2010

The ActiveLecture System

Sanuja Dabade
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the [Computer Sciences Commons](#)

Recommended Citation

Dabade, Sanuja, "The ActiveLecture System" (2010). *Master's Projects*. 60.

DOI: <https://doi.org/10.31979/etd.rxgz-yz5c>

https://scholarworks.sjsu.edu/etd_projects/60

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.

The ActiveLecture System

A Project Report

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirement for the Degree

Master of Computer Science

By

Sanuja Dabade

May 2010

© 2010

Sanuja Dabade

ALL RIGHTS RESERVED

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Cay Horstmann

Dr. Robert Chun

Dr. Johnny Martin

APPROVED FOR THE UNIVERSITY

ABSTRACT

ACTIVE LECTURING SYSTEM

By Sanuja Dabade

For an instructor, it has always been a challenging task to keep students engaged during the lecture and assess them in real time. Instructors use different methodologies to address the challenge of keeping students engaged during the class and increase their participation. Widely used methodologies for active learning include clickers and Tablet PCs. Clickers are small handheld devices which are used to collect student responses and present these responses graphically. Tablet PCs provide a functionality to use natural handwriting to provide feedback. Web browsers are a cost effective approaches. I have used a web based technology to create active learning system called ActiveLecture. It provides a mechanism for students and an instructor to communicate effectively during the lecture. I converted the ActiveLecture system from Java EE 5.0 to Java EE 6.0 platform. I added new functionalities to the ActiveLecture system and gathered statistics and video captures to evaluate them. Finally, I performed an analysis based on the collected data that can help instructor assess the effectiveness of the ActiveLecture system.

ACKNOWLEDGEMENTS

I would like to thank Dr. Cay Horstmann for giving me an opportunity to work with him for my master's project. I am thankful for his valuable suggestions, tireless energy to review my work and helping me in all possible ways to complete this project.

I would also like to thank Dr. Robert Chun and Dr. Johnny Martin for their critical review of the report and for accepting my committee member request.

On a personal note, I would like to thank my parents and my brother for their continuous support and encouragement for my higher education studies.

Table of Contents

Introduction	1
1 Related Work	2
1.1 Clickers	2
1.2 A Java-based Classroom Response System for Teaching Java	2
1.3 Network Response Analyzer System	2
1.4 Classroom Presenter	3
1.5 Ubiquitous Presenter	4
1.6 Active Learning Using Mobile Phones	5
1.7 MessageGrid	5
1.8 Success and Failure of Audience Response System in the Classroom	5
2 Java EE	6
2.1 JSF	6
2.1.1 Composite Components	6
2.1.2 Ajax Support	8
2.1.3 Annotations	9
2.2 JPA	10
2.3 The Flow of Managed Beans -> Session Beans -> Persistent Data	12
3 ActiveLecture System	14
3.1 Structure of the ActiveLecture Project	14
3.2 Existing Features	14
3.2.1 Support Unregistered and Registered Instructors	15
3.2.2 Discuss Current Topic	15
3.2.3 Student Interface to Discuss Current Topic	16
4 New features added in the ActiveLecture system	17
4.1 Count Number of Student Responses Using Ajax	17
4.2 Identified and Anonymous Student Responses	18
4.3 Show Student Responses in a New Browser Window	20
4.4 A Feature to Add a Visibility to Each Topic	22
4.5 Intellisense to Display the Number of Choices	22
4.6 Student Login	24
4.7 Removal of Woodstock Components from the ActiveLecture Project	26
4.8 Code Statistics of the ActiveLecture Project	26
5 Evaluation	27
5.1 Quantitative Data Analysis	27

5.2	Qualitative Data Analysis	28
5.3	Analyzing Video Records.....	30
5.4	Analysis of Statistics Gathered Using a Video Recording	34
6	Conclusion	39
	References	40

Introduction

Active learning techniques are widely used by instructors to engage students in the class and get a feedback from students in real time. These techniques help an instructor to know the performance of students in the class, their ability of understanding topics and so on. Some of these techniques require special hardware. For example clickers are used to collect student responses on multiple choice questions in real time. Tablet PCs are also used to collect student responses. Web browsers are a cost effective alternative. The ActiveLecture project uses a web browser technique. The ActiveLecture system provides interaction between students and an instructor. The goal of this project was to provide additional functions in the ActiveLecture project and make an analysis by gathering statistics about student's responses which will help an instructor to use the tool effectively.

Chapter 1

1 Related Work

Different techniques have been developed to increase student participation in the class and allow instructors to get feedback from students in real time. The following section explains the available techniques and methodologies used for active learning purpose.

1.1 Clickers

Personal response systems (clickers) are one useful technology to address the challenge of engaging students in class and to determine the extent of their learning [4]. Clickers are small hand held devices. Using clickers, students can respond anonymously to multiple choice questions during the lecture. These responses are automatically verified against correct answer and are shown graphically to the class. Clickers can be readily used to do the following: i) pre-assess students' learning, ii) get students to confront common misconceptions, iii) increase students' retention in the class iv) test students' understanding of course content, v) facilitate discussion, and vi) take class attendance [4].

1.2 A Java-based Classroom Response System for Teaching Java

Matthias Hauswirth and Andrea Adamoli have developed a Java based classroom response system. Instead of using ordinary hardware clickers, they used software clickers, implemented in Java, that allow for much richer problem types than the traditional multiple-choice question [5]. Informa [6] is an extensible framework for group response systems. Informa does not require special hardware, but it requires Java-enabled computers for the instructor and all students [5]. But it can also be run from any device that supports Java. The Informa application needs to be installed on both student and an instructor machine but the ActiveLecture system does not require any application installation.

