no
;  writable = yes

# A publicly accessible directory, read/write to all users. Note that all
files
# created in the directory by users will be owned by the default user, so
# any user with access can delete any other user's files. Obviously this
# directory must be writable by the default user. Another user could of
course
# be specified, in which case all files would be owned by that user
instead.
;[public]
;   path = /usr/somewhere/else/public
;   public = yes
;   only guest = yes
;   writable = yes
;   printable = no
```

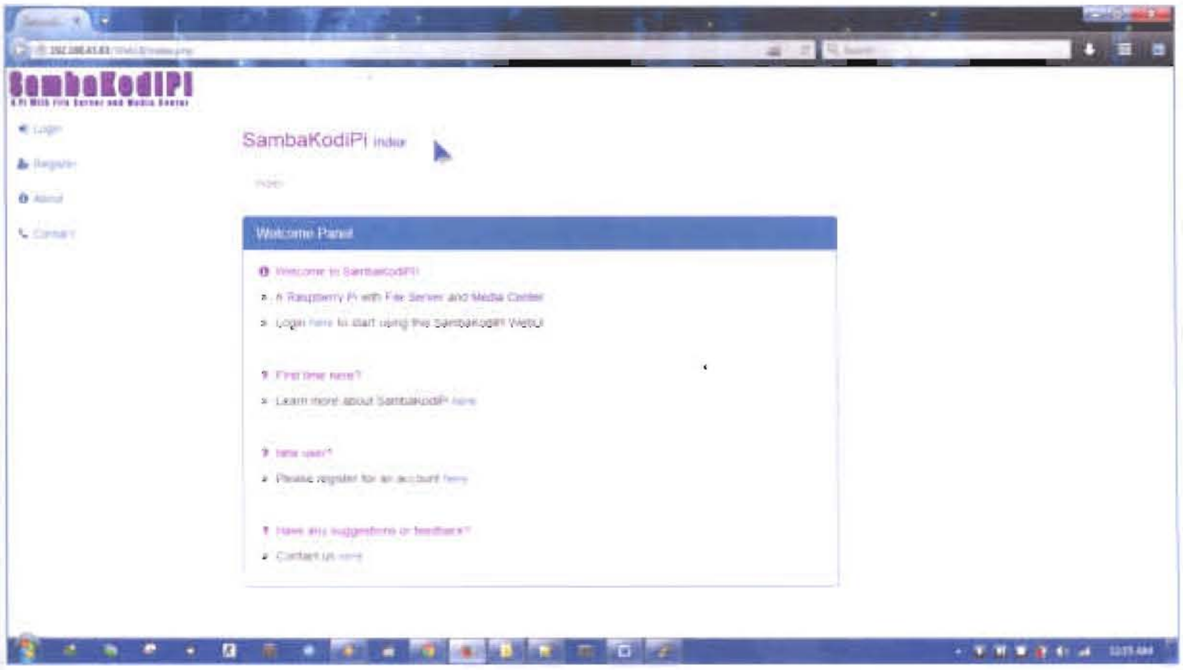
```
# The following two entries demonstrate how to share a directory so that
two
# users can place files there that will be owned by the specific users. In
this
# setup, the directory should be writable by both users and should have the
# sticky bit set on it to prevent abuse. Obviously this could be extended
to
# as many users as required.
;[myshare]
;   comment = Mary's and Fred's stuff
;   path = /usr/somewhere/shared
;   valid users = mary fred
;   public = no
;   writable = yes
;   printable = no
;   create mask = 0765
```

8.2 Appendix B – Modified content of Samba configuration (smb.conf)

```
[global]
netbios name = Samba
server string = Samba TEST SERVER
workgroup = MSHOME
security = user
hosts allow = 127. 192.168.1. 192.168.43.
interfaces = 127.0.0.1/8 192.168.1.11 192.168.1.111 192.168.43.83
remote announce = 192.168.1.255 192.168.43.1
printcap name = /etc/printcap
load printers = yes
printing = bsd
guest account = smbguest
log file = /var/log/samba/samba.log
max log size = 1000
encrypt passwords = yes
socket options = 'TCP_NODELAY'
local master = no
domain master = no
preferred master = no
domain logons = no
os level = 33
logon path = \\%L\profiles\%u
logon script = %G.bat
wins support = no
wins proxy = no
dns proxy = no

[Raspberry Pi Webserver]
```

```
comment = 'Pi Web server folder - Just for Debugging'
path = /var/www/
browseable = yes
writable = yes
read only = no
public = no
valid users = pi

[Samba CONF Folder]
comment = 'Just for Debugging'
path = /etc/samba/
browseable = yes
writable = yes
read only = no
public = no

[Media Folder]
comment = 'Default media folder'
path = /media/
browseable = yes
writable = yes
read only = no
public = no

[test-share-R]
comment = 'testing share on actual server'
path = /test/test-share-R/
browseable = yes
writable = no
read only = yes
public = no

[test]
comment = 'test'
path = /test/test/
browseable = yes
writable = no
read only = yes
public = no

[samba-backup]
comment = 'samba backup folder'
path = /home/pi/samba/
browseable = yes
writable = yes
read only = no
public = no
```

8.3 Appendix C – The screenshots of WebUI pages.


SambaKodiPI

A Pi With File Server and Media Center

Login

Register

About

Contact

SambaKodiPi index

Index

Welcome Panel

Welcome to SambaKodiPi

» A Raspberry Pi with File Server and Media Center

» Login here to start using the SambaKodiPi WebUI

? First time here?

» Learn more about SambaKodiPi here

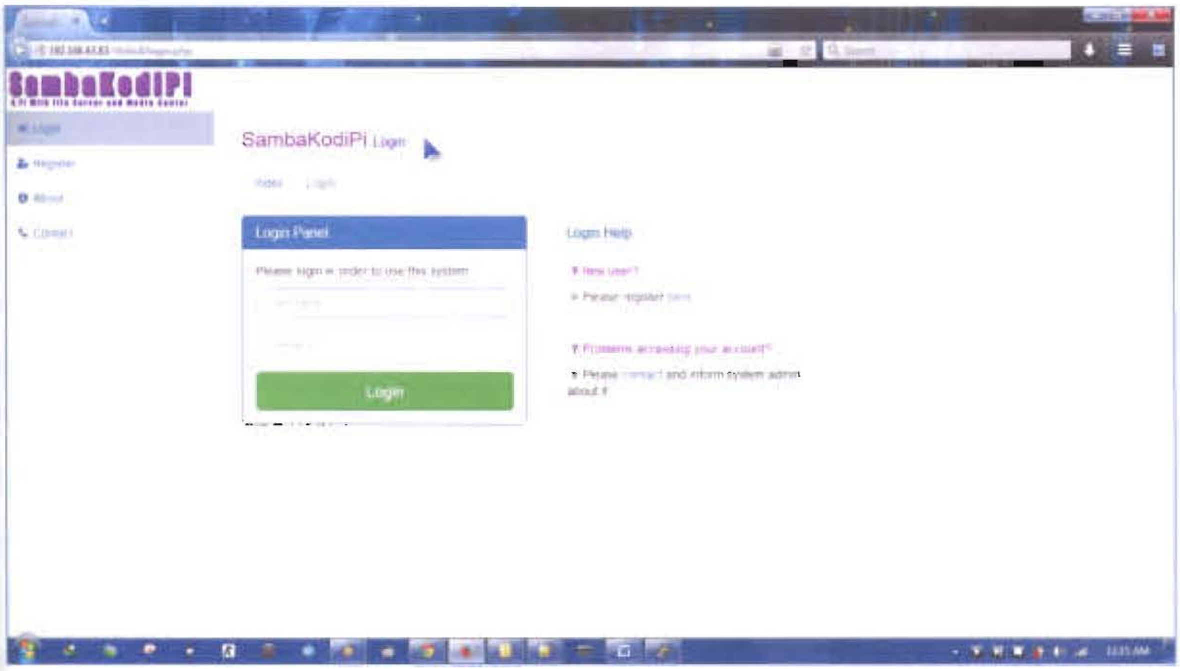
? New user?

» Please register for an account here

? Have any suggestions or feedback?

» Contact us here

Figure 8.1: Screenshot for index.php page.


SambaKodiPI

A Pi With File Server and Media Center

Login

Register

About

Contact

SambaKodiPi Login

Index / Login

Login Panel

Please login in order to use this system

Login

Login Help

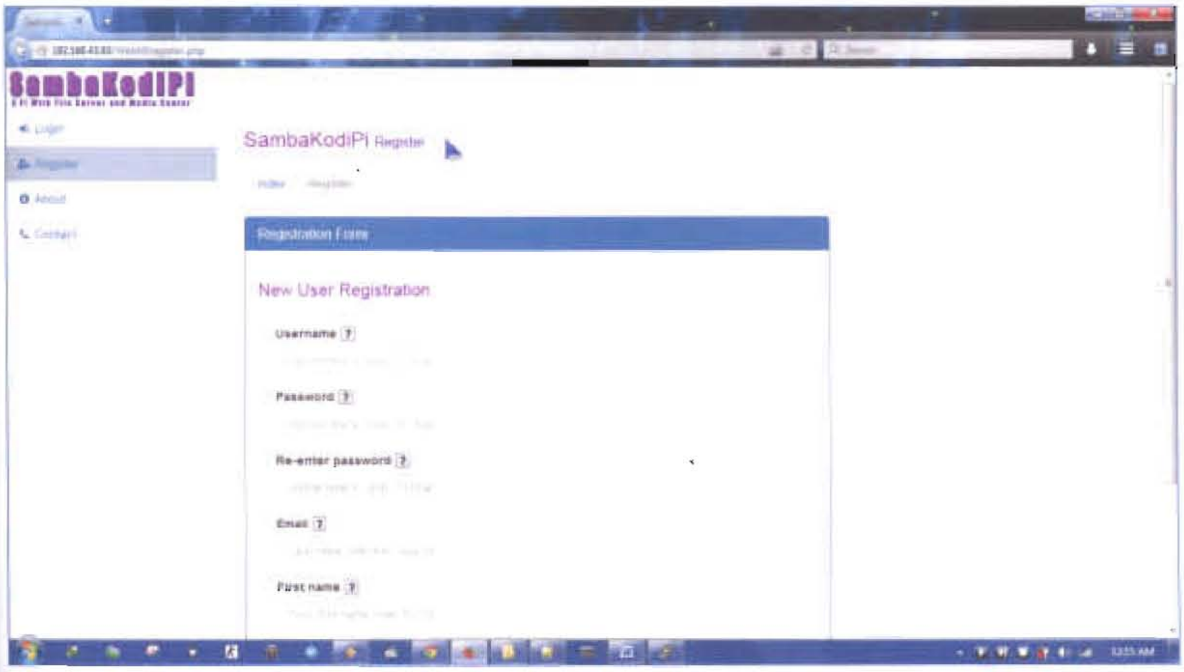
? New user?

» Please register here

? Problems accessing your account?

» Please contact and inform system admin about it

Figure 8.2: Screenshot for login.php page.



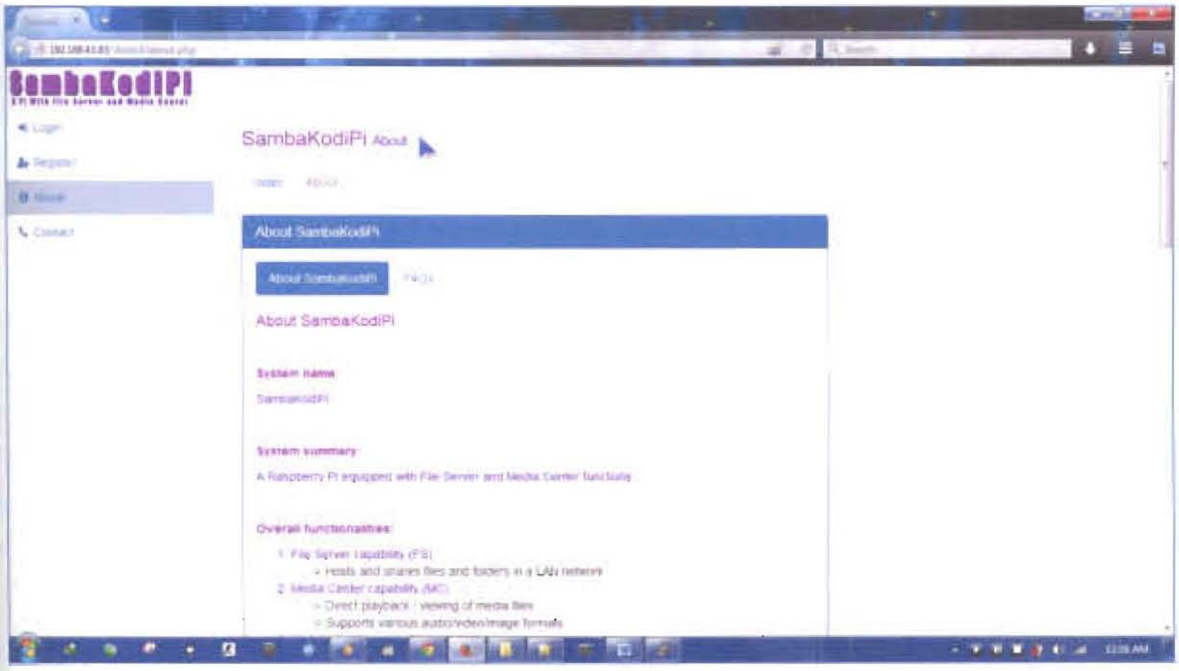
SambaKodiPi Register

Registration Form

New User Registration

Username
Password
Re-enter password
Email
First name

Figure 8.3: Screenshot for register.php page.



SambaKodiPi About

About SambaKodiPi

About SambaKodiPi

System name

SambaKodiPi

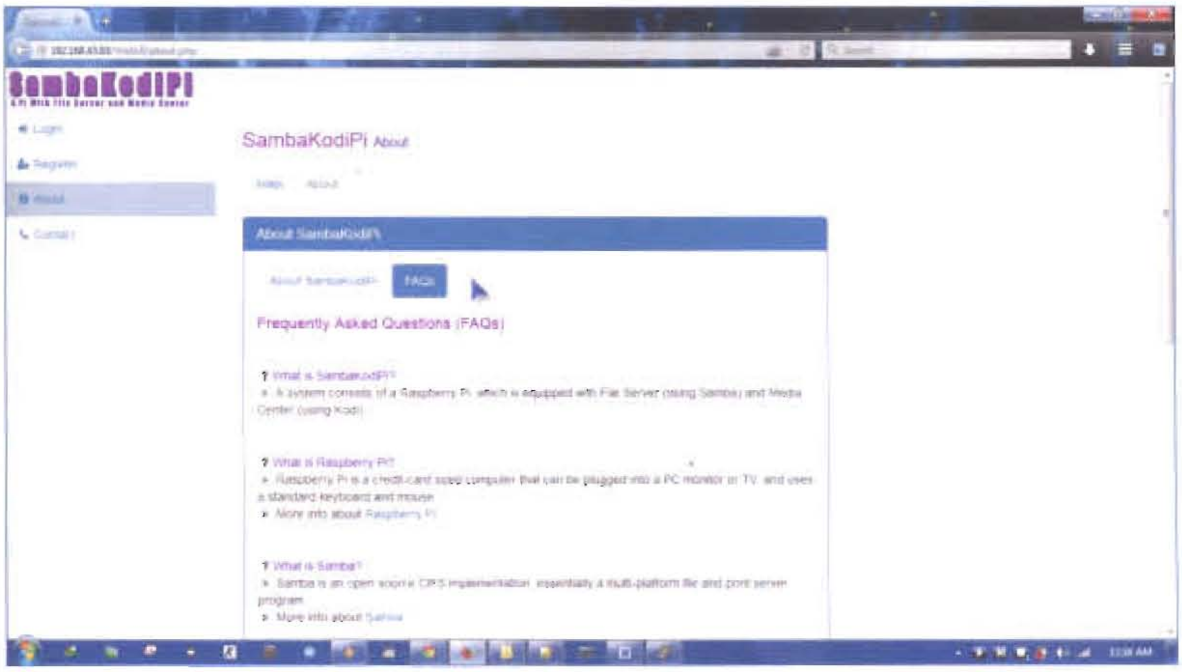
System summary

A Raspberry Pi equipped with File Server and Media Center functions

Overall functionalities:

1. File Server capability (FS)
   - Hosts and shares files and folders in a LAN network
2. Media Center capability (MC)
   - Direct playback / viewing of media files
   - Supports various audio/video/image formats

Figure 8.4: Screenshot for about.php page (about SambaKodiPi tab).



