The 17th Winona Computer Science Undergraduate Research Symposium

April 26, 2017

9:30am to 12:00pm

Gildemeister Hall (GI) 325

Winona State University
Winona, MN

Sponsored by the Department of Computer Science at
Winona State University

WINONA
STATE UNIVERSITY
Computer Science Department
http://cs.winona.edu

Table of Contents

Performance Analysis of Raspbian and Ubuntu Mate on a Raspberry Pi

Jakiul Alam

Department of Computer Science

Winona State University

Winona, Minnesota

JaAlam13@winona.edu

***Abstract*— Raspberry Pi has been in the center of a lot of "Do it Yourself" projects since it hit the market. The computer is very compact, inexpensive and powerful. However, the price point dictates that some tradeoffs be made. This is where the operating systems come into play. They are responsible for using the resources available on the device as efficiently as possible. In this research project, I have used UnixBench to compare Ubuntu Mate and Raspbian. I have run tests like Dhrystone 2, Whetstone, Process Creation which test aspects of OS that are general indicators of how gracefully they handle memory management and process management. Ubuntu Mate edged it out over the Raspbian.**

Keywords—raspberry pi; benchmark; raspbian; ubuntu mate;

I. INTRODUCTION

Raspberry Pi is a credit card sized computer that was developed by Eben Upton and a group of computer enthusiasts to inspire and instill in children the fundamentals of computing. From powering a high-tech mirror that tells you weather to being the brains behind a face plotter, it has been used to do a lot of credible projects. A huge community has sprung up and a lot of tech giants have started taking heed into this project. Raspberry Pi's are extremely inexpensive, they range from $27 to $39. As expected the trade-off for keeping prices low is limiting the performance of these devices. This is where operating systems play a big role to make these devices more efficient in terms of speed, time and power.

Based on where it is being used and viewpoint, operating systems can take many shapes and forms. "An operating system is a program that manages a computer's hardware" [8], satisfies our view of an operating system. It is a key element in a computing environment that makes sure resources are being allocated correctly, efficiently and effectively. There are lots of resources that the OS has to keep track of. In this paper, we looked at how they manage the distribution of CPU time among running processes and how they allocate memory for these processes. Each operating system approaches the problems differently. They have different sets of algorithms and schemes for carrying out these tasks.

Process management entails a variety of tasks. They range from allocating CPU time, inter-process communication, protecting resources allocated to a process and synchronization of processes. On the other hand, memory management is concerned with keeping as many processes running as possible.

Given the time it has been around, you can choose from quite a sizeable number of operating systems to put on a Raspberry Pi. Raspbian is the officially recommended operating system for the Raspberry Pi. It is based on Debian distribution of Linux. It is a community driven, open source, free software and is still under active development according to their website [3]. Also, it has over 35000 pre-compiled software packages that can be installed on the Raspberry Pi since it carries the popular Debian desktop environment. It is currently the most used operating system on a Pi.

Ubuntu Mate developed by Martin Wimpress and Rohith Madhavan specifically for the Raspberry Pi computers. They have optimized the operating system so it runs on the device as efficiently as possible. Ubuntu is one of, if not the, largest deployed Linux based desktop operating systems in the world. Linux is at the heart of Ubuntu and makes it possible to create secure, powerful and versatile operating systems, such as Ubuntu and Android. Android is now in the hands of billions of people around the world and it's also powered by Linux. The MATE Desktop is one such implementation of a desktop environment and includes a file manager which can connect you to your local and networked files, a text editor, calculator, archive manager, image viewer, document viewer, system monitor and terminal. All of which are highly customizable and managed via a control center [2].

To compare these two operating systems, we used benchmarking software. Some of these software, come with predefined parameters or sometimes you can customize them. For our purpose, we are using Byte Unix Benchmark tool that have predefined set of tests using predefined parameters. In essence, Benchmarking is running the same set of tests on different products you are testing and comparing the score produced from those tests.

Proceedings of the 17thWinona Computer Science Undergraduate Research Seminar, April 26, 2017, Winona, MN, US.

Hypothesis: Ubuntu manages processes and memory more efficiently than Raspbian on the Raspberry Pi.

II. Methodology

For this project, we have two different OS on two identical memory cards and we booted onto them separately. We used UnixBench5.1.3 which is an open source tool and is available on Google code for anybody to download. It is based off of Byte Unix benchmark algorithm and requires you to download certain Perl libraries to run. It can measure the performance of bash scripts, CPUs in multithreading and single threading. It can also measure the performance for parallel tasks. Also, specific disk IO for small and large files are performed. These tests should be sufficient to come to a conclusion. A full run of the benchmark tool performs Dhrystone2, Double-Precision Whetstone, execl Throughput, file copy using variable buffer sizes and block sizes, pipe throughput and context switching, process creation, shell script and system call overhead. A lot of these tests overlap in aspects they test. For simplicity, will be considering Dhrystone, Whetstone, Shell script runs and process creation as markers for process management and context switching, file copying, execl throughput as markers for memory management.

Dhrystone is a general-performance benchmark test originally developed by Reinhold Weicker in 1984 updated to Dhrystone 2 in 1988 [6]. This benchmark is used to measure and compare the performance of different computers or, in this case, the efficiency of the code generated for the same computer by different compilers. The test reports general performance in loops per second. Like most benchmark test, Dhrystone consists of standard code and concentrates on string handling. It uses no floating-point operations. It is heavily influenced by hardware and software design, compiler and linker options, code optimizing, cache memory, wait states, and integer data types.

Whetstone is a benchmark test which attempts to measure the speed and efficiency at which a computer performs floating-point operations. The result of the test is given in units called kilo-whetstones-per-second or KWIPS. The Whetstone is a synthetic benchmark designed to measure the behavior of scientific programs. It contains several modules that are meant to represent a mix of operations typically performed in scientific applications. A wide variety of C functions including sin, cos, sqrt, exp, and log are used as well as integer and floating-point math operations, array accesses, conditional branches, and procedure calls. The primary aim of this benchmark is to measure the performance of both integer and floating-point arithmetic [10]. Both of these tests are a good indicator of how well an operating system handles process management. Also, there is an additional test which shows how well the operating system handles the process spawning in UnixBench, it is called Process Creation. Other than that, it tests scripts by running them in parallel.

Pipe based context switching test measures the number of times two processes can exchange an increasing integer through a pipe. The pipe-based context switching test is more like a real-world application. The test program spawns a child process with which it carries on a bi-directional pipe conversation.

File copying measures the rate at which data can be transferred from one file to another, using various buffer sizes. The file read, write and copy tests capture the number of characters that can be written, read and copied in a specified time. This indicates the speed of read and writes which depend extensively on the usage of RAM.

Process creation test measure the number of times a process can fork and reap a child that immediately exits. Process creation refers to actually creating process control blocks and memory allocations for new processes, so this applies directly to memory bandwidth. Typically, this benchmark would be used to compare various implementations of operating system process creation calls.

III. Test Environment and Setup

We used a Raspberry Pi 3 model B with 2 identical 32gb class 10 memory card, one with Ubuntu Mate and one with Raspbian. Both were the latest releases at the time of testing. Table 1 shows the operating system information and table 2 shows the configuration of a Raspberry Pi 3 model B.

Table 1. Operating System Specifications

Operating System	Ubuntu Mate	Raspbian
Distribution	Ubuntu	Debian
Kernel Version	4.4.38-v7+	4.4.34-v7+
OS Version (codename)	16.04.2LTS (xenial)	8.0 (jessie)
Disk Space Occupied	3.4 GB	3.8GB

Table 2. Raspberry Pi 3 Model B Configuration

Introduction Date	2/29/2016
CPU	Quad Cortex A53 1.2GHz
Instruction Set	ARMv8-A
GPU	400MHz
RAM	1GB SDRAM
Ethernet	10/100
Wireless	802.11n/Bluetooth 4.0
Price	$35

The Raspberry Pi was connected to an Element monitor via HDMI and the resolution on both OS was set to 1360x768 pixels. Both were connected to the internet via Ethernet port and wireless was turned off in both cases.

After booting to the operating system, we installed all the pending updates and rebooted the system. We used the terminal to install the specific libraries required to run UnixBench5.1.3. Then we ran the benchmark test using the terminal. Upon completion of the test, the results were generated immediately and displayed onto the terminal.

Figure 1. Raspberry Pi 3 Model B

IV. Results

The UnixBench benchmark runs two sets of tests, one of which uses a single core and the other uses all available cores for the processors. The following tables show a summary of the results from both sets of tables and compares the results from both operating systems.

