

THE I-TEXT DISTANCE LEARNING SYSTEM

A Project

presented to

the Faculty of the Communication Department

at Southern Utah University

In Partial Fulfillment

of the Requirement for the Degree

Master of Arts in Professional Communication

by

DAVID A HARRIS

Dr. Jon Smith, Project Supervisor

December 2009

APPROVAL PAGE

The undersigned, appointed by the dean of Humanities and Social Science, have examined the project entitled

THE I-TEXT DISTANCE LEARNING SYSTEM

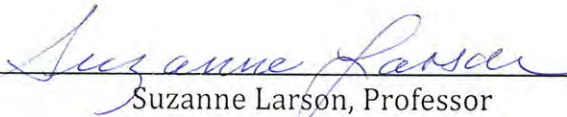
presented by David Harris,

a candidate for the degree of Master of Arts in Professional Communication,

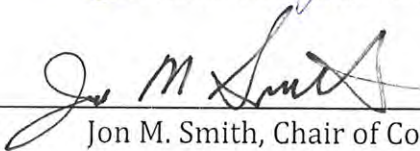
and hereby certify that, in their opinion, it is worthy of acceptance.



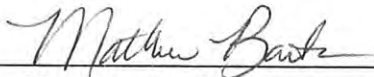
Brian L. Heuett, Professor



Suzanne Larson, Professor



Jon M. Smith, Chair of Committee



Matthew Barton, Graduate Director



James McDonald, Dean of HSS

TABLE OF CONTENTS

LIST OF TABLES	iii
INTRODUCTION	1
RATIONALE	1
METHOD	3
RESULTS	6
CONCLUSION	8
APPENDIX	
1. Sample I-Text	10
2. Questionnaire Distributed to Students	39
3. Source Code for the I-Text and Related Consoles	42
REFERENCES	108

LIST OF TABLES

Table	Page
1. Results of Student Survey	7

THE I-TEXT DISTANCE LEARNING SYSTEM

Many students benefit from taking classes via distance education. Valuable aspects of distance education include students' ability to stay at home during class time, to continue their employment, and to enjoy education with a great amount of freedom. But, does this freedom come at a cost? Are students able to navigate distance learning systems? The project that is the subject of this paper sought to determine the answers to these, and other, questions.

The project was a prototype distance education system called an I-Text (or *Interactive Textbook*). Students of a blended class were given the opportunity to use the I-Text and compare it to another distance education system, called Blackboard Vista. At the end of one semester, students were surveyed to determine their opinion regarding which system was better at certain aspects of education. The project (including source code), an example I-Text, the questionnaire, and the results of the survey are all presented in this paper.

Rationale

Many distance course systems are Internet based and require students to jump from link to link, searching for the portion of the course that contains the information pertinent to their current assignment. For example, an instructor may place course assignments in a syllabus, but those assignments are explained and submitted at an "assignments" link. In many cases, the information may differ between the syllabus and the assignments link. Also, some professors place course content in the "course content" area of the online system, while others place content in the "assignments" area. In short, course information, assignments, and content are spread throughout the learning system, and students must search out the information (which may not be entirely accurate across the system). Researchers have termed the ability to jump from various types of content to other types of content in a non-linear fashion *hypermedia* (Chen, 2002). Students that are presented a course in a hypermedia environment are given the freedom to explore the course in their own way. Researchers have studied the ability of a hypermedia environment to help students learn specific topics (Chen, 2002; Mitchell, Chen, & Macradie, 2005a; Mitchell, Chen, & Macradie, 2005b; Scheiter, & Gerjets, 2007). The studies examined how hypermedia

either helps or hinders students' ability to learn course content. In many cases, the studies have concluded that the use of hypermedia may cause confusion (e.g.-Kinshuk, 2009) or undue cognitive load (e.g.-Chen & Macredie, 2004) for students.

Chen (2002) developed the Cognitive Model of Hypermedia Learning in which she shows the need for coursework presentation to match the needs of both Field Dependant and Field Independent learners. Field Dependant and Field Independent learners differ in their cognitive approaches to learning. Field Independent students are better able to cope with coursework presented in a non-linear, or hypermedia, way. Field Dependant students are better able to cope with coursework that is linear and structured.

Three years later, Chen participated in a study (Mitchell, Chen, & Macredie, 2005a) that found that students were better able to use a non-linear, or hypermedia, structure when they had significant levels of system experience. In other words, when students are more familiar with the way in which courses are presented, they are more able to navigate the course topics. Mitchell, Chen, & Macredie also published a study that same year (2005b) that found that not only does system experience affect learners' ability to navigate hypermedia, but domain level expertise (the experience a particular student has in the subject area) also affects navigability of hypermedia.

So, a student's cognitive learning style, as well as their experience using a particular system affects their ability to navigate in a hypermedia environment. Dixie State College of Utah currently uses Blackboard Vista as its distance learning system. BB Vista is a hypermedia learning environment where course instructors can place course content on various links within the system. Students navigate the system by clicking on the various hyperlinks and examining the content therein. The BB Vista system provides a system that is well suited for students with a Field Independent cognitive learning style, but lacks the ability to provide a structured approach for Field Dependant learners.

A supplemental system to BB Vista is required in order to provide the continuous, top-down approach to distance learning required by these students. The greatest amount of structure that is most

familiar to students would be for the course to progress like a book: from one subject area to the next, without jumping around the system using links.

The system should also provide all of the elements of communication that students are familiar with in other distance education systems and in face-to-face classroom settings. This paper presents a project that attempted to take communication in current systems one step further. Current distance education systems mainly offer asynchronous forms of communication like e-mail. Some offer videoconferencing, but use of videoconferencing and other forms of synchronous communication is not well executed. What is needed is a true “distance” classroom where students are separated geographically, but not socially.

Distance education settings also cause issues for professors. Unlike a traditional classroom setting, a distance classroom causes repetitive communication events for instructors. Different students may have the same question. In a traditional classroom, all students hear the professor’s answer to a student’s question. Thus, communication is streamlined for all students. Current distance education systems attempt to reduce this problem by allowing professors to write announcements that are delivered to the entire class (or the professor must e-mail all students). This current solution does not reduce professor workload nor does it effectively provide answers to problems. What is needed is a system that facilitates synchronous communication between the professor and all students in the course, and asynchronous communication, though still available, takes a back seat.

Method

In order for the requirements listed above to be fulfilled, the project highlighted in this paper culminated in the production of what the project’s designer termed an “I-Text.” The I-Text is an instructor-produced pdf document filled with Rich Internet Applications (RIAs). The RIAs are open source Flash components developed by the project designer that instructors can choose to meet the needs of their courses. The project enabled students to use the I-Text to complete learning objectives either online or offline and participate in course meetings and discussions online, all without ever leaving the I-Text. The I-Text is a document viewable in Adobe Acrobat 9 and distributable over the Internet, FTP, or

via thumb drive. An entire course can be distributed in one, self-contained document. Below is a list of components that can be incorporated into an I-Text:

1. Communication Console – The main drawback of current distance education systems is a lack of knowledge on the part of students as to the availability of their instructor. Many systems use asynchronous forms of communication (i.e.-messaging and e-mail) for student-teacher communication. The I-Text Communication Console gives students additional communication opportunities. Four forms of communication are available in the console: phone, chat, videoconference, and messaging. In addition, the Communication Console allows instructors to signal availability to students for the various forms of communication. For example, instructors can make themselves available for videoconferencing, but not phone calls or chats. Students can check the Communication Console for instructor availability.
2. Syllabus Console – Course syllabi is managed in real-time using the Syllabus Console. Instructors can make changes to the syllabus and those changes will be reflected in students' I-Texts, even after the I-Text has been distributed.
3. Chat Console – The Chat Console gives instructors and students the ability to chat in real-time. The console provides one-to-one communication as well as one-to-many and many-to-many communication.
4. Message Console – Announcements and other messages are delivered to students using the Message Console. This console allows instructors to communicate with students and vice versa.
5. Live Console – This console allows instructors to stream live lectures to I-Text students. Students are able to participate in courses as if in the classroom.
6. Video Console – Recorded lectures, demonstrations, and other video recordings are available to I-Text students using the Video Console. Videos can be paused and restarted, as the student requires.

7. Quiz Console – Students can take quizzes using the Quiz Console. Instructors can build quizzes using four question types: multiple choice, true-false, fill in the blank, and short answer. The quizzes are available during certain times and students are given a time limit to complete and submit the quiz. Finally, students can have additional attempts at the discretion of the instructor.
8. Assignment Console – Instructors can place assignments in the I-Text and update those assignments in real-time using the Assignment Console.

The course instructor is able to place the components above in any page of the I-Text.

All Flash Actionscript 3.0 and Flex 3.0 code for each of the consoles is available in Appendix C.

One important outcome of the I-Text system is the top-down approach. Students that are Field Dependant are able to complete a course by working from the top of the I-Text document to the bottom. Students are not required to bounce around from link to link in order to find assignments, syllabi, and course content. This book-like structure eliminates confusion and instructor intervention simply answering questions unrelated to the course material. In addition, students that are not familiar with the system or the subject of the course should find the I-Text navigable as it is presented like a book.

Finally, the I-Text system is able to facilitate synchronous forms of communication as much as possible. Students are empowered to communicate effectively with professors, and vice-versa. The most important outcome of this project was to provide a system where students and teachers are separated geographically, but not socially. The I-Text does this by providing live video stream capabilities within the book itself. Students can ask questions of other students and the instructor immediately.

In order to test the ability of the I-Text to aid students understanding of specific course topics, a sample I-Text was produced and provided to a class of undergraduates participating in a course at Dixie State College. The course being offered was a blended course in which students came to class to participate in lectures, but course reading, assignments and other course content was provided online. In the past, the course content was only provided using BB Vista. For purposes of this project, students had the opportunity to use both BB Vista and the I-Text for course content over the course of the semester.

The I-Text was developed using Acrobat 9, Flash CS3, Flex 3.0, PHP and MySQL. The individual consoles were written in Actionscript 3.0 and compiled using Flash CS3. The consoles communicate with a web server using PHP as an intermediary. Database information was stored in a MySQL database and retrieved using PHP. In addition to the I-Text consoles, the course professor utilized the Professor Console to manage course content. The Professor Console was written in Flex 3.0 and compiled using a free compiler provided by Adobe Systems. Finally, the I-Text was produced and the RIA's embedded using Acrobat 9. For the survey, the existing course text, written by the course designer was used as the main I-Text reading content. Both BB Vista and the I-Text contained identical information. The only difference was in how the content was structured. BB Vista used traditional hypermedia organization, and the I-Text used the top-down approach. A sample chapter from the I-Text is provided in Appendix A.

Upon completion of the course, the students were provided with a questionnaire to gather their feelings toward BB Vista and the I-Text systems. This questionnaire is provided in Appendix B. The main purpose of the questionnaire was to determine which system was better at providing students with course information on three different levels: Navigation, Communication, and Overall Performance.

Results

Twenty-two students completed the course. Of those twenty-two students, fifteen completed the survey. The results of the survey are presented in Table 1. Overall attitudes toward the I-Text were mixed. Some students preferred the I-Text and others preferred BB Vista. However, all students liked the level of communication offered by the I-Text.

Results show that students found both systems to offer the same level of feedback. This could be because the professor offered the same amount of feedback to all students regardless of the system they were using. This result could also be attributed to the course work itself. In many cases, the course work did not lend itself to feedback. Students may have not known how to answer since they may have wanted to choose, for example, "No Feedback Provided."

Table 1 - Results of Student Survey ($n = 15$)

	BB Vista Better	Both the Same	I-Text Better
Feedback	1	12	2
Navigation	6	5	4
Easier to Understand	4	8	3
Easier in General	4	7	4
Communicate Course Expectations	3	5	7
Better in General	2	7	6
Guidance	4	5	6
Instructor Communication	3	2	10
Class Feeling	2	10	3
Success	3	8	4

Students that thought the I-Text was more navigable made statements to the effect that the I-Text presented information more clearly, that it read like a book, and that they did not have to click as much to access the information they needed. Students that thought BB Vista was more navigable stated that they were used to BB Vista because they had used it before in other courses. Further testing is required with students who have not used either system in the past to determine if new users find the I-Text more navigable than BB Vista.

The survey showed that students did not consider the I-Text to improve on ease-of-use, both in understanding and in general. These results could possibly be attributed to the fact that many (if not all) of the students had used BB Vista in the past. They may already be used to it, and therefore, find it easier to use than a system with which they are unfamiliar.

Students were quite happy with the level of communication offered by the I-Text. Some comments were that students knew when they could contact the professor. Others were happy that

contact was synchronous (though they used different words). Further study is necessary to determine in what ways communication was better with the I-Text system.

The differences in opinion with regard to overall impression of the I-Text may be explained by the differences in cognitive learning style of the individual students. Some students may prefer the structured approach of the I-Text while others may prefer the hypermedia approach of BB Vista. Further study is required to determine (scientifically of course) whether cognitive learning style affects attitudes toward the I-Text.

Students found that both systems offered the same level of guidance. Indeed, this could be attributed to the course content rather than the system itself. In the end, the professor must be the one offering guidance to students. If the professor is slow to respond or if the content itself is not clear, the system being used does not matter.

Students found the I-Text to offer better instructor communication. The Communication Console (or anything like it) is not found in BB Vista. Students clearly thought the I-Text offered better access to the professor. Students who were learning at a distance and did not understand course content had the ability to contact the professor to get answers to questions and were clearly happy with the capability.

The last two items on the survey, Class Feeling and Success, were viewed as equal between the two systems. Distance education itself may not lend itself to a class feeling or both systems may just have performed equally well (or unwell) at offering students a feeling of being a part of a group or being more successful.

Conclusion

The I-Text Distance Learning System was successful at providing students with better communication opportunities between students and the course instructor. However, overall attitudes toward the system were mixed. Additional study is required to determine in what ways the I-Text system can be improved. The system itself proved to be quite robust. Assignments were submitted and communication took place in the exact way in which the system was designed. The open-source nature of

the system allowed for slight modifications to fit the needs of the course itself. Students seemed to adapt well to the new environment.

The I-Text system was successful on many levels. Modifications to the system will serve to improve the ability for the system to be used more fully in undergraduate courses. In fact, upon finding out about the system, a colleague in the communication department at Dixie State College expressed great interest in the system and desires to utilize it in his course next semester. What greater success is there than that?

APPENDIX 1

SAMPLE I-TEXT

Following is a sample I-Text. The text content is Copyright 2009 Randal Chase and is used with permission. The I-Text requires at least Adobe Acrobat 9 to view correctly.

Interactive-Textbook For

Loading Course Info, Please Wait. . .



Note: If you do not see course information listed above, this I-Textbook is having a problem communicating with the I-Textbook server. Please ensure that you have an Internet connection and that you have allowed this document to communicate with the server. If you continue to have problems, contact your instructor.



Course Syllabus

Loading Syllabus, Please Wait. . .



Chat



Messages



Elements, Trends, and Issues of Digital Technology



Instructions for Chapter 3

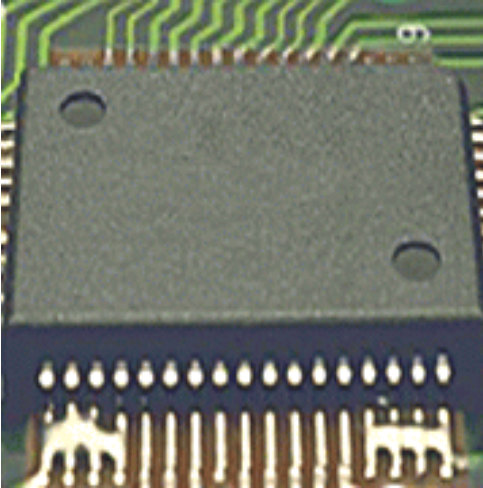
1. Read the text of Chapter 3 found on the next handful of pages (42-57).
2. Also read the Chapter Summary on pg. 57, Important Terms and Concepts on pg. 58, and the Study Questions on pg. 60.
3. Complete the two assignments found on pages 62 and 63 before their due dates.

Complete Assignment 1 before: ~~Friday~~, September 11, 2009 - 11:59 p.m.

Complete Assignment 2 before: ~~Friday~~, September 11, 2009 - 11:59 p.m.

3 Digital Devices

Digital Hardware	43
Internal System Components	44
<i>Central Processing Unit</i>	44
<i>System Board</i>	44
<i>Processor</i>	44
System Effectiveness Factors	45
<i>Processor Bit Value</i>	45
<i>Processor Cycling Speed</i>	45
<i>System Memory</i>	46
<i>System Bus (Connections)</i>	47
<i>Disks, Disk Drives, and Thumb Drives</i>	49
Digital Software and Peripherals	51
<i>Software</i>	51
<i>Application Software</i>	51
<i>Operating Systems</i>	51
<i>Peripheral Devices</i>	52
Chapter Summary	57
Important Terms and Concepts	58
Activities & Discussions	59
Study Questions	60
Chapter References	61



Chapter 3

Digital Devices

Modern digital devices have a venerable history dating back to 1936. Early digital devices were basically electrical logic machines and complex number calculators. The first special-purpose electronic digital computer appeared in 1939 and the first general-purpose one—the ENIAC—in 1946. Devices using microprocessors didn't appear until 1971, and personal computers like the Apple (1976) and IBM PC (1981) came even later (Cringely, 1996; Tripod, 1994). By the end of the century, however, microprocessor-based digital devices of all kinds had changed nearly every aspect of our lives.

Strictly speaking, all digital devices are computers. Digital televisions, cellular telephones, palm-sized *PDA*s (personal digital assistants), and digital cameras all use *processors* (intelligent control chips) to accomplish their work. So do non-media devices like cars, microwave ovens, and toasters. So when we talk about digital devices in this chapter the discussion is not limited to desktop computers. Learning a few things about how digital devices work is an essential foundation for the discussions later in this text of the elements and issues of digital communication.

This chapter provides a basic explanation of some of the internal components of digital devices—those that reside inside the system and are seldom seen by the average user. It also discusses a few of the most basic external components, called *peripherals*, that are part of the rapidly expanding world of digital devices. And it discusses briefly the software, called *operating systems*, that operates digital devices at their most basic level.

Digital devices operate on three levels: the hardware, the operating system, and the application software (see Table 3-1).

Table 3-1 Three Levels of Digital Devices

System Level:	Examples:
Application Software	Word Processing Software
Operating System	Windows, MacOS
Hardware System (BIOS)	Computers, Palm Devices

The most basic of these levels is the hardware, sometimes called the *BIOS* (basic input/output system). Consisting of metal and plastic, hardware is basically unintelligent and needs an *operating system* (such as Windows) to perform useful functions. The operating system actually operates the hardware. Finally, application programs like Microsoft Word or the programs that run on palm devices do specific sets of tasks for the user. *Application software* relies upon the operating system to tell the hardware what to do, so applications must be designed for specific operating systems like Windows (Boyce, 1998; Boyce et al., 1995; Livingston & Straub, 1995; Microsoft, 1996).

DIGITAL HARDWARE

Personal computers are the most familiar of all digital hardware devices. Historians disagree about who built the first “personal computer,” and of course the answer depends upon the definition. Edmund Berkeley conceived and wrote about the idea in 1949, and there were many personal-sized desktop computers invented throughout the 1950s and 1960s. Many consider the Xerox Alto the first PC in the modern sense, because it featured a mouse, a GUI, and fast networking, all in 1973.

The Apple II was first mass-produced personal computer on the market, in 1977. But IBM PC-compatible computers eventually took the lead due to wider availability of useful software. This occurred primarily because IBM adopted an *open architecture* policy— providing software vendors open access to the PC’s technical information to encourage them to write software for the PC. Apple computers were easier to use, had superior graphics, and offered a user-friendly interface, but the lack of application software limited their utility with business and personal users. Personal computers started the digital revolution that has now changed nearly every aspect of our economy and our lives.

Hardware consists of physical devices—the components you can see and touch. Digital hardware always contains a processor, usually in the form of a computer chip, located somewhere in the *central processing unit (CPU)*. Hardware also includes peripheral devices such as keyboards or printers.

Internal System Components

Central Processing Unit (CPU)

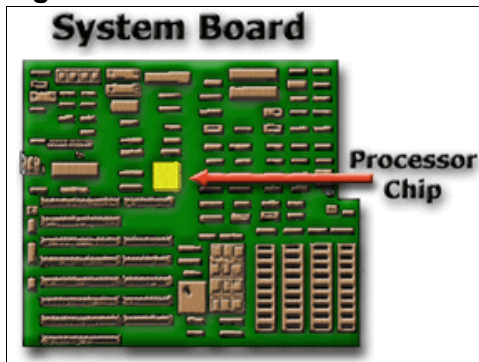
Every digital device has a central processing unit or *CPU*. On personal computers, the CPU usually consists of a rectangular box that lies flat on your desk or stands up like a tower, as shown in Figure 3.1. Smaller digital devices, like cellular phones, provide all the functions of a CPU within the device itself. The CPU contains a *system board* which does the actual processing of programs and data. The CPU may also contain internal devices such as disk drives. Many CPUs also have *ports* that permit plugging in of external devices such as keyboards or speakers.