SambaKodiPi About

About SambaKodiPi

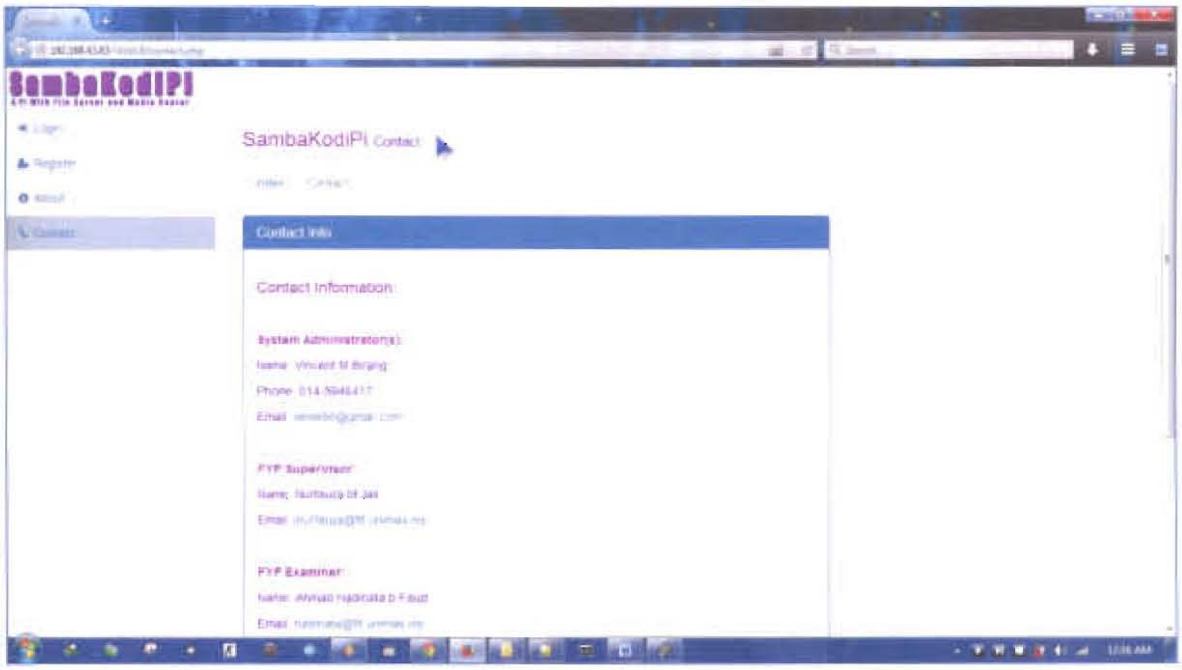
Frequently Asked Questions (FAQs)

? What is SambaKodiPi?
» A system consists of a Raspberry Pi, which is equipped with File Server (using Samba) and Media Center (using Kodi).

? What is Raspberry Pi?
» Raspberry Pi is a credit-card sized computer that can be plugged into a PC monitor or TV, and uses a standard keyboard and mouse.
» More info about Raspberry Pi

? What is Samba?
» Samba is an open source CIFS implementation, essentially a multi-platform file and print server program.
» More info about Samba

Figure 8.5: Screenshot for about.php page (FAQs tab).



SambaKodiPi Contact

Contact Info

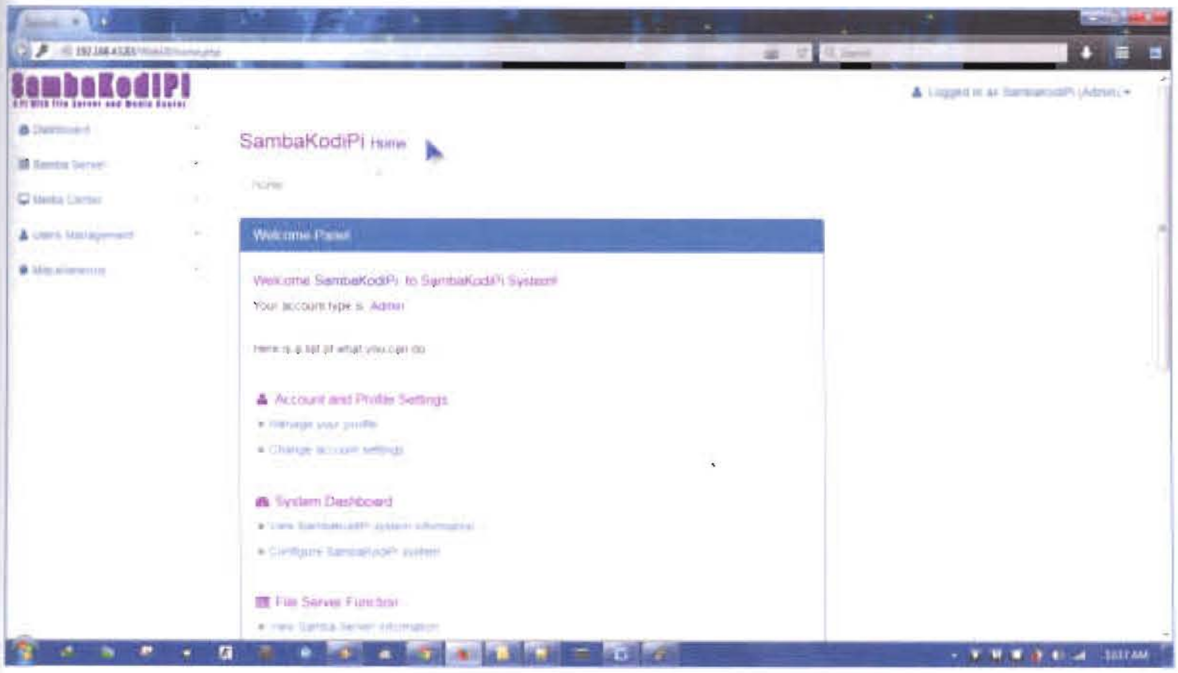
Contact Information:

System Administrator(s):
Name: Vincent M Birang
Phone: 014-5946417
Email: [illegible]

FYP Supervisor:
Name: Nurfauza bt Jali
Email: [illegible]

FYP Examiner:
Name: Ahmad Hadinata b Fauzi
Email: [illegible]

Figure 8.6: Screenshot for contact.php page.



SambaKodiPi Home

Home

Welcome Panel

Welcome SambaKodiPi to SambaKodiPi System!

Your account type is Admin

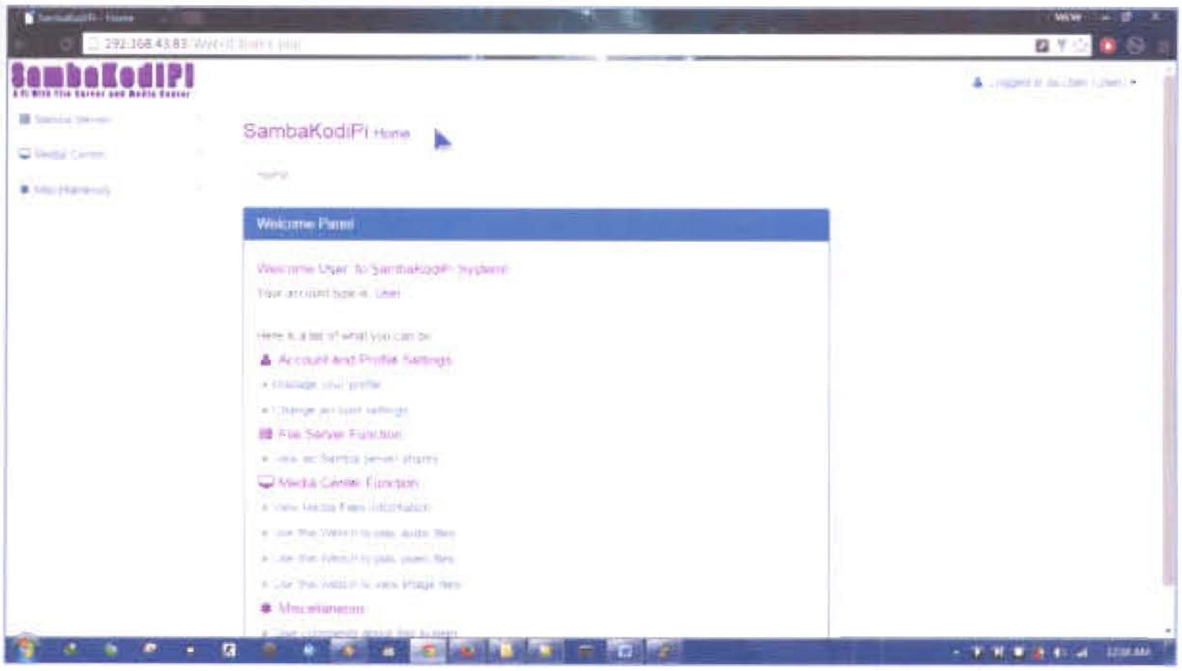
Here is a list of what you can do

Account and Profile Settings
- Manage your profile
- Change account settings

System Dashboard
- View SambaKodiPi system information
- Configure SambaKodiPi system

File Server Function
- View Samba Server information

Figure 8.7: Screenshot for home.php page (admin view).



SambaKodiPi Home

Home

Welcome Panel

Welcome User to SambaKodiPi System!

Your account type is User

Here is a list of what you can do

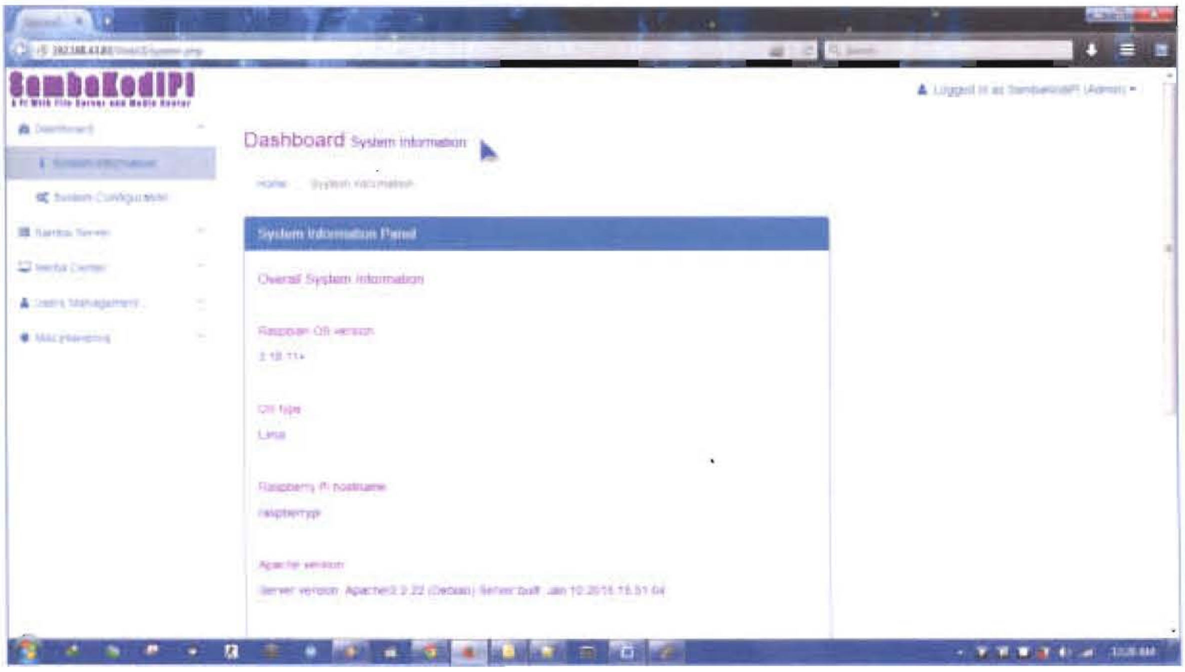
Account and Profile Settings
- Manage your profile
- Change account settings

File Server Function
- View all Samba server shares

Media Center Function
- View Media Files information
- Use the WebUI to play audio files
- Use the WebUI to play video files
- Use the WebUI to view image files

Miscellaneous
- Give comments about the system

Figure 8.8: Screenshot for home.php page (user view).


Dashboard System Information

Home / System Information

System Information Panel

Overall System Information

Raspbian OS version

3.18.11+

OS type

Linux

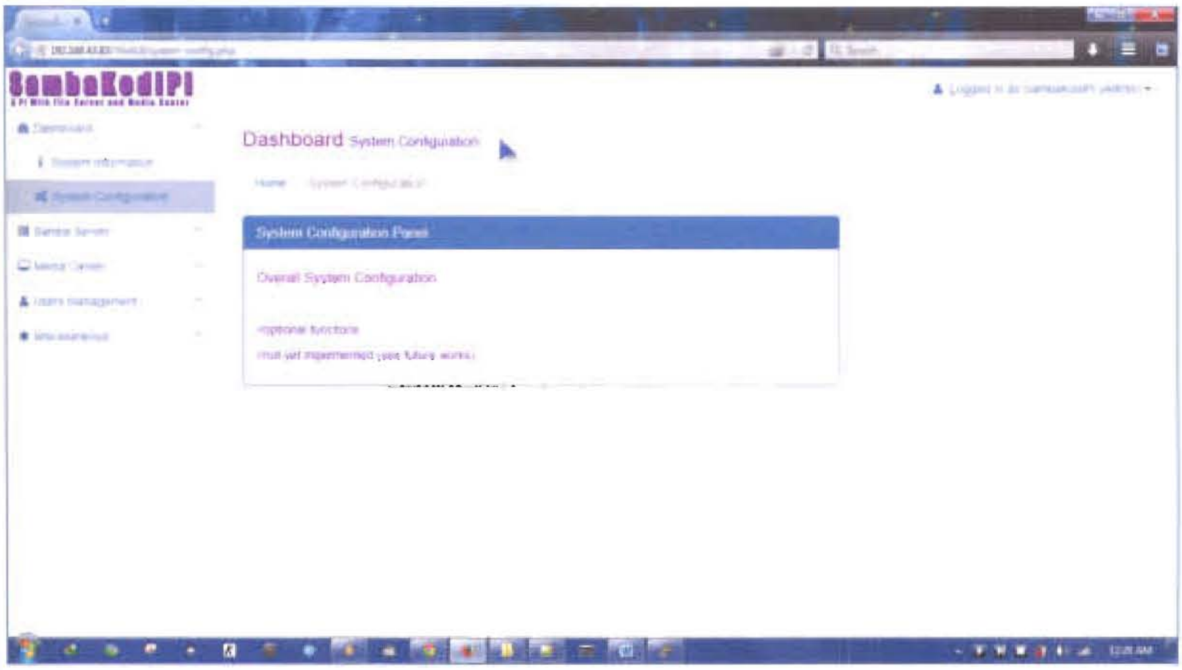
Raspberry Pi hostname

raspberrypi

Apache version

Server version: Apache/2.2.22 (Debian) Server built: Jan 10 2015 15:51:04

Figure 8.9: Screenshot for system.php page.


Dashboard System Configuration

Home / System Configuration

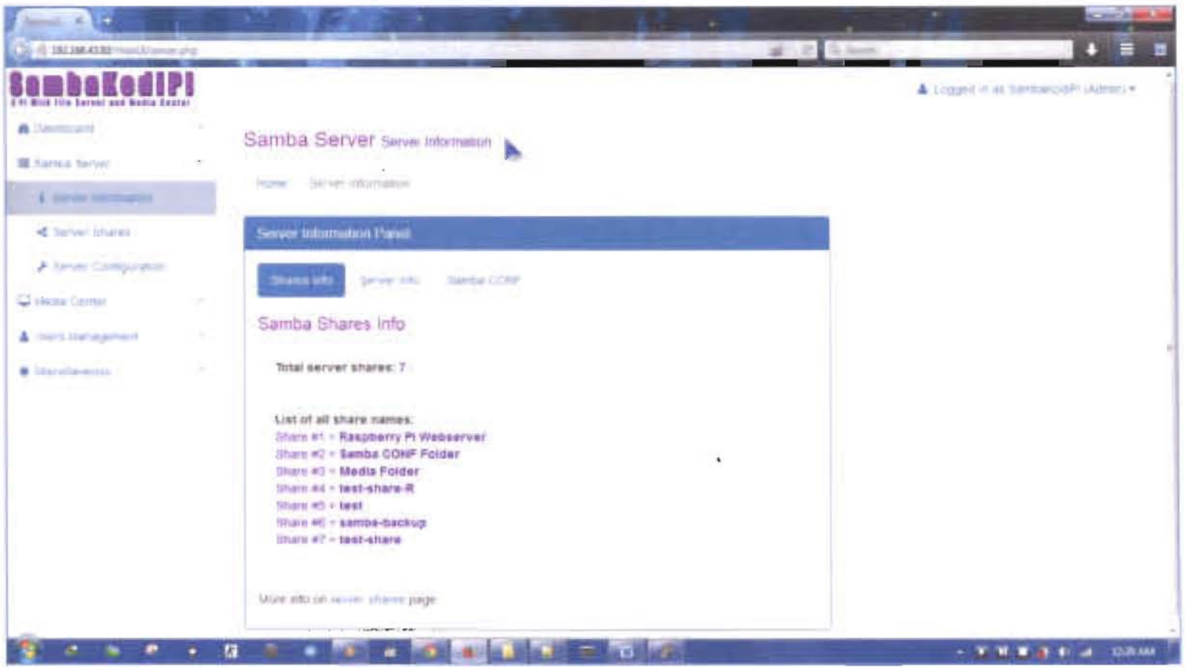
System Configuration Panel

Overall System Configuration

>optional functions

(not yet implemented (see future work))

Figure 8.10: Screenshot for system-config.php page.



