

California State University, San Bernardino

**CSUSB ScholarWorks**

---

Theses Digitization Project

John M. Pfau Library

---

2001

## **Presentations world wide systems**

Sandra Marie Hengstebeck

Follow this and additional works at: <https://scholarworks.lib.csusb.edu/etd-project>



Part of the [Digital Communications and Networking Commons](#)

---

### **Recommended Citation**

Hengstebeck, Sandra Marie, "Presentations world wide systems" (2001). *Theses Digitization Project*. 1922.

<https://scholarworks.lib.csusb.edu/etd-project/1922>

This Project is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact [scholarworks@csusb.edu](mailto:scholarworks@csusb.edu).

PRESENTATIONS WORLD WIDE  
SYSTEMS

---

A Project  
Presented to the  
Faculty of  
California State University,  
San Bernardino

---

In Partial Fulfillment  
of the Requirements for the Degree  
Master of Science  
in  
Computer Science

---

by  
Sandra Marie Hengstebeck

June 2001

PRESENTATIONS WORLD WIDE  
SYSTEMS

---

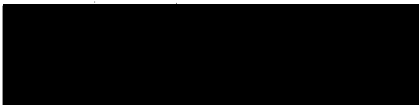
A Project  
Presented to the  
Faculty of  
California State University,  
San Bernardino

---

by  
Sandra Marie Hengstebeck

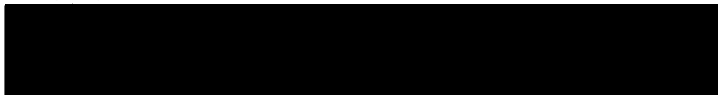
June 2001

Approved by:

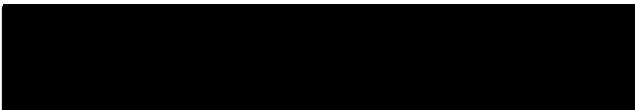


George Georgiou, Computer Science

6/8/01  
Date



Richard Botting



Kerstin Voigt

## ABSTRACT

As we enter the 21<sup>st</sup> century, new communication technologies are abundant. The Internet Service Provider has branched from modems to direct service links and cable. These new communication technologies facilitate changes in the education system. The Presentations World Wide System (PWWS) was developed to utilize the benefits of this technology.

The Presentations World Wide System is a software program that provides an instructor with a presentation tool that is convenient and easy to use. The purpose of PWWS is to allow students to view a live presentation through an Internet browser and allow the instructor to have control over the presentation as desired. The use of the Internet provides the students with the opportunity of viewing the presentation regardless of geographical locale.

The PWWS software program was written in Java to provide platform independence. The system was created with a graphical user interface to provide a user-friendly format. It provides the instructor with options in displaying the presentation and is capable of saving a



slide presentation for later viewing. Even after the selection of the presentation, the instructor can make changes during the presentation. The viewer of the presentation is not required to download additional software.

## TABLE OF CONTENTS

ABSTRACT.....	iii
LIST OF TABLES.....	ix
LIST OF FIGURES.....	xi
CHAPTER ONE: SOFTWARE REQUIREMENTS SPECIFICATION	
1.1 Introduction.....	1
1.1.1 Review Of Existing Products.....	3
1.1.2 Purpose.....	5
1.1.3 Scope.....	5
1.1.4 Definitions, Acronyms, and Abbreviations .....	6
1.1.4.1 Definitions .....	6
1.1.4.2 Acronyms and Abbreviations.....	8
1.1.5 Overview.....	9
1.2 Overall Description.....	9
1.2.1 Product Perspective.....	9
1.2.1.1 System Interfaces.....	10
1.2.1.2 Presenter Interfaces.....	11
1.2.1.2.1 Setup/Run Selection Box.....	11
1.2.1.2.2 Setup Window.....	12
1.2.1.2.3 Viewer Window.....	14
1.2.1.2.4 Run Window.....	15

1.2.1.3 Hardware Interfaces.....	16
1.2.1.4 Software Interfaces.....	16
1.2.1.5 Communications Interfaces.....	17
1.2.1.6 Memory Constraints.....	17
1.2.1.7 Operations.....	17
1.2.1.8 Site Adaptation Requirements .....	17
1.2.2 Product Functions.....	17
1.2.2.1 Setting Up the Presentation....	19
1.2.2.2 Giving the Presentation.....	20
1.2.3 User Characteristics.....	21
1.2.4 Constraints.....	21
1.2.5 Assumptions and Dependencies.....	22
1.3 Specific Requirements.....	22
1.3.1 External Interface Requirements.....	22
1.3.1.1 Setup Page.....	22
1.3.1.2 Run Page.....	24
1.3.2 Functional Requirements.....	26
1.3.2.1 Setup Page.....	26
1.3.2.2 Run Page.....	27
1.3.3 Performance Requirements.....	28
1.3.4 Software System Attributes.....	29
1.3.4.1 Maintainability.....	29

1.3.4.2 Portability.....	29
CHAPTER TWO: SOFTWARE DESIGN	
2.1 Preliminary Design.....	30
2.1.1 Use Case Diagram.....	30
2.1.2 Class Diagram.....	31
2.2 Detailed Design.....	32
2.2.1 Setup Design.....	32
2.2.1.1 Information Design.....	33
2.2.1.2 Required TextField Design.....	34
2.2.1.3 File Directory Design.....	35
2.2.1.4 Files Selected Design.....	36
2.2.1.5 Button Panel Design.....	37
2.2.1.5.1 Viewer Panel Design.....	39
2.2.2 Run Design.....	40
2.2.2.1 Link Panel Design.....	46
2.2.2.2 Browser Panel Design.....	47
2.2.2.3 Button Panel Design.....	48
2.2.3 Parser Design.....	51
2.2.4 Open Files Design.....	54
2.2.5 Save Files Design.....	55
2.2.6 Save Live Presentation Design.....	56
2.2.7 Adjust Slide Design.....	59

CHAPTER THREE: SOFTWARE QUALITY ASSURANCE	
3.1 Unit Test Plan.....	61
3.2 Integration Test Plan.....	62
3.3 System Test Plan.....	73
CHAPTER FOUR: MAINTENANCE.....	75
CHAPTER FIVE: USERS MANUAL	
5.1. System Requirements.....	76
5.2. Starting the Program.....	77
5.3. Setup Presentation.....	77
5.4. Run Presentation.....	79
CHAPTER SIX: CONCLUSION, LIMITATIONS AND FUTURE DIRECTIONS.....	83
APPENDIX A: SOURCE CODE OF PRESENTATIONS WORLD WIDE SYSTEMS.....	85
APPENDIX B: SCREEN SHOTS OF PRESENTATIONS WORLD WIDE SYSTEMS.....	129
REFERENCES.....	136

## LIST OF TABLES

Table 1.	Setup.....	32
Table 2.	Information.....	33
Table 3.	Textfields.....	34
Table 4.	JFileChooser.....	35
Table 5.	Files.....	36
Table 6.	Buttons.....	37
Table 7.	Viewer.....	39
Table 8.	FrameAndPanel.....	40
Table 9.	FillBrowser.....	41
Table 10.	SetDirectory.....	42
Table 11.	DisperseData.....	43
Table 12.	PostingLineByLine.....	44
Table 13.	SaveLiveAttributes.....	45
Table 14.	LinkPanel.....	46
Table 15.	Browser.....	47
Table 16.	Button Panel.....	48
Table 17.	HtmlParser.....	51
Table 18.	GrabLines.....	52
Table 19.	Openfiles.....	54
Table 20.	Savefile.....	55
Table 21.	SaveLivePresentation.....	56

Table 22. AddFile.....	57
Table 23. AddLink.....	58
Table 24. AddApplet.....	59
Table 25. Unit Test Plan.....	62
Table 26. Test Case 1 - Setup/Run Box.....	64
Table 27. Test Case 2 - Setup Window.....	65
Table 28. Test Case 3 - Run Window.....	69
Table 29. System Test.....	74

## LIST OF FIGURES

Figure 1. Setup/Run Selection Box.....	12
Figure 2. Setup Window.....	12
Figure 3. Viewer Window.....	15
Figure 4. Run Window.....	16
Figure 5. Presenter Use Case Diagram.....	18
Figure 6. Viewer Use Case Diagram.....	18
Figure 7. Class Diagram.....	31



# CHAPTER ONE

## SOFTWARE REQUIREMENTS SPECIFICATION

### 1.1 Introduction

In the past, computers were a rarity in the American household, but times have changed. A computer in the household is as common as a microwave oven. Due to this fact, computers have become a mainstay in society. In addition, people's lives have changed over the years. As freeways become more congested and population continues to increase, traveling from home to just about anywhere has become difficult and time-consuming. It has also become common for employees to live long distances from their workplace and students to commute long distances to their schools. With the introduction of the computer and the web, it has alleviated some of these problems. The computer has allowed users to remain at home when conducting business. This ranges from banking and shopping to teleconferencing.

The computer has also found its way into educational institutions. Students are becoming more computer savvy and teachers are finding the computer a great resource. A classroom of today may consist of students seated behind

computer screens as the instructor lectures. However, many times due to busy schedules, traffic, and weather conditions, students are unable to attend. Additionally, students with handicaps, illnesses, or debilitating circumstances, find attending class impossible. With the availability of computers and internet access, it would be conducive if students were still able to view the lecture.

Another benefit to having a web based lecture presentation available to students is the ability to review the lecture. Often students find that trying to write down notes while listening to the lecture is detrimental to the learning experience. Part of their concentration is on taking notes and part is on the teacher and trying to comprehend and understand the new material. With this resource, the student could listen to the lecture the first time and make notes on it in review.

This project documentation describes the development of a web-based presentation tool, Presentations World Wide Systems. This tool allows the instructor the ability to give a lecture across the Internet and also save it for later viewing.

### 1.1.1 Review of Existing Products

Research was done on the currently available web presentation services. They all had advantages and disadvantages. Examples of presentation services on the web are WebEx, Astound Conference Center, and PlaceWare Web Conferencing.

WebEx was chosen as Editor's Choice by PC Magazine in the December 17, 1999 issue. (3) The WebEx service includes whiteboards, annotation tools, polling functions, chatting, and one-way presentations. If you only have up to four participants in your meeting and keep it within 30 minutes, then you can almost use it for free. The disadvantages of WebEx appear to be the requirement of scheduling a conference for a one time presentation. If the user has more than four attendees or the presentation is longer than 30 minutes, then there is a substantial cost. The lowest plan is \$25 per month for the first four users and \$10 for each additional user, with a \$.15 per user per minute cost. (Simone)

Astound Conference Center has services similar to WebEx, however it does not offer free-form drawing tools. One advantage of Astound is the ability to meet immediately

or schedule a meeting for later. A disadvantage of Astound is the fact that the links provided to the viewer are not live. Only the presenter can use them. The pricing requires a contract for a monthly rate. The shortest contract is a 3 month contract at \$25 per user. There is a one-time meeting available for \$.30 per minute per person if the meeting is less than 30 minutes . (Astound)

PlaceWare Web Conferencing offers most of the services that WebEx and Astound offer. It provides exceptional 3-D graphics. The presenter is also given the ability to add slides on the fly. However, it requires three separate downloads and was indicated in PC Magazine to be difficult to set up. (4) While trying to determine a price, on the low end, PlaceWare indicated that they have a separate service called MyPlaceWare which allows meetings with up to five attendees for free. The PlaceWare Web Conferencing Service is different than the other two web services, because it doesn't charge by the minute. It has a one-time set up expense of \$3,000 and a \$600 per person per year fee. (Simone)

### 1.1.2 Purpose

This Software Requirement Specification (SRS) documents the agreements concerning the purpose, characteristics and specific requirements of the Presentations World Wide System (PWWS). The system was constructed by Sandra Hengstebeck, as a Master's Project, on behalf of the Department of Computer Science, California State University San Bernardino.

Presenter requirements for the system were gathered by interviews with George Georgiou Ph.D., and Richard Botting Ph.D.

### 1.1.3 Scope

The PWWS provides interfaces and tools to offer web browser presentations for viewing through Wide Area Networks (WAN) and execution within a Local Area Network (LAN). PWWS employs an Internet based interface to allow a student to view a presentation from anywhere in the world as it is presented. The presenter will execute the PWWS on a local computer connected to a LAN or through an File Transfer Protocol (FTP) connection that contains the web server. Previous to the execution of the presentation, the presenter uses the PWWS to select html

files created beforehand for the presentation. This selection (the lineup of files, whether the live presentation will be saved and other relevant information) can be saved for presenting later or the presentation may begin immediately. When the presentation is executed, the presenter has the option to adhere to the previously selected lineup or diverge from this sequence as necessary. There are several options available to the presenter, such as line-by-line execution of the presentation or editing. If the presenter of the presentation has selected the option to allow a viewer to view the saved live presentation at a later date, a directory is created where the live presentation is saved. The viewer will use the URL of this directory for later viewing. This will allow viewers the ability to access the presentation at their own convenience.

#### 1.1.4 Definitions, Acronyms, and Abbreviations

##### 1.1.4.1 Definitions.

**Browser:** A computer program that enables the presenter to read Hypertext in files or on the world wide web. It is capable of retrieving HTML documents

that include references to images and Java byte code and renders them into a presenter-readable document.

Client: Any computer that is hooked up to a computer network.

HTML: Hypertext Markup Language. A set of codes that can be inserted into text files to indicate special typefaces, inserted images, links to other documents, etc.

Internet: A cooperative message forwarding system, linking computer networks all over the world.

Java: An object oriented language developed by Sun Microsystems. Java programs are capable of running on most popular computer platforms without the need for recompilation. It was developed to enable networked computers to transmit computations to each other, not just data. Java Programs are compiled not into machine code, which would not be portable, but into a concise code known as a Java byte code.

Presenter: The executor of the PWWS, generally an educator, who will create the presentation using the HTML pages that are uploaded to the web server. This

person will also be in charge of advancing to the next slide during the presentation.

Session: A related set of communications transactions between two or more network devices.

Textbox: An area that allows a presenter to input information with multiple lines.

Textfield: An area that allows a presenter to input information in a single line.

Viewer: The person, most likely a student, who will be viewing the presentation after it is created and posted to the web server.

Web Server: A computer that is attached to the Internet and contains web pages (HTML files) that can be viewed using a web browser.

#### 1.1.4.2 Acronyms and Abbreviations.

CSUSB: California State University, San Bernardino

PWWS: Presentations World Wide System

GUI: Graphical User Interface

HTML: Hyper Text Markup Language

HTTP: Hyper Text Transfer Protocol

IEEE: Institute of Electrical and Electronics Engineers



NSF: National Science Foundation  
OS: Operating System  
FTP: File Transfer Protocol  
SRS: Software Requirements Specification  
URL: Universal Resource Locator

#### 1.1.5 Overview

Section 2 of this document follows the guidelines of IEEE Std 830-1993 IEEE Recommended Practice of Software Requirements Specifications. This section provides product perspective, a summary of product functions, a description of the characteristics of the expected presenters, a listing of development constraints, and a list of assumptions and dependencies.

Section 3 of this document presents the specific requirements for this system. They are organized by mode, following the SRS Section 3 template shown in IEEE Std 830-1998, Annex A, Paragraph A.1.

### 1.2 Overall Description

#### 1.2.1 Product Perspective

The PWWS will be a Java program, and will be integrated with existing web pages. The presenter interfaces will be Graphical User Interfaces (GUIs) used

by the presenter via an existing Web Server system, the viewers will use Internet Browsers. The initial presenter interface is a GUI that provides the presenter with instructions, textfields for inputting information, selection of files and selection of running or saving the presentation. The primary presenter interface is a GUI that displays the html files (as seen in an Internet browser) as they are presented, links to the files, buttons to display the next or redisplay changes to an html file.

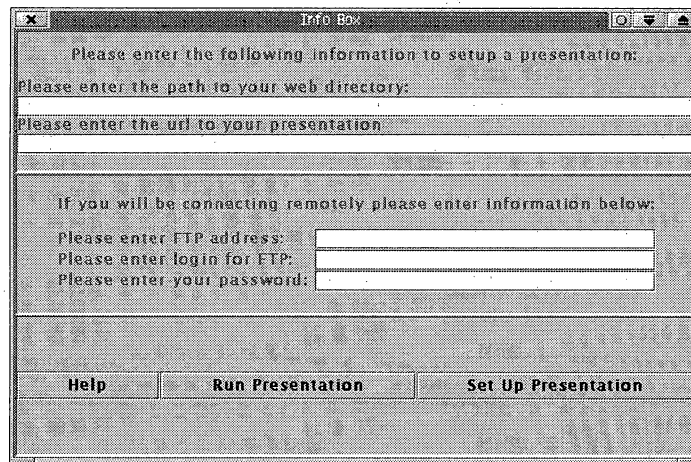
The hardware interface requirement is that it must run on a Unix type operating system with an existing web server. The software interface is that it must support current versions of Netscape and Internet Explorer and Java. The communications interface requires support for Hyper Text Transfer Protocol (HTTP) and File Transfer Protocol (FTP).

1.2.1.1 System Interfaces. The PWWS will provide the presenter's interface. Netscape Navigator and Internet Explorer 4.0 or greater will be providing the interface for the viewer. HTML and Java will be used and will have their own calls for system resources

1.2.1.2 Presenter Interfaces. Presenter interfaces for PWWS will be designed using Java. The following features will be incorporated to produce a more descriptive representation of the interface:

1.2.1.2.1 Setup/Run Selection Box. The presenter will be shown a Setup/Run Selection Box, which will provide three buttons. The Setup Presentation button will take the presenter to the Setup Window. The Run Presentation button will take the presenter to the Run (execution of presentation) Window. The User's Guide Button will provide a window displaying the User Manual. Textfields will require the presenter to input the web server directory path and the URL of the web server directory. The presenter may work remotely by inputting the FTP address, the presenter's login and password. See Figure 1.