1.3 Network Response Analyzer System

The network based response analyzer system collects students' responses in real time during lecture in a large classroom or distance learning situations [2]. This system is implemented using Active Server Pages and a Windows 2003 server. Student PCs, instructor PC and the response server are connected using client-server architecture. Students are provided choices to answer a particular question. All student responses are stored in the database along with student's identity.

Figure 1.1 shows a system configuration of response analyzer system.

Figure 1.1: System Configuration [2]

1.4 Classroom Presenter

Classroom Presenter [1] is a tablet PC based system. Classroom Presenter is developed using multicast networking. It adds natural hand writing capability to the slide presentation. It focuses on increasing interaction between students and an instructor. Figure 1.2 shows an instructor view of classroom presenter.

Figure1.2: Classroom Presenter [1]

Figure 1.3 shows a student interface of Classroom Presenter. In order to participate in the classroom discussion students require a tablet PC. The student interface shows a current slide selected by an instructor. Students can switch between text, pen, eraser and highlighter. Students can submit a current slide with attached contents to an instructor.

Figure 1.3: Student interface of Classroom Presenter

1.5 Ubiquitous Presenter

Ubiquitous Presenter (UP) expands Presenter via common web technologies to support non-tablet audiences and enhance student control [3]. It overcomes limitations of Classroom Presenter. UP is developed using client-server architecture. UP provides similar functionalities to CP. However students do not require Tablet PC in UP. Students can submit answers using web browsers. Figure 1.4 shows the setup of Ubiquitous Presenter System.

Figure 1.4 Setup of Ubiquitous Presenter system

1.6 Active Learning Using Mobile Phones

Cell phones are also used for active learning purpose since they are widely available. Lindquist et al. developed a mobile based response system by expanding Ubiquitous Presenter. It allows students to submit solutions to active learning exercises in the form of text or photo messages. In an exploratory study, students found that text messaging worked well for exercises with multiple choice or short answers [7].

1.7 MessageGrid

Roy P. Pargas and Dhaval M. Shah presented an approach to teaching an algorithms and data structures course (CS4) using MessageGrid [12, 13]. MessageGrid is a web-based software system that allows the instructor and students to interact electronically during (as well as outside of) class [11]. Their conclusion of experiment is that feedback about where students are having difficulty enables the instructor to adjust lesson plans for subsequent lectures [11]. And because the feedback is obtained long before a major test, there is opportunity to remediate [11].

1.8 Success and Failure of Audience Response System in the Classroom

Trevor Murphy used clickers to study the success and failure of audience response system in the classroom. He listed benefits included student anonymity, active student engagement, and instant feedback for students on how well or poorly they understand the material [9]. Inhibiting factors included comfort with PowerPoint and the clicker software, distributing and collecting clickers, and the potential time commitments of using the technology [9].

Chapter 2

2 Java EE

The ActiveLecture project is developed using Java EE technology. Java EE is an industry standard for designing, developing, assembling and deploying component-based enterprise applications. It separates business logic from presentation logic. This gives an ability to represent the same data in different formats without changing the business logic. It supports reusability, enhanceability, scalability, interoperability of components. Following section explains the concepts of Java EE used in the ActiveLecture system.

2.1 JSF

Java Server Faces is a server side technology. It is used in web applications to create User Interfaces. It is a part of Java EE standard. JSF responds to client events but code for UI runs on the server. It follows the Model-View-Controller design pattern.

I have used following functionalities of Java Server Faces while implementing the ActiveLecture system.

2.1.1 Composite Components

A reusable custom UI component called as composite component can be easily created in JSF 2.0. Definition and implementation of composite components is defined in XML file. This file includes elements from <http://java.sun.com/jsf/composite> namespace and xhtml contents.

Following code snippet shows the contents of the polltag.xhtml file to create a composite component.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:cc="http://java.sun.com/jsf/composite">
<body>
    <cc:interface>
        <cc:attribute name="interval" required="true"
            type="java.lang.Integer"/>
        <cc:attribute name="timeout" required="false" type="java.lang.Integer"/>
        <cc:attribute name="execute" required="false" type="java.lang.String"/>
        <cc:attribute name="render" required="false" type="java.lang.String"/>
    </cc:interface>
    <cc:implementation>
```

```

<span id="#{cc.clientId}">
  <h:outputScript name="poll/polltag.js" target="head" />
  <h:commandButton id="hidden" type="button" style="display: none;">
    <f:ajax execute="#{cc.attrs.execute}" render="#{cc.attrs.render}"/>
  </h:commandButton>
</span>
<script type="text/javascript">
  jsf.ajax.addOnError(ActiveLecture.polltag.init("#{cc.clientId}", "#{cc.attrs.interval}", "#{cc.attrs.timeout}"));
</script>
</cc:implementation>
</body>
</html>

```

The `<cc:interface>` tag specifies what can be included inside a component. It includes the information about attributes, their data types and rendering information of these attributes. The `<cc:implementation>` tag includes the code which will be replaced when this component is referred in the page. The above code snippet refers to a javascript function `init` which is defined in a file `polltag.js`. This file is included under the same folder as the JSF component.

The `polltag.xhtml` file is kept in the resources folder. The file name is used a tag name while referring to the composite component. The location within the resources directory specifies the tag namespace. So if the above component is included inside the `...resources/poll` folder with the name `polltag.xhtml`, it can be referred on the facelet page in the following manner.

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:f="http://java.sun.com/jsf/core"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  xmlns:ezc="http://java.sun.com/jsf/composite/poll">
<h:head title="http://ActiveLecture.org">
</h:head>
<body onLoad="formLoad();" >
  <h:form id="form1">
    <h:outputText id="DisplayCount" value="Number of Responded Students:
      #{mainRB.numberofRespondedStudents}"/>
    <ezc:polltag id="poll" interval="200" timeout="20000"
      render=":form1:DisplayCount"/>
  </h:form>
</body>
</html>

```

The `xmlns:ezc=http://java.sun.com/jsf/composite/poll` library is included in the beginning to access a composite component. At run time, the line `<ezc:polltag id="poll" interval="200" timeout="20000" render=":form1:DisplayCount"/>` is replaced by the code in the composite component.