Samba Server Server Information

Server Information Panel

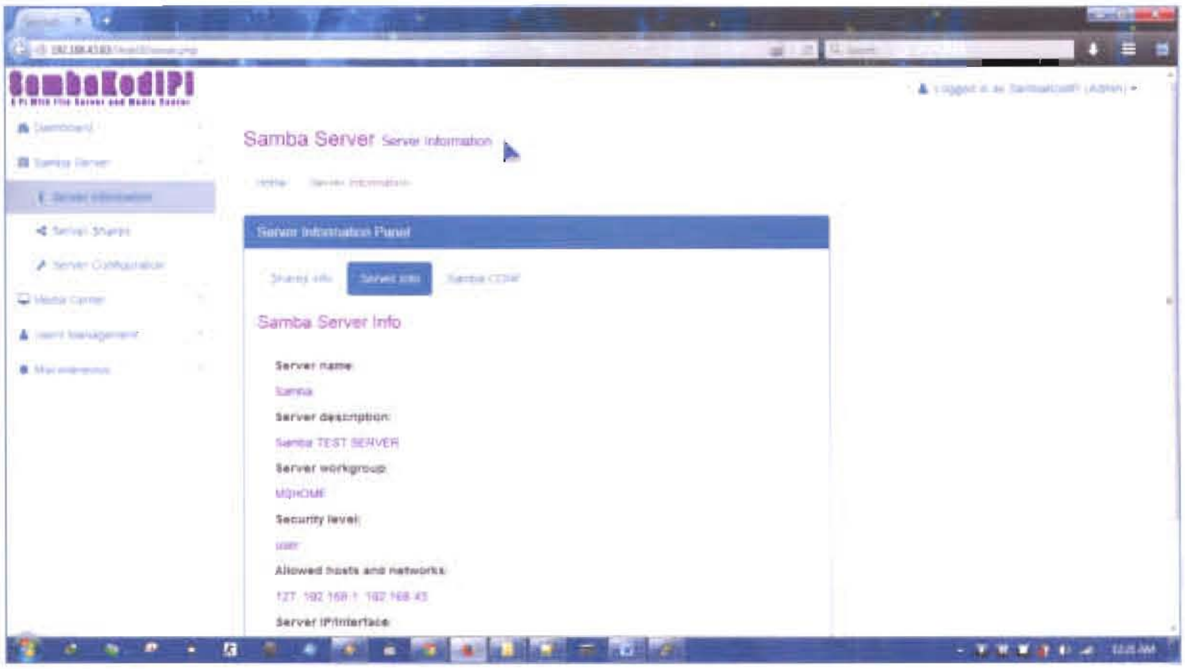
Samba Shares Info

Total server shares: 7

List of all share names:
Share #1 - Raspberry Pi Webserver
Share #2 - Samba CONF Folder
Share #3 - Media Folder
Share #4 - test-share-R
Share #5 - test
Share #6 - samba-backup
Share #7 - test-share

More info on server shares page

Figure 8.11: Screenshot for server.php page, tab shares info.



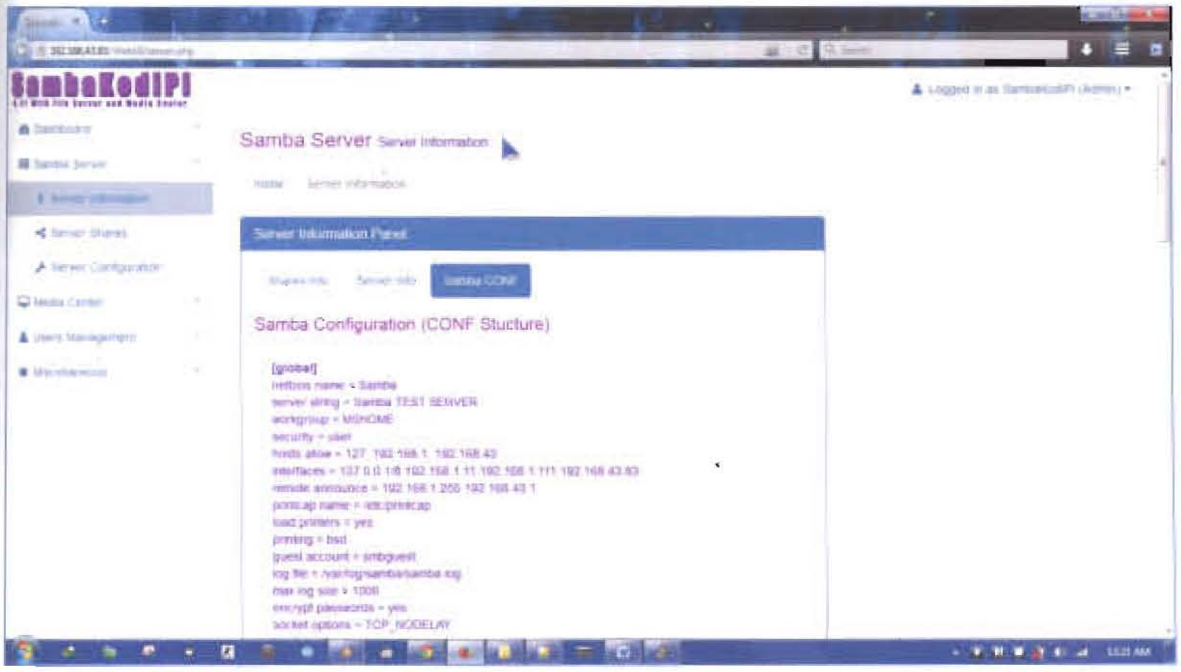
Samba Server Server Information

Server Information Panel

Samba Server Info

Server name:
Samba
Server description:
Samba TEST SERVER
Server workgroup:
MSHOME
Security level:
user
Allowed hosts and networks:
127. 192.168.1. 192.168.43.
Server IP/interface:

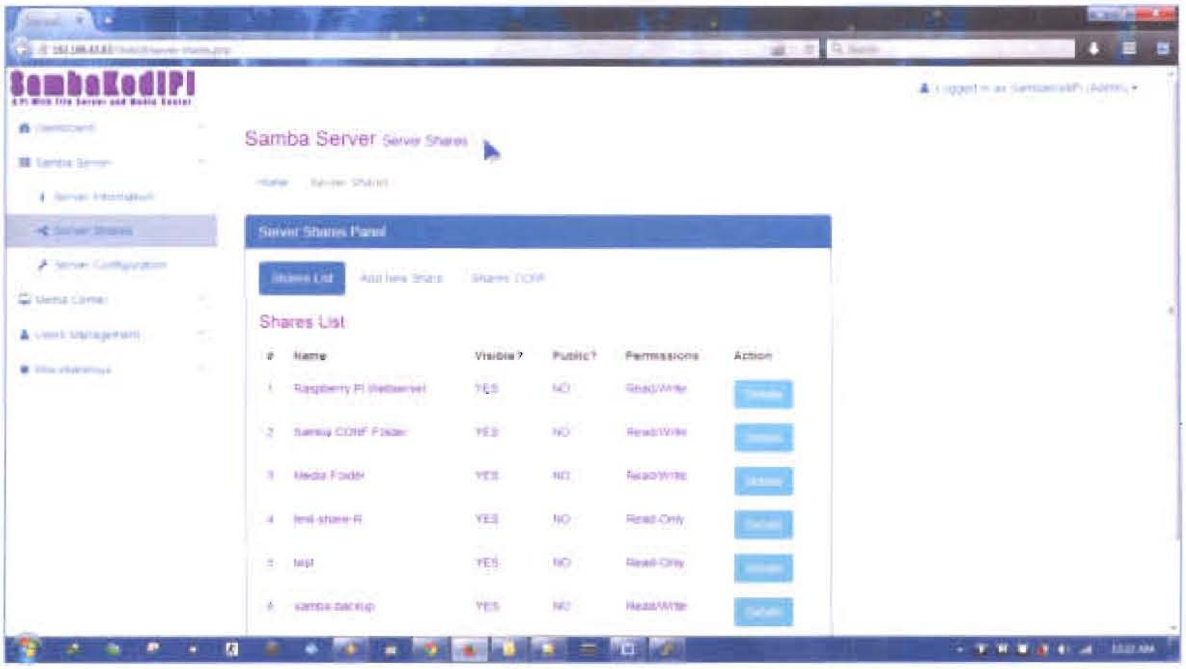
Figure 8.12: Screenshot for server.php page, tab server info.


Samba Server Server Information

Samba Configuration (CONF Stucture)

```
[global]
netbios name = Samba
server string = Samba TEST SERVER
workgroup = MSHOME
security = user
hosts allow = 127. 192.168.1. 192.168.43.
interfaces = 127.0.0.1/8 192.168.1.11 192.168.1.111 192.168.43.83
remote announce = 192.168.1.255 192.168.43.1
printcap name = /etc/printcap
load printers = yes
printing = bsd
guest account = smbguest
log file = /var/log/samba/samba.log
max log size = 1000
encrypt passwords = yes
socket options = TCP_NODELAY
```

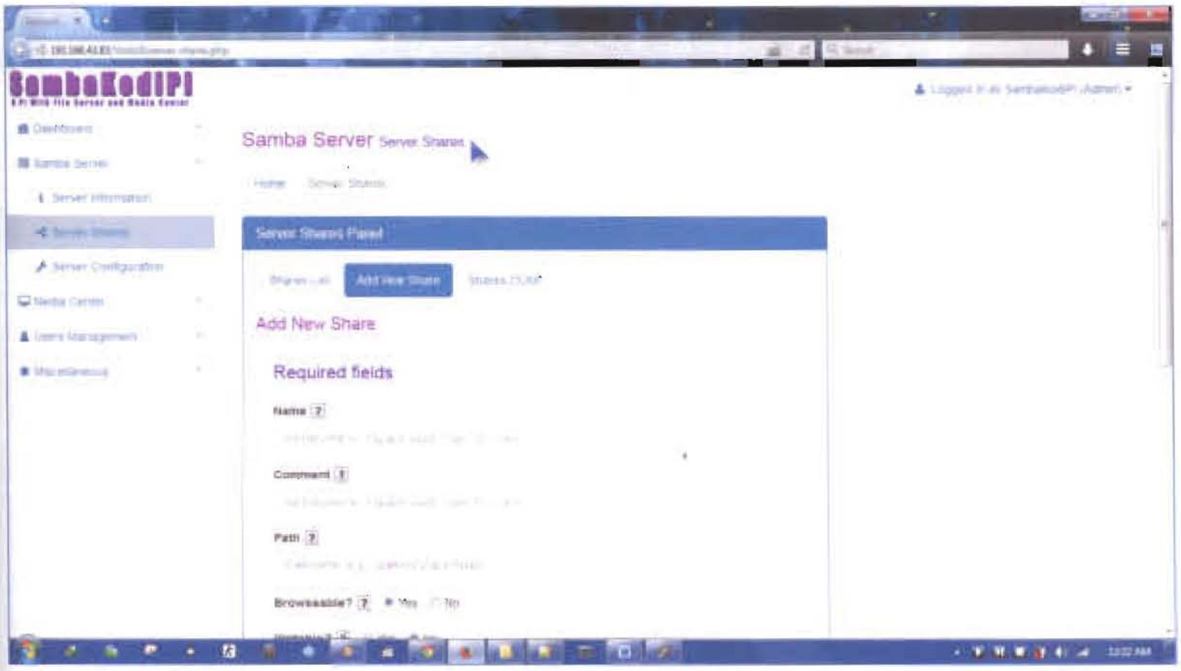
Figure 8.13: Screenshot for server.php page, tab Samba CONF.


Samba Server Server Shares

Shares List

#	Name	Visible?	Public?	Permissions	Action
1	Raspberry PI Webserver	YES	NO	Read/Write	[illegible]
2	Samba CONF Folder	YES	NO	Read/Write	[illegible]
3	Media Folder	YES	NO	Read/Write	[illegible]
4	test-share-R	YES	NO	Read-Only	[illegible]
5	test	YES	NO	Read-Only	[illegible]
6	samba backup	YES	NO	Read/Write	[illegible]

Figure 8.14: Screenshot for server-shares.php page, tab shares info.



Add New Share

Required fields

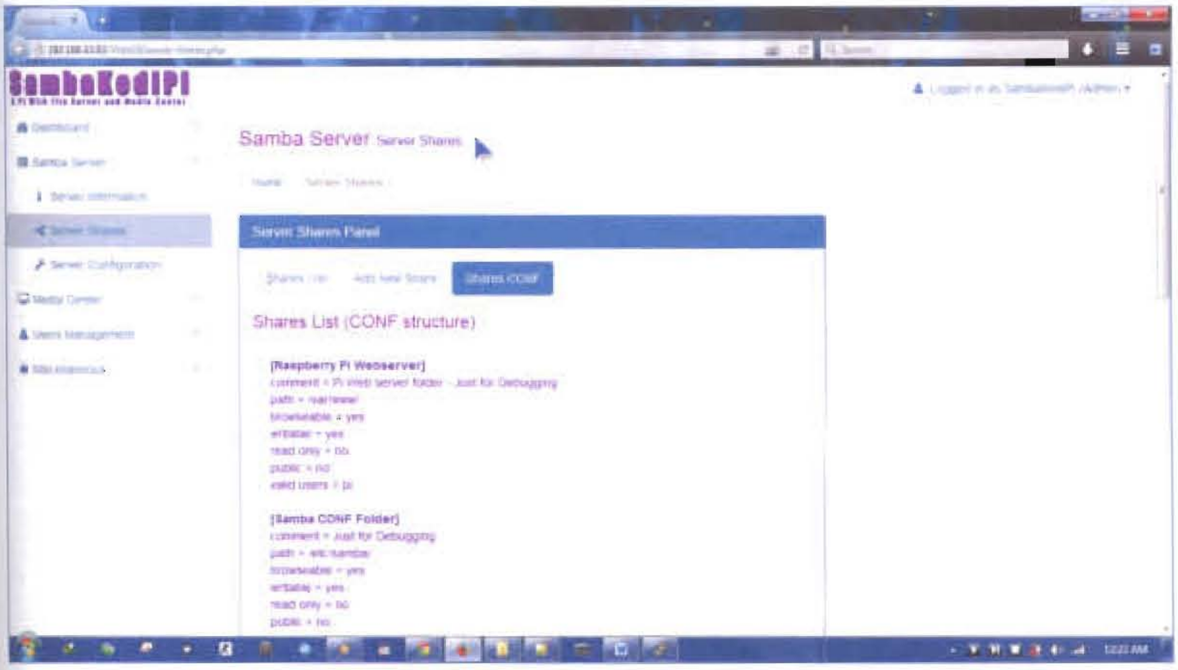
Name

Comment

Path

Browseable? ● Yes ○ No

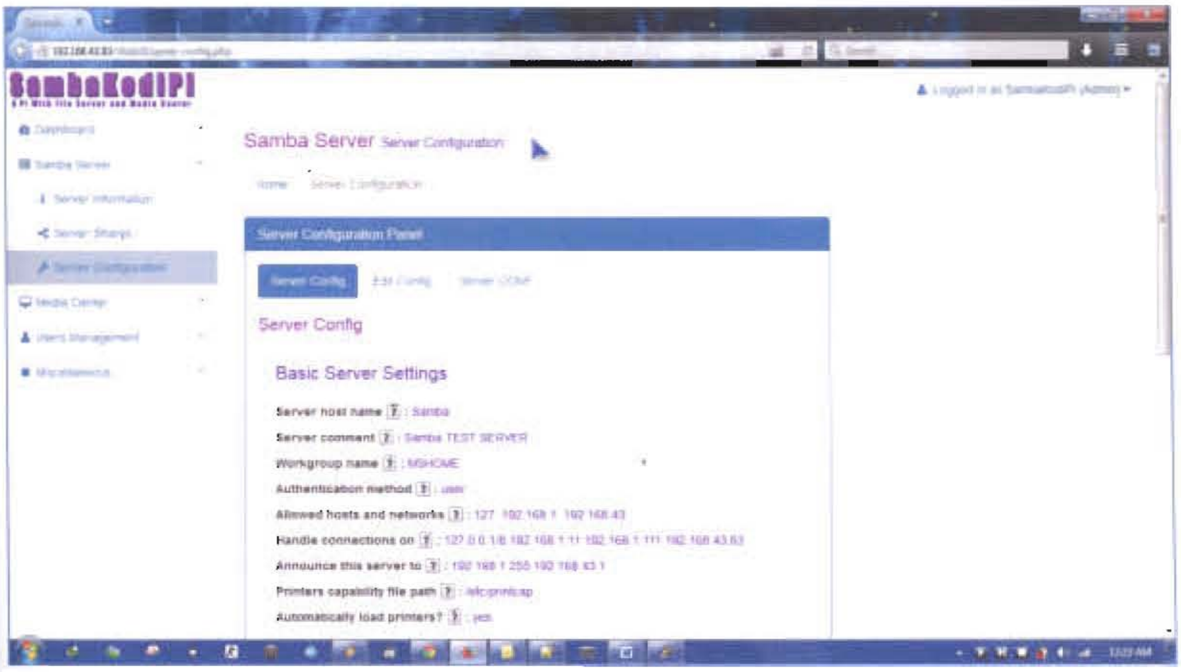
Figure 8.15: Screenshot for server-shares.php page, tab add new share.



Shares List (CONF structure)

```
[Raspberry Pi Webserver]
comment = Pi Web server folder - Just for Debugging
path = /var/www/
browseable = yes
writable = yes
read only = no
public = no
valid users = pi

[Samba CONF Folder]
comment = Just for Debugging
path = /etc/samba/
browseable = yes
writable = yes
read only = no
public = no
```

Figure 8.16: Screenshot for server-shares.php page, tab shares CONF.