Figure 1. Setup/Run Selection Box



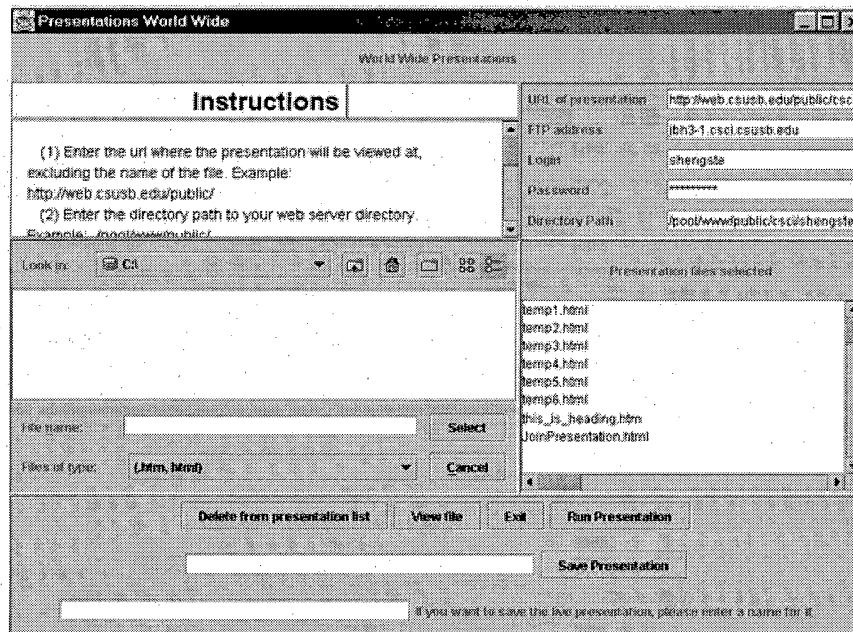
The 'Info Box' dialog contains the following text and fields:

- Please enter the following information to setup a presentation:
- Please enter the path to your web directory:
- Please enter the url to your presentation:
- If you will be connecting remotely please enter information below:
- Please enter FTP address:
- Please enter login for FTP:
- Please enter your password:

Buttons at the bottom: Help, Run Presentation, Set Up Presentation

1.2.1.2.2 Setup Window. The Setup Window provides the initial GUI of the PWWS program. This window is split into five areas. See Figure 2.

Figure 2. Setup Window



The 'Presentations World Wide' window is divided into several sections:

- Instructions:** (1) Enter the url where the presentation will be viewed at, excluding the name of the file. Example: `http://web.csusb.edu/public/`. (2) Enter the directory path to your web server directory. Example: `/pool/www/public/csc/shengstat`.
- Look in:** C:\
- File name:**  **Select**
- Files of type:** (htm, html) **Cancel**
- URL of presentation:** `http://web.csusb.edu/public/csc/`
- FTP address:** `ibh3-1.csc.csusb.edu`
- Login:** `shengsta`
- Password:** `*****`
- Directory Path:** `/pool/www/public/csc/shengstat`
- Presentation files selected:**
  - temp1.html
  - temp2.html
  - temp3.html
  - temp4.html
  - temp5.html
  - temp6.html
  - this\_is\_heading.htm
  - JoinPresentation.html
- Buttons:** Delete from presentation list, View file, Exit, Run Presentation, Save Presentation
- Footer:** If you want to save the live presentation, please enter a name for it:

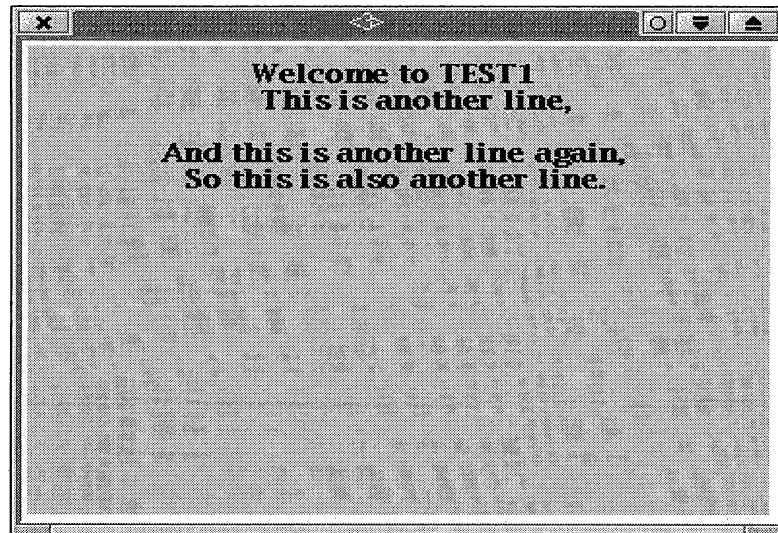
1. The instruction panel will provide the presenter with information on how to use the PWWS system.
2. An information panel where the presenter may change information on the web directory, URL paths of where presentation files will be stored and viewed from, ftp path, the presenter's login and password. FTP information can still be added or changed at this time.
3. A file selection panel, which will provide the presenter with a visual link to the directory where the html files are located. The presenter highlights and clicks the select button to choose the html file for the presentation. If the presenter is connected to the web server through an FTP connection, an additional window will display the HTML files located in the web server directory.
4. The presentation files selected panel, which displays the html files selected in the sequence they will be shown. The user can edit the names or sequence of these files with basic text editor methods.
5. The control panel, that provides buttons for removing a file from the presentation files selected panel, viewing a file that has been highlighted in the presentation files

selected panel, saving the presentation for later presentation, executing the presentation, and exiting. Two textfields are provided for giving a name to the presentation for saving the presentation for later presentation and for giving a name to the directory that will be created for saving the live presentation for viewing after the actual presentation.

Validity checking is provided to indicate that all required fields have been filled. If the fields have not been filled an error message will be displayed indicating which fields need to be filled. See Appendix B.

1.2.1.2.3 Viewer Window. The Viewer Window is secondary GUI that becomes visible when the presenter selects a file in the presentation files selected panel and clicks the View File button. The window will display the html file, as it will be seen on the Internet browser. This helps the presenter check the sequence that has been chosen and whether this is the correct file intended and URL path entered. See Figure 3.

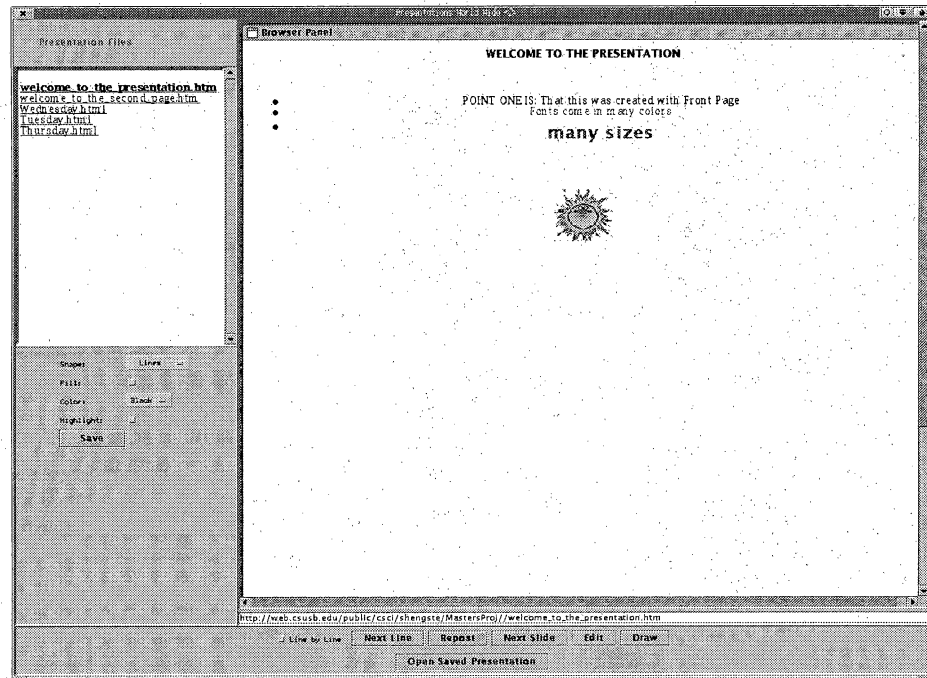
Figure 3. Viewer Window



1.2.1.2.4 Run Window. The Run Window is the primary GUI. This window is divided into four sections:

1. The Link Panel, which provides a presenter to links to the files that were selected in the Setup Window.
2. The Browser Panel, which provides an area that displays and allows changes to the html file as the presenter is executing the presentation.
3. The Drawing Panel, which provides tools for drawing shapes and lines on html files.
4. The Control Panel, which provides button and checkboxes for controlling the display of the presentation. See Figure 4.

Figure 4. Run Window



1.2.1.3 Hardware Interfaces. Essentially all hardware interfaces will be provided by the operating system. PWWS will not implement any hardware interfaces.

1.2.1.4 Software Interfaces. Software interfaces are written in Java with HTML support, and server/client networking. The following browsers would support PWWS:

- a. Netscape Navigator 4.0 and above
- b. Internet Explorer 4.0 and above



1.2.1.5 Communications Interfaces. PWWS will implement communication interfaces, such needed interfaces will be handled through the browser.

1.2.1.6 Memory Constraints. PWWS- requires a minimum of 16 Megabytes of Random Access Memory.

1.2.1.7 Operations. The presenter can assess the system and remain active as long as the system does not get affected by unforeseen in the network.

1.2.1.8 Site Adaptation Requirements. PWWS requires a Unix type operating system with a web server. The web server needs to be able to run Netscape 4.0 or Internet Explorer 4.0.

## 1.2.2 Product Functions

Figures 5 and 6 provide a Use Case Diagram that graphically depicts the presenter and the viewer with the principal functions of the PWWS. The two functions are further described in the following subsections 1.2.2.1 to 1.2.2.2, and the actors in the diagram are further described in section 1.2.3.

Figure 5. Presenter Use Case Diagram

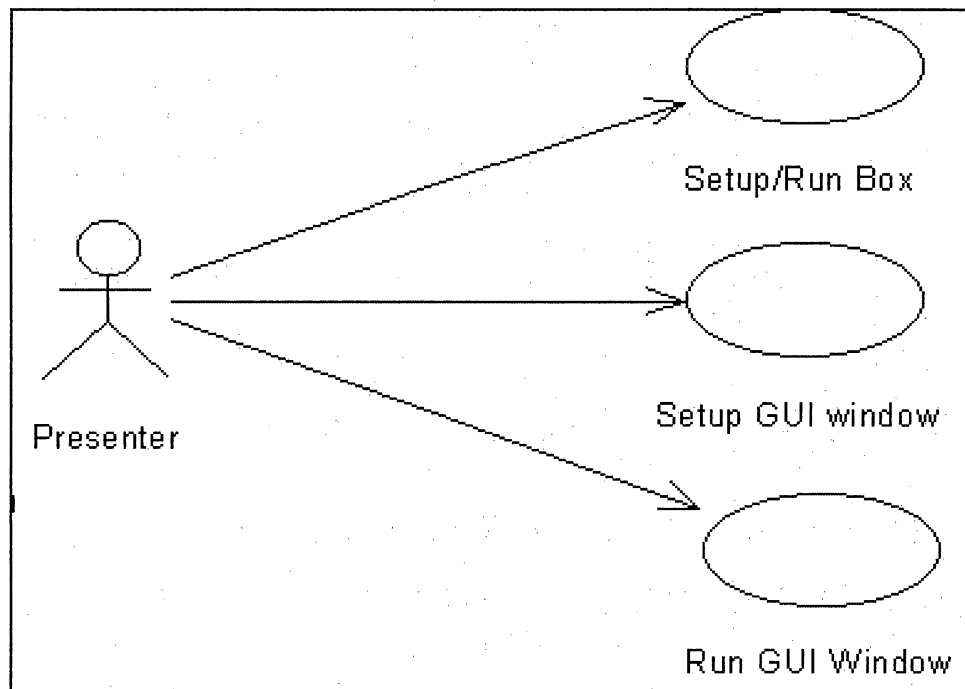
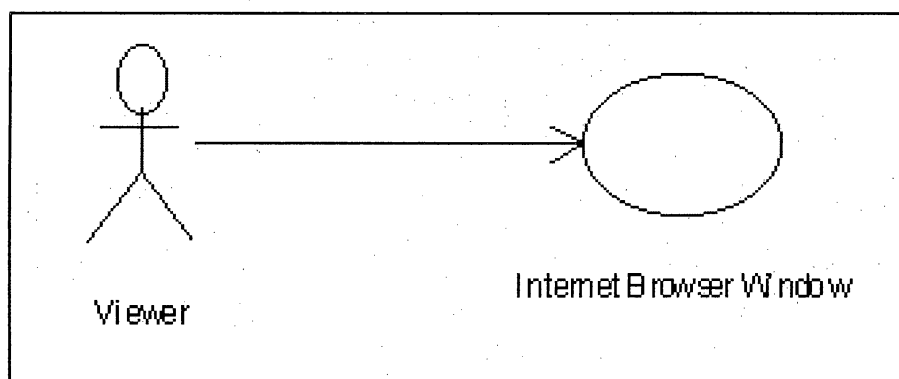


Figure 6. Viewer Use Case Diagram



1.2.2.1 Setting Up the Presentation. PWWS requires that the presenter sets up the presentation before giving the presentation. The presenter will be provided with instructions on how to setup the presentation. From following the instructions, the presenter will know which data must be supplied to the presentation for its execution.

The Setup window offers a view of the presenter's files and allows he or she to select the files for the presentation. There is an option where the files can be viewed. This provides a method to verify that the correct file has been selected or that the files selected have been placed in correct sequence for presenting.

Another option available is to save the live presentation. When the presentation is executed, the files will be saved to a directory. The files will be in the order they were actually presented and will have connecting links.

Once the presenter has finished setting up the presentation, the sequence and information can be saved for later execution or the presentation can be executed immediately.

1.2.2.2 Giving the Presentation. When the presenter begins the presentation, the first html file will be inserted into the browser panel and will be available to the viewers. If the presentation was executed immediately, when the Run window appears the first html file is already in place.

If the presentation has been saved, the Run window will be empty upon opening. The presenter will be able to view the available saved presentation files and select the appropriate one. Upon selection the first html file will be inserted into a browser window with a link to the following files in order of a live presentation.

The presentation can be shown one line at a time or the whole page. This can be chosen on a file-by-file basis. The presentation can be delivered in order by clicking the "Next Slide" button, but does not necessarily have to be shown in order. When the button is clicked the viewers will also see the new file.

The html files text can be edited and/or highlighted during the presentation. As changes are made the presenter will click the "Repost" button and the viewers will see the changes.

HTML files not included in the original sequence can be displayed by clicking "View Extra File". A window will appear displaying the files and directories. The directories may be traversed to find the extra file. It is required that the file be within the web server directory pool.

#### 1.2.3 User Characteristics

The following capabilities are assumed for the presenter. She must know the path to her web directory and the URL address to that directory. She must be able to create HTML files from some type of Web Editor tool, such as DreamWeaver or FrontPage.

The viewer is assumed to have an appropriate web browser and be Internet literate. The viewer will log onto the URL indicated by the presenter, which is the URL plus temp.html.

#### 1.2.4 Constraints

The presentation is made available through the Internet and because of this for security purposes the application must also be compatible with Internet secure communications using the Secure Socket Layer.

#### 1.2.5 Assumptions and Dependencies

These requirements assume there are no applicable hardware limitations. It is also assumed that system administration and maintenance issues will be dealt with by the host system.

### 1.3 Specific Requirements

This section contains the software requirements to a level of sophistication that include a description of every input, every output, and all functions performed by

#### 1.3.1 External Interface Requirements

This section is a detailed description of all the inputs and outputs from the software system.

1.3.1.1 Setup Page. This GUI will require the presenter to input the path to his or her web directory and the URL to this directory. This information is needed for the actual presentation, saving the presentation order or viewing a file. Error checks will verify fields.

The page will provide a file directory for the presenter to select the HTML files in the order for the presentation. If the presenter is connected through an FTP connection a separate window will display the HTML in

the web directory. She will highlight the file in the file directory panel and click the "Select" button. The files selected will appear in order in the selected files for presentation panel.

The presenter may choose to view the files in this panel, by highlighting the file name and clicking the "View" button. The file will be opened and read by the system, the system will open an additional browser window to display the file. The presenter can remove a file from the order by highlighting the file name and clicking the "Remove from Presentation" button.

If the presenter wants to make the live presentation available to students after the presentation has been given, he or she inputs a name into the specified textfield. When the presentation is given, a directory will be created. The files will be added to the directory in the order of the live presentation. These files will have links added to make them to make them easily viewable by the students.

If the presenter wants to give the presentation at a later time, he or she inputs a name into the "save presentation" textfield and clicks the "Save Presentation"

button. The web directory path, the URL, the directory name for saving the live presentation (if selected) and the sequence of file names are saved to a text file.

1.3.1.2 Run Page. If the presenter has activated this page from the Setup Page by clicking the "Run" button, the first HTML file is displayed in the browser panel. Links to the HTML files are displayed in the link panel. If the presenter has activated this page from the Setup/Run Box, the browser panel and link panel will be empty. The presenter will click the "Open Saved Presentation" button and a file directory window will appear. The user will select the presentation and click the "Open" button. The first HTML file will fill the browser panel and the URL address links to the additional HTML files will fill the link panel. If an error has been made in entering the web directory path or URL, the presenter may click the "Change Path" button and return to the Setup/Run Box window.

As the presenter gives the lecture, he or she can open the next HTML file in the sequence by clicking the "Next" button. If the presenter chooses to not go in order, the HTML files can be selected by clicking on the



file names in the link panel. As each file is displayed to the presenter, it will be displayed to the viewer.

If the presenter chooses to have an HTML file displayed one line at a time, he or she click the "Line By Line" checkbox. To display the next line, he or she clicks the "Next Line" button. This option can be turned on or off at any time before a new HTML file is selected.

If the presenter chooses to edit the text of the HTML file, he or she clicks the "Edit" button, highlights the text and makes the change. The HTML edit mode will stay functional only for that file. Upon execution of the next slide, the edit capabilities will be turned off. For the viewer to see the change, the presenter must click the "Repost" button.

The presenter may decide to show an HTML file that was not in the original sequence. The user clicks the "View Extra File" button and window displaying the files and directories available to the presenter. The available files are not required to be within the presenter's web directory, but must be within the public realm of the local web server and have public permissions.

### 1.3.2 Functional Requirements

This section will define the fundamental actions that take place in the software accepting and processing of the inputs and outputs.