Figure 3.1



System Board

Figure 3.2



The main or master circuit board that lies inside the CPU is called the *system board* (see Figure 3.2). All other components and devices (such as the processor, memory chips, and external devices) connect to this larger, main system board. Because this board controls and gives direction to all other devices of the system, it is sometimes called the *mother board*. It is the central mechanism through which all the components of a digital device interact.

Processor

The *processor* is the brain of a digital device. It usually consists of a single chip, located on the system board, that does all of the actual processing of programs and data (see Figure 3.2). An example of a processor chip is the Intel Pentium chip that has dominated PCs for many years. Over the years, computer processors have

become faster and increasingly capable of performing more demanding tasks such as multimedia and 3D graphics.

SYSTEM EFFECTIVENESS FACTORS

There are four factors which impact the speed and effectiveness of a digital device. When deciding which of the many available devices to buy and use, a knowledgeable consumer will consider all four of these factors. They are also important to the discussion of devices and services that will follow in this text. The four factors that affect device speed are:

1. Processor bit value (e.g., 16-bit or 32-bit)
2. Processor cycling speed (measured in MHz)
3. System memory size (particularly RAM memory)
4. System bus capacity (measured in bit value)

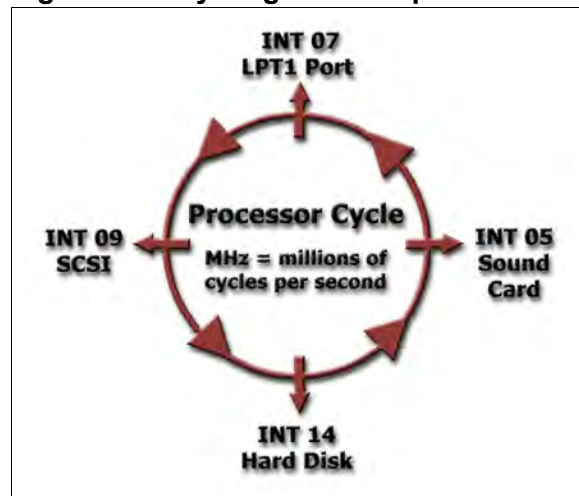
Processor Bit-Value

The *processor bit-value* is the number of “bits of data” a processor grabs each time it takes a “bite” from your program or data files. The larger the bite, the faster the device will process the programs and data. For example, a 32-bit processor manipulates twice as many bits per cycle as a 16-bit processor does. The differences in speed among processors can be dramatic—a 2-to-10 times increase in processing speed with each bit-value upgrade.

Processor Cycling Speed

The *cycling speed* of the processor is controlled by a timing clock on the system board. This clock does not tell time; it controls how quickly the processor performs its tasks. These timing devices can be thought of as racetracks around which the digital processor is cycling, waiting for something to do (see Figure 3.3). The cycling speed of a device is measured in *megahertz* (MHz) which represents millions-of-cycles-per-second. The same processor will perform much faster in a 400 MHz system than in a 200 MHz system.

Figure 3.3 Cycling & Interrupts

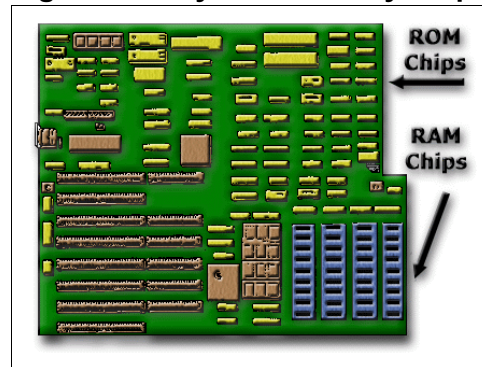


As requests are made by the various parts of the system (e.g., the sound card), the system interrupts its cycling pattern and performs the requested task. For that reason, the locations where these interruptions occur are called *interrupts*. Just about every device or function of a digital system will occupy one or more interrupts, and usually two devices cannot occupy the same one. Many times, when devices on a digital system fail to work, it is because two or more devices are attempting to use the same interrupt.

System Memory

The third factor in device speed is the amount of *system memory* the device contains. System memory should not be confused with the size of disk drives or other storage devices. Disk drives are like file drawers into which data and programs can be placed and retrieved at any time. System memory, by comparison, is located in the banks of chips that are inserted into the system board (see Figure 3.4). System memory is usually measured in megabytes (MB) or gigabytes (GB) of storage. System memory is used by devices in two ways: as ROM or RAM.

Figure 3.4 System Memory Chips



ROM Memory: Not all system memory is available to the user. Some parts of system memory are permanently encoded with instructions or data. This kind of memory is called *read only memory (ROM)* because the system cannot write any information to these areas; it can only read from them. Usually, ROM memory is located on chips that have been manufactured with read-only instructions (the yellow chips in Figure 3.4). ROM technology is important because it allows the system to permanently store often repeated instructions or data that would otherwise use up RAM memory. Small digital devices such as palm computers make extensive use of ROM memory technology.

RAM Memory: *Random access memory (RAM)* consists of rows or banks of chips that, when the device runs a program, can temporarily hold the applicable programs and data (the blue chips in Figure 3.4). RAM memory is a temporary work space—like a desktop or a scratch pad—where information can be stored and manipulated while the system is powered on. RAM memory completely erases when you turn off the device, or when you *boot* (reset) it by pushing a restart button.

The amount of RAM limits the size and complexity of programs that can operate on a digital device. Generally speaking, the more RAM the better, because over the years software programs have become ever-more demanding of RAM memory to perform their tasks. In personal computer, anything less than 16 MB of RAM is rare, and most programs require 32 MB, 64 MB, or more. For palm devices anywhere from 2 MB to 8 MB is common.

System Bus (Connections)

The fourth factor in device speed is the *bus*—the connections and wires within a computer system. These can be thought of as a “bus” with a set carrying capacity, transporting its “passengers” from point to point. In this case, the passengers are bits of data that are carried from one part of the system to another (e.g., from a file to the display). The carrying capacity of the bus greatly affects system speed (Chase, 1999; Fulton, 1998; Kraynak, 1998).

Over the years, computer systems have used a variety of bus types to accomplish their work (see Table 3-6). The earliest was the serial bus. The most common for printers was the parallel bus. More recently, systems have used SCSI buses to connect multiple devices. Laptop computers generally use PCMCIA buses. And recently, USB buses have become a common standard.

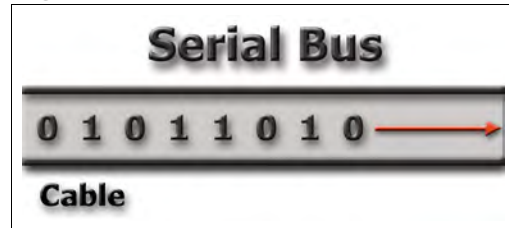
Table 3-6 Types of Buses Used on Personal Computers

Year	Interface Type	Developer(s)	Explanation
	Serial bus	Various	Original bus for computer systems
1974	Mouse, Network	Xerox	Developed mouse, network, GUI, windows
1986	Parallel bus	Inmos UK	First parallel port processing
1986	SCSI bus	Various	Small Computer System Interface standard
1987	Plug ‘N’ Play	Apple Mac II	First self-installing hardware devices
1987	MCA bus	IBM	Microchannel Architecture non-PC compatible
1988	ISA bus	IBM	Industry Standard Architecture was compatible
1988	EISA bus	Various	Enhanced ISA adopted by IBM competitors
1989	PCMCIA bus	Various	PC Memory Card Int’l.Assoc. notebook slot std
1992	PCI bus	Intel	Peripheral Component Interconnect standard
1992	VESA VL-bus		
1993	Plug ‘N’ Play	Microsoft	First Microsoft plug ‘n’ play capability
1997	USB bus		Universal Serial Bus is the current standard

(Sources: Chase, 1999; Cringely, 1996; Mesa, 1998; Tripod, 1994)

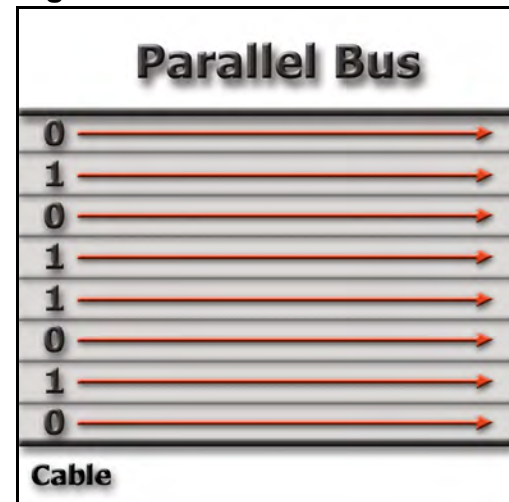
Serial buses send bits in sequence, one right after another in “series” or, in other words, in “serial” fashion (see Figure 3.5). Serial buses were among the earliest types used in computer systems. Many early printers on personal computers used a serial connection, and were therefore called “serial printers.” External serial devices connect with digital devices through serial ports. The first serial port is called COM1, the second (if any) COM2, and so forth.

Figure 3.5



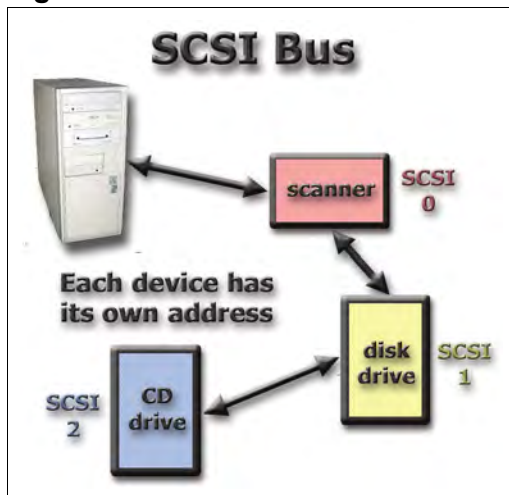
Parallel buses transport multiple bits over multiple wires simultaneously— or in other words in “parallel” fashion (see Figure 3.6). On cables can be accomplished by devoting one wire to each bit path, allowing all bits to travel at once through multiple wires within the cable. External parallel devices connect with digital devices through parallel ports. The first parallel port is called LPT1, the second (if any) LPT2, and so forth. Most printers on personal computers use parallel connections and are therefore called “parallel printers.”

Figure 3.6



SCSI buses were once an industry standard for connecting multiple peripheral devices through a single port. *SCSI* (pronounced “skuzzy”) is an acronym for Small Computer System Interface. A single SCSI adapter can control up to seven different devices at once, each with its own SCSI address along a continuous chain of connected devices (see Figure 3.7).

Figure 3.7



Today, smaller digital devices like laptop computers, palm devices, and cell phones use the *PCMCIA bus* (PC Memory Card International Association) because of its extremely small, flat connectors. And now, on digital devices of all kinds, the

industry standard is becoming the *USB* (Universal System Bus) that allows multiple devices to be connected simultaneously (like SCSI buses) but does not require re-booting the system each time a device is plugged in (Chase 1999; Fulton, 1998; Kraynak, 1998).

Disks, Disk Drives, and Thumb Drives

Disk drives are used on many digital devices, ranging from digital cameras to personal computers. They are used to store data and programs in a more permanent manner. Unlike memory storage, disk storage is not erased when the system is turned off (Chase, 1999; Fulton, 1998; Kraynak, 1998).

Fixed disk drives are located inside the CPU and are fixed—they remain inside the CPU and cannot be removed like diskettes. *Fixed disks* are circular metal disks covered with oxide. Because they are metal and inflexible, they are sometimes called *hard disks*. Fixed disks spin at much faster speeds than removable diskettes, making them 10 to 20 times faster. Over the years, fixed disks have increased dramatically in their capacity (see DigiTrends box below).

Figure 3.8



Removable disk drives allow users to insert removable media, store information on them, and then remove them. Examples of removable media used in the past are diskettes and Zip drives (see Figure 3.8).

Compact Discs (CDs) replaced diskettes and Zip drives because they can hold large quantities of content—74

minutes of music or 650 MB of data. CDs are metal discs coated with plastic. The metal layer is pocked with depressions or pits that correspond to the 0s and 1s of digital encoding. These pits are detected by tiny laser optics as the disc spins in the CD drive. This type of CD, called a *CD-ROM* (*compact disc—read only memory*) is usually mass produced in factories, and because they are pitted data can only be written to them once. *Writeable CDs* are special CDs on which data can be written by the user. There are two general types: CD-R can be written to only once by the user (write only). CD-RW can be written to multiple times (write and rewrite).

Digital Video Discs (DVDs) are the most recent development in removable discs. These extremely high capacity discs were originally used to play movies and other video content. DVDs are much like CDs in size and function, but they have a vastly larger carrying capacity—up to 3.5 billion bytes (gigabytes). An entire movie can be placed on one CD-sized disk. Video stores now rent DVDs rather than videotapes, and DVD recorders are a popular item on new computers because they can store so much data.

Thumb Drives (Memory Sticks) are the latest version of removable storage, and have become very popular because of their portability and large capacity. Even though they are smaller than CDs and DVDs (see Figure 3.9) they can hold many gigabytes of storage, making them capable of storing large video files as well as entire music libraries and many books. The most convenient aspect of these sticks is their use of a USB connector, which allows them to be plugged into virtually any computer with ease. It is likely that USB thumb drives will continue to increase in capacity and will become the standard in removable storage for years to come.

Figure 3.9



DigiTrends

Increasing Disk Capacities

The original floppy diskette for PCs held only 180 KB of storage. Over the years, this was replaced by higher capacity floppies and then by 3.5 inch diskettes. Today, USB “thumb drives” can hold 8, 16, or even 32 gigabytes (GB) of data. Fixed disks (hard drives) have also increased their storage capacity dramatically. The first ones held only 10 MB. Today, even though they are physically much smaller, they hold hundreds of GB and even terabytes (TB) of data.

Year	Drive Type	Capacity	Explanation
1978	5.25" floppy disk	180 KB	Single-sided, standard density for Apple
1980	Winchester Disk	10 MB	Multiple-layer, double-sided hard disks
1981	3.5" diskette	437 KB	Single-sided, standard density
1983	3.5" diskette	760 KB	Double-sided, standard density
1983	5.25" floppy disk	360 KB	Double-sided, standard density for IBM PC/XT
1983	CD-ROM	650 MB	First CD-ROM drives developed
1990	105 MB Hard Disk	105 MB	Designed for NeXT computer
1990	3.5" diskette	2.88 MB	Designed for NeXT computer
1990	2x CD-ROM Drive	Various	Original CD-ROM drives for PCs
1992	CD-R	650 MB	Writable CD drives appear
1995	4x CD-ROM Drive	Various	Increased speed CD-ROM drives appear
1995	DVD Drives	Various	Digital Video Disk drives appear
1995	LS 120 Drive	120 MB	Compaq drive also reads 1.44 MB std floppies
1995	Zip Drive	100 MB	Iomega's super-floppy removable appears
1997	CD-RW	650 MB	Re-Writable CD drives appear
1998	DVD-ROM	5.2 GB	2.6 GB per side
Today	Thumb Drives	8, 16, 32 GB	Easily portable and compatible USB devices

(Sources: Chase, 1999; Cringely, 1996; Mesa, 1998; Tripod, 1994)

DIGITAL SOFTWARE AND PERIPHERALS

Digital devices have changed dramatically since the inception of the personal computer in the 1970s, as have all of their peripheral supporting devices have also improved—displays, printers, scanners, sound systems, and the like— along with operating systems and software, providing capabilities not even imagined three decades ago (Chase, 1999; Cringely, 1996; Mesa, 1998; Tripod, 1994). Many of these changes are documented in the nearly annual editions of the book *Communication Technology Update* which provide much of the trend data cited in this text (Grant, 1994, 1996; Grant & Meadows, 1998, 2000; Grant & Sung, 1992; Grant & Wilkenson, 1993). The purpose of the rest of this chapter is to summarize and discuss these changes, starting with a brief explanation of operating systems and software, then proceeding to some of peripheral devices that support and add functionality to all digital systems.

Software

Software consists of the programs, data, and metadata that run on the hardware. Both operating systems and application programs are considered software, though they operate at different levels of the system (see Table 4-1). Software is usually loaded onto a digital device from program files obtained from the program vendor. Software is written in *computer languages* (Basic, Unix, Java, HTML, etc.) which the system converts to the bits and bytes that the processor and other components can understand. When the user runs these programs, the system loads them into RAM and executes them from there. This is true for both operating system software and application software.

Application Software

The software programs used to do specific tasks are called *application software*—software that is designed to fulfill the user's particular needs. Common tasks include word processing, e-mail, Internet access, financial management, and more. Application software is also called *content* when applied to all of the products and services available on digital devices. Table 3-7 shows a brief history of “firsts” in personal computer application software development.

Content is discussed in greater detail in Chapter 6, and no attempt will be made to discuss content or applications at this point. It is mentioned here only to distinguish it from operating systems and hardware.

Operating Systems

Most digital hardware systems require an operating system. Application software talks to the operating system, which in turn talks to the hardware (see Table 1-1). Most modern operating systems use graphical user interfaces or *GUI* (gooey) for short. These permit use of a mouse, a pen, or some other device to point at, highlight, touch, or click on graphic icons. When the user makes a selection in this manner, it activates a particular feature of the software. GUIs first appeared on Xerox Alto computers, then Apple computers, and now, with Microsoft Windows, on nearly every digital device from desktop computers to palm devices. GUI is popular because of its *user friendliness* or ease of use.

Operating systems (OS) are, in fact, software—but they are very special software that helps hardware devices to do their work. Some digital devices use their own proprietary operating systems, which makes them incompatible with systems that use a different OS. Tables 3-8 and 3-9 show how personal computer operating systems have developed over time. Recently, the latest versions of the operating systems have been appearing on smaller, more portable digital devices as well.

Operating systems do more than just help application software programs do their work. They also work with the hardware, providing instructions to both internal and external components in their native *machine language*—0s and 1s.

If digital devices are to ever become truly interoperable, they will need a common operating system. Microsoft is one of several companies vying to supply that universal operating system, which in their case, of course, would be Windows. It is likely that televisions, telephones, PDAs, as well as computers will all eventually use similar operating system software.

Peripheral Devices

Hardware components that operate outside of the main digital device are called *peripherals*. For computers, these include things like monitors, printers, modems, and scanners. For hand-held digital devices like cell phones and palm devices, they might include modems, keyboards, or larger displays. Many of the most visible changes to digital systems occur in their peripheral devices. Those changes occur just about every year—sometimes more than once per year—adding new features and conveniences not available before.

Input Devices: There are many *input devices* for digital devices. The most obvious on personal computers are the *keyboard* and the *mouse*. Small handheld devices like PDAs and cellular phones also have input devices such as alphabetic and numeric keyboard displays that the user “types” on by touching the screen.

Displays are screens or monitors on which the user's choices and work appear while using the system. Most large digital displays, such as computer monitors, use progressive scan technology (see Chapter 2), but displays on smaller hand-held devices usually use *LCD (liquid crystal display)* technology (see Figure 3.10). Most LCD displays are *monochrome*—they display text in a single color (usually green or black) against a neutral background. Monochrome monitors were standard on early Apples and PCs but are very rare today on personal computers. While most small portable hand-held devices like PDAs and cell phones use monochrome LCD displays, some recent models offer color displays capable of showing Internet pages and the like. *Full-color* displays are capable of showing from 16 to several million different colors, depending upon the type of display. These are the standard monitor type for personal computers, and most software today is written to take advantage of color displays.

Figure 3.10 Displays for Digital Devices



DigiTrends

Display Developments

<u>Year</u>	<u>Monitor Type</u>	<u>Developer(s)</u>	<u>Explanation</u>
1976	CPU Encased	Apple	First B&W monitors encased in the CPU box
1977	Detached	Apple	First non-CPU encased monitor for Apple II
1977	RGB Color	Apple	First color monitors for Apple II
1984	EGA	IBM	640x350 with 16 colors; For IBM-PC XTs and ATs
1986	Multisync	NEC	
1987	VGA	IBM	640x480 with 16 colors; For IBM PS/2 systems
	XGA	IBM	Even higher resolution for graphic-intensive pgms
1988	SVGA	Various	Current standard; High resolution, millions of colors
1988	LCD B&W	Various	B&W LCD displays used on first laptop computers
1990	SVGA 32-bit	Various	32-bit video cards with greater display speed
1992	Video RAM	Various	First video cards with additional RAM onboard
2000s	Flat screen	Various	The standard screen for most monitors today.

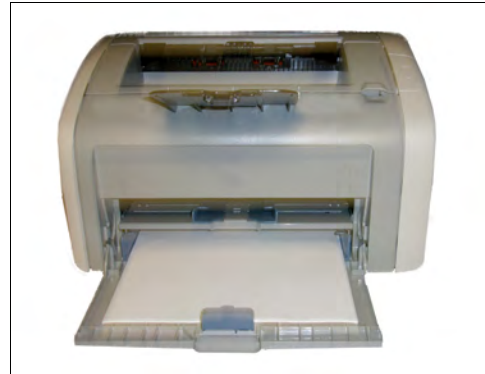
(Sources: Auter, 1998; Chase, 1999; Cringely, 1996; Mesa, 1998; Tripod, 1994)

Printers are an important peripheral device on larger digital systems like personal computers. They are an increasingly popular accessory for PDAs and cell phones, particularly those that offer fax capabilities.