Samba Server Server Configuration

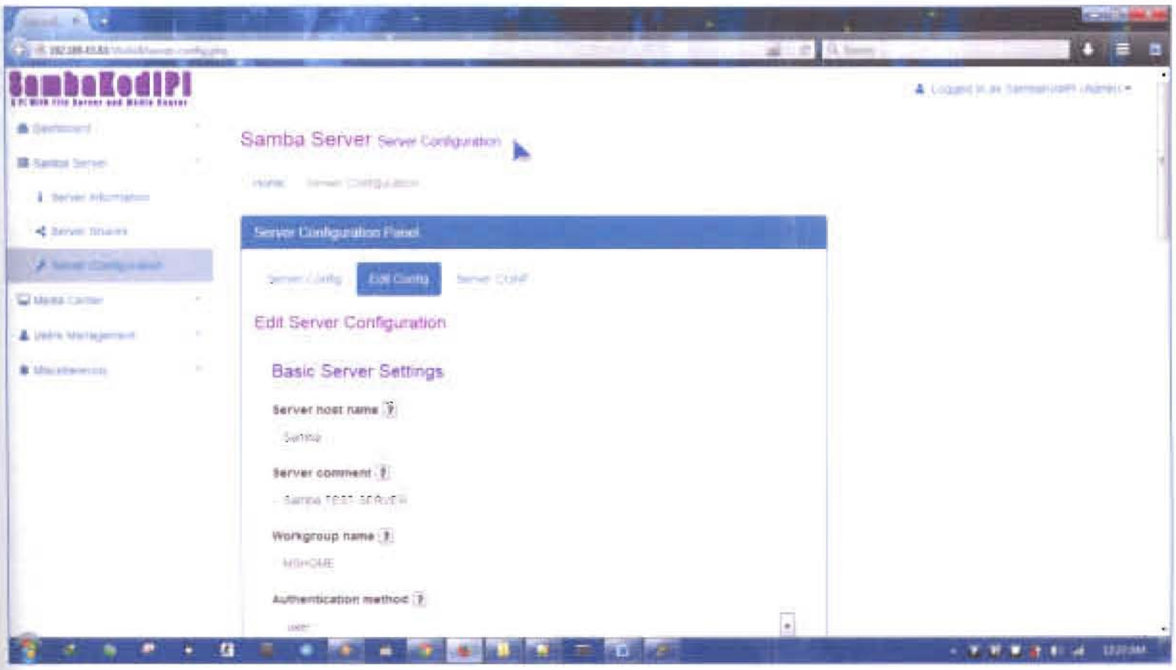
Server Configuration Panel

Server Config

Basic Server Settings

Server host name [?] : Samba
Server comment [?] : Samba TEST SERVER
Workgroup name [?] : MSHOME
Authentication method [?] : user
Allowed hosts and networks [?] : 127. 192.168.1. 192.168.43.
Handle connections on [?] : 127.0.0.1/8 192.168.1.11 192.168.1.111 192.168.43.63
Announce this server to [?] : 192.168.1.255 192.168.43.1
Printers capability file path [?] : /etc/printcap
Automatically load printers? [?] : yes

Figure 8.17: Screeushot for server-config.php page, tab server config.


Samba Server Server Configuration

Server Configuration Panel

Edit Server Configuration

Basic Server Settings

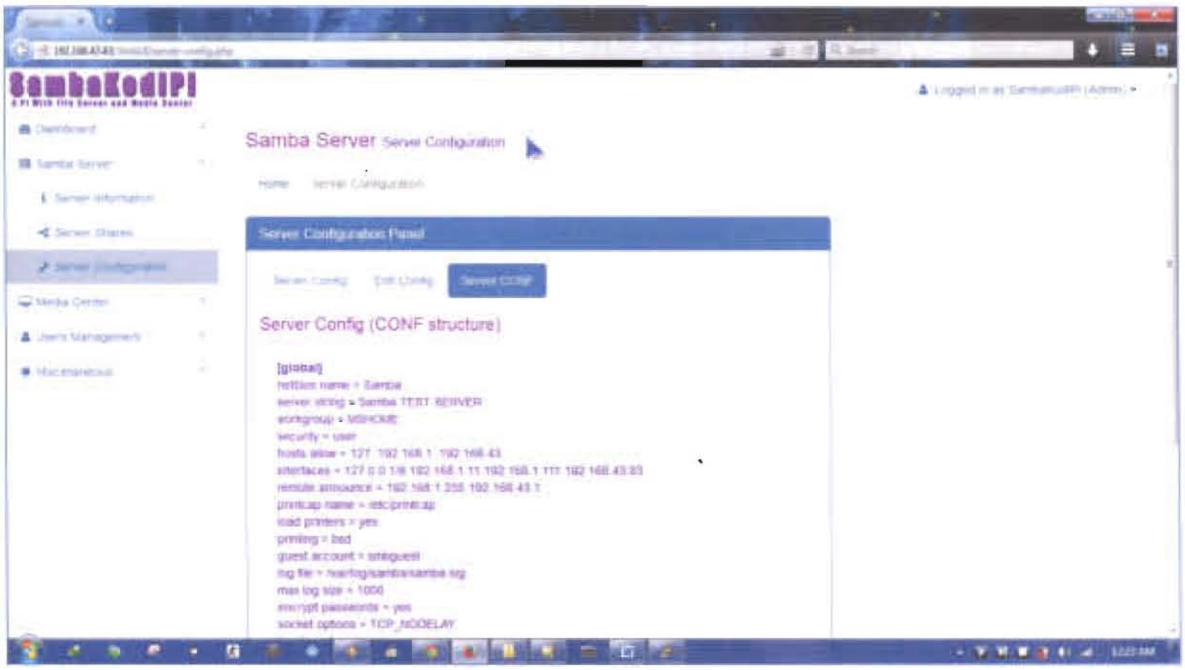
Server host name [?]
Samba

Server comment [?]
Samba TEST SERVER

Workgroup name [?]
MSHOME

Authentication method [?]
user

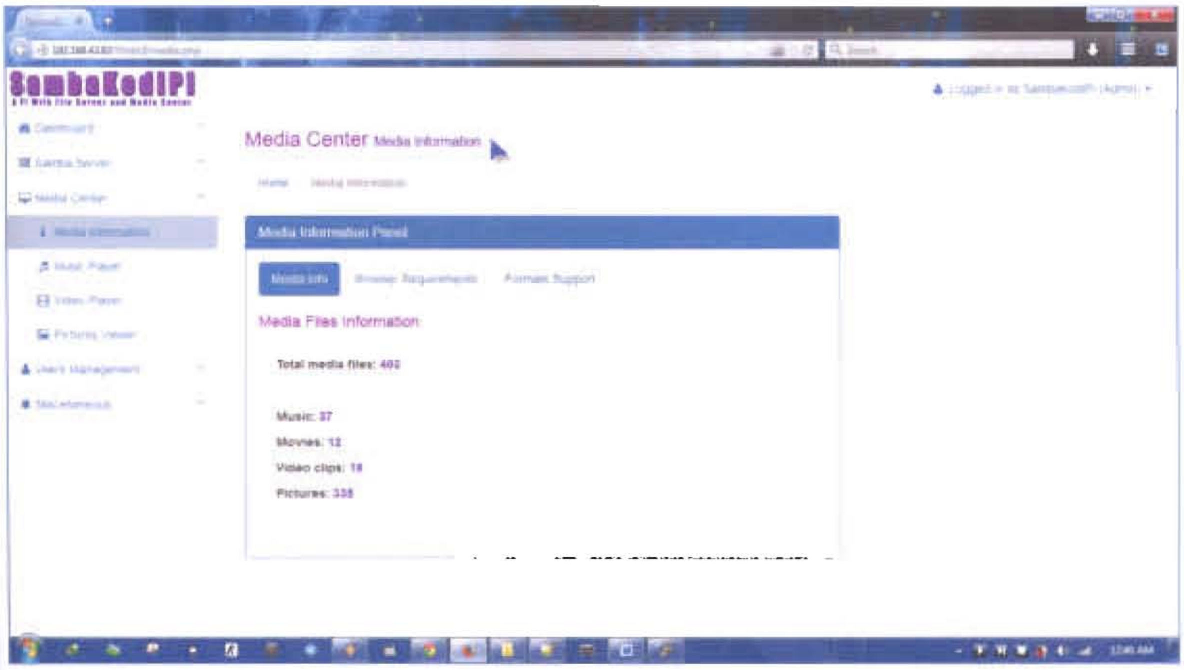
Figure 8.18: Screenshot for server-config.php page, tab edit config.



Server Config (CONF structure)

```
[global]
netbios name = Samba
server string = Samba TEST SERVER
workgroup = MSHOME
security = user
hosts allow = 127. 192.168.1. 192.168.43.
interfaces = 127.0.0.1/8 192.168.1.11 192.168.1.111 192.168.43.33
remote announce = 192.168.1.255 192.168.43.1
printcap name = /etc/printcap
load printers = yes
printing = bsd
guest account = smbguest
log file = /var/log/samba/samba.log
max log size = 1000
encrypt passwords = yes
socket options = TCP_NODELAY
```

Figure 8.19: Screenshot for server-config.php page, tab server CONF.



Media Files Information

Total media files: 402

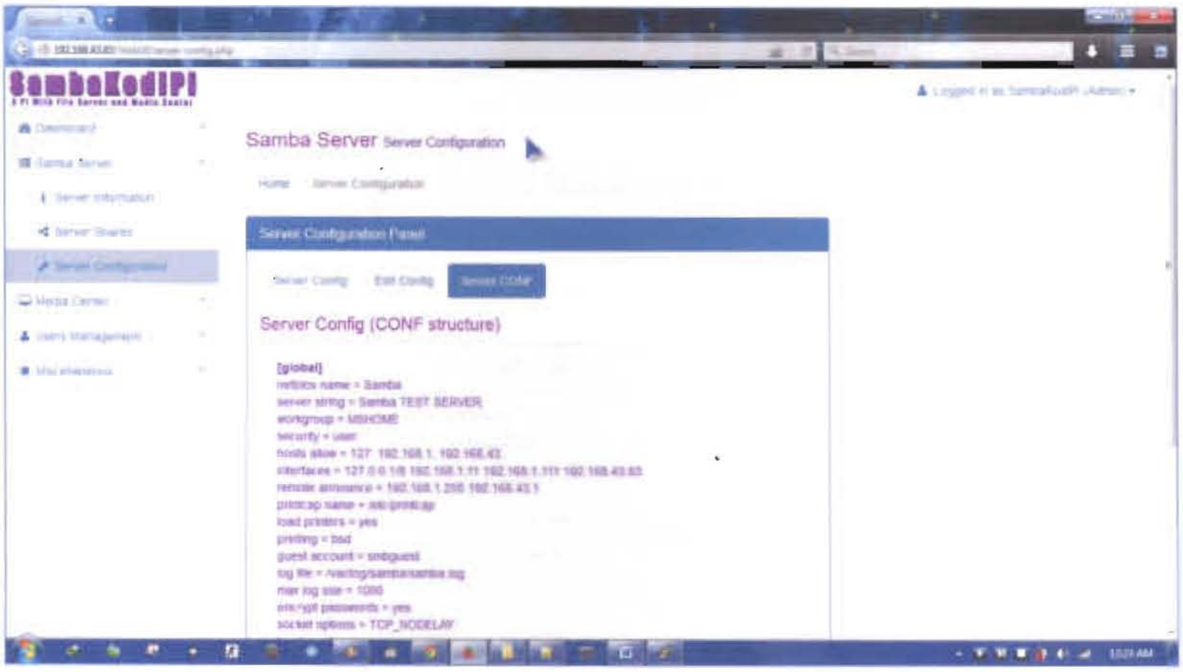
Music: 37

Movies: 12

Video clips: 18

Pictures: 335

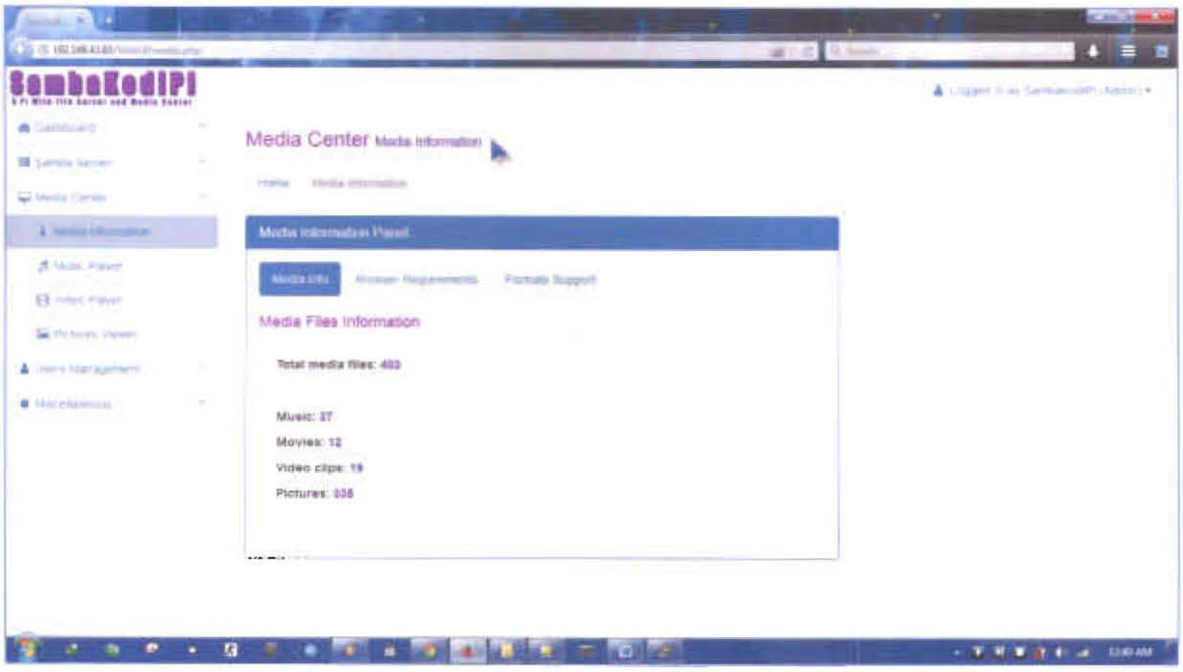
Figure 8.20: Screenshot for media.php page (media info tab).



Server Config (CONF structure)

```
[global]
netbios name = Samba
server string = Samba TEST SERVER
workgroup = MSHOME
security = user
hosts allow = 127. 192.168.1. 192.168.43.
interfaces = 127.0.0.1/8 192.168.1.11 192.168.1.111 192.168.43.83
remote announce = 192.168.1.255 192.168.43.1
printcap name = /etc/printcap
load printers = yes
printing = bsd
guest account = smbguest
log file = /var/log/samba/samba.log
max log size = 1000
encrypt passwords = yes
socket options = TCP_NODELAY
```

Figure 8.19: Screenshot for server-config.php page, tab server CONF.



Media Files Information

Total media files: 403

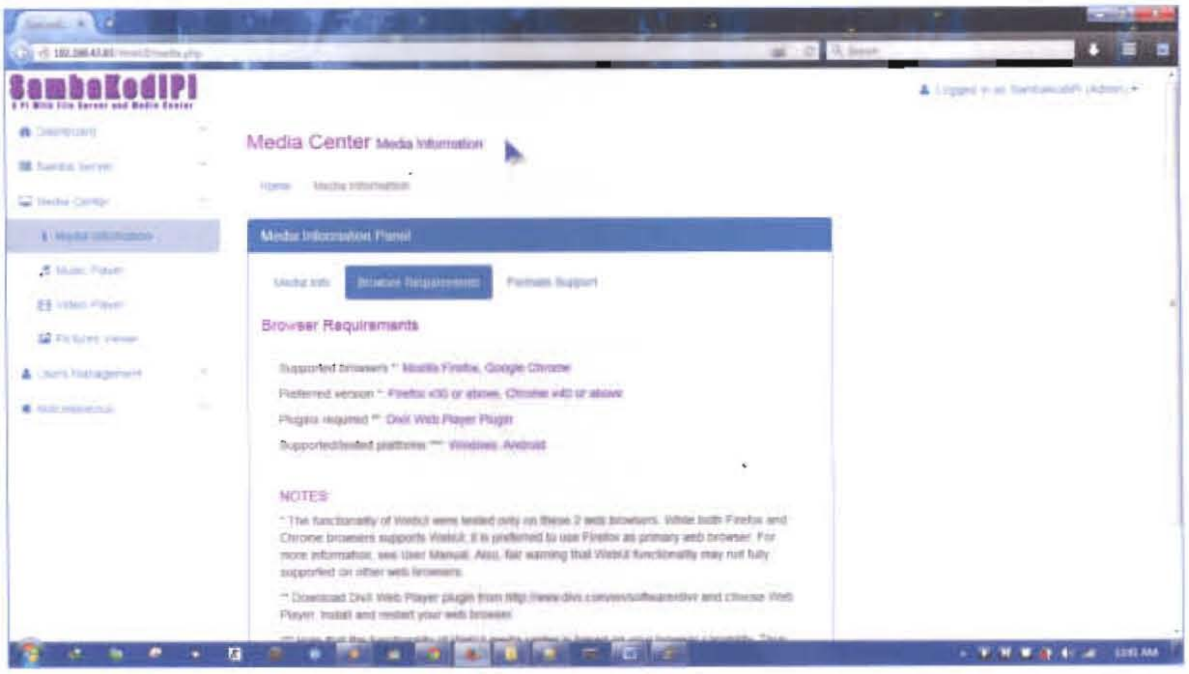
Music: 37

Movies: 12

Video clips: 19

Pictures: 335

Figure 8.20: Screenshot for media.php page (media info tab).