Table 3. Results of running single copy of the test

System Benchmarks Index Values	**Ubuntu Mate**	**Raspbian**
Dhrystone 2 using register variables	5107464.5 lps	4344901.7 lps
Double-Precision Whetstone	1019.9 MWIPS	736.4 MWIPS
Execl Throughput	741.6 lps	773.1 lps
File Copy 1024 bufsize 2000 maxblocks	144790.3 KBps	143881.1 KBps
File Copy 256 bufsize 500 maxblocks	43066.0 KBps	43539.3 KBps
File Copy 4096 bufsize 8000 maxblocks	347839.0 KBps	335002.4 KBps
Pipe Throughput	276246.1 lps	310841.3 lps
Pipe-based Context Switching	54936.7 lps	54462.4 lps
Process Creation	2347.3 lps	2438.4 lps
Shell Scripts (1 concurrent)	2126.6 lpm	2116.7 lpm
Shell Scripts (8 concurrent)	516.5 lpm	489.8 lpm
System Call Overhead	700690.9 lps	693554.5 lps
System Benchmarks Index Score	313.9	303.6

Table 4. Running four parallel copies of the test

System Benchmarks Index Values	Ubuntu Mate	Raspbian
Dhrystone 2 using register variables	15197180.0 lps	12139250.8 lps
Double-Precision Whetstone	3257.4 MWIPS	2187.1 MWIPS
Execl Throughput	1825.3 lps	1631.4 lps
File Copy 1024 bufsize 2000 maxblocks	162635.9 KBps	158476.0 KBps
File Copy 256 bufsize 500 maxblocks	44114.0 KBps	44042.5 KBps
File Copy 4096 bufsize 8000 maxblocks	406946.6 KBps	412090.6 KBps
Pipe Throughput	775798.7 lps	858033.4 lps
Pipe-based Context Switching	124035.8 lps	120903.3 lps
Process Creation	3658.3 lps	3629.3 lps
Shell Scripts (1 concurrent)	3196.0 lpm	3096.9 lpm
Shell Scripts (8 concurrent)	412.3 lpm	401.9 lpm
System Call Overhead	1975942.6 lps	1945043.9 lps
System Benchmarks Index Score	560.6	526.4

V. Analysis

A. *Process Management*

Comparing the performances for the Dhrystone 2 and Whetstone first. These tests measure how fast these OS handle big calculation which gives us a clear indication of how well it manages processes. During the Dhrystone 2 test using the single core Ubuntu Mate performed 15% faster at 5107464.5 loops per second compared to 4344901.7 loops per second on Raspbian. During floating point calculations in the Whetstone test the difference was more noticeable. Ubuntu Mate performed 28% faster than Raspbian. The pipe throughput test measures the rate at which inter-process communication is done. In this case, Raspbian performed 12% faster than Ubuntu Mate. This would mean process synchronization on Raspbian is fairly smoother than it is on Ubuntu Mate. The process creation test measure the number of times a process can fork and reap a child that immediately exits. During our tests, Raspbian performed 4% faster than Ubuntu Mate. On the shell script test Ubuntu Mate performed 10% faster both times. The trends are preserved during the test with all cores in use. Overall Ubuntu Mate performed slightly better than Raspbian.

B. *Memory Management*

The execl throughput was the first test for Memory Management, here bunch of current processes are replaced by new processes and memory is allocated to them as part of the process control block. Raspbian performed 5% faster than Ubuntu Mate. However, with more of the test being done at the same time, Ubuntu Mate performed 11% faster than Raspbian. Ubuntu Mate reached speeds of up to 44114.0 KBps while copying files compared to Raspbian's 44042.5 KBps. Finally, the Ubuntu Mate handled context switching 3% faster than Raspbian.

VI. Conclusion

Overall the in both sets of tests Ubuntu Mate got a higher score than Raspbian. In the single core test Ubuntu Mate scored 313.9 versus Raspbian's 303.6. The performance gap was a little wider when put under more load, Ubuntu Mate scored 560.6 and Raspbian scored 526.4. Although the scores are not that different, Ubuntu mate scored marginally better than Raspbian. This leads us to accept our hypothesis that Ubuntu handles process and memory management better than Raspbian.

References

[1] "About Debian" www.debian.org https://www.debian.org/intro/about Accessed: February 2017

[2] "About" ubuntu-mate.org https://ubuntu-mate.org/about/ Accessed: February 2017

[3] "About Raspbian" www.raspbian.org https://www.raspbian.org/RaspbianAbout Accessed: February 2017

[4] "About Us" www.raspberrypi.org https://www.raspberrypi.org/about/ Accessed: February 2017

[5] "Byte-Unix Benchmark" code.google.com https://code.google.com/p/byte-unixbench Accessed: February 2017

[6] "Dhrystone" en.wikipedia.org https://en.wikipedia.org/wiki/Dhrystone Accessed: February 2017.

[7] N. Hatt, A. Sivitz and B.A Kuperman, "Benchmarking Operating Systems", Oberlin College Research Publication 2007, https://www.cs.oberlin.edu/~kuperman/research/papers/audlib2007mcurcsm.pdf Accessed: February 2017

[8] Silberschatz, A., Galvin, P. B., & Gagne, G. (2005). *Operating system concepts* (9th ed.). Hoboken, NJ: J. Wiley & Sons.

[9] "The 14th Winona Computer Science Undergraduate Research Symposium" cs.winona.edu http://cs.winona.edu/conference.php Accessed: February 2017

[10] "Whetstone (benchmark)" en.wikipedia.org https://en.wikipedia.org/wiki/Whetstone_(benchmark) Accessed: February 2017.

iOS Application Development with Firebase as a Mobile Platform

Justin Bergeron
Computer Science
Winona State University
Winona, USA
jbergeron12@winona.edu

***Abstract*— Firebase is a mobile and web application platform with tools and features created to help developers build high quality apps. It includes free features that developers can mix-and-match to fit their specific needs. BetterYou is a Health & Fitness iOS mobile application that has food recipes and a list of all users and their posts. This study investigated the effectiveness of Firebases Spark plan with iOS application development and deployment by conducting a usability test and a survey. From the results, this study found out that not only is Firebase an effective development platform but it also scored high on user satisfaction reviews.**

Keywords—iOS, Applicaton Development, Firebase

I. Introduction

Firebase is a mobile and web application platform with tools and features created to help developers build high quality apps. It includes free features that developers can mix-and-match to fit their specific needs [1]. A mobile application development platform (MADP) is software businesses can use to quickly develop top quality apps. A business has the choice of building their own MADP or purchase one from the many third-party products available. The third-parties typically offer features like Backend as a Service (BaaS) and administration tools for application programming interfaces (APIs). Behind every software application is a comprehensive series of backend services intended to support the front end you see and use every day [1]. The workload involved in developing/creating this backend technology is not a simple task. Many organizations are choosing to save time and money by using a BaaS.

II. BetterYou

A. BetterYou the Software

BetterYou is a Health & Fitness application that has food recipies and a list of all users and their posts. Once a user is logged in there are two main screens, a Meals tab and a Community tab. On the Meals tab, there are five sections listed: Recipe of the Day, Breakfast, Lunch, Dinner and Meal Prepping. Within those sections are recipes which once selected bring you to their specific screen listing the ingredients and directions to make. On the Community tab, all the users in the app are listed with their Facebook Profile Picture and Display name. Tapping a user's name brings you to their profile. A user's profile displays all the recipes that they have posted to BetterYou. When visiting your profile, a little circle with a plus appears in the top right corner. Tapping it brings you to the recipe uploading page. Users can type in the title of the recipe and then list the ingredients and directions. Once submitted the recipe is added to the user's profile page which can then be viewed by all users.

B. Firebase with BetterYou

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. This study investigated the effectiveness of the Firebase Spark plan with iOS application development and deployment. Spark is Firebase's free plan which offers great and easy to implement features. User authentication, database, and analytics are some of the features utilized by the BetterYou app. Facebook was used for BetterYou's user authentication.

The first time a user opens the app, they are prompted to "Login with Facebook." Once the user is verified with their Facebook credentials, Firebase does the rest. Their display name, profile picture, and user ID are all added to the users branch in the database. The database also stores the recipes in the app. This includes the pre-loaded and the user posted recipes. When a user posts a recipe, it is added underneath their name in the users branch as well. The structure style of the database is a JSON file that can be queried and edited in real-time. JavaScript Object Notion (JSON) is a minimal, readable format for structuring data. It is used primarily to share data between a server and web application, as an alternative to XML. Each time a user loads a new screen the app queries the database and loads in the specific information being requested. When a user posts a recipe, the JSON file updates within seconds and the changes can then be viewed by other users.

Analytics gives useful information about the user's interactions with the app. Active Users, User Engagement, Retention Cohorts, and Devices are the four sections being used for BetterYou. Active Users displays a visual representation of monthly, weekly, and daily user totals. User Engagement displays daily engagement, daily engagement per user, sessions per user, and average session duration. Retention

Proceedings of the 17th Winona Computer Science Undergraduate Research Seminar, April 27, 2017, Winona, MN, US.

Cohorts displays user retention rate over a 5-week span and gauges if users are coming back after the first week. Devices displays the phone model and OS version of users., the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

III. HYPOTHESIS

Is Firebase an effective mobile platform for iOS development?

IV. METHODS

A. Usability Study

Usability Studies have been a common practice since the 1990s to assess the effectiveness of user interfaces [2][3] and to collect feedback from users to improve existing user interfaces. This research performed a usability study to test the effectiveness of Firebase with iOS mobile application development. The research was executed with WSU students ranging from freshman to seniors using a survey with goals they had to perform within the app.

B. Demographics of Participants

A total of 50 Winona State University college students participated in the testing. They ranged from freshman to graduate students. Each user had to have an iPhone and a Facebook account. There were 22 males and 28 females.