Laser printers use technology similar to photocopiers and are very fast. An entire page is produced at one time, including both text and graphics (see Figure 3.11). Laser printers provide typeset-quality characters and are the standard type of printer in use today.

Because they do not use “impact” methods to form letters, laser printers cannot print multiple-copy forms like checks. But laser printers can print pictures (black and white or color) with nearly photographic quality. Most laser printers do only black and white and are reasonably inexpensive. Color laser printers can be very expensive but produce very good reproductions of just about any image or page.

Figure 3.11 Laser Printer



Ink-Jet printers are generally more compact than laser printers (see Figure 3.12). To form letters and images, they squirt small dots of ink through as many as 50 pin-like jets, each smaller than a human hair, arranged in a matrix (Lenert & Jordache, 1996). Most ink-jet printers have four ink cartridges—cyan, yellow, magenta, and black—and are called *CYMK printers* as a result.

As the ink permeates the paper and spreads out slightly, a fully-formed letter results. Most ink-jet printers are slower than laser printers, but they produce images that are nearly as good and are much less expensive. Color ink-jet printers are very popular because they are relatively inexpensive and have high enough resolution to produce nearly photo-quality images when printing on special photographic papers.

Figure 3.12 Ink Jet Printer



DigiTrends

<u>Year</u>	<u>Printer Type</u>	<u>Developer(s)</u>	<u>Explanation</u>	1977
	Laser Printer	Xerox	First laser printers cost \$350,000	
1978	MX-80 dot matrix	Epson	First dot matrix printer for PCs	
1984	HP LaserJet	Hewlett-Packard	First HP LaserJet printer	
1986	LaserWriter Plus	Apple	First Apple laser printer	
1990	HP InkJet	Hewlett-Packard	First HP InkJet printer	
1993	QMS Colorscript	QMS	First color laser printer	
1994	Xerox 9400	Xerox	First color laser printer by Xerox	
1994	LaserJet 4	Hewlett-Packard	First 600dpi laser printer for PCs	
1995	HP Color LaserJet	Hewlett-Packard	First HP color laser jet printer	
1997	Xerox C55	Xerox	Prices dropped to about \$3,000	

(Sources: Chase, 1999; Lenert & Jordache, 1998)

Printer Developments

Modems: Long before telephone lines were capable of digital transmissions, computers used modems (see Figure 3.13) to transmit digital data. Modems convert the 0s and 1s of digital data into audible tones that can be sent down analog phone lines, then re-convert those tones into 0s and 1s at the receiving end. This process, called modulation/demodulation, is where modems got their name—modem is short for modulate/demodulate. The Internet has made modems an essential part of modern computer systems and many hand-held devices as well.

Figure 3.13 External Modem



The first modems were essentially two cups into which a phone receiver was placed. These *acoustic couplers* were very slow by modern standards—only 300 *baud* (bits-per-second, or bps). Eventually, modems became internal devices and were increasingly faster, achieving speeds of 14,400 bps, 28,800 bps, 33,600 bps, and 56,000 bps along the way. Today, broadband services like telephone DSL lines use modems with speeds of 640,000 bps or more. Since the modem at the sending end may not match the speed of the one at the receiving end, the transmission speed can be no more than the slowest of the two connected modems. To decide what that speed will be, modems engage in *handshaking* when they first connect. This process, which sounds like a series of screeches, allows differently-abled modems to negotiate the speed and protocol for their connection.

DigiTrends

Modem Developments

<u>Year</u>	<u>Modem Speed</u>	<u>Developer(s)</u>	<u>Model Name/Explanation</u>
1979	300 baud (bps)	Hayes	Micro Modem; Used acoustic couplers
1984	1200 baud (bps)	Hayes	Used add-on card technology
1987	9600 K bps	U.S. Robotics	HST model greatly enhanced speeds
1993	14.4 K bps	Hayes	First with more than 10,000K speed
1994	28.8 K bps	Hayes	Most modems operate at 28.8K speed
1995	33.6 K bps	U.S. Robotics	Highest reliable speed on phone lines
1996	56.0 K bps	U.S. Robotics	Few connections actually reach this speed
1997	128 K bps	ISDN phone lines	Available since 1987; Faster speeds now
1998	40 M bps	Cable modems	Speeds vary; This was maximum speed
1998	56 M bps	DSL phone lines	Speeds vary; This was maximum speed
Today:	Twisted copper pairs	64 K bps	Maximum speed of standard phone lines
	ISDN phone lines	128 K bps	Maximum speed of ISDN phone lines
	Coaxial cables	160 M bps	Maximum speed of coaxial cable
	Fiber Optics	160 G bps	Capacity of a single fiber optic strand

(Sources: Chase, 1999; Berquist, 1998, 2000)

Sound Systems: The advent of the Internet has promoted *multimedia*— the combined use of many different sensory experiences in a single program or display. Multimedia content includes the use of sound, making sound capability an essential part of modern digital systems. Today, digital systems play sounds from a variety of sources: (1) CD files, (2) *wav files* created by sampling analog sound waves up to 44,100 times per second, and (3) *midi files* which contain notes (like those on sheet music) that are played by digitally simulated instruments. Recently, streaming sound files on the Internet have permitted the reception of radio stations and other live audio content, using compressed audio sources such as *MP3 files*.

DigiTrends

Audio Developments

<u>Year</u>	<u>Development</u>	<u>Developer(s)</u>	<u>Explanation</u>
1982	Compact Discs	Sony and Philips	Introduced as alternative to tapes
1985	MIDI sound	Atari	Non-wave based sound encoding
1990	Digital Audiotape (DAT)	Sony	First introduced to United States
1991	SoundBlaster Pro Deluxe	Creative Labs	First 8-bit PC stereo sound card
1992	Digital Compact Cassettes	Philips	Proposed DAT tech on cassettes
1992	Minidiscs	Sony	A cross between CDs and diskettes
1993	SoundBlaster Pro 16	Creative Labs	First 16-bit PC stereo sound card
1996	SoundBlaster Pro 32	Creative Labs	First 32-bit PC stereo sound card
1998	SoundBlaster Pro 64	Creative Labs	First 64-bit PC stereo sound card
1998	QSound QS7777 chip	Various	3-D sound tech for surround sound
1999	CD-R and CD-RW	Various	Writable and Re-writable CDs
1999	DVD audio	Various	2 hours surround or 4 hours stereo
1999	MP3 players	Diamond Rio	High-quality compressed audio
2000s	iPod players	Apple	Extremely popular mobile players

(Sources: Chase, 1999; Roussell, 1998; Carlin, 2000)

Scanners: Multimedia has also increased the importance of graphics. An essential part of graphic design is a *scanner*, usually a flatbed device that scans both images and text placed on its glass surface (see Figure 3.14). Images are converted into any one of several graphic file formats (see Chapter 2). These images can later be retrieved, displayed, printed, edited, or transported to another user.

Figure 3.14 Flatbed Scanner



The scanning of text is called *OCR (Optical Character Recognition)*. The system makes an image of the printed page and then uses sophisticated comparisons to determine what letter or symbol each character on the page represents. In this way, a printed page can be converted into a data file consisting of the actual words that were printed on the page previously.

CHAPTER SUMMARY

Most modern digital devices act like, and are in fact, computers. To understand trends in modern digital media, one must understand the basic internal components of digital devices. Internal components include system boards, memory chips, internal connections, and disk drives. The main system board is the mother board, which lies inside the CPU and controls all other functions and devices. The main component of the system board is the processor, which is the actual “brain” of the computer. Processors have increased significantly in speed and capacity over the years. Memory—both ROM and RAM—has also increased to meet the ever-increasing demands of software programs.

The connections between components are called the bus. The earliest of these was the serial bus, followed by the parallel bus. The SCSI bus allowed multiple devices to be plugged into a single port. The PCMCIA bus is small and flat and therefore very useful for laptop computers. The most recent standard is the USB bus, which allows multiple devices just as the SCSI bus does, but does not require re-setting the computer every time a device is plugged in.

Disk drives are of two types: removable and nonremovable. Fixed disks are not removable but are much faster and have higher capacities than any removable disk. Current fixed disks hold gigabytes of information. Removable disks include diskettes, Zip disks, CDs, and DVDs.

Both hardware and application software need an operating system to do their work. There have been dramatic changes to operating systems, roughly corresponding to the increased power of processors and the increasing RAM demands of programs. Operating systems must be carefully matched to both the processor and the application software.

Peripheral devices have significantly improved in the last 20 years. Displays, printers, modems, sound systems, and scanners have vastly increased the utility of all digital systems, from palm sized PDAs to desktop computers. Today, the demands of multimedia require the sophisticated use of all these devices and capabilities, and most digital devices make use of two or more of them at the same time.

IMPORTANT TERMS AND CONCEPTS

(See full definitions in the Glossary, Appendix A)

Concept	Page	Concept	Page	Concept	Page
PDAs	42	serial buses.	48	peripherals.....	52
processors.	42	parallel buses.	48	displays.....	53
operating system....	42	SCSI buses.	48	monochrome.....	53
open architecture. ...	43	PCMCIA.....	48	LCD.....	53
hardware.	44	USB.....	49	full-color.....	53
CPU.....	44	fixed (hard) disks....	49	laser printers.....	54
ports.....	44	diskettes.	49	ink-jet printers.....	54
system board.....	44	ZIP disks.	49	CYMK printers.	54
mother board.	44	CDs.	49	modems.	55
processor.	44	CD-ROM.....	49	acoustic couplers....	55
processor bit-value... 45		DVD.....	50	baud.	55
cycling speed.....	45	software.	51	handshaking.	55
megahertz (MHz)....	45	computer languages..	51	multimedia.....	56
interrupts.....	46	application software..	51	wav files.	56
system memory.....	46	content.	51	midi files.....	56
ROM.	46	GUI.....	52	scanners.....	57
RAM.	46	user friendliness.	52	OCR.....	57
bus.	47	machine language. ...	52		

ACTIVITIES & DISCUSSIONS

The following are potential topics for a paper or class discussion.

DigiTrends:

- **Increasing Disk Capacities** (p.50): Visit a web site that sells computer disk drives. What is the smallest fixed disk available at the site? What is the largest? Are there significant differences in the cost of these drives, based on their size? Where does the largest jump in price occur (between what two sizes)? Now visit at least one major computer vendor on the web (e.g., Dell, Gateway, etc.) What is the smallest and the largest amount of disk drive capacity featured on their systems? What do you conclude about disk storage from all these data? Discuss your findings with the class.
- **Display Developments** (p.53): Visit a local electronics or computer store and investigate the types of displays used by digital devices today. Include in your analysis at least one from each of the following categories: desktop computers, laptop computers, PDAs, cell phones, portable CD or MP3 players. What types of displays do each of these use? Which devices offer more than one type of display? What are the differences in cost when more than one display is available? For what reason would somebody adopt the more expensive display—that is, what additional functionality (if any) does it provide? Discuss your findings with the class.
- **Printer Developments** (p.55): Visit a local computer or office products store and investigate the types of printers available today. Count how many different printers are available in each of the following categories: dot-matrix, ink-jet, black and white laser, color laser. Choose one popular model in each category and compare the page per minute (ppm) speed of that printer with the one you selected in the other categories. Also compare their prices. Which printer would you select for black and white printing, and why? Which printer for color, and why? Discuss your findings with the class.
- **Modem Developments** (p.56): Contact your local cable television provider and investigate cable modems. How many bits per second (bps) do they transmit, and what do they cost (both to acquire and on a monthly basis)? Contact your local telephone company and obtain the same information for DSL (digital subscriber line) modems. How do they compare in terms of cost and speed? How do they compare to the 56K bps modems that many home

computers now use, both in terms of cost and speed? What additional features do they offer? Discuss your findings with the class.

- **Audio Developments** (p.56): Search the Internet for information on MP3 music files and players. How long (in terms of minutes) is a typical song? How large (in terms of KB or MB) is a typical file for that song? Since a CD offers 72 minutes of music in approximately 650 MB, it takes about 9 MB to store a minute's worth of music on a CD. How do MP3 files compare in terms of MB or KB per minute? Listen to an MP3 song and determine whether the quality of the music is less than on a CD. Discuss your findings with the class.

STUDY QUESTIONS

1. What are bits, bytes, megabytes, and gigabytes? When and how are these measurements used when comparing features of digital devices?
2. What is open architecture, and how does it explain why PCs have enjoyed greater market success than Apples? In what ways have Apples excelled over PCs through the years?
3. What (in terms of bits and typical megahertz) are the differences between an 80486 and a Pentium processor? What is the effect of these differences?
4. What is RAM and how does a computer use it? How does it relate to computer speed?
5. What is a "bus," and how does this relate to processors and their computing power?
6. What kinds of buses are there, and how are (or were) they typically used?
7. How does the operating system relate to the bit-value of the processor? How do both of these relate to the design of application software?
8. In what ways have digital devices changed since the 1970s? At what level(s) of the system have these changes occurred?
9. What are the most common peripheral devices for digital devices? What are the most common types of each of them?

10. How do laser printers and ink jet printers compare to each other? Which is the most cost effective in printing color?
11. What is an acoustic coupler and how does it compare to modern modems? What is handshaking, and why is it necessary between modems?

CHAPTER REFERENCES

- Auter, P.J. (1998). Personal communication: Pagers, palmtops, and PDAs. In A.E. Grant & J.H. Meadows (Eds.) Communication technology update (6th ed.) (pp. 263-268). Boston, MA: Focal Press in association with Technology Futures, Inc.
- Berquist, L. (1998). Broadband networks. In A.E. Grant & J.H. Meadows (Eds.) Communication technology update (6th ed.) (pp. 213-222). Boston, MA: Focal Press in association with Technology Futures, Inc.
- Berquist, L. (2000). Broadband networks. In A.E. Grant & J.H. Meadows (Eds.) Communication technology update (7th ed.) (pp. 223-231). Boston, MA: Focal Press in association with Technology Futures, Inc.
- Boyce, P.J. (1998). Microsoft Windows 98: User manual. Indianapolis, Indiana: Que, a division of Macmillan USA.
- Boyce, P.J., Tidrow, R., et al. (1995). Inside Windows 95. Indianapolis, Indiana: New Riders Publishing.
- Carlin, T. (2000). Digital audio. In A.E. Grant & J.H. Meadows (Eds.) Communication technology update (7th ed.) (pp. 189-206). Boston, MA: Focal Press in association with Technology Futures, Inc.
- Chase, R.S. (1999). "Computer literacy lab: An introductory lab about the internal and peripheral components of computer systems." Salt Lake City, Utah: Salt Lake Community College.
- Cringely, R.X. (1996). "A history of the computer: micro" and "A history of the computer: network." [<http://www.pbs.org/nerds/timeline/micro.html>].
- Fulton, J. (1998). The complete idiot's guide to upgrading and repairing PCs. Indianapolis, Indiana: Que, a division of Macmillan USA.
- Grant, A.E. (Ed.). (1994). Communication technology update (3rd ed.). Boston, MA: Focal Press in association with Technology Futures, Inc.

- Grant, A.E. (Ed.). (1996). Communication technology update (5th ed.). Boston, MA: Focal Press in association with Technology Futures, Inc.
- Grant, A.E., & Meadows, J.H. (Eds.). (2000). Communication technology update (7th ed.). Boston, MA: Focal Press in association with Technology Futures, Inc.
- Grant, A.E. & Meadows, J.H. (1998). Communication technology update (6th ed.) (pp. 1-6). Boston, MA: Focal Press in association with Technology Futures, Inc.
- Grant, A.E., & Sung, L. (Eds.). (1992). Communication technology update. Austin, TX: Technology Futures, Inc.
- Grant, A.E., & Wilkinson, K.T. (Eds.). (1993). Communication technology update: 1993-1994. Austin, TX: Technology Futures, Inc.
- Kraynak, J. (1998). The complete idiot's guide to PCs, 6th Edition. Indianapolis, Indiana: Que, a division of Macmillan USA.
- Lenert & Jordache (1998). Document printing technologies. In A.E. Grant & J.H. Meadows (Eds.) Communication technology update (6th ed.) (pp. 142-150). Boston, MA: Focal Press in association with Technology Futures, Inc.
- Livingston, B. & Straub, D. (1995). Windows 95 Secrets, 3rd Edition. Foster City, CA: IDG Books.
- Martin, D.R. (1998). Cable distributed telephony and data services. In A.E. Grant & J.H. Meadows (Eds.) Communication technology update (6th ed.) (pp. 242-247). Boston, MA: Focal Press in association with Technology Futures, Inc.
- Mesa, A. (1998). "Apple history timeline: What happened when, in summary."
[<http://www.mackido.com/History/AppleTimeline.html>]
- Microsoft (1996). Getting results with Microsoft Office for Windows 95, 7th Edition. Bellevue, WA: Microsoft Corporation.
- Risely, D. & Dockter, M.A. (2000). "A CPU history."
[<http://www.pcmach.com/article.htm?cpuhistory>].
- Roussell, J.R. (1998). Digital audio. In A.E. Grant & J.H. Meadows (Eds.) Communication technology update (6th ed.) (pp. 190-196). Boston, MA: Focal Press in association with Technology Futures, Inc.
- Tripod.com (1994). "History of computers: 1920s to 1990s."
[http://members.tripod.com/visionary_enterprises/History1.html]

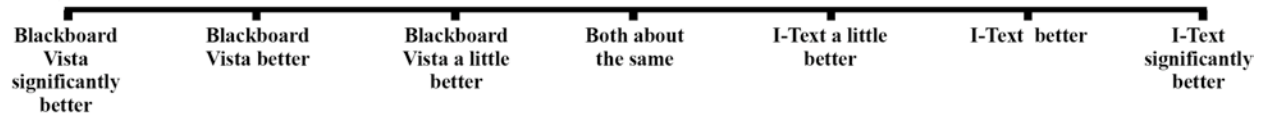




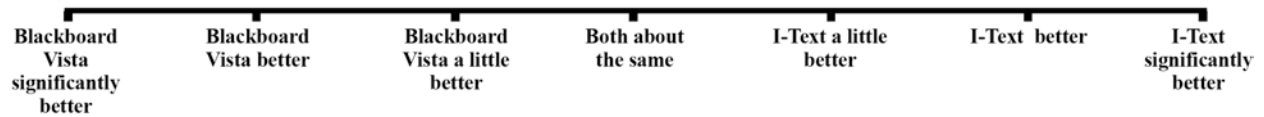
APPENDIX 2

QUESTIONNAIRE DISTRIBUTED TO STUDENTS

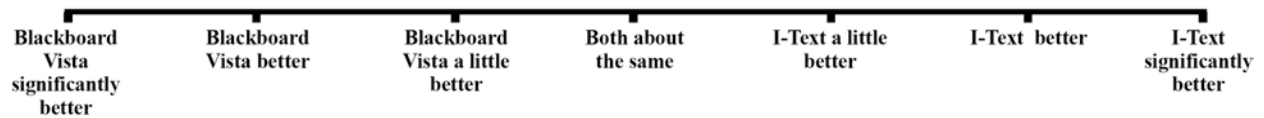
Which Distance Ed system was better at providing prompt feedback?



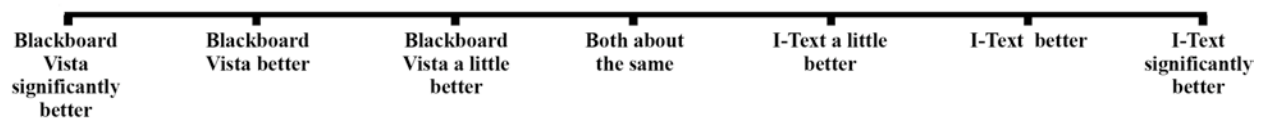
Which Distance Ed system was easier to navigate?



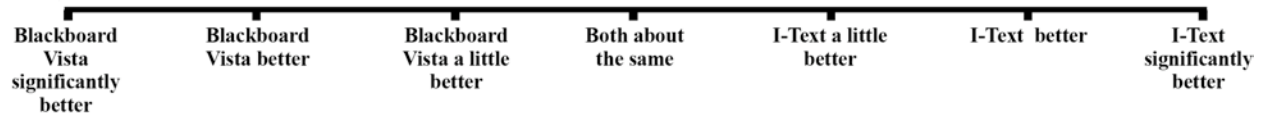
Which Distance Ed system made it easier to understand course requirements?



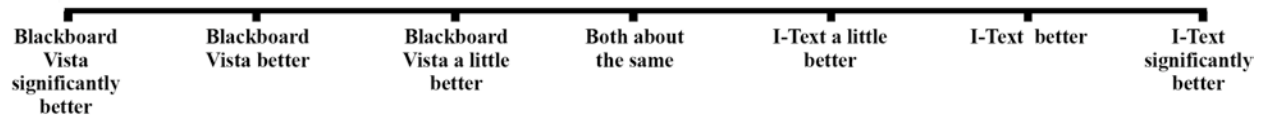
Which Distance Ed system was easier to use in general?