Media Center Media Information

Browser Requirements

Supported browsers * Mozilla Firefox, Google Chrome

Preferred version * Firefox v30 or above, Chrome v40 or above

Plugins required ** DivX Web Player Plugin

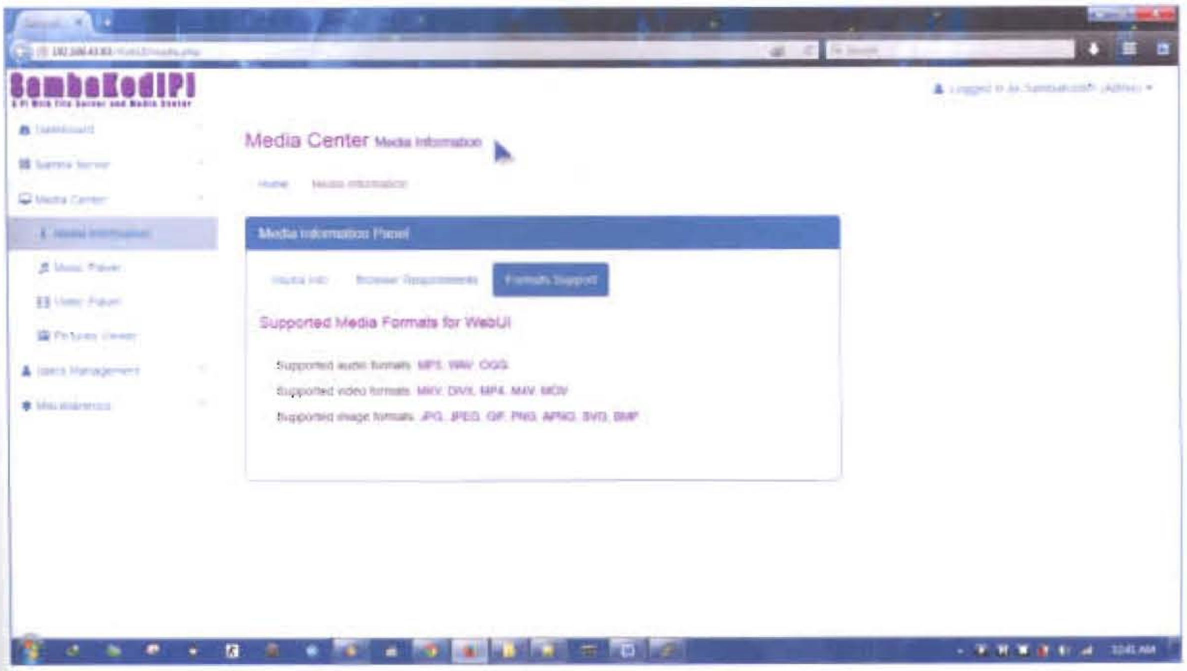
Supported/tested platforms *** Windows, Android

NOTES:

* The functionality of WebUI were tested only on these 2 web browsers. While both Firefox and Chrome browsers supports WebUI, it is preferred to use Firefox as primary web browser. For more information, see User Manual. Also, fair warning that WebUI functionality may not fully supported on other web browsers.

** Download DivX Web Player plugin from http://www.divx.com/en/software/divx and choose Web Player. Install and restart your web browser.

Figure 8.21: Screenshot for media.php page (browser requirements tab).



Media Center Media Information

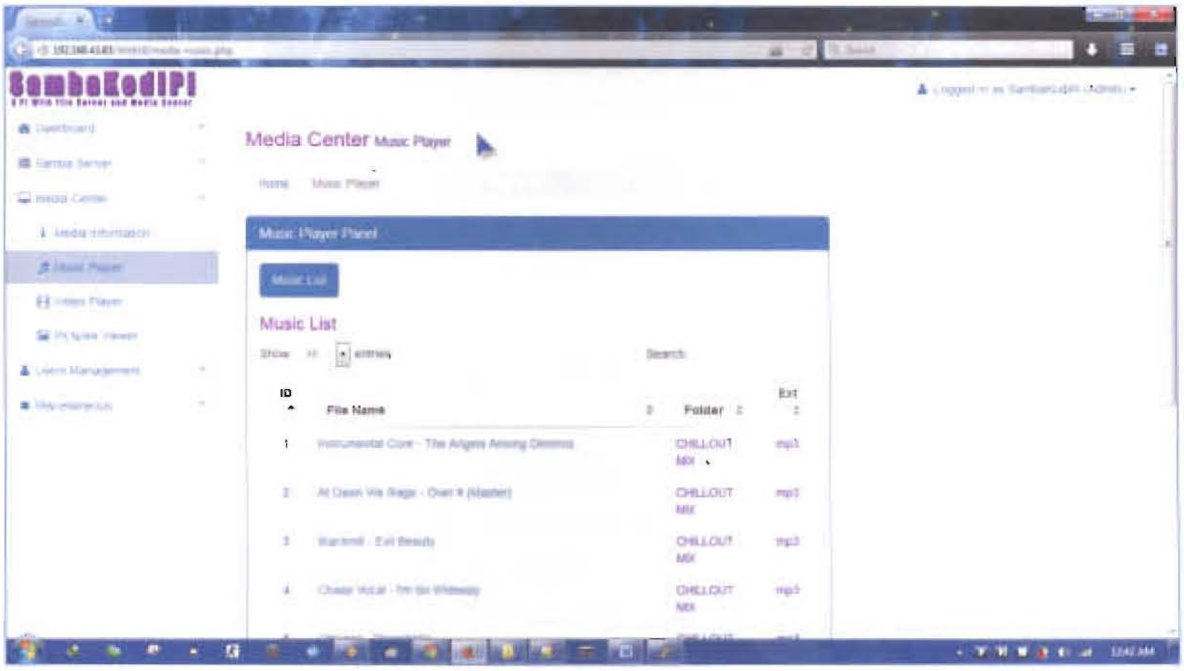
Supported Media Formats for WebUI

Supported audio formats: MP3, WAV, OGG

Supported video formats: MKV, DIVX, MP4, M4V, MOV

Supported image formats: JPG, JPEG, GIF, PNG, APNG, SVG, BMP

Figure 8.22: Screenshot for media.php page (formats support tab).


SambaKodiPi

Media Center Music Player

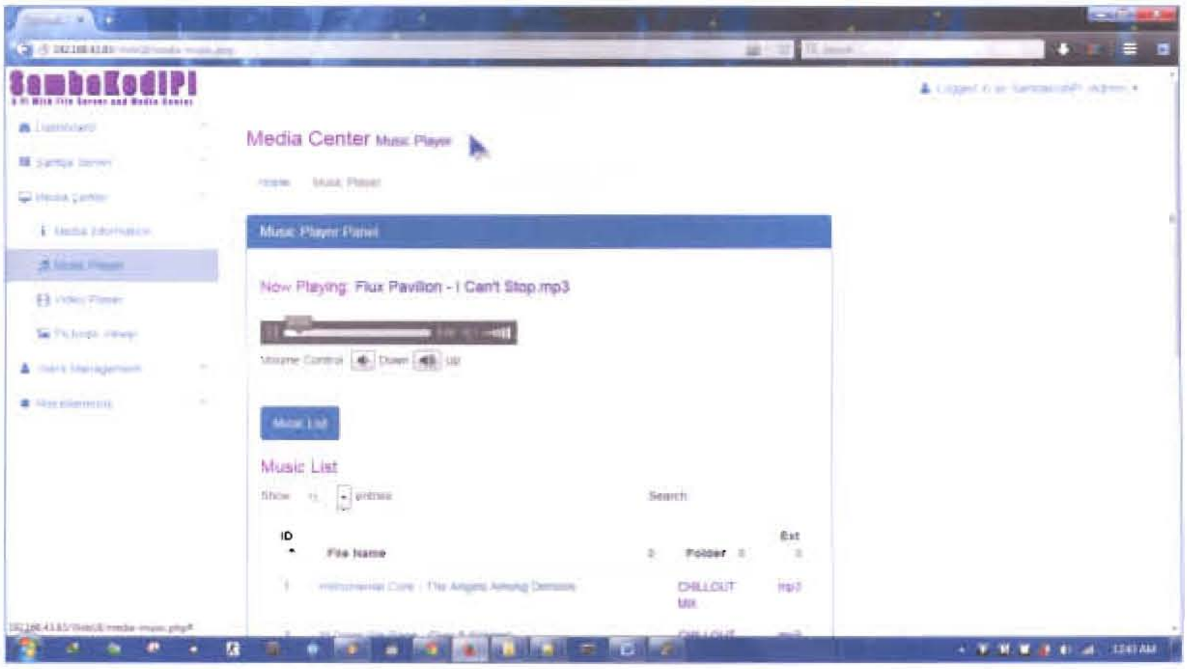
Music Player Panel

Music List

Music List

ID	File Name	Folder	Ext
1	Instrumental Core - The Angels Among Demons	CHILLOUT MIX	mp3
2	[illegible]	CHILLOUT MIX	mp3
3	[illegible]	CHILLOUT MIX	mp3
4	[illegible]	CHILLOUT MIX	mp3

Figure 8.23: Screenshot for media-music.php page.


SambaKodiPi

Media Center Music Player

Music Player Panel

Now Playing: Flux Pavilion - I Can't Stop.mp3

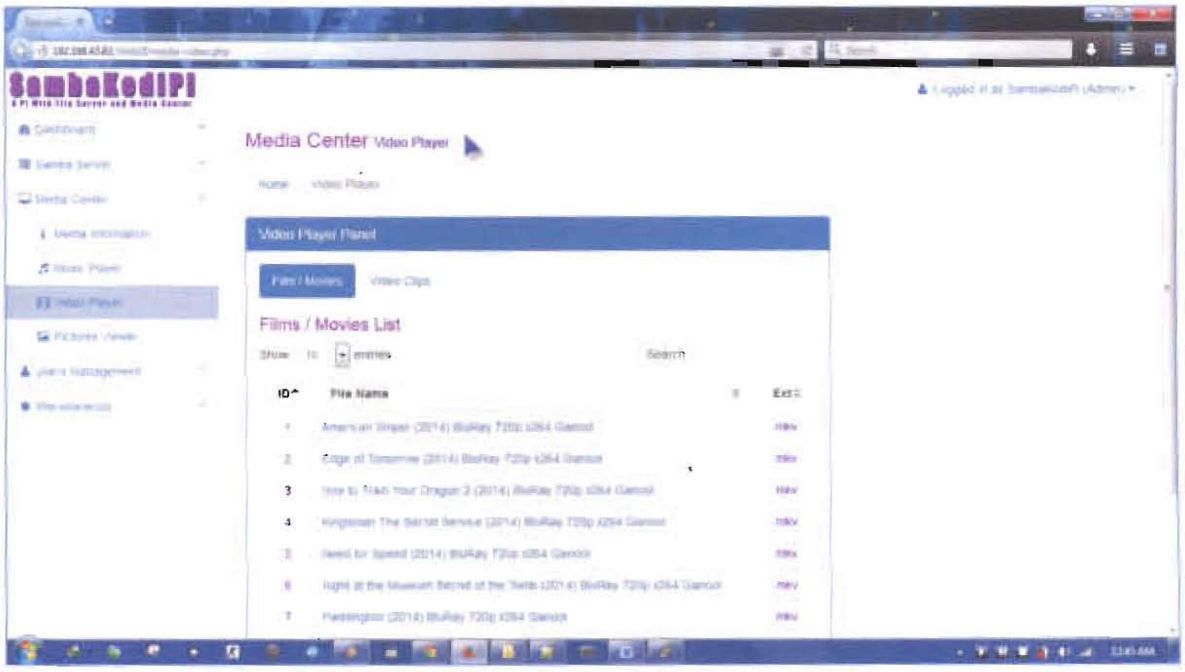
Volume Control Down Up

Music List

Music List

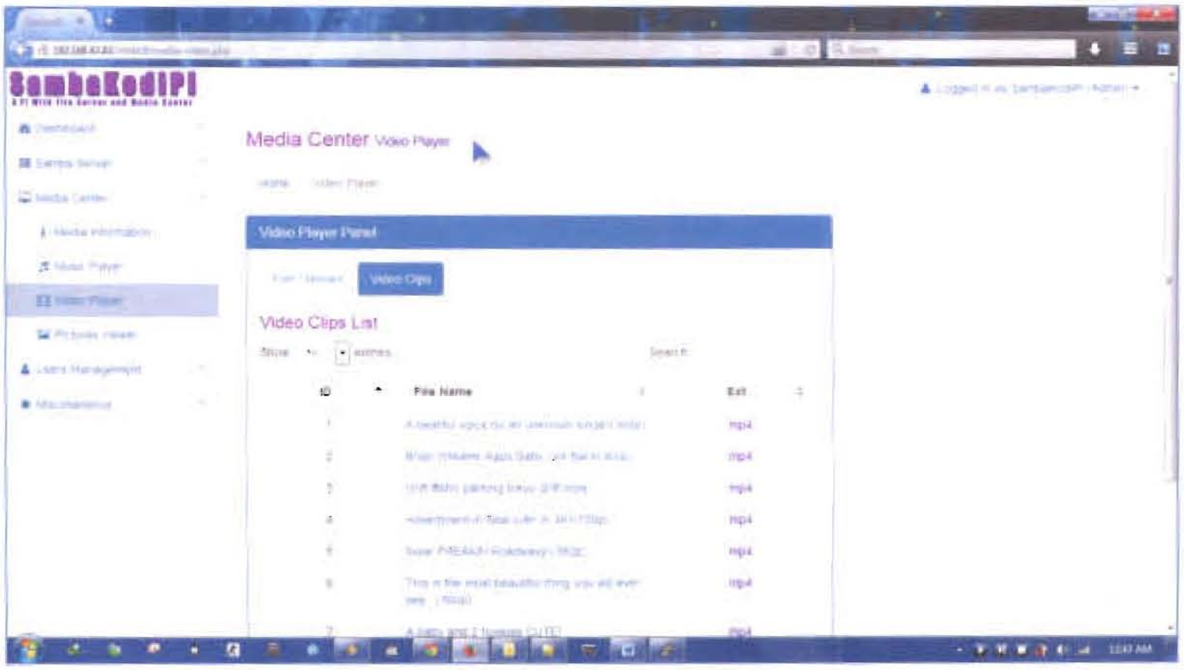
ID	File Name	Folder	Ext
1	Instrumental Core - The Angels Among Demons	CHILLOUT MIX	mp3

Figure 8.24: Screenshot for media-music.php page (with audio playback).



ID	File Name	Ext
1	American Sniper (2014) BluRay 720p x264 Ganool	mkv
2	Edge of Tomorrow (2014) BluRay 720p x264 Ganool	mkv
3	How to Train Your Dragon 2 (2014) BluRay 720p x264 Ganool	mkv
4	Kingsman The Secret Service (2014) BluRay 720p x264 Ganool	mkv
5	Need for Speed (2014) BluRay 720p x264 Ganool	mkv
6	Night at the Museum Secret of the Tomb (2014) BluRay 720p x264 Ganool	mkv
7	Paddington (2014) BluRay 720p x264 Ganool	mkv

Figure 8.25: Screenshot for media-video.php page (fi:m/movies tab).



ID	File Name	Ext
1	[illegible]	mp4
2	[illegible]	mp4
3	[illegible]	mp4
4	[illegible]	mp4
5	[illegible]	mp4
6	This is the most beautiful thing you will ever see [illegible]	mp4
7	A baby and 2 horses CUTE	mp4

Figure 8.26: Screenshot for media-video.php page (video clips tab).

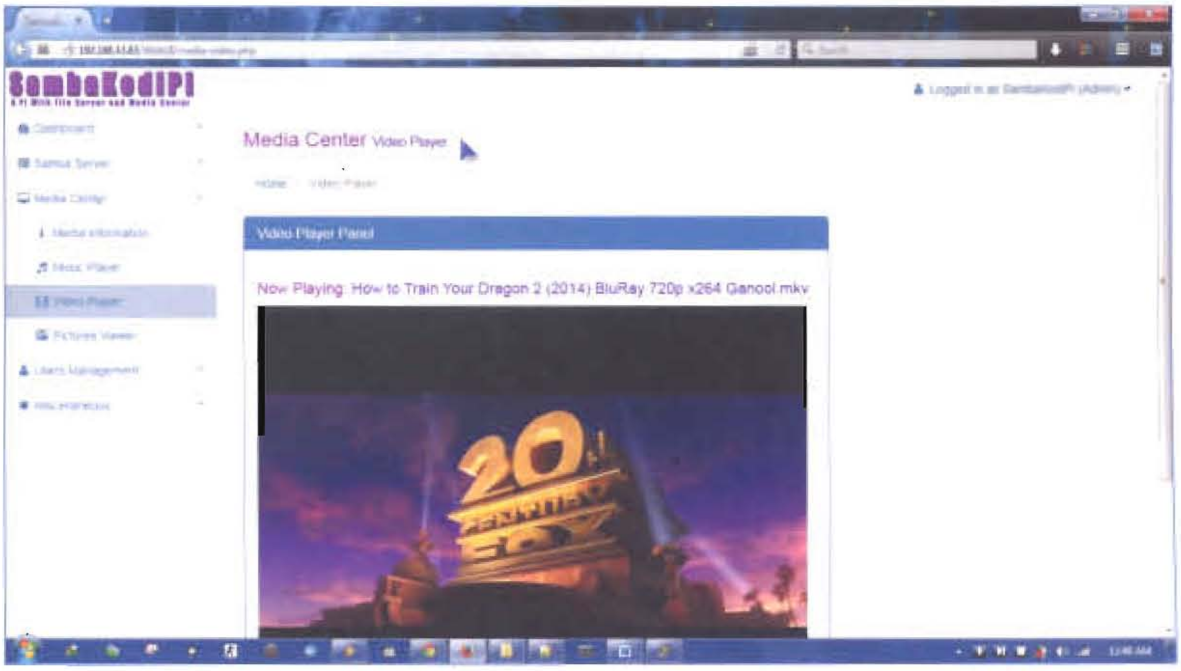

SambaKodiPI
Media Center Video Player
Video Player Panel
Now Playing: How to Train Your Dragon 2 (2014) BluRay 720p x264 Ganool.mkv

Figure 8.27: Screenshot for media-video.php page (with video playback).

