

2009

## Active Lecture: An Interactive Lecturing System

Himavanthara Sajja  
*San Jose State University*

Follow this and additional works at: [https://scholarworks.sjsu.edu/etd\\_projects](https://scholarworks.sjsu.edu/etd_projects)



Part of the [Computer Sciences Commons](#)

---

### Recommended Citation

Sajja, Himavanthara, "Active Lecture: An Interactive Lecturing System" (2009). *Master's Projects*. 109.  
DOI: <https://doi.org/10.31979/etd.hzg8-2k7b>  
[https://scholarworks.sjsu.edu/etd\\_projects/109](https://scholarworks.sjsu.edu/etd_projects/109)

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact [scholarworks@sjsu.edu](mailto:scholarworks@sjsu.edu).

CS298 Spring 2009  
Writing Project

Active Lecture: An Interactive Lecturing System

Advisor: Dr. Cay Horstmann  
Student: Himavanthara Sajja

A Project Report  
Presented to

The Faculty of the Department of Computer Science  
San José State University

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

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

---

Dr. Cay Horstmann

---

Dr. Chris Pollett

---

Dr. David Taylor

APPROVED FOR THE UNIVERSITY

---

## ACKNOWLEDGEMENTS

I express my deep sense of gratitude to my advisor Dr. Cay Horstmann for giving me the opportunity to work with him and suggesting the excellent project topic the Active Lecture system. I thank the committee members, Drs. Chris Pollet and David Taylor. I appreciate their critical review of the report and excellent suggestions for improving the report. In addition, I thank my wife Malleeswari Boddu and my son Anish for their patience and support through my Masters program. Finally, I thank my parents, Sri Rudra Murthy Sajja and Smt. Nagaveni Sajja for encouraging me to pursue higher education.

## ABSTRACT

Classroom lecturing has been improving with the advances in information technology. The Active Lecture, an advanced lecturing system, is developed as part of this project. The Active Lecture system allows the students to be more interactive during a lecture process. The Active Lecture system has features that are helpful to both the instructors and the students before, during, and after the lectures. The Active Lecture system is deployed and is made available for use in a classroom. The usage of the Active Lecture system is compared with the usage of Ubiquitous Presenter in a classroom setting.

## Table of Contents

1. Introduction.....	1
2. Existing Work.....	2
2.1 Classroom Technologies .....	2
2.1.1 Smart Boards.....	2
2.1.2 Clickers .....	2
2.1.3 Tablet PCs.....	2
2.2 Related Classroom Systems .....	2
2.2.1 Classroom Presenter.....	3
2.2.2 Ubiquitous Presenter .....	4
2.2.3 Elluminate .....	6
3. Features of the Active Lecture System .....	8
3.1 Instructor Interface Features.....	8
3.1.1 Support Unregistered and Registered Instructors .....	8
3.1.2 Start a Lecture .....	10
3.1.3 Add New Topic.....	10
3.1.4 Launch Capture Tool .....	11
3.1.5 List of Topics .....	13
3.1.6 Discuss Current Topic.....	14
3.1.7 Download as PDF .....	15
3.1.8 Student view from Instructor Interface .....	15
3.1.9 Configurable Help.....	16
3.1.10 Manage Lectures .....	18
3.2 Student Interface Features .....	19
3.2.1 Highlight the Discussion Topic .....	19
3.2.2 Multi-part Student Response.....	19
3.2.3 Allow Multiple Student Responses.....	20
3.2.4 Anonymous Student Response.....	21
3.2.5 Automatic Refresh of Topics .....	21
3.2.6 Downloading Lecture Notes .....	21
4. Implementation of the Active Lecture System.....	22
4.1 Active Lecture Server.....	22
4.1.1 Database Objects.....	22
4.1.2 Enterprise Java Beans .....	22
4.1.3 Email Lecture Details.....	23
4.1.4 Automatic Refresh of List of Topics.....	25
4.1.5 Discuss Current Topic.....	27
4.1.6 Launching of the Active Lecture Client.....	28
4.1.7 File Upload Servlet .....	30
4.1.8 Generating Lecture Notes in PDF .....	31
4.1.9 Configurable Help.....	33
4.2 Active Lecture Client .....	34
4.2.1 Capture Screen Shot.....	34
4.2.2 Save Image.....	35

4.2.3 Packaging and Signing the Client .....	35
5. Implementation Challenges.....	37
5.1 Development using Java, JavaEE, and AJAX.....	38
5.2 Providing Complete Solution using Different Technologies .....	38
6. Analysis and Results .....	39
6.1 Comparison with Ubiquitous Presenter.....	39
6.1.1 Screen Captures versus Marked Slides .....	39
6.1.2 Student Responses .....	40
6.2 Active Lecture System Student Response Analysis.....	40
6.2.1 Image Response Analysis .....	40
6.2.2 Text and Choice Responses Analysis .....	41
7. Conclusion .....	42
7.1 Key Achievements .....	42
7.2 Future Enhancements .....	42
7.3 Further Analysis .....	43
References.....	44

## List of Tables

<i>Table 1</i> Feature Comparison Table.....	3
--	---

## List of Figures

<i>Figure 1</i> Classroom Presenter.....	4
<i>Figure 2</i> Student Interface of Ubiquitous Presenter .....	5
<i>Figure 3</i> Student Submission Window in Ubiquitous Presenter .....	6
<i>Figure 4</i> Participant Interface in Elluminate.....	7
<i>Figure 5</i> Unregistered Instructors .....	9
<i>Figure 6</i> Registered Instructors Sign in Section .....	9
<i>Figure 7</i> Instructor Lecture Page .....	10
<i>Figure 8</i> Add New Topic .....	11
<i>Figure 9</i> Enter New Topic .....	11
<i>Figure 10</i> Launch Capture Tool.....	12
<i>Figure 11</i> Save Captured Image .....	13
<i>Figure 12</i> List of Topics .....	13
<i>Figure 13</i> Automatic Addition of Image Topic .....	14
<i>Figure 14</i> Discuss Current Topic.....	15
<i>Figure 15</i> Download as PDF .....	15
<i>Figure 16</i> Student View Link in Instructor Interface.....	16
<i>Figure 17</i> Student Interface Window.....	16
<i>Figure 18</i> Configurable Help.....	17
<i>Figure 19</i> Show Help.....	17
<i>Figure 20</i> Help Details.....	18
<i>Figure 21</i> Manage Lectures for Registered Instructors.....	18
<i>Figure 22</i> Student Response Page .....	19
<i>Figure 23</i> Table Diagram.....	22
<i>Figure 24</i> Object Diagram .....	23
<i>Figure 25</i> Properties for JavaMail Setup.....	24
<i>Figure 26</i> Additional Properties for JavaMail Setup.....	24
<i>Figure 27</i> Security Warning before Downloading Active Lecture Client .....	36
<i>Figure 28</i> Student Response Choice List.....	41



## 1. Introduction

Improving the delivery of classroom lectures is an ongoing process. Several educational institutes have taken the initiative to develop improved lecture delivery systems by imbedding the advances in information technology and the concepts of interactive learning. The purpose of this report is to introduce and provide details about a new lecturing system called the Active Lecture system. The scope of this report is to summarize the current state of classroom technologies, features, and the extent of implementation. This report also provides a brief analysis of the new system.

A traditional classroom uses blackboards and chalk. Whiteboards and color pens followed blackboards and chalk. Transparency projection systems were later introduced to facilitate communicating large amounts of information in short durations. Slides were introduced later to help the audience during lectures. With the advent of projection systems and an increase in the availability of computers, lectures were delivered as presentations. Smart input devices followed these advances in computers. Today, we are in the middle of another change as these new technologies enable us to make classroom lectures more interactive.

## 2. Existing Work

Touch screens and smart board technologies make static presentations more lively. Instructors can highlight and write on the slides while delivering lectures to make the sessions more interactive and interesting.

### *2.1 Classroom Technologies*

Several kinds of new computer hardware technologies were developed in the last decade. Some of these hardware devices are designed mainly for instructors, some are designed for students, and some are designed for both. For example, smart boards are designed for instructors; clickers are designed for students; and tablet PCs are designed to be used by both instructors and students.

#### *2.1.1 Smart Boards*

There are several different smart board technologies. The basic idea is to use a pen-like input device on a projection screen. This works by using a smart board that passes the input to the computer with a pen-like device. This device is specifically designed for the instructor. In 2004, Friedland worked on a teaching mechanism using electronic smart boards [1].

#### *2.1.2 Clickers*

Clickers are hardware devices similar to TV remote controls that allow users to provide anonymous responses to questions that are asked during a lecture. They are also called audience response systems. The instructor's presentation is specifically designed for the clickers and asks questions. Then, the students can submit the responses using the clickers. The summary of the responses from the students is shown as a graph. Many studies have confirmed that learning improves when lectures use these clicker systems [2]. However, the clicker systems are not adopted widely because students are required to buy these hardware devices in order to use these systems [3].

#### *2.1.3 Tablet PCs*

The advent of touch screens has enhanced the creation of many applications. The support for touch screens in commonly used operating systems like Windows has opened up ways to use these features in lecture rooms. The tablet PC allowed users to highlight easily without the use of a mouse. This is also useful for instructors to highlight the contents that they present. During 2004, Simon [4] started experimenting with tablet PCs for use with lecturing systems. French also proposed using tablet PCs in a collaborative learning environment in 2007 [5]. Microsoft encouraged the use of tablet PCs in a collaborative environment with ConferenceXP [6].

### *2.2 Related Classroom Systems*

There are many other systems with similar concepts, where a lecturer presents the slides and receives responses from the students. One early system that experimented with slides and ink is

Classroom 2000 [7]. Another system that uses the same concept is Classroom Presenter [8]. It was mainly designed to integrate power point slides with pen-based devices. A number of extensions were developed to Classroom Presenter, notably Ubiquitous Presenter, Tablet PC System, and Classroom Learning Partner [9]. Ubiquitous Presenter provides a web interface to the Classroom Presenter [10]. ActiveClass [11] is another lecturing system that uses wireless devices like PDAs. Elluminate is an application whose primary purpose is to present web-based seminars that have features similar to that of the Active Lecture system. The features of these systems are compared in Table 1.

*Table 1* Feature Comparison Table

Feature/Application	Classroom Presenter	Ubiquitous Presenter	Elluminate	Active Lecture
Ink slides using pen based systems	Y	Y	Y	Y
Ink slides using non-pen based systems	N	Y	Y	Y
Support power point slides	Y	Y	Y	N
Support whiteboard slide	Y	Y	Y	Y
Support screen capture	N	N	Y	Y
Text, choice as student response	N	Y	Y	Y
Image as student response	Y	Y	Y	Y
Named student response	Y	Y	Y	N
Anonymous student response	N	Y	N	Y
Selective display of student responses	Y	Y	Y	N
Voice message as response	N	N	Y	N
Refresh list of topics/slides	Y	Y	N	Y
Support registered instructors	Y	Y	Y	Y
Support unregistered instructors	N	N	N	Y
Download as PDF	N	N	N	Y

In the sections below, Classroom Presenter, Ubiquitous Presenter, and Elluminate are explained in more detail, as they are similar to the Active Lecture system.

### *2.2.1 Classroom Presenter*

Classroom Presenter [8, 12] was developed by the University of Washington, Seattle, Washington. Classroom Presenter allows instructors to create a prepared presentation and use a tablet PC to highlight the content of the slides. Classroom Presenter uses a multicast networking broadcast model that needs installation of software in workstation and student systems. In this broadcast model, the instructor tablet PC controls the presentation by displaying projector mode slides on the projector that is attached to a separate workstation. The students view the student

mode of the same slides. Students can draw on the slides and submit to the presenter. Classroom Presenter assumes that the student is also using a tablet device. The drawing is stored separately from the actual slides. This allows the instructor to show specific drawings on the projector to the class.

A screen shot of Classroom Presenter is shown in Figure 1 [13]. The main features of Classroom Presenter [23] that are available in Ubiquitous Presenter are:

1. Ink power point slides using a pen-based system
2. Instructor, projector, and student view of slides [14, 15]
3. Film strip for slides preview
4. Whiteboard slide support
5. Toolbar with multicolor ink, text, erase, undo, redo, resize slide support

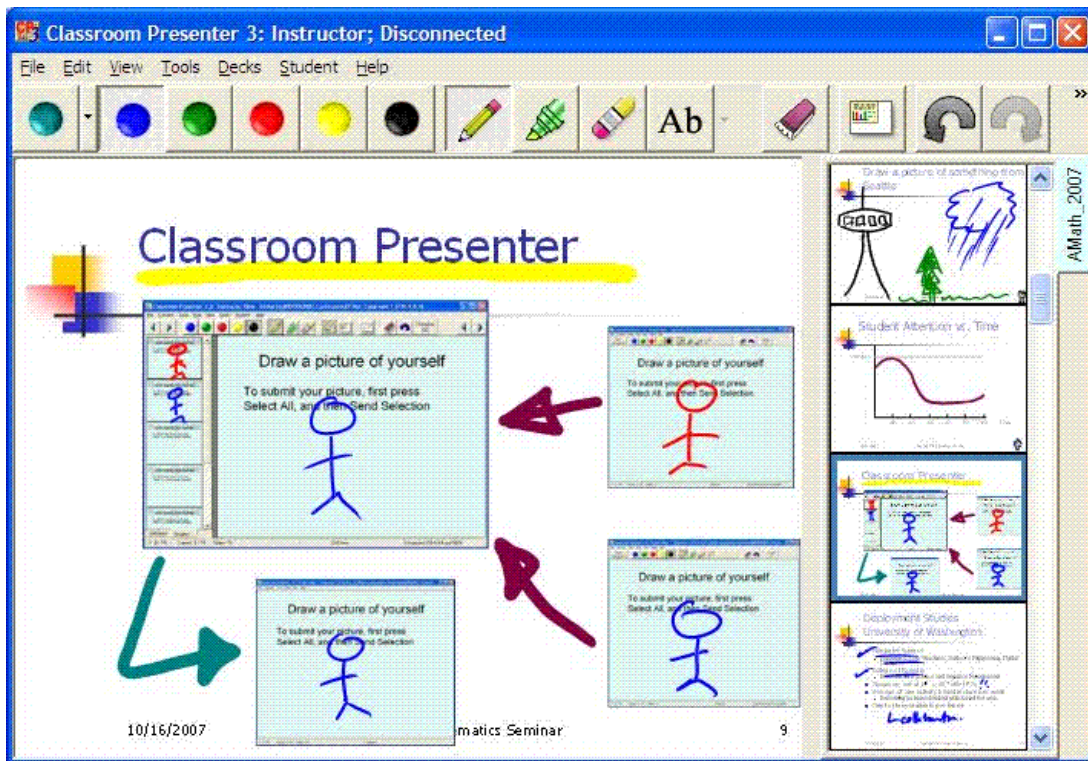


Figure 1 Classroom Presenter

The student submission feature was introduced first as an extension to the Classroom Presenter in “Tablet PC System” [4], and later made part of the Classroom Presenter.