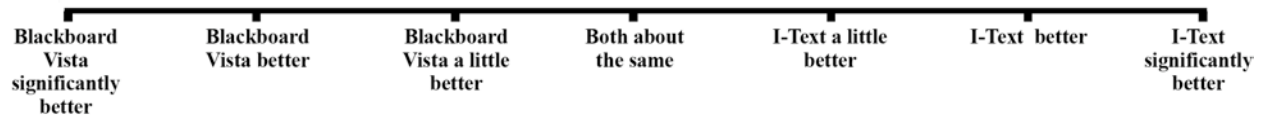
Which Distance Ed system was better at helping you understand course expectations?



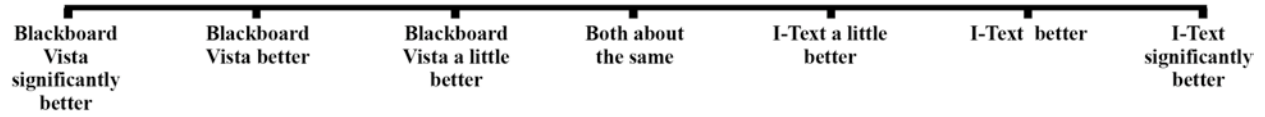
Which Distance Ed system was better?



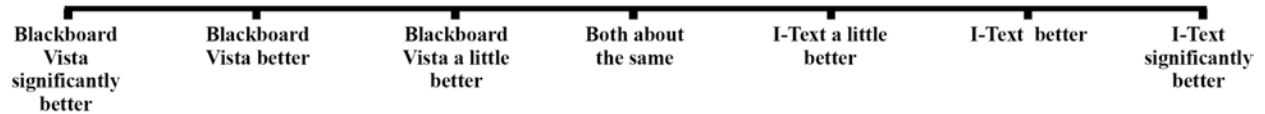
Which Distance Ed system was better at guiding you through the requirements of each chapter?



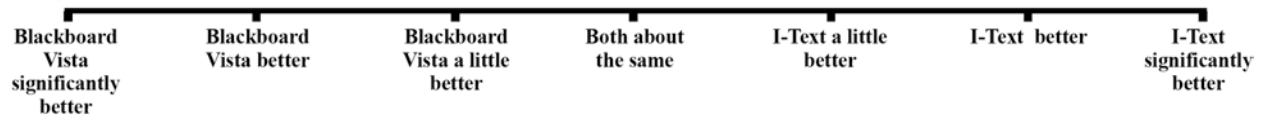
Which Distance Ed system helped you to communicate better with your instructor?



Which Distance Ed system was better at helping you feel a part of the class?



Which Distance Ed system was better at helping you to be successful in the course?



If you had the choice, which Distance Ed system would you prefer in the future? (Circle one)

Blackboard Vista

Both about the same

I-Text

Please provide comments regarding how you felt about each of the Distance Ed systems used during this semester:

APPENDIX 3

SOURCE CODE FOR THE I-TEXT AND RELATED CONSOLES

The following is source code for each of the RIA's included as part of the I-Text system. Unless noted, the source code is written in Actionscript 3.0 and was compiled in Flash CS3.

Communication Console:

```

import flash.net.URLLoader;
import flash.net.URLVariables;
import flash.net.URLRequest;
import flash.net.URLRequestMethod;
import flash.net.URLLoaderDataFormat;
import flash.net.sendToURL;
import flash.events.*;
import flash.display.Bitmap;
import flash.display.Stage;
import fl.controls.Button;

var extUserName           :String;
var extCourseName        :String;
var mainURL              :String = "http://www.advancedcognition.com/";
var commVarSend          :URLRequest = new
URLRequest(mainURL+"checkavailability.php");
var commVarLoader        :URLLoader = new URLLoader;
var phoneAvail           :Sprite = new Sprite();
var phoneAvailLoader     :Loader = new Loader();
var phoneLoader          :Loader = new Loader();
var chatLoader           :Loader = new Loader();
var vcLoader             :Loader = new Loader();
var phoneNotLoader       :Loader = new Loader();
var chatNotLoader        :Loader = new Loader();
var vcNotLoader          :Loader = new Loader();
var messageLoader        :Loader = new Loader();
var messageNotLoader     :Loader = new Loader();
var currentChatAvailability :String;
var currentPhoneAvailability :String;
var currentVcAvailability  :String;
var currentMessageStatus  :int;
var url                  :String = "";
var urlRequest           :URLRequest = new URLRequest();
//beginCommunication();

ExternalInterface.addCallback("authenticationHandler", authenticationHandler);

//authenticationHandler();
url = mainURL + "images/phoneAvailPhoto.jpg";
urlRequest.url = url;
phoneLoader.load(urlRequest);
phoneLoader.y = 50;

```

```

url = mainURL + "images/chatAvailPhoto.jpg";
urlRequest.url = url;
chatLoader.load(urlRequest);
chatLoader.y = 180;

url = mainURL + "images/vcAvailPhoto.jpg";
urlRequest.url = url;
vcLoader.load(urlRequest);
vcLoader.y = 310;

url = mainURL + "images/newMessages.jpg";
urlRequest.url = url;
messageLoader.load(urlRequest);
messageLoader.y = 440;

url = mainURL + "images/phoneNotAvailPhoto.jpg";
urlRequest.url = url;
phoneNotLoader.load(urlRequest);
phoneNotLoader.y = 50;

url = mainURL + "images/chatNotAvailPhoto.jpg";
urlRequest.url = url;
chatNotLoader.load(urlRequest);
chatNotLoader.y = 180;

url = mainURL + "images/vcNotAvailPhoto.jpg";
urlRequest.url = url;
vcNotLoader.load(urlRequest);
vcNotLoader.y = 310;

url = mainURL + "images/noMessages.jpg";
urlRequest.url = url;
messageNotLoader.load(urlRequest);
messageNotLoader.y = 440;

//var chatButton:Button = new Button();
chatButton.label = "Start Chat";
chatButton.move(4, 280);
chatButton.addEventListener(MouseEvent.CLICK, chatButtonClickHandler);
//var vcButton:Button = new Button();
vcButton.label = "Skype Me";
vcButton.move(4, 409.9);
vcButton.addEventListener(MouseEvent.CLICK, vcButtonClickHandler);

messageButton.label = "New Message(s)";
messageButton.move(4, 540);
messageButton.addEventListener(MouseEvent.CLICK, messageButtonClickHandler);

newMessageButton.label = "Create Message";
newMessageButton.move(4, 570);
newMessageButton.addEventListener(MouseEvent.CLICK, newMessageButtonClickHandler);

function authenticationHandler(){

    titleText.text = "Instructor\nAvailability";

```



```

ExternalInterface.call("eval", "var AM = this.getField('AuthMessage');");
ExternalInterface.call("eval", "AM.textColor = color.red;");
ExternalInterface.call("eval", "AM.value = 'Working, please wait. . .';");
ExternalInterface.call("eval", "var CN = this.getField('course');");
extCourseName = ExternalInterface.call("eval", "CN.value;");
ExternalInterface.call("eval", "var UN = this.getField('UserName');");
extUserName = ExternalInterface.call("eval", "UN.value;");

if(extUserName != ""){

    var variables2:URLVariables = new URLVariables();
    var varSend2:URLRequest = new URLRequest(mainURL+"checkUN.php");
    var varLoader2:URLLoader = new URLLoader;
    varLoader2.addEventListener(Event.COMPLETE, completeHandler2);

    function completeHandler2(event:Event):void {
        //Now check to see if UN authenticated
        //tb.text = "Got Here";
        var authResult:String = String(varLoader2.data["authenticate"]);
        if(authResult == ""){
            authResult = "None";
        }
        //authResult = "all";
        //tb.text = authResult;
        //tb.text = "Hi";
        ExternalInterface.call("eval", "var AM = this.getField('AuthMessage');");
var AL = this.getField('AuthLevel');");
        ExternalInterface.call("eval", "AM.textColor = color.blue;");
        switch(authResult){
            case "None":
                ExternalInterface.call("eval", "AM.textColor =
color.red;");
                ExternalInterface.call("eval", "AM.value = 'Your
username is invalid';AL.value = 'none;");
                break;
            case "all":
                ExternalInterface.call("eval", "AM.textColor =
color.blue;");
                ExternalInterface.call("eval", "AM.value = 'Success!!
You have access to all components';AL.value = 'all;");
                beginCommunication();
                break;
            case "video":
                ExternalInterface.call("eval", "AM.textColor =
color.blue;");
                ExternalInterface.call("eval", "AM.value = 'Success!!
You have access to video components only';AL.value = 'video;");
                break;
            case "quiz":
                ExternalInterface.call("eval", "AM.textColor =
color.blue;");
                ExternalInterface.call("eval", "AM.value = 'Success!!
You have access to all quizzes';AL.value = 'quiz;");
                break;
        }
    }
}

```

```

    }
    varSend2.method = URLRequestMethod.POST;
    varLoader2.dataFormat = URLLoaderDataFormat.VARIABLES;
    varSend2.data = variables2;
    variables2.UN = extUserName;
    variables2.CN = extCourseName;
    varLoader2.load(varSend2);

    }else{
        ExternalInterface.call("eval", "AM.textColor = color.red;");
        ExternalInterface.call("eval", "AM.value = 'Please enter a username';AL.value =
'none';");
        //tb.text = "Enter Username";
    }
}

function beginCommunication():void{

    var commVariables:URLVariables = new URLVariables();
    ////////////////////////////////////Change for Acrobat
    //extUserName = "daboncanplay";
    //extCourseName = "Comm 1250";
    ////////////////////////////////////
    addChild(phoneLoader);
    addChild(chatLoader);
    addChild(vcLoader);
    addChild(phoneNotLoader);
    addChild(chatNotLoader);
    addChild(vcNotLoader);
    addChild(chatButton);
    addChild(vcButton);
    addChild(messageLoader);
    addChild(messageNotLoader);
    addChild(messageButton);
    addChild(newMessageButton);
    newMessageButton.visible = true;
    commVarLoader.addEventListener(Event.COMPLETE, commVarcompleteHandler);
    commVarSend.method = URLRequestMethod.POST;
    commVarLoader.dataFormat = URLLoaderDataFormat.VARIABLES;
    commVarSend.data = commVariables;
    commVariables.UN = extUserName;
    commVariables.CN = extCourseName;

    var countdownTimer:Timer = new Timer(3000);
    countdownTimer.addEventListener(TimerEvent.TIMER, updateTime);
    countdownTimer.start();
    commVarLoader.load(commVarSend);

}

function commVarcompleteHandler(event:Event){

```

```

if(currentPhoneAvailability != String(commVarLoader.data["phoneavailability"])){

    if(String(commVarLoader.data["phoneavailability"]) == "yes"){

        phoneLoader.visible = true;
        phoneNotLoader.visible = false;
    }else{

        phoneNotLoader.visible = true;
        phoneLoader.visible = false;
    }
    currentPhoneAvailability = String(commVarLoader.data["phoneavailability"])
}
if(currentChatAvailability != String(commVarLoader.data["chatavailability"])){
    if(String(commVarLoader.data["chatavailability"]) == "yes"){
        chatLoader.visible = true;
        chatButton.visible = true;
        chatNotLoader.visible = false;

        //trace(currentAvailability);
    }else{
        chatNotLoader.visible = true;
        chatButton.visible = false;
        chatLoader.visible = false;

    }
    currentChatAvailability = String(commVarLoader.data["chatavailability"])
}
if(currentVcAvailability != String(commVarLoader.data["vcavailability"])){
    if(String(commVarLoader.data["vcavailability"]) == "yes"){
        vcLoader.visible = true;
        vcButton.visible = true;
        vcNotLoader.visible = false;

        //trace(currentAvailability);
    }else{
        vcNotLoader.visible = true;
        vcButton.visible = false;
        vcLoader.visible = false;

    }
    currentVcAvailability = String(commVarLoader.data["vcavailability"])
}
if(currentMessageStatus != int(commVarLoader.data["unreadMessages"])){
    if(int(commVarLoader.data["unreadMessages"]) > 0){
        messageLoader.visible = true;
        messageButton.visible = true;
        messageNotLoader.visible = false;

        //trace(currentAvailability);
    }else{
        messageNotLoader.visible = true;
        messageButton.visible = false;
    }
}

```

```

        messageLoader.visible = false;

        }
        currentMessageStatus = int(commVarLoader.data["unreadMessages"])
    }
    //trace(commVarLoader.data["unreadMessages"]);
}

function updateTime(event:Event):void{

    commVarLoader.load(commVarSend);

}

function vcButtonClickHandler(event:MouseEvent):void
{
    //var request          :URLRequest = new URLRequest("skype:dabon?call");
    //navigateToURL(request);
    ExternalInterface.call("eval", "this.getURL('skype:professor_harris?call');");
}

function chatButtonClickHandler(event:MouseEvent):void
{
    ExternalInterface.call("eval", "this.pageNum = 2; var annot = this.getAnnotsRichMedia(2)[0];
if(!annot.activated) annot.activated = true;");

}

function messageButtonClickHandler(event:MouseEvent):void
{
    ExternalInterface.call("eval", "this.pageNum = 2; var annot = this.getAnnotsRichMedia(2)[1];
if(!annot.activated) annot.activated = true;");
    ExternalInterface.call("eval", "annot.callAS('getMessages');");

}

function newMessageButtonClickHandler(event:MouseEvent):void
{
    ExternalInterface.call("eval", "this.pageNum = 2; var annot = this.getAnnotsRichMedia(2)[1];
if(!annot.activated) annot.activated = true;");
    ExternalInterface.call("eval", "annot.callAS('createMessage');");

}

```

Syllabus Console:

```

import fl.controls.TextArea;
import flash.net.URLLoader;
import flash.net.URLVariables;
import flash.net.URLRequest;
import flash.net.URLRequestMethod;
import flash.net.URLLoaderDataFormat;

```

```

import flash.net.sendToURL;
import flash.text.TextFormat;
import flash.text.TextFieldAutoSize;
import fl.controls.Label;

var titleText          :Label = new Label();
var titleTextFormat    :TextFormat = new TextFormat();
//var dueDateText      :Label = new Label();
//var dueDateTextFormat :TextFormat = new TextFormat();
//var descriptionLabel  :Label = new Label();
//var descriptionLabelFormat :TextFormat = new TextFormat();
var syllabusBodyText   :TextArea = new TextArea();
var syllabusBodyTextFormat :TextFormat = new TextFormat();
var variables2         :URLVariables = new URLVariables();
var mainURL            :String = "http://www.advancedcognition.com/";
var varSend2          :URLRequest = new

URLRequest(mainURL+"getsyllabus.php");
var varLoader2        :URLLoader = new URLLoader;
var extUserName        :String;
//var extCourseName    :String;
var extCourseName      :String = root.loaderInfo.parameters.courseName;
titleText.text = "Loading Syllabus, Please Wait. . .";
titleText.width = 300;
addChild(titleText);
////////////////////change
//extCourseName = "Comm 1250";
////////////////////

/*
var monthLabels:Array = new Array("January",
    "February",
    "March",
    "April",
    "May",
    "June",
    "July",
    "August",
    "September",
    "October",
    "November",
    "December");
*/

//ExternalInterface.addCallback("getAssignment", getAssignment);

varLoader2.addEventListener(Event.COMPLETE, completeHandler2);
varSend2.method = URLRequestMethod.POST;
varLoader2.dataFormat = URLLoaderDataFormat.VARIABLES;
varSend2.data = variables2;

/*
ExternalInterface.call("eval", "var UN = this.getField('UserName');");
extUserName = ExternalInterface.call("eval", "UN.value;");

ExternalInterface.call("eval", "var CN = this.getField('course');");

```

```

extCourseName = ExternalInterface.call("eval", "CN.value;");
//////////Change
extUserName = "daboncanplay";
extCourseName = "Comm 1250";
assignmentNumber = "1";
//////////
*/
//variables2.UN = extUserName;
variables2.CN = extCourseName;
//variables2.assignmentNumber = assignmentNumber;
//trace("assignNum="+extUserName);
varLoader2.load(varSend2);
//messageText.text = "\n\nLoading Messages. . .";

function completeHandler2(event:Event):void
{

    titleTextFormat.size = 20;
    titleText.setStyle("textFormat", titleTextFormat);
    titleText.autoSize = TextFieldAutoSize.CENTER;
    titleText.y = 10;
    titleText.width = 800;
    titleText.text = varLoader2.data['courseTitle'] + " -- Syllabus";

    /*
    dueDateTextFormat.color = "0xFF0000";
    dueDateTextFormat.size = 14;
    dueDateText.setStyle("textFormat", dueDateTextFormat);
    dueDateText.x = 660;
    dueDateText.y = 40;
    dueDateText.autoSize = TextFieldAutoSize.RIGHT;
    dueDateText.text = "Due: " + formattedDateTime;
    addChild(dueDateText);

    descriptionLabelFormat.size = 14;
    descriptionLabel.setStyle("textFormat", descriptionLabelFormat);
    descriptionLabel.move(20, 40);
    descriptionLabel.text = "Description";
    addChild(descriptionLabel);
    */
    syllabusBodyTextFormat.size = 14;
    syllabusBodyTextFormat.font = "Arial";
    syllabusBodyText.setStyle("textFormat", syllabusBodyTextFormat);
    syllabusBodyText.x = 10;
    syllabusBodyText.y = 60;
    syllabusBodyText.width = 780;
    syllabusBodyText.height = 700;
    syllabusBodyText.editable = false;
    syllabusBodyText.text = varLoader2.data['syllabus'];
    //trace(varLoader2.data);
    addChild(syllabusBodyText);

}

```

Chat Console:

Note: The Chat Console utilized a chat application that was external to the I-Text system.

```
import flash.display.Loader;
import flash.net.URLRequest;

Security.allowDomain("*")

//_root.documentRoot = "http://www.advancedcognition.com";

var url                :String =
"http://www.advancedcognition.com/flashchat/preloader.swf";
var loader              :Loader = new Loader();
var request              :URLRequest = new URLRequest(url);

loader.load(request);
loader.addEventListener(Event.COMPLETE, completeHandler);
addChild(loader);

function completeHandler(event:Event):void{

}
```

Chat Console:

```
import fl.controls.TextArea;
import flash.net.URLLoader;
import flash.net.URLVariables;
import flash.net.URLRequest;
import flash.net.URLRequestMethod;
import flash.net.URLLoaderDataFormat;
import flash.net.sendToURL;
import flash.text.TextFormat;
import fl.controls.Label;

var messageText        :TextArea = new TextArea();
var extUserName         :String = "";
var extCourseName       :String = "";
var mainURL             :String = "http://www.advancedcognition.com/";
var messageTextFormat   :TextFormat = new TextFormat();
var outline             :Outline = new Outline();
var newMessageSprite    :newMessageWindow;
var monthLabels:Array = new Array("January",
    "February",
    "March",
    "April",
    "May",
    "June",
    "July",
```

```

        "August",
        "September",
        "October",
        "November",
        "December");

ExternalInterface.addCallback("getMessages", getMessages);
ExternalInterface.addCallback("createMessage", createMessage);

ExternalInterface.call("eval", "var UN = this.getField('UserName');");
extUserName = ExternalInterface.call("eval", "UN.value;");
ExternalInterface.call("eval", "var CN = this.getField('course');");
extCourseName = ExternalInterface.call("eval", "CN.value;");

////////////////////////////////////////Change
//extUserName = "daboncanplay";
//extCourseName = "Comm 2500";
////////////////////////////////////////
newMessageSprite = new newMessageWindow();
newMessageSprite.x = 3;
newMessageSprite.y = 10;

addChild(outline);

var variables2:URLVariables = new URLVariables();
var varSend2:URLRequest = new URLRequest(mainURL+"getannouncements.php");
var varLoader2:URLLoader = new URLLoader;
varLoader2.addEventListener(Event.COMPLETE, completeHandler2);

function completeHandler2(event:Event):void {

    var totalMessages:int = Number(varLoader2.data["numberOfMessages"]);
    messageText.text = "\n\n";
    for(var Mindex = 1; Mindex <= totalMessages; Mindex++){
        var Mtmp = "message" + Mindex;
        var DTtmp = "timeDate" + Mindex;
        var Ftmp = "from" + Mindex;
        var sqlDT:String = varLoader2.data[DTtmp];
        var Myear:String = sqlDT.slice(0, 4);
        var Mmonth:String = sqlDT.slice(5, 7);
        var Mday:String = sqlDT.slice(8, 10);
        var MHour:int = int(sqlDT.slice(11, 13));
        var MMin:String = sqlDT.slice(14, 16);
        var ampm:String = "a.m.";
        if(int(MHour) > 12){
            MHour -= 12;
            ampm = "p.m.";
        }
        //trace(MHour);
        //trace(MMin);
        var formattedDateTime:String = monthLabels[int(Mmonth) - 1] + " " + int(Mday) + ", "
+ Myear + " - " + MHour + ":" + MMin + " " + ampm;