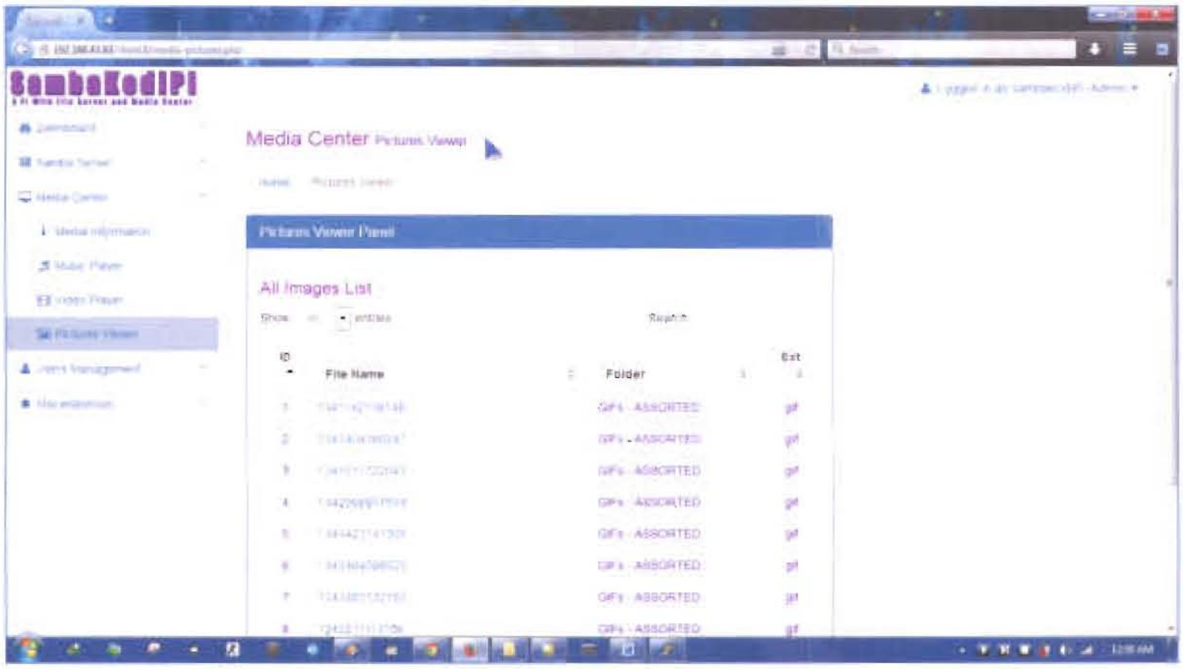
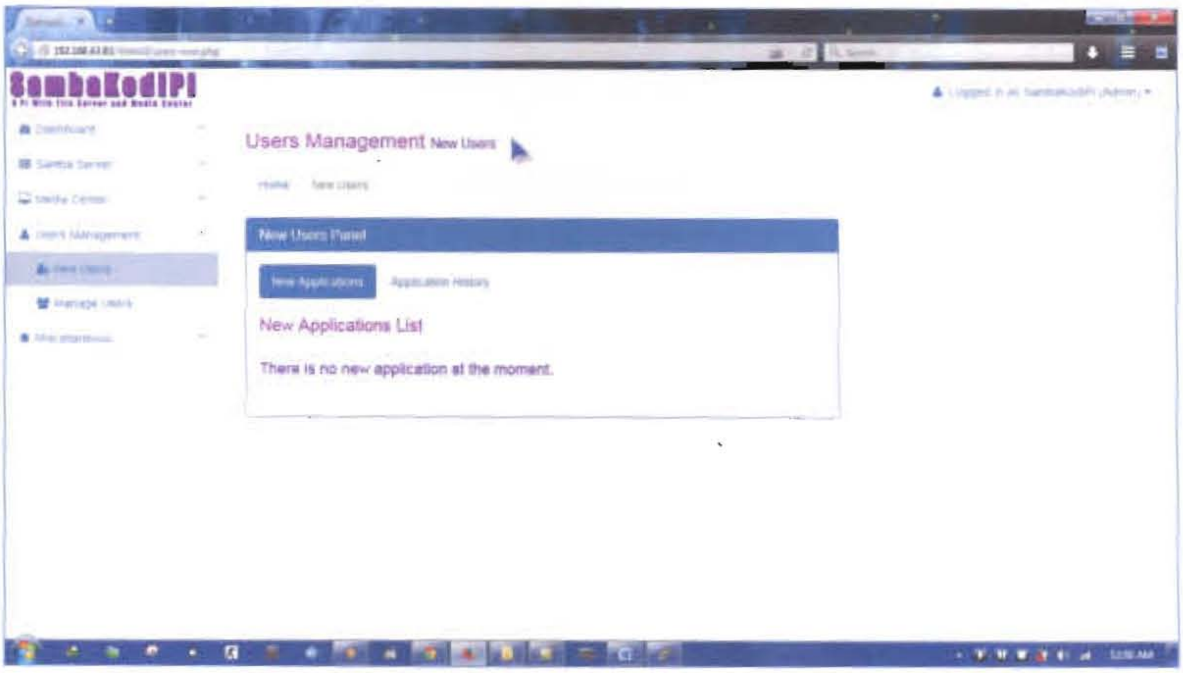


Figure 8.28: Screenshot for media-pictures.php page.


SambaKodiPI

Users Management New Users

Home New Users

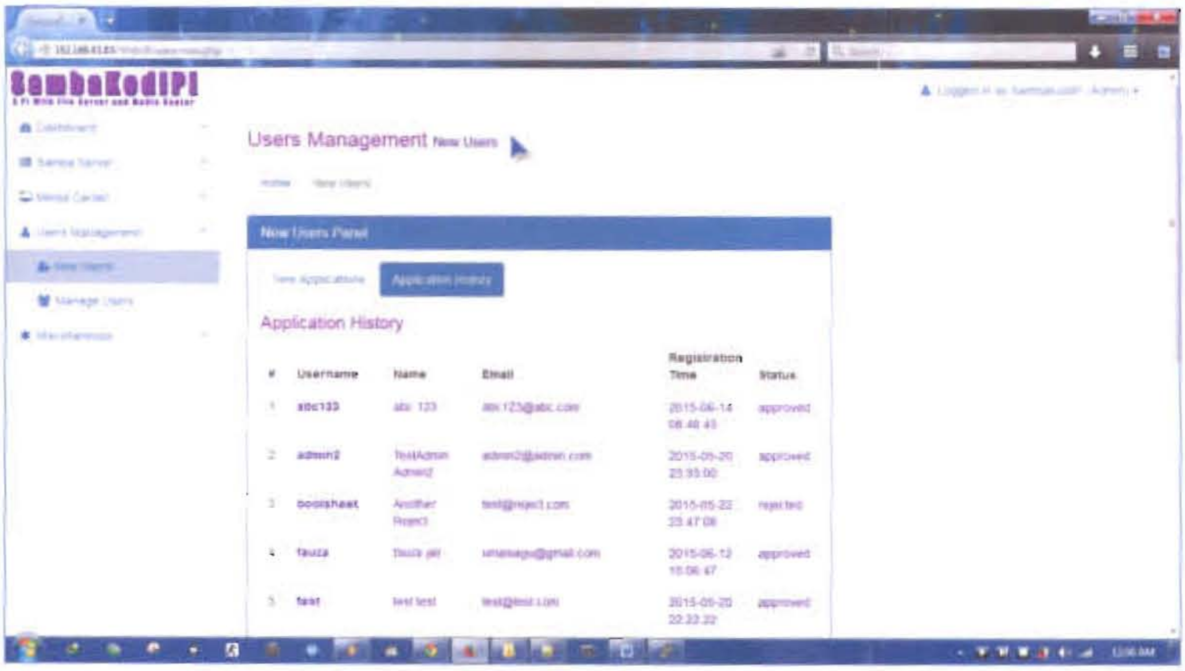
New Users Panel

New Applications Application History

New Applications List

There is no new application at the moment.

Figure 8.29: Screenshot for users-new.php page (new applications tab).


SambaKodiPI

Users Management New Users

Home New Users

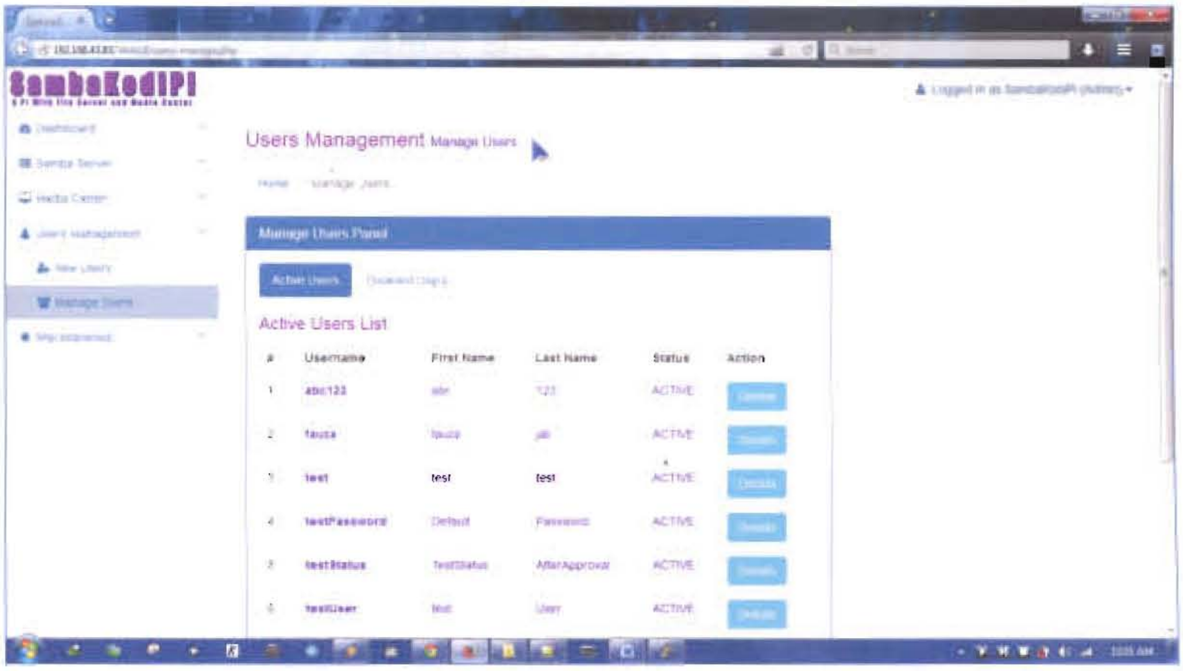
New Users Panel

New Applications Application History

Application History

#	Username	Name	Email	Registration Time	Status
1	abc123	abc 123	abc123@abc.com	2015-06-14 08:48:43	approved
2	admin2	TestAdmin Admin2	admin2@admin.com	2015-05-20 23:33:00	approved
3	boolsheet	Another Project	test@roject.com	2015-05-22 23:47:08	rejected
4	fauza	fauza [illegible]	[illegible]@gmail.com	2015-06-12 15:06:47	approved
5	test	test test	test@test.com	2015-05-20 22:22:22	approved

Figure 8.30: Screenshot for users-new.php page (application history tab).

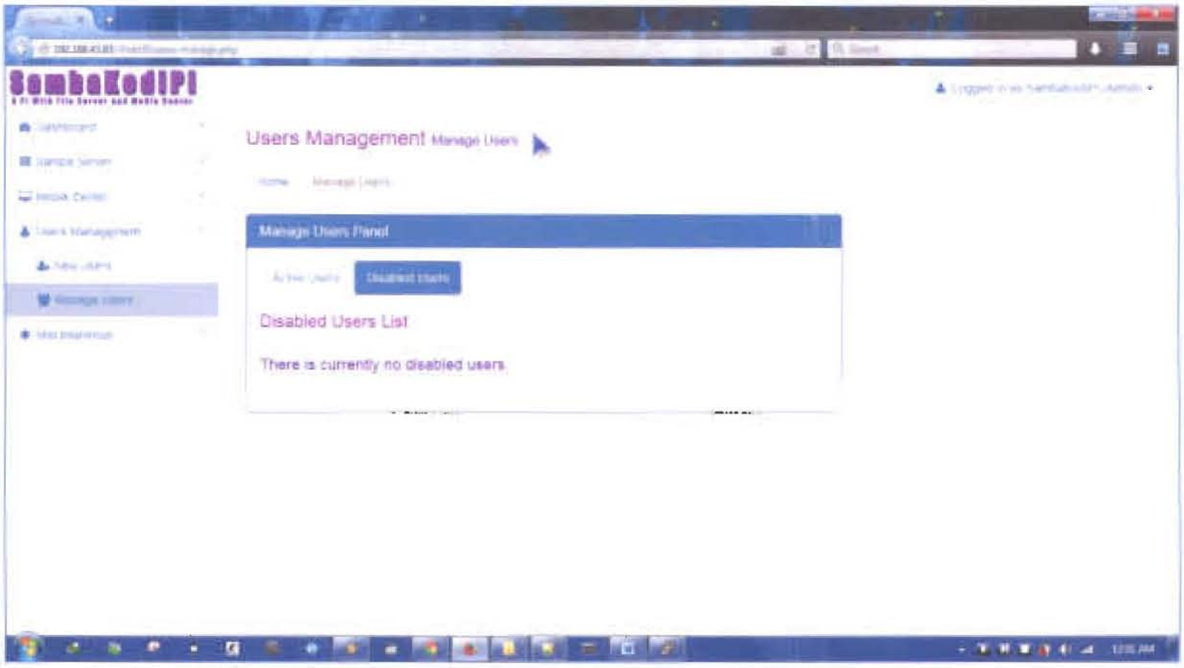

Users Management Manage Users

Manage Users Panel

Active Users List

#	Username	First Name	Last Name	Status	Action
1	abc123	abc	123	ACTIVE	[illegible]
2	fauza	fauza	[illegible]	ACTIVE	Details
3	test	test	test	ACTIVE	Details
4	testPassword	Default	Password	ACTIVE	Details
5	testStatus	TestStatus	AfterApproval	ACTIVE	Details
6	testUser	test	User	ACTIVE	Details

Figure 8.31: Screenshot for users-manage.php page (active users tab).

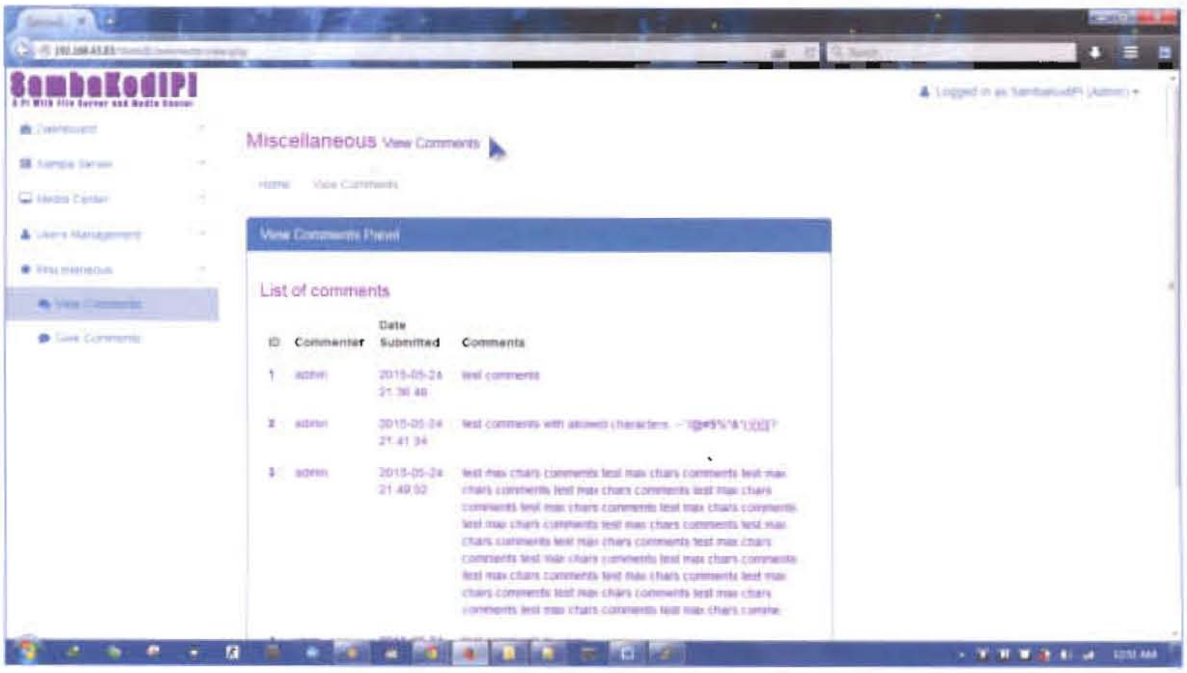

Users Management Manage Users

Manage Users Panel

Disabled Users List

There is currently no disabled users.