1.3.2.1 Setup Page. The system shall collect the data from each of the textfields upon clicking the "Run" button or "Save the presentation button". The required data will be verified as not null. If required data is absent, the system shall display a message to the presenter requesting the required data.

The system shall collect the file names of the selected HTML files. If no files have been selected, the system shall display a message to the presenter requesting that files be selected.

The system shall collect the path from the web directory textfield and concatenate it with file names for viewing. If the presenter has input an erroneous URL, the view browser will indicate that no such file exists.

The system shall read all the textfield data, create a text file, and save the data in the text file. The text file is placed in the indicated web directory, if the presenter has chosen to click the "Save Presentation"

button. If the user has indicated an FTP connection the text file will be saved to directory that PWWS is executed from on the local machine or presenter's directory and to the web server directory. This will allow the presenter to execute the administration of the presentation either through an FTP connection or within the LAN.

1.3.2.2 Run Page. The system shall display the HTML files in the browser panel or indicate that no such file can be found, indicating an erroneous URL address or that the HTML files are not in the indicated web directory.

The system shall insert into each HTML file an applet tag so that continuity of the presentation is attained for the viewer. Upon the reselection of an HTML file by the presenter, the system shall check for previous insertion of applet tag and not make the insertion.

The system shall allow Line By Line viewing of an HTML file and indicate by displaying a message after the last line that there are no more lines. The system will parse the HTML file so that each line of text is considered a line, parsing out tags and non-breaking spaces.

The system shall highlight in the link panel the HTML files that have been displayed in the sequence and which HTML file is next. After the last HTML file has been displayed, the system will indicate with a message that the sequence presentation has ended. The system will allow the presenter to still select HTML files from the link panel.

The system shall create a directory using the data from the "Save Live Presentation" textfield as the name of the directory. The system will copy all graphic files from the web directory to the new directory. The system shall copy the live sequence of the presentation, add links to file, rename the file and place in the new directory. Renaming the files will alleviate any problems in the case an HTML file was selected more than once.

#### 1.3.3 Performance Requirements

The number of viewers that may view the PWWS is only limited by what the network and web server can handle. Performance of PWWS is limited by network traffic and speed of the web server.

### 1.3.4 Software System Attributes

1.3.4.1 Maintainability. All modules will be discriminated by class name. A group of classes that labor in conjunction with each other to achieve a particular task will be held in a directory, thus yielding a hierarchy of directories and class modules. This structure will aid in maintaining all modules organized and therefore maximizing maintenance facility.

1.3.4.2 Portability. PWWS will be 100% portable since it will be written in Java, a proven portable language. The only determinant of how easily the PWWS is ported from one architecture to another is having the latest version of the Java Virtual Machine installed on the web-server machine.

## CHAPTER TWO

### SOFTWARE DESIGN

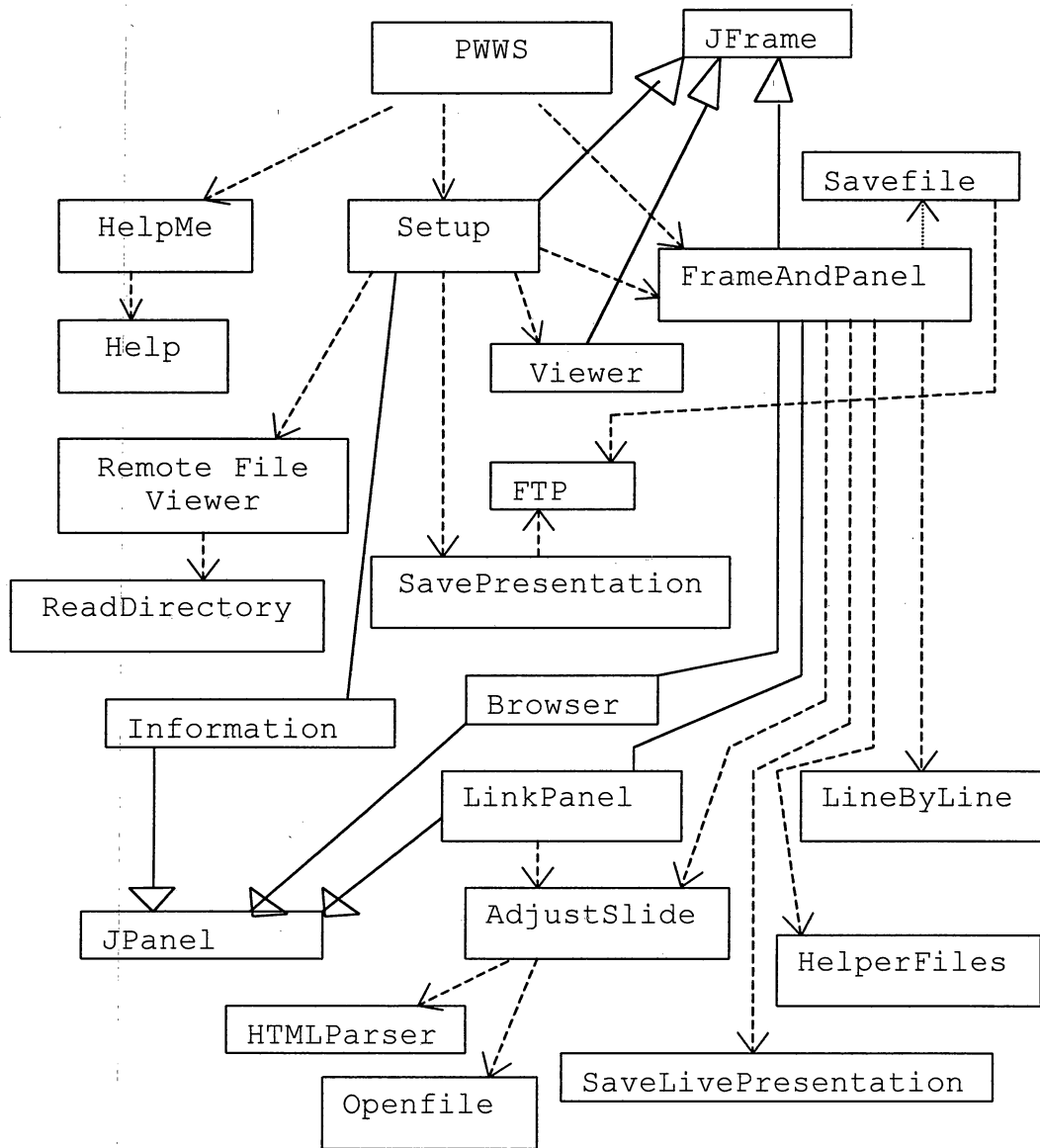
#### 2.1 Preliminary Design

##### 2.1.1 Use Case Diagram

The Use-case modeling of the product is without regard to sequencing. This step is largely action oriented. It is concerned with the interaction between the classes of the software product and the users of the product. See Figure 5 and 6.

### 2.1.2 Class Diagram

Figure 7. Class Diagram



## 2.2 Detailed Design

### 2.2.1 Setup Design

Table 1. Setup

Class Name	Setup
Where Used	PWWS
Purpose	Display Setup GUI
Sub Items	Information Panel, Required Textfields, File Directory Panel, Files Selected Panel, Button Panel
Notes	Initial GUI for PWWS

#### Procedure Setup

##### Begin

```
Declare GridBagLayout
Declare GridBagConstraints
Set Border
Add Heading
Add Information Panel
Add Required Textfields
Add File Directory Panel
Add Files Selected Panel
Add Button Panel
```



End

#### 2.2.1.1 Information Design.

Table 2. Information

Class Name	Information
Where Used	Setup GUI
Purpose	To provide panel with explanation of how to use PWWS
Sub Items	None
Note	Non-Editable

#### Procedure Information

Begin

Declare Heading String

Declare Data String

Create Information Panel

Create Heading Panel

Create Data Panel

Insert Heading and Data Strings

Insert Heading and Data Panel

Return Information Panel

End

### 2.2.1.2 Required TextField Design.

Table 3. Textfields

Class Name	Textfields
Where Used	Setup, Java Class
Purpose	Create Panel for required fields
Sub Items	None
Notes	JPanel

Procedure textfields

Begin

    Declare GridBagLayout

    Declare GridBagConstraints

    Set Border

    Add JLabels

    Add JTextFields

End

### 2.2.1.3 File Directory Design.

Table 4: JFileChooser

Class Name	JFileChooser
Where Used	Setup, Java Class
Purpose	Provide method for presenter to access their HTML files.
Sub Items	ActionListener
Notes	Adds filenames to Files Selected Panel

#### Procedure JFileChooser

Begin

Set Border

Set ToolTipText

Set ApproveButton

Add ActionListener

If selected

Add filename to list in Files Selected Panel

End If

End

#### 2.2.1.4 Files Selected Design.

Table 5. Files

Class Name	TextArea
Where Used	Setup, Java Class
Purpose	Create text area to display selected HTML files.
Sub Items	Viewer, Mouse Listener
Notes	Non-Editable

Procedure files

Begin

    Declare ScrollPane

    Add TextArea

    Add MouseListener

    If filename is selected

        Highlight text

    End If

End

### 2.2.1.5 Button Panel Design.

Table 6. Buttons

Class Name	Buttons
Where Used	Setup, Java Class
Purpose	Create a panel for buttons and non-required text fields.
Sub Items	ActionListener
Notes	None

#### Procedure Buttons

Begin

Declare GridLayout

Create 3 Panels

Add Buttons and Text Fields

Add ActionListener

If Save Presentation Button selected

Verify Directory Path text field not null

Verify URL Path text field not null

Verify Presentation Name text field not null

Verify files text area is not null

If all text fields not null

Concatenate directory path with presentation name.

```

        Save all information to file presentation.txt in
            web directory.

Else

    Display message indicating required fields.

End if

Else if Delete From Presentation List button

    Remove selected text.

Else if Viewer button

    If URL text field is not null

        Concatenate URL with filename

        Send to Viewer Panel

    Else

        Display message requesting URL field.

    End if

Else if Run button

    Verify Directory Path text field not null

    Verify URL Path text field not null

    Verify Presentation Name text field not null

    Verify files text area is not null

    If all fields not null

        Set required fields in Run GUI

        Set browser panel in Run GUI

```

Set link panel in Run GUI

Else

Display message indicating required fields.

End If

End If

#### 2.2.1.5.1 Viewer Panel Design.

Table 7. Viewer

Class Name	Viewer
Where Used	Setup -> Button Panel
Purpose	To Display an HTML file for verification
Sub Items	None
Notes	Uses JEditorPane

Procedure Viewer

Begin

Declare JEditorPane

Declare BorderLayout

Open window and display HTML file

End

### 2.2.2 Run Design

Table 8. FrameAndPanel

Class Name	FrameAndPanel
Where Used	PWWS
Purpose	Create panel to execute presentation
Sub Items	Link Panel, Browser Panel, Button Panel, and Drawing Panel
Notes	Drawing Panel interface

#### Procedure FrameAndPanel

Begin

    Declare BorderLayout

    Add Browser Panel to center panel

    Add Button Panel to south panel

    Add Link and Drawing Panel to west panel

    Set required fields

    Add WindowListener

    If window closes and save live presentation chosen

        Read file from browser panel

        Save file to Save Live Presentation directory

        Close window

Else



Close window

End if

End

Table 9. FillBrowser

Method Name	fillBrowser
Where Used	FrameAndPanel
Purpose	Upon execution of Run GUI, Verify for Save Live Presentation option, verify for Line by Line option, verify applet tag inserted
Sub Items	None
Notes	None

Procedure fillBrowser

Begin

Get HTML file name

If Save Live Presentation is selected

Place file into Save Live directory

End if

Add Applet tag

If Line by Line option is selected

Display only first line of HTML file

```

Else
    Display whole HTML file
End if
End

```

Table 10. SetDirectory

Method Name	setDirectory
Where Used	FrameAndPanel
Purpose	Verify web directory ending, set directory path in FrameAndPanel, LinkPanel and AdjustSlide classes
Sub Items	
Notes	

Procedure setDirectory

Begin

    If directory path ends in "/"

        Set global directory path

    Else

        Add "/" to directory path

        Set global directory path

    End if

Set global directory path in LinkPanel

Set global directory path in AdjustSlide  
End

Table 11. DisperseData

Method Name	disperseData
Where Used	FrameAndPanel -> Open button
Purpose	To disseminate the data of a saved presentation
Sub Items	
Notes	

Procedure disperseData

Begin

Receive data from saved presentation text file

Parse data

Add URL to filenames and send to the link panel

Get directory path for setDirectory

Get URL path for setURL

Get Save Live Presentation for setSaveLive

Send file name to fillBrowser

End

Table 12. PostingLineByLine

Method Name	postingLineByLine
Where Used	FrameAndPanel -> fillBrowser, slide, and LinkPanel
Purpose	Set vector for posting line by line of HTML file to browser pane
Sub Items	
Notes	

Procedure postingLineByLine

Begin

Get vector of parsed lined of HTML file

Get where body of HTML file begins

Get where end of body of HTML file begins

Create a LineByLine file with adjusted number of lines

Add LineByLine file to browser pane

End

Table 13. SaveLiveAttributes

Method Name	saveLiveAttributes
Where Used	FrameAndPanel
Purpose	Set up save live presentation directory
Sub Items	SystemCall
Notes	Future FTP

#### Procedure saveLiveAttributes

Begin

Create new directory for Save Live Presentation

Copy all jpeg and gif files to Save Live Presentation

Set directory path for Save Live Presentation

Set URL path for Save Live Presentation

Set LinkPanel for Save Live Presentation

Save first file to Save Live Presentation directory

Add link to first file

End

### 2.2.2.1 Link Panel Design.

Table 14. LinkPanel

Class Name	LinkPanel
Where Used	FrameAndPanel
Purpose	Create active links of HTML files that will display in browser panel
Sub Items	HyperLinkListener
Notes	Browser panel as argument  Uses AdjustSlide

#### Procedure LinkPanel

Begin

    Declare BorderLayout

    Declare JEditorPane

    Add HyperLinkListener

    If filename is selected

        Open file

        Add applet tag

        If Save Live Presentation selected

            Add filename

            Add link

    End If

```

    If Line By Line selected
        Fill browser panel one line
    Else
        Fill browser panel whole file
    End If
End If
End

```

#### 2.2.2.2 Browser Panel Design.

Table 15. Browser

Class Name	Browser
Where Used	FrameAndPanel
Purpose	To display HTML files as presented
Sub Items	None
Notes	JInternalFrame for GlassPane

Procedure Browser

```

Begin
    Declare JInternalFrame
    Declare JEditorPane
    Declare GlassPane
    If display HTML file
        Open HTML file in browser panel
    End If
End

```

Else

Browser panel is blank

End If

If editing is selected

Set browser panel to editable

Set GlassPane to off

End If

If drawing is selected

Set browser panel to not editable

Set GlassPane to on

End if

End

### 2.2.2.3 Button Panel Design.

Table 16. Button Panel

Panel Name	Button panel
Where Used	FrameAndPanel
Purpose	To execute options for the browser panel
Sub Items	ActionListener, ItemListener,
Notes	



Procedure buttonPanel

Begin

    Declare GridLayout

    Add buttons

    Add ItemListener

    Add ActionListener

    If radioButton selected

        link panel set Line by Line to selected

    Else

        link panel set Line by Line to not selected

    End if

    Else If Repost button selected

        get data from browser window

        save data to temporary file

    End if

    Else If Next Line button selected

        If Line by Line is selected

            increase Line-by-Line vector index

            create new LineByLine file

            If the start vector index > end vector index

                Show message indicating no more lines

        End if

```

        Place new LineByLine file in browser pane

    Else

        Display message indicating option not    chosen

    End if

Else If Next Slide button selected

    If no addition HTML files in sequence

        Display message indicating presentation has ended

    Else

        Get next HTML file name

        Add Applet tag

        If Save Live Presentation selected

            Add file to Save Live Presentation directory

        End if

        If Line by Line selected

            Add LineByLine file to browser panel

        Else

            Add HTML file to browser panel

        End if

        Highlight files in link panel

        Increase vector index

    End if

Else If Edit button is selected

```

```

If Line by Line selected
    Display message editing not allowed
Else
    Edit the browser panel
End If

Else If Open button is selected
    Open JFileChooser
    Open selected file
    Read file
    Disperse data
End If

End

```

### 2.2.3 Parser Design

Table 17. HtmlParser

Class Name	HtmlParser
Where Used	FrameAndPanel
Purpose	For Line by Line type presentation
Sub Items	
Notes	

Procedure HtmlParser

Begin

```

Declare a Vector

Initialize flags and body Indexes

End

```

Table 18. GrabLines

Method Name	grabLines
Where Used	HtmlParser
Purpose	Create a vector for creating Line by Line files
Sub Items	
Notes	is passed file as a string

Procedure grabLines

Begin

    Declare StringTokenizer for lines

    Declare StringTokenizer for words

    While lines available

        While words available

            If word is "<"

                Starting a tag

            Else If word is ">"

                Ending a tag

            Else If word is body and a tag

```

    Set bodyStarts index
Else If word is /body and a tag
    Set bodyEnds index
Else if not in a tag and in the body
    If not a blank line
        If non-breaking spaces
            If also text
                Indicate text found
            End If
        Else
            Indicate text found
        End If
    End If
Else If not in body
    If not a complete line
        Create complete line
        Add complete line to vector
    Else
        Add line to vector
    End If
Else If not complete line and in body
    Place in temporary string

```

```

Else If complete line but no text found
    Place in temporary string
Else If complete line, is text, and in body
    If temporary string is empty
        Place line in vector
    Else
        Create complete line
        Place complete line in vector
    End If
End If
End If
End While
End While
End

```

#### 2.2.4 Open Files Design

Table 19. Openfiles

Class Name	openfiles
Where Used	FrameAndPanel, AdjustSlide, and SaveLivePresentation
Purpose	Open files and get data
Sub Items	
Notes	

Procedure openfiles

Begin

    If name of file is not null

        Open file input stream

        Read file data

        Close file

        Return file data

    End If

End

#### 2.2.5 Save Files Design

Table 20. Savefile

Class Name	savefile
Where Used	AdjustSlide, LineByLine, SaveLivePresentation, and SavePresentation
Purpose	To save a file
Sub Items	
Notes	

Procedure savefile

Begin

```

Create a file

Write data to file with buffered writer

Flush data

Close file

End

```

#### 2.2.6 Save Live Presentation Design

Table 21. SaveLivePresentation

Class Name	SaveLivePresentation
Where Used	FrameAndPanel
Purpose	To provide slides for viewing after the live presentation.
Sub Items	
Notes	

#### Procedure SaveLivePresentation

```

Begin

Create slide name

Create indexing of slides

End

```



Table 22. AddFile

Method Name	addFile
Where Used	FrameAndPanel
Purpose	Open a file, read data, adjust directory path
Sub Items	
Notes	

Procedure addFile

Begin

Open the HTML file

Get the data

Define Save Live Presentation directory path

Remove the Applet tag

End

Table 23. AddLink

Method Name	addLink
Where Used	FrameAndPanel
Purpose	When next HTML file is chosen, add link to previous slide to this file
Sub Items	
Notes	

Procedure addLink

Begin

Find the end of the body of the HTML file

Insert a link to the next slide before the end of body

Save the file to the Save Live Presentation directory

End

### 2.2.7 Adjust Slide Design

Table 24. AddApplet

Method Name	addApplet
Where Used	FrameAndPanel
Purpose	Add the refresh Applet tag to the HTML files and place parsed HTML file into vector for possible line-by-line viewing.
Sub Items	
Notes	

Procedure addApplet

Begin

Open file

Create temporary file directory and URL paths

Find end of head tag

If no previous insertion of Applet tag

Insert Applet tag before end of head tag

Insert Div tags for layers

Insert Java script for browser styles

End If

Have data parsed and placed into vector

Save HTML file with Applet tag for browser panel

Save HTML file with Applet tag for viewers

End

## CHAPTER THREE

### SOFTWARE QUALITY ASSURANCE

#### 3.1 Unit Test Plan

All unit testing will conform to the general requirements defined in this section as follows:

- Each unit will compile and assemble without error.
- Execution errors, such as illegal memory accesses and overflow/underflow conditions, will not occur.
- Each executable statement will execute at least once.
- Each decision will take on true/false values at least once.
- Each unit will be tested using nominal, zero, extreme, and erroneous values for each input parameter.