```



```

        //trace(formattedDateTime);

        messageText.appendText("\t");
        messageText.appendText(formattedDateTime + " -- FROM: " + varLoader2.data[Ftmp] +
" --> ");

        messageText.appendText(varLoader2.data[Mtmp]);
        messageText.appendText("\n\n");
    }

}

varSend2.method = URLRequestMethod.POST;
varLoader2.dataFormat = URLLoaderDataFormat.VARIABLES;
varSend2.data = variables2;

function getMessages():void{
    newMessageSprite.visible = false;
    ExternalInterface.call("eval", "var UN = this.getField('UserName');");
    extUserName = ExternalInterface.call("eval", "UN.value;");

    ExternalInterface.call("eval", "var CN = this.getField('course');");
    extCourseName = ExternalInterface.call("eval", "CN.value;");
    variables2.UN = extUserName;
    variables2.CN = extCourseName;
    varLoader2.load(varSend2);
    messageText.text = "\n\n\tLoading Messages. . .";
}

messageText.y = 5;
messageText.width = 797;
messageText.height = 291;
//messageText.text = "This is the message.This is the message.This is the message.This is the message.This
is the message.This is the message.This is the message.This is the message.This is the message.This is the
message.This is the message.";
messageTextFormat.size = 14;
messageTextFormat.color = "0x0000FF";
messageText.setStyle("textFormat", messageTextFormat);
messageText.setStyle("upSkin",Sprite);
messageText.setStyle("focusRectSkin",Sprite);
messageText.editable = false;
addChild(messageText);
newMessageSprite.visible = false;
addChild(newMessageSprite);
//getMessages();
//createMessage();

function createMessage():void{

    ExternalInterface.call("eval", "var UN = this.getField('UserName');");
    extUserName = ExternalInterface.call("eval", "UN.value;");

```

```

        ExternalInterface.call("eval", "var CN = this.getField('course');");
        extCourseName = ExternalInterface.call("eval", "CN.value;");
        newMessageSprite.doNewMessage("Jim", extCourseName);
        newMessageSprite.extUserName = extUserName;
        newMessageSprite.extCourseName = extCourseName;
        newMessageSprite.submitButton.label = "Send";
        newMessageSprite.submitButton.enabled = true;
        newMessageSprite.visible = true;

    }

```

Live Console:

Note: The Live Console utilized a FLV playback component found within the Flash authoring environment.

```
myFLVPlayback.isLive=true;
```

Video Console:

```

//import flash.external.ExternalInterface;

play_btn.visible = false;
stop_btn.visible = false;
seek_bar.visible = false;
volume_bar.visible = false;
mute_btn.visible = false;

import flash.events.*;
import fl.video.*;

var extAuthLevel      :String;
var videoFile         :String = root.loaderInfo.parameters.videoFile;
var mainURL           :String = "http://www.advancedcognition.com/";

ExternalInterface.addCallback("playClickHandler", playClickHandler);

//load_btn.addEventListener(MouseEvent.CLICK, playClickHandler);
flvDisplay.addEventListener(VideoEvent.COMPLETE, completeHandler);
//playClickHandler();
function playClickHandler(){

    //tb.text = "Received";

```

```

if(ExternalInterface.available){
    ExternalInterface.call("eval", "var UN = this.getField('AuthLevel');");
    extAuthLevel = ExternalInterface.call("eval", "UN.value;");
    //extUserName = "Howdy Doody";

    //Now check to see if UN authenticated
    //tb.text = "Got Here";

    play_btn.visible = true;
    stop_btn.visible = true;
    seek_bar.visible = true;
    volume_bar.visible = true;
    mute_btn.visible = true;
    flvDisplay.playPauseButton = play_btn;
    flvDisplay.stopButton = stop_btn;
    flvDisplay.seekBar = seek_bar;
    flvDisplay.volumeBar = volume_bar;
    flvDisplay.muteButton = mute_btn;
    //tb.text = authResult;
    //tb.text = "Hi";
    flvDisplay.source =
"http://www.advancedcognition.com/video/"+videoFile;

}

}

function completeHandler(event:VideoEvent):void
{

    flvDisplay.source = null;

}

```

Quiz Console:

```

import fl.containers.ScrollPane;
import fl.controls.ScrollPolicy;
import fl.controls.Button;
import fl.controls.TextArea;
import flash.net.URLLoader;
import flash.net.URLVariables;
import flash.net.URLRequest;
import flash.net.URLRequestMethod;
import flash.net.URLLoaderDataFormat;
import flash.net.sendToURL;
import flash.text.TextFormat;

```

```

import fl.controls.Label;
import flash.display.Graphics;
import flash.display.Shape;

var myScrollPane
var heightOffset
var questions
var submit
var xmlPath
root.loaderInfo.parameters.quizFileName;
var numberOfQuestions
var xmlData
var loader
var line
var answerArray
var instructionsText
var insLabel
var tf
var areYouSureSubmit
var areYouSureCancel
var mainURL
var extAuthLevel
var extUserName
var extCourseName
var quizNumber
var quizVersion
var attemptsTotal
var attemptsSoFar
var attemptsTextBox
var currentAttempt
var attemptsRemaining
var quizID
var studentID
var currentAttemptID
var varLoader3
var varLoader4
var time_txt
var then
var timeLimit
var time_txtFormat
//playClickHandler();

:ScrollPane = new ScrollPane();
:int = 0;
:Sprite = new Sprite();
:Button = new Button();
:String = "http://www.advancedcognition.com/" +
:String;
:XML;
:URLLoader = new URLLoader();
:Shape = new Shape();
:Array = new Array();
:TextArea = new TextArea();
:TextField = new TextField();
:TextField = new TextField();
:Button = new Button();
:Button = new Button();
:String = "http://www.advancedcognition.com/";
:String;
:String;
:String;
:String = root.loaderInfo.parameters.quizNumber;
:String;
:int;
:int;
:TextField = new TextField();
:int;
:int;
:int;
:int;
:int;
:int;
:URLLoader = new URLLoader();
:URLLoader = new URLLoader();
:TextField = new TextField();
:Number;
:Number = 3600000;
:TextFormat = new TextFormat();

ExternalInterface.addCallback("playClickHandler", playClickHandler);

loader.addEventListener(Event.COMPLETE, onDataHandler);

var variables2:URLVariables = new URLVariables();
var varSend2:URLRequest = new URLRequest(mainURL+"updatestudentquiz.php");
var varLoader2:URLLoader = new URLLoader();
varLoader2.addEventListener(Event.COMPLETE, completeHandler2);

function completeHandler2(event:Event):void {
    //Now check to see if UN authenticated
    //tb.text = "Got Here";

```

```

    attemptsTotal = Number(varLoader2.data["totalAttempts"]);
    attemptsSoFar = Number(varLoader2.data["attemptsSoFar"]);
    quizVersion = varLoader2.data["version"];
    quizID = Number(varLoader2.data["quizID"]);
    studentID = Number(varLoader2.data["studentID"]);
    if(attemptsTotal > attemptsSoFar){
        currentAttempt = attemptsSoFar + 1;
        attemptsRemaining = attemptsTotal - currentAttempt;
        loader.load(new URLRequest(xmlPath + "?cachebuster=" + new Date().getTime()));
    }else{
        tf.text = "You are not allowed to attempt this quiz again.";
    }
}

varSend2.method = URLRequestMethod.POST;
varLoader2.dataFormat = URLLoaderDataFormat.VARIABLES;
varSend2.data = variables2;

function playClickHandler(){
    if(ExternalInterface.available){
        ExternalInterface.call("eval", "var AL = this.getField('AuthLevel');");
        ExternalInterface.call("eval", "var UN = this.getField('UserName');");
        extUserName = ExternalInterface.call("eval", "UN.value;");
        ExternalInterface.call("eval", "var CN = this.getField('course');");
        extCourseName = ExternalInterface.call("eval", "CN.value;");
        if(extUserName == null){
            tf.text = "You must enter your User Name on page 1.";
            tf.width = 300;
            addChild(tf);
        }else{
            var tempExtAuthLevel:String = ExternalInterface.call("eval", "AL.value;");
            switch(tempExtAuthLevel) {
                case "all":
                case "quiz":
                    tf.text = "Loading Quiz - Please Wait. . .";
                    tf.width = 300;
                    addChild(tf);
                    extAuthLevel = "Yes";
                    variables2.UN = extUserName;
                    variables2.CN = extCourseName;
                    variables2.quizNumber = quizNumber;
                    varLoader2.load(varSend2);

                    break;
                default:
                    tf.text = "You are not authorized to view this quiz.";
                    tf.width = 300;
                    addChild(tf);
                    break;
            }
        }
    }
}

function onDataHandler(event:Event):void

```

```

{
    if((event.target as URLLoader) != null )
    {
        // Save data
        xmlData = new XML(loader.data);
        //bannerAdsTotal = xmlData.bannerAds;
        //trace(bannerAdsTotal);
        numberOfQuestions = xmlData.question.length();
        timeLimit = xmlData.timeAllowed;
        var titleFormat:TextFormat = new TextFormat();
        titleFormat.size = 50;
        titleFormat.align = TextFormatAlign.CENTER;

        var attemptsFormat:TextFormat = new TextFormat();
        attemptsFormat.size = 16;
        attemptsFormat.color = 0xFF0000;
        attemptsTextBox.defaultTextFormat = attemptsFormat;
        attemptsTextBox.text = "Attempt: " + currentAttempt + "\nYou have " +
attemptsRemaining + " attempt(s) remaining.";
        attemptsTextBox.width = 220;
        attemptsTextBox.x = 10;
        attemptsTextBox.y = 10;
        addChild(attemptsTextBox);

        tf.x = 10;
        tf.y = 10;
        tf.width = 690;
        tf.defaultTextFormat = titleFormat;
        tf.text = xmlData.quizTitle;
        addChild(tf);

        insLabel.text = "Instructions:";
        insLabel.x = 10;
        insLabel.y = 75;
        addChild(insLabel);

        var instructionsFormat:TextFormat = new TextFormat();
        instructionsFormat.size = 14;
        instructionsFormat.color = 0xFF0000;
        instructionsText.htmlText = xmlData.instructions;
        instructionsText.editable = false;
        instructionsText.x = 20;
        instructionsText.y = 90;
        instructionsText.width = 680;
        instructionsText.height = 100;
        instructionsText.setStyle("textFormat", instructionsFormat);
        instructionsText.setStyle("upSkin",Sprite);
        instructionsText.setStyle("focusRectSkin",Sprite);
        addChild(instructionsText);
        line.graphics.lineStyle(1, 0xcccccc);
        line.graphics.moveTo(30, 200);
        line.graphics.lineTo(640, 200);
        addChild(line);
        // Set section buttons
        //trace(int(numberOfQuestions));
        for(var index:int = 0; index < int(numberOfQuestions);index++){

```

```

        var qr1:QuestionRenderer = new QuestionRenderer(index + 1,
xmlData.question[index].qType, xmlData.question[index].qContent, xmlData.question[index].multChoiceA,
xmlData.question[index].multChoiceB, xmlData.question[index].multChoiceC,
xmlData.question[index].multChoiceD, xmlData.question[index].multChoiceE,
xmlData.question[index].multChoiceF, xmlData.question[index].multChoiceG);
        qr1.name = "question" + index;
        qr1.y = heightOffset;
        questions.addChild(qr1);

        if(xmlData.question[index].qType == "True/False" ||
xmlData.question[index].qType == "Fill in the Blank"){
            heightOffset += 150
        }
        if(xmlData.question[index].qType == "Short Answer"){
            heightOffset += 350;
        }
        if(xmlData.question[index].qType == "Multiple Choice"){

            if(xmlData.question[index].multChoiceA != "") {
                heightOffset += 140;
            }
            if(xmlData.question[index].multChoiceB != "") {
                heightOffset += 20;
            }
            if(xmlData.question[index].multChoiceC != "") {
                heightOffset += 20;
            }
            if(xmlData.question[index].multChoiceD != "") {
                heightOffset += 20;
            }
            if(xmlData.question[index].multChoiceE != "") {
                heightOffset += 20;
            }
            if(xmlData.question[index].multChoiceF != "") {
                heightOffset += 20;
            }

            if(xmlData.question[index].multChoiceG != "") {
                heightOffset += 20;
            }

        }
    }
    var endLabel:TextField = new TextField();
    endLabel.text = "End of Quiz";
    endLabel.x = 300;
    endLabel.y = heightOffset + 30;
    var endFormat:TextFormat = new TextFormat();
    endFormat.size = 14;
    endFormat.color = 0xFF0000;
    endLabel.setTextFormat(endFormat);
    questions.addChild(endLabel);

```

```

        var variables3:URLVariables = new URLVariables();
        var varSend3:URLRequest = new URLRequest(mainURL+"addstudentattempt.php");
        varLoader3.addEventListener(Event.COMPLETE, completeHandler3);
        varSend3.method = URLRequestMethod.POST;
        varLoader3.dataFormat = URLLoaderDataFormat.VARIABLES;
        varSend3.data = variables3;
        variables3.quizVersion = quizVersion;
        variables3.attempt = currentAttempt;
        variables3.quizID = quizID;
        variables3.stuID = studentID;
        varLoader3.load(varSend3);

    }
    else{
        trace("ERROR: URLLoader returned null");
    }
}

function completeHandler3(event:Event){
    currentAttemptID = Number(varLoader3.data["attemptID"]);
    myScrollPane.source = questions;
    addChild(submit);
    time_txtFormat.size = 20;
    time_txt.defaultTextFormat = time_txtFormat;
    time_txt.x = 600;
    time_txt.y = 30;
    time_txt.width = 100;
    //time_txt.text = "hello dave";
    addChild(time_txt);

    var now:Date = new Date();
    var rightNow:Number = now.getTime();
    then = rightNow + timeLimit;
    var countdownTimer:Timer = new Timer(1000);
    countdownTimer.addEventListener(TimerEvent.TIMER, updateTime);
    countdownTimer.start();
}

function updateTime(e:TimerEvent):void{
    //Current time
    var currentNow:Date = new Date();
    var timeLeft:Number = then - currentNow.getTime();
    //Converting the remaining time into seconds, minutes, hours, and days
    var seconds:Number = Math.floor(timeLeft / 1000);
    var minutes:Number = Math.floor(seconds / 60);
    var hours:Number = Math.floor(minutes / 60);
    var days:Number = Math.floor(hours / 24);

    //Storing the remainder of this division problem
    seconds %= 60;
    minutes %= 60;
    hours %= 24;
}

```



```

if(hours == 0 && minutes < 5 && seconds % 2 == 0){
    time_txtFormat.color = 0xFF0000;
    time_txt.defaultTextFormat = time_txtFormat;
}

if(hours == 0 && minutes < 5 && seconds % 2 == 1){
    time_txtFormat.color = 0x000000;
    time_txt.defaultTextFormat = time_txtFormat;
}

if(hours == 0 && minutes == 0 && seconds == 0){
    submittedForSureHandler(new MouseEvent(MouseEvent.CLICK));
}

//Converting numerical values into strings so that
//we string all of these numbers together for the display
var sec:String = seconds.toString();
var min:String = minutes.toString();
var hrs:String = hours.toString();
var d:String = days.toString();

//Setting up a few restrictions for when the current time reaches a single digit
if (sec.length < 2) {
    sec = "0" + sec;
}

if (min.length < 2) {
    min = "0" + min;
}

if (hrs.length < 2) {
    hrs = "0" + hrs;
}

//Stringing all of the numbers together for the display
var time:String = d + ":" + hrs + ":" + min + ":" + sec;
//Setting the string to the display

time_txt.text = time;
}

submit.addEventListener(MouseEvent.CLICK, submitHandler);
submit.x = 100;
submit.y = 820;
submit.label = "Submit";

myScrollPane.setSize(690, 600);
myScrollPane.move(10, 210);
myScrollPane.setStyle("upSkin",Sprite);
myScrollPane.horizontalScrollPolicy = ScrollPolicy.OFF
myScrollPane.addEventListener(Event.COMPLETE, completeHandler);

```

```

addChild(myScrollPane);

var areYouSure:Sprite = new Sprite();
var screen:Shape = new Shape();
screen.graphics.beginFill(0x000000);
screen.graphics.drawRect(0, 0, 720, 864);
screen.alpha = .5;

var box:Shape = new Shape();
box.graphics.beginFill(0xFFFFFFFF);
box.graphics.drawRect(230, 250, 270, 150);

var areYouSureLabel:TextField = new TextField();
areYouSureLabel.wordWrap = true;
areYouSureLabel.x = 255;
areYouSureLabel.y = 255;
areYouSureLabel.width = 200;
areYouSureLabel.height = 100;
var areYouSureLabelFormat:TextFormat = new TextFormat();

function submitHandler(event:MouseEvent):void {
    ////////////////////////////////////Check if any answers are null before
    submitting!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

    var nullChecker:int = 0;
    for(var index:int = 0; index<int(numberOfQuestions);index++){
        var temp:Object = questions.getChildByName("question" + index);

        if(temp.answer == null){
            nullChecker += 1;
        }

    }

    areYouSureLabel.text = "Are you sure you want to submit this quiz?\n\n";
    if(nullChecker > 0){
        areYouSureLabel.appendText("You have " + nullChecker + " questions unanswered!");
    }

    areYouSureLabelFormat.align = "center";
    areYouSureLabelFormat.color = 0xFF0000;
    areYouSureLabelFormat.size = 16;
    areYouSureLabel.setTextFormat(areYouSureLabelFormat);

    areYouSureSubmit.label = "Submit";
    areYouSureSubmit.move(253, 350);
    areYouSureSubmit.addEventListener(MouseEvent.CLICK, submittedForSureHandler);
}

```

```

areYouSureCancel.label = "Cancel";
areYouSureCancel.move(375, 350);
areYouSureCancel.addEventListener(MouseEvent.CLICK, cancelledForSureHandler);

areYouSure.addChild(screen);
areYouSure.addChild(box);
areYouSure.addChild(areYouSureLabel);
areYouSure.addChild(areYouSureSubmit);
areYouSure.addChild(areYouSureCancel);

addChild(areYouSure);

}

function submittedForSureHandler(event:Event):void{
    for(var index:int = 0; index<int(numberOfQuestions);index++){
        var temp:Object = questions.getChildByName("question" + index);

        answerArray.push(temp.answer);

    }
    var variables4:URLVariables = new URLVariables();
    var varSend4:URLRequest = new URLRequest(mainURL+"addstudentanswers.php");
    varLoader4.addEventListener(Event.COMPLETE, completeHandler4);
    varSend4.method = URLRequestMethod.POST;
    varLoader4.dataFormat = URLLoaderDataFormat.VARIABLES;
    varSend4.data = variables4;
    variables4.attemptID = currentAttemptID;
    variables4.answers = answerArray.join("!!!!");
    varLoader4.load(varSend4);

    /*
    for (var index1:int = 0; index1<answerArray.length; index1++){
        var answerNumber:int = index1 + 1;

        trace(answerNumber + ". " + answerArray[index1]);
    }
    */

}

function completeHandler4(event:Event){
    areYouSureLabelFormat.align = "center";
    areYouSureLabelFormat.color = 0xFF0000;
    areYouSureLabelFormat.size = 16;

    areYouSureLabel.text = "Your quiz was submitted successfully.";
    areYouSureLabel.setTextFormat(areYouSureLabelFormat);
    addChild(areYouSureLabel);
    removeChild(submit);
    removeChild(myScrollPane);
    removeChild(instructionsText);
    removeChild(insLabel);

```

```

        removeChild(tf);
        removeChild(line);
        removeChild(attemptsTextBox);
        removeChild(time_txt);
        areYouSureSubmit.enabled = false;
        areYouSureCancel.enabled = false;
    }

    function cancelledForSureHandler(event:Event):void{

        answerArray = [];
        removeChild(areYouSure);

    }

    function completeHandler(event:Event):void {
        myScrollPane.update();
    }

```

Assignment Console:

```

import fl.controls.TextArea;
import flash.net.URLLoader;
import flash.net.URLVariables;
import flash.net.URLRequest;
import flash.net.URLRequestMethod;
import flash.net.URLLoaderDataFormat;
import flash.net.sendToURL;
import flash.text.TextFormat;
import flash.text.TextFieldAutoSize;
import fl.controls.Label;

var titleText          :Label = new Label();
var titleTextFormat    :TextFormat = new TextFormat();
var dueDateText        :Label = new Label();
var dueDateTextFormat  :TextFormat = new TextFormat();
var descriptionLabel    :Label = new Label();
var descriptionLabelFormat :TextFormat = new TextFormat();
var assignBodyText     :TextArea = new TextArea();
var assignBodyTextFormat :TextFormat = new TextFormat();
var variables2         :URLVariables = new URLVariables();
var mainURL            :String = "http://www.advancedcognition.com/";
var varSend2          :URLRequest = new
URLRequest(mainURL+"getassignments.php");
var varLoader2        :URLLoader = new URLLoader;
var extUserName        :String;
var extCourseName      :String = root.loaderInfo.parameters.courseName;
var assignmentNumber   :String = root.loaderInfo.parameters.assignmentNumber;
var monthLabels:Array = new Array("January",
    "February",
    "March",
    "April",