C. User Tasks and Questions

The equations are an exception to the prescribed specifications of this template. This study conducted a usability test and a survey. The usability test consisted of 5 tasks that each user was to try and complete. If any of the tasks could not be accomplished the user was asked to explain why. The tasks are shown in Table 1.

The survey gathered information about the users experience while conducting the usability test. Data collected consisted of:

(1) Successful or Unsuccessful Facebook login

(2) Successful or Unsuccessful Database query

(3) Successful or Unsuccessful Database entry

(4) Load times.

Facebook login is whether the user had a successful login. Database query consisted of users being able to view the correct recipes descriptions dependent on their selection. Database entry consisted of users being able to upload a recipe of their own into the Database for others to view. Load times consisted of the average time it took for information and images to load onto the users' screen. Results of the data collected are shown in figure 3 & 4.

D. Usability Goals

Table1: Tasks asked to perform

1	Login with Facebook.
2	Access all Categories under the Meals tab (ROTD, Breakfast, Lunch, Dinner, Meal Prepping) View each Categories meals list and access their specific details.
3	Find your Facebook Display Name and Profile Picture under the Community tab.
4	Once clicked onto your name, attempt to post a recipe.
5	View other user's profiles.

E. BetterYou Life Cycle

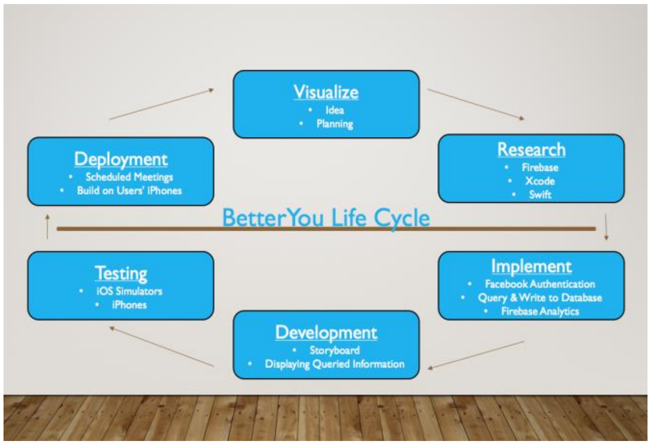


Figure 2: Life Cycle

- Visualize: The idea and planning began during the beginning weeks of January 2017. The idea stemmed from the want of an application to share meals cooked with friends.
- Research: Knowing we wanted to share photos within an app, we were going to need a good cloud storage solution. Firebase was selected because of its great guides and reviews. For developing on iOS, Firebase supports Xcode as an IDE and Swift as a language. Using these three tools, we then determined what features we wanted in the app.
- Implement: User authentication, Database/Storage, and Analytics were selected to be implemented into the app. Reading through Firebase's helpful guides, we could create an implementation plan for the selected features.
- Development: The first step of development was creating a Storyboard in Xcode. This gives the

layout and flow of the app. From there, we used the implementation plan to develop the correct functions and methods to connect the app with Firebase.

- Testing: Xcode has built in iOS simulators that cover the wide range of Apple Devices. Testing on the simulators first allowed us to discover and then remove bugs/errors. We then moved to physical iPhones for further testing.
- Deployment: Meetings were scheduled with testers so the app could be built to their iPhone. The usability test and survey were then given to the testers and collected once finished.Word Formatting toolbar.

F. Software Development

We began with a blank Xcode project that would later be integrated with Firebase. Xcode is an integrated development environment (IDE) for macOS containing a plethora of software development tools created by Apple for developing software for macOS, iOS, watchOS and tvOS [4]. Once you create your free account with Firebase you are brought to your own personal dashboard. Here is where you will be able to create multiple apps and keep track of them with ease. When creating a project in Firebase with iOS you must already have an Xcode project created. This will automatically create a unique iOS bundle ID. This ID is very important; it will link your Xcode project to Firebase. Once those two are linked up, a file called GoogleService-Info.plist will be downloaded to your computer [1]. Before adding the file to your Xcode project you must install the Firebase software development kit (SDK). This is a three-line command that you run in Terminal. Once the Firebase Pods are installed, adding the GoogleService-Info.plist file is as easy as dragging and dropping it into your file list in Xcode. After completing those steps, you are directed to your customized dashboard within Firebase. It's here where you can start to add features to your app with the code snippets provided.

G. Software Development

BetterYou utilizes three main Features from Firebase.

1.) Authenticating users using Facebook login on iOS.

2.) Read and Write Data on iOS.

3.) Analytics for iOS.

1.) Authenticating users using Facebook login on iOS takes 5 steps.

- Integrate Facebook Login into your app by following the developer's documentation. When you initialize the FBSDKLoginButton object, set a delegate to receive login and logout events [1].
- Import the Firebase module in your UIApplicationDelegate subclass [1].
- Configure FIRApp in your application's application:didFinishLaunchingWithOptions: method [1].
- After a user successfully signs in, get an access token for the signed-in user and exchange it for a Firebase credential [1].
- Finally, authenticate with Firebase using the newly obtained Firebase credential [1].

Once completed Facebook Login is a great feature that will give your app a professional first impression.

2.) Read and Write Data on iOS is a series of Database calls that either query into an array or edit the current JSON file. For basic write operations, the setValue method is used to save data to a specified reference, replacing any existing data at that path. This method can be used to pass the four compatible JSON types:

- NSString
- NSNumber
- NSDictionary
- NSArray

Once it is decided what type your data will be you can use setValue as follows (Swift):

self.ref.child("users").child(user.uid).setValue(["username" : username])

Using setValue in this way overwrites data at the specified location, including any child nodes. However, you can still update a child without rewriting the entire object [1]. If you want to allow users to update their profiles you could update the username as follows (Swift):

self.ref.child("users/(user.uid)/username").setValue(username)

Using these write options, users can add data that can then be accessed by anyone.

Reading data once in Firebase is one of the recommended examples [1]. This type of Database query takes a snapshot of the "child" or branch of a specific section in the JSON file and puts all the data into an array of your style. Once the data is queried into the array, you can use that information to populate tables or display specific information within your app. You can also set up a listener. For instance, having a listener with a news feed display is a better solution to database querying. A listener can wait for any action to occur and then carry out a task. The news feed example would be every time a user creates a new post the listener hears that and then automatically updates the news feed for all users to see.

3.) Analytics for iOS is the simplest of the three to add to your app. With just two lines of code, your app can be configured and ready to run Analytics.

- import Firebase
- FIRApp.configure()

Once your app is configured to Firebase Analytics, you have instant access to useful information about the user's interactions with the app. Active users, User engagement, Retention cohorts, and Devices are the four sections being used for BetterYou. Active users display a visual of your monthly, weekly, and daily user totals. User engagement displays Daily engagement, Daily engagement per user, Sessions per user, Avg. session duration. Retention cohorts displays user retention rate of a 5-week span meaning after the first week use are your users coming back. Devices displays the phone model and OS version that your users have.

V. Results and Analysis

Users were asked questions, as shown in Table 1, to try and perform a series of goals. The following charts are a visual representation of the results.

Figure 3, displays number of successful and unsuccessful goals. Results found that users had a success rate of 93% when interacting with Firebases Real Time Database. The failures that did occur are believed to be internal coding errors with the app, not Firebase.

Figure 4, displays user's response about screen load times. The load times that did take longer than 5 seconds were typically when images were being loaded onto the screen. The text would load but the images would sometimes take a little longer.

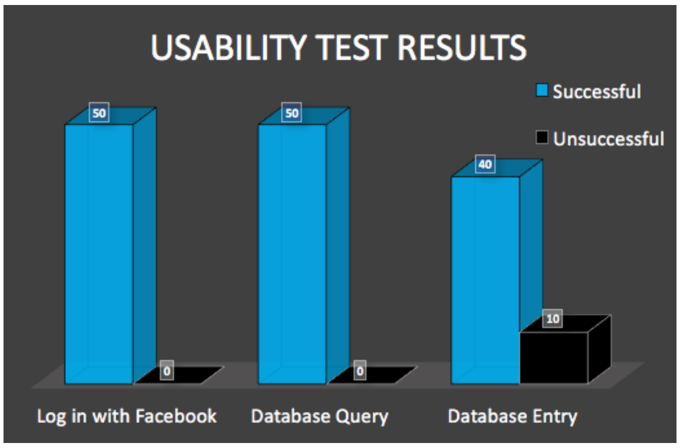


Figure 3: Usability Test Results Per User

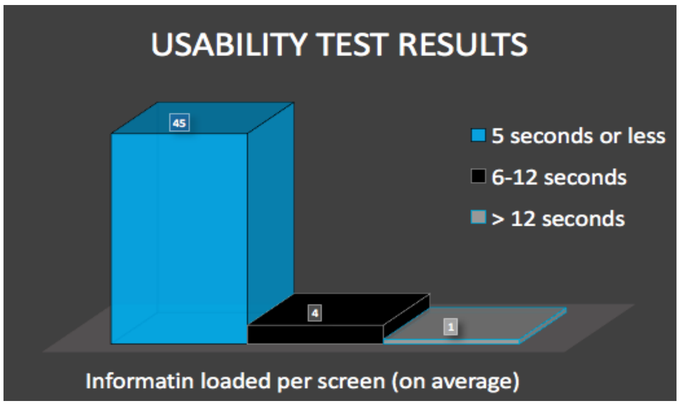


Figure 4: Information Load Times Per User

VI. Results and Analysis

The primary goal of this research is to gather data that supports or denies the hypothesis. The results of the usability test and survey speak for themselves. Throughout the Software Life Cycle of the project, Firebase proves that it is indeed an effective MADP for iOS Development. From the beginning stages to deployment, their get started guides, Code Labs, API references, and Samples were extremely helpful and easy to follow. Moving forward, Firebase will continue to be BetterYou's backbone. This study accepts the hypothesis that Firebase is an effective mobile platform for iOS development.