The following checklist was used as the Unit Test Plan for this project and applied to each Unit. See Table 25.

Table 25. Unit Test Plan

Check all path and both sides of all branches	
Ensure that all instructions execute	
Verify operation at normal parameter values	
Verify operation at limit parameter values	
Verify operation outside of limit parameter values	
Check the use of all called objects	
Verify the handling of all data structures	
Verify the handling of all files	
Check normal termination of all loops	
Check abnormal termination of all loops	
Verify the handling of all error conditions	
Check timing and synchronization	
Verify all hardware dependencies	

### 3.2 Integration Test Plan

Integration testing provides a level of confidence that the software comprising all the units performs in accordance with the requirements specified in the SRS. The top-down implementation and integration method was chosen for this project. This type of testing allows testing on

the top module first. Once this module has passed a specific test the same test is executed after the integration of the top and next module. If the test fails it will indicate the error is either in the interface between the two modules or in the second module. This is continued until all modules have been integrated.

Test cases are developed to include the following:

- Tests to demonstrate that the units execute together when linked.
- Tests to evaluate that control paths are supported.
- Tests to evaluate interfacing between units and the passing of data values between units.
- Tests to evaluate performance attributes, such as error recognition, and recovery.

The integration testing will conform to the following general requirements:

- Units contained in the program will link without error.
- Unit interfaces will be exercised.

Table 26. Test Case 1 - Setup/Run Box

Action	Result	Verified
User clicks Setup button.	Setup GUI opens.	
User clicks Run the Presentation button.	Run GUI opens with blank browser window and blank link panel.	



Table 27. Test Case 2 - Setup Window

Action	Result	Verified
<p>Presenter clicks</p> <p>Save button</p> <p>1. URL path, web directory, and save presentation name not provided.</p> <p>2. URL path not provided.</p> <p>3. Web directory path not provided.</p> <p>4. Save presentation name not provided.</p> <p>5. No files selected for presentation.</p> <p>6. All required fields provided.</p> <p>7. Save Live</p>	<p>1. Message displayed indicating all missing fields.</p> <p>2. Message displayed indicating URL missing field.</p> <p>3. Message displayed indicating web directory missing field.</p> <p>4. Message displayed requesting presentation name.</p> <p>5. Message displayed requesting files.</p> <p>6. A text file (presentationName.txt) written to web directory with the required data.</p>	

<p>Presentation name is provided.</p>	<p>7. A text file (presentationName.txt) written to web directory with the required data and the Save Live Presentation name.</p>	
<p>Presenter clicks Run the Presentation button:</p> <ol style="list-style-type: none"> <li>1. URL path, web directory, and save presentation name not provided.</li> <li>2. URL path not provided.</li> <li>3. Web directory path not provided.</li> <li>4. Save presentation name not</li> </ol>	<ol style="list-style-type: none"> <li>1. Message displayed indicating all missing fields.</li> <li>2. Message displayed indicating URL missing field.</li> <li>3. Message displayed indicating web directory missing field.</li> <li>4. Message displayed requesting presentation name.</li> <li>5. Message displayed requesting files.</li> </ol>	

<p>provided.</p> <p>5. No files selected for presentation.</p> <p>6. All required fields provided.</p> <p>7. Save Live Presentation file name provided.</p>	<p>6. The Run GUI window opens with links and first slide visible, all required fields are initialized.</p> <p>7. The Run GUI window opens with links and first slide visible, all required fields are initialized, and a directory is created in the presenter's web directory with the Save Live Presentation name.</p>	
<p>Presenter clicks the Viewer button:</p> <p>1. File name is not highlighted and</p>	<p>1. Viewer window opens indicating no such file.</p> <p>2. Viewer window</p>	

<p>URL is correct.</p> <p>2. File name is highlighted and URL is correct.</p> <p>3. URL is incorrect.</p>	<p>opens displaying the HTML file.</p> <p>3. Viewer window opens indicating no such file.</p>	
<p>Presenter clicks the Exit button.</p>	<p>Setup GUI window closes</p>	
<p>Presenter click the Remove from Presentation button:</p> <p>1. No file name is selected.</p> <p>2. File name is selected.</p>	<p>1. Nothing is removed</p> <p>2. File name selected is removed.</p>	
<p>Presenter clicks Select button:</p> <p>1. File name is selected.</p> <p>2. File name is not selected.</p>	<p>1. File name is placed is the Files Selected for Presentation panel.</p> <p>2. Nothing occurs.</p>	

Table 28. Test Case 3 - Run Window

Action	Result	Verified
Presenter clicks the Open Saved Presentation button.	A window pops up and displays the presenter's home directory.	
<p>Presenter clicks the Open button:</p> <p>1. Saved presentation is selected.</p> <p>2. Saved presentation is not selected.</p>	<p>1. The first HTML file in the presentation is displayed in the browser panel and to the viewer.</p> <p>The list of links to the HTML files are displayed in the link panel.</p> <p>2. Nothing occurs.</p>	
<p>Presenter clicks the Next Slide button:</p> <p>1. A HTML file is available and the line-by-line</p>	<p>1. The next HTML file is displayed in the browser panel and to the viewer.</p> <p>The link panel</p>	

<p>option is not selected.</p> <p>2. A HTML file is available and the line-by-line option is selected.</p> <p>3. No HTML file is available.</p> <p>4. Save Live Presentation option was selected.</p>	<p>indicates the next file that is being displayed.</p> <p>2. The first line of the next HTML file is displayed in the browser panel and to the viewer. The link panel indicates the next file that is being displayed.</p> <p>3. A message indicating the presentation has ended is displayed.</p> <p>4. HTML file is copied to the Save Live Presentation directory and link is added.</p>	
---	--	--

<p>Presenter clicks the Next Line button:</p> <ol style="list-style-type: none"> <li>1. Line-by-Line not selected.</li> <li>2. File displaying Line-by-Line and Line-by-Line selected.</li> <li>3. Whole file displayed and Line-by-Line selected.</li> </ol>	<ol style="list-style-type: none"> <li>1. Message indicating Line-by-Line option not selected.</li> <li>2. The next line is displayed in browser panel and to viewer.</li> <li>3. Message indicating no more lines available.</li> </ol>	
<p>Presenter clicks the Edit button:</p> <ol style="list-style-type: none"> <li>1. Line-by-Line option is not selected.</li> <li>2. Line-by-Line option is selected.</li> </ol>	<ol style="list-style-type: none"> <li>1. Presenter is able to edit text on browser window.</li> <li>2. Message is displayed indicating editing is not allowed with the Line-by-Line option.</li> </ol>	
<p>Presenter clicks the</p>	<p>Data is read from</p>	

Repost button.	browser panel and displayed to viewer.	
<p>Presenter clicks a link in the link panel:</p> <ol style="list-style-type: none"> <li>1. Line-by-Line option has been selected.</li> <li>2. Line-by-Line option has not been selected.</li> <li>3. Save Live Presentation was selected.</li> </ol>	<ol style="list-style-type: none"> <li>1. The first line of the HTML file selected is displayed in browser and to viewer.</li> <li>2. The whole HTML file selected is displayed in browser and to viewer.</li> <li>3. The HTML file is copied to the Save Live Presentation directory and a link is added.</li> </ol>	



### 3.3 System Test Plan

The system test plan intends to prove that:

- The functionality that is as specified by the Software Requirements Specifications Documentation is met.
- The software is of high quality and achieves the standards required.
- The software interfaces correctly with existing systems.

See Table 29.

Table 29. System Test

Presenter System	Viewer System	Verified
Linux RedHat 7.0	Netscape 4.0+ on Linux System, within LAN	
Linux RedHat 7.0	Internet Explorer 5.0+, on Windows System, within LAN	
Linux RedHat 7.0	Netscape 4.0+ on Windows System, across WAN	
Linux RedHat 7.0	Internet Explorer 5.0+ on Windows System, across WAN	

## CHAPTER FOUR

### MAINTENANCE

Compiler: JDK1.3 or Higher

Library: JDK1.3 Packages or Higher

Operating System: Linux RedHat 7.0

Web Server: Unix Operating System

Final Version Source Files: PWWS CD.

Java Doc: PWWS CD.

Class Files: PWWS CD.

Executable Program: PWWS CD.

PWWS CD: CSUSB Computer Science Department.

## CHAPTER FIVE

### USERS MANUAL

#### 5.1. System Requirements

The Presenter of the Presentations World Wide Systems (PWWS) will need to create HTML files for a presentation before using the system. These HTML files can be created with any HTML editor, such as MacroMedia's DreamWeaver or MicroSoft's FrontPage. The creator can create these files by hand, but is required by the PWWS to have certain tags present in the code. These are: `<head>`, `</head>`, `<body>`, and `</body>`. The HTML files should be uploaded to the presenter's web directory beforehand, along with any graphics files that pertain to the presentation. Once these preliminary files are created and uploaded, the presenter is able to use PWWS. If the user places links to other web pages within the created HTML files, the links will be active to both the presenter and the viewer.

The Presentations World Wide System has been developed to run on a Windows or Linux RedHat operating system. This system must be connected to a web server through a local area network or an FTP connection. This system must be able to execute Java programs. The viewers

of the system are not required to be within the local area network, but are required to be able to connect to the Internet through a Java enabled browser such as, Netscape or Internet Explorer. The viewer of the presentation is not required to download additional software.

### 5.2. Starting the Program

The presenter will execute the PWWS by typing "PWWS" at the command line on a Linux operating system. A box displaying three buttons will appear: the User's Manual, Setup the Presentation and Run the Presentation. If the presenter has not created a presentation through the Setup portion at a previous time, then this step must be implemented first. The presenter is required to input the URL of the web directory for the presentation and also the local directory path for the presentation in the textfields. If an FTP connection is required to connect to the web server, the presenter must enter the FTP address, the login and password.

### 5.3. Setup Presentation

The required URL of the web directory for the presentation, the local directory path for the

presentation, the FTP address, login, and password will appear in the right upper portion of the Setup GUI. In the left center portion of the GUI, the presenter can access her web directory if connected within a LAN. With an FTP connection a window will appear with the HTML files displayed that are located in the specified web directory. The presenter should select the HTML files in the sequence that is desired, by highlighting each HTML file and click the Select button. The HTML files will be displayed in the Files Selected for Presentation panel, in the sequence chosen.

To verify the correct sequence or verify the file, highlight the file in the Files Selected for Presentation panel and click the "View File" button. The file will be previewed in another window, only if URL path is correct.

To remove a file from the selection of files for presentation, highlight the file in the Files Selected for Presentation panel and click the "Remove from Presentation" button. The file name will be removed.

To provide the viewers with the opportunity to view the Live Presentation after the fact, a name is entered into the indicated input field at the bottom of the Setup

GUI window. This name will be used to create a directory in the presenter's web directory for the recorded presentation.

To save the URL, web directory path, files selected for the presentation and option of saving the live presentation for later execution; place a title for the presentation in the indicated box at the bottom of the Setup GUI window and click the "Save" button. This will save the information to the presenter's web directory and can be opened for later execution.

To execute the presentation immediately, click the "Run Presentation" button. This opens the Run GUI window with the first HTML file displayed in the browser panel, the available links in the link panel, and the first HTML file is available to the viewers at the requested URL/JoinPresentation.html which provides button link.

#### 5.4. Run Presentation

If the presenter has entered from the Setup GUI Window, then skip to the next paragraph. If the presenter has entered the Run Window from clicking the "Run Presentation" button in the Setup/Run box, continue with this paragraph. The presenter must have saved the

information required for the presentation at a previous time. To execute the presentation, click the "Open Saved Presentation" button at the bottom of the window. A file directory to the presenter's web directory will pop up. The presenter should select the "presentation.pwvs" file and click the "Open" button. The first HTML file will be displayed in the browser panel, the available links in the link panel, and the first HTML file is available to the viewers at the requested URL/JoinPresentation.html.

If the presenter wants to present the HTML file one line of text at a time, click the Line-by-Line radio button, and click the link to the first HTML file. (This option could have been selected previous to opening a Saved Presentation). To show another line, click the "Next Line" button. This option can be turned off at any time. If the option is no longer desired in the middle of a Line-by-Line showing of an HTML file and the whole file is desired; turn off the option and click the corresponding (the last highlighted) link in the Link Panel. The whole HTML file will be displayed.

If the presenter wishes to change the text during the showing, click the "Edit" button and make changes. Some



changes make be difficult within the HTML pages. The presenter also has the option of highlighting text. For the viewers to see the changes or highlighting, the presenter must click the "Repost" button, once the changes are made.

To show the next HTML file in the sequence, the presenter clicks the "Next Slide" button. The HTML file will be displayed in the browser panel and to the viewer.

If the presenter wishes to go out of sequence, the file can be chosen by clicking on the corresponding link in the Link Panel. The file will be displayed in the browser panel and to the viewer. If the presenter decides to present a file that was not included in the original sequence, the presenter clicks the "View Extra File" button. A window displaying files and directories are displayed. The presenter may choose any HTML file that is available to the public within the LAN

If the presenter provided links within the HTML pages, the links are active to both the presenter and viewer. The viewer will need to either click the right mouse button and choose back or return to the presentation from the JoinPresentation.html file. This will return to

the presentation in progress. The presenter needs to click the "Next Slide" button or a slide in the link panel to return to the presentation.

## CHAPTER SIX

### CONCLUSION, LIMITATIONS AND FUTURE DIRECTIONS

The Presentations World Wide System provides students and instructors with a method of using the Internet to enhance the learning experience. It is developed to be user-friendly and self-explanatory. Once a library of HTML files for instruction is developed, the PWWS makes it a simple matter to create and give presentations. The options available in the PWWS make the presentation alterable during showing if necessary or if desired. This makes the PWWS an attractive tool for presentations. With the slide creation of the presentation, the PWWS is helpful to students to review. PWWS supports student learning in preparing for tests. This combination makes the PWWS beneficial to instructors and students.

The Presentations World Wide System was initially created for use on any LAN with a Unix type operating system. The addition of being able to execute most options through an FTP connection has expanded the utilization of the PWWS. Minor adaptations using file transfer protocol (FTP) are being incorporated to include these option in the PWWS system for use across a WAN.

Enabling the presenter to Setup and Run the PWWS remotely, allowing more freedom for the presenter.

The PWWS is limited by the use of the JEditorPane class in Java for the browser panel in the Presentation GUI. This class does not allow the presenter to see the execution of applets within the presenter's browser panel. The HTML parser class was implemented to handle HTML4 tags and has not been tested using other versions of HTML. The introduction of graphics in the PWWS created the need to make the size of the viewer's Internet browser to be resized to a static size of 800 x 600. The HTML files created for presentation must fit this size Internet browser window.

Additionally the PWWS is incorporating an audio feature that will allow live streaming audio of the presentation to those viewers not in the classroom and for the presenters. Taping of the audio stream would benefit the slide presentation and enhance a review. This would require a plug-in downloaded to the viewer's computer, such as RealAudioPlayer, which is available at no expense. A RealAudio server would be required for the network server to provide this option.

APPENDIX A:  
SOURCE CODE OF PRESENTATIONS WORLD WIDE SYSTEMS

```

//Author: Sandra Hengstebeck
//Class Name: AdjustSlide.java
//This class program add the refresh AppletTag to the HTML
//file. It also calls the HTMLParser to place the file
//into a Vector for line-by-line execution. Also inserts
//layer tags for graphics.