```

```

    "May",
    "June",
    "July",
    "August",
    "September",
    "October",
    "November",
    "December");

```

```

ExternalInterface.addCallback("getAssignment", getAssignment);

```

```

varLoader2.addEventListener(Event.COMPLETE, completeHandler2);
varSend2.method = URLRequestMethod.POST;
varLoader2.dataFormat = URLLoaderDataFormat.VARIABLES;
varSend2.data = variables2;
getAssignment();

```

```

function getAssignment():void{
    ExternalInterface.call("eval", "var UN = this.getField('UserName');");
    extUserName = ExternalInterface.call("eval", "UN.value;");

    ExternalInterface.call("eval", "var CN = this.getField('course');");
    //extCourseName = ExternalInterface.call("eval", "CN.value;");
    //////////////////////////////////////////////////Change
    //extUserName = "daboncanplay";
    //extCourseName = "Comm 1250";
    //assignmentNumber = "1";
    //////////////////////////////////////////////////
    variables2.UN = extUserName;
    variables2.CN = extCourseName;
    variables2.assignmentNumber = assignmentNumber;
    trace(variables2.CN);
    varLoader2.load(varSend2);
    //messageText.text = "\n\nLoading Messages. . .";

}
function completeHandler2(event:Event):void
{

```

```

    var sqlDT:String = varLoader2.data['dueDate'];
    trace(varLoader2.data['numrows']);
    var Myear:String = sqlDT.slice(0, 4);
    var Mmonth:String = sqlDT.slice(5, 7);
    var Mday:String = sqlDT.slice(8, 10);
    var MHour:int = int(sqlDT.slice(11, 13));
    var MMin:String = sqlDT.slice(14, 16);
    var ampm:String = "a.m.";
    if(int(MHour) > 12){
        MHour -= 12;
        ampm = "p.m.";
    }

```

```

        var formattedDateTime:String = monthLabels[int(Mmonth) - 1] + " " + int(Mday) + ", " + Myear
+ " - " + MHour + ":" + MMin + " " + ampm;

```

```

titleTextFormat.size = 20;
titleText.setStyle("textFormat", titleTextFormat);
titleText.autoSize = TextFieldAutoSize.CENTER;
titleText.width = 800;
titleText.text = "Assignment " + assignmentNumber + " -- " + varLoader2.data['assignTitle'];
addChild(titleText);

dueDateTextFormat.color = "0xFF0000";
dueDateTextFormat.size = 14;
dueDateText.setStyle("textFormat", dueDateTextFormat);
dueDateText.x = 660;
dueDateText.y = 40;
dueDateText.autoSize = TextFieldAutoSize.RIGHT;
dueDateText.text = "Due: " + formattedDateTime;
addChild(dueDateText);

descriptionLabelFormat.size = 14;
descriptionLabel.setStyle("textFormat", descriptionLabelFormat);
descriptionLabel.move(20, 40);
descriptionLabel.text = "Description";
addChild(descriptionLabel);

assignBodyTextFormat.size = 14;
assignBodyTextFormat.font = "Arial";
assignBodyText.setStyle("textFormat", assignBodyTextFormat);
assignBodyText.x = 10;
assignBodyText.y = 60;
assignBodyText.width = 780;
assignBodyText.height = 700;
assignBodyText.editable = false;
assignBodyText.text = varLoader2.data['description'];
addChild(assignBodyText);
}

```

Professor Console (Main Application Window):

Note: The Professor Console was developed in Flex 3.0 and is written in MXML.

```

<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:comp="components.*"
    layout="absolute"
    creationComplete="init()">

    <mx:Script>
        <![CDATA[

            import com.ProfessorConsole.UIHandlers.*;

            private var professorName:String = "David Harris";

```

```

private function init():void
{

    coursePanelInstance.getCourses(professorName);
    messagesWindow.loadMessages(professorName);
    availabilityWindow.init(professorName);

}

private function courseChosenHandler(event:Event):void
{

    courseInfoPanel.loadInfo(coursePanelInstance.chosenCourse);
    syllabusPanel.loadSyllabus(coursePanelInstance.chosenCourse);
    assignmentsPanel.loadAssignments(coursePanelInstance.chosenCourse);
    quizzesPanel.loadQuizList(coursePanelInstance.chosenCourse);
}

private function messagesMaximizeWindowHandler(event:Event):void
{

    //First bring interface back to restored state
    restoreAllWindows();
    //Maximize Messages Window
    maximizeWindow(messagesWindow);
}

private function messagesRestoreWindowHandler(event:Event):void
{

    restoreAllWindows();
}

private function chatMaximizeWindowHandler(event:Event):void
{

    //First bring interface back to restored state
    restoreAllWindows();
    //Maximize Chat Window
    maximizeWindow(chatWindow);
}

private function chatRestoreWindowHandler(event:Event):void
{

    restoreAllWindows();
}

private function liveMaximizeWindowHandler(event:Event):void
{

    //First bring interface back to restored state
    restoreAllWindows();
    //Maximize Live Window
    //maximizeWindow(liveWindow);
}

```

```

private function liveRestoreWindowHandler(event:Event):void
{
    restoreAllWindows();
}

private function availabilityMaximizeWindowHandler(event:Event):void
{
    //First bring interface back to restored state
    restoreAllWindows();
    //Maximize Messages Window
    maximizeWindow(availabilityWindow);
}

private function availabilityRestoreWindowHandler(event:Event):void
{
    restoreAllWindows();
}

private function maximizeWindow(window:Object):void
{
    window.setStyle("top", "topMiddleContentSection:0");
    window.setStyle("bottom", "bottomMiddleContentSection:0");
    window.setStyle("left", "leftMiddleContentSection:0");
    window.setStyle("right", "rightMiddleContentSection:0");
}

private function restoreAllWindows():void
{
    //Restore Messages Window
    messagesWindow.setStyle("top", 10);
    messagesWindow.setStyle("bottom", "topContentSection:5");
    messagesWindow.setStyle("left", "rightContentSection:5");
    messagesWindow.setStyle("right", 10);
    messagesWindow.maximized = false;
    //Restore Chat Window
    chatWindow.setStyle("top", "topMiddleContentSection:5");
    chatWindow.setStyle("bottom", "topMiddleContentSection:5");
    chatWindow.setStyle("left", "rightContentSection:5");
    chatWindow.setStyle("right", 10);
    chatWindow.maximized = false;
    /* //Restore Live Window
    liveWindow.setStyle("top", "bottomMiddleContentSection:5");
    liveWindow.setStyle("bottom", "bottomMiddleContentSection:5");
    liveWindow.setStyle("left", "rightContentSection:5");
    liveWindow.setStyle("right", 10);
    liveWindow.maximized = false; */
    //Restore Availability Window
    availabilityWindow.setStyle("top", "bottomMiddleContentSection:5");
    availabilityWindow.setStyle("bottom", "bottomContentSection:5");
    availabilityWindow.setStyle("left", "rightContentSection:5");
    availabilityWindow.setStyle("right", 10);
}

```



```

        availabilityWindow.maximized = false;
    }
    ]]>
</mx:Script>

<mx:constraintColumns>
    <mx:ConstraintColumn id="leftContentSection"
        width="25%" />
    <mx:ConstraintColumn id="leftMiddleContentSection"
        width="25%" />
    <mx:ConstraintColumn id="rightMiddleContentSection"
        width="25%" />
    <mx:ConstraintColumn id="rightContentSection"
        width="25%" />
</mx:constraintColumns>

<mx:constraintRows>
    <mx:ConstraintRow id="wayTopContentSection"
        height="12%" />
    <mx:ConstraintRow id="topContentSection"
        height="13%" />
    <mx:ConstraintRow id="topMiddleContentSection"
        height="25%" />
    <mx:ConstraintRow id="bottomMiddleContentSection"
        height="25%" />
    <mx:ConstraintRow id="bottomContentSection"
        height="25%" />
</mx:constraintRows>

<mx:Panel id="manageCoursePanel"
    top="topContentSection:5" bottom="10"
    left="10" right="rightMiddleContentSection:5"
    title="Manage Course">

    <mx:TabNavigator id="manageTabNavigator"
        width="100%" height="100%"
        creationPolicy="all">
        <mx:VBox label="Course Info"
            horizontalAlign="center" verticalAlign="center">
            <comp:CourseInfoPanel id="courseInfoPanel" />
        </mx:VBox>
        <!-- <mx:VBox label="Students">
        <mx:Label text="TabNavigator container panel 1"/>
        </mx:VBox> -->

        <mx:VBox label="Syllabus">
            <comp:SyllabusPanel id="syllabusPanel" />
        </mx:VBox>

        <mx:VBox label="Assignments">
            <comp:AssignmentsPanel id="assignmentsPanel" />
        </mx:VBox>

        <mx:VBox label="Quizzes">
            <comp:QuizzesPanel id="quizzesPanel" />

```

```

</mx:VBox>

<!-- <mx:VBox label="Grades">
    <mx:Label text="TabNavigator container panel 3"/>
</mx:VBox> -->

<mx:VBox label="Calendar">
    <mx:Label text="Calendar (Coming Soon)" />
</mx:VBox>

</mx:TabNavigator>
</mx:Panel>

<mx:Panel id="coursePanel"
    title="Courses"
    layout="horizontal"
    color="0x000000"
    top="10"
    bottom="wayTopContentSection:5"
    left="10"
    right="rightMiddleContentSection:5">

    <comp:CoursePanel id="coursePanelInstance"
        courseChosen="courseChosenHandler(event)" />

</mx:Panel>

<comp:Messages id="messagesWindow"
    top="10"
    bottom="topContentSection:5"
    left="rightContentSection:5" right="10"
    restoreWindow="messagesRestoreWindowHandler(event)"
    maximizeWindow="messagesMaximizeWindowHandler(event)" />
<comp:Chat id="chatWindow"
    top="topMiddleContentSection:5"
    bottom="topMiddleContentSection:5"
    left="rightContentSection:5" right="10"
    restoreWindow="chatRestoreWindowHandler(event)"
    maximizeWindow="chatMaximizeWindowHandler(event)" />
<!-- <comp:Live id="liveWindow"
    top="bottomMiddleContentSection:5"
    bottom="bottomMiddleContentSection:5"
    left="rightContentSection:5" right="10"
    restoreWindow="liveRestoreWindowHandler(event)"
    maximizeWindow="liveMaximizeWindowHandler(event)" /> -->
<comp:Availability id="availabilityWindow"
    top="bottomMiddleContentSection:5"
    bottom="10"
    left="rightContentSection:5" right="10"
    restoreWindow="availabilityRestoreWindowHandler(event)"
    maximizeWindow="availabilityMaximizeWindowHandler(event)" />

</mx:Application>

```

Professor Console (Add Assignment Component):

```

<?xml version="1.0" encoding="utf-8"?>
<mx:TitleWindow xmlns:mx="http://www.adobe.com/2006/mxml"
    width="100%" height="100%" horizontalAlign="center"
    showCloseButton="true" close="PopUpManager.removePopUp(this);" >

    <mx:Metadata>
        [Event(name="refreshAssignmentsList", type="flash.events.Event")]
    </mx:Metadata>

    <mx:Script>
        <![CDATA[

            //import statements here
            import mx.rpc.events.ResultEvent;
            import mx.managers.PopUpManager;

            //var declarations here
            [Bindable]
            public var windowTitle:String = "";

            [Bindable]
            public var courseName:String;

            [Bindable]
            public var currentAssignmentNumber:String;

            [Bindable]
            public var assignmentTitle:String;

            [Bindable]
            public var description:String;

            [Bindable]
            public var dueDate:String;

            [Bindable]
            public var updateType:String;

            public function updateAssignments(event:Event):void
            {

                assignmentsUpdater.send();

            }

            private function resultHandler(event:ResultEvent):void
            {

                var refreshListObject:Event = new Event("refreshAssignmentsList");
                dispatchEvent(refreshListObject);

```

```

        PopUpManager.removePopUp(this);
    }

    ]]>
</mx:Script>

<mx:HTTPService id="assignmentsUpdater"
    url="http://www.advancedcognition.com/updateassignments.php"
    method="POST"
    resultFormat="flashvars"
    result="resultHandler(event)">

    <mx:request>
        <CN>{courseName}</CN>
        <updateType>{updateType}</updateType>
        <assignmentNumber>{nextNumber.text}</assignmentNumber>
        <title>{assignTitleInput.text}</title>
        <description>{descriptionInput.text}</description>
        <dueDate>{dueDateInput.text}</dueDate>
    </mx:request>
</mx:HTTPService>

<mx:Form width="100%">
<mx:FormHeading fontSize="10" label="Course Information" />

<mx:FormItem label="Assignment:">
    <mx:Label id="nextNumber" width="300" text="{currentAssignmentNumber}" />
</mx:FormItem>

<mx:FormItem label="Title:">
    <mx:TextInput id="assignTitleInput" width="400" text="{assignmentTitle}" />
</mx:FormItem>

<mx:FormItem label="Description:">
    <mx:TextArea id="descriptionInput" width="600" height="300" text="{description}" />
</mx:FormItem>

<mx:FormItem label="Due Date:">
    <mx:TextInput id="dueDateInput" width="400" text="{dueDate}" />
</mx:FormItem>

</mx:Form>

<mx:HBox>
<mx:Button label="Cancel" click="PopUpManager.removePopUp(this);"/>
    <mx:Button label="Finish" click="updateAssignments(event);"/>
    <mx:Label id="debugger" text="{assignmentsUpdater.lastResult.result}" />
</mx:HBox>

</mx:TitleWindow>

```

Professor Console (Add Message Component):

```

<?xml version="1.0" encoding="utf-8"?>
<mx:TitleWindow xmlns:mx="http://www.adobe.com/2006/mxml"
    width="100%" height="100%" horizontalAlign="center"
    showCloseButton="true" close="PopUpManager.removePopUp(this);" >

    <mx:Metadata>
        [Event(name="refreshAssignmentsList", type="flash.events.Event")]
    </mx:Metadata>

    <mx:Script>
        <![CDATA[

            //import statements here
            import mx.rpc.events.ResultEvent;
            import mx.managers.PopUpManager;
            import mx.collections.ArrayCollection;

            //var declarations here
            [Bindable]
            public var windowTitle:String = "";

            [Bindable]
            public var profName:String;

            [Bindable]
            private var courses:XML;

            [Bindable]
            private var studentListDP:ArrayCollection = new ArrayCollection();

            [Bindable]
            private var chosenCourse:String;

            [Bindable]
            private var toIDs:String;

            public function init():void
            {

                courseList.send();

            }

            public function sendMessage(event:Event):void
            {

                toIDs = "";
                for(var n:int = 0; n < studentList.selectedItems.length; n++){

                    if(n !== 0){
                        toIDs = toIDs.concat("---");
                    }


```

```

        toIDs = toIDs.concat(studentList.selectedItems[n].data);
    }
    newMessageService.send();
}

private function studentListResultHandler(event:ResultEvent):void
{
    studentListDP.removeAll();
    for(var index:int = 1; index <= event.result.NumberOfStudents;
index++)
    {
        var tmpStuID:String = "stu_id" + String(index);
        var tmpStuName:String = "student" + String(index);
        var tmpLabel:String = event.result[tmpStuName];
        var tmpData:String = event.result[tmpStuID];
        var o:Object = { label:tmpLabel, data:tmpData };
        studentListDP.addItem(o);
        studentList.dataProvider = studentListDP;
    }
}

private function courseResultHandler(event:ResultEvent):void
{
    courses = event.result as XML;
}

private function courseChosenHandler(event:Event):void
{
    studentListDP.removeAll();
    studentListDP.addItem({ label:"Loading. . ." });
    studentList.dataProvider = studentListDP;
    chosenCourse = ComboBox(event.target).selectedItem.identifier;
    getStudentListService.send();
}

private function resultHandler(event:ResultEvent):void
{
    if(event.result.message == "sent"){
        debugger.text = "Message sent successfully";
    }else{
        debugger.text = "Message failed";
    }
}

```

```

    ]]>
</mx:Script>

<mx:HTTPService id="courseList"
    url="http://www.advancedcognition.com/getcourses.php"
    method="POST"
    resultFormat="e4x"
    result="courseResultHandler(event)">

    <mx:request>
        <PN>{ profName }</PN>
    </mx:request>
</mx:HTTPService>

<mx:HTTPService id="getStudentListService"
    url="http://www.advancedcognition.com/getstudentlist.php"
    method="POST"
    resultFormat="flashvars"
    result="studentListResultHandler(event)">

    <mx:request>
        <CN>{ chosenCourse }</CN>

    </mx:request>
</mx:HTTPService>

<mx:HTTPService id="newMessageService"
    url="http://www.advancedcognition.com/sendmessage.php"
    method="POST"
    resultFormat="flashvars"
    result="resultHandler(event)">

    <mx:request>
        <CN>{ chosenCourse }</CN>
        <fromProf>yes</fromProf>
        <fromName>{ profName }</fromName>
        <toIDs>{ toIDs }</toIDs>
        <body>{ messageText.text }</body>

    </mx:request>
</mx:HTTPService>

<mx:Form width="100%">
    <mx:FormHeading fontSize="10" label="New Message" />
<mx:VBox>

    <mx:FormItem label="Course:">
        <mx:ComboBox id="course"
            dataProvider="{ courses.Course }"
            width="300"
            change="courseChosenHandler(event)" />

    </mx:FormItem>
</mx:HBox>

```

```

<mx:FormItem label="Student(s):">
    <mx:List id="studentList"
        dataProvider="{ studentListDP}"
        width="200"
        allowMultipleSelection="true" />
</mx:FormItem>

<mx:FormItem label="Message:">
    <mx:TextArea id="messageText" width="200" height="150" />
</mx:FormItem>

</mx:HBox>
</mx:VBox>
</mx:Form>

    <mx:HBox>
    <mx:Button label="Done" click="PopUpManager.removePopUp(this);"/>
        <mx:Button label="Send" click="sendMessage(event);"/>
        <mx:Label id="debugger" text="" />
    </mx:HBox>

</mx:TitleWindow>

```

Professor Console (Assignment Panel Component):

```

<?xml version="1.0" encoding="utf-8"?>
<mx:VBox xmlns:mx="http://www.adobe.com/2006/mxml"
    width="100%" height="100%" >

    <mx:Script>
        <![CDATA[

            //import statements here

            import mx.rpc.events.ResultEvent;
            import mx.controls.Alert;
            import mx.events.CloseEvent;
            import mx.managers.PopUpManager;
            import flash.geom.Point;

            //var declarations here

            public var currentAssignmentNumber:String;

            [Bindable]
            private var assignmentResultData:XMLList = new XMLList();

            [Bindable]
            private var courseName:String = "";

            [Bindable]
            private var selectedAssignmentNumber:String;

```



```

public function loadAssignments(courseName:String):void
{
    this.courseName = courseName;
    assignmentsList.send();
}

private function newCourseHandler(event:Event):void
{
    var newCourseObject:Event = new Event("newCourse");
    dispatchEvent(newCourseObject);
}

private function resultHandler(event:ResultEvent):void
{
    assignmentResultData = event.result.assignment
    var currentNumber:XMLList = event.result.currentAssignmentNumber
    currentAssignmentNumber = currentNumber[0].number;
}

private function updateResultHandler(event:ResultEvent):void
{
    assignmentsList.send();
}

private function deleteSelectedAssignmentHandler(event:MouseEvent):void
{
    Alert.yesLabel = "Yes";
    Alert.noLabel = "No";
    Alert.show("Are you sure?", "Delete Assignment", 3, this, alertClickHandler);
}

private function alertClickHandler(event:CloseEvent):void {
    if (event.detail==Alert.YES)
    {
        selectedAssignmentNumber =
String(assignmentResultData[assignments.selectedIndex].Number);
        assignmentsUpdater.send();
    }
}

private function newAssignmentHandler(event:MouseEvent):void
{

```

```

        var
editor:AddAssignmentPanel=AddAssignmentPanel(PopUpManager.createPopUp( this, AddAssignmentPanel, true
));
        //PopUpManager.centerPopUp(editor);
        editor.x = (stage.width / 2) - (editor.width / 2);
        editor.y = (stage.height / 2) - (editor.height / 2);
        editor.currentAssignmentNumber =
String(int(this.currentAssignmentNumber) + 1);
        editor.courseName = courseName;
        editor.updateType = "new";
        editor.addEventListener("refreshAssignmentsList",
refreshListHandler);

    }

    private function editSelectedAssignmentHandler(event:MouseEvent):void
    {

        var
editor:AddAssignmentPanel=AddAssignmentPanel(PopUpManager.createPopUp( this, AddAssignmentPanel, true
));
        //var point1:Point = new Point();
        // Calculate position of TitleWindow in Application's coordinates.
        //point1.x=newAssignmentButton.x;
        //point1.y=newAssignmentButton.y;
        //point1=newAssignmentButton.localToGlobal(point1);
        //editor.x=point1.x-300;
        //editor.y=point1.y-400;
        editor.x = (stage.width / 2) - (editor.width / 2);
        editor.y = (stage.height / 2) - (editor.height / 2);
        editor.currentAssignmentNumber =
String(assignmentResultData[assignments.selectedIndex].Number);
        editor.courseName = courseName;
        editor.assignmentTitle =
String(assignmentResultData[assignments.selectedIndex].Title);
        editor.description =
String(assignmentResultData[assignments.selectedIndex].Description);
        editor.dueDate =
String(assignmentResultData[assignments.selectedIndex].DueDate);
        editor.updateType = "edit";
        editor.addEventListener("refreshAssignmentsList",
refreshListHandler);

    }

    private function refreshListHandler(event:Event):void
    {

        assignmentsList.send();

    }

]]>
</mx:Script>

<mx:HTTPService id="assignmentsList"