2.1.2 Ajax Support

JSF 2.0 includes an API for JavaScript. `jsf.ajax.request()` sends an Ajax request to a current view. This method formulates request payload which includes the collected data, registers callback and posts the request back into Faces lifecycle.

`PartialViewContext` object provides the state of the Ajax request. Information about targeted components for rendering/processing is collected through this object. The `processPartial()` method performs a partial subtree rendering/processing using this information. The callback updates the DOM tree on the client side automatically.

The following example explains a simple counter using Ajax.

```
<h:outputScript name="jsf.js" library="javax.faces" target="head"/>
<h:outputText id="DisplayCount" value="#{count.count}"/>
<h:commandButton id="mybutton" value="Count onclick="jsf.ajax.request(this,
    event, {execute: this.id, render: ' DisplayCount '}); return false;"/>

@ManagedBean(name = "count")
@SessionScoped
public class Count {
    Integer count = 0;
    public Integer getCount() {
        return count++;
    }
}
```

The `<h:outputScript>` element specifies to include the Ajax library into the page. Without including the JSF Ajax library into the page, the `jsf.ajax.request()` call cannot be made. The `onclick` method of a `mybutton` calls `jsf.ajax.request` and also returns `false`, specifying that submit is not called for the entire form. Three parameters are passed to the `jsf.ajax.request` object: calling object, calling event and a parameter with properties `execute` and `render`. The `execute` property takes parameter ids of all JSF components which will be executed and the `render` property tells which JSF components need to be updated.

The `jsf.ajax.request()` JavaScript API is a primitive that is primarily targeted for use by frameworks as well as by the JSF implementation itself. JSF 2.0 also includes a declarative approach that is intended to be more convenient for page authors. This approach leverages the `<f:ajax>` tag.

The following figure shows a screenshot of instructor interface showing the count of responded students at a given interval of time.

Figure 2.1: Incrementing student response count dynamically using Ajax

2.1.3 Annotations

The following section explains JSF 2.0 annotations used in ActiveLecture project.

As shown in the following code snippet, the `@ManagedBean` annotation is added above the bean class. The name for the bean is specified using attribute name. If the name is not specified, the class name is used as a bean name with the first letter in lower case. This managed bean is referred as `#{loginRequestBean.instructorEmail}` where `instructorEmail` is a name of the method or shortcut for getter/setter method.

```
@ManagedBean(name = "loginRB")
@RequestScoped
public class LoginRequestBean {
    @Resource(name = "mail/ActiveLectureMail")
    private Session mailSession;
    private String instructorEmail = "";

    @ManagedProperty(name = "mainSessionBean", value = "#{mainSB}")
    Private MainSessionBean mainSB;

    public void setMainSessionBean(MainSessionBean mainSB) {
        this.mainSB = mainSB;
    }

    Public void setInstructorEmail(String instructorEmail) {
        this.instructorEmail = instructorEmail;
    }
}
```

The following line of code shows how to call a method from a managed bean on a JSF page. `<h:inputText value="#{loginRB.instructorEmail}"/>`
 It calls a method `setInstructorEmail` from `LoginrequestBean` when a page is submitted.

As shown in the above code snippet, the scope of the bean is defined by using annotations in JSF 2.0 like `@RequestScoped`, `@SessionScoped`, `@ApplicationScoped`, `@ViewScoped`, `@CustomScoped` and `@NoneScoped`.

2.2 JPA

Relational data is managed using Java Persistence API. It provides an object/relational mapping. An entity is a persistent data object and typically corresponds to a table in the relational database. As shown the following code snippet, `Student` is the name of a table in the database. A field marked with `@Id` annotation corresponds to the primary key of a table in the relational database. `@GeneratedValue` annotation indicates that a value is generated in the database and an attribute `strategy` says that a value is generated automatically. All getter/setter methods are used to insert/read values in the columns of a `Student` table.

```
import java.io.Serializable;
import java.util.HashSet;
import java.util.Set;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
@Entity
public class Student implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String userName;
    private String password;
    private String firstName;
    private String lastName;
    private String email;

    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    @Override
    public String toString() {
        return "com.horstmann.activelecture.entity.Student[id=" + id + "]";
    }
}
```

```

public String getUsername() {
    return userName;
}
public void setUsername(String userName) {
    this.userName = userName;
}
public String getFirstName() {
    return firstName;
}
public void setFirstName(String firstName) {
    this.firstName = firstName;
}
public String getLastName() {
    return lastName;
}
public void setLastName(String lastName) {
    this.lastName = lastName;
}
public String getEmail() {
    return email;
}
public void setEmail(String email) {
    this.email = email;
}
public String getPassword() {
    return password;
}
public void setPassword(String password) {
    this.password = password;
}
}

```

The `EntityManager` API creates and removes persistent entity instances, finds entities by the entity's primary key, and allows queries to be run on entities [5]. The `findStudent` method finds a student from the student table using a provided name. Using an instance of an entity manager, a query is created. This query is fired on Student table to get the students with the matching name. Similarly, the `create` method saves the student information in the Student table using the `persist` method.

```

@Stateless
public class StudentsImpl implements Students{
    @PersistenceContext
    private EntityManager em;

    public void create(Student student) {
        em.persist(student);
    }
    public void edit(Student student) {
        em.merge(student);
    }
    public void remove(Student student) {
        em.remove(em.merge(student));
    }
}