### 2.2.2 Ubiquitous Presenter

Ubiquitous Presenter [10, 16] was developed as an extension to the Classroom Presenter. Ubiquitous Presenter is designed as a web interface for Classroom Presenter. This allows the students to participate in the class using normal PCs and a web browser rather than tablet devices

and special client software. Ubiquitous Presenter also allows students to view and work on the slides in both synchronous and asynchronous modes as shown in Figure 2 [17]. In synchronous mode, students will be viewing the same slides as that of instructors. In asynchronous mode, students can view and work on any slide in the presentation. The web interface of Ubiquitous Presenter allows a wide audience to use the system. Ubiquitous Presenter is developed using PHP and Java.

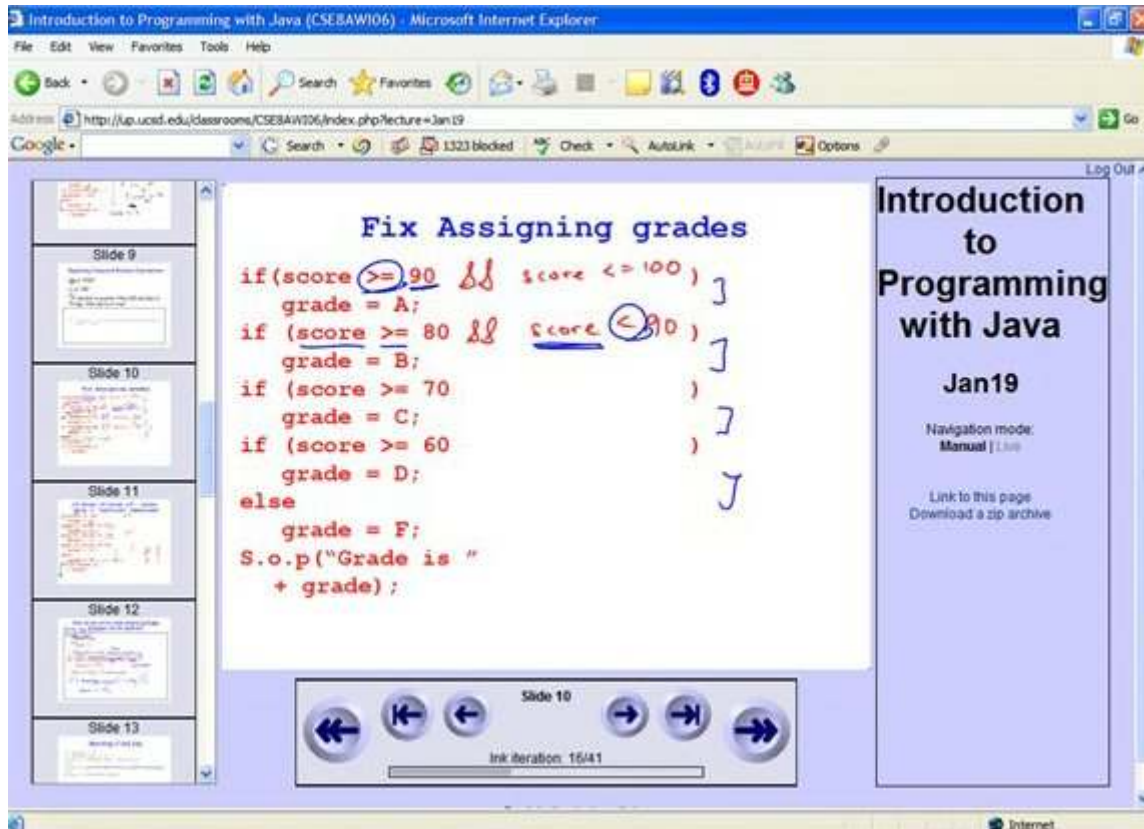


Figure 2 Student Interface of Ubiquitous Presenter

A sample student submission window of Ubiquitous Presenter is shown in Figure 3 [10]. Student submissions supported by Ubiquitous Presenter contain the following features:

1. Allow text, choice list, or inked student responses
2. Tag student names to responses, viewable only by instructor
3. Allow anonymous student responses [18]
4. Allow selective display of student responses without student names
5. Addition of new whiteboard slides during lecture

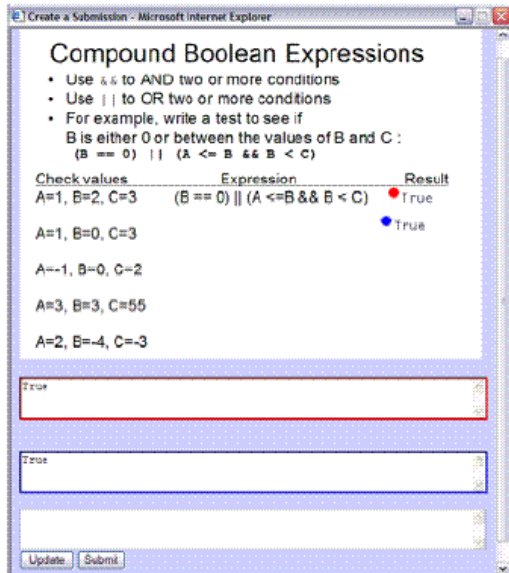


Figure 3 Student Submission Window in Ubiquitous Presenter

In 2007, the Ubiquitous Presenter was further enhanced to allow student input from mobile phones. This extension allows the students to submit “text or photo messages” [19].

### 2.2.3 *Illuminate*

Illuminate is a collaboration tool that can be used to conduct seminars over the web. Illuminate is mainly designed for web conferencing [20]. The moderator facilitates and the active participants who drive the conference. The moderator can create questions and identify the answer types that are displayed to users.

Illuminate has the following features as shown in Figure 4 [20]:

1. Participant list
2. Polling
3. Raise hand
4. Text message
5. Voice message
6. Whiteboard tool

The whiteboard tool allows users to take screen shots of selected regions and perform draw and write operations on the screen shots.

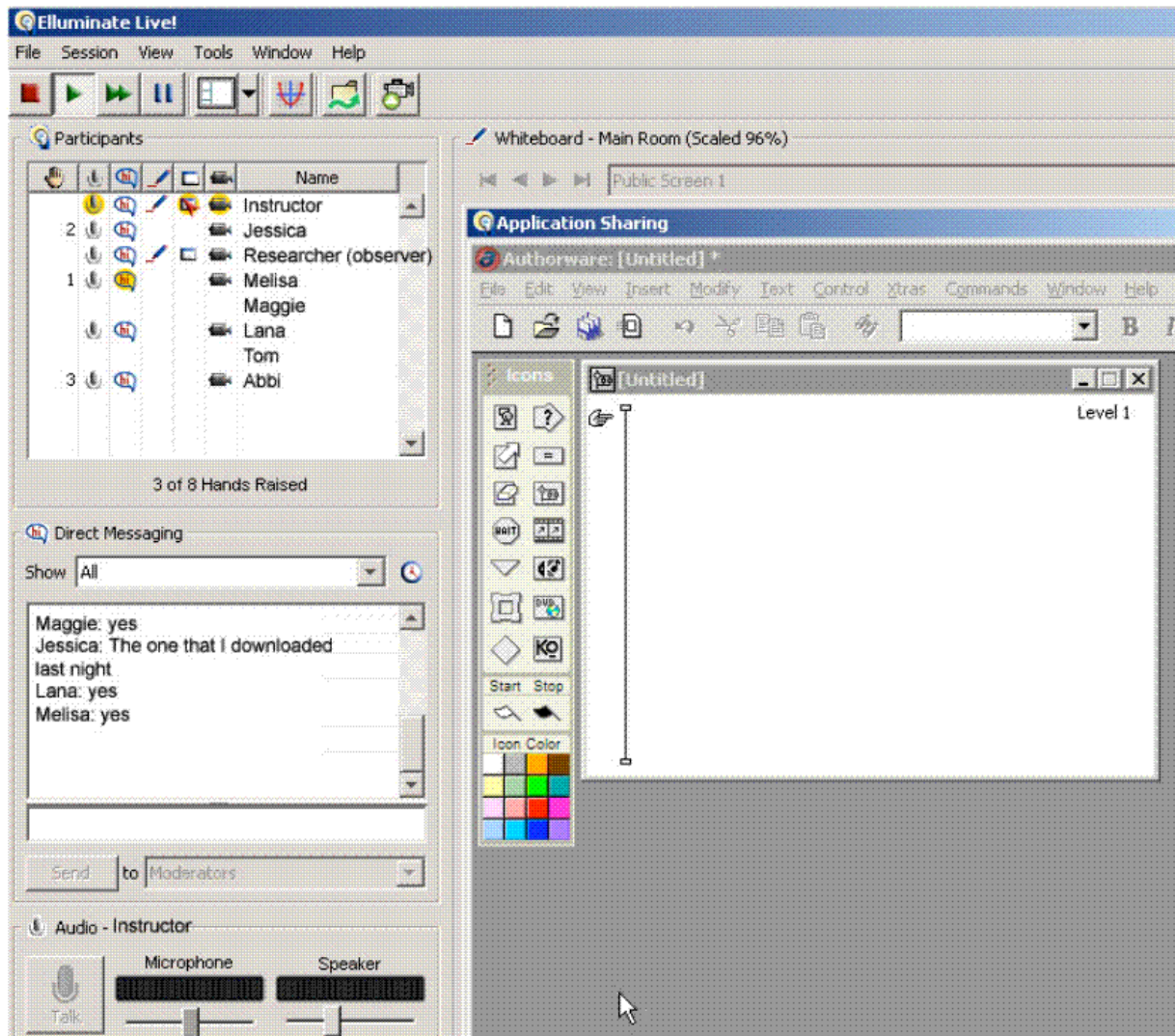


Figure 4 Participant Interface in Elluminate

As Elluminate is a collaboration tool for remote participants that allows them to share their thoughts to other remote participants, Elluminate is less often used in a classroom setting.

### 3. Features of the Active Lecture System

The Active Lecture system is designed for the classroom setting and not for internet delivered “long distance” courses. The tools that are provided in this system enable the instructors to create a dynamic environment where the instructors and the students participate. Lecture notes can be saved during the process. The instructors can create topics for a lecture and allow the students to interact and provide responses to the topics that are being discussed. As the requirements for the instructor interface differ from the student interface, the system provides different interfaces for both kinds of end-users. The innovative features of the proposed Active Lecture system are:

1. Screen capture as topic
2. Text as topic
3. Screen capture as student response
4. Show all student responses
5. Lecture notes in PDF format
6. All responses are anonymous
7. Support both unregistered and registered instructors
8. Allow multiple student responses
9. Automatic refresh of topics

The important features that are supported in the instructor and student interfaces are explained in this section.

#### *3.1 Instructor Interface Features*

Instructors use the instructor interface of the system. The features of the instructor interface are as follows:

1. Support unregistered and registered instructors
2. Start a lecture
3. Add new topic
4. Launch capture tool
5. List of topics
6. Discuss current topic
7. Download as PDF
8. Student view from instructor interface
9. Configurable help
10. Manage lectures

##### *3.1.1 Support Unregistered and Registered Instructors*

The Active Lecture system allows instructors to use the system either anonymously or as registered users. In order to make it as easy as possible for new users to try this system, unregistered instructors are allowed to create lectures and experiment with the system.



Unregistered instructor features can also be used by registered instructors to create sample lectures that they do not want to track later.

Unregistered instructors can start lectures and use the system similar to that of registered instructors. However, they will have to remember their lecture identifiers (IDs) to retrieve their lectures. As shown in Figure 5, unregistered instructors also have an option to provide email addresses in which case the lecture IDs are sent to their emails. Unregistered users can retrieve the lectures that they created if they have their lecture IDs. Unregistered users have an option to secure a lecture by entering a password for the lecture while creating it.

### Unregistered Instructors

You can use this system without registering.

<input type="button" value="New Lecture"/>	email (optional)	<input type="text"/>	If you supply your email address, you will get a URL + ID reminder
	password (optional)	<input type="text"/>	If you supply a password, you can resume the lecture later
<input type="button" value="Retrieve Lecture"/>	ID (required)	<input type="text" value="0"/>	
	password	<input type="text"/>	Supply to resume, leave blank to view

Figure 5 Unregistered Instructors

Registered instructors have extra features that are not available for unregistered instructors. Registered instructors can maintain the lectures that they created more easily with the features to view the “list of lectures” they created. Registered instructors can retrieve those previous lectures, or download notes with one click. Registered instructors can sign in as shown in Figure 6.

### Registered Instructors

Not Registered? [Sign Up](#)

#### Instructor Sign In

User Name (required)	<input type="text"/>
password	<input type="text"/>
<input type="button" value="Instructor's New Lecture"/>	<input type="button" value="List Instructor's Lectures"/>

Figure 6 Registered Instructors Sign in Section

Any instructor can sign up to become a registered user by clicking on the “Sign Up” link and entering the registration details. A registered instructor can log in and use all the features that are available for registered instructors.

### 3.1.2 Start a Lecture

Instructors can start a lecture from different places.

1. Unregistered instructors can start a lecture by selecting
  - (a) “New Lecture” in the “Unregistered Instructors” section.
2. Registered Users can start a lecture by selecting
  - (a) “Instructor’s New Lecture” in the “Registered Instructors” section
  - (b) “New Lecture” from “List of Lectures”
  - (c) “New Lecture” from “Confirm Registration” pages.

When an instructor starts a lecture, the instructor is taken to the main page for the lecture that is shown in Figure 7. This is the main page for the instructor during the lecture. The instructor can perform various operations on this page: create new text topic by selecting “New Topic” button; create new image topic by selecting “Launch Capture Tool”; or start discussion on the current topic by selecting “Discuss Current Topic” button.