References

[1] Firebase, 2017. https://firebase.google.com

[2] Usability, 2016. www.Usability.gov

[3] Nielsen, J., and Mack, R. L. (Eds.) (1994). Usability Inspection Methods, John Wiley & Sons, New York.

[4] Apple, 2017. https://developer.apple.com

[5] Zobel, Justin. Writing for computer science. London: Springer, 2014. Print..

[6] "Automated Usability Testing for Mobile Applications." *Proceedings of the 10th International Conference on Web Information Systems and Technologies.*

[7] Firebase, 2017. https://console.firebase.google.com

[8] Apple, 2017. https://developer.apple.com/iOS/human-interface-guidelines

A Web-Based Approach to Virtual Guitar Amplification

Richard Marquez
Computer Science Department
Winona State University
Winona, MN 55987
RMarquez14@winona.edu

***Abstract*— Guitar amplifiers take the electrical signal from an instrument and strengthen it to be pushed out through a speaker system. They are often prohibitively expensive or overly complex. A new system has been developed to make virtual guitar amplification available through the web. This allows users to be able to access virtual guitar amplification more quickly and easily than through a traditional digital audio workstation.**

Keywords—audio; signal processing; web; amplifier; guitar

I. Introduction

A. Background Information

An electric guitar uses magnetic "pickups" to convert the mechanical vibrations of its ferrous strings into an electrical signal. This signal is too weak to drive a speaker by itself so it is first sent through an amplifier. A guitar amplifier amplifies the signal so that it can produce sound through one or more speakers. An amplifier also provides a set of basic sliding controls including: volume—the loudness of the output; gain—the sensitivity of the input; tone—the amount of treble present; and reverb—simulation of a reverberant environment.

Physical guitar amplifiers are often expensive and physically unwieldy which can be prohibitive traits for hobbyist players. Increases in computer performance have led to the proliferation of software-based virtual guitar amplification. However, virtual guitar amplifiers can also be expensive and overly complex, often requiring special training as a part of a full-scale digital audio workstation (DAW).

The chart in Fig. 1 shows the popularity of various digital audio workstations as voted on by readers of Ask.Audio, a resource for educational materials by digital music makers. GarageBand and its associated paid version, Logic Pro, together make up some of the most popular DAWs [1] due to their relative simplicity and availability on macOS. The relatively low bar to entry of these DAWs is still prohibitive to many players who do not have access to an Apple computer, have no interest in recording software, or do not have the time to download, install, and learn how to operate the software.

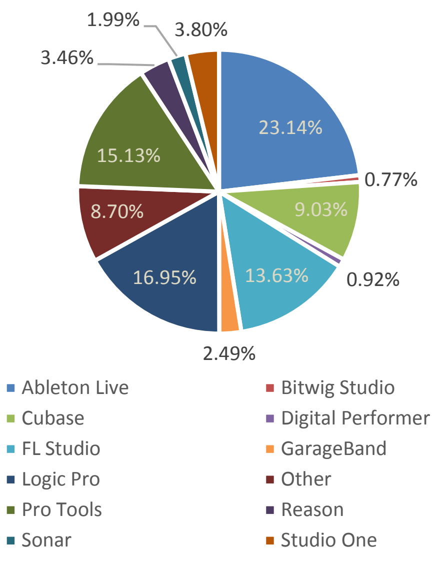


Figure. 1. Popularity of digital audio workstations

B. Research Purpose

When I began learning to play guitar I was in the situation of needing an amplifier. I did not want to make the financial commitment of purchasing a physical amp. The alternative was to use a virtual amp, with GarageBand seeming to be the best available choice given my circumstances: I had a Mac computer and was searching for a free option. I was in no need of the slew of professional recording features, as shown in Fig. 2, that dominate the application. As such, the learning curve was unnecessarily steep and nearly destroyed the effort altogether.

That personal story is a demonstration of the need for a simpler and accessible virtual guitar amplifier. The goal of this project is to build such a software that makes straightforward guitar amplification available in the browser space.

A web-based guitar amplifier makes virtual guitar amplification available through the web browser. Virtual guitar amplification will be able to be achieved more quickly and easily through a web application dedicated solely to guitar amplification, rather than a traditional full-featured desktop DAW.

Figure. 2. GarageBand example project

II. METHODOLOGY

A. Development Strategy

This is a software development project that used industry standard web technologies to develop a single-page web application that provides standard guitar amplifier functionality given an audio input. Functionality includes amplification of the input signal as well as controls for volume, gain, tone, and reverb. Fig. 3 shows the final user interface. It features adjustable circular controls similar to physical amplifiers for the preceding effects. It also includes a waveform display of the output signal.

HTML5 and CSS3 form the base of technology used for front-end development. The Bootstrap framework was used to provide page structure and responsive capabilities. The audio waveform was drawn using the standard HTML5 canvas element. The JavaScript language and associated web APIs were used to develop the back-end of the application. A combination of jQuery and standard JavaScript were used to connect the data fed from the back-end into the front-end, and vice versa. The standard Web Audio API was used to interface with audio input and output [2].

JetBrains WebStorm was used as the primary IDE for development with an educational license. Git was used for version control, and the source of the project is available publicly on GitHub. The web application was tested on the Safari, Chrome, and Firefox browsers on macOS and Windows systems to ensure cross-compatibility.

An iterative approach to development was taken. Amplification and output of the input signal form the core of the application. Once the core was complete the effects controls for volume, gain, tone, and reverb were added. The waveform display was developed after all of the preceding functionality and serves primarily as a UI element.

B. Software Internals

Connecting the guitar to the computer with the 1/4 inch to USB cable makes it appear as a standard audio input device. The browser's audioContext object is used access the input and output streams of the chosen audio devices (the guitar and system speakers, respectively). Audio nodes are created and attached to the audioContext's graph to process the signal.

1) Tone

Tone indicates the amount of treble present in the signal. A biquad filter is used to achieve this effect. A highshelf biquad filter node is created and connected to the input audio device's stream (the beginning of the signal chain) to modify the range of frequencies that are attenuated [3]. Selecting a high tone

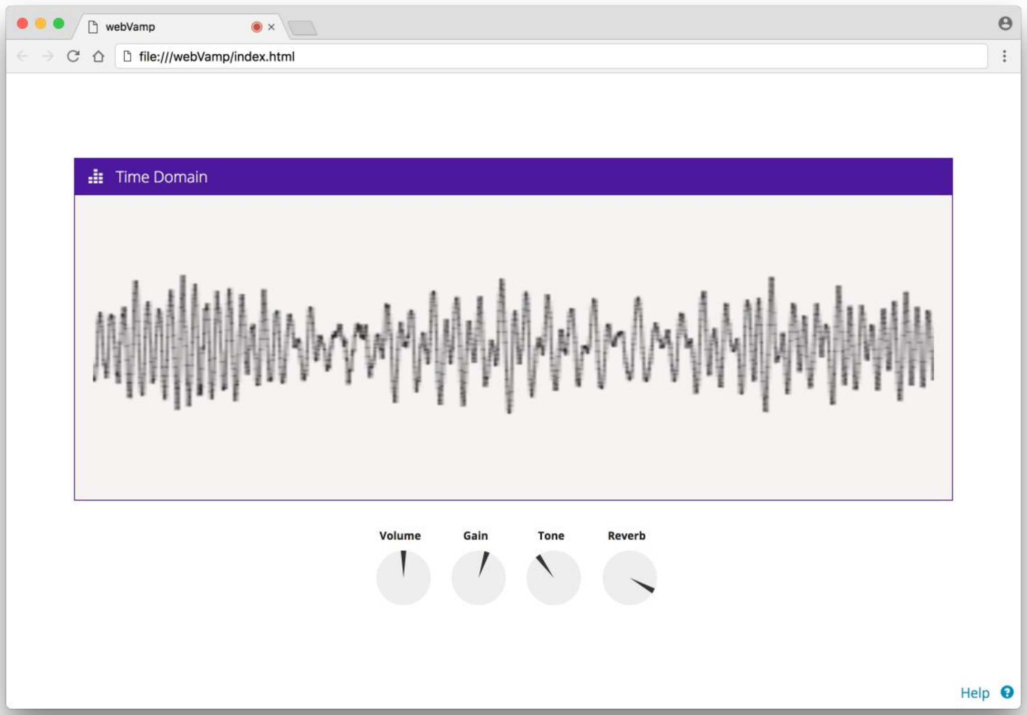


Figure. 3. Finished UI of web-based amp

value from the UI knob will raise the threshold of attenuated frequencies; i.e. the higher the tone, the more treble.