```

```

    }
    public Student findStudent(long studentId){
        return em.find(Student.class, studentId);
    }
    public Student findStudent(String studentUserName){
        if(studentUserName==null || studentUserName.trim().length() == 0)
        {
            return null;
        }
        try {
            Query q = em.createQuery(
                "SELECT x FROM Student x WHERE x.userName = :un").setParameter("un",
                studentUserName);
            List<Student> students = (List<Student>) q.getResultList();
            Student student = null;

            if (!students.isEmpty()){
                student = students.get(0);
            }
            return student;
        }
        catch (RuntimeException ex) {
            Logger.getLogger(getClass().getPackage().getName()).log(Level.SEVERE, "",
                ex);
            throw ex;
        }
    }
}

```

2.3 The Flow of Managed Beans -> Session Beans -> Persistent Data

The following code snippet shows a property present in the managed bean. In order to get a value of the anonymous column from the database, the managed bean first calls a method using an instance of the lectures session bean to get a current topic id. This method returns a row from the topics table. Finally, the `getAnonymous` property returns a value of the anonymous column from the persistent object.

```

private boolean anonymous;
public boolean getAnonymous()
{
    Topic t = lectures.currentTopicId(lectureId);
    this.anonymous = t.getAnonymous();
    return this.anonymous;
}

public void setAnonymous(boolean anonymous)
{
    Topic t = lectures.currentTopicId(lectureId);
    lectures.updateTopicInfo(lectureId, t.getId(), anonymous);
    this.anonymous = anonymous;
}

```

`currentTopicId` method is present in lectures bean.

```

public Topic currentTopicId(long lectureId){
    if (lectureId == 0) {
        return null;
    }
    try {
        Query q = em.createQuery("SELECT y FROM Lecture x JOIN x.topics y WHERE
            x.id = :id AND NOT
            ((y.text IS NULL OR y.text = '') AND y.imageId IS NULL) ORDER BY
            y.creationTime").setParameter("id", lectureId);
        List<Topic> results = (List<Topic>) q.getResultList();
        for (Topic topic : results) {
            if (topic.getOpened())
            {
                return topic;
            }
        }
        return null;
    }
    catch (RuntimeException ex) {
        Logger.getLogger(getClass().getPackage().getName()).log(Level.SEVERE, "",
            ex);
        throw ex;
    }
}

```

Chapter 3

3 ActiveLecture System

The ActiveLecture system was initially developed by Dr. Horstmann and later enhanced by Mr. Himavantha. It was developed using Java Enterprise Edition 5.0

3.1 Structure of the ActiveLecture Project

Figure 3.1 shows the object diagram of entity beans in the ActiveLecture system. Each entity corresponds to a database table in the relational database. Relationships can be defined for entity beans. As shown in Figure 3.1 Instructor and Lecture entities have one to many relationship since an instructor can have multiple lectures and a lecture belongs to an instructor. Similarly Lecture and Topic entities have one to many relationship. Topic and Response entities are derived from the ItemBase entity. Entity beans are persistent in nature. Session beans implement the business logic and call entity beans to perform database related operations. Session beans can be stateless or stateful. But session beans are not persistent. The ActiveLecture has two session beans. LectureImpl which implements Lectures, and ResponseData which implements PageData. Clients call methods present in session beans which in turn call methods on an entity manager to perform database operations.

Figure 3.1 Object diagram of ActiveLecture system

3.2 Existing Features

The following sections explain functionalities and limitations present in the ActiveLecture system before I worked on it.

3.2.1 Support Unregistered and Registered Instructors

Using this feature, unregistered instructors can start a new lecture. However, unregistered instructors need to remember a lecture id to retrieve an existing lecture. Registered instructors can retrieve previous lectures by signing in to the ActiveLecture system. After successful login, a list of previous lectures is shown to an instructor. Figure 3.2 shows the instructor interface for unregistered instructors. Using the New Lecture button, an instructor can start a new lecture. In order to retrieve a previous lecture, an instructor needs to provide a valid lecture id and password if present. Using the Retrieve Lecture button, an instructor can open a previous lecture.

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 3.2: Instructor interface for unregistered instructors

Previously, the student interface did not provide a registration facility. Thus student's responses were always anonymous. This system was unable to provide a student's identity to an instructor.

3.2.2 Discuss Current Topic

An instructor could discuss a current topic by selecting a topic from the list of topics and clicking on the Discuss Current Topic button. Once an instructor hits the Discuss Current Topic button, all the topics present in the list of topics were shown on the student interface. The currently discussed topic was shown with a different background color on instructor and student interfaces to avoid confusion in the student view that was available to an instructor.

<input type="button" value="New Topic"/>	<input type="button" value="Discuss Current Topic"/>	<input type="button" value="Download as PDF"/>	<input type="button" value="Close Lecture"/>	Hide Help
--	--	--	--	---------------------------

Figure 3.3: Discuss current topic

There were two limitations of this approach. First, it was unable to provide a facility to an instructor to show only selected topics from the list of topics to students. Second, Discuss Current Topic always discusses a topic anonymously. Thus student responses were always anonymous.

3.2.3 Student Interface to Discuss Current Topic

Students could discuss a current topic by selecting a topic under discussion from the list of topics section. Choices were shown only when students click on the Discuss button.

Students needed to manually click on the Discuss button in order to view the choices. Also, students were always provided with 5 choices and a textbox to submit an answer as shown in figure 3.4. This was misleading to students. Even if an instructor does not provide choices to a question, they are always shown on student interface. This can confuse students while submitting an answer. Only the provided number of choices should be displayed to students.

The screenshot shows a yellow rectangular interface. At the top, it says "Enter your response and click the Save button when you are done:". Below this, there is a row of radio buttons. The first is labeled "Text" and is selected. The others are labeled "Choice 1", "Choice 2", "Choice 3", "Choice 4", and "Choice 5". To the right of these radio buttons is a "Save" button. Below the radio buttons is a large white text input area with a vertical scrollbar on the right side.

Figure 3.4: Show choices on student interface

Apart from these features other functionalities implemented in the ActiveLecture system which include downloading lecture notes, launching of a capture tool, and providing configurable help.