Figure 8.32: Screenshot for users-manage.php page (disabled nsers tab).


SambaKodiPI

Miscellaneous View Comments

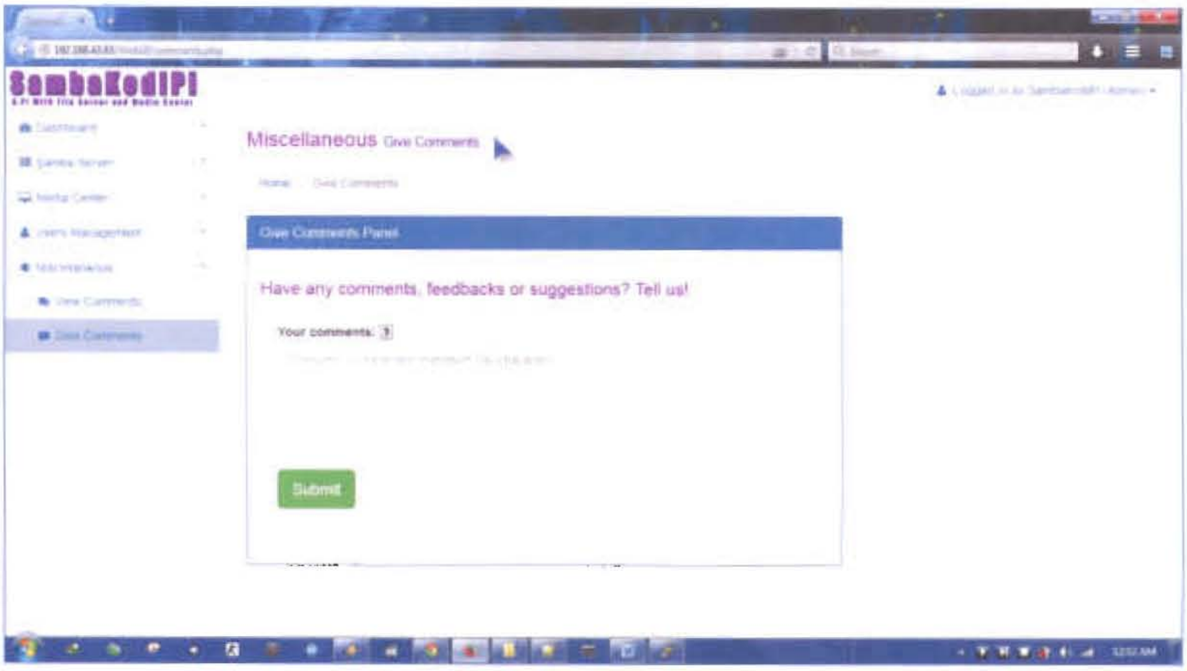
Home / View Comments

View Comments Panel

List of comments

ID	Commenter	Date Submitted	Comments
1	admin	2015-05-24 21:36:48	test comments
2	admin	2015-05-24 21:41:34	test comments with allowed characters. ~!@#$%^&*()[]{}?
3	admin	2015-05-24 21:49:02	test max chars comments test max chars comments test max chars comments test max chars comments test max chars comments test max chars comments test max chars comments test max chars comments test max chars comments test max chars comments test max chars comments test max chars comments test max chars comments test max chars comments test max chars comments test max chars comments test max chars comments test max chars comments test max chars comme

Figure 8.33: Screenshot for comments-view.php page.


SambaKodiPI

Miscellaneous Give Comments

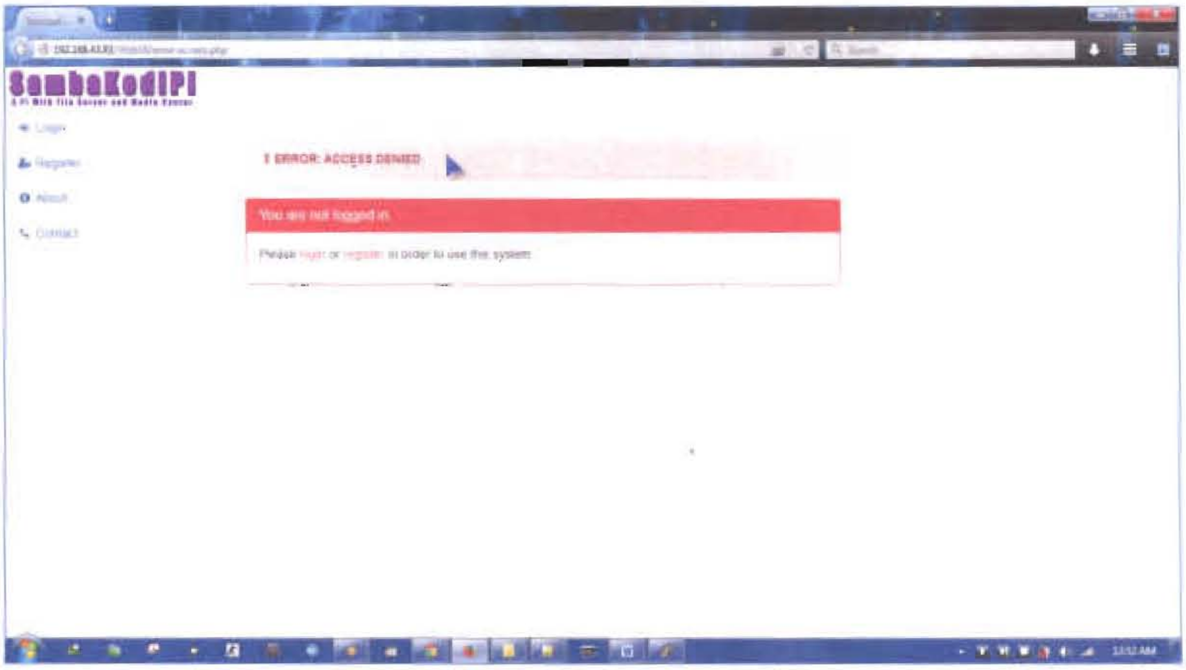
Home / Give Comments

Give Comments Panel

Have any comments, feedbacks or suggestions? Tell us!

Your comments: [illegible]

Figure 8.34: Screenshot for comments.php page.

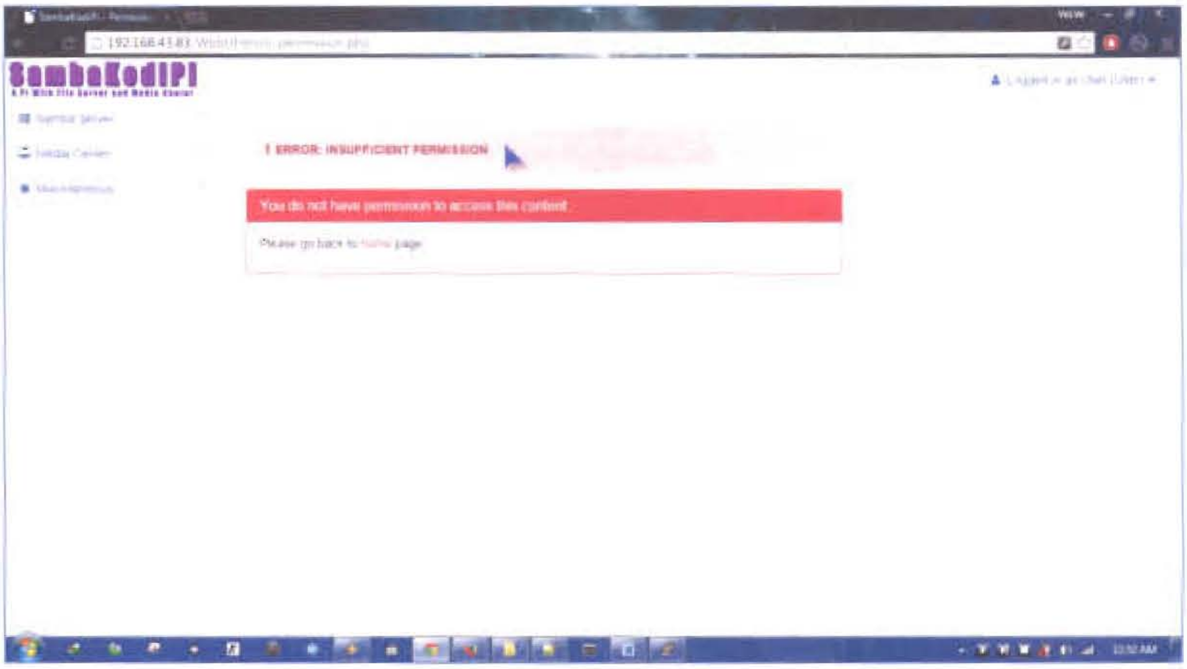


! ERROR: ACCESS DENIED

You are not logged in.

Please login or register in order to use the system

Figure 8.35: Screeushot for error-access.php page.



! ERROR: INSUFFICIENT PERMISSION

You do not have permission to access this content

Please go back to home page

Figure 8.36: Screenshot for error-permissions.php page.

8.4 Appendix D – Component Testing Results

Table 8.1: Unit testing results for system components and devices.

Name	Type	Model/Version	Condition	Functionality
Raspberry Pi	Device	Model B+	OK	OK
Modem/Router	Device	Speedtouch585 v6	OK	OK
PC Monitor	Device	Acer x193HQ	OK	OK
Stereo speakers	Device	Logitech Z103	OK	OK
Laptop	Device	Acer Aspire 4750G	OK	OK
Keyboard	Device	Logitech K120	OK	OK
Mouse	Device	Logitech M185	OK	OK
Laptop	Device	Acer 4750G	OK	OK
Smartphone	Device	Sony Xperia E	OK	OK
Smartphone	Device	Sony Xperia Z1	OK	OK
USB drive	Device	DT G2 8GB	OK	OK
External HDD	Device	WD 500GB	OK	OK
Raspbian	Software	3.12.22+	OK	OK
Kodi	Software	14.1	OK	OK
Samba	Software	3.6.6	OK	OK
GAdmin-Samba	Software	0.2.9	OK	OK
Apache	Software	2.2.22 (Debian)	OK	OK
MySQL	Software	14.14, dist 5.5.41	OK	OK
PHP5	Software	5.4.39	OK	OK
Dreamweaver	Software	CS5.5 v11.5	OK	OK
Notepad++ (Win)	Software	6.5.5	OK	OK
Geany (Linux)	Software	1.22	OK	OK
DiffMerge	Software	4.2.0	OK	OK
WinRAR	Software	3.5.1	OK	OK
Mozilla Firefox	Software	36.0	OK	OK
Google Chrome	Software	41.0.2272.101 m	OK	OK
HDMI cable	Cable	Standard HDMI	OK	OK

LAN cables	Cable	Standard LAN	OK	OK
HDMI to DVI	Adapter	F-M adapter	OK	OK
VGA to VGA	Adapter	M-M adapter	OK	OK

Table 8.2: Unit testing results for each of WebUI pages.

No	Page name	Structure	Correctness	Functionality
1	index.php	OK	OK	OK
2	login.php	OK	OK	OK
3	register.php	OK	OK	OK
4	about.php	OK	OK	OK
5	contact.php	OK	OK	OK
6	home.php	OK	OK	OK
7	system.php	OK	OK	OK
8	system-config.php	OK	OK	OK
9	server.php	OK	OK	OK
10	server-shares.php	OK	OK	OK
11	server-config.php	OK	OK	OK
12	media.php	OK	OK	OK
13	media-music.php	OK	OK	OK
14	media-video.php	OK	OK	OK
15	media-pictures.php	OK	OK	OK
16	users-new.php	OK	OK	OK
17	users-manage.php	OK	OK	OK
18	comments.php	OK	OK	OK
19	comments-view.php	OK	OK	OK
20	profile.php	OK	OK	OK
21	account.php	OK	OK	OK
22	error-access.php	OK	OK	OK
23	error-permission.php	OK	OK	OK
24	index.html	OK	OK	OK

Table 8.3: Unit testing results for PHP functions in SKP-Base.php file.

No	Function name	Structure	Correctness	Functionality
1	initSession()	OK	OK	OK
2	authUserLogin()	OK	OK	OK
3	initLogoutUser()	OK	OK	OK
4	initAuthorizedUserType()	OK	OK	OK
5	isAuthorized()	OK	OK	OK
6	GetSQLValueString()	OK	OK	OK
7	dbaseQueryToAuthUser()	OK	OK	OK
8	freeUserVar()	OK	OK	OK
9	isAdminSection()	OK	OK	OK
10	isUserSection()	OK	OK	OK
11	processUserReg()	OK	OK	OK
12	getNewApplications()	OK	OK	OK
13	getActiveUsers()	OK	OK	OK
14	getDisabledUsers()	OK	OK	OK
15	approveNewUser()	OK	OK	OK
16	updateApplicationStatus()	OK	OK	OK
17	resetUserPassword()	OK	OK	OK
18	disableUserAccount()	OK	OK	OK
19	enableUserAccount()	OK	OK	OK
20	getProfileData()	OK	OK	OK
21	updateProfileData()	OK	OK	OK
22	getAccountData()	OK	OK	OK
23	changeUserPassword()	OK	OK	OK
24	getAllUsernames()	OK	OK	OK
26	isUsernameExist()	OK	OK	OK
27	insertComment()	OK	OK	OK
28	getAllComments()	OK	OK	OK

Table 8.4: Unit testing results for PHP functions in SKP-FS.php file.

No	Function name	Structure	Correctness	Functionality
1	isNormalArray()	OK	OK	OK
2	isIndexedArray()	OK	OK	OK
3	displayIndexedArray()	OK	OK	OK
4	isInNArray()	OK	OK	OK
5	displayNArray()	OK	OK	OK
6	printFound()	OK	OK	OK
7	setValueInArray()	OK	OK	OK
8	KV_isInAArray()	OK	OK	OK
9	K_isInAArray()	OK	OK	OK
10	V_isInAArray()	OK	OK	OK
11	S_isInAArray()	OK	OK	OK
12	setValueInAArray_KV()	OK	OK	OK
13	setValueInAArray_K()	OK	OK	OK
14	setValueInAArray_V()	OK	OK	OK
15	displayCONF()	OK	OK	OK
16	parseCONF()	OK	OK	OK
17	getSectionName()	OK	OK	OK
18	getAllSectionName()	OK	OK	OK
19	printAllSectionName()	OK	OK	OK
20	getSectionContent()	OK	OK	OK
21	getAArrayLevel()	OK	OK	OK
22	ERROR_FS()	OK	OK	OK
23	SambaCONF class	OK	OK	OK
24	SambaCONF()*	OK	OK	OK
25	parseCONF()*	OK	OK	OK
26	displayCONF()*	OK	OK	OK
27	displayGlobalSection()*	OK	OK	OK
28	displayAllShareSection()*	OK	OK	OK
29	getAllShareSection()*	OK	OK	OK

30	printAllShareNames()*	OK	OK	OK
31	getTotalShares()*	OK	OK	OK
32	getSectionContent()*	OK	OK	OK
33	removeSection()*	OK	OK	OK
34	getBackupCONF()*	OK	OK	OK
35	getParsedCONF()*	OK	OK	OK
36	restoreParsedCONF()*	OK	OK	OK
37	processNewShare()*	OK	OK	OK
38	addNewShare()*	OK	OK	OK
39	processServerEdit()*	OK	OK	OK
40	overwriteGlobalSection()*	OK	OK	OK
41	CONF_ArrayToString()*	OK	OK	OK
42	writeIntoSambaCONF()*	OK	OK	OK

Note: All functions marked with * are all class functions of SambaCONF class.

Table 8.5: Unit testing results for PHP functions in SKP-MC.php file.

No	Function name	Structure	Correctness	Functionality
1	ERROR_MC()	OK	OK	OK
2	scanDirectory()	OK	OK	OK
3	filterPathInfo_MKV()	OK	OK	OK
4	filterPathInfo()	OK	OK	OK
5	scanDirectoryWithEXT()	OK	OK	OK
6	displayPathInfo()	OK	OK	OK
7	getAbsoluteVideoPath()	OK	OK	OK
8	getFullAudioPath()	OK	OK	OK
9	getFullImagePath()	OK	OK	OK
10	getAudioFilesCount()	OK	OK	OK
11	getVideoFilesCount()	OK	OK	OK
12	getImageFilesCount()	OK	OK	OK
13	countFilesWithEXT()	OK	OK	OK

8.5 Appendix E – Integration Testing Results

Table 8.6: Integration testing results for each of SambaKodiPi components.