2) Reverb

The reverb effect controls simulation of a reverberant environment. This typically involves complex calculations or an intricate set of interconnecting audio nodes. To simplify this process, the soundbank-reverb library (developed by mmckegg on GitHub) was used. A reverb node from that library is created and connected to the tone node. The UI's reverb knob adjusts the time property of the reverb node as it is spun; i.e. the higher the reverb, the longer it lasts.

3) Amplification and Volume

The amplification and volume of the signal can be considered one and the same from a programmatic point of view. A gain node is created that takes the reverb node as its input. The reverb node is used instead of the direct input stream so that the volumes of the effects are also taken into account. The gain property of the gain node used to represent the volume of the entire signal chain. The UI's volume knob changes the value of the gain property as it is spun.

4) Gain

The gain control modifies the sensitivity of the input. To achieve this effect, another gain node was created that resides slightly askew to rest of the graph. It is connected to both the reverb node and the input stream. The UI's gain knob changes the value of the node's gain property as it is spun; i.e. the more gain, the more sensitive the input.

5) Waveform

The audio waveform is displayed in the time domain. The amplitude of the signal is represented on the y-axis and time on the x-axis. Because of the relatively small amount of signal processing taking place within the application there were no performance issues resulting in signal buffer lag. A script processor is attached to the input stream to redraw the graph whenever audio is processed. It is drawn on an HTML5 canvas element.

C. *Testing*

1) Test Process

A survey was executed to assess the hypothesis. The user was tasked with achieving a usable tone from either GarageBand or the new system given an electric guitar. The user may have had experience with the instrument, but may not have had experience with any digital audio workstation software.

The user was provided with an electric guitar, 1/4-inch audio to USB cable, and a MacBook Pro (Early 2015) with GarageBand 10.1.6 and the web amp system installed. The user was asked to obtain a usable tone from either GarageBand or the new system. The user was shown links on the desktop to the appropriate software. No further instruction was given throughout the survey.

It should be noted that the time taken for GarageBand to be downloaded and installed (an unnecessary step for the web amp) was not part of the test. GarageBand 10.1.6 is a 956 MB package which takes approximately 19 minutes to download on a 7 Mbit/s connection [4].

2) Usability Survey

The time it took to acquire a usable tone was recorded. The user was then asked to rate the ease of use of the software on a scale from 1 (being extremely difficult) to 5 (extremely easy). The user was also asked to rate how likely they were to recommend the software to a beginner guitar player on a scale from 1 (extremely unlikely) to 5 (extremely likely).

These metrics were initially recorded on paper by the user and were then transferred into a Microsoft Excel spreadsheet. As suggested by the Nielsen Norman Group for user experience research [5] there was a total of twenty users in an effort to achieve statistically significant results. Half of the pool tested GarageBand, and the other half tested the new system.

III. Results and Analysis

There were a total of twenty users used to gather information about the software usability. Ten of the users tested GarageBand, and ten users tested the new web-based system. No users had prior experience playing guitar or using any type of digital audio workstation.

A. *Time to Amplification*

The overall mean time it took to achieve amplification, as shown in Fig. 4, was 1.3 minutes for the web amp and 5.1 minutes for GarageBand. As noted earlier, the approximately 19 minutes it takes to download and install GarageBand (an unnecessary step for the web amp) was not part of the test. Even without that added time a t-test showed a statistically significant difference between the results with a p-value < 0.05.

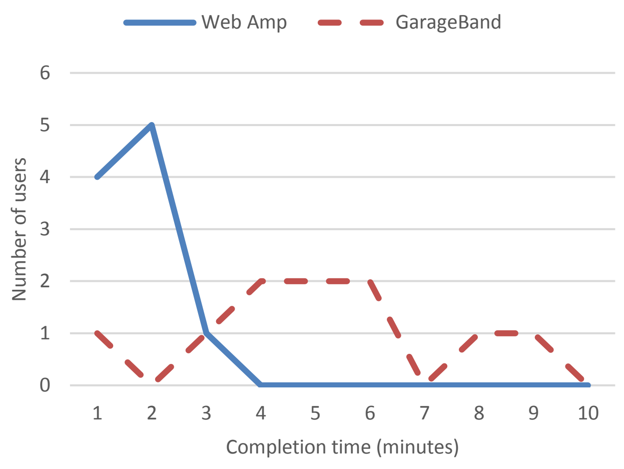


Figure 4. Time to completion results

B. *Ease of Use*

The overall mean rating for the ease of use of the software, as shown in Fig. 5, was a score of 4.6/5 for the web amp and 2.8/5 for GarageBand. A t-test showed a statistically significant difference between the results with a p-value < 0.05. to 5 (extremely likely).

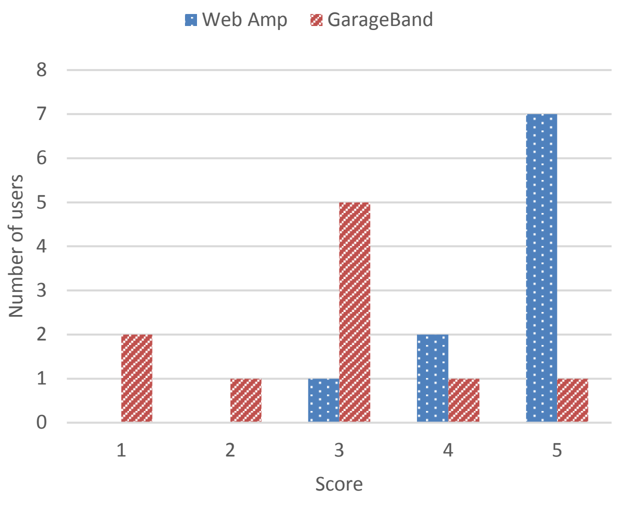


Figure 5. Ease of use results

C. *Likelihood of Recommendation*

As shown in Fig. 6, the overall mean rating for the likelihood the user would recommend the software to a beginner guitar player was highly skewed in favor of the web amp, with a score of 4.8/5, as opposed to GarageBand with a score of 2.8/5. A t-test showed a statistically significant difference between the results with a p-value < 0.05.

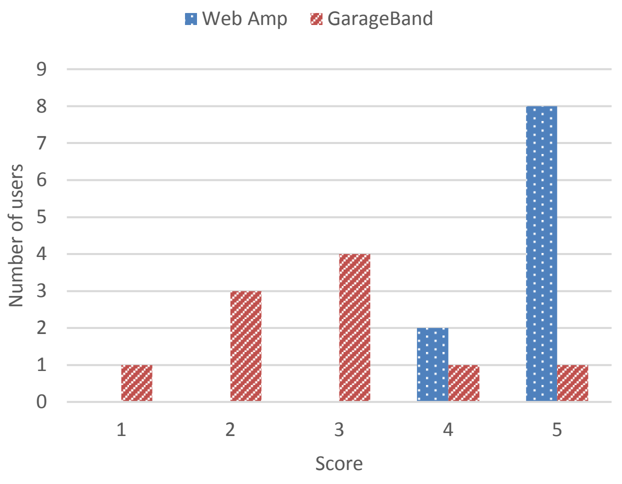


Figure 6. Likelihood of recommendation results

IV. Conclusion

The web amp software was developed as planned and used to compare usability with GarageBand as a traditional digital audio workstation. The statistically significant data acquired through the survey supports the initial hypothesis that a web-based amp with dedicated functionality can be used to achieve virtual guitar amplification more quickly and easily than through a traditional DAW.

The general accessibility of the functionality is also vastly increased by the web amp. GarageBand is available only on macOS systems and requires a large download. The web amp requires no download and is available on any system with a web browser that supports the Web Audio API (e.g. Chrome 53.0, Firefox 36).

V. Further Work

This project is open-source and available on GitHub (https://github.com/richard92m/webVamp). The next step is to have the application hosted online and made available for public usage.

The web amp should work on mobile devices given the appropriate hardware connections and a browser that supports the Web Audio API. This was not tested during the course of this research project.

This web-based virtual guitar amplifier stands as a nice proof of concept and exploration of the Web Audio API; however, there is a wide open space in the market for a full-fledged DAW in the browser. Soundtrap (http://soundtrap.com) is an example of the type of applications that will be coming in the near future. This open-source project may be expanded upon.

Acknowledgment

I would like to thank Dr. Narayan Debnath, Dr. Sudharsan Iyengar, and Dr. Mingrui Zhang for advising and reviews throughout this project. I would also like to thank the Winona State University Computer Science Department for this opportunity.

References

[1] R. Sethi, "The top 11 most popular DAWs," in *Ask.Audio*, 2015. [Online]. Available: https://ask.audio/articles/the-top-11-most-popular-daws-you-voted-for. Accessed: Feb. 7, 2017.

[2] I. Hickson, "Media Capture and Streams," in *W3C*, 2016. [Online]. Available: https://www.w3.org/TR/mediacapture-streams/. Accessed: Feb. 7, 2017.

[3] B. Smus, *Web audio API*. Sudbury, MA, United States: O'Reilly Media, Inc, USA, 2013.

[4] Akamai, "Q4 2016 State of the Internet/Connectivity Executive Summary," in *Akamai*, 2016. [Online]. Available: https://www.akamai.com/us/en/our-thinking/state-of-the-internet-report/index.jsp. Accessed: Apr. 1, 2017.