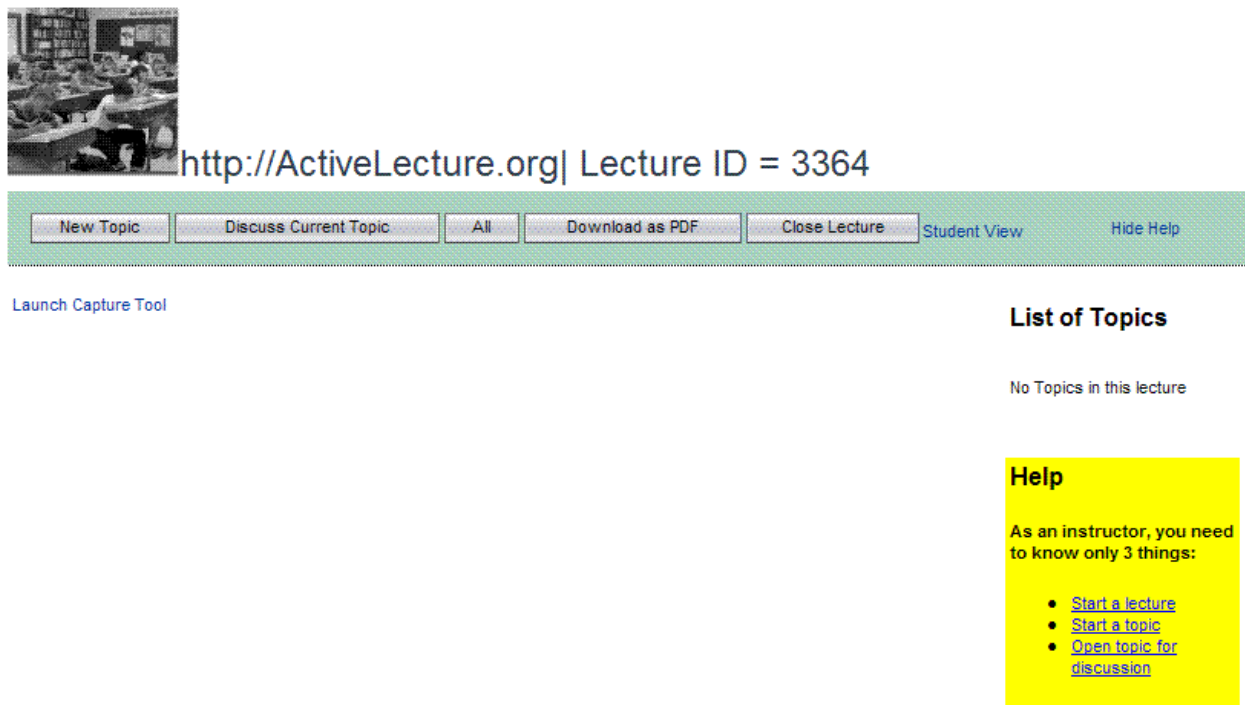
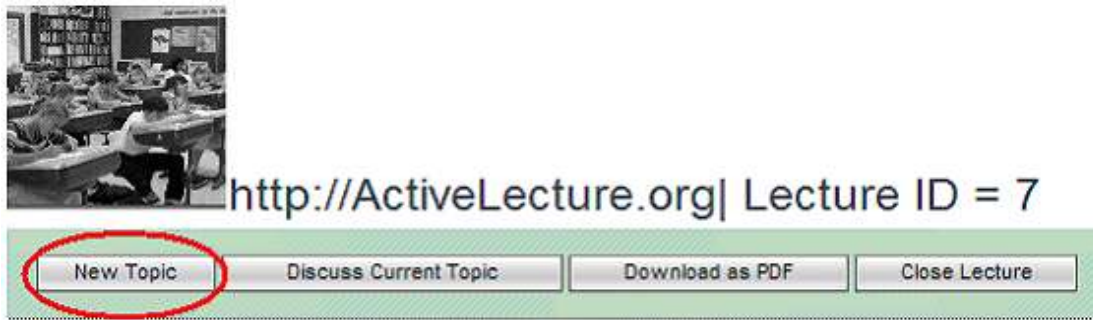


Figure 7 Instructor Lecture Page

### 3.1.3 Add New Topic

A lecture primarily contains several topics. An instructor can start a topic and make this topic available for discussion. The topics can be entered as either a Text Input or Image Capture and they are entered using “New Topic” or “Launch Capture Tool,” respectively. In Ubiquitous Presenter, all the slides are loaded from a PowerPoint presentation [8]. The instructor can only “Add Whiteboard” while lecturing [10] which limits the creativity of the instructor in the class.

The Active Lecture system does not have this limitation. An instructor can also add a new topic with text input by selecting “New Topic” button as shown in Figure 8.



Launch Capture Tool

Figure 8 Add New Topic

This will open a new topic region as shown in Figure 9. The instructor can enter the text for the new topic and click on “Save” to save the new topic.

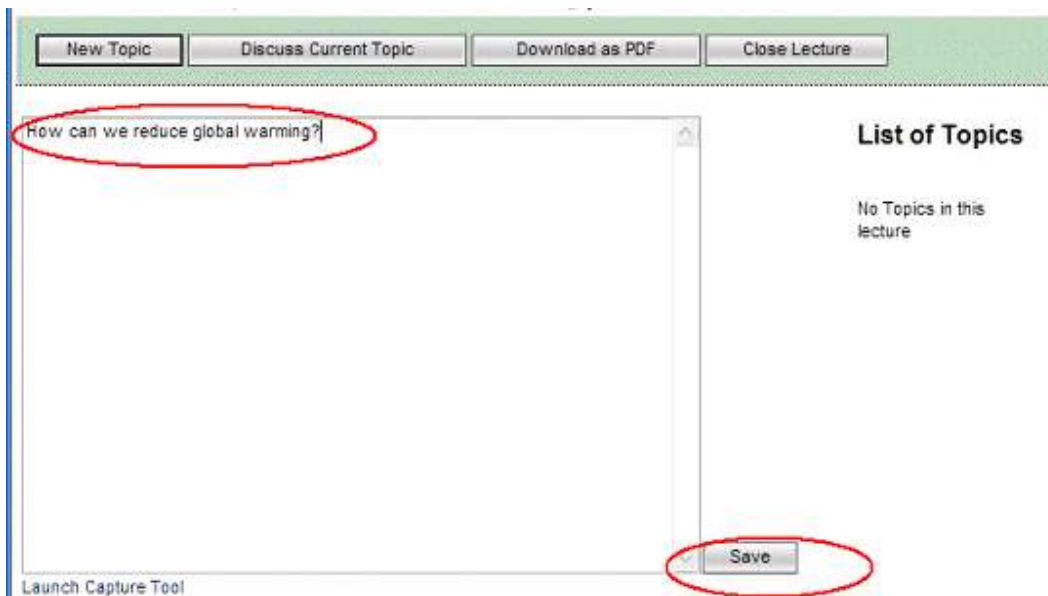


Figure 9 Enter New Topic

### 3.1.4 Launch Capture Tool

The launch capture tool allows the user to add a screen shot as a topic to the lecture. The launch capture tool is engaged by clicking on “Launch Capture Tool” link in the main lecture page as shown in Figure 10. When the user clicks on “Launch Capture Tool”, a small window, which allows a user to take screen shots, is available.

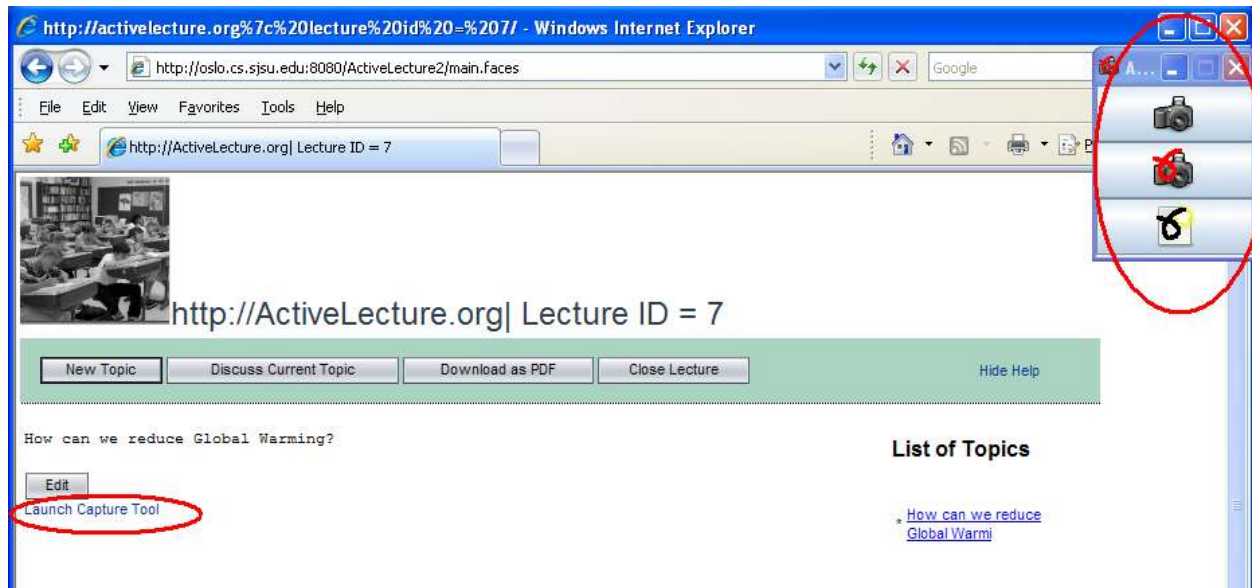


Figure 10 Launch Capture Tool

An Instructor can capture the images in three ways:

1. Screen shot
2. Screen shot and Draw
3. Draw

This capture tool is used when the instructor is giving a demo to the students using some other tool (i.e., Java IDE) and wants to take a screen capture. Using the first option, the instructor can simply save the screen capture as part of the lecture notes or allow the students to provide responses to the new image topic. Using the second option, an instructor can draw or write on the captured screen shot before saving it. Using the third option, an instructor can draw or write on a whiteboard. This is achieved by using the drawing tool of the “Launch Capture Tool”. An Instructor can save and exit the drawing tool by clicking on the “Send” icon as shown in Figure 11. This image is automatically added as a new topic to the lecture.

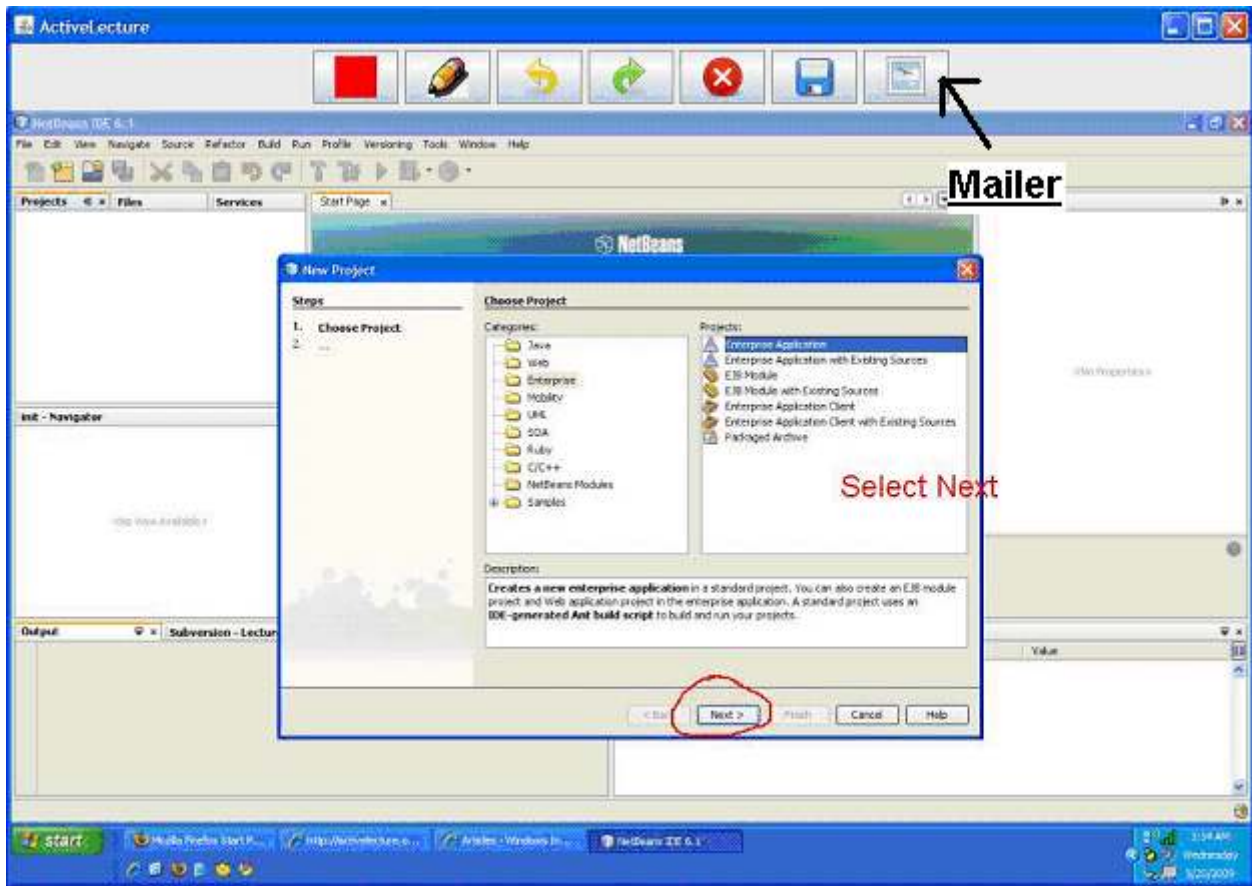


Figure 11 Save Captured Image

### 3.1.5 List of Topics

The “List of Topics” region will show the topics that are added to this lecture as shown in Figure 12.



Figure 12 List of Topics

The topics list is refreshed every 5 seconds by polling the server to check whether any new topics are available. In this way, when an image is added to the server, “Image” is listed as a topic in the list. Instructors can open this image topic by selecting the image link as shown in Figure 13.