Chapter 4

4 New features added in the ActiveLecture system

The following sections explain functionalities that I implemented in the ActiveLecture system.

4.1 Count Number of Student Responses Using Ajax

This feature provides an instructor an exact count of responding students at a given interval of time. It is implemented using Ajax. Ajax is an acronym for Asynchronous JavaScript and XML. Usually in web applications, an entire page is submitted to the server when a button is clicked in the browser. After receiving a response from the server, entire page is reloaded. This process is time consuming. Ajax solves this problem. It only sends a part of the page to the web server which needs to be updated. This does not refresh the entire page unnecessarily. When a response is received from the server, the current page is updated by manipulating the DOM (Domain Object Model).

When an instructor adds a topic to the lecture and hits the Discuss Topic Anonymously or Discuss Topic as Identified button, a topic is made available to students. Students respond to a topic by selecting one of the available choices or by typing text in the textbox area. The response is submitted when a student clicks on the Save button. This response is saved in the database.

The instructor interface shows the count of responding students at a given interval time as shown in the Figure 4.1. This count is initialized to zero before a question is made available to students. The count is increased dynamically as students start responding to questions.

The feedback is essential for the instructor to know when to close the discussion.

http://ActiveLecture.org| Lecture ID = 1154

New Topic All Download as PDF Close Lecture [Student View](#) [Hide](#) [Help](#)

Stop [Peek](#)

Number of Responded Students: 1

What is 2 times 2?

Edit [Launch Capture Tool](#)

List of Topics

- * [What is 2 times 2?](#)

Help

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

- [Start a lecture](#)
- [Start a topic](#)
- [Open topic for discussion](#)

Figure 4.1: Number of responded students

4.2 Identified and Anonymous Student Responses

This feature provides a choice to an instructor to discuss the current topic as anonymously or as identified. If a topic is discussed anonymously by an instructor, the student's identity is not revealed to an instructor. On the other hand, if a topic is discussed as identified, the identity of a student is revealed to an instructor along with the student's response. Logged in students are not asked to fill a separate form to provide their names. All non logged in students are asked to provide their identity by filling a form as shown in the Figure 4.2

<http://ActiveLecture.org> | lecture ID 1359

Student Information

First Name(required)

Last Name(required)

Submit

Provide both Firstname and Lastname

Figure 4.2 Student information

Some students do not prefer to participate in the discussion if responses are identified. On the other hand, identified student responses give information to an instructor about a student's performance in the class. In section 4.2, I analyze the difference in students' behavior when responses are identified against when responses are anonymous.

A topic can be discussed anonymously by clicking on the Discuss Current Topic (anonymous) button or it can be discussed as identified by clicking on Discuss Current Topic (identified) button as shown in the figure 4.3. Before discussing a topic, an instructor can add new topics by clicking on the New Topic button. A list of all topics in the lecture is shown on the right hand side. Each topic has a corresponding checkbox which is either checked or unchecked. This checkbox specifies whether a topic is hidden or visible to students. When a new topic is added, by default it is made invisible to students. When an instructor clicks on one of the discuss buttons, visibility of the currently selected topic is automatically changed to visible and a checkbox for the corresponding topic gets checked automatically.

Figure 4.3: Discuss current topic as identified or as anonymous

4.3 Show Student Responses in a New Browser Window

This feature shows students response in a new browser window at a given point of time. As shown in the figure 4.4, when an instructor clicks on the Peek button, a new browser window is opened showing student's responses at a given point of time. If a topic has been discussed as identified, responses are shown along with student's identity. If an answer is submitted using a textbox, a suffix t is added to the answer as shown in the figure 4.4.

Figure 4.4: Identified student responses

Similarly, all the anonymous student responses are shown if a topic is discussed anonymously by an instructor. Figure 4.5 shows anonymous students response. 2 * 1 indicates that two students answered with choice one.

Figure 4.5: Anonymous student responses

4.4 A Feature to Add a Visibility to Each Topic

According to the old design, whenever a new topic is created by an instructor, it is immediately made visible on the student interface. So an instructor cannot add the topic before the lecture begins. Otherwise students can discuss a topic even before lecture begins in real time.

The design is modified to give a more flexibility on the visibility of each topic to an instructor. A new column is added in the database table to keep the track of visibility of each topic. In the instructor interface, visibility of a topic is displayed using a checkbox. While adding a new lecture, an instructor can set the visibility of a topic using Hide Topic button as shown in the figure 4.3. Checked indicates visible topics and unchecked indicates hidden topics. When an instructor clicks on the Save button, a topic is added in the topic list shown on the right hand side as shown in the figure 4.6. Each topic has its corresponding checkbox to display the visibility. If an instructor wants to change the visibility of a topic manually, he/she can click on the edit button and check/uncheck the Hide Topic checkbox.

Figure 4.6: Showing list of visible and hidden topics on instructor interface

4.5 Intellisense to Display the Number of Choices

According to the previous design, irrespective of the number of choices provided by an instructor, student interface is showing six choices when a topic is discussed. This

can mislead both an instructor and students. I added intellisense to an instructor interface to capture the number of choices provided for a topic.

When an instructor adds a new topic, a special method in the backend calculates the number of choices provided by an instructor for a particular topic. When a topic is opened for a discussion, it checks this value and shows choices accordingly on a student interface.

Figure 4.7 shows a student's interface. It shows the number of choices provided by an instructor. Since the choices are provided by an instructor, only textbox is shown on student interface.

sanuja

The screenshot shows a web-based student interface. At the top left, there is a small thumbnail image of a classroom. Below it, a blue navigation bar contains buttons: 'All', 'Download as PDF', 'Logout', and 'Refresh'. To the right of the navigation bar, the text '1154' is visible. The main content area has a yellow background. At the top, the question 'What is 2 times 2?' is displayed. Below the question, it says 'Enter your response and click the **Save** button when you are done:'. There is a radio button labeled 'Text' which is selected. Below this is a large white text input box. At the bottom left of the input box, there are two buttons: 'Save' and 'Add Image'. On the right side of the yellow area, there is a section titled 'List of Topics' which contains a single entry: '* What is 2 times 2?'.