[5] J. Nielsen, "How many test users in a usability study?" in *Nielsen Norman Group*, 2012. [Online]. Available: https://www.nngroup.com/articles/how-many-test-users/. Accessed: Feb. 10, 2017.

[6] R. G. Lyons, *Understanding digital signal processing*, 3rd ed. Boston, MA, United States: Prentice Hall, 2010.

The Use of XML to Store Patient Medical Record

Data Storage of Lung Cancer Treatment Outcomes Predicting Tool

Hok Lam Ou-Yong
Winona State University: Computer Science Department
Winona, MN 55987, USA
houyong13@winona.edu

***Abstract*—This research involves the restructuring of data storage for a web-based software Lung Cancer Treatment Outcomes Predicting Tool, which predicts lung cancer treatment outcomes. The current application was developed ten years ago and the software needs to be redesigned due the unnecessary software maintenance overheads. Extensive markup language is aimed to create a structured data form as part of the remodeling, this approach is taken to avoid a large number of flat files for storing medical record of lung cancer patients.**

I. Introduction

According to American Lung Association, lung cancer has one of the lowest 5-year survival rates among other leading cancer sites, only 15% of lung cancer patients are diagnosed in the early stage and almost half of the lung cancer patients do not survive within one year after being diagnosed [1]. In treating lung cancer, a web-based software application Lung Cancer Treatment Outcomes Prediction Tool (LCTOPT) was jointly developed by Winona State University and Mayo Clinic ten years ago and has been in active use at Mayo Clinic since then. There are five different clinical models in LCTOPT: Non-Small Cell (NSC), Limited Small Cell (LSC), Extensive Small Cell (ESC), Quality of Life (QoL), and Post-Surgery Recurrence (PSR). In each model, LCTOPT calculates the survival rates of a lung cancer patient based on the patient current condition and the selection of treatments. The survival rates are presented as survival curves on the web interface.

The current software uses the R statistical package in Java and the R server runs side by side of Java runtime environment with Apache Tomcat Version 6.0 Server. Patient information is added via SQL queries so all information needs to be passed between three different servers [2]. It has introduced unnecessary software maintenance overhead. During the redesign process, a Java version of the Cox Proportional Hazard (CPH) model implementation is developed with a simplified design to replace R statistical package and enforce standards by Health Level 7 [3]. It computes the survival rates of patients and significantly reduces the complexity of LCTOPT.

With more treatment models added over the past decade and more diagnosed patients live longer beyond five years, the prediction of survival rates is extended from 5-year to cover a 10-year period. A new responsive design allows displaying results on different mobile devices. Flot (JavaScript plotting for jQuery) is used for the web interface to present survival curves on a graph with features included checkboxes for selecting which curves to show, annotations to interpret survival rate accurately, and automatic resizing of graph when window size changes. These features are all combined together in the JavaScript functions of the JavaServer Page (JSP).

However, the current software stores patient information in multiple comma-separated values (CSV) files, which CSV is one of the types of flat file. Thus, this storage system is unorganized and non-optimized with different clinical models. Restructuring of data storage is indispensable to prevent multiple flat files. To complete the reformation, using extensive markup language (XML) is targeted to help store data and reduce the size and the complexity of the entire software. XML is a neutral meta-language that allows course contents separation and comprehensible with simple syntax; visual XML editors also acknowledge validation based on XML files' DTD (Document Type Definition) and the validity provides smoother function [4]. Since the developed Java version of CPH model implementation has simplified and streamlined, the XML data form storage implementation is intended to allow saving and reading data more efficiently for both new and existing patients in LCTOPT. In this paper, methods and results of comparing XML and CSV are presented to comprehend if storing data using XML file format has better latency on data saving and data retrieving than that of flat file format.

II. Software Redesign

A. Combination of Waterfall and Agile

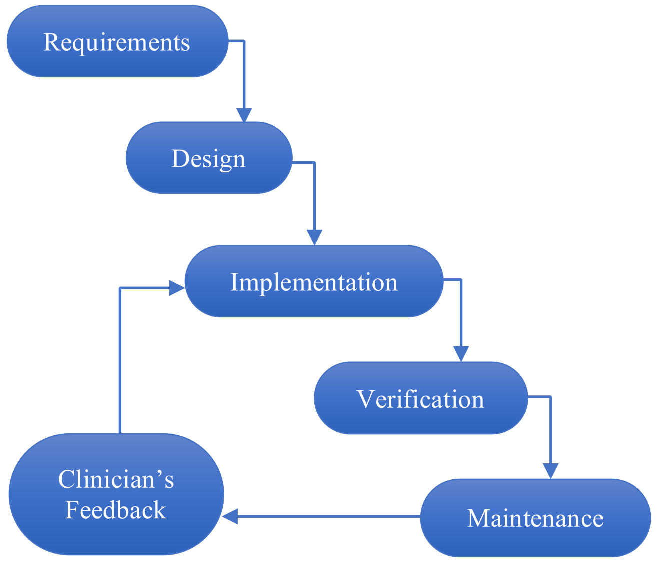


Fig. 1. Software development flow overview.

In this research project, waterfall model and Agile software development processes are combined, and given in Fig. 1. In the first phase, project requirements were collected from Mayo Clinic and system requirement specification was prepared. Then the application architecture was designed to fulfill the requirements in the next phase. During implementation phase, program was written to for building the application. Testing cases was performed in the verification phase to validate the application. In the maintenance phase, software tools would be updated periodically for better performance. Feedback was then received from clinicians in the following phase. Phases of implementation, verification, maintenance, and clinician's feedback are under agile development circle.

B. *Architecture*

The whole architecture behind includes different languages, tools, and platforms in LCTOPT, and shown in Fig. 2. Eclipse with a Java SE Platform is used as Integrated Development Environment (IDE) for programming the survival rates calculation of CPH model. For servlet container, Apache Tomcat Version 8.0 Server runs as the web server. The use of JavaServlet Page (JSP) as a front-end interface for input and output can dynamically generate different views of courseware utilizing Java XML data-binding and embed HTML codes [4]. The new XML data storage implementation interact with JSP for data saving and data accessing. JSP connects with the web server through the Internet and web interface is displayed on desktop or mobile devices with responsive design for clinicians and patients at Mayo Clinic.

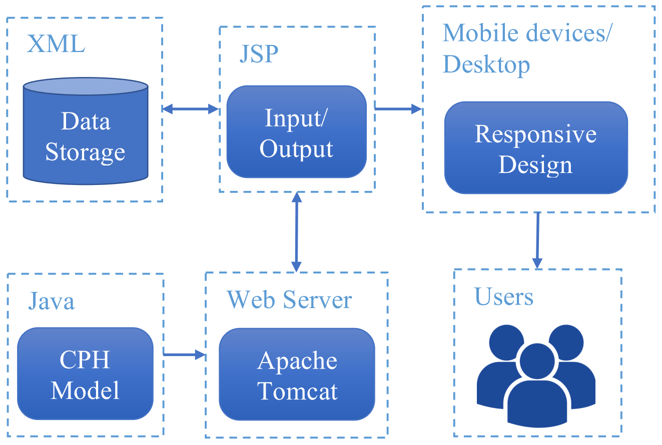


Fig. 2. Application achitectural design.

C. *Clinical Models*

Each of the five clinical models has certain specific information to be stored. General information of a lung cancer patient includes: age, gender, smoking history, cell type, stage, grade, and self-symptom [5]. Additional information is input based of which clinical model.

1) Non-Small Cell (NSC): It only obtains general information without any additional information needed.

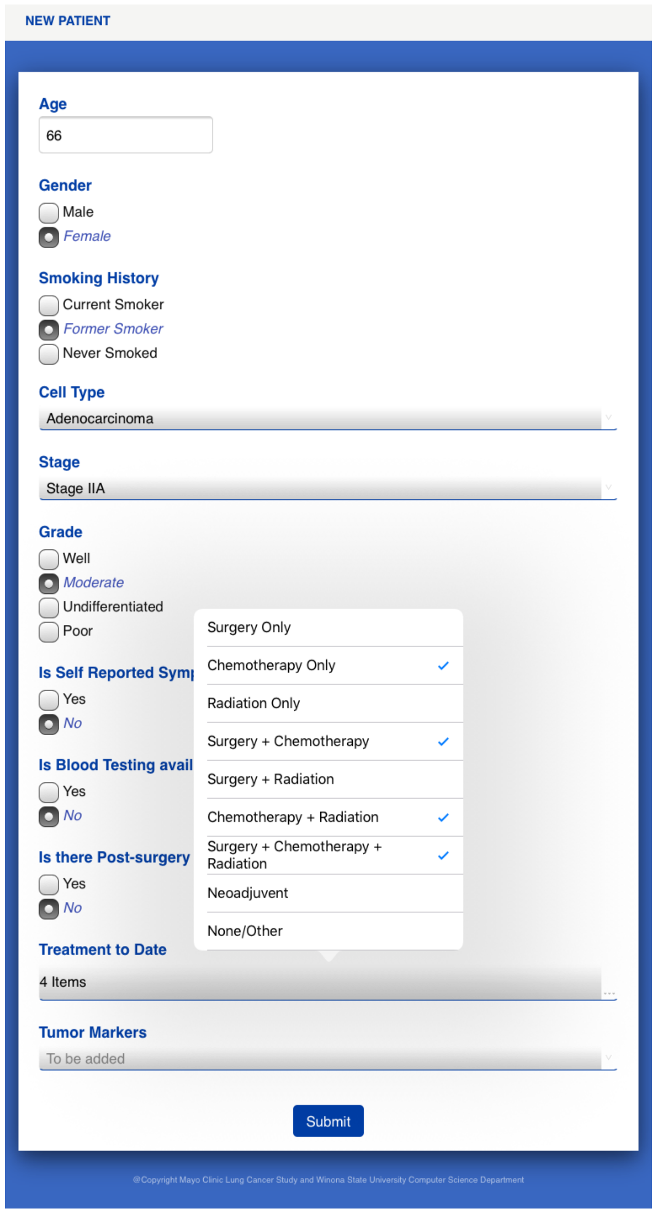
2) Limited Small Cell (LSC): Input of stage is disabled, blood marker is added automatically. Additional data includes: prophylactic cranial irradiation, red cell distribution width, lymphocyte, neutrophil, platelet, hemoglobin, ECOG performance score, and cessation of smoking.