```

```

url="http://www.advancedcognition.com/getallassignments.php"
method="POST"
resultFormat="e4x"
result="resultHandler(event)">

<mx:request>
    <CN>{courseName}</CN>
</mx:request>
</mx:HTTPService>

<mx:HTTPService id="assignmentsUpdater"
url="http://www.advancedcognition.com/updateassignments.php"
method="POST"
resultFormat="flashvars"
result="updateResultHandler(event)">

    <mx:request>
        <CN>{courseName}</CN>
        <assignmentNumber>{selectedAssignmentNumber}</assignmentNumber>
        <updateType>delete</updateType>
    </mx:request>
</mx:HTTPService>

<mx:DataGrid id="assignments"
    dataProvider="{assignmentResultData}"
    width="95%" height="90%"
    rowHeight="30">
    <mx:columns>
        <mx:DataGridColumn dataField="Number" headerText="Number"/>
        <mx:DataGridColumn dataField="Title" headerText="Title"/>
        <mx:DataGridColumn dataField="Description" headerText="Description"/>
        <mx:DataGridColumn dataField="DueDate" headerText="Due Date"/>
    </mx:columns>
</mx:DataGrid>

<mx:HBox horizontalAlign="center" width="100%">
    <mx:Button id="deleteAssignmentButton"
        label="Archive Selected Assignment"
        click="deleteSelectedAssignmentHandler(event)" />
    <mx:Button id="newAssignmentButton"
        label="Add New Assignment"
        click="newAssignmentHandler(event)" />
    <mx:Button id="editAssignmentButton"
        label="Edit Selected Assignment"
        click="editSelectedAssignmentHandler(event)" />
    <mx:Label id="debug" width="200" text="{assignmentsUpdater.lastResult.result}" />
</mx:HBox>

</mx:VBox>

```

Professor Console (Availability Component):

```
<?xml version="1.0" encoding="utf-8"?>
```

```

<comp:MaximizablePanel xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:comp="components.*"
  windowTitle="Update Availability">

  <mx:Script>
    <![CDATA[

      //import statements here

      //var declarations here

      [Bindable]
      private var profName:String;

      public function init(profName:String):void
      {

        profName = this.profName;
        phoneAvail.init("phone", "phoneAvailPhoto.jpg",
"phoneNotAvailPhoto.jpg");

        chatAvail.init("chat", "chatAvailPhoto.jpg", "chatNotAvailPhoto.jpg");
        vcAvail.init("vc", "vcAvailPhoto.jpg", "vcNotAvailPhoto.jpg");

      }

    ]]>
  </mx:Script>
  <mx:VBox>
    <mx:HBox width="100%">
      <comp:AvailabilityIcon id="phoneAvail" />
      <comp:AvailabilityIcon id="chatAvail"/>
    </mx:HBox>
    <comp:AvailabilityIcon id="vcAvail"/>
  </mx:VBox>
</comp:MaximizablePanel>

```

Professor Console (Availability Icon Component):

```

<?xml version="1.0" encoding="utf-8"?>
<mx:Canvas xmlns:mx="http://www.adobe.com/2006/mxml"
  xmlns:comp="components.*" >

  <mx:Script>
    <![CDATA[

      //import statements here
      import mx.rpc.events.ResultEvent;

      //var declarations here

      [Bindable]
      public var availType:String;

      private var currentAvail:String;

```

```

[Bindable]
private var changeCurrentAvail:String;

[Bindable]
public var availImageName:String;

[Bindable]
public var notAvailImageName:String;

[Bindable]
private var changeUrl:String =
"http://www.advancedcognition.com/changeavailability.php";

[Bindable]
private var checkUrl:String =
"http://www.advancedcognition.com/checkavailability.php";

private var imageUrl:String = "http://www.advancedcognition.com/images/";

[Bindable]
private var availImageUrl:String = imageUrl + availImageName;

[Bindable]
private var notAvailImageUrl:String = imageUrl + notAvailImageName;

public function init(availType1:String, availImageName1:String,
notAvailImageName1:String):void
{
    availType = availType1;
    availImageName = availImageName1;
    notAvailImageName = notAvailImageName1;
    getAvailability.send();
    availImageUrl = imageUrl + availImageName;
    notAvailImageUrl = imageUrl + notAvailImageName;
    availImage.load(availImageUrl);
    notAvailImage.load(notAvailImageUrl);
    //availImage.visible = false;
    //notAvailImage.visible = false;
}

private function getAvailabilityResultHandler(event:ResultEvent):void
{
    var typeString:String = availType + "availability";
    currentAvail = event.result[typeString];
    if(event.result[typeString] == "yes")
    {
        notAvailImage.visible = false;
    }else{
        notAvailImage.visible = true;
    }
}

```

```

        }
        //debugger.text = "here" + event.result[typeString];
    }

    private function changeAvailabilityResultHandler(event:ResultEvent):void
    {

        getAvailability.send();

    }

    private function toggleButtonClickHandler(event:Event):void
    {

        var sendObject:Object = {};
        if(currentAvail == "yes")
        {

            sendObject[availType] = "no";

        }else{

            sendObject[availType] = "yes";

        }

        updateAvailability.send(sendObject);

    }
    ]]>
</mx:Script>

<mx:HTTPService id="getAvailability"
    url="{checkUrl}"
    method="POST"
    resultFormat="flashvars"
    result="getAvailabilityResultHandler(event)">

</mx:HTTPService>

<mx:HTTPService id="updateAvailability"
    url="{changeUrl}"
    method="POST"
    resultFormat="flashvars"
    result="changeAvailabilityResultHandler(event)">

</mx:HTTPService>

<mx:Image id="availImage" />
<mx:Image id="notAvailImage" />

<mx:Button id="toggleButton"

```

```

        x="25" y="130"
        label="Toggle"
        click="toggleButtonClickHandler(event)" />

<mx:Label id="debugger" />
</mx:Canvas>

```

Professor Console (Chat Component):

```

<?xml version="1.0" encoding="utf-8"?>
<comp:MaximizablePanel xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:comp="components.*"
    windowTitle="Chat">

    <mx:Script>
        <![CDATA[

                //import statements here

                //var declarations here

                [Bindable]
                public var dummy:Object;

            ]]>
    </mx:Script>

    <mx:SWFLoader id="chatSwf"
        source="preloader.swf"
        width="100%" height="100%" />

</comp:MaximizablePanel>

```

Professor Console (Course Information Panel Component):

```

<?xml version="1.0" encoding="utf-8"?>
<mx:VBox xmlns:mx="http://www.adobe.com/2006/mxml"
    width="450" height="400"
    horizontalAlign="center">

    <mx:Script>
        <![CDATA[

                //import statements here

                //var declarations here

                [Bindable]
                public var courseName:String = "";

                public function loadInfo(courseName:String):void

```

```

        {
            successLabel.text = "";
            this.courseName = courseName;
            infoLoader.send();
        }

        private function updateInfo():void
        {
            successLabel.text = "";
            infoUpdater.send();
        }
    }

</mx:Script>

<mx:HTTPService id="infoLoader"
    url="http://www.advancedcognition.com/getcourseinfo.php"
    method="POST"
    resultFormat="flashvars">

    <mx:request>
        <CN>{ courseName } </CN>
    </mx:request>

</mx:HTTPService>

<mx:HTTPService id="infoUpdater"
    url="http://www.advancedcognition.com/updateinfo.php"
    method="POST"
    resultFormat="flashvars">

    <mx:request>
        <CN>{ courseName } </CN>
        <title>{ cname.text } </title>
        <instructor>{ instructor.text } </instructor>
        <time>{ times.text } </time>
        <room>{ place.text } </room>
    </mx:request>

</mx:HTTPService>

<mx:Form width="100%">
<mx:FormHeading fontSize="10" label="Course Information" />

<mx:FormItem label="Course Title:">
    <mx:Label id="cname" width="300" text="{infoLoader.lastResult.title}" />
</mx:FormItem>

<mx:FormItem label="Instructor:">
    <mx:Label id="instructor" width="300" text="{infoLoader.lastResult.instructor}" />
</mx:FormItem>

<mx:FormItem label="Meeting Times:">

```



```

        <mx:TextInput id="times" width="200" text="{infoLoader.lastResult.time}" />
    </mx:FormItem>

    <mx:FormItem label="Meeting Place:">
        <mx:TextInput id="place" width="200" text="{infoLoader.lastResult.room}" />
    </mx:FormItem>

</mx:Form>

<mx:Button label="Update Course Information" click="updateInfo()" />
<mx:Label id="successLabel" text="{infoUpdater.lastResult.result}" />
</mx:VBox>

```

Professor Console (Course Panel Component):

```

<?xml version="1.0" encoding="utf-8"?>
<mx:HBox xmlns:mx="http://www.adobe.com/2006/mxml"
    width="100%" height="100%">

    <mx:Metadata>
        [Event(name="courseChosen", type="flash.events.Event")]
        [Event(name="newCourse", type="flash.events.Event")]
    </mx:Metadata>

    <mx:Script>
        <![CDATA[

            //import statements here
            import mx.collections.ArrayCollection;
            import mx.rpc.events.ResultEvent;

            //var declarations here

            public var chosenCourse:String;

            [Bindable]
            public var windowTitle:String = "";

            [Bindable]
            private var courses:XML;

            [Bindable]
            private var profName:String;

            public function getCourses(profName:String):void
            {

                this.profName = profName;
                courseList.send();

            }

            private function courseChosenHandler(event:Event):void
            {

```

```

        chosenCourse = ComboBox(event.target).selectedItem.identifier;
        var courseChosenObject:Event = new Event("courseChosen");
        dispatchEvent(courseChosenObject);

    }

    private function newCourseHandler(event:Event):void
    {

        var newCourseObject:Event = new Event("newCourse");
        dispatchEvent(newCourseObject);

    }

    private function resultHandler(event:ResultEvent):void
    {

        courses = event.result as XML;
        //debug.text = event.result.Courses;

    }

    ]]>
</mx:Script>

<mx:Label text="Course Editor" />
<mx:HTTPService id="courseList"
    url="http://www.advancedcognition.com/getcourses.php"
    method="POST"
    resultFormat="e4x"
    result="resultHandler(event)">

    <mx:request>
        <PN>{ profName }</PN>
    </mx:request>
</mx:HTTPService>

<mx:ComboBox id="course"
    dataProvider="{ courses.Course }"
    width="300"
    change="courseChosenHandler(event)" />

</mx:HBox>

```

Professor Console (Maximizable Panel Component):

```

<?xml version="1.0" encoding="utf-8"?>
<mx:TitleWindow xmlns:mx="http://www.adobe.com/2006/mxml"
    title="{ windowTitle }"
    showCloseButton="true"
    close="maximizeHandler(event)">

```

```

<mx:Metadata>
    [Event(name="maximizeWindow", type="flash.events.Event")]
    [Event(name="restoreWindow", type="flash.events.Event")]
</mx:Metadata>

<mx:Script>
    <![CDATA[

        //import statements here

        //var declarations here

        [Bindable]
        public var windowTitle:String = "";
        public var maximized:Boolean = false;

        private function maximizeHandler(event:Event):void
        {

            if(maximized)
            {
                //Restore
                var clickRestore:Event = new Event("restoreWindow");
                dispatchEvent(clickRestore);
                maximized = false;

            }else{
                //Maximize
                var clickMaximize:Event = new Event("maximizeWindow");
                dispatchEvent(clickMaximize);
                maximized = true;

            }

        }

    ]]>
</mx:Script>

<!-- <mx:Label text="Hey There" /> -->

</mx:TitleWindow>

```

Professor Console (Messages Component):

```

<?xml version="1.0" encoding="utf-8"?>
<comp:MaximizablePanel xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:comp="components.*"
    windowTitle="Messages"
    horizontalAlign="center" >

    <mx:Script>
        <![CDATA[

```

```

//import statements here
import mx.rpc.events.ResultEvent;
import mx.managers.PopUpManager;

//var declarations here
private var monthLabels:Array = new Array("January",
"February",
"March",
"April",
"May",
"June",
"July",
"August",
"September",
"October",
"November",
"December");

[Bindable]
public var userName:String = "";

[Bindable]
private var messageText:String = "";

public function loadMessages(professorName:String):void
{

    userName = professorName;
    messageLoader.send();

}

public function messageLoaderResult(event:ResultEvent):void
{
    messageText = "";
    var totalMessages:int = Number(event.result.numberOfMessages);

    for(var Mindex:int = 1; Mindex <= totalMessages; Mindex++){
        var Mtmp:String = "message" + String(Mindex);

        var DTtmp:String = "timeDate" + String(Mindex);
        var Ftmp:String = "from" + String(Mindex);
        var sqlDT:String = event.result[DTtmp];
        var Myear:String = sqlDT.slice(0, 4);
        var Mmonth:String = sqlDT.slice(5, 7);
        var Mday:String = sqlDT.slice(8, 10);
        var MHour:int = int(sqlDT.slice(11, 13));
        var MMin:String = sqlDT.slice(14, 16);
        var ampm:String = "a.m.";
        if(int(MHour) > 12){
            MHour -= 12;
            ampm = "p.m.";
        }
        var formattedDateTime:String = monthLabels[int(Mmonth) - 1] + " " +
int(Mday) + ", " + Myear + " - " + MHour + ":" + MMin + " " + ampm;

```

```

        messageText += "\t";
        messageText += formattedDateTime + " -- FROM: " +

event.result[Ftmp] + " --> ";

        messageText += event.result[Mtmp];
        messageText += "\n\n\n";
    }

}

private function newMessageHandler(event:MouseEvent):void
{

    var

newMessageEditor:AddMessagePanel=AddMessagePanel(PopUpManager.createPopUp( this, AddMessagePanel,
true ));

        newMessageEditor.x = (stage.width / 2) - (newMessageEditor.width /
2);

        newMessageEditor.y = (stage.height / 2) - (newMessageEditor.height /
2);

        newMessageEditor.profName = userName;
        newMessageEditor.init();

    }

    ]]>
</mx:Script>

<mx:HTTPService id="messageLoader"
    url="http://www.advancedcognition.com/getannouncements.php"
    method="POST"
    resultFormat="flashvars"
    result="messageLoaderResult(event)" >

    <mx:request>
        <UN>{userName}</UN>
        <professor>yes</professor>
    </mx:request>

</mx:HTTPService>

<mx:TextArea id="messageTextArea" width="100%" height="90%" editable="false" >
    <mx:text>
        {messageText}
    </mx:text>
</mx:TextArea>

<mx:Button id="checkMessageButton"
    label="Check Messages"
    click="messageLoader.send();" />

<mx:Button id="newMessageButton"
    label="New Message"
    click="newMessageHandler(event)" />

```

```
</comp:MaximizablePanel>
```

Professor Console (Question Editor Component):

```
<?xml version="1.0" encoding="utf-8"?>
<mx:TitleWindow xmlns:mx="http://www.adobe.com/2006/mxml"
    width="100%" height="100%" horizontalAlign="center"
    showCloseButton="true" close="PopUpManager.removePopUp(this);" >

    <mx:Metadata>
        [Event(name="updateQuestionsList", type="events.UpdateQuestionListEvent")]
    </mx:Metadata>

    <mx:Script>
        <![CDATA[

            //import statements here
            import events.UpdateQuestionListEvent;
            import mx.managers.PopUpManager;
            import valueObjects.Question;

            //var declarations here
            [Bindable]
            public var windowTitle:String = "";

            [Bindable]
            public var currentQuestionNumber:String;

            [Bindable]
            public var questionContent:String;

            [Bindable]
            public var questionType:String;

            public var choicesString:String;

            [Bindable]
            private var choiceA:String;

            [Bindable]
            private var choiceB:String;

            [Bindable]
            private var choiceC:String;

            [Bindable]
            private var choiceD:String;

            [Bindable]
            private var choiceE:String;
```

```

[Bindable]
private var choiceF:String;

[Bindable]
private var choiceG:String;

[Bindable]
public var questionTypes:Array = new Array("Multiple Choice",
"True/False", "Fill in the Blank", "Short Answer");

public function init():void
{
    questionTypeCBox.selectedIndex =
questionTypes.indexOf(questionType);
    updateChoicesForm();
    var choiceArray:Array = choicesString.split("--- ");
    choiceA = choiceArray[0].substr(3);
    choiceB = choiceArray[1].substr(3);
    choiceC = choiceArray[2].substr(3);
    choiceD = choiceArray[3].substr(3);
    choiceE = choiceArray[4].substr(3);
    choiceF = choiceArray[5].substr(3);
    choiceG = choiceArray[6].substr(3);
}

private function questionTypeCBoxChangeHandler(event:Event):void
{
    updateChoicesForm();
}

private function updateChoicesForm():void
{
    if(questionTypeCBox.selectedItem == "Multiple Choice")
    {
        choicesForm.enabled = true;
    }else{
        choicesForm.enabled = false;
    }
}

```

```

private function doneButtonClickHandler(event:Event):void
{

    choicesString = "";
    if(choiceAText.text != "")
    {
        choicesString += "A: " + choiceAText.text;
    }
    if(choiceBText.text != "")
    {
        choicesString += " --- B: " + choiceBText.text;
    }
    if(choiceCText.text != "")
    {
        choicesString += " --- C: " + choiceCText.text;
    }
    if(choiceDText.text != "")
    {
        choicesString += " --- D: " + choiceDText.text;
    }
    if(choiceEText.text != "")
    {
        choicesString += " --- E: " + choiceEText.text;
    }
    if(choiceFText.text != "")
    {
        choicesString += " --- F: " + choiceFText.text;
    }
    if(choiceGText.text != "")
    {
        choicesString += " --- G: " + choiceGText.text;
    }

    var questionData:Question = new Question();

    questionData.questionNumber = questionNumber.text;

    questionData.questionType =
questionTypes[questionTypeCBox.selectedIndex];
    questionData.questionContent = questionContentText.text;
    questionData.choicesString = choicesString;

    var questionEventObject:UpdateQuestionListEvent = new
UpdateQuestionListEvent("updateQuestionsList", questionData);

    dispatchEvent(questionEventObject);
    //debugger.text = "here";
    //PopUpManager.removePopUp(this);

}

```



```

    ]]>
</mx:Script>

<mx:Form width="100%">
  <mx:FormHeading fontSize="10" label="Question Editor" />
</mx:VBox>

  <mx:FormItem label="      Question:">
    <mx:Label id="questionNumber"
      text="{ currentQuestionNumber}" />
  </mx:FormItem>

  <mx:FormItem label="    Question Type:">
    <mx:ComboBox id="questionTypeCBox"
      dataProvider="{ questionTypes}"
      width="200"
      change="questionTypeCBoxChangeHandler(event)" />
  </mx:FormItem>

  <mx:FormItem label="Question Content:">
    <mx:TextArea id="questionContentText"
      width="400"
      text="{ questionContent}" />
  </mx:FormItem>

</mx:HBox horizontalAlign="right" >

  <mx:Form id="choicesForm" >
    <mx:FormHeading fontSize="10" label="Multiple Choices" />

    <mx:FormItem label="A:">
      <mx:TextInput id="choiceAText"
        width="400"
        text="{ choiceA}" />
    </mx:FormItem>
    <mx:FormItem label="B:">
      <mx:TextInput id="choiceBText"
        width="400"
        text="{ choiceB}" />
    </mx:FormItem>
    <mx:FormItem label="C:">
      <mx:TextInput id="choiceCText"
        width="400"
        text="{ choiceC}" />
    </mx:FormItem>
    <mx:FormItem label="D:">

```

```

        <mx:TextInput id="choiceDText"
            width="400"
            text="{ choiceD}" />
    </mx:FormItem>
    <mx:FormItem label="E:">
        <mx:TextInput id="choiceEText"
            width="400"
            text="{ choiceE}" />
    </mx:FormItem>
    <mx:FormItem label="F:">
        <mx:TextInput id="choiceFText"
            width="400"
            text="{ choiceF}" />
    </mx:FormItem>
    <mx:FormItem label="G:">
        <mx:TextInput id="choiceGText"
            width="400"
            text="{ choiceG}" />
    </mx:FormItem>
</mx:Form>
</mx:HBox>
</mx:VBox>
</mx:Form>

    <mx:HBox>
    <mx:Button label="Done" click="doneButtonClickHandler(event)"/>
        <mx:Label id="debugger" text="" />
    </mx:HBox>

</mx:TitleWindow>