Figure 4.7: Student interface showing no choices are provided

Similarly if an instructor provides four choices to a topic, student's interface displays four choices and a textbox as shown in the figure 4.8.

The screenshot displays the ActiveLecture student interface. At the top left, there is a small image of a classroom. Below it, a blue navigation bar contains buttons for 'All', 'Download as PDF', 'Logout', and 'Refresh'. The main content area has a yellow background. On the left, a quiz question asks 'What is 2 time 2?' with four multiple-choice options: '1.4', '2.6', '3.8', and '4.None of the above'. Below the question, a prompt says 'Enter your response and click the Save button when you are done:'. There are five radio buttons labeled 'Choice 1', 'Choice 2', 'Choice 3', 'Choice 4', and 'Text'. At the bottom left are 'Save' and 'Add Image' buttons. On the right side, a 'List of Topics' section shows a topic 'What is 2 time 2?' with a star icon and a list of numbers '2? 1.4 2.6 3'.

What is 2 time 2?

1.4

2.6

3.8

4.None of the above

Enter your response and click the Save button when you are done:

☐ Choice 1 ☐ Choice 2 ☐ Choice 3 ☐ Choice 4 ☐ Text

Save

Add Image

List of Topics

* What is 2 time 2? 1.4 2.6 3

Figure 4.8: Student interface showing number of choices provided

4.6 Student Login

Previously, student login was not supported in the ActiveLecture system. According to the new design, students are also provided with login option. In order to get the username and password, student needs to register himself first. Figure 4.8 shows student registration form.

<http://ActiveLecture.org> | lecture ID 1154

Register Student

User Name	<input type="text" value="Sanuja"/>
password	<input type="password" value="..."/>
First Name	<input type="text" value="Sanuja"/>
Last Name	<input type="text" value="Dabade"/>
email (optional)	<input type="text" value="sdabade@gmail.com"/>
<input type="button" value="Submit"/>	

Figure 4.8: Student registration

Once registered successfully, students get their own username and password. Using this information, students can sign in before joining a lecture. Signed in students do not require to provide their first name and last name when a topic is discussed as identified. While submitting an answer, their login information is used to capture a first name and a last name. Figure 4.9 shows a student's interface for registered and unregistered students.

Figure 4.9: Student interface

4.7 Removal of Woodstock Components from the ActiveLecture Project

The Woodstock library is a JSF 1.x compatible component library with many advanced components. It was used in the prior version of ActiveLecture. The Woodstock library is not supported in Java EE 6.0. I have replaced all Woodstock components with regular or custom JSF components while migrating the ActiveLecture from Java EE 5.0 to Java EE 6.0

4.8 Code Statistics of the ActiveLecture Project

Table 4.1 shows statistic about the total number of classes, total number of JSF pages, and total LOC (Java + JSF) of the project before I started and after I finished.

	Before I started	After I finished
Total Number of Classes	19	23
Total Number of JSF Pages	15	21
LOC (Java +JSF) of project	4186	5762

Table 4.1 Code Statistics of ActiveLecture project

Chapter 5

5 Evaluation

In order to perform an evaluation of the ActiveLecture system, I have collected statistics about student's responses. The ActiveLecture application has been used by Dr. Horstmann in the Spring 2010 CS 46A class. I have analyzed the data collected during nine lectures. In each class he discussed two topics to collect student responses. Data is collected about anonymous and identified student responses.

5.1 Quantitative Data Analysis

Table 1 gives quantitative information about this data.

Topic id	Type	Count of responded students	Count of students with correct answer	Choices provided	Number of students present in the class
310	Identified	27	8	Yes	27
311	Identified	27	10	No	27
456	Identified	24	5	Yes	24
457	Identified	20	13	No	24
543	Identified	21	5	No	21
544	Identified	20	3	No	21
662	Identified	23	10	Yes	26
808	Anonymous	24	13	Yes	25
809	Anonymous	24	N/A	Yes	25
907	Anonymous	19	10	Yes	19
1061	Anonymous	24	10	Yes	24
1062	Identified	20	11	Yes	20
1159	Anonymous	22	10	Yes	22
1160	Identified	3	0	No	22
1302	Anonymous	15	13	No	18
1303	Anonymous	13	8	Yes	18

Table 5.1: Data about student responses

Table 5.1 shows the data about how many students responded to a topic, how many students responded with the correct answer and whether choices were provided for the corresponding question or not. These statistics can help an instructor to formulate questions in the future and an instructor can take an appropriate action accordingly.

5.2 Qualitative Data Analysis

Figure 5.1 shows percentage of responded students with correct answer. This information can be useful for an instructor to formulate the questions in future.

Figure 5.1: Average number of correct responses

Gathered statistics can also be analyzed by plotting bar charts for percentage of identified student responses and percentage of anonymous student responses.

Figure 5.2: Percentage of identified student responses

Figure 5.3: Percentage of anonymous student responses

Figure 5.2 shows percentage of identified student responses and figure 5.3 shows percentage of anonymous student responses. By observing these two graphs it can be stated that revealing an identity does not affect on a decision of a student to response to a topic. This information can be helpful to an instructor to make a decision about whether to discuss a topic as identified or as anonymous.

An instructor might want to discuss a topic as identified for checking attendance or whether students did their pre-class reading. Similarly, an instructor might want to discuss a topic anonymously to check the understanding of something that is introduced in today's lecture or something that an instructor did a while ago and don't know whether students remember.