```

```

import java.awt.*;
import java.io.*;
import java.util.*; //Vector

```

```

public class AdjustSlide{
    private String filePath, appletPath, tempPath;
    private String linePath;
        //http:... and web-directory path
    private String urlPath, path;
    private String slideData, startData, endData;
    private String adjustedData;
    private String appletString= new String("<APPLET HTTP" +
        "-EQUIV=\"Refresh\" "+ "CONTENT=\"5; URL=");
    private savefile RS, S;
    private String temp = new String("temp.html");
    private HtmlParser hp = new HtmlParser();
    private Vector lineVector = new Vector();
    private int startIndex, endIndex;

```

```

AdjustSlide()
{}

```

```

public void addApplet(String fileName)
{
    //directory path and orig filename
    filePath = path.concat(fileName);
    linePath = path;
        //open the file
    openfile O = new openfile(filePath);
        //directorypath + "temp.html"
    tempPath = path.concat(temp);
        //urlpath + "temp.html"
    appletPath = urlPath.concat(temp);
    slideData = O.getFileData(); //get the code

    /*** For inserting appletTag ***/
        //get where end of head is

```

```

int indexEndHead = slideData.indexOf("</head");
    //get code before end of head
startData = slideData.substring(0, indexEndHead);
    //get code after end of head
endData = slideData.substring(indexEndHead);

/**** UNIX ****/
//add refresh data to head and rename file temp
    //no refresh tag yet
if(slideData.indexOf(appletString+appletPath) == -1)
    adjustedData = startData.concat
        (appletString + appletPath +
         "\"> \r\n" + endData);
else    //refresh tag already there
    adjustedData = slideData;

/**** For inserting layerTags ****/
int bodyB = adjustedData.indexOf("<body");
int bodyBegins = adjustedData.indexOf(">", bodyB);
int bodyEnds = adjustedData.indexOf("</body");

String part1 = adjustedData.substring(0,
    bodyBegins+1);
String part3 = adjustedData.substring
    (bodyBegins+1, bodyEnds);
String part5 = adjustedData.substring(bodyEnds);
String part2 = new String("<layer NAME=\"one\" \" +
    \"LEFT=25 TOP=25 Z-INDEX=1>");
String part4 = new String("</layer>"+
    "<layer NAME=\"two\" LEFT=25 TOP=25 Z-INDEX=2>"+
    "<img src= \"myImg.jpg\">");
String adjustedData2 = new String();
if(adjustedData.indexOf(part2) == -1)
    adjustedData = part1 + part2 + part3 +
        part4 + part5;

/**** Browser.setPage() would not refresh with temp,
    added RS ****/
HtmlParser hp = new HtmlParser();
hp.grabLines(adjustedData);
lineVector = hp.getLines();
startIndex = hp.firstLineIndex();

```

```

        endIndex = hp.lastLineIndex();

        //slide used for presenter.
        RS = new savefile(filePath, adjustedData);
        //slide used for viewer.
        S = new savefile(tempPath, adjustedData);
    }

    public Vector getLineVector()
    {
        return lineVector;
    }

    public int getFirstIndex()
    {
        return startIndex;
    }

    public int getLastIndex()
    {
        return endIndex;
    }

    public String getTempPath()
    {
        return tempPath;
    }

    public void setDirectoryPath(String dp)
    {
        path = dp;
    }

    public void setURLPath(String up)
    {
        urlPath = up;
    }
}

//Author: Sandra Hengstebeck
//Class name: Browser.java
//This class program provides the panel that displays the
//html files to the presenter as the viewer sees it. The
//presenter can edit the page and repost it.

```



```

import javax.swing.*;    //JPanel
import java.awt.*;       //BorderLayout
import java.io.*;        //IOException
import java.awt.event.MouseEvent;

public class Browser extends JPanel
{
    String theData, theWebPath;
    JEditorPane jt;
    JInternalFrame jf;
    JComponent jc;
    JScrollPane pane;
    final JTextField input;
    DrawingPanel d;

    /*** Is called by FrameAndPanel.java    ***/
    /*** This function provides a blank panel    ***/
    Browser(DrawingPanel D)
    {
        d = D;
        jf = new JInternalFrame("Browser Panel");
        setLayout (new BorderLayout (5, 5));
        setBorder
            (BorderFactory.createLoweredBevelBorder());

        jt = new JEditorPane();
        input = new JTextField();
        // make read-only set false
        jt.setEditable(false);
        jc = (JComponent)jf.getContentPane();
        jc.setLayout(new BorderLayout());
        jc.add(jt, BorderLayout.CENTER);
        pane = new JScrollPane();
        pane.setBorder
            (BorderFactory.createLoweredBevelBorder());
        pane.getViewport().add(jf);
        add(pane, BorderLayout.CENTER);

        add (input, BorderLayout.SOUTH);
        setSize(800,550);
        setVisible(true);
        jf.setVisible(true);
    }
}

```

```

/**** This function inserts the URL file into the panel
Is called by FrameAndPanel.java
and LinkPanel.java ****/
public void setPresentation(String webpath)
{
    try
    {
        jt.setEditable(false); //not editable
        //needed for saving Live presentation
        theWebPath = webpath;
        jt.setPage(webpath); //add the html file to panel
        //add the name of file to textfield
        input.setText(webpath);    }
    catch(IOException i)
    {
        System.out.println("exception in Browser"+
            "occurred");
    }
}

/**** This function is selected when the presenter
click the Edit Button. Is called in
FrameAndPanel.java ****/

public void editPage()
{
    jf.setGlassPane(d);
    jf.getGlassPane().setVisible(false);

    jt.setEditable(true);
}

/**** This function is selected when the presenter
clicks the Draw button. It is called in
FrameAndPanel.java **/

public void draw()
{
    jt.setEditable(false);

    jf.setGlassPane(d);
    jf.getGlassPane().setVisible(true);
}

```

```

/**** This function is executed when the presenter
clicks repost. Is called in FrameAndPanel.java ****/

    public String getData()
    {
        theData = new String(jt.getText());
        return theData;
    }

/**** This function is executed when exiting the
programming, if the presenter has chosen to save
the live presentation.
Is called in FrameAndPanel.java ****/

    public String getFileShowing()
    {
        int i = theWebPath.lastIndexOf("/");
        String s = theWebPath.substring(i+1);
        return s;
    }
}

//Author: Sandra Hengstebeck
//Class Name: FrameAndPanel.java
//This class program provides the primary GUI and Button
//interactions

import java.awt.*;           //GridBagLayout
import java.awt.event.*;     //actionPerformed
import javax.swing.*;        //JFileChooser
import java.util.*;          //StringTokenizer
import java.io.*;            //File

// Subclass JFrame so you can display a window
public class FrameAndPanel extends JFrame {

    int maxLBLfiles = 4; //maximum Line By Lines allowed
// Set up constants for width and height of frame
    static final int WIDTH = 300;
    static final int HEIGHT = 100;
    Browser presentation; //area where slides are shown
    LinkPanel linksArea; //area where links are displayed
    DrawingPanel dP ;

```

```

/** File names placed into a vector for slide.action() */
Vector fileVector;      //where the links are saved
int vectorIndex = 1;    //vector slide index
    //path to website and directory
String directoryPath, webPath;
AdjustSlide adjust; //adds the applet tag to the files
String saveLinks;      //the list of html files

/** save the Live presentation */
String saveLiveFileName;
SaveLivePresentation saveLivePres;
boolean saveItLive = false;

/** the temp.html is the file the viewers watch */
    //viewers file name to go to at url
String temp = "temp.html";

/** the Line By Line presentation */
    //vector of each line in html file
Vector lines = new Vector();
    //indexes of vector for line by line viewing
int lineIndexS, lineIndexE;
    //paths for line by line
String LBLtempPath, LBLdirPath, LBLfileName;
int LBL = 0;
Checkbox radioButton;      //for line by line option

// Add a constructor for our frame.
FrameAndPanel(String title)
{
    // Set the title of the frame
    super(title);

/** window listener in case of saving Live presentation.
**/
    addWindowListener(new
        java.awt.event.WindowListener()
        {
            public void windowOpened(WindowEvent e) {};
            public void windowClosed(WindowEvent e) {};
            public void windowIconified(WindowEvent e) {};
            public void windowDeiconified(WindowEvent e) {};
            public void windowActivated(WindowEvent e) {};
        }
    );

```

```

    public void windowDeactivated(WindowEvent e) {};
    public void windowClosing(WindowEvent e)
    {
        if(saveItLive)
        {
            String f = presentation.getFileShowing();
            saveLivePres.addFile(f);
            saveLivePres.setLastLiveFile();
        }
        /** need to grab last file shown, copy and saveLive **/
        System.exit(0);
    }
});
Container c = getContentPane();
adjust = new AdjustSlide();//prep 4 adding applettag
/**** Initialization for line by line option ***/
LBLtempPath = new String();
LBLdirPath = new String();
LBLfileName = new String();
/**** AREA SETUP FOR PRESENTATION HTML WINDOW ****/
dP = new DrawingPanel();
presentation = new Browser(dP);
c.add(presentation, BorderLayout.CENTER);

/**** AREA SETUP FOR LINKS PANEL AND DRAWING COMPONENTS ****/
GridBagLayout gridbag = new GridBagLayout();
GridBagConstraints constraints = new
    GridBagConstraints();

JPanel linksPanel = new JPanel();
linksPanel.setLayout(gridbag);

/** Heading **/
JLabel Heading = new JLabel("Presentation Files");
buildConstraints(constraints, 0,0,1,1,100,10);
constraints.fill = GridBagConstraints.BOTH;
gridbag.setConstraints(Heading, constraints);
linksPanel.add(Heading);

/** Links Panel **/
JScrollPane pane = new JScrollPane(
    JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
    JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

```

```

pane.setBorder
    (BorderFactory.createLoweredBevelBorder());
linksArea = new LinkPanel(presentation);
pane.getViewport().add(linksArea);
buildConstraints(constraints, 0,1,1,1,0,45);
constraints.fill = GridBagConstraints.BOTH;
gridbag.setConstraints(pane, constraints);
linksPanel.add(pane);

/***** AREA SETUP FOR DRAWING COMPONENTS *****/
JPanel drawPanel = new JPanel();
ChoicePanel cP = new ChoicePanel(dP);
drawPanel.add(cP);

drawPanel.setBorder
    (BorderFactory.createLoweredBevelBorder());
buildConstraints(constraints, 0,2,1,1,0,45);
constraints.fill = GridBagConstraints.BOTH;
gridbag.setConstraints(drawPanel, constraints);
linksPanel.add(drawPanel);
c.add(linksPanel, BorderLayout.WEST);

/***** AREA SETUP FOR BUTTONS ON BOTTOM PANEL *****/
JPanel jp = new JPanel(new GridLayout(2,1));
jp.setBorder
    (BorderFactory.createLoweredBevelBorder());
JPanel jp1 = new JPanel();
JPanel jp2 = new JPanel();
radioButton = new Checkbox("Line by Line", false);
JButton repostButton = new JButton("Repost");
JButton lineButton = new JButton("Next Line");
JButton slideButton = new JButton("Next Slide");
JButton editButton = new JButton("Edit");
JButton drawButton = new JButton("Draw");
radioButton.addItemListener(new
    java.awt.event.ItemListener()
{
    public void itemStateChanged(ItemEvent e)
    {
        if(e.getStateChange() == ItemEvent.SELECTED)
            linksArea.setLineByLine(true);
        else
            linksArea.setLineByLine(false);
    }
}

```

```

});
repostButton.addActionListener(new
    java.awt.event.ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        /** WILL BE GETTING FROM ALYSHA HERE **/
        /** temp is for viewer through website **/
        adjust(
            savefile s = new savefile(directoryPath
                + temp, presentation.getData());
        }
    });
lineButton.addActionListener(new
    java.awt.event.ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        if(radioButton.getState())//chose line by line
        {
            ++lineIndexS;
            /** fileName has to change to be reposted for the
                presenter ***/
            getLBLfileName();
            LineByLine lbl = new LineByLine();
            lbl.adjust(lines, lineIndexS, lineIndexE,
                LBLtempPath, directoryPath+LBLfileName);
            if(lineIndexS > lineIndexE)
                JOptionPane.showMessageDialog
                    (FrameAndPanel.this, "No more lines on"+
                        "this page");
            else
                presentation.setPresentation
                    (webPath+LBLfileName);
        }
        else //user hasn't selected this option
        {
            JOptionPane.showMessageDialog
                (FrameAndPanel.this, "You have not"+
                    "selected this option" );
        }
    }
});
slideButton.addActionListener(new

```

```

        java.awt.event.ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            /** Error checking for last slide in presentation lineup
            **/
            if(fileVector.size() == vectorIndex)
            {
                JOptionPane.showMessageDialog
                    (FrameAndPanel.this, "End of"+
                     "Presentation");
            }
            /** Not last slide, continue.. **/
            else
            {
                /** get next slide **/
                String slideName = new String(
                    (String)fileVector.get(vectorIndex));
                /** add refresh tags to slide **/
                adjust.addApplet(slideName); //adjust file
            /** For later viewing, add the next slide link**/
                if(saveItLive)
                {
                    saveLivePres.addFile(slideName);
                    saveLivePres.addLink();
                }

                /** For line by line viewing **/
                if(radioButton.getState())
                    postingLineByLine(slideName, adjust);

                /** Not line by line viewing **/
                else
                    presentation.setPresentation(webPath+
                        slideName);

            /** Highlighting of used slide in link Panel **/
                fillLinks(webPath, saveLinks, vectorIndex);

                vectorIndex++; //increment. vector for next time
            }
        }
    });
    editButton.addActionListener(new

```



```

        java.awt.event.ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            if (radioButton.getState())
                JOptionPane.showMessageDialog
                    (FrameAndPanel.this, "Can NOT edit in"+
                     "line by line mode");
            else
                presentation.editPage();
        }
    });
drawButton.addActionListener(new
    java.awt.event.ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        presentation.draw();
    }
});

jpl.add(radioButton);
jpl.add(lineButton);
jpl.add(repostButton);
jpl.add(slideButton);
jpl.add(editButton);
jpl.add(drawButton);

JButton openButton = new JButton("Open Saved
    Presentation");
openButton.addActionListener(new
    java.awt.event.ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        vectorIndex = 1;
        File fl = new File("no.html");
        JFileChooser listoffiles = new JFileChooser();
        int select = listoffiles.showOpenDialog(null);
        if (select == JFileChooser.APPROVE_OPTION)
            fl = listoffiles.getSelectedFile();
        String fname = fl.getName();
        try
        {

```

```

        FileInputStream infile = new
            FileInputStream(fl);
    try
    {
        int x = infile.available(); //size of file
        byte filetext[] = new byte[x];
        infile.read(filetext); //read file
        //convert to string
        String filewords = new String(filetext);
        infile.close();
        disperseData(filewords); //fill frame & panel
    }
    catch(IOException ex)
    {
        System.out.println("failed read");
    }
}
catch(FileNotFoundException f)
{
    System.out.println(fname+" could not be"+
        " found");
}
}
});
jp2.add(openButton);
jp.add(jp1);
jp.add(jp2);
c.add(jp, BorderLayout.SOUTH);
setSize(800,600);
setVisible(true);
}

/** When program first loads this GUI,
    fill browser panel with first html file */
public void fillBrowser()
{
    String fileN = new
        String((String)fileVector.firstElement());
    directoryPath = directoryPath.trim();

    /** If user has specified they want the live
        presentation saved */
    saveLiveFileName = saveLiveFileName.trim();
    if(saveLiveFileName.length() != 0)

```

```

        saveLiveAttributes(fileN);

        /*** Add the refresh applet tag ***/
        linksArea.sendAdjustSlide(adjust);
        adjust.addApplet(fileN);

    /** If user has specified to display line by line ***/
    if(radioButton.getState())
        postingLineByLine(fileN, adjust);

    /** User did not specify line by line **/
    else
        presentation.setPresentation(webPath+fileN);
}

/*** The link panel is a JEditorPane which allows active
links. To do this it must be in html code. This sets up the
String holding the links with <a href... It also sets up a
vector with the filename in it. The vecI is used for
highlighting the links as they are used by the
slide.actionPerformed() ***/

public void fillLinks(String site, String links, int
                    vecI)
{
    saveLinks = links;
    fileVector = new Vector();
    int first = 0;
    String linkHtml = new String("<html>
        <head><body><b><br>");
    StringTokenizer getName = new
        StringTokenizer(links, "\n");
    while(getName.hasMoreTokens())
    {
        String s = getName.nextToken();
        fileVector.add(s.trim());
        linkHtml = linkHtml.concat("<a href=\"\" +
            site.trim()+\"/\");
        linkHtml = linkHtml.concat(s.trim() + "\">\" +
            s.trim() + "</a><br>");

        //for making links bold, with slide.action()
        if(first == vecI)
            linkHtml += "</b>";
    }
}

```

```

        first++;
    }
    linkHtml = linkHtml.concat("</body> </head>" +
        "</html>");
    linksArea.setLinks(linkHtml);
    linksArea.sendFrameAndPanel(this);
}

/** called by disperseData() and Setup.java */
public void setDirectory(String dirPath)
{
    /** directory path for unix adds / */
    if(dirPath.endsWith("/"))
        directoryPath = dirPath;
    else
        directoryPath = new String(dirPath + "/");
    linksArea.setDirPath(directoryPath);
    adjust.setDirectoryPath(directoryPath);
}

public void setURL(String urlPath)
{
    if(urlPath.endsWith("/"))
        webPath = urlPath;
    else
        webPath = new String(urlPath + "/");
    adjust.setURLPath(webPath);
}

private void buildConstraints(GridBagConstraints gbc,
    int gx, int gy, int gw, int gh, int wx, int wy)
{
    gbc.gridx = gx;
    gbc.gridy = gy;
    gbc.gridwidth = gw;
    gbc.gridheight = gh;
    gbc.weightx = wx;
    gbc.weighty = wy;
}

/** After reading the saved presentation file,
    this function parses the data for use */
/** Is called by open.ActionPerformed() */
private void disperseData(String presStringFile)

```

```

{
    int i = presStringFile.indexOf("***theslides***");
    int u = presStringFile.indexOf("***url***");
    int d = presStringFile.indexOf("***dir***");
    int s = presStringFile.indexOf(
        "***theSaveLiveName***");
    fillLinks(presStringFile.substring(u+7,d).trim(),
        presStringFile.substring(i+13).trim(), 0);
    setDirectory(presStringFile.substring(d+7,
        s).trim());
    setURL(presStringFile.substring(u+7,d).trim());
    setSaveLive(presStringFile.substring(s+21,
        i).trim());
    fillBrowser();
}

/** AdjustSlide places each html file as called into a
vector, which is then used for posting each line. This
function is used by the first time the file fills the
browser Panel. Used by LinkPanel, fillBrowser(), and
slide.actionPerformed() */
public void postingLineByLine
    (String fileN, AdjustSlide adj)
{
    /** UNIX */
    //get the vector file was placed into
    lines = adj.getLineVector();
    lineIndexS = adj.getFirstIndex();
    lineIndexE = adj.getLastIndex();
    LBLtempPath = adj.getTempPath();
    getLinePath();
    getLBLfileName();
    LineByLine lbl = new LineByLine();

    lbl.adjust(lines, lineIndexS, lineIndexE,
        LBLtempPath, LBLdirPath);
    presentation.setPresentation(webPath+
        LBLfileName);
}

private void getLBLfileName()
{
    LBLfileName = new
        String("LBL"+LBLfileName.valueOf(LBL)+".html");
}