```

Professor Console (Quiz Editor Component):

```

<?xml version="1.0" encoding="utf-8"?>
<mx:TitleWindow xmlns:mx="http://www.adobe.com/2006/mxml"
    width="100%" height="100%"
    showCloseButton="true" close="PopUpManager.removePopUp(this);" >

    <mx:Script>
        <![CDATA[

                //import statements here
                import events.UpdateQuestionListEvent;
                import mx.rpc.events.ResultEvent;
                import mx.controls.Alert;

```

```

import mx.events.CloseEvent;
import mx.managers.PopUpManager;
import flash.geom.Point;
import mx.core.IFlexDisplayObject;

//var declarations here

public var currentQuizNumber:String;

[Bindable]
private var submitQuiz_id:String = "1";

[Bindable]
private var currentQuizVersion:String;

[Bindable]
private var quizResultData:XMLList = new XMLList();

[Bindable]
private var quizInformationResultData:XMLList = new XMLList();

[Bindable]
private var quizQuestionsResultData:XMLList = new XMLList();

[Bindable]
public var courseName:String = "";

[Bindable]
public var selectedQuizNumber:String;

[Bindable]
private var months:Array = new Array("January", "February", "March",
"April", "May", "June", "July", "August", "September", "October", "November", "December");

[Bindable]
private var dayNumbers:Array = new Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31);

[Bindable]
private var attemptsNumbers:Array = new Array("Unlimited", "1", "2",
"3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15");

[Bindable]
private var timeAllowedNumbers:Array = new Array("Unlimited", "15
minutes", "30 minutes", "45 minutes", "1 hour", "1 hour 15 minutes", "1 hour 30 minutes", "1 hour 45
minutes", "2 hours");

[Bindable]
private var yearNumbers:Array = new Array(2009, 2010, 2011, 2012,
2013, 2014, 2015);

```

```

public function loadQuiz():void
{
    quizzesInformationList.send();
}

private function submitChangesClickHandler(event:Event):void
{
    debug.text = "event yeah";
}

private function newQuizHandler(event:Event):void
{
    //var newCourseObject:Event = new Event("newCourse");
    //dispatchEvent(newCourseObject);
}

private function quizListResultHandler(event:ResultEvent):void
{
    quizResultData = event.result.quiz;

    //var currentNumber:XMLList =
event.result.currentAssignmentNumber;
    //currentAssignmentNumber = currentNumber[0].number;
}

private function
quizzesInformationListResultHandler(event:ResultEvent):void
{
    quizInformationResultData = event.result.quiz;
    quizQuestionsResultData =
quizInformationResultData.questions.question;
    currentQuizVersion =
quizInformationResultData.currentQuizVersion.number;
    debug.text = currentQuizVersion;
    questions.dataProvider = quizQuestionsResultData;
    quizTitleInput.text = quizInformationResultData.title;
    chooseQuizCBox.text = quizInformationResultData.title;
    instructionsInput.text = quizInformationResultData.instructions;
    //debug.text = String(attemptsNumbers.indexOf(String(quizInformationResultData.attempts)));
    attemptsAllowedCBox.selectedIndex =
attemptsNumbers.indexOf(String(quizInformationResultData.attempts));
    timeAllowedCBox.selectedIndex =
timeAllowedNumbers.indexOf(String(quizInformationResultData.timeAllowed));
}

```

```

beginMonthCBox.selectedIndex =
months.indexOf(String(quizInformationResultData.startMonth));
beginDayCBox.selectedIndex =
dayNumbers.indexOf(Number(quizInformationResultData.startDay));
beginYearCBox.selectedIndex =
yearNumbers.indexOf(Number(quizInformationResultData.startYear));
endMonthCBox.selectedIndex =
months.indexOf(String(quizInformationResultData.endMonth));
endDayCBox.selectedIndex =
dayNumbers.indexOf(Number(quizInformationResultData.endDay));
endYearCBox.selectedIndex =
yearNumbers.indexOf(Number(quizInformationResultData.endYear));
//var currentNumber:XMLList =
event.result.currentAssignmentNumber;
//currentAssignmentNumber = currentNumber[0].number;
}

private function updateResultHandler(event:ResultEvent):void
{

    //assignmentsList.send();

}

private function deleteSelectedQuestionHandler(event:MouseEvent):void
{

    Alert.yesLabel = "Yes";
    Alert.noLabel = "No";
    Alert.show("Are you sure?", "Delete Question", 3, this,
alertClickHandler);
}

private function alertClickHandler(event:CloseEvent):void {

    if (event.detail==Alert.YES)
    {
        delete
quizQuestionsResultData[questions.selectedIndex];

        for(var x:uint = 0; x <=
quizQuestionsResultData.length(); x++)
        {

            quizQuestionsResultData[x].questionNumber =
x + 1;

            //debug.text = "Hi" + String(x);

        }

        //debug.text = String(quizQuestionsResultData.length());

```

```

        questions.dataProvider = quizQuestionsResultData;

    }

}

private function newQuestionHandler(event:MouseEvent):void
{

    var
editor:QuestionEditorPanel=QuestionEditorPanel(PopUpManager.createPopUp( this,
QuestionEditorPanel, true ));

    editor.x = (stage.width / 2) - (editor.width / 2);
    editor.y = (stage.height / 2) - (editor.height / 2);

    editor.currentQuestionNumber =
String(quizQuestionsResultData.length() + 1);
    //editor.courseName = courseName;
    //editor.updateType = "new";
    //editor.addEventListener("refreshAssignmentsList",
refreshListHandler);

}

private function editSelectedQuestionHandler(event:MouseEvent):void
{

    var
editor:QuestionEditorPanel=QuestionEditorPanel(PopUpManager.createPopUp( this,
QuestionEditorPanel, true ));

    editor.x = (stage.width / 2) - (editor.width / 2);
    editor.y = (stage.height / 2) - (editor.height / 2);

    editor.currentQuestionNumber =
String(quizQuestionsResultData[questions.selectedIndex].questionNumber);
    editor.questionContent =
String(quizQuestionsResultData[questions.selectedIndex].questionContent);
    editor.questionType =
String(quizQuestionsResultData[questions.selectedIndex].questionType);
    editor.choicesString =
String(quizQuestionsResultData[questions.selectedIndex].choicesString);
    editor.init();
    systemManager.addEventListener("updateQuestionsList",
updateQuestionsListEventHandler);

    //editor.updateType = "edit";
    //editor.addEventListener("refreshAssignmentsList",
refreshListHandler);

}

```

```

        private function
updateQuestionsListEventHandler(event:UpdateQuestionListEvent):void
        {

            debug.text = "Event dispatched";

        }

        private function submitChanges(event:Event):void
        {

        }

        private function refreshListHandler(event:Event):void
        {

            //assignmentsList.send();

        }

    ]]>
</mx:Script>

<mx:HTTPService id="quizzesInformationList"
    url="http://www.advancedcognition.com/getquizinformation.php"
    method="POST"
    resultFormat="e4x"
    result="quizzesInformationListResultHandler(event)">

    <mx:request>
        <CN>{courseName}</CN>
        <quizNumber>{selectedQuizNumber}</quizNumber>
    </mx:request>
</mx:HTTPService>

<mx:HTTPService id="quizUpdater"
    url="http://www.advancedcognition.com/processxml.php"
    method="POST"
    resultFormat="flashvars"
    result="updateResultHandler(event)">

    <mx:request>
        <course_id>1</course_id>
        <quiz_id>1</quiz_id>

        <updateType>delete</updateType>
    </mx:request>
</mx:HTTPService>

```

```

<mx:HBox>
    <mx:Label text="Currently Editing:" />
    <mx:Label id="chooseQuizCBox" />

</mx:HBox>

<mx:Form width="100%">
    <mx:FormHeading fontSize="10" label="Quiz Editor" />

    <mx:HBox width="100%">

        <mx:Label text=" Quiz Title:" />
        <mx:TextInput id="quizTitleInput" width="300" text="" />

        <mx:Label text="Attempts Allowed:" />
        <mx:ComboBox id="attemptsAllowedCBox"
            width="143"
            dataProvider="{ attemptsNumbers}" />

        <mx:Label text="Time Allowed:" />
        <mx:ComboBox id="timeAllowedCBox"
            width="143"
            dataProvider="{ timeAllowedNumbers}" />

    </mx:HBox>

    <mx:HBox width="100%">

        <mx:Label text="Instructions:" />
        <mx:TextArea id="instructionsInput"
            width="805" height="80" >
            <mx:text>

                </mx:text>
        </mx:TextArea>

    </mx:HBox>

    <mx:HBox width="100%">

        <mx:Label text="Dates Available - Begin:" />
        <mx:ComboBox id="beginMonthCBox" dataProvider="{ months}" />
        <mx:ComboBox id="beginDayCBox" dataProvider="{ dayNumbers}" />
        <mx:ComboBox id="beginYearCBox" dataProvider="{ yearNumbers}"

/>

        <mx:Label text="Dates Available - End:" />
        <mx:ComboBox id="endMonthCBox" dataProvider="{ months}" />
        <mx:ComboBox id="endDayCBox" dataProvider="{ dayNumbers}" />
        <mx:ComboBox id="endYearCBox" dataProvider="{ yearNumbers}" />

```



```

        </mx:HBox>

    </mx:Form>

    <mx:DataGrid id="questions"
        dataProvider="{quizQuestionsResultData}"
        width="95%" height="90%" >
        <mx:columns>
            <mx:DataGridColumn width='60' dataField="questionNumber"
headerText="Question Number"/>
            <mx:DataGridColumn width='100' dataField="questionType"
headerText="Question Type"/>
            <mx:DataGridColumn dataField="questionContent" headerText="Question Content"/>
            <mx:DataGridColumn dataField="choicesString" headerText="Choices (if needed)"/>
        </mx:columns>
    </mx:DataGrid>

    <mx:HBox horizontalAlign="center" width="100%">
        <mx:Button id="deleteQuestionButton"
            label="Delete Selected Question"
            click="deleteSelectedQuestionHandler(event)" />
        <mx:Button id="newQuestionButton"
            label="Add New Question"
            click="newQuestionHandler(event)" />
        <mx:Button id="editQuestionButton"
            label="Edit Selected Question"
            click="editSelectedQuestionHandler(event)" />

        <mx:Button id="testbutton"
            label="Submit Changes"
            click="submitChangesClickHandler(event)" />

        <mx:Label id="debug" width="200" text="" />
    </mx:HBox>

</mx:TitleWindow>

```

Professor Console (Quizzes Panel Component):

```

<?xml version="1.0" encoding="utf-8"?>
<mx:VBox xmlns:mx="http://www.adobe.com/2006/mxml"
    width="100%" height="100%" >

    <mx:Script>
        <![CDATA[

```

```

//import statements here

import mx.rpc.events.ResultEvent;
import mx.controls.Alert;
import mx.events.CloseEvent;
import mx.managers.PopUpManager;
import flash.geom.Point;

//var declarations here

public var currentQuizVersion:String;

[Bindable]
private var quizResultData:XMLList = new XMLList();

[Bindable]
private var quizInformationResultData:XMLList = new XMLList();

[Bindable]
private var quizQuestionsResultData:XMLList = new XMLList();

[Bindable]
private var courseName:String = "";

[Bindable]
private var selectedQuizNumber:String;

[Bindable]
private var months:Array = new Array("January", "February", "March",
"April", "May", "June", "July", "August", "September", "October", "November", "December");

[Bindable]
private var dayNumbers:Array = new Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31);

[Bindable]
private var attemptsNumbers:Array = new Array("Unlimited", "1", "2",
"3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15");

[Bindable]
private var timeAllowedNumbers:Array = new Array("Unlimited", "15
minutes", "30 minutes", "45 minutes", "1 hour", "1 hour 15 minutes", "1 hour 30 minutes", "1 hour 45
minutes", "2 hours");

[Bindable]
private var yearNumbers:Array = new Array(2009, 2010, 2011, 2012,
2013, 2014, 2015);

public function loadQuizList(courseName:String):void
{

```

```

        this.courseName = courseName;
        quizzesList.send();
    }

    private function quizChosenHandler(event:Event):void
    {

        selectedQuizNumber = chooseQuizCBox.selectedItem.data;
        quizzesInformationList.send();

    }

    private function quizListResultHandler(event:ResultEvent):void
    {

        quizResultData = event.result.quiz;

        var currentNumber:XMLList = event.result.currentQuizVersion;
        currentQuizVersion = currentNumber[0].number;

    }

    private function
    quizzesInformationListResultHandler(event:ResultEvent):void
    {

        quizInformationResultData = event.result.quiz;
        quizQuestionsResultData =
        quizInformationResultData.questions.question;

        questions.dataProvider = quizQuestionsResultData;
        quizTitleInput.text = quizInformationResultData.title;
        instructionsInput.text = quizInformationResultData.instructions;
        debug.text = String(attemptsNumbers.indexOf(String(quizInformationResultData.attempts)));
        attemptsAllowed.text =
        String(quizInformationResultData.attempts);
        timeAllowed.text =
        String(quizInformationResultData.timeAllowed);
        beginDateLabel.text =
        String(quizInformationResultData.startMonth) + " " + String(quizInformationResultData.startDay) + ", "
        + String(quizInformationResultData.startYear);
        endDateLabel.text =
        String(quizInformationResultData.endMonth) + " " + String(quizInformationResultData.endDay) + ", "
        + String(quizInformationResultData.endYear);
        //var currentNumber:XMLList =
        event.result.currentAssignmentNumber;
        //currentAssignmentNumber = currentNumber[0].number;

    }

    private function updateResultHandler(event:ResultEvent):void

```

```

        {

            //assignmentsList.send();

        }

private function archiveQuizHandler(event:MouseEvent):void
{

    Alert.yesLabel = "Yes";
    Alert.noLabel = "No";
    Alert.show("Are you sure?", "Delete Question", 3, this,
alertClickHandler);
    }

private function alertClickHandler(event:CloseEvent):void {

    if (event.detail==Alert.YES)
    {
        //selectedAssignmentNumber =
String(quizResultData[assignments.selectedIndex].Number);
        //assignmentsUpdater.send();

    }

}

private function newQuizHandler(event:MouseEvent):void
{

    var editor:QuizEditor=QuizEditor(PopUpManager.createPopUp(
this, QuizEditor, true ));

    editor.x = (stage.width / 2) - (editor.width / 2);
    editor.y = (stage.height / 2) - (editor.height / 2);

    //editor.currentQuizNumber =
String(int(this.currentQuizVersion) + 1);
    editor.courseName = courseName;
    editor.selectedQuizNumber = selectedQuizNumber;
    //editor.updateType = "new";
    //editor.addEventListener("refreshAssignmentsList",
refreshListHandler);

}

private function editQuizHandler(event:MouseEvent):void
{

```

```

this, QuizEditor, true ));

var editor:QuizEditor=QuizEditor(PopUpManager.createPopUp(
    editor.x = (stage.width / 2) - (editor.width / 2);
    editor.y = (stage.height / 2) - (editor.height / 2);

    //editor.currentQuizNumber =
String(int(this.currentQuizVersion) + 1);
    editor.courseName = courseName;
    editor.selectedQuizNumber = selectedQuizNumber;
    editor.loadQuiz();
    //editor.description =
String(quizResultData[assignments.selectedIndex].Description);
    //editor.dueDate =
String(quizResultData[assignments.selectedIndex].DueDate);
    //editor.updateType = "edit";
    //editor.addEventListener("refreshAssignmentsList",
refreshListHandler);
    }

```

```

private function refreshListHandler(event:Event):void
{

    //assignmentsList.send();

}

```

```

]]>
</mx:Script>

```

```

<mx:HTTPService id="quizzesList"
    url="http://www.advancedcognition.com/getquizlist.php"
    method="POST"
    resultFormat="e4x"
    result="quizListResultHandler(event)">

```

```

<mx:request>
    <CN>{ courseName } </CN>
</mx:request>

```

```

</mx:HTTPService>

```

```

<mx:HTTPService id="quizzesInformationList"
    url="http://www.advancedcognition.com/getquizinformation.php"
    method="POST"
    resultFormat="e4x"
    result="quizzesInformationListResultHandler(event)">

```

```

<mx:request>
    <CN>{ courseName } </CN>
    <quizNumber>{ selectedQuizNumber } </quizNumber>

```

```

        </mx:request>
    </mx:HTTPService>

    <mx:HBox>
        <mx:Label text="Currently Viewing:" />
        <mx:ComboBox id="chooseQuizCBox"
            dataProvider="{quizResultData}"
            change="quizChosenHandler(event)" />

        <mx:Button id="newQuizButton"
            label="New Quiz"
            click="newQuizHandler(event)" />
    </mx:HBox>

```

```

    <mx:HBox width="100%">

        <mx:Label text=" Quiz Title:" />
        <mx:Label id="quizTitleInput" width="300" text="" />

        <mx:Label text="Attempts Allowed:" />
        <mx:Label id="attemptsAllowed"
            width="143" />

        <mx:Label text="Time Allowed:" />
        <mx:Label id="timeAllowed"
            width="143" />

```

```

    </mx:HBox>

```

```

    <mx:HBox width="100%">

        <mx:Label text="Instructions:" />
        <mx:Text id="instructionsInput"
            width="805" height="80" />

```

```

    </mx:HBox>

```

```

    <mx:HBox width="100%">

        <mx:Label text="Dates Available - Begin:" />
        <mx:Label id="beginDateLabel" />

        <mx:Label text="Dates Available - End:" />
        <mx:Label id="endDateLabel" />

```

```

    </mx:HBox>

```

```

<mx:DataGrid id="questions"
    dataProvider="{quizQuestionsResultData}"
    width="95%" height="60%" >
    <mx:columns>
        <mx:DataGridColumn width='60' dataField="questionNumber"
headerText="Question Number"/>
        <mx:DataGridColumn width='100' dataField="questionType"
headerText="Question Type"/>
        <mx:DataGridColumn dataField="questionContent" headerText="Question Content"/>
        <mx:DataGridColumn dataField="choicesString" headerText="Choices (if needed)"/>
    </mx:columns>
</mx:DataGrid>

<mx:HBox horizontalAlign="center" width="100%">
    <mx:Button id="archiveQuizButton"
        label="Archive Quiz"
        click="archiveQuizHandler(event)" />
    <mx:Button id="editQuestionButton"
        label="Edit Quiz"
        click="editQuizHandler(event)" />

    <mx:Label id="debug" width="200" text="" />
</mx:HBox>

</mx:VBox>

```

Professor Console (Syllabus Panel Component):

```

<?xml version="1.0" encoding="utf-8"?>
<mx:VBox xmlns:mx="http://www.adobe.com/2006/mxml"
    width="100%" height="100%"
    horizontalAlign="center">

    <mx:Script>
        <![CDATA[

            //import statements here

            //var declarations here

            [Bindable]
            public var courseName:String = "";

            public function loadSyllabus(courseName:String):void
            {

```

```

        successLabel.text = "";
        this.courseName = courseName;
        syllabusLoader.send();
    }

    private function updateSyllabus():void
    {

        successLabel.text = "";
        syllabusUpdater.send();
    }

    ]]>
</mx:Script>

<mx:HTTPService id="syllabusLoader"
    url="http://www.advancedcognition.com/getsyllabus.php"
    method="POST"
    resultFormat="flashvars">

    <mx:request>
        <CN>{ courseName }</CN>
    </mx:request>

</mx:HTTPService>

<mx:HTTPService id="syllabusUpdater"
    url="http://www.advancedcognition.com/updatesyllabus.php"
    method="POST"
    resultFormat="flashvars">

    <mx:request>
        <CN>{ courseName }</CN>
        <syllabus>{ syllabusText.text }</syllabus>
    </mx:request>

</mx:HTTPService>

<mx:TextArea id="syllabusText"
    width="100%" height="90%">
    <mx:text>
        { syllabusLoader.lastResult.syllabus }
    </mx:text>

</mx:TextArea>

<mx:Button label="Update Syllabus" click="updateSyllabus()" />
<mx:Label id="successLabel" text="{ syllabusUpdater.lastResult.result}" />
</mx:VBox>

```


REFERENCES

- Chen, S. (2002). A cognitive model for non-linear learning in hypermedia programmes. *British Journal of Educational Technology* 33(4), 449-460.
- Chen, S. and Macredie, R. (2004). Cognitive modeling of student learning in web-based instructional programs. *International Journal of Human-Computer Interaction*, 17(3), 375-402.
- Kinshuk, Liu, T., & Graf, S. (2009). Coping with mismatched courses: Students' behaviour and performance in courses mismatched to their learning styles. *Education Tech Research*, 57, 739-752.
- Mitchell, T., Chen, S., & Macradie, R. (2005a). Hypermedia learning and prior knowledge: Domain expertise vs. system expertise. *Journal of Computer Assisted Learning* 21, 53-64.
- Mitchell, T., Chen, S., & Macradie, R. (2005b). The relationship between web enjoyment and student perceptions and learning using a web-based tutorial. *Learning, Media and Technology* 30(1), 27-40.
- Scheiter, K. & Gerjets, P. (2007). Learner control in hypermedia environments. *Educational Psychology Review*, 19, 285-307.