5.3 Analyzing Video Records

A video recording technique is used to capture videos of lectures. Dr. Horstmann captured videos of all lectures in which he used the ActiveLecture system. It has been observed that sometimes an instructor did not use the ActiveLecture application in a way it is supposed to or sometimes even though an application is used in a correct way by an instructor it did not function appropriately due to existing bugs. I found this technique as an effective mechanism to figure out the cause of problems while using the ActiveLecture in the class.

The following sections explain problems faced by Dr. Horstmann and students while using the ActiveLecture system and how they were identified using the video recording technique.

Scenario 1:

As shown in figure 5.4, an instructor entered the lecture id and password to open an existing lecture. An instructor pressed an enter key to open this lecture. As shown in the figure 5.5, the lecture is opened successfully. But the right hand side shows that there no topics in this lecture. Even though topics were included in the lecture before the lecture begins, ActiveLecture was not showing any topics during the lecture.

I observed the video closely and figured out that lecture id in both the figures is different. This clearly states that an instructor did something different than usual. After typing the correct ID and password, the instructor hit an enter key instead of pressing on the Retrieve Topic button manually. As shown in figure 5.4, focus was on New Topic button which was trying to open a new lecture instead of starting an existing one. So the conclusion is to open an existing lecture after entering lecture id and password, an instructor needs to click on Retrieve Topic button manually as shown in the figure 5.6.

Unregistered Instructors

You can use this system without registering.

New Lecture

email (optional) If you supply your email address, you will get a URL + ID reminder

password (optional) If you supply a password, you can resume the lecture later

Retrieve Lecture

ID (required)

password Supply to resume, leave blank to view

Figure 5.4: Enter lecture ID and password to retrieve the existing lecture

http://ActiveLecture.org | Lecture ID = 545

[New Topic](#) [Reopen Lecture](#) [All](#) [Download as PDF](#) [Close Lecture](#) [Student View](#) [Hide Help](#)

Number of Responded Students: 0

[Launch Capture Tool](#)

List of Topics

No Topics in this lecture

Figure 5.5: Started new lecture

Figure 5.6: Enter lecture ID and password to retrieve the existing lecture

Scenario 2:

Clicking on the New Topic button on an instructor interface adds a new topic to the List of Topics. As shown in the figure 5.7 an instructor has already discussed one topic and opened a new one for discussion in the same lecture. The list of open topics is shown on the right hand side. The current open topic for discussion is displayed with green background.

Figure 5.7: Discuss topic

A few students complained that student interface is not showing a textarea to enter an answer. To understand the problem, an instructor opened a student interface using a different browser and joined the same lecture. As shown in figure 5.8, the currently opened topic for a discussion is shown with green background and currently selected topic by a student is indicated by a star. So those students who have selected a different topic than a topic under discussion were not able to discuss a current topic.

5.8 Showing a selected topic and a topic under discussion are different

In order to resolve this issue, a student needs to click on a topic under discussion, (one shown with the green background) which selects the current topic as a topic under discussion as shown in the figure 5.9.

5.9 Showing a selected topic and a topic under discussion as same

5.4 Analysis of Statistics Gathered Using a Video Recording

Also, using video recording it can be easily determined how much time a student takes to respond to a particular question. Using this information an instructor gets idea about how long a topic should be kept open for a discussion.

The following table shows the statistics about when the first student responded, what was the question asked, and whether a problem is easy, medium or hard. Dr. Horstmann provided the mentioned difficulty levels of all questions. It also shows average time taken by students in the class to answer a given question. The last column shows how many students were present in the class.

	First response	Average response time	Question	Difficulty level	Present students	Total time
1	30 sec	1 min 30 sec	int i=1; while(i<5){ i = i+1; print i;	Easy	27	2 min 18 sec

			}			
2	2 min 13 sec	3 min	What is wrong with the letter class? (related to the constructor) Ans: sender and recipient are hard coded.	Medium	27	4 min
3	1 min 28sec	1 min 40 sec	How many times following loop prints "Hello" for(int i=3;i<=11;i+=2) print "Hello"	Easy	24	3 min 20 sec
4	31 sec	1 min	What was the key insight that helped to solve p6.9?	Easy	24	2 min
5	2 min 2 sec	3 min 30 sec	Random integer between 1 and 49?	Medium	21	5 min
6	1 min 15 sec	1 min 51 sec	What does following pseudocode do? Ans: prints vowel groups	Hard	21	3min
7	1 min	1 min 5 sec	What does the following code print? int[] mystery = { 1, 2, 3, 4, 5 }; int sum = 0; for (int i = 1; i < mystery.length; i++) sum += mystery[i]; System.out.println(sum);	Easy	23	3 min 30 sec
8	1 min 12 sec	2 min	Describe in plain English what the following method does. public void mystery(ArrayList<Integer> values) { for (int i = 0; i < values.size(); i++){ if (values.get(i) < 0){ values.set(i, 0); } }}}	Medium	23	2 min 58 sec
9	3 min 22 sec	3 min 35 sec	In plain English, what does this code do? for (int i = 0; i < values.length; i++){ if (i > 0){ System.out.print(", "); } System.out.print(values[i]); }	Easy	19	4 min
10	20 sec	40 sec	How are you preparing for the semester project?	Easy	19	1 min 12 sec
11	1min 16 sec	3 min	What does the following code do? String[][] a = new String[10][10];\r for (int i = 0; i < 10; i++) for (int j = 0; j < 10; j++) if (j % 2 == 0) a[i][j] = 'x'; else a[i][j] = 'o';	Medium	22	3 min 30 sec
12	2 min 5 sec	2 min 20 sec	Suppose a CS46A project team discovers code on the internet that is very similar to	Easy	22	2min 50 sec

			the project that they are working on.			
13	1 min 6 sec	1 min 30 sec	Suppose your friend has a problem with a program that runs without exceptions but prints the wrong output. What is the FIRST thing to try?	Easy	20	2min
14	2 min	3 min 30 sec	Prints the areas of all rectangles in an array list, one area per line.	Medium	22	7 min
15	2 min	4 min 18 sec	Write a loop to print all the numbers from 1 to 100. Print * after all the numbers divisible by 7.	Medium	18	6 min
16	1 min 58 sec	2 min 30 sec	What is wrong with the following code that is intended to swap the first and last element of an array?	Easy	18	4 min