```

```

        LBL++;
        //allow 4 LBL files max in users directory
        LBL = LBL % maxLBLfiles;
    }

    public void getLinePath()
    {
        LBLdirPath = directoryPath.concat(
            "LBL"+LBLdirPath.valueOf(LBL)+".html");
    }

    /** Set the save Live presentation directory name */
    /**Is called by Setup.java and from disperseData() */
    public void setSaveLive(String saveLive)
    {
        saveLiveFileName = saveLive;
    }

    /** Unix calls to make a directory and copy all jpegs
    and      gifs from this directory to the new
    directory. Used by fillBrowser() */

    private void saveLiveAttributes(String firstFile)
    {
        /*** Create a directory specified by user */
        SystemCall sc = new SystemCall("mkdir "
            +directoryPath+saveLiveFileName);

        /*** If the user included jpegs or gifs in presentation
            Need to copy to new directory */
        File f = new File(directoryPath);
        File a[] = f.listFiles();
        for(int i = 0; i < a.length; i++)
        {
            if(a[i].getName().endsWith(".jpg") ||
                a[i].getName().endsWith(".gif"))
            {
                String imagefiles = a[i].getName();
                SystemCall sc2 = new SystemCall("cp "
                    +directoryPath+imagefiles+ " "
                    +directoryPath+saveLiveFileName);
            }
        }
    }
}

```

```

    /*** Call constructor of SaveLivePresentation ***/
    saveLivePres = new
        SaveLivePresentation(saveLiveFileName);
    saveItLive = true;
    saveLivePres.setDirPath(directoryPath);
    saveLivePres.setWebPath(webPath);
    linksArea.setSaveLive();
    linksArea.sendSaveLive(saveLivePres);
    saveLivePres.addFile(firstFile);
    saveLivePres.addLink();
}
}

//Author: Sandra Hengstebeck
//Class Name: HtmlParser.java
//This file takes html codes and parses it in several
//ways. It doesn't do anything to the code before the
//body. In the body it counts lines by finding actual
//text, yet ignoring &nbsp;. This was necessary for doing
//the Line By Line option used in FrameAndPanel.

import java.util.*; //StringTokenizer
import java.text.*; //CharacterIterator

public class HtmlParser{
    private Vector vectorLines;
    private boolean flag;
    int bodyStarts, bodyEnds;
    int stringIndexBodyEnds;

    HtmlParser()
    {
        vectorLines = new Vector();
        flag = false; // find where body starts
        bodyStarts = -1;
        bodyEnds = -1;
        stringIndexBodyEnds = 0;
    }

    public void grabLines(String data)
    {
        stringIndexBodyEnds = data.indexOf("</body");
        boolean aline = false;
        boolean wholeline = true;

```

```

boolean text = true;
boolean inBody = false;
boolean tokenK = false;
boolean ampersand = false;
String savedline = new String();
int x = 0;
StringTokenizer token = new StringTokenizer
    (data, "\n\r");
while(token.hasMoreTokens())
{
    String line = token.nextToken();
    StringTokenizer tags = new
        StringTokenizer(line, "<>", true);
    while(tags.hasMoreTokens())
    {
        //get text between & including < >
        String words = tags.nextToken();

        if(words.equals("<"))          //starting a tag
        {
            text = false;          //therefore not text
            //closing bracket for whole line
            wholeline = false;
        }
        else if(words.equals(">"))      //end of tag
        {
            text = true;           //therefore could be text
            //closing bracket can be whole line
            wholeline = true;       }
            // starting body
        else if(!text && words.startsWith("body"))
        {
            inBody = true;         //now in the body
            //index of vector where body starts
            bodyStarts = x;
        }

        //in a tag ending body
        else if(!text && words.startsWith("/body"))
        {
            inBody = false;        //no longer in the body
            bodyEnds = x; //index of vec where body ends
        }

        //possible text and in the body
        else if(text && inBody)    {

```



```

words = words.trim(); //remove whitespaces
if(words.length() != 0) //if not blank line
{
    //there are &nbsp; in line
    if(words.indexOf("&nbsp;") != -1)
    {
        StringTokenizer space = new
            StringTokenizer(words, " ;&");
        //check each word

        while(space.hasMoreTokens())
        {
            // is not a &nbsp;
            if(!space.nextToken().equals("&nbsp;"))
                tokenK = true; //Then it is TEXT
        }
    }
    else
        tokenK = true; //It is TEXT
}
}

//just add lines to vector not in body yet
if(!inBody)
{
    //no other lines previously saved
    if(savedline.length() == 0)
        //place line in vector
        vectorLines.add(line+"\n");
    else //tags left when end of body was found
    {
        savedline += line; //add all together
        //add to vector
        vectorLines.add(savedline+"\n");
    }
    x++; //increment vector index
    savedline = " "; //clear the savedline string
}

//no closing tag, line not finished
else if(!wholeline && inBody)
    savedline += line; //tags are not closed
    //closing tag, but no text
else if(wholeline && !tokenK && inBody)
    savedline += line; //just add to temp string
    //closing tag, and text
else if(wholeline && tokenK && inBody)

```

```

        {
            //no tags saved in temporary string
            if(savedline.length() == 0)
                //add line to vector
                vectorLines.add(line + "\n");
            else //tags saved in temporary string
            {
                //add line to temporary string
                savedline += line;
                //add temp string to vector
                vectorLines.add(savedline + "\n");
            }
            x++; //increment vector index
            savedline = " "; //clear temporary string
            tokenK = false; //clear actual text was found
        }
    }

    public int firstLineIndex()
    {
        return bodyStarts +1;
    }

    public int lastLineIndex()
    {
        return bodyEnds ;
    }

    public Vector getLines()
    {
        return vectorLines;
    }

    public int getEndOfBodyIndex()
    {
        return stringIndexBodyEnds;
    }
}

//Author : Sandra Hengstebeck
//Class Name: Information.java
//This class program displays the information in the
//Instruction Panel.

```

```

import java.awt.*;           //Font
import javax.swing.*;

public class Information extends JPanel
{
    String heading = new String(" Instructions ");
    String explain = new String("    (1) Enter the url "+
        "where the presentation will be viewed at, " +
        "excluding the name of the file. Example: " +
        "http://web.csusb.edu/public/ \n    "+
        "(2) Enter the directory path to your web server "
        + "directory. Example: " +
        " /pool/www/public/ \n    (3) Select the html"
        + " files from your web directory to include in " +
        "the presentation. Place them in the order "+ "that
        you would like to view them. \n    (4) If " + "you
        would like to present the presentation " + "later,
        assign the presentation a name and click" + "the "Save
        Presentation button. \n    (5) If you"
        + " would like to view a file, select the file." +
        "Once it is in the Presentation "
        + "Files Selected Box, highlight it and click "
        + "the View file button. \n (6) You may either " +
        "Exit or Run the Presentation.");

    public Information()
    {
        Font h = new Font("Helvetica",Font.BOLD, 24);
        Font e = new Font("Helvetica", Font.PLAIN, 14);
        setLayout(new BorderLayout());
        JTextField head = new JTextField(heading, 50);
        head.setFont(h);
        head.setHorizontalAlignment(JTextField.CENTER);
        add(head, BorderLayout.NORTH);
        JTextArea body = new JTextArea(explain);
        body.setFont(e);
        body.setLineWrap(true);
        body.setWrapStyleWord(true);
        body.setMargin(new Insets(20,15,20,15));
        JScrollPane pane = new JScrollPane(
            JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,

```

```

JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
    pane.getViewPort().add(body);
    add(pane, BorderLayout.CENTER);
}

public JPanel getI()
{
    return this;
}
}

//Author: Sandra Hengstebeck
//Class Name: LineByLine.java
//This class program grabs the vector of lines and with
the //indexes, creates an html file for //displaying.

import java.util.*; //StringTokenizer

public class LineByLine{

    LineByLine()
    {
    }

    public void adjust(Vector lines, int lineIndexS,
        int lineIndexE, String tempPath, String dirPath)
    {
        int x = 0;
        String page = new String();
        //start of file to textlines by lineIndexS
        while(x < lineIndexS)    {
            page += (String)lines.get(x);
            page += "\r";
            x++;
        }
        x = lineIndexE;
        //line including </body> to end of file
        while(x < lines.size()-1)    {
            page += (String)lines.get(x);
            page += "\r";
            x++;
        }

        //slide used for presenter.
    }
}

```

```

        savefile RS = new savefile(dirPath, page);
        //slide used for viewer.

        savefile S = new savefile(tempPath, page);
    }
}

//Author: Sandra Hengstebeck
//Class Name: LinkPanel.java
//This class program displays the links in a JEditorPane as
//an html file, this way the links can be activated as
//normal urls. The browser panel is required for the
//presenter to view the html files.
//AdjustSlide is needed to add the applettags and places
//lines in vector in case user decides to do line by line
//presentation.

import javax.swing.*; //JEditorPane
import java.awt.*; //borderlayout
import javax.swing.event.*; //hyperlinklistener
import java.net.*; //url
import java.io.*; //IOException

public class LinkPanel extends JPanel
{
    String theData, urlPath;
    JEditorPane jt;
    String directoryPath;
    boolean LBL = false;
    boolean saveLive = false;

    /*** passed to LinkPanel for presentation options. **/
    Browser browserPanel;
    FrameAndPanel FAP;
    AdjustSlide AS;
    SaveLivePresentation saveItLive;

    LinkPanel(Browser browserP)
    {
        browserPanel = browserP;
        setLayout (new BorderLayout (5, 5));
        setSize(300,300);
        setBorder
            (BorderFactory.createLoweredBevelBorder());
    }
}

```

```

jt = new JEditorPane();
// make read-only set false
jt.setEditable(false);

// follow links
/***** MAKE HYPERLINKS IN HTML PAGE ACTIVE */
jt.addHyperlinkListener(new HyperlinkListener ()
{
    public void hyperlinkUpdate(final
        HyperlinkEvent e)
    {
        if (e.getEventType() ==
            HyperlinkEvent.EventType.ACTIVATED)
        {
            SwingUtilities.invokeLater(new Runnable()
            {
                public void run()
                {
                    URL url = e.getURL(); //get the whole url
                    //gets url
                    String u = new String(url.toString());
                    //find where filename begins
                    int slash = u.lastIndexOf("/");
                    //get the url path
                    urlPath = u.substring(0, slash + 1);
                    //get filename
                    String file = u.substring(slash + 1);
                    AS.addApplet(file); //add applet tag
                    //true if saving live presentation
                    if(saveLive)
                    {
                        saveItLive.addFile(file);
                        saveItLive.addLink();
                    }
                    if(LBL)//true if user sel. Line By Line
                        FAP.postingLineByLine(file, AS);
                    else //otherwise not Line By Line
                        browserPanel.setPresentation(urlPath +
                            file);
                }
            });
        }
    }
});

```

```

/*****

JScrollPane pane = new JScrollPane();
pane.setBorder
    (BorderFactory.createLoweredBevelBorder());
pane.getViewport().add(jt);
add(pane, BorderLayout.CENTER);
setVisible(true);
}

public void setPresentation(String webpath)
{
    try
    {
        jt.setPage(webpath);
    }
    catch(IOException i)
    {
        System.out.println("exception occurred");
    }
}

public void setLinks(String text)
{
    jt.setEditorKit(new
        javax.swing.text.html.HTMLEditorKit());
    jt.setText(text);
}

public void setDirPath(String path)
{
    directoryPath = path;
}

public void setLineByLine(boolean LBLbutton)
{
    LBL = LBLbutton;
}

public void setSaveLive()
{
    saveLive = true;
}

```

```

public void sendSaveLive(SaveLivePresentation sl)
{
    saveItLive = sl;
}

public void sendFrameAndPanel(FrameAndPanel fap)
{
    FAP = fap;
}

public void sendAdjustSlide(AdjustSlide as)
{
    AS = as;
}
}

//Author: Sandra Hengstebeck
//Class Name: openfile.java

import java.io.*; //FileInputStream
import java.util.*; //Vector

public class openfile
{
    Vector lines = new Vector();
    String fname = new String();
    File fl;
    String filewords;

public openfile(String afile)
{
    if(afile.length() != 0)//for files in control path
    {
        fname = afile; //get filename
        fl = new File(afile);
    }
    try
    {
        FileInputStream infile = new FileInputStream(fl);
        try
        {
            int x = infile.available(); //size of file
            byte filetext[] = new byte[x];

```



```

        infile.read(filetext);           //read file
        //convert to string
        filewords = new String(filetext);
        infile.close();
    }
    catch(IOException ex)
    {
        System.out.println("failed read");
    }
}
catch(FileNotFoundException f)
{
    System.out.println(fname+" could not be found");
}
}

public String getFileData()
{
    return filewords;
}
}

//Author : Sandra Hengstebeck
//Class Name: PresentationIntro.java
//Small box with two button, one to go to Setup GUI
//one to go Run Presentation GUI

import java.awt.*;    //GridLayout
import java.awt.event.*;    //actionPerformed
import javax.swing.*;    //JButton

public class PWWS implements ActionListener
{
    JButton button1;    //Set Up Presentation button
    JButton button2;    //Run Presentation button

    public PWWS()
    {
        //create the box
        JFrame frame = new JFrame("Info Box");
        frame.setSize(new Dimension(200, 145));
        frame.setResizable(false);
        frame.setLocation(220, 168);
    }
}

```

```

frame.getContentPane().setLayout(new
    GridLayout(2,1));

//create interior of the box
button1 = new JButton("Set Up Presentation");
button2 = new JButton("Run Presentation");
button1.setActionCommand("setup");
button1.addActionListener(this);
button2.setActionCommand("run");
button2.addActionListener(this);

//add label to interior box
frame.getContentPane().add(button2);
frame.getContentPane().add(button1);
frame.setVisible(true);
frame.addWindowListener(new WindowAdapter()
{
    public void windowClosing(WindowEvent e)
    {
        System.exit(0);
    }
});
}

public void actionPerformed(ActionEvent e)
{
    if(e.getActionCommand().equals("setup"))
    {
        Setup su = new Setup("Presentations World" +
            "Wide");

/***** If you want setup gui on "X" to exit whole program
add this: *****/
/*****
su.addWindowListener(new WindowAdapter()
{
    public void windowClosing(WindowEvent e)
    {
        System.exit(0);
    }
});
*****/
}
}

```

```

else if(e.getActionCommand().equals("run"))
{
    FrameAndPanel spt = new FrameAndPanel
        ("Presentations World Wide");
    spt.addWindowListener(new WindowAdapter()
    {
        public void windowClosing(WindowEvent e)
        {
            System.exit(0);
        }
    });
}

public static void main(String args[])
{
    PWWS pi = new PWWS();
}

//Author: Sandra Hengstebeck
//Class Name: savefile.java
//This file is used for saving data into a file.

import java.io.*; //IOException FileWriter

public class savefile
{
    public savefile(String path, String text)
    {
        File tempFile = new File(path);
        try //uses FileWriter
        {
            //path + filename + version
            FileWriter fw = new FileWriter(path);
            BufferedWriter out = new BufferedWriter(fw);
            out.write(text); //send data to file
            out.flush();
            out.close(); //close file
        }
        catch(IOException ex)
        {
            System.out.println("failed savefile");
        }
    }
}

```

```

    }
}
}

```

```

//Author: Sandra Hengstebeck
//Class name: SaveLivePresentation.java
//This class program will add links to the html files,
//the link will connect to the next html file that was
//shown in the actual presentation, it will add them to the
//directory that was indicated by the presenter.