Name	Test Criteria	Result
Raspberry Pi component	Inspection on physical defects of Pi board -checking for faulty peripheral parts and slots	Pass
	Inspection on plugged devices connection -checking that all external devices are plugged correctly	Pass
	Functionality of Raspberry Pi when switched on -whether the Raspberry Pi is starting when switched on	Pass
	Inspection on LED lights when switched on -verifying that the LED lights are on and blipping	Pass
	Inspection on of Raspbian OS installation -checking that the OS is complete and with broken parts	Pass
	Functionality of Raspbian OS -checking that the OS is functioning properly	Pass
File server component	Inspection on Samba program installation -checking that Samba is complete with no broken parts	Pass
	Functionality of Samba program -checking that Samba server is running properly	Pass
	Accessibility of Samba configuration file -checking that the file exists and can be accessed	Pass
	Functionality of Samba server shares -checking that Samba is capable of sharing folders and they are visible and accessible from outside using laptop	Pass
Media center	Inspection on Kodi program installation -checking that Kodi is complete with no broken parts	Pass
	Compatibility of Kodi program -checking that Kodi is running normally on Raspbian	Pass
	Functionality of Kodi program -checking that Kodi can search and play media files	Pass

	Accessibility of media files used by Kodi -checking that the media files are accessible by Kodi	Pass
LAMP webserver component	Inspection on Apache2, MySQL and PHP5 installation -checking that the programs exits and are complete	Pass
	Functionality of LAMP webserver -checking that the webserver is hosting properly by accessing the default server page in /var/www/	Pass
	Accessibility of LAMP webserver -checking that the default webserver page is accessible through localhost and from outside on laptop PC	Pass
	Functionality of PHPMyAdmin sub-component -checking that it is accessible on browser through localhost using proper login credential	Pass
WebUI component	Inspection on WebUI web pages (*.php and *.html) -checking that all web pages are complete and exist	Pass
	Inspection on WebUI files -checking that all WebUI files (the CSS, JavaScript, database connection and PHP function files) are complete and exist in their proper folder location	Pass
	Inspection on WebUI files and structure -checking that the folders are in their intended location	Pass
	Inspection on each web page extended source code -checking the syntax, parameters and correctness of extended source code sections that call or use any CSS, JavaScript or PHP code or function	Pass
	Proper interface display of each WebUI page -checking that each page is displayed properly as intended (also validating WebUI interface)	Pass
	Proper overall syntax of each WebUI page -checking that each page is coded properly (no extra or missing tags, using browser *View page source* function)	Pass

Table 8.7: Integration testing results for interaction between integrated components.

Interaction	Test Criteria	Result
Interaction between Raspbian with file server integrated components	-checking that Rasbian can access the folders, files and programs of Samba	Pass
	-Samba can run normally without unexpected error/crash	Pass
	-checking that Raspbian can create and access server share folders	Pass
Interaction between Raspbian with media center integrated components	-checking that Rasbian can access the folders, files and programs of Kodi	Pass
	-Kodi can run normally without unexpected error/crash	Pass
	-checking that Raspbian can create and access media folders used by Kodi	Pass
Interaction between Raspbian with LAMP webserver integrated components	-checking that Rasbian can access the webserver folders and PHPMyAdmin site	Pass
	-checking that the database tables for SambaKodiPi exist and accessible through PHPMyAdmin	Pass
	-checking that Apache2, MySQL and PHP5 are compatible with Raspbian and can run normally without unexpected error/crash	Pass
Interaction between Raspbian with WebUI integrated components	-checking that Rasbian can access the files in SambaKodiPi folder and its subfolders under */var/www/* root folder	Pass
	-checking that Raspbian can access the WebUI from localhost and login into the WebUI	Pass
Interaction between WebUI with LAMP webserver integrated	-checking that the LAMP webserver is hosting the WebUI pages correctly by accessing the WebUI from laptop PC through the assigned IP address	Pass

components	for Raspberry Pi device	
	-checking that each of the WebUI pages are accessible from laptop PC (using Admin account)	Pass
Interaction between WebUI and file server integrated components	-verifying that WebUI can read and access Samba configuration file under */etc/samba/* folder	Pass
	-verifying that WebUI has access and can create files inside */home/pi/samba/* folder to store backup for Samba configuration file	Pass
	-checking that the WebUI parser function can parse the content of Samba configuration file correctly and verify its parsed content with actual content in *smb.conf* file (this was done through test page: *test.php*)	Pass
	-checking that the WebUI backup function can make a copy *smb.conf* file into its designated backup folder (this was done through test page: *test.php*)	Pass
Interaction between WebUI and media center integrated components	-checking that the WebUI has access to media file root folder and scan the directory for media files (this was done through test page: *test.php*)	Pass
	-checking that the WebUI can play audio files inside music root folder (this was done through test page: *test.php*)	Pass
	-checking that the WebUI can video files inside video root folder (this was done through test page: *test.php*)	Pass
	-checking that the WebUI can access and display image files inside iamge root folder (this was done through test page: *test.php*)	Pass

8.6 Appendix F – System Testing Results

Table 8.8: Functional testing results for final system.

Test ID	Test Steps	Expected Function	Actual Function	Status	Remarks
1	Test final system functions with its user requirement: File server capability	System to have a file server function in a private network	File server function hosted by Samba, in a LAN network	Pass	-
		The file server to host and share files to authorized clients	Samba file server capable to host and share any files to both authorized and guest users	Pass	-
		Client to have at least read permission to shared data in server share	Samba file server allows both read/write and read-only permissions for its server share	Pass	-
2	Test final system functions with its user requirement: Media center capability	System to have a media center function provided for users	Primary media center function is provided by Kodi program, secondary provided by WebUI component using web player	Pass	WebUI media center function is an additional secondary function
		The media center able to access, open and play various media files format stored in system storage	Kodi program is capable of playing various audio, video and image formats, while WebUI web player supports limited formats	Pass	
		Users to be able to listen to and view media playback through standard audio and video output devices	Kodi program is capable to output media files playback to both PC monitor and HDTV using HDMI cable while WebUI media center outputs audio using HTML player and video using DivX web player	Pass	
3	Test final system functions with its user requirement:	Final system to have a web-based user interface to interact with file server function	Final system includes WebUI component that is hosted by LAMP webserver, accessible to users through web	Pass	-

	Main system interface		browsers, providing file server configuration as its main function.		\
4	Test final system functions with its system interface requirements: File server function	System interface to have login function before user can start using its file server function	WebUI has login function, accessible to registered users only	Pass	-
		System interface to have file sharing management including add new share, modify or delete current share	WebUI has file server function where admin can view server shares, add new shares, edit or delete current share	Pass	-
		System interface to have server configuration functions to change security settings and shares access	WebUI has file server function where admin can configure various basic server settings and also advanced server settings	Pass	-

Table 8.9: Security testing results for final system.

Test ID	Test Case	Test Steps	Expected Result	Actual Result	Status	Remarks
1	Test WebUI login access restriction	User try to access any page manually by URL that requires login	WebUI will not display the requested page and redirect user to error page	Access error page is displayed, stating that user is not logged in	Pass	-
2	Test WebUI access restriction for admin-only content	Logged in normal user try to access any page manually by URL that requires admin rights	WebUI will not display the requested page and redirect user to error page	Permission error page is displayed, stating that user has insufficient permission to view the page	Pass	-
3	Test WebUI subfolders restriction	User try to access certain folders that store WebUI database connection info and its PHP functions	Web browser shall display error where the page or content is not found on server	Internal server error is displayed and the requested folder remains inaccessible	Pass	-

Table 8.10: Accuracy testing results for final system.

Test ID	Test Case	Test Steps	Expected Result	Actual Result	Status	Remarks
1	Test Kodi program to display list of media files	Using Kodi browser function for its pictures, music and videos	The list of media files stored in media folder shall be listed properly in Kodi	List of media files are listed down correctly, sorted by name (default)	Pass	-
2	Test Kodi program to play the correct file selected	Using Kodi to play media files	Kodi shall play the selected file accordingly	The media file being played is the one selected	Pass	-
3	Test all WebUI links of its own web pages	Browsing all WebUI pages using admin account	All web page shall be linked correctly and display the right page	All web page are linked correctly and display the right page	Pass	-
4	Test all WebUI forms and user input	Using the registration form	The form shall accept user input and register new user correctly	The form accepted user input and registered new user correctly	Pass	Each form text input has its own input validation pattern, and each form has its own validation function
		Using the comment form	The form shall accept user input and store user comments into database table	The form accepted user input and stored comments into database table	Pass	
		Using the change user profile form	The form shall enable users to edit their email, first name and last name while username field is disabled	The form accepted inputs for user profile change correctly and username field is disabled	Pass	
		Using the change user password form	The form shall enable users to change their password	The form accepted user input to and updated user password correctly	Pass	
		Using the adding new	The form shall accept user	The form accepted user	Pass	

		share form	input and add new server share	input and added new server share correctly		
		Using the edit existing share form	The form shall display the selected share to be edited on its input fields and update the share details	The form displayed the selected share and updated the share details correctly	Pass	
		Using the edit server settings form	The form shall display the server settings correct value on its input fields and edit server settings correctly	The form displayed the server settings values correctly and updated server settings correctly	Pass	
5	Test other admin actions	Using accept / reject user application function	The WebUI shall display new application details correctly and store user details in user table if approved	The WebUI displayed new application details correctly and stored user details in user table after it is approved	Pass	-
		Using disable / enable user account function	The WebUI shall display user account info correctly and disable / enable the selected user account	The WebUI displayed user info correctly and correctly disabled / enabled the selected user account	Pass	-
		Using reset user password function	The WebUI shall display user account info correctly and reset user password to default	The WebUI displayed user account info correctly and correctly reset user password to default	Pass	-
		Using delete existing server share function	The WebUI shall display the user share details and	The WebUI displayed the user share details	Pass	-

			delete the server share section from Samba CONF file	correctly and deleted the server share section accordingly		
6	Test WebUI media center function	Using WebUI media center to browse media files stored in media folder	The WebUI shall correctly display the media files (audio, video and pictures) in a table	The WebUI correctly displayed the media files in a table, complete with its extension name and search function for specific file	Pass	-
		Using WebUI media center to play audio file	The WebUI shall load the HTML5 audio player and play the selected audio file	The WebUI loaded the HTML5 audio player and correctly played the selected audio file	Pass	-
		Using WebUI media center to play video file	The WebUI shall load the DivX web player and play the selected video file	The WebUI loaded the DivX web player and correctly played the selected video file	Pass	-
		Using WebUI media center to view image file	The WebUI shall load the blueimp image gallery and display the selected image file	The WebUI loaded the blueimp image gallery and correctly displayed the selected image file	Pass	-

Table 8.11: The interoperability testing results for final system.

Test ID	Test Case	Test Steps	Expected Result	Actual Result	Status	Remarks
1	Test file server inter-operability	Testing the file server function to broadcast its	The shared folders shall be visible and	The shared folders are visible and	Pass	-

		shared folders to other platforms (Windows and Android)	accessible to laptop PC and Android smartphone	accessible to laptop PC and Android smartphone, based on their access permissions		
2	Test WebUI inter-operability	Accessing the WebUI on different browsers and platform (Firefox and Chrome in Windows and Android)	The WebUI shall be accessible and displayed correctly on both Firefox and Chrome browsers, both for Windows and Android platform	The WebUI are accessible and displayed correctly on both Firefox and Chrome browsers, both for Windows and Android platform	Pass	WebUI display is responsive and supports normal display for small display screen

Table 8.12: Reliability testing results for final system.

Test ID	Test Case	Test Steps	Expected Result	Actual Result	Status	Remarks
1	Test reliability of system running Kodi program	Using Kodi program to play a 900MB movie file continuously for 4 hours	The Kodi program shall be able to keep playing the video file without crashing / error	The Kodi program managed to play the video file without any error or crashing	Pass	-
2	Test reliability of WebUI media center to play media files	Using WebUI video player function to play a 900MB movie file continuously for 4 hours	The WebUI video player shall be able to keep playing the video file without crashing / error	The WebUI video player managed to play the video file without any error or crashing	Pass	The reliability of WebUI also depends on web browser reliability
3	Test reliability of system running Samba file server	Letting the system running Samba file server function be switched on for 24 hours	The system shall be able to run the file server function continuously for 24 hours without crashing / error	The system managed to run normally and file server function and shares were still accessible normally	Pass	The reliability of the file server also depends on router reliability

Table 8.13: Performance testing results for file transfer speed.

Performance Test Setup / Condition	Time to copy file to Desktop (t/s)				File transfer speed (TS/Mbps)
	Run				
	1	2	3	Average	
XHDD - Powered USB Hub - Raspberry Pi (using File Manager)	176.2	184.5	176.9	179.2	**5.58**
XHDD - Powered USB Hub - Raspberry Pi (using LXTerminal)	181.5	177.4	182.9	180.6	**5.54**
XHDD - Powered USB Hub - Raspberry Pi (using default command-line, no GUI)	173.6	174.5	172.7	173.6	**5.76**
XHDD - Powered USB Hub - Raspberry Pi - Router - Laptop (the file as server share)	230.1	227.1	226.5	227.9	**4.39**
XHDD - Laptop USB 2.0 slot	34.7	34	35.5	34.7	**28.79**
XHDD - Powered USB Hub - Laptop USB 2.0 slot	35.2	34.9	35.3	35.1	**28.46**
XHDD - Laptop USB 3.0 slot	18.1	17.4	18.3	17.9	**55.76**
XHDD - Powered USB Hub - Laptop USB 3.0 slot	32.2	31.3	32.4	32.0	**31.28**

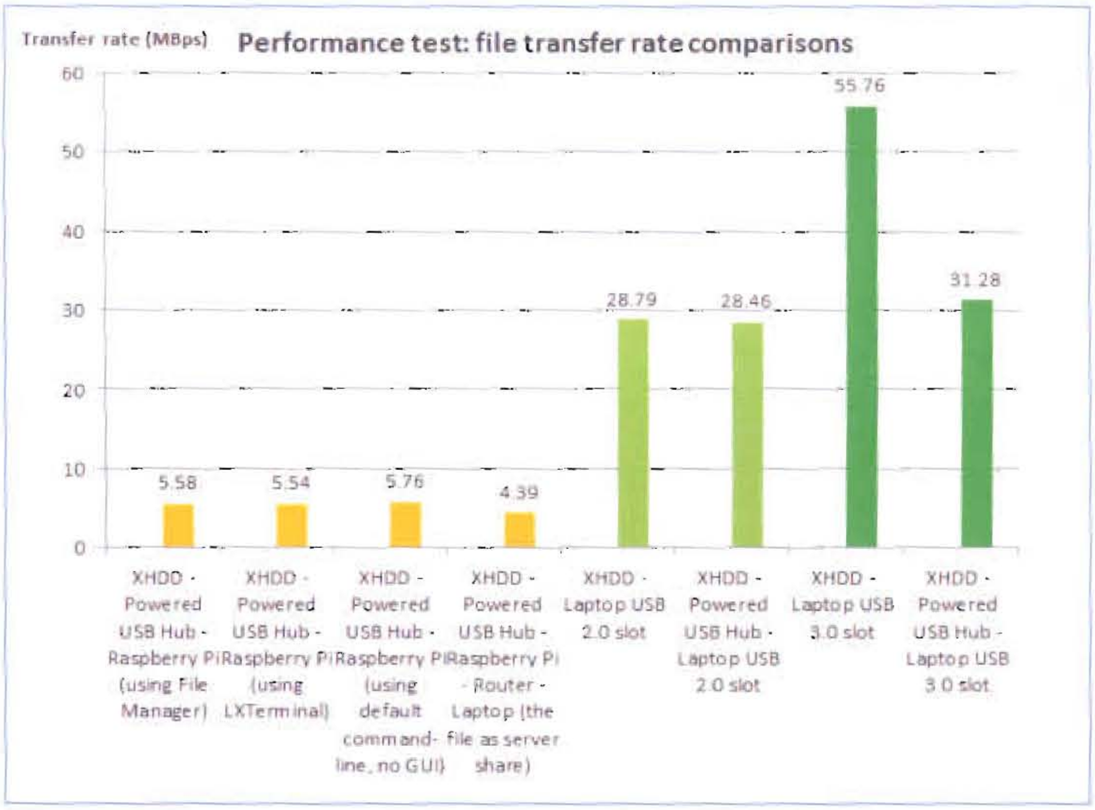


Figure 8.37: The graph for the performance testing on file transfer speed.

8.7 Appendix G – Acceptance Testing Results

Table 8.14: The user acceptance testing results for final system.

Test ID	Test Case	Test Steps	Expected Result	Actual Result	Status	Remarks
1	Test user acceptance of system functionality	Testing user acceptance on file server function	Depends on user expectation	User is fully satisfied with the file server function	Pass	-
		Testing user acceptance on the primary media center function (using Kodi program)	Depends on user expectation	User is fully satisfied with the Kodi program as the primary media center program	Pass	-
		Testing user acceptance on the WebUI file server configuration functions	Depends on user expectation	User is fully satisfied with the WebUI file server configuration functions as they are working correctly	Pass	Custom parser function can still be developed to parse *smb.conf*
		Testing user acceptance on the WebUI media center functions	Depends on user expectation	User is fully satisfied with the WebUI media center functions as they are capable to play media files on web browser	Pass	Even though format support is limited, it may improve in the future
		Testing user acceptance on the WebUI management functions	Depends on user expectation	User is fully satisfied with WebUI management functions as they are working correctly	Pass	-
2	Test user acceptance of WebUI design	Testing user acceptance on the WebUI design	Depends on user expectation	User is fully satisfied with the responsive WebUI design	Pass	-