Table 5.2: Statistics of response time, question and difficulty level

It can be stated that students respond quickly if a difficulty level of a question is easy. For medium or hard level questions, the response time becomes twice as compared to the time required to respond to an easy question. Also, it can be observed that students need more time if a question asks to write a code. These statistics give the instructor information about how much time a student needs to answer a particular question. This can help an instructor to formulate the exam questions in future.

Figure 5.10: Comparison of the average time of first response for a first question vs. a second question

In each lecture, Dr. Horstmann discussed two questions. Figure 5.10 shows a comparison of the average response time of a first question vs. a second question. Since a difficulty level of a first question and a second question are not the same, it is hard to comment on the student log in time. However, from figure 5.10, it can be stated that when a difficulty level of a first and a second question is same, response time for a first question is higher as compared to the second. Thus, it can be concluded that students need extra time to log in to the ActiveLecture system which introduces a delay in the first response time. The exact log in time of student cannot be stated due to the lack of sufficient data.

Figure 5.11 Comparison of average response time of students with the difficulty level Easy, Medium and Hard

Figure 5.11 shows a comparison of average response time of easy, medium and hard questions. It can be observed that on an average, students need double the time to respond to a medium level question than an easy level question. Instructor can use these statistics to formulate questions in the future.

Figure 5.12 Showing first response time, average response time and total time for all questions

Figure 5.12 shows the first response time, the average response time and the total time for all discussed questions. The difficulty level of the first seven questions is easy, the next eight questions is medium followed by the hard. From the above graph it can be observed that, on an average the instructor has provided double time for the medium level questions as compared to the easy level questions. So the medium level questions will take more lecture time as compared to the easy ones. In future, this information can help the instructor to make a decision about the difficulty level of the questions to be discussed during the lecture depending on the available time.

Chapter 6

6 Conclusion

In this project I have implemented useful features in the ActiveLecture system to help instructors to gain extra knowledge about student responses. An instructor has used this system in the class to gather statistics about student responses. These statistics are analyzed graphically, which would help an instructor to formulate questions in the future. Also an instructor gets an idea about performance of students in the class.

I found that a video recording technique is a useful feature to capture user errors and bugs in the ActiveLecture system. I have also used this tool to gather statistics about student response time.

References

- [1] 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. DOI= <http://doi.acm.org/10.1145/1275521.1275536>
- [2] Hirayama, M. J. and Yamamoto, T. 2009. Network response analyzer system for interactive lectures in classroom or distance learning. In *Proceedings of the 2nd international Conference on interaction Sciences: information Technology, Culture and Human* (Seoul, Korea, November 24 - 26, 2009). ICIS '09, vol. 403. ACM, New York, NY, 236-239. DOI= <http://doi.acm.org/10.1145/1655925.1655967>
- [3] 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. DOI= <http://doi.acm.org/10.1145/1047344.1047394>
- [4] Taneja, A. 2009. The influence of personal response systems on students' perceived learning outcomes and course satisfaction. *J. Comput. Small Coll.* 25, 2 (Dec. 2009), 5-11.
- [5] Hauswirth, M. and Adamoli, A. 2009. Solve & evaluate with informa: a Java-based classroom response system for teaching Java. In *Proceedings of the 7th international Conference on Principles and Practice of Programming in Java* (Calgary, Alberta, Canada, August 27 - 28, 2009). PPPJ '09. ACM, New York, NY, 1-10. DOI= <http://doi.acm.org/10.1145/1596655.1596657>
- [6] Matthias Hauswirth. Informa: An extensible framework for group response systems. In *Proceedings of the 4th International Conference on Collaborative Computing (CollaborateCom'08)*, November 2008.
- [7] Markett, C., Sánchez, I. A., Weber, S., and Tangney, B. 2006. Using short message service to encourage interactivity in the classroom. *Comput. Educ.* 46, 3 (Apr. 2006), 280-293. DOI= <http://dx.doi.org/10.1016/j.compedu.2005.11.014>
- [8] 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. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education* (Covington, Kentucky, USA, March 07 - 11, 2007). SIGCSE '07. ACM, New York, NY, 384-388. DOI= <http://doi.acm.org/10.1145/1227310.1227445>

- [9] 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. DOI= <http://doi.acm.org/10.1145/1449956.1449969>
- [11] Pargas, R. P. and Shah, D. M. 2006. Things are clicking in computer science courses. 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, 474-478. DOI= <http://doi.acm.org/10.1145/1121341.1121489>
- [12] Pargas, R.P., Levin, A.R. and Austin, J., "Work-In-Progress: Providing Interactivity in a Technology-Rich Classroom", *Proceedings of the 2005 ASEE/IEEE Frontiers in Education Conference*, Indianapolis, IN, pp. F3D 13-14
- [13] Pargas, R.P., "Using *MessageGrid* to Promote Student Collaboration", *Cognition and Exploratory Learning in a Digital Age 2005*,
- [14] The Java EE 5 Tutorial. Web site:
<http://java.sun.com/javaee/5/docs/tutorial/doc/bnbqw.html>
- [15] What is New in JSF 2.0. Web Site:
http://weblogs.java.net/blog/driscoll/archive/2008/11/a_simple_ajax_j.html
- [16] Ed Burns, Roger Kitain. Version 2.0 JavaServer™ Faces Specification.