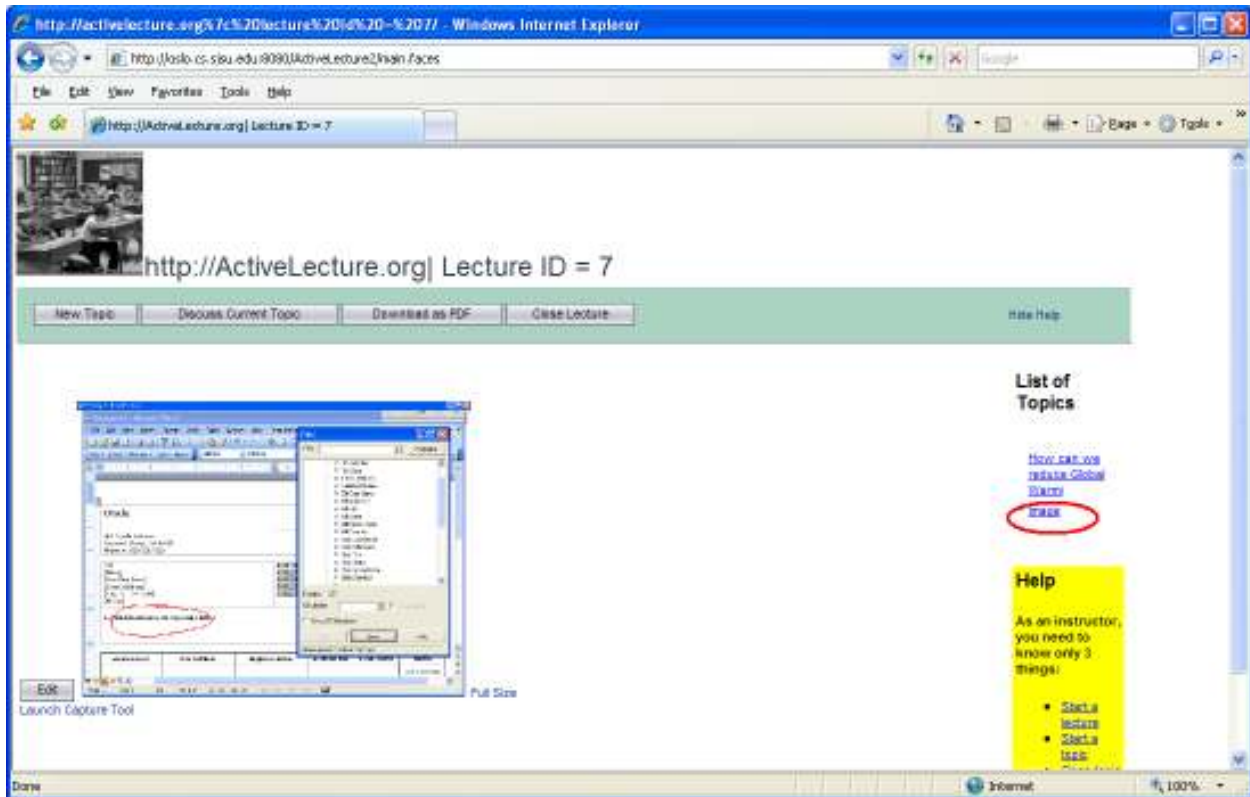


Figure 13 Automatic Addition of Image Topic

The current topic in this “List of Topics” is indicated by “\*” in front of the topic.

### 3.1.6 Discuss Current Topic

In the Active Lecture system’s main lecture page, any topic that is being viewed can be enabled for discussion. The students will be able to submit responses only to the topics that are open for discussion. The instructor has to select the “Discuss Current Topic” button to enable the current topic for discussion and allow students to provide responses. The instructor can wait while the students submit the responses. The instructor can also move on to another topic while allowing students to submit responses to the topic for which discussion is still enabled. The instructor can come back to the topic and disable the discussion on that particular topic. A sample main page with a topic under discussion is shown in Figure 14. The number of responses submitted by the students is shown on the page. The instructor can also peek at the responses submitted by the students, while the topic is still under discussion. If the instructor stops the discussion, the student submitted responses are listed at the end of the topic.

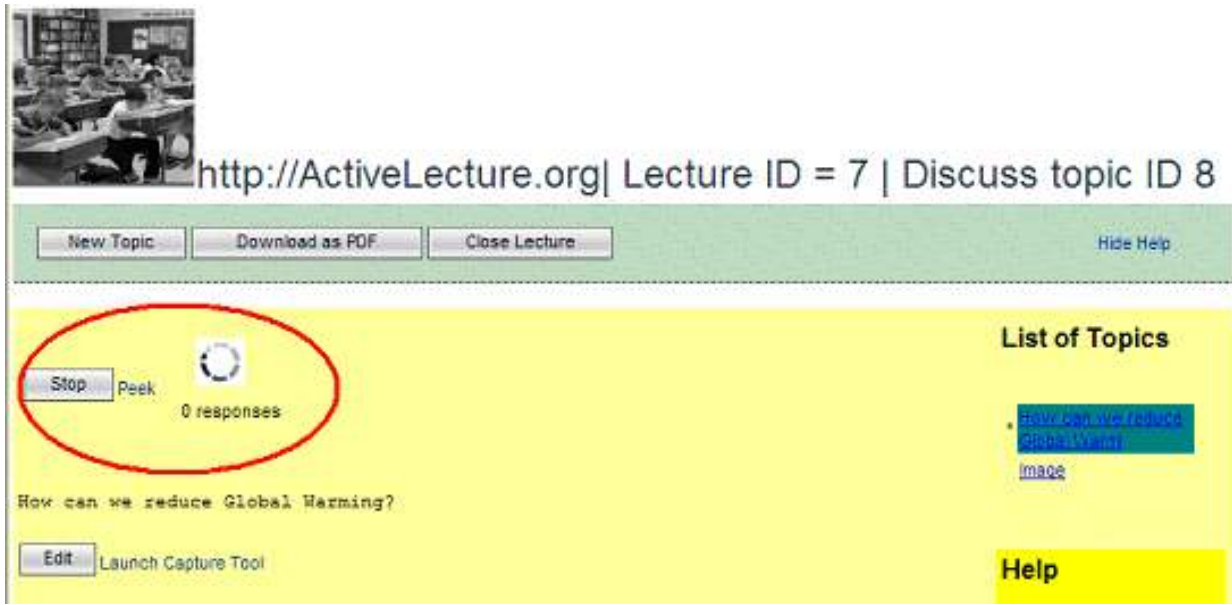


Figure 14 Discuss Current Topic

### 3.1.7 Download as PDF

The lecture notes can be downloaded as a PDF document as shown in Figure 15. This will enable the instructor to download and make it available at a particular location. This will also allow instructors to publish lecture notes in a separate location independent of the system. Students do not have to worry about unavailability of the system while they are preparing for exams and they can access the lecture notes even when the Active Lecture system is not available. The downloaded lecture notes contain both the topics and the student responses in the current lecture. This feature is available for both the instructor and students.

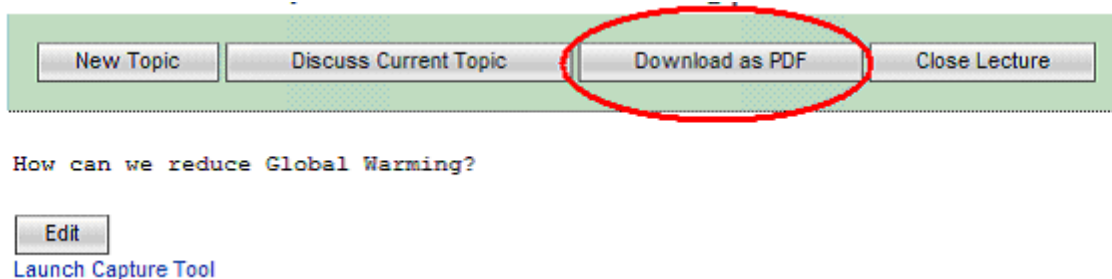


Figure 15 Download as PDF

### 3.1.8 Student View from Instructor Interface

In the main lecture page of the instructor interface, there is a provision to view the student interface of the current lecture. The student view as shown in Figure 17 will be opened in a separate window or tab without closing the current open lecture interface. This will allow the instructor to view and perform the operations of a student from the same browser. This feature will allow the instructor to know whether the students will be able to view the topics and submit

responses. The student view can be opened by clicking on the “Student View” link from the instructor main lecture page as shown in Figure 16.

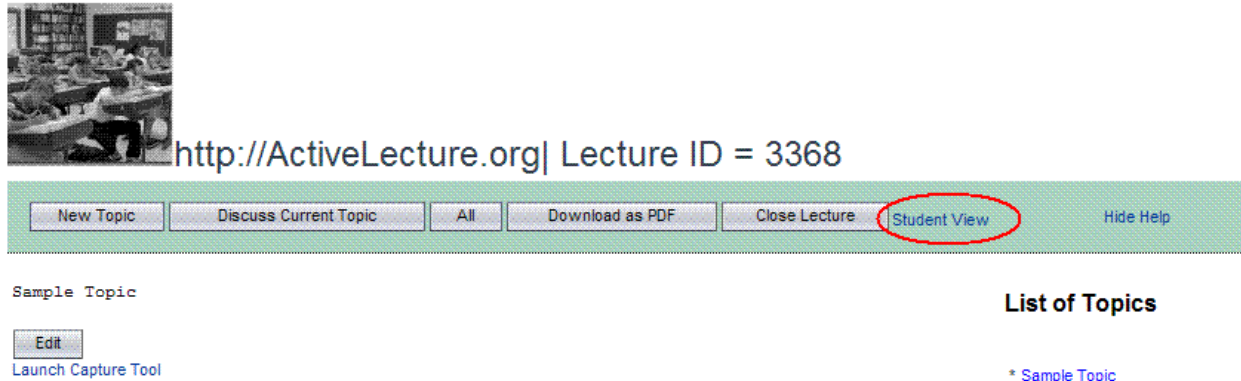


Figure 16 Student View Link in Instructor Interface

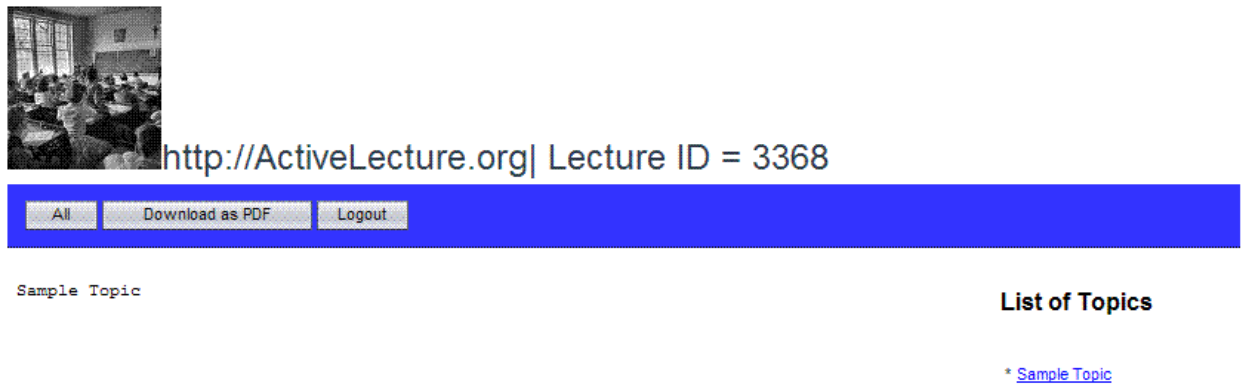


Figure 17 Student Interface Window

### 3.1.9 Configurable Help

The Active Lecture system has a configurable help functionality that is marked in Figure 18. The Help for the instructor is shown in yellow region. The instructor can hide the help region by clicking on “Hide Help” link in the buttons panel. This replaces the “Hide Help” link with “Show Help” link and hides the yellow Help region. An instructor can view the help region again by clicking on “Show Help” link as shown in Figure 19.



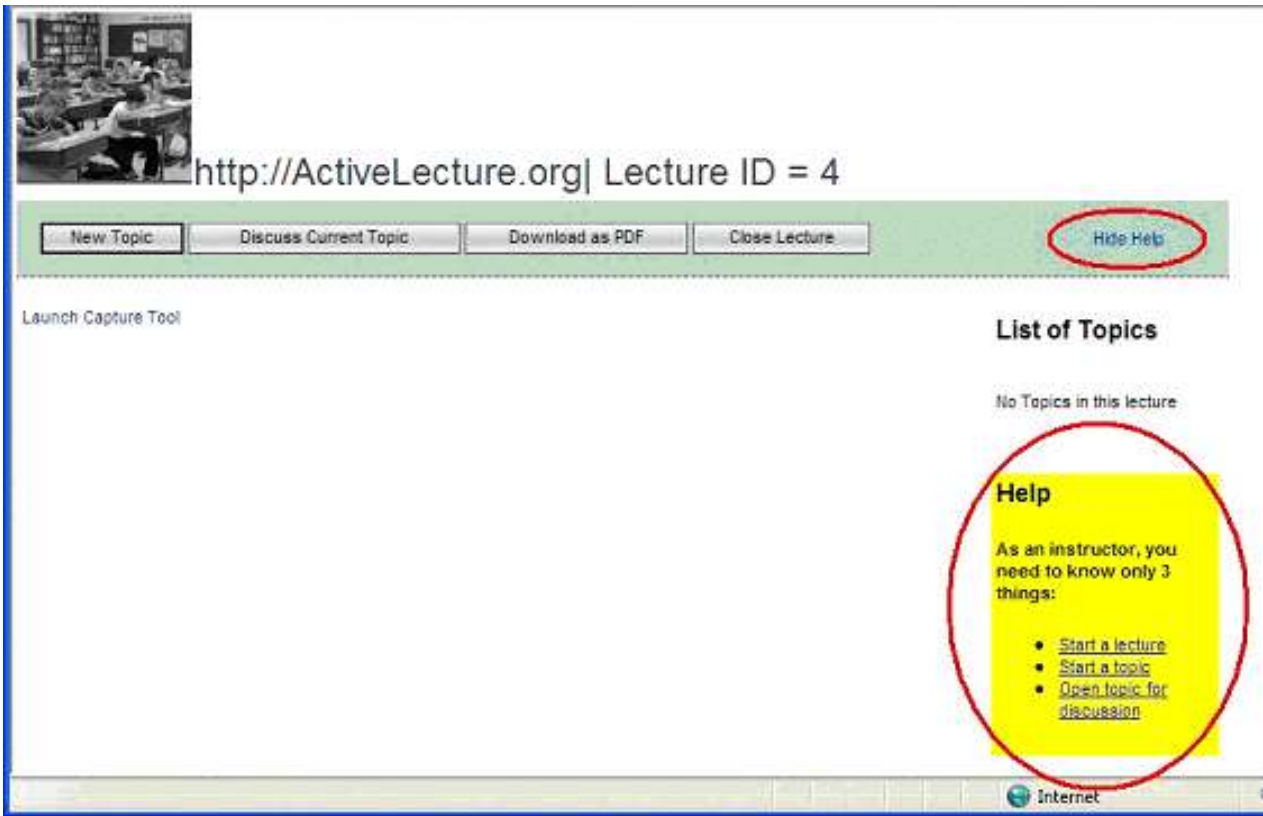


Figure 18 Configurable Help



Figure 19 Show Help

**Help**

As an instructor, you need to know only 3 things:

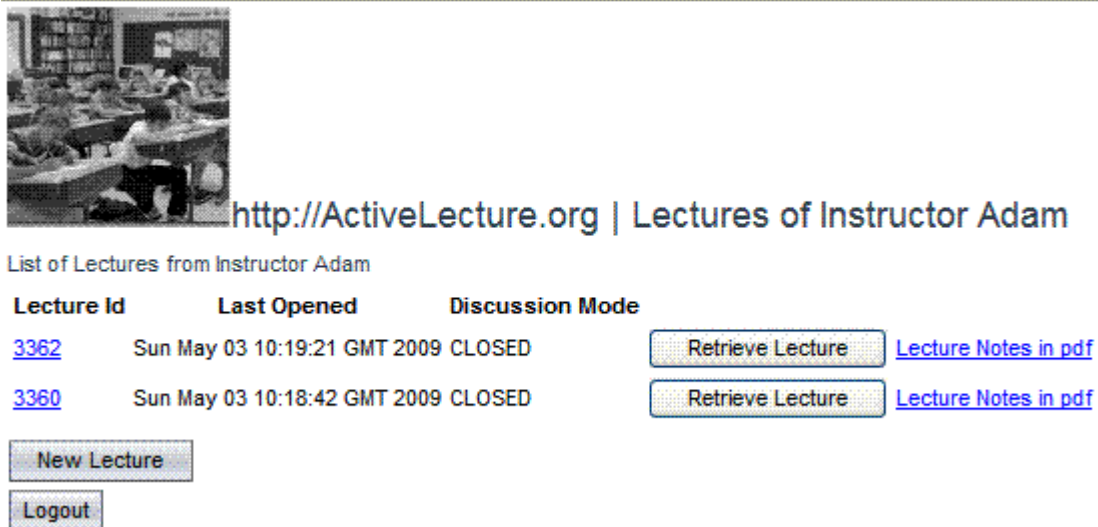
- [Start a lecture](#)  
You have already done this.
- [Start a topic](#)
- [Open topic for discussion](#)

Figure 20 Help Details

In the Help region, instructors can select any of the links to see detailed help for the selected help item as shown in Figure 20. They can unhide the help detail by clicking on the same help item again.