```

```

import java.awt.*;
import java.io.*;

```

```

public class SaveLivePresentation {
    private String liveName, aSlide, slide, html;
    private String file, data;
    private String directoryPath, urlPath;
    private String linkString, appletTag;
    private int x = 0;

    SaveLivePresentation(String saveName)
    {
        slide = new String("slide");
        html = new String(".html");
        liveName = saveName;
        appletTag = new String("<APPLET HTTP-" +
                               "EQUIV=\"Refresh\"");
        aSlide = slide + slide.valueOf(x) + html;
    }

    /** add a file to a directory for open it */
    public void addFile(String fileName)
    {
        //original directory and filename
        file = directoryPath + fileName;
        data = removeRefresh(openIt()); //remov refresh tag
        /** file = directory path and the dir. of saved live
            presentation and slide00x.html */
        file = directoryPath+liveName+"/"+aSlide;
        x++;
    }
}

```

```

/** Add a link to the next slide */
public void addLink()
{
    aSlide = slide + slide.valueOf(x) + html;
    //end of the body of html file
    int endBody = data.indexOf("</body");
    String data2 = data.substring(0,endBody);
    linkString = new String("<br><center><a href=\"");
    linkString += urlPath + liveName + "/" + aSlide;
    linkString += "\"> Next Slide </a></center>";
    data2 += linkString;
    data2 += data.substring(endBody);
    //save file in the new directory
    savefile s = new savefile(file, data2);
}

/** Open the file to add the link for after
    presentation viewing */
private String openIt()
{
    openfile o = new openfile(file);    //open the file
    String k = o.getFileData();          //get the text
    return k;
}

/** Remove the refresh tag, so won't be confused in
    later viewing */
private String removeRefresh(String fileData)
{
    int startApplet = fileData.indexOf(appletTag);
    int endApplet = fileData.indexOf(">", startApplet);
    String noRefreshData =
        fileData.substring(0,startApplet);
    noRefreshData += fileData.substring(endApplet+1);
    return noRefreshData;
}

public void setLastLiveFile()
{
    savefile s = new savefile(file, data);
}

public void setDirPath(String dirPath)
{

```

```

        directoryPath = dirPath;
    }

    public void setWebPath(String webPath)
    {
        urlPath = webPath;
    }
}

//Author: Sandra Hengstebeck
//Class Name: SavePresentation.java
//Place presentation in a text file for later viewing

import java.io.*; //IOException
import javax.swing.*; //JTextField

public class SavePresentation
{
    public SavePresentation(String presName,
        JTextField url, JTextField dir, String saveLive,
        JTextArea files)
    {
        String savedData = new String("**url**\n"
            + url.getText() + "\n" + "**dir**\n"
            + dir.getText() +
            "\n" + "**theSaveLiveName**\n"
            + saveLive + "\n" + "**theslides**\n" +
            files.getText());

        File tempFile = new File(presName);
        try //uses FileWriter
        {
            FileWriter fw = new FileWriter(tempFile);
            BufferedWriter out = new BufferedWriter(fw);
            out.write(savedData); //send data to file
            out.flush();
            out.close(); //close file
        }
        catch(IOException ex)
        {
            System.out.println("failed savePres");
        }
    }
}

```

```

}

//Author:  Sandra Hengstebeck
//Class Name:  Setup.java
//This class program is the initial GUI window that is
//used to set up the presentation.  The user must input
//the web directory path and the url of this directory.
//The window is split into 6 areas:  (1) the title area,
(2) //the instruction panel, (3) the required textfield
area, //(4) the JFileChooser, (5) the list of files
selected with //JFileChooser, (6) buttons and option
textfields.  User //must select files for presentation.
They can be viewed, //saved, or presented at this time.
The user has to create //the html files for presentation,
in advance of using this //system.

import java.awt.*;          //GridBagLayout
import java.awt.event.*;    //actionPerformed
import javax.swing.*;       //JButton JOptionPane
import java.util.*;         //for Date

import java.awt.Graphics;

// Subclass JFrame so you can display a window
public class Setup extends JFrame {

// Set up constants for width and height of frame
static final int WIDTH = 300;
static final int HEIGHT = 100;
JTextField url, dirPath, presName, saveLiveName;
JLabel urlLabel, dirLabel;
JLabel Heading;
JTextArea files;
JPanel Info;
JTable table;
String saveLive;

// Add a constructor for our frame.
Setup(String title)
{
    // Set the title of the frame
    super(title);
    Information i = new Information();
    String str = new String("World Wide" +

```

```

        " Presentations");
Heading = new JLabel(str,JLabel.CENTER );
url = new JTextField();
dirPath = new JTextField();
urlLabel = new JLabel(" URL of presentation");
dirLabel = new JLabel(" Directory Path");
files = new JTextArea(50, 180);

/** JFile Chooser **/
    final JFileChooser listoffiles = new
        JFileChooser();//get files
listoffiles.setApproveButtonText("Select");
listoffiles.setApproveButtonToolTipText
    ("Select for presentation");
listoffiles.setBorder
    (BorderFactory.createLoweredBevelBorder());

/** Page Setup **/
    Container c = getContentPane();
    GridBagLayout gridbag = new GridBagLayout();
    GridBagConstraints constraints = new
        GridBagConstraints();
    c.setLayout(gridbag);

/** Heading **/
    buildConstraints(constraints, 0,0,2,1,0,20);
    gridbag.setConstraints(Heading, constraints);
    c.add(Heading);

/** Information **/
    JPanel Info = i.getI();
    buildConstraints(constraints, 0,1,1,1,50,30);
    constraints.fill = GridBagConstraints.BOTH;
    gridbag.setConstraints(Info, constraints);
    c.add(Info);

/** Panel, right, top **/
    JPanel textfields = new JPanel();
    textfields.setBorder
        (BorderFactory.createLoweredBevelBorder());
    GridBagLayout gbl = new GridBagLayout();
    textfields.setLayout(gbl);
    /*******
/** URL Label **/

```



```

        buildConstraints(constraints, 0,0,1,1,10,20);
        gbl.setConstraints(urlLabel, constraints);
        textfields.add(urlLabel);

/** URL Text Field */
        buildConstraints(constraints, 1,0,1,1,90,0);
        constraints.fill = GridBagConstraints.HORIZONTAL;
        gbl.setConstraints(url, constraints);
        textfields.add(url);

/** Directory Label */
        buildConstraints(constraints, 0,4,1,1,0,20);
        gbl.setConstraints(dirLabel, constraints);
        textfields.add(dirLabel);

/** Directory Text Field */
        buildConstraints(constraints, 1,4,1,1,0,0);
        gbl.setConstraints(dirPath, constraints);
        textfields.add(dirPath);
/*****
        buildConstraints(constraints, 1,1,1,1,50,0);
        constraints.fill = GridBagConstraints.BOTH;
        gridbag.setConstraints(textfields, constraints);
        c.add(textfields);

/** JFile Chooser */
        buildConstraints(constraints, 0,2,1,1,0,30);
        constraints.fill = GridBagConstraints.BOTH;
        gridbag.setConstraints(listoffiles, constraints);
        c.add(listoffiles);

/** Panel right, bottom */
        JPanel filePanel = new JPanel();
        filePanel.setBorder
            (BorderFactory.createLoweredBevelBorder());
        GridBagLayout gbl2 = new GridBagLayout();
        filePanel.setLayout(gbl2);
*****/
/** JLabel */
        JLabel Heading2 = new JLabel(
            " Presentation files selected ",
            JLabel.CENTER);
        buildConstraints(constraints, 0,0,1,1,100,20);
        gbl2.setConstraints(Heading2, constraints);

```

```

filePanel.add(Heading2);

/** Text Area for selected files */
JScrollPane pane = new JScrollPane(
    JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
    JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
pane.getViewPort().add(files);
buildConstraints(constraints, 0,1,1,1,0,80);
constraints.fill = GridBagConstraints.BOTH;
gbl2.setConstraints(pane, constraints);
files.setEditable(false);
files.addMouseListener(new
    java.awt.event.MouseListener()
{
    public void mouseMoved(MouseEvent e){}
    public void mouseEntered(MouseEvent e){}
    public void mouseExited(MouseEvent e){}
    public void mousePressed(MouseEvent e){}
    public void mouseReleased(MouseEvent e){}
    public void mouseDragged(MouseEvent e){}
    public void mouseClicked(MouseEvent e)
    {
        try
        {
            int lineNum =
                files.getLineOfOffset
                    (files.getCaretPosition());
            files.select(files.getLineStartOffset
                (lineNum),
                    files.getLineEndOffset(lineNum));
        }
        catch(javax.swing.text.
            BadLocationException ble)
        {
            System.out.println("Text Area of " +
                "Setup.java");
        }
    }
});
filePanel.add(pane);

/*****
buildConstraints(constraints, 1,2,1,1,0,0);

```

```

constraints.fill = GridBagConstraints.BOTH;
gridbag.setConstraints(filePanel, constraints);
c.add(filePanel);

/**** Buttons *****/
JPanel jp = new JPanel();
jp.setLayout(new GridLayout(3,1));
jp.setBorder
    (BorderFactory.createLoweredBevelBorder());
buildConstraints(constraints, 0,3,2,1,0,20);
gridbag.setConstraints(jp, constraints);

JPanel jp1 = new JPanel();
JPanel jp2 = new JPanel();
JPanel jp3 = new JPanel();

saveLiveName = new JTextField(30);
presName = new JTextField(30);
JLabel liveLabel = new JLabel("If you want to" +
"save the live presentation, please enter a "
+ "name for it");
JButton saveButton = new JButton("Save" +
    " Presentation");
JButton delButton = new JButton("Delete from " +
    "presentation list");
JButton viewButton = new JButton("View file");
JButton exitButton = new JButton("Exit");
JButton runButton = new JButton("Run " +
    "Presentation");
saveButton.addActionListener(new
    java.awt.event.ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        String message = new String();
        if(dirPath.getText().length() == 0)
            message += "Please enter a directory" +
                " path.\n";
        if(url.getText().length() == 0)
            message += "Please enter the url of " +
                "presentation web site.\n";
        if(files.getText().length() == 0)
            message += "Please select html files for "
                + "presentation.\n";
    }
}

```

```

String name = presName.getText();
if(name.length() == 0)
    message += "Please enter a name for the " +
        "presentation.";

if(message.length() != 0)
    JOptionPane.showMessageDialog
        (Setup.this, message);
else
{
    /*** UNIX - LAN ***/
    String s = new
        String(dirPath.getText()+"/");

    name = name.concat(".txt");
    name = s.concat(name);
    SavePresentation sp = new SavePresentation(
        name,url,dirPath,
        saveLiveName.getText(),files);
}
});
delButton.addActionListener(new
    java.awt.event.ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        files.replaceSelection("");
    }
});
viewButton.addActionListener(new
    java.awt.event.ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        String message = new String();
        if(url.getText().length() == 0)
            message += "Please enter the url of " +
                "presentation web site.\n";
/*  NEED TO CHECK FOR SELECTED TEXT  */

        if(message.length() != 0)
            JOptionPane.showMessageDialog
                (Setup.this, message);

```

```

else
{
    String viewpath = new String(url.getText()+
        "/" + files.getSelectedText());
    Viewer view = new Viewer(viewpath);
}
}
});
exitButton.addActionListener(new
    java.awt.event.ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        dispose();
    }
});
runButton.addActionListener(new
    java.awt.event.ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        String message = new String();
        if(dirPath.getText().length() == 0)
            message += "Please enter a directory " +
                "path.\n";
        if(url.getText().length() == 0)
            message += "Please enter the url of " +
                "presentation web site.\n";
        if(files.getText().length() == 0)
            message += "Please select html files for" +
                " presentation.";

        if(message.length() != 0)
            JOptionPane.showMessageDialog
                (Setup.this, message);
        else
        {
            FrameAndPanel fp = new
                FrameAndPanel("Presentations World Wide");
            //save the live pres here.
            fp.setSaveLive(saveLiveName.getText());
            fp.setURL(url.getText());    //set URL
            //0=first slide
            fp.fillLinks(url.getText(),

```

```

        files.getText(), 0);
        //set directory path
        fp.setDirectory(dirPath.getText());
        //place first slide in browser
        fp.fillBrowser();
    }
}
});

jp3.add(saveLiveName);
jp3.add(liveLabel);
jp2.add(presName);
jp2.add(saveButton);
jp1.add(delButton);
jp1.add(viewButton);
jp1.add(exitButton);
jp1.add(runButton);
jp.add(jp1);
jp.add(jp2);
jp.add(jp3);
c.add(jp, constraints);
setSize(800,600);
setVisible(true);

listoffiles.addActionListener(new
    java.awt.event.ActionListener()
{
    //Select Button
    public void actionPerformed(ActionEvent e)
    {
        if(e.getActionCommand().equals(new
            String("ApproveSelection")))
        {
            files.append(
                listoffiles.getSelectedFile().getName()+
                "\n");
            listoffiles.cancelSelection();
        }
    }
});
}

private void buildConstraints(GridBagConstraints
gbc, int gx, int gy, int gw, int gh, int wx, int wy)

```

```

    {
        gbc.gridx = gx;
        gbc.gridy = gy;
        gbc.gridwidth = gw;
        gbc.gridheight = gh;
        gbc.weightx = wx;
        gbc.weighty = wy;
    }
}

```

```

//Author: Sandra Hengstebeck
//Class Name: SystemCall.java
//This class program is for Unix System Calls. In this
//case it is used for making a directory for the saved
//Live Presentation.

```

```

import java.awt.*;
import java.lang.*;
import java.io.*;

```

```

public class SystemCall
{
    //example SystemCall("rm temp.html");
    public SystemCall(String call)
    {
        try{
            Process P = Runtime.getRuntime().exec(call);

            P.waitFor();
            P.destroy();
        }
        catch(Exception E)
        {
            System.out.println("Process error: " + E);
        }
    }
}

```

```

//Author: Sandra Hengstebeck
//Class Name: Viewer.java
//This class program is to open a html file and display it
//in a JEditorPane. The user can check their sequence of

```

the //presentation or if they can't remember which file it is, //they can view it. This is used by Setup.java, by clicking //a button.

```
import java.awt.*;
import javax.swing.*; //JScrollPane
import java.io.*; //IOException

// Subclass JFrame so you can display a window
public class Viewer extends JFrame {

    // Set up constants for width and height of frame
    static final int WIDTH = 300;
    static final int HEIGHT = 100;

    // Add a constructor for our frame.
    Viewer(String path)
    {
        try
        {
            JEditorPane jt = new JEditorPane(path);
            jt.setEditable(false);
            JScrollPane pane = new JScrollPane();
            pane.setBorder
                (BorderFactory.createLoweredBevelBorder());
            pane.getViewport().add(jt);

            Container c = getContentPane();
            c.add(pane, BorderLayout.CENTER);

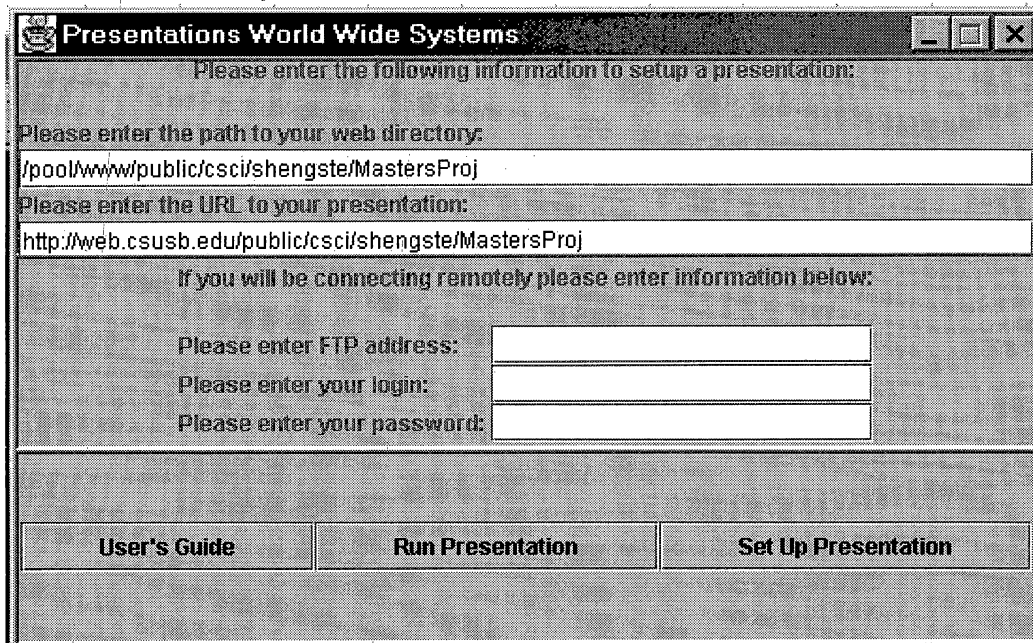
            setSize(400,300);
            setVisible(true);
        }
        catch(IOException i)
        {
            System.out.println("exception occurred in" +
                " Viewer.java");
        }
    }
}
```



APPENDIX B:

SCREEN SHOTS OF PRESENTATIONS WORLD WIDE SYSTEMS

## INTRODUCTION



**Presentations World Wide Systems**

Please enter the following information to setup a presentation:

Please enter the path to your web directory:

Please enter the URL to your presentation:

If you will be connecting remotely please enter information below:

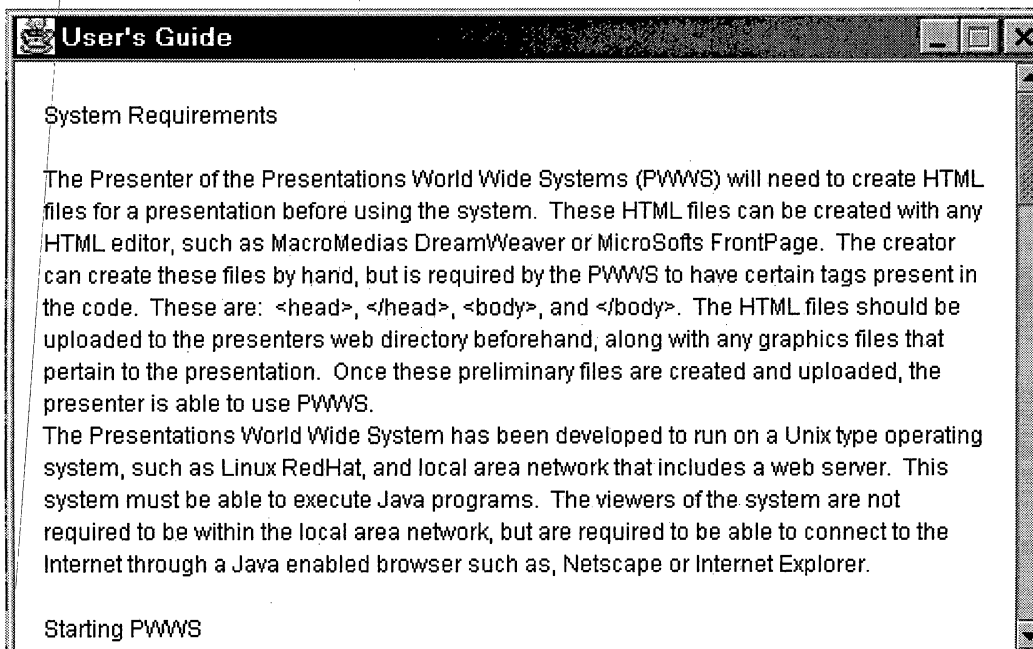
Please enter FTP address:

Please enter your login:

Please enter your password:

**User's Guide**      **Run Presentation**      **Set Up Presentation**

## USER'S GUIDE



**User's Guide**

### System Requirements

The Presenter of the Presentations World Wide Systems (PWWS) will need to create HTML files for a presentation before using the system. These HTML files can be created with any HTML editor, such as MacroMedias DreamWeaver or MicroSofts FrontPage. The creator can create these files by hand, but is required by the PWWS to have certain tags present in the code. These are: <head>, </head>, <body>, and </body>. The HTML files should be uploaded to the presenters web directory beforehand, along with any graphics files that pertain to the presentation. Once these preliminary files are created and uploaded, the presenter is able to use PWWS.

The Presentations World Wide System has been developed to run on a Unix type operating system, such as Linux RedHat, and local area network that includes a web server. This system must be able to execute Java programs. The viewers of the system are not required to be within the local area network, but are required to be able to connect to the Internet through a Java enabled browser such as, Netscape or Internet Explorer.