3) Extensive Small Cell (ESC): It is very similar to LSC model except replacing cessation of smoking information with data of liver metastases and number of metastatic sites.

4) Quality of Life (QoL): If self-reported symptom is available, extra information requires comorbidities, symptom values of fatigue, symptom values of cough, and symptom values of short-breath.

5) Post-Surgery Recurrence (PSR): When there is post-surgery recurrence, additional information includes performance status at recurrence, symptoms at recurrence, liver recurrence, and number of recurrent foci.

D. *User Interface*



NEW PATIENT

Age
66

Gender
Male [✓ Female]

Smoking History
Current Smoker [✓ Former Smoker] Never Smoked

Cell Type
Adenocarcinoma

Stage
Stage IIA

Grade
Well [✓ Moderate] Undifferentiated Poor

Is Self Reported Symp
Yes [✓ No]

Is Blood Testing avail
Yes [✓ No]

Is there Post-surgery
Yes [✓ No]

Treatment to Date
4 Items

Surgery Only	
Chemotherapy Only	✓
Radiation Only	
Surgery + Chemotherapy	✓
Surgery + Radiation	
Chemotherapy + Radiation	✓
Surgery + Chemotherapy + Radiation	✓
Neoadjuvent	
None/Other	

Tumor Markers
To be added

Submit

©Copyright Mayo Clinic Lung Cancer Study and Winona State University Computer Science Department

Fig. 3. Screenshot of newPaitent.jsp in LCTOPT showing the input form of a lung cancer patient.

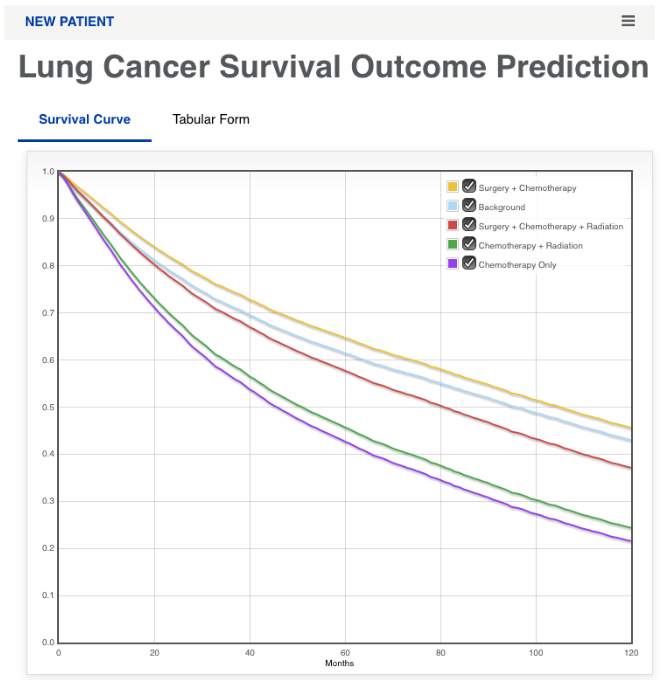
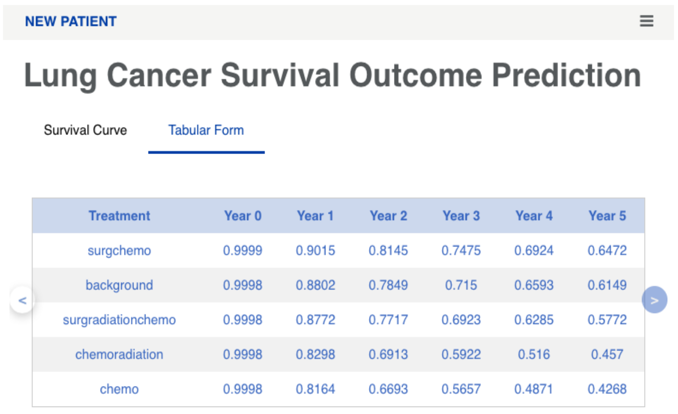


Fig. 4. Screenshot of nscCurve.jsp in LCTOPT showing the graph of the survival curves.



Treatment	Year 0	Year 1	Year 2	Year 3	Year 4	Year 5
surgchemo	0.9999	0.9015	0.8145	0.7475	0.6924	0.6472
background	0.9998	0.8802	0.7849	0.715	0.6593	0.6149
surgradiationchemo	0.9998	0.8772	0.7717	0.6923	0.6285	0.5772
chemoradiation	0.9998	0.8298	0.6913	0.5922	0.516	0.457
chemo	0.9998	0.8164	0.6693	0.5657	0.4871	0.4268

Fig. 5. Screenshot of nscCurve.jsp in LCTOPT showing the table of the survival rates.

E. *XML Implementation*

1) Structural design: Patient information is input by clinicians on the web interface, additional information depends on which cell type and selection of treatments. Data of new patients is data-centric, it has a well-defined structure and contains updateable data, such data is structured nicely as XML format. The XML structure schema for storing medical record of existing patients in NSC and QoL clinical models is as shown in Fig. 3. <Patients> is the root node in the file. Each patient record is stored as a child node of the root node as <patient id=""> with a unique clinic ID attribute. General information are stored as the child nodes of <patient> node. They are <agedx>, <gender>, <smk>, <celltype>, <stage>, and <grade> one-to-one. <reported>, <bloodMark>, and <recurrence> are child nodes that for determination of the clinical models. More child nodes are also stored if additional information is required. For the example with ID number 1002 in Fig. 3, four more child nodes <comorb>, <fatigue>, <cough>, and <dyspnea> are appended in <patient> node because the value of <reported> node is "Yes" which indicates the clinical model is QoL. Every <patient> node has <entryTime> node as the last child node to store the date of data entry for record purpose.

```
<?xml version="1.0" encoding="UTF-8"?>
<Patients>
  <patient id="1001">
      <agedx>66</agedx>
      <gender>Female</gender>
      <smk>Former</smk>
      <celltype>Adenocarcinoma</celltype>
      <stage>StageIIA</stage>
      <grade>Moderate</grade>
      <reported>No</reported>
      <bloodMark>No</bloodMark>
      <recurrence>No</recurrence>
      <entryTime>4/1/2017</entryTime>
  </patient>
  <patient id="1002">
      <agedx>83</agedx>
      <gender>Male</gender>
      <smk>Current</smk>
      <celltype>Adenocarcinoma</celltype>
      <stage>StageIIIB</stage>
      <grade>Poor</grade>
      <reported>Yes</reported>
      <bloodMark>No</bloodMark>
      <recurrence>No</recurrence>
      <comorb>Cocancer</comorb>
      <fatigue>3</fatigue>
      <cough>2</cough>
      <dyspnea>4</dyspnea>
      <entryTime>4/2/2017</entryTime>
  </patient>
</Patients>
```

Fig. 6. XML format examples of lung cancer patients in NSC and QoL clinical models respectively.

2) Java servlet read/write parser: Two servlet parsers are are save parser and read parser. Input of a new patient information on the web interface invokes the save parser in the Java Servlet and information is saved in the file with a unique clinic ID. The read parser is invoked in the Java Servlet after entering unique clinic ID of an existing patient on the web interface and then retrieves data of that patient. In the redsigned LCTOPT, only one form action is allowed to output the survival rates. Java servlet parser leads to two actions required in the same JSP form for both saving/reading data and displaying survival rates.

3) XPath in JSP: XML storage efficiently accommodate this type of data requirements since it is widely utilized for data-centric management [6]. With storing patient data within XML files as a plain text data format, it is ideally suited for such a format because of its wide availability of tools and parsers [7]. XPath is a syntax using path expressions to direct in XML files. Saving and reading data with XML documents can be done with Java XPath in JSP without using any servlet, which lets JSP form remain one action in LCTOPT to present graph and table with the survival curves and the survival rates respectively.