### 3.1.10 Manage Lectures

“Manage Lectures” is available for registered instructors. Registered instructors can log in and access the list of lectures that they have created. The list of lectures lists the lecture ID, when it was last opened, and whether the lecture is open or closed for discussions as shown in Figure 21. The instructor can retrieve any specific lecture to continue the lecture or access the lecture notes for any lecture. The instructor can also start a new lecture from the “Manage Lectures” page. This “Manage Lectures” feature can be accessed from the instructor login page using “List Instructor’s Lectures”, or “List All Lectures” from the main lecture page.



http://ActiveLecture.org | Lectures of Instructor Adam

List of Lectures from Instructor Adam

Lecture Id	Last Opened	Discussion Mode
<a href="#">3362</a>	Sun May 03 10:19:21 GMT 2009	CLOSED <input type="button" value="Retrieve Lecture"/> <a href="#">Lecture Notes in pdf</a>
<a href="#">3360</a>	Sun May 03 10:18:42 GMT 2009	CLOSED <input type="button" value="Retrieve Lecture"/> <a href="#">Lecture Notes in pdf</a>

Figure 21 Manage Lectures for Registered Instructors

### 3.2 Student Interface Features

Students use the student interface of the system. The features of the student interface are as follows:

1. Highlight the discussion topic
2. Multi-part student response
3. Allow multiple student responses
4. Anonymous student response
5. Automatic refresh of topics
6. Downloading lecture notes

#### 3.2.1 Highlight the Discussion Topic

An important feature of the system is that when a lecture topic is open for discussion, the instructor interface will show a timer waiting for the inputs from the students. This will let the students know that they can respond to the topic during this time. The topic that is open for discussion will be highlighted in yellow in the instructor interface as well as the student interface. This will allow the students to identify which topic is under discussion. In the list of topics for both the student and the instructor, these topics will be highlighted in a different color which makes it easier to differentiate from any other topic that is not available for discussion. Once a student selects a topic that is available for discussion, the topic details will be highlighted in yellow, similar to that shown in the instructor interface. There is also a message on the top of the topic that indicates that the current topic is open for discussion. Students can select the Discuss button to submit a response to the topic.

#### 3.2.2 Multi-part Student Response

Once a student selects the option to discuss, a new editable page is available for the student to provide a response to the current topic in discussion. The interface for providing the response is also created with a simple interface even though there are different ways to submit a response as shown in Figure 22.

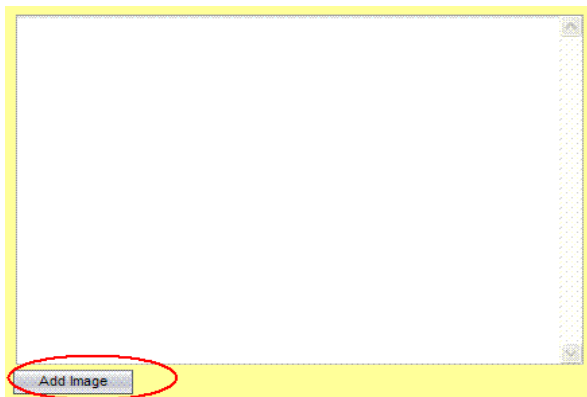


Figure 22 Student Response Page

In the Active Lecture system, the student response can be of three types. It can be text, an image, or a choice list. The user can also submit the response using one or multiple types of input for the response. This is a useful feature because the students will be able to provide the responses in their own way. The text response feature allows the students to submit the responses with their own ideas and explanations. For example, when the instructor asks a multiple choice question by opening a topic for discussion, the students can not only select a choice as a response, they can even enter the explanation about why they think that answer is correct. This will allow the instructor and the rest of the class to understand why each answer is selected. Many times those reasons may be valid or provide impetus for further discussion.

Students can also submit images as part of the response. When an instructor asks the students to perform a task in any tool that the class may be discussing, the best way to submit a response is by taking a screen shot. The student can also provide a brief explanation in the text region that will be a description for the image or screen capture. Students also are able to draw on screen shots that they have taken. This allows the students to pinpoint what they are discussing. They can even include text as part of the screen shot. It literally is a drawing board. Initially, the drawing board is blank or it can have a background image.

This provision is similar to the drawing board used by the instructor. Students will use the drawing board to add an image as part of a response to any existing topic. However, instructors can create new topics with the drawing board. When a student takes a screen shot or saves a drawing in the drawing board, the image is shown in the response screen automatically. The student can then enter other details as text or a choice before submitting a response. The student will know what response he or she is submitting and how this response looks. Once a student submits a response, he or she is given a confirmation message saying that the response is submitted.

### *3.2.3 Allow Multiple Student Responses*

There is a provision to submit more than one response from the same machine to an ongoing topic that is available for discussion. This is a very useful feature as this allows the students to share the same machine and submit responses individually. For example, if a student “A” does not have a computer and wants to submit a response, he or she can use the computer of another student and submit the response. Even though Student “B” has submitted a response for the open topic, the second response is also accepted by the system. This helps in a classroom where multiple students share computers.

### *3.2.4 Anonymous Student Response*

The most important feature of the system is to allow the students to submit all the responses anonymously. Multiple students can submit responses to the lecture topics without moving out of the current page as the students do not logout or log in to the Active Lecture system. Students will not shy away from answering a question in the Active Lecture system, as they remain anonymous when they are answering. The students can even provide explanations that will give them more options to express ideas and present their thoughts to the class.

### *3.2.5 Automatic Refresh of List of Topics*

The list of topics section in the student interface is designed to refresh periodically to display the topics that are listed in the instructor's main page. After a student joins a lecture, if the instructor adds any topic, the student interface will automatically list the new topic.

### *3.2.6 Downloading Lecture Notes*

Students also have the option to download the lecture notes at any time. They can get the lecture notes, while the lecture is going on or when the lecture is over. These lecture notes can be downloaded as a file in PDF format. Students can save the lecture notes locally and review the notes when they are offline.

## 4. Implementation of the Active Lecture System

The Active Lecture system is designed and implemented as an integrated solution of two Java applications. The first application is a server-side Java Enterprise Edition (JavaEE) application. The second application is a Java client application. The basic interface of the Active Lecture system is developed as a server-side application. The capturing tool is developed as a client application and is launched from the Active Lecture server.

### 4.1 Active Lecture Server

The Active Lecture Server application is developed using JavaEE technology. This is also called a middleware application. This follows the Model-View-Controller approach. This middleware application has two parts, a Model, and a User Interface (UI). The Model part contains the Enterprise Java Beans implementation of the application. The Model contains connections to the database and entity beans code that accesses and manipulates database objects. The Model also contains session beans that are used in the UI, which internally call entity beans to make the changes to the database objects. The UI project contains the Java Server Faces (JSF) pages, managed beans, and Java Servlets. Both the instructor and student user interfaces are developed in this project.

#### 4.1.1 Database Objects

The database objects that are used by Active Lecture Server are stored in a PostgreSQL database. The entity-relationship diagram for the tables used by Active Lecture Server is shown in Figure 23.

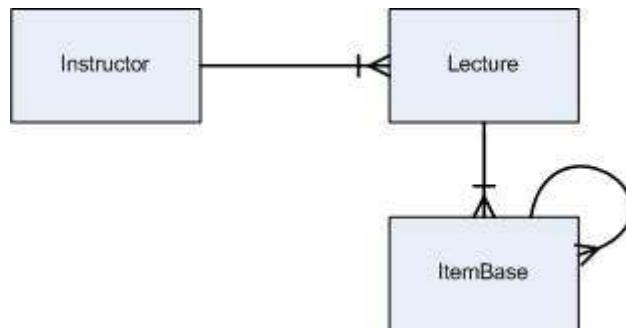


Figure 23 Table Diagram

#### 4.1.2 Enterprise Java Beans

In Active Lecture Server, Enterprise Java Beans provide the interface to perform database operations. There are two types of Java Beans, entity beans and session beans. Entity beans contain the interface for the individual tables. Session beans provide operations that are performed on the entity beans. Session beans are available for the UI project to perform any operations on the database objects. The object model of the EJBs in the Active Lecture system is shown in Figure 24.

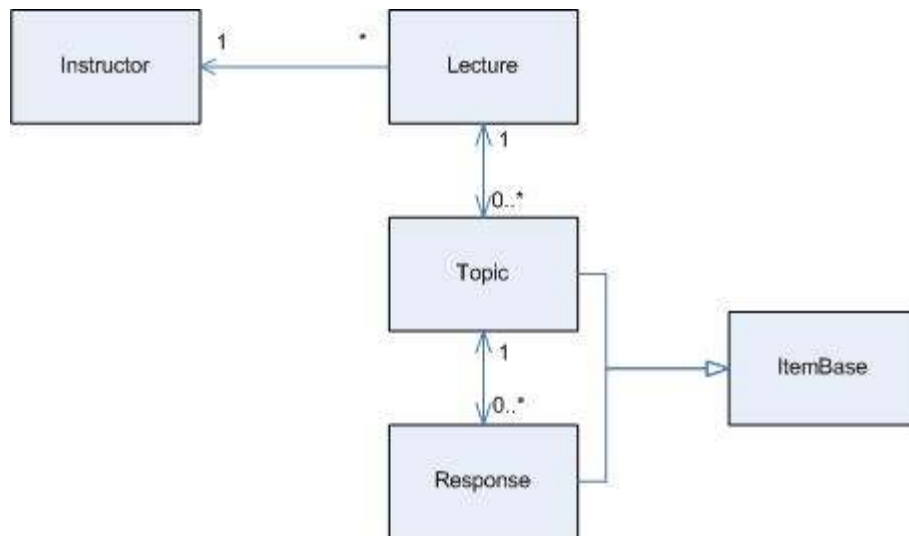


Figure 24 Object Diagram

The following sections will explain the implementation of some of the interesting features in detail.

#### 4.1.3 Email Lecture Details

When an unregistered instructor starts a new lecture by providing an email address, the Active Lecture system will send an email to the instructor with an URL to the Active Lecture website and lecture ID of the new lecture. One of the key features to support instructors is to send the lecture information as email to them. The Active Lecture system uses the Java Mail API in JavaEE to implement this feature. The mail resource that is used by the Active Lecture system needs to be setup in the Java application server as "mail/ActiveLectureMail". The JavaMail session resource should be on the Java application server as shown in Figure 25 and Figure 26.

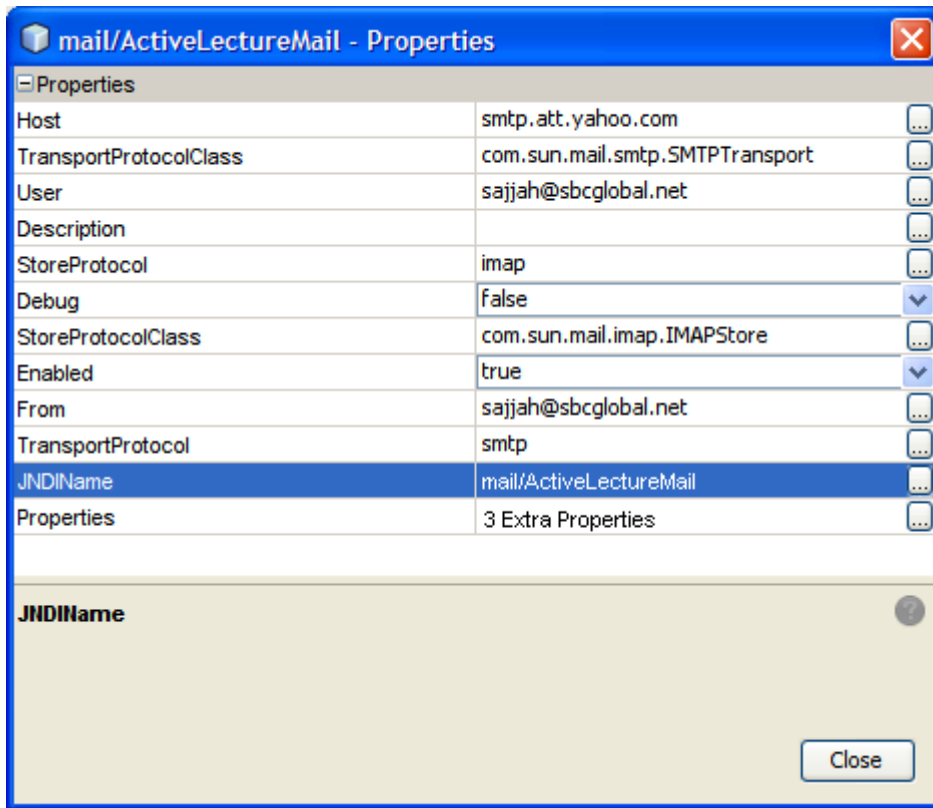


Figure 25 Properties for JavaMail Setup

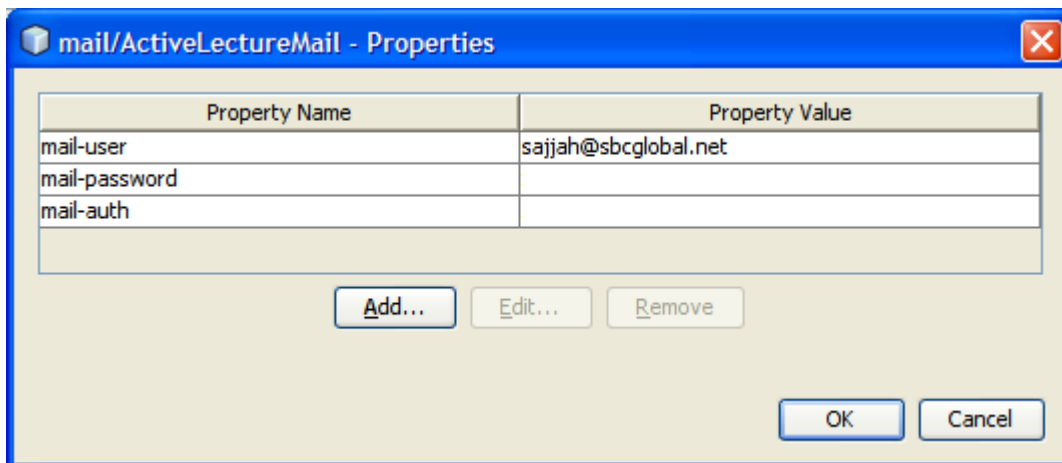


Figure 26 Additional Properties for JavaMail Setup

In the code, the JNDI name for the mail resource is hard coded. The resource name for the mail resource is specified in the code using `@Resource(name = "mail/ActiveLectureMail")` annotation. Then, a mail session object is created using the class `javax.mail.Session`. The code fragment for this declaration is shown below:



```
@Resource(name = "mail/ActiveLectureMail")
private Session mailSession;
```

Sending mail using JavaMail uses the following process. First a message object is created for the mail session using `new MimeMessage(mailSession)` with the following properties and save using `saveChanges()` method.