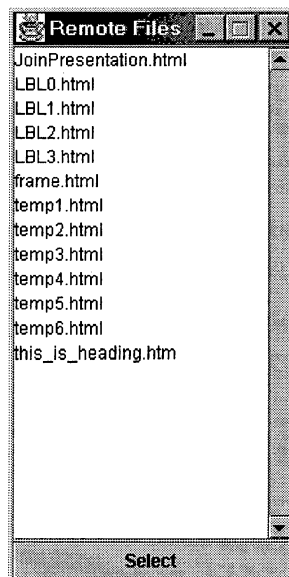
### Starting PWWS

## REMOTE CONNECTION SETUP WINDOW

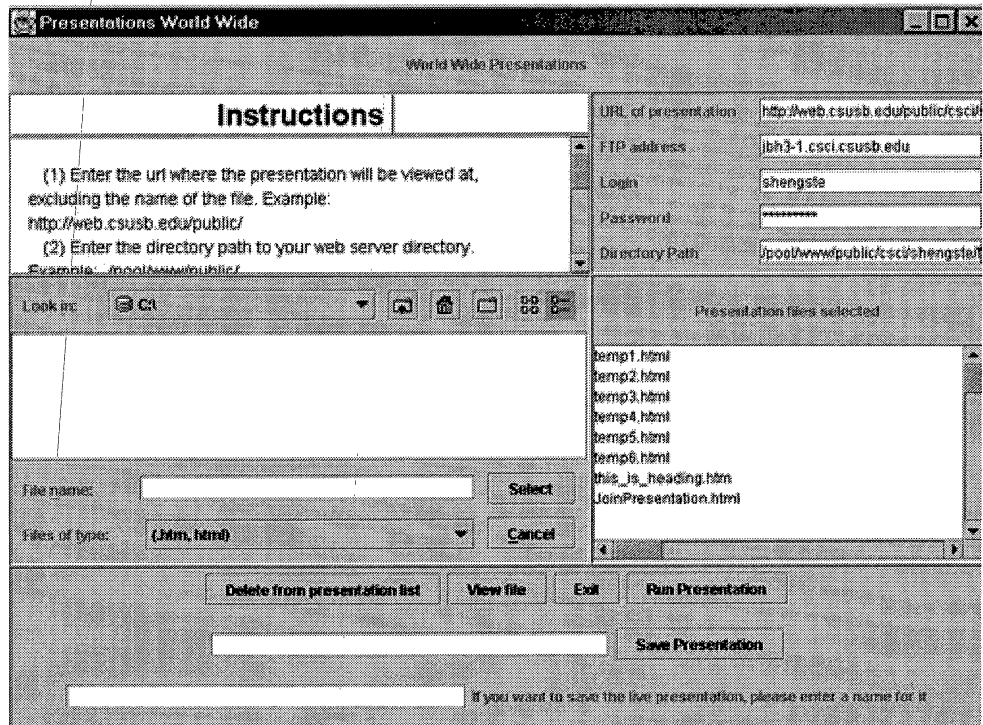
The window is titled "Presentations World Wide" and contains the following sections:

- Instructions:**
  - (1) Enter the uri where the presentation will be viewed at, excluding the name of the file. Example: `http://web.csusb.edu/public/`
  - (2) Enter the directory path to your web server directory. Example: `/pool/www/public/csci/shengste/`
- Form Fields:**
  - URL of presentation: `http://web.csusb.edu/public/csci/`
  - FTP address: `lth3-1.csci.csusb.edu`
  - Login: `shengste`
  - Password: `*****`
  - Directory Path: `/pool/www/public/csci/shengste/`
- File Selection:**
  - Look in: `C:\`
  - File name:
  - Files of type: `(.htm, .html)`
  - Buttons: **Select**, **Cancel**
- Actions:**
  - Buttons: **Delete from presentation list**, **View file**, **Exit**, **Run Presentation**
  - Save Presentation** button with a text input field below it.
  - Text: "If you want to save the live presentation, please enter a name for it"

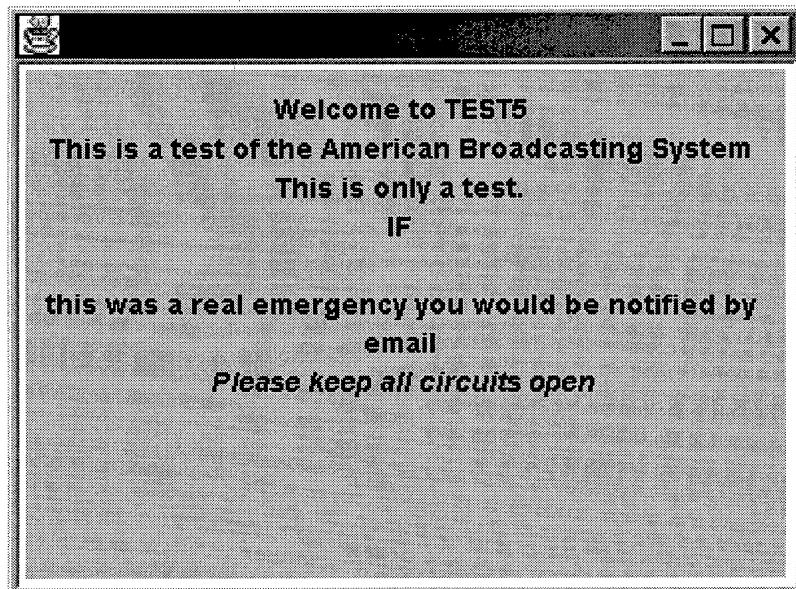
## REMOTE FILE CHOOSER



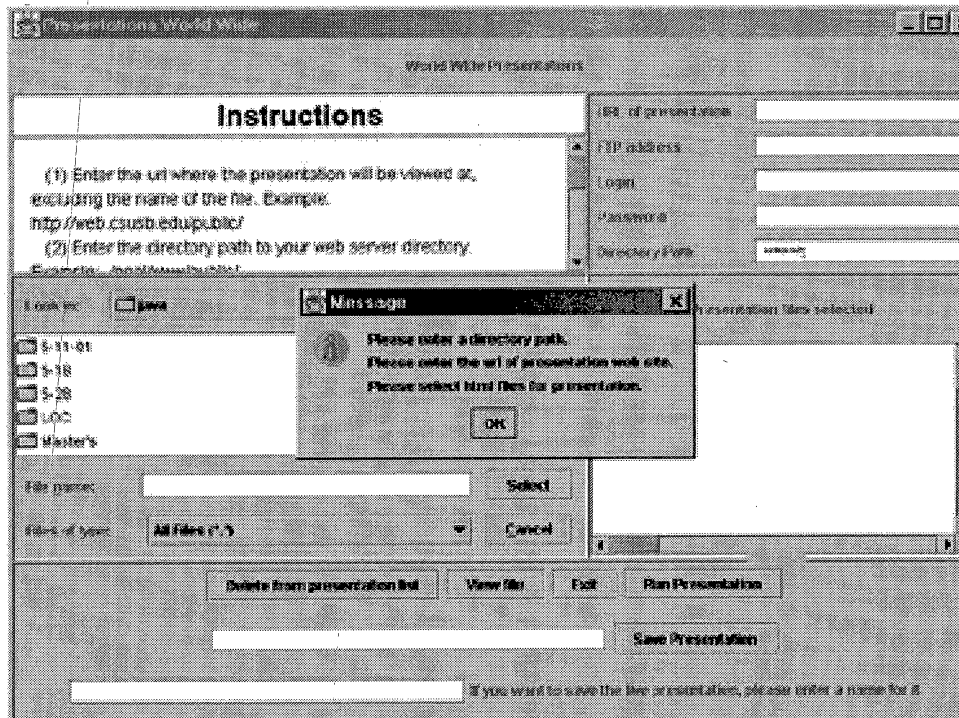
## REMOTE SETUP WINDOW WITH FILES SELECTED



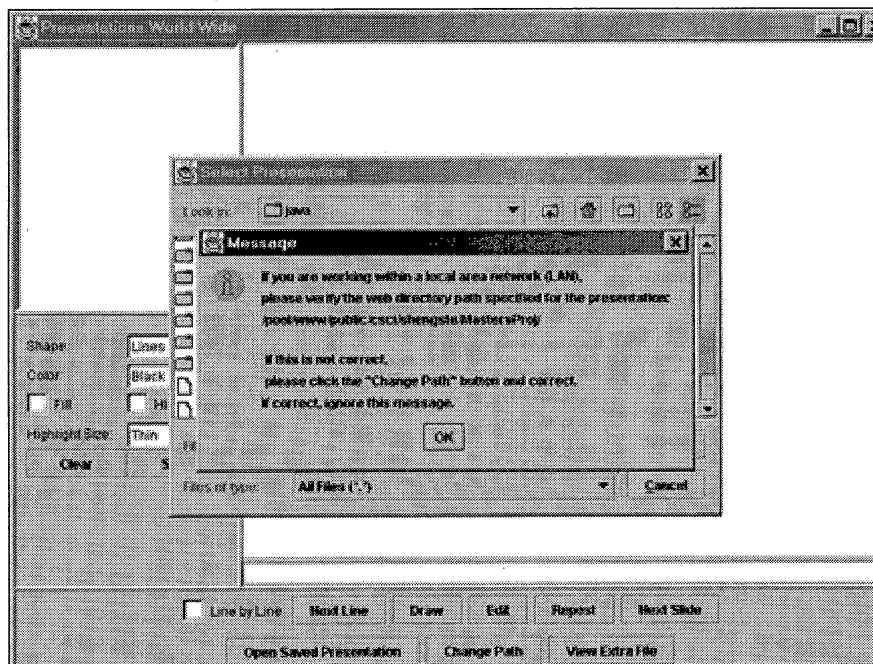
## VIEWER WINDOW



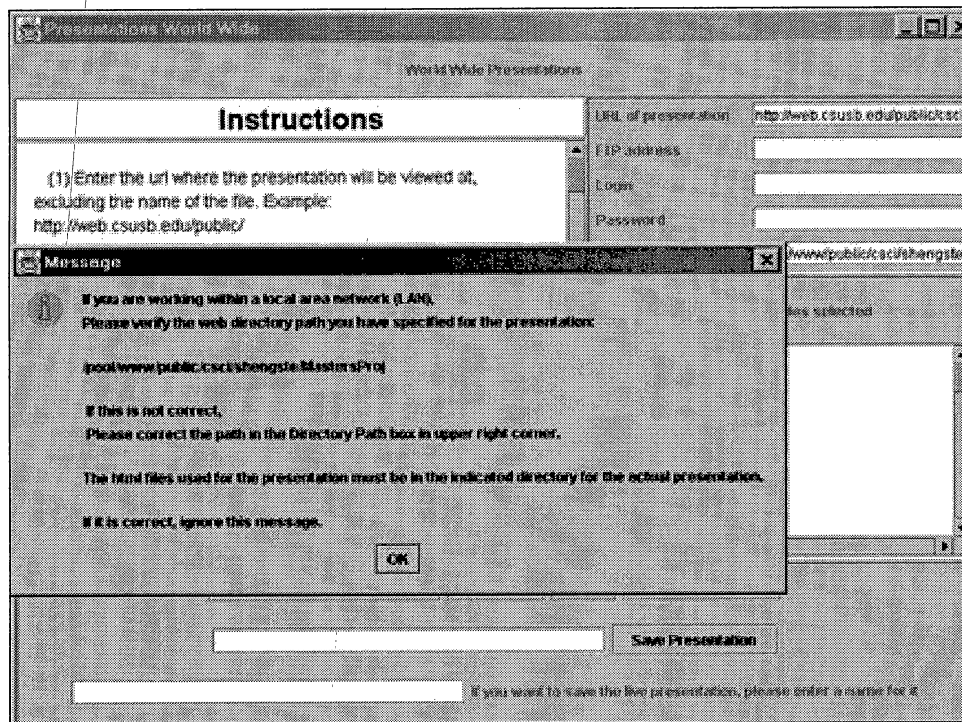
## ERROR-Setup Page



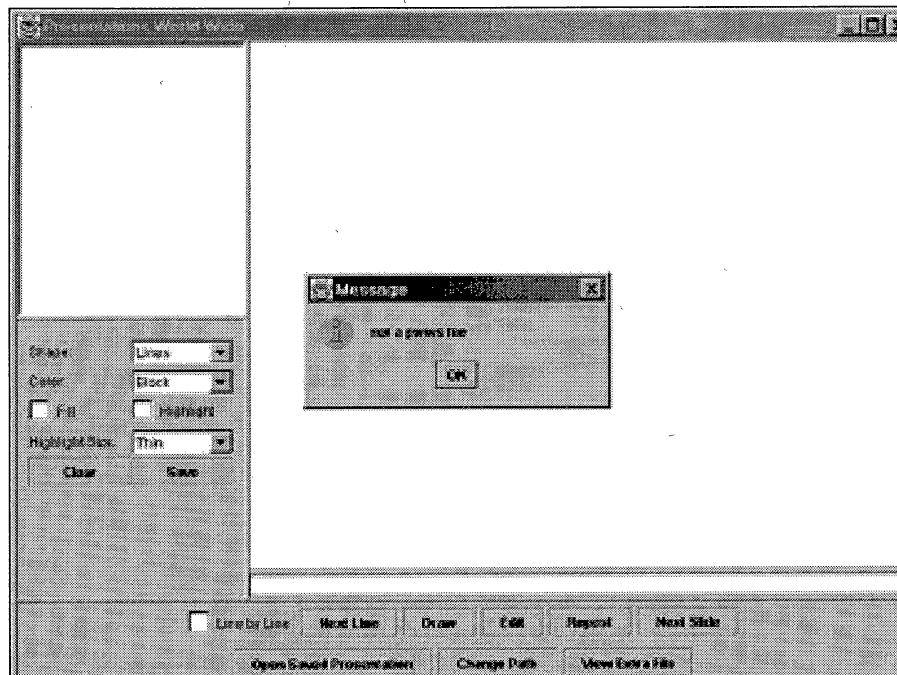
## ERROR-Directory Change



## ERROR-Directory Change Setup Page

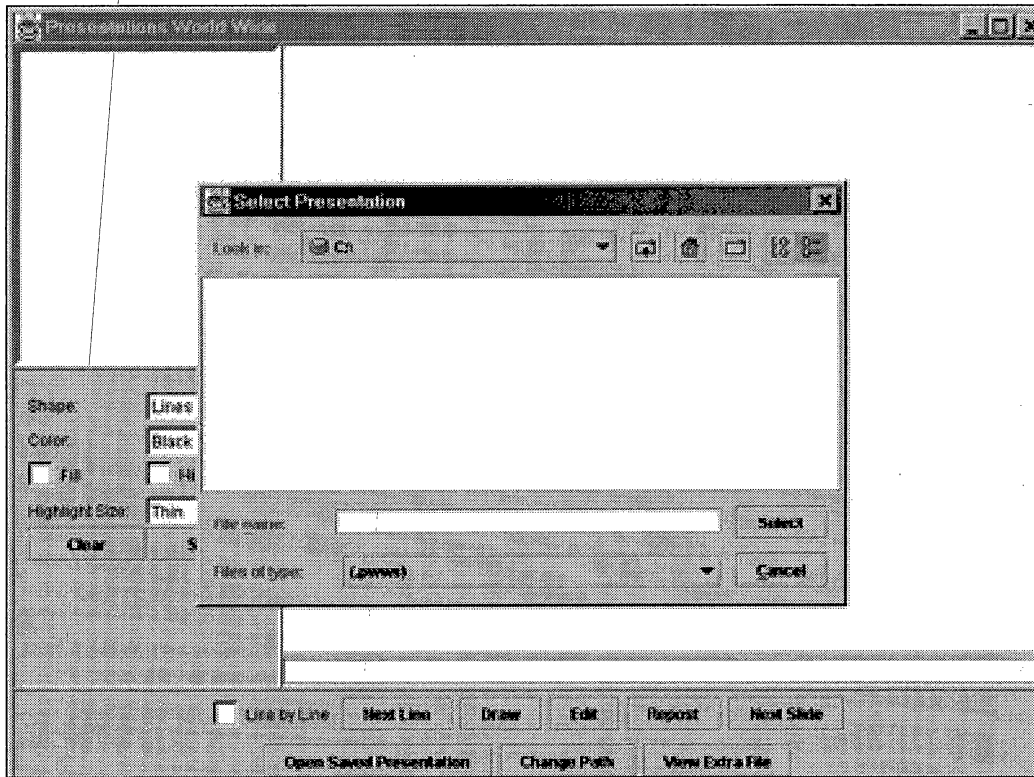


## ERROR Non-PWWS file





## FILE CHOOSER



## REFERENCES

- Algorithma 2001. <http://139.182.133.122/2001/>
- Astound Conference Center.  
[http://astound.com/wc/prod/prod\\_002.html](http://astound.com/wc/prod/prod_002.html)
- Downing, Douglas A., Covington, Michael A., Covington, Melody Mauldin, Dictionary of Computer and Internet Terms, 6th Edition. Barron's Educational Series, Inc. Hauppauge, New York. 1998.
- Humphrey, Watts S. A Discipline for Software Engineering. Addison-Wesley. May 1995.
- IEEE STD 830-1993. IEEE Recommended Practice of Software Requirements Specifications
- IEEE STD 830-1998 IEEE Recommended Practice of Software Requirements
- Kim, Dohyon Donte. The Internet Instructional Aid. California State University, San Bernardino. San Bernardino, California. March 1999.
- Phillip, Lee Anne. Practical HTML 4. Que/MacMillan Computer Publishing. 1998.
- Schach, Stephen R. Classical and Object-Oriented Software Engineering with UML and JAVA. Fourth edition. WCB/McGraw-Hill. 1999.
- Simone, Luisa. "Editor's Choice". PC Magazine. December 17, 1999.  
<http://www.zdnet.com/pcmag/stories/reviews/0,6755,2408782,00.html/>
- Simone, Luisa. "MyPlaceWare/PlaceWare 3.5 Conference Center. December 17, 1999.  
<http://www.zdnet.com/pcmag/stories/reviews/0,6755,2408779,00.html/>



Simone, Luisa. "WebEx.com/WebEx Meeting Center".

December 17, 1999.

<http://www.zdnet.com/pcmag/stories/reviews/0,6755,2408781,00.html/>

WebCT Helping Educators Transform Education.

<http://www.webct.com/Specifications> - Annex A