III. Testing and Validation

Comparison between XML format and CSV format is to test which method can save and retrieve data in LCTOPT more efficiently. We focus on the measurement of parsing data in NSC model on two main criteria: data memory size and data latency. Java servlet read/write parser is used to associate with both XML and CSV files for calculating the results. For data memory size, three datasets are randomly generated as patient data. The number of patients in these three different datasets are 10, 50, 250 and each dataset is saved in both XML format and CSV format. The space taken is measured to get the average memory size of one patient data for each format. For data latency, average data access times with 1,000 randomly generated data records are measured by continuously ten runtimes. Results are extracted from logs and calculated with mathematical functions for both formats.

Since LCTOPT is developed based on statistical models, the redesigned software needs to be validated before being put into use. Five Mayo oncologists were interviewed to select the best treatment for a number of patients. Given in Table I is an example of the treatment selection comparison between the expectations from oncologists and the predictions from LCTOPT, for a patient who is a 66-year-old female former smoker that diagnosed a moderate grade lung cancer with adenocarcinoma cell type based on different stages. The choices of best treatment from clinician matched closely with the software predictions.

TABLE I. Example of Treatment Prediction Comparison between Clinician and Software for a Lung Cancer Patient with Different Stages

Stage	Treatment Prediction	
	Clinician	*LCTOPT*
IA	Surgery only	Surgery + Chemotherapy
IB	Surgery + Chemotherapy	Surgery + Chemotherapy
IIA	Surgery + Chemotherapy	Surgery + Chemotherapy
IIB	Surgery + Chemotherapy	Surgery + Chemotherapy
IIIA	Surgery + Chemotherapy	Surgery + Chemotherapy
	Surgery + Chemotherapy + Radiaion	Surgery only
IIIB	Chemotherapy + Radiaion	Surgery + Chemotherapy
	Surgery only	Surgery only
IV	Surgery + Chemotherapy	Surgery + Chemotherapy
	Surgery only	Surgery only

IV. Results

A. Memory Size

Within three different datasets, memory sizes of three different datasets are measured. XML file format contains more memory space than that of CSV file format, given in Table II. The average memory spaces needed of one patient data are 308 bytes for XML file format and 68 bytes for CSV file format.

TABLE II. Memory Size Results of XML and CSV

Number of record	Memory Size (byte)	
	XML	*CSV*
10	2,978	755
50	16,757	3,417
250	72,680	14,588

B. Data Access Time

XML file format also obtains more data latency than that of CSV file format on most of the models as calculated, given in Table III. The average access times to retrieve a patient data among 1,000 records are 138.9 milliseconds for XML format and 60.3 milliseconds for CSV format.

TABLE III. Data Access Time Results of XML and CSV

Run (n^{th})	Access Time (millisecond)	
	XML	*CSV*
1	122	257
2	279	107
3	210	20
4	115	24
5	69	37
6	321	10
7	100	58
8	45	23
9	76	23
10	52	44

V. Analysis

The file format with less memory space needed and shorter time delay is preferable in LCTOPT. With the big data of patient medical record, a database server is not necessarily needed for data management in this application but an optimized data storage. Another server for database might indeed add more complexity to the application. The results for both XML and flat file formats are gathered and the difference is observed from Fig. 7. for average memory size and from Fig. 8. for average data access time. CSV file format consumes less data usage that may improve the storage efficiency thus overall performance. For LCTOPT as a real-life project, the significance of reducing data latency counters more quickly and it makes the whole decision making process faster as well. However, the complexity of adding extra servlets in this software and the poor-structured flat files data storage should also be encountered for the choice whether using XML format or CSV format.

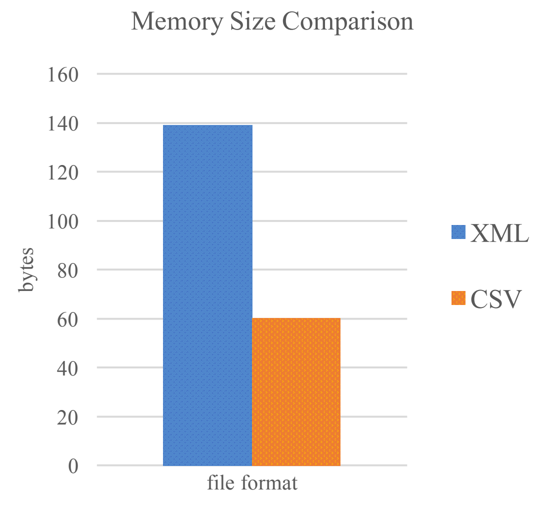


Fig. 7. Comparison of the average memory size in bytes of a medical record for a lung cancer patient in NSC clinical model.

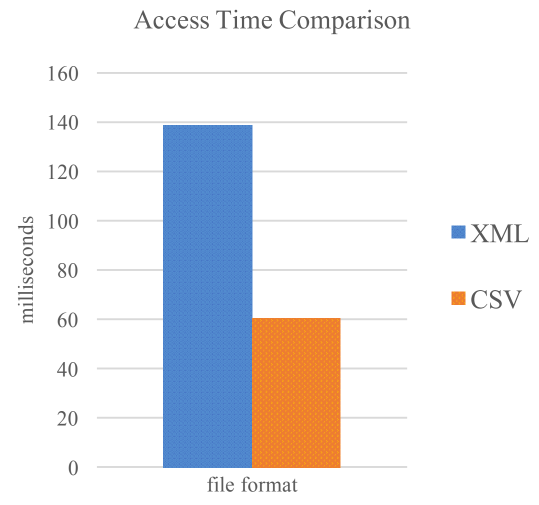


Fig. 8. Comparison of the average data access time in millisecond among 1,000 medical records in NSC clinical model.

VI. Conclusion

The goal of developing the XML data storage for LCTOPT is to function more efficiently, plus its self-organized structure can handle different clinical models all at once as each model acquires different coefficient data. Though the use of XML indeed holds more memory space and contains higher data latency, which is inferior than the use of flat files. For flat files data storage implementation, adding Java servlet is essential among the interaction of CPH program, Apache server, and JSP. Nevertheless, the use of XPath expression on JSP can support data retrieving from XML files without a single servlet. XML data storage is still a potential practice for LCTOPT.

Acknowledgment

I would like to thank to Dr. Mingrui Zhang, Dr. Iyengar Sudharsan, Dr. Narayan Debnath for guiding me throughout this project. Also, I would like to thank Winona State University Computer Science Department and Mayo Clinic for supporting this research. Java CPH model was initially developed by Nathan Martin for his senior project.

References

[1] U.S. National Institutes of Health. National Cancer Institute. SEER Cancer Statistics Review, 1975-2013.

[2] Zhang, M., Liu, Y., Jiang, Y., Sun, Z., & Yang, P. (2011, August). Model based user interface design for predicting lung cancer treatment outcomes. In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE* (pp. 75-78). IEEE

[3] Martin, N. (2016, April). Java Implementation of Cox Proportional Hazards Model. In *The 16th Winona Computer Science Undergraduate Research Symposium* (p. 28).

[4] P. Wubbelt, G. Femandez, and J. Heymer, "Clinical trial management and remote data entry on the Internet based on XML case report forms," Stud. Health Technol. Inf., vol. 77, pp. 333-337, 2000.

[5] Gegg-Harrison, T., Zhang, M., Meng, N., Sun, Z., & Yang, P. (2009, September). Porting a cancer treatment prediction to a mobile device. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE* (pp. 6218-6221). IEEE.

[6] Qu, C., Gamper, J., & Nejdl, W. (2001). A collaborative courseware generating system based on webdav, xml, and jsp. In *Advanced Learning Technologies, 2001. Proceedings. IEEE International Conference on* (pp. 197-198). IEEE.

[7] Emerick, J. (2002). Managing XML data storage. *Crossroads*, *8*(4), 6-11.

[8] Xie, D., Marks, R., Zhang, M., Jiang, G., Jatoi, A., Garces, Y. I., ... & Yang, P. (2015). Nomograms predict overall survival for patients with small-cell lung cancer incorporating pretreatment peripheral blood markers. *Journal of Thoracic Oncology*, *10*(8), 1213-1220.

[9] Cheville, A. L., Novotny, P. J., Sloan, J. A., Basford, J. R., Wampfler, J. A., Garces, Y. I., ... & Yang, P. (2011). The value of a symptom cluster of fatigue, dyspnea, and cough in predicting clinical outcomes in lung cancer survivors. *Journal of pain and symptom management*, *42*(2), 213-221.

[10] Sun, Z., Aubry, M. C., Deschamps, C., Marks, R. S., Okuno, S. H., Williams, B. A., ... & Yang, P. (2006). Histologic grade is an independent prognostic factor for survival in non–small cell lung cancer: An analysis of 5018 hospital-and 712 population-based cases. *The Journal of thoracic and cardiovascular surgery*, *131*(5), 1014-1020.

[11] Williams, B. A., Sugimura, H., Endo, C., Nichols, F. C., Cassivi, S. D., Allen, M. S., ... & Yang, P. (2006). Predicting postrecurrence survival among completely resected nonsmall-cell lung cancer patients. *The Annals of thoracic surgery*, *81*(3), 1021-1027.

[12] Turau, V. (2002, March). A framework for automatic generation of web-based data entry applications based on XML. In *Proceedings of the 2002 ACM symposium on Applied computing* (pp. 1121-1126). ACM.