1. Recipient
2. Subject
3. Text

Later, the `mailSession` gets the Simple Mail Transfer Protocol transport object, connects to the mail server, sends the message to the recipients, and closes the connection.

#### *4.1.4 Automatic Refresh of List of Topics*

The list of topics region is available for both instructor and student. This region lists the topic names of all the topics in the current lecture. If the topic is a text topic, the first 30 characters of the topic text are displayed as the topic name. If the topic is an image, `Image<number>` is displayed as topic name. The `<number>` is a sequence number in each lecture that starts with 1 and is incremented by 1.

Design and development of “List of Topics” region is complex because this region needs to show the topics added by both Active Lecture Server and Active Lecture Client. The other complexity is to show the list of topics in both the instructor and the student interfaces. The list of topics region, in the student main page, needs to be refreshed periodically to reflect the topics in the server.

The process to refresh the list of topics region is implemented using AJAX. To achieve this, two attributes are defined in the main lecture page. One attribute is defined as a hidden static text; the other is defined as the placeholder panelgroup that will render the HTML text. The JSF declaration for these attributes is as show in the code fragment below.

```
<w:staticText id="tableText" text="{mainSB.topicsTable}"
  visible="false"/>
<w:panelGroup id="pgTopicsTable" block="true"/>
```

The timer for automatic refresh is started on the internet browser, when the HTML page is loaded, using the tag `<w:body onLoad="formLoad()">`. When the body of the HTML page is loading, the `formLoad()` JavaScript function, as shown in below code fragment, is invoked.

```
function formLoad(){
  // to show the topics on page load
  var b1 = setTimeout('loadField()',100);

  // to refresh list of topics based on time interval
  var b = setInterval('refreshField()',5000);

  return b1;
```

```
}
```

The timer invokes the `refreshField()` method for each time interval. This in turn calls the `domNode2.refresh()` method to refresh the static text attribute with the actual HTML for the list of topics table. The static text refresh will execute a method in the managed bean, which executes the query on the database and fetches the current list of topics. This method then generates an HTML table for the list of topics. The HTML code generated is refreshed on to the `tableText` field. Later, the `innerHTML` property of the placeholder panel is set to the new HTML table code and this is rendered dynamically on the page. In this way, the list of topics is refreshed in both the instructor and the student interfaces. The JavaScript code for the `refreshField()` function is shown in the below code fragment.

```
function refreshField() {
    // Hidden field which contains the HTML for topics table.
    var domNode2 =
        document.getElementById("form1:pgp1:pgTopics:tableText");

    // HTML component on the page that displays the list of topics
    var domNode3 =
        document.getElementById("form1:pgp1:pgTopics:pgTopicsTable");

    // Refresh to run the serverside code
    // to fetch the current list of topics
    var a = domNode2.refresh();

    // Replace the existing HTML with the new list of topics html
    domNode3.innerHTML=domNode2.getProps().value;

    return a;
}
```

The backing bean code used for refreshing the table is shown in the below code fragment.

```
public String getInternalTopicsTable
(long cItemId, long cReferencedTopicId) {

    topics = lectures.getLectureTopics(lectureId);

    if (this.topics == null || this.topics.isEmpty()) {
        return "No Topics in this lecture";
    }

    int imageCount=0;
    StringBuilder tablesb = new StringBuilder("<table>");

    for (Topic topic : this.topics) {
        if ((topic.getText() != null) || (topic.getImageId() != null)) {

            tablesb.append("<tr><td>");

            if (cReferencedTopicId > 0) {
                tablesb.append(cReferencedTopicId == topic.getId()
                    ? "*" : "");
            } else {
                tablesb.append(cItemId == topic.getId() ? "*" : "");
            }
        }
    }
}
```

```

        tablesb.append("</td>");

        if (topic.getOpened()) {
            tablesb.append("<td class='openTopicStyle'>");
        } else {
            tablesb.append("<td>");
        }

        tablesb.append("<a id=\"topics1\"
            href=\"#\" onclick=\"submitTopic('\";
            tablesb.append(topic.getId());
        tablesb.append(\"'\">");
        String a = topic.getText();

        if (topic.getImageId() != null) {
            tablesb.append("Image").append(++imageCount);
        } else if ((a != null) && !(a.equals(""))) {
            tablesb.append((topic.getText().length() > 30
                ? topic.getText().substring(0, 30)
                : topic.getText()));
        }

        tablesb.append("</a>");
        tablesb.append("</td></tr>");
    }
}

tablesb.append("</table>");

return tablesb.toString();
}

```

#### 4.1.5 Discuss Current Topic

The instructor selects “Discuss Current Topic” to discuss the currently displayed topic. The “Discuss Current Topic” button is available only when a lecture is open for discussion and the current topic is not under discussion. This is shown in the code fragment below.

```

<w:button actionExpression="#{mainRB.start_action}"
    rendered="#{mainRB.pageData.lectureOpen and not
    mainRB.pageData.discussionOpen}" text="Discuss Current Topic"/>

```

When the topic is opened for discussion, a timer icon and the number of responses are displayed on the main lecture page. When a topic is under discussion, the number of responses on the lecture page is refreshed, every 10 seconds, with the number of responses in the database. This is implemented using a “Woodstock component” progress bar [21, 22]. Woodstock components are extensions to JSF. The Woodstock library contains JSF and AJAX components that can be used to make web applications more dynamic. The progress bar component periodically executes the mainRB.progressBar1 component specified in the binding attribute and refreshes the status on the page with the number of responses. The JSF definition for this component is as shown in the code below.

```

<w:progressBar binding="#{mainRB.progressBar1}" refreshRate="10000"
    status="#{mainRB.pageData.responseCount} responses"
    taskState="stopped" type="BUSY"/>

```

When the instructor clicks on “Discuss Current Action”, the task state of the progress bar is changed to `ProgressBar.TASK_RUNNING` as shown in the managed bean code below.

```
public String start_action() {
    save_action();
    mainSB.setDiscussionMode(true);
    progressBar1.setTaskState(ProgressBar.TASK_RUNNING);
    loadPageData();
    return null;
}
```

The `progressBar1` is refreshed every 10 seconds. This refresh runs the following managed bean setter method to reload the responses for the current topic.

```
public void setProgressBar(ProgressBar progressBar1) {
    loadPageData();
    this.progressBar1 = progressBar1;
}
```

Setting the task state to `ProgressBar.TASK_STOPPED` as shown in the below code fragment stops the progress bar action:

```
public String stop_action() {
    mainSB.setDiscussionMode(false);
    progressBar1.setTaskState(ProgressBar.TASK_STOPPED);
    loadPageData();
    return null;
}
```

There is a provision to peek at the responses while the topic is still under discussion. The JSF code fragment that opens a bubble with responses when the mouse moves over the link is shown below.

```
<w:hyperlink onMouseOver="openBubble(event);"
onMouseOut="document.getElementById
('form1:pgp1:pgp2:grp:bubble').close();" text="Peek"/>
```

The Java Script for the bubble that refreshes and displays the responses is shown in the below code fragment.

```
function openBubble(event) {
    var bubble = document.getElementById("form1:pgp1:pgp2:grp:bubble");
    bubble.refresh();
    bubble.open(event);
}
```

#### *4.1.6 Launching of the Active Lecture Client*

Both instructors and students use the Active Lecture Client application to create and upload images. However, the way the Active Lecture Client is launched and used is different for both of them.

For instructors, Active Lecture Client is launched using Java Web Start. When the instructor clicks on the “Launch Capturing Tool”, the Java application server generates a JNLP file and sends it to the browser. The browser launches Java Web Start and runs this file.

The code that generates the JNLP file is as shown below:

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.0+"
  codebase="<%=jnlpCodeBase%>"
  href="<%=jnlpRefURL%>">

  <resources>
    <j2se version="1.5+"/>
    <jar href="resources/Client.jar"/>
    <property name="com.horstmann.activelecture.lecture"
      value="<%= request.getParameter("lecture") %>"/>
    <property name="com.horstmann.activelecture.downloadURL"
      value="<%= request.getParameter("downloadURL") %>"/>
    <property name="com.horstmann.activelecture.uploadURL"
      value="<%= request.getParameter("uploadURL") %>"/>
    <property name="com.horstmann.activelecture.date"
      value="<%= new java.util.Date() %>"/>
  </resources>
  <security>
    <all-permissions/>
  </security>
  <application-desc main-class="com.horstmann.activelecture.Main"/>
</jnlp>
```

The JNLP file contains reference to the Active Lecture Client JAR file `resources/Client.jar` to be downloaded and the main class to be run. Java Web Start reads the information from the JNLP file and runs the Java class listed in that file. The important tags in the JNLP file are `<resources>`, `<security>`, and `<application-desc>`. The `<resources>` tag provides the details about where to find the classes that need to be run by Java Web Start. Active Lecture Client classes are available in `resources/Client.jar`. Java Web Start downloads this JAR file and runs the main class given in the tag `<application-desc main-class="com.horstmann.activelecture.Main"/>`. In this case, it runs the class “`com.horstmann.activelecture.Main`”. The Active Lecture Client is launched with properties that are passed in the JNLP file. These properties pass the variables from the server application to the Active Lecture Client. The important properties that are used by Active Lecture Client are lecture ID, download URL, and upload URL. This JNLP file is sent to the browser using URL rewriting from a JSP file.

```
response.setContentType("application/x-java-jnlp-file");
```

The JNLP file should not be cached in the browser because the Active Lecture Client needs to be launched from the same browser for different lectures and topics. If the JNLP file is cached, the parameters that are passed in the JNLP file will launch the Active Lecture Client for the same lecture. This caching issue is resolved by setting the following to the HTTP response object.

```
response.setHeader("Expires", "Sat, 6 May 1995 12:00:00 GMT");
response.setHeader("Pragma", "no-cache"); //HTTP 1.0
```

```

response.setHeader("Cache-Control", "no-store, no-cache,
    must-revalidate, private");

//Microsoft
response.addHeader("Cache-Control", "post-check=0, pre-check=0");

//prevents caching at the proxy server
response.setDateHeader("Expires", 0);

```

For students, the Active Lecture Client is launched as a Java applet. Students will be able to upload images when they are submitting a response using the Active Lecture Client. Students can select a choice, enter text, or add an image to the response. Students can click on the “Add Image” button that redirects the browser to the student upload page. This student upload page contains an applet based on the Active Lecture Client. This applet launches the user interface of the Active Lecture Client. The code specified below launches the Active Lecture Client as an applet.

```

<applet width="1" height="1" code="com/horstmann/activelecture/Main.class"
archive="resources/Client.jar"
<h:outputText escape="false" value="&lt;param
name='com.horstmann.activelecture.name'
value='{mainSB.studentUploadName}' /&gt;"/>
<h:outputText escape="false" value="&lt;param
name='com.horstmann.activelecture.next'
value='{uploadRB.studentPageURL}' /&gt;"/>
<h:outputText escape="false" value="&lt;param
name='com.horstmann.activelecture.downloadURL'
value='{uploadRB.studentDownloadURL}' /&gt;"/>
<h:outputText escape="false" value="&lt;param
name='com.horstmann.activelecture.uploadURL'
value='{uploadRB.studentUploadURL}' /&gt;"/>
<h:outputText escape="false" value="&lt;param
name='com.horstmann.activelecture.student'
value='{mainSB.student}' /&gt;"/>
<h:outputText escape="false" value="&lt;param
name='com.horstmann.activelecture.lecture'
value='{mainSB.lectureId}' /&gt;"/>
</applet>

```

As seen in the code above, the student upload also runs the same class `com/horstmann/activelecture/Main.class` in `resources/Client.jar`. However, as this is an applet, the method that is called will be the applet life cycle method `init()`. Students can add only one image per response. The image name is generated when the student clicks on “Add Image” button and passed as parameter to the Active Lecture Client applet. The parameter that is used is `com.horstmann.activelecture.name`. The student upload program also passes the lecture ID as parameter to the applet. The Active Lecture Client program uses these parameters to upload an image and attach it to the appropriate response.

#### 4.1.7 File Upload Servlet

An image is uploaded, from the client application, by submitting an HTTP request to the `FileUpload` servlet. In the server, file upload directory can be configured using `com.horstmann.activelecture.FileUpload.directory` variable in the `web.xml`.

The file upload servlet uses the Apache Commons FileUpload [23] to upload the files. This servlet takes file name, lecture ID, and image content as multipart content. The following code fragment will provide the details about how the `FileItem` is extracted from the HTTP request and saved into a file.

```
boolean isMultipart =
    org.apache.commons.fileupload.FileUpload.isMultipartContent(request);

if (isMultipart) {
    DiskFileUpload upload = new DiskFileUpload();

    upload.setRepositoryPath(directory.getAbsolutePath());
    try {
        List<FileItem> items = (List<FileItem>) upload.parseRequest(request);
        FileItem fileItem = null;
        String name = null;
        String lectureId = null;
        boolean student = true;

        for (FileItem item : items) {
            if (item.isFormField()) {
                if (item.getFieldName().
                    equals("com.horstmann.activelecture.name")) {
                    name = item.getString();
                } else if (item.getFieldName().
                    equals("com.horstmann.activelecture.student")) {
                    student = !item.getString().equals("false");
                } else if (item.getFieldName().
                    equals("com.horstmann.activelecture.lecture")) {
                    lectureId = item.getString();
                }
            } else {
                fileItem = item;
            }

            if (name != null && fileItem != null) {
                File file = new File(directory, name);
                fileItem.write(file);
            }
        }
    }
}
```

#### 4.1.8 Generating Lecture Notes in PDF

The generation of lecture notes in PDF is implemented as a servlet called `LectureNotes`. The `LectureNotes` servlet takes the `lectureId` as URL parameter and generates the PDF document using the `iText` API [24]. `iText` is a Java based API for PDF generation. We can generate PDF documents dynamically by calling this API. The approach to generate a PDF is shown in the code below. First, the response content type is set to `application/pdf`. Then, a `Document` object is created using the `iText` class `com.lowagie.text.Document`. This document is set as object for response output stream. Later, the `Document` object is opened using `document.open()` and the data that needs to be generated to the PDF is added to the opened document using the `document.add()` method. After adding the content, the document is closed using

`document.close()`. This stops the streaming to the browser and the PDF document is displayed on the browser.

PDF generation in the Active Lecture system adds both text and images to the PDF document. First, the document generation section queries all the topics in the current lecture. It then iterates through each topic and identifies whether it is a text topic or image topic. If the topic is a text topic, the content is added as a new paragraph for each topic. If the topic is an image topic, a new image instance is generated using the class `com.lowagie.text.Image` by passing the URL for the image. The image is scaled down to 50 percent. The scaled image object is then added to the document object. The responses for each topic are retrieved and all the responses are also added to the document under the corresponding topic. If the response has text and image, text is added to the PDF document as a paragraph followed by the image portion. If the response contains only text or image, it adds the text or image to the document. This addition of the responses is done iteratively until all the responses to the corresponding topic are added to the document.

```
response.setContentType("application/pdf");
Document document = new Document();

PdfWriter.getInstance(document, response.getOutputStream());
document.open();

for (PageData pageData : pageDataList) {
    if (pageData.getReferencedImageId() != null) {
        String url = urlContext + "/getImage?id=" +
            pageData.getReferencedImageId();
        Image jpg3 = Image.getInstance(new URL(url));
        jpg3.scalePercent(50);
        document.add(jpg3);
    } else if (pageData.getReferencedText() != null) {
        document.add(new Paragraph(pageData.getReferencedText()));
    }

    if (pageData.getText() != null) {
        document.add(new Paragraph(pageData.getText()));
    }

    if (pageData.getImageId() != null) {
        String url = urlContext + "/getImage?id=" +
            pageData.getImageId();
        Image jpg3 = Image.getInstance(new URL(url));
        jpg3.scalePercent(50);
        document.add(jpg3);
    }

    List<ResponseData> responsesList = pageData.getResponses();

    if (responsesList!=null)
    {
        for (ResponseData responseData : responsesList) {
            if (responseData.getFrequency() > 1) {
                document.add(new Paragraph(responseData.getFrequency()));
            }

            if (responseData.getText() != null) {
                document.add(new Paragraph(responseData.getText()));
            }
        }
    }
}
```



```

        if (responseData.getImageId() != null) {
            String url = urlContext + "/getImage?id=" +
                responseData.getImageId();
            Image jpg3 = Image.getInstance(new URL(url));
            jpg3.scalePercent(50);
            document.add(jpg3);
        }
    }
}

document.close();

```

#### 4.1.9 Configurable Help

“Configurable Help” feature is developed using JSF, but the hide/show items of the Help region are developed using Java Script. The rendered property of the “Show Help” and “Hide Help” links is set in such a way that the “Show Help” link is enabled when the session variable `showHelp` is `false`, but “Hide Help” is enabled when `showHelp` is `true`. In the Help region, help is listed as three bullet links. When the user selects each link, more detailed help is shown for the selected link. This is achieved using the following Java Script. Two CSS classes that are used to hide and show the help details are listed in the following code fragment:

```

.hidden { display: none; }
.unhidden { display: block; }

```

The JavaScript function that is used to change the CSS class is as shown below:

```

function unhide(divId)
{
    var i = document.getElementById(divId);
    if (i)
    {
        i.className=(i.className=='hidden')?'unhidden':'hidden';
    }
}

```

The code to hide and show the help details in the help region is as follows:

```

<ul>
  <li>
    <a href="javascript:unhide('StartLecture');">
      Start a lecture
    </a>
  </li>
  <div id="StartLecture" class="hidden">
    You have already done this.
  </div>
  <li>
    <a href="javascript:unhide('StartTopic');">
      Start a topic
    </a>
  </li>
  <div id="StartTopic" class="hidden">

```

```

        Select "New Topic" button.
    </div>
    <li>
        <a href="javascript:unhide('DiscussTopic');">
            Open topic for discussion
        </a>
    </li>
    <div id="DiscussTopic" class="hidden">
        Select "Discuss Current Topic" button.
    </div>
</ul>

```

By default, the `div` section is hidden as the `hidden` class is used. When the user clicks on the link, `unhide('StartLecture')` is executed to toggle the class to `unhidden`. If the instructor clicks the second time, the class is toggled from `unhidden` to `hidden`. All the other links in this Help region use the same technique.

## 4.2 Active Lecture Client

Active Lecture Client is a Java based graphical user interface application. Active Lecture Client has the following features:

1. Capture screen shot
2. Save image
3. Packaging and signing the client

### 4.2.1 Capture Screen shot

The instructor or student can take a screen shot from Active Lecture Client by clicking on the camera icon or the camera with scribble icon. When the user clicks on the camera, Active Lecture Client will take the screen shot and upload it to the server. However, when the user clicks on the camera with scribble icon, the screen shot is copied onto a scribble pane where the user can add ink and text on the screen capture before saving the image to the server. In both cases, the screen shot taken is resized to .75 scale factor so that the instructor will not confuse it with the actual window.

The code fragment to take the screen shot is shown below. First, the controller window is hidden and then the screen capture is taken to the buffer. The buffered screen capture is then drawn onto a `Graphics2D` image component by applying the scale factor.

```

public BufferedImage getScreenShot(double scaleFactor) {
    Main.controller.applet.window.setVisible(false);

    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Rectangle rect = new Rectangle(0, 0, screenSize.width,
        screenSize.height);
    BufferedImage capture;

    try {
        capture = new Robot().createScreenCapture(rect);
    } catch (AWTException ex) {

```

```

        capture = new BufferedImage(screenSize.width, screenSize.height,
            BufferedImage.TYPE_INT_ARGB);
    }

    int width = (int) (capture.getWidth() * scaleFactor);
    int height = (int) (capture.getHeight() * scaleFactor);

    BufferedImage image = new BufferedImage(width, height,
        BufferedImage.TYPE_INT_ARGB);
    Graphics2D g2 = (Graphics2D) image.getGraphics();
    g2.setRenderingHint(RenderingHints.KEY_RENDERING,
        RenderingHints.VALUE_RENDER_QUALITY);
    g2.drawImage(capture, 0, 0, width, height, null);
    return image;
}

```

#### 4.2.2 Save Image

Active Lecture Client allows users to save the image to the server and either exit or continue working on the same image. Saving the image is performed using Apache Commons HttpClient and FileUpload utilities [23, 25]. A multi-part message is created with the file and other string parameters. This multi-part message is passed as a parameter to the HTTP request to upload the file. The timeout for each file upload request is set as 8 seconds in the Active Lecture Client. Here is the code snippet to save the image:

```

File f1 = File.createTempFile("capture", extension);
ImageIO.write(img, extension.substring(1), f1);

String name = null;
String url = null;
String lectureId = null;
name = getParameter("com.horstmann.activelecture.name");
url = getParameter("com.horstmann.activelecture.uploadURL");
lectureId = getParameter("com.horstmann.activelecture.lecture");
Part[] parts = {
    new StringPart("com.horstmann.activelecture.name",
        student ? name : "null"),
    new StringPart("com.horstmann.activelecture.student", "" + student),
    new StringPart("com.horstmann.activelecture.lecture", lectureId),
    new FilePart(name == null ? f1.getName() : name, f1)
};

HttpClient client = new HttpClient();
PostMethod filePost = new PostMethod(url);
client.getHttpClientConnectionManager().getParams().setConnectionTimeout(8000);
filePost.setRequestEntity(new MultipartRequestEntity(parts,
    filePost.getParams()));

int statusCode = client.executeMethod(filePost);
filePost.releaseConnection();

```

#### 4.2.3 Packaging and Signing the Client

Active Lecture Client is packaged into a JAR file `Client.jar` and made available to the Active Lecture Server application by copying it to the `resources` directory of the Server. Users will be

downloading the `Client.jar` from the web browser when they run JNLP file or as applet. `Client.jar` is signed with a Thawte Freemail Certificate [26, 27] so that the users will get one friendly warning message as shown in Figure 27. The user has to accept to download the file and run it.

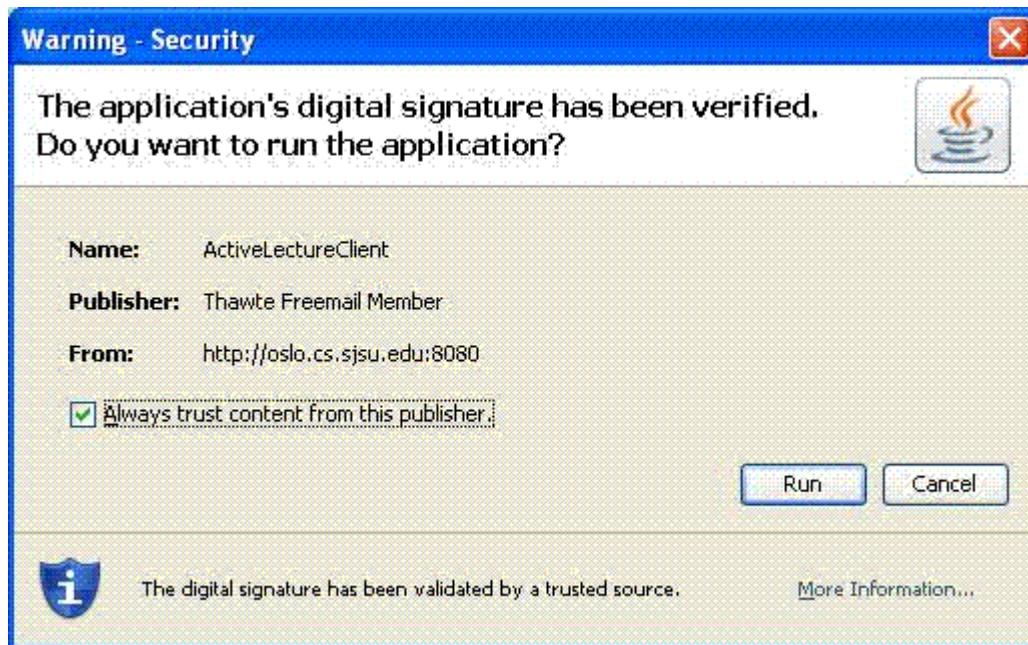


Figure 27 Security Warning before Downloading Active Lecture Client

The following snippet of the Ant build for the client application is used to package and sign the JAR. As each JAR that is downloaded asks for confirmation to install, to minimize the number of warning messages to one, all the other code from the dependent JAR files like `httpclient.jar`, `logging.jar`, and `codec.jar` are unzipped and packaged into the same `Client.jar` file. The manifest files that are copied while extracting these files are deleted to make sure that the MANIFEST directory contains only the current signature. The JAR file that is created is signed as shown in the `signjar` tag [27]. Here is the Ant target to create `Client.jar` and sign it with a key:

```
<target name="build-jar" depends="compile">
  <copy todir="${build.dir}">
    <fileset dir="${src.dir}">
      <include name="**/*.png"/>
      <include name="**/*.properties"/>
    </fileset>
  </copy>
  <unzip src="${commons.httpclient.jar}" dest="${build.dir}" />
  <unzip src="${commons.logging.jar}" dest="${build.dir}" />
  <unzip src="${commons.codec.jar}" dest="${build.dir}" />
  <delete dir="${manifest.dir}"/>
  <jar destfile="${dist.dir}/${jar.name}.jar" basedir="${build.dir}">
    <manifest>
      <attribute name="Main-Class"
        value="com.horstmann.activelecture.Main"/>
    </manifest>
  </jar>
  <signjar jar="${dist.dir}/${jar.name}.jar" key="${key}" />
</target>
```

```
    </manifest>
  </jar>
  <signjar jar="${dist.dir}/${jar.name}.jar" keystore="${keystore.file}"
    storepass="${keystore.password}" alias="${keystore.alias}"
    verbose="true" />
</target>
```

## 5. Implementation Challenges

It is challenging to design and develop a system that uses several technologies. The main challenges of this project were:

1. Development using Java, JavaEE, and AJAX
2. Providing complete solution using different technologies

### *5.1 Development using Java, JavaEE, and AJAX*

It was very challenging to learn the technologies in Java and JavaEE in a short period of time and develop the Active Lecture system. At times, it has been very challenging because some of the features are not available in the tool. For example, there is very limited support for the AJAX components in JavaEE technology. I had to do comprehensive research to understand that the HTML in DIV components can be replaced dynamically to refresh the content on the pages. Different approaches were tried in efforts to understand that the panelgroup component in JSF will generate DIV component in HTML, but it cannot be refreshed. Therefore, I needed to come out with a plan to create hidden attributes that refresh the content and load the DIV component at the client site using JavaScript.

### *5.2 Providing Complete Solution using Different Technologies*

Deploying a complete working the Active Lecture system was a very challenging task. Packaging and signing of the Active Lecture Client with Apache Commons was very challenging as it involved many technologies. It needed an understanding of the Java Web Start, Java Network Launching Protocol (JNLP) files, the certificate mechanism, ANT tool, extracting the contents of Apache httpclient.jar, logging.jar and codec.jar (Commons JARs), packaging the contents of all the jars into one common JAR and signing it. Signing the client failed initially because the manifest was referring to signatures from the extracted Apache Commons JARs. Deleting the MANIFEST directory from extracted common JARS fixed the signing issue. Providing a student view from the instructor interface in the same browser session was very difficult. As both instructor and student interfaces share objects, providing a working student interface in the same session turned out to be very challenging. In addition, I had to study and understand several technologies like Woodstock components [21], iText APIs, and other Java extensions to implement the complete solution.

## 6. Analysis and Results

The usage of the Active Lecture system was studied by analyzing the data collected from using it in a classroom setting. The following usages were studied for the Active Lecture system:

1. Comparison with Ubiquitous Presenter
2. Active Lecture System Student Response Analysis

### *6.1 Comparison with Ubiquitous Presenter*

We conducted a study to compare the Active Lecture system against Ubiquitous Presenter. The main objective of the study is to compare the usage of these systems. To study the usage, the following features are compared between the Active Lecture system and Ubiquitous Presenter:

1. Screen captures versus marked slides
2. Student responses

These two are good metrics for comparing the usage of these systems. Comparing screen captures in the Active Lecture system with marked slides in Ubiquitous Presenter gave us the metrics of how actively the instructor used the system by measuring the number of times the instructor has actively updated the data in the lecture while delivering the lecture in the class. Comparing the number of responses from the students allowed us to measure the usage of the Active Lecture system by the students. This usage analysis helped us to understand how actively students participated in the lectures by submitting the responses.

The study was conducted with one instructor teaching lectures one class with the Active Lecture system and the lectures of another class with Ubiquitous Presenter. Both the Active Lecture system and the Ubiquitous Presenter were used to lecture different courses in the same semester (Spring 2008). The Active Lecture system was used to deliver 18 lectures for CS 258, a graduate course. Ubiquitous Presenter was used to deliver 21 lectures for CS 40, an undergraduate course. There were 13 students in the CS258 class where the Active Lecture system was used, but 31 students in the CS40 class where Ubiquitous Presenter was used.

#### *6.1.1 Screen Captures versus Marked Slides*

The average number of screen captures in the Active Lecture system lectures was compared against the average number of marked slides in Ubiquitous Presenter lectures. In the Active Lecture system, screen captures are generally taken when an instructor is giving a lecture to the students. So, this indicates how actively instructor used the Active Lecture system. In Ubiquitous Presenter, the instructor prepares the lecture beforehand by creating the slides. In the class, the instructor presents the lecture slides and highlights or writes on the slide. These marked slides gave us the information about how actively the Ubiquitous Presenter is used by the instructor. The average number of screen captures in the Active Lecture system is 6.1 per lecture, but the average number of marked slides in Ubiquitous Presenter per lecture is 5.4. The standard deviation of the screen captures in the Active Lecture system is 3.96. The standard deviation of

marked slides in Ubiquitous Presenter is 4.1. The Active Lecture system has a noticeable advantage in this area. This may be because of the feature allowing the instructor to take screen shots of the desktop. The study would have given well-demonstrated results if the study were conducted with multiple instructors teaching multiple courses.

### *6.1.2 Student Responses*

The data from the lectures were analyzed to identify which system elicited more responses from the students when compared to the other system. This gives a good picture of how actively students participated by submitting the responses. The responses from students were analyzed in multiple ways. First, the average number of responses from the students per lecture in the Active Lecture system was compared with the average number of responses per lecture in Ubiquitous Presenter. The Active Lecture system gave better averages than Ubiquitous Presenter. For example, the average number of responses per student for a lecture when the Active Lecture system was used is 2.15, but the average number of responses per student for a lecture for Ubiquitous Presenter is 1.

When the total number of responses per lecture is compared, Ubiquitous Presenter has more responses per lecture because Ubiquitous Presenter has more students and more topics per lecture. In the Active Lecture system, if we improve the total number of topics per lecture, it will improve the total number of responses as well. For this study, the Active Lecture system was used while teaching a Graduate level Computer Science course (CS 258), while Ubiquitous Presenter was used while teaching an introductory course (CS 40) for non-Computer Science majors. Comparison of the systems while they were used in similar courses would have given a better analysis of student responses.

### *6.2 Active Lecture System Student Response Analysis*

The student responses for the Active Lecture system lectures were analyzed to understand the usefulness of different types of student responses in the Active Lecture system. This analysis uses the lectures from a CS 258 class where the Active Lecture system was used in the previous study. This analysis uses the student responses from the same 19 lectures. There were 13 students in that CS 258 class. The analysis of the Active Lecture system student responses contains two parts:

1. Image response analysis
2. Text and choice response analysis

#### *6.2.1 Image Response Analysis*

Students can submit the responses with or without an image in the Active Lecture system. Students can submit a screen shot of their desktop with any problem that they are facing when discussing a topic. Other students can share the screen shots if they had successes. This allows them to collaborate. As part of this analysis, the student responses with images were compared against responses without images for the same lectures in CS 258 class. When compared against



the total number of responses in the class, around 10 percent of the student responses were image responses. The most interesting fact identified in this analysis is that all the image responses are accompanied with text. This means, students are interested in providing a description about the image when they upload images.

### 6.2.2 Text and Choice Responses Analysis

The Active Lecture system allows students to submit responses with or without selecting a choice. Students can submit responses to a topic under discussion by selecting whether it is plain text or by selecting one of the choices in a choice list. The options in the choice list are “Choice 1”, “Choice 2”, “Choice 3”, “Choice 4” and “Choice 5” as shown in Figure 28.

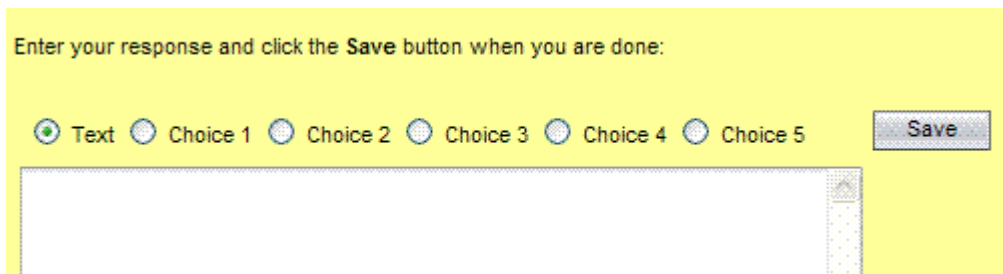


Figure 28 Student Response Choice List

To evaluate the usefulness of this feature, the student responses for the lectures from the same CS 258 class were studied to find out which type of responses were submitted frequently. The student responses without images were segregated into the responses with text only, the responses with choice selection only, and the responses with choice and text. For this analysis, the percentages for these different types of responses were calculated and compared.

When the student responses are compared, 71 percent of the student responses are found to be text only responses, while 29 percent of the responses have selected a choice. Students seem to like the text only option more than the different kinds of choice options available. When different choice options are compared, 64 percent of them are choice only responses and 36 percent are choice with text responses. This statistical analysis tells us that when multiple-choice questions are asked, more students are inclined not to provide reasoning for their choice in the form of text.

## 7. Conclusion

The Active Lecture system helps lecturing by providing new features that are useful to the users. The development of the Active Lecture system is innovative and challenging. It is also helpful to instructors and students as it brings out the interaction between them.

### 7.1 Key Achievements

1. The Active Lecture system has been developed with the following important features:
  - (a) Refresh list of topics automatically for students and instructors
  - (b) Allow multiple student responses from same computer
  - (c) Generate lecture notes in PDF
  - (d) Registration and authentication of registered instructors
  - (e) Managing Lectures
  - (f) Saving annotated images incrementally
  - (g) Packaging Active Lecture Client
2. The Active Lecture system has been analyzed by considering the following features:
  - (a) Comparison with Ubiquitous Presenter
  - (b) Usage of student responses

### 7.2 Future Enhancements

The Active Lecture system can be enhanced to be more useful and reliable to the instructors by adding the following improvements:

1. Upload presentations (PowerPoint, Open Office)
2. Improve response reporting
3. Upload a file, such as tutorials and problem solutions
4. Security

Adding upload presentations to the Active Lecture system will allow the instructors to add the lecture content easily. This will definitely improve the number of topics or slides the instructor uses per lecture, as instructors will be uploading the slides from presentations. The response reporting can be improved by providing “Summary of Responses” which will count the responses based on selected choice. Adding a feature to upload any type of file will allow instructors to upload content that is generated by other tools for reference. The lecture notes will allow instructors and students to download the uploaded reference files for future use. This feature will encourage the instructors to use the system more often, as this will remove the post lecture process of making the files available elsewhere. Analyzing the vulnerabilities of the system can enhance the security of the system. The security of the image upload can be studied and enhanced. In addition, hiding the extensions of the files in URL using URL rewriting can thwart some obvious cyber attacks. These enhancements will improve the usage and security of the Active Lecture system.

### *7.3 Further Analysis*

Re-analyzing the usage of the Active Lecture system in classroom setting after making the above-mentioned enhancements would help us to understand the improvements in regard to the usefulness of the system. In addition, comparing the registered instructors to the unregistered instructors over a period of normal usage of the system will allow us to understand whether the instructors are leaning towards using the anonymous option or not.

## References

- [1] Friedland, G., Knipping, L., Rojas, R., and Tapia, E. 2004. Teaching with an intelligent electronic chalkboard. In Proceedings of the 2004 ACM SIGMM Workshop on Effective Telepresence (New York, NY, USA, October 15 - 15, 2004). ETP '04. ACM, New York, NY, 16-23.
- [2] Jones, P. (1999). Improving learning in lectures using keypad-response units. The Proceedings of the 8th Annual Teaching Learning Forum. The University of Western Australia, 3-4 February 1999. 173-177.
- [3] Murphy, T. 2008. Success and failure of audience response systems in the classroom. In Proceedings of the 36th Annual ACM SIGUCCS Conference on User Services Conference (Portland, OR, USA, October 19 - 22, 2008). SIGUCCS '08. ACM, New York, NY, 33-38.
- [4] Simon, B., Anderson, R., Hoyer, C., and Su, J. 2004. Preliminary experiences with a tablet PC based system to support active learning in computer science courses. In Proceedings of the 9th Annual SIGCSE Conference on innovation and Technology in Computer Science Education (Leeds, United Kingdom, June 28 - 30, 2004). ITiCSE '04. ACM, New York, NY, 213-217.
- [5] French, J. H. 2007. Beyond the tablet PC: using the tablet PC in a collaborative learning environment. *J. Comput. Small Coll.* 23, 2 (Dec. 2007), 84-89.
- [6] Beavers, J., Chou, T., Hinrichs R., Moffatt C., Pahud M., and Eaton J. V. (2004). The learning experience project: Enabling collaborative learning with ConferenceXP. Microsoft Research Technical Report MSR-TR-2004-42, April 2004.
- [7] Abowd, G. 1999. Classroom 2000: An experiment with the instrumentation of a living educational environment. *IBM Systems Journal*, Volume 38, Number 4, 1999.
- [8] Simon, B., Anderson, R., and Wolfman, S. 2003. Activating computer architecture with classroom presenter. In Proceedings of the 2003 Workshop on Computer Architecture Education: Held in Conjunction with the 30th international Symposium on Computer Architecture (San Diego, California). WCAE '03. ACM, New York, NY, 11.
- [9] Koile, K., Chevalier, K., Low, C., Pal, S., Rogal, A., Singer, D., Sorensen, J., Tay, K. S., and Wu, K. 2007. Supporting Pen-Based Classroom Interaction: New Findings and Functionality for Classroom Learning Partner. In Proceedings of the First international Workshop on Pen-Based Learning Technologies (May 24 - 25, 2007). PLT. IEEE Computer Society, Washington, DC, 14.
- [10] Wilkerson, M., Griswold, W. G., and Simon, B. 2005. Ubiquitous presenter: increasing student access and control in a digital lecturing environment. In Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education (St. Louis, Missouri, USA, February 23 - 27, 2005). SIGCSE '05. ACM, New York, NY, 116-120.

- [11] Ratto, M., Shapiro, R.B., Truong, T.M. and Griswold, W.G. 2003. The activeclass project: Experiments in encouraging classroom participation. In Proceedings of CSCL 2003, Kluwer, pp. 477-486, June 2003.
- [12] Anderson, R., McDowell, L. and Simon, B. (2005) Use of classroom presenter in engineering courses. In Proceedings of ASEE/IEEE frontiers in education 2005, T2G-13-T2G-18.
- [13] Andersen. R. University of Washington. Retrieved May 19, 2009, from <http://classroompresenter.cs.washington.edu>
- [14] Anderson, R., Anderson, R., Simon, B., Wolfman, S. A., VanDeGrift, T., and Yasuhara, K. 2004. Experiences with a tablet PC based lecture presentation system in computer science courses. In Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education (Norfolk, Virginia, USA, March 03 - 07, 2004). SIGCSE '04. ACM, New York, NY, 56-60.
- [15] Anderson, R. J., Anderson, R., VanDeGriff, T., Wolfman, S. A., and Yasuhara, K. 2003. Classroom presentation from the tablet PC. In Proceedings of the 8th Annual Conference on innovation and Technology in Computer Science Education (Thessaloniki, Greece, June 30 - July 02, 2003). D. Finkel, Ed. ITiCSE '03. ACM, New York, NY, 238-238.
- [16] Griswold, W. G. and Simon, B. 2006. Ubiquitous presenter: fast, scalable active learning for the whole classroom. In Proceedings of the 11th Annual SIGCSE Conference on innovation and Technology in Computer Science Education (Bologna, Italy, June 26 - 28, 2006). ITiCSE '06. ACM, New York, NY, 358-358.
- [17] Ubiquitous Presenter, UC San Diego, Retrieved May 19, 2009, from <http://up.ucsd.edu/about/WhatIsUP.html>
- [18] Denning, T., Griswold, W. G., Simon, B., and Wilkerson, M. 2006. Multimodal communication in the classroom: what does it mean for us?. In Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education (Houston, Texas, USA, March 03 - 05, 2006). SIGCSE '06. ACM, New York, NY, 219-223.
- [19] Lindquist, D., Denning, T., Kelly, M., Malani, R., Griswold, W. G., and Simon, B. 2007. Exploring the potential of mobile phones for active learning in the classroom. SIGCSE Bull. 39, 1 (Mar. 2007), 384-388.
- [20] Wang, S, Hsu, H, 2008. Use of the Webinar Tool (Elluminate) to Support Training: The Effects of Webinar-Learning Implementation from Student-Trainers' Perspective. Journal of Interactive Online Learning. Vol. 7, No. 3, pp. 175-194, Winter 2008
- [21] Project Woodstock. Retrieved May 19, 2009, from <https://woodstock.dev.java.net/index.html>

- [22] Project Woodstock 4.3 Tag Library. Retrieved May 19, 2009, from <http://webdev2.sun.com/woodstock-tlddocs/webuijsf/progressBar.html>
- [23] Apache Commons FileUpload 1.2.1. Retrieved May 19, 2009, from <http://commons.apache.org/fileupload/>
- [24] iText Homepage. Retrieved May 19, 2009, from <http://www.lowagie.com/iText/>
- [25] Jakarta Commons HttpClient. Retrieved May 19, 2009, from <http://hc.apache.org/httpclient-3.x/index.html>
- [26] Thawte. Personal E-mail Certificates. Retrieved May 19, 2009, from <http://www.thawte.com/secure-email/personal-email-certificates>
- [27] Dallaway. R. Java Web Start and Code Signing. Retrieved May 19, 2009, from <http://www.dallaway.com/acad/webstart/>