The LectureScribe Platform

CSC 492

June 9th, 2015

Kevin Backers, Kevin Feutz, Erik Owen

Cal Poly Software Engineering

Abstract

The LectureScribe platform is a free classroom transcription service that is deployed as a website. The platform allows professors to talk into a microphone, and deaf or hard of hearing students have the ability to see what the professor is saying in real time on their personal device. LectureScribe was created by three Cal Poly Software Engineering students. This document details the platform itself, as well as the developers' process of creating this software system over the span of two academic quarters.

Table of Contents

1. Introduction

The LectureScribe System is a website that assists deaf and hard of hearing students in the classroom environment at universities. Students and teachers can create their accounts through the website and have their own personal login credentials. Professors can record their lectures using a listening device, such as a lapel microphone, of their choice. Students follow along and have a live transcription of the lecture on their own computer. Students can also add comments or notes After a lecture is completed, both students and professors can download the text transcription for reference or annotations.

1.1. Description of the Problem

1.1.1 Students With Disabilities at Cal Poly

According to Cal Poly's Disability Resource Center (DRC), it served 742 students with permanent disabilities in the 2013-2014 school year. This number has been increasing, and may be even higher now, just one year later. One group within that is students that are deaf, hearing impaired, or hard of hearing. Cal Poly reports that there are 21 students who are deaf, hearing impaired, or hard of hearing.

1.1.2 What the Cal Poly DRC Provides

According to their website, "The Disability Resource Center (DRC) of Cal Poly provides Transcription Services, using the Typewell Transcription System [8], for deaf and hard of hearing students who are registered with our office". The DRC also provides sign language interpreting services. This service helps students follow along with lectures. It does not, however, provide written notes to the student.

1.1.3 The Cost to Cal Poly

The DRC's website does not share how much they pay for this system, but some base prices are available on the Typewell website. You must pay for training and then pay annual fees for the software licenses. The TypeWell Basic Skills Course costs $540, and the refresher course costs $340. A basic individual software license costs $109 annually, and a premium individual software license costs $219 annually. The sign language interpreting services don't have a specific cost available, but this service relies on a lot of coordination, and "is complex and may involve outside interpreting agencies who require advance notice" according to the DRC website. The DRC also shared on their website that the "DRC was allocated $196,525 for ADA services for the Academic Access/Graduation Initiative. A significant increase in demand from Deaf students for sign language interpreting and transcription services led to expenditures in excess of the anticipated and budgeted amount. The difference was covered by DRC state/university funds."

1.1.4 Significance

For our group, the information stated above makes the problem relevant and interesting. This project is an opportunity to help the school better serve the community of students. More importantly, we can assist students with disabilities and provide them better support to learn and grow at Cal Poly.

1.2. Overview of the Solution

We have created a web based solution to this problem, called LectureScribe. The system has two types of users, professors and students. Both have accounts and login information for authentication purposes. Professors can manage courses, lectures, and students. This means they can create courses, invite students to courses, and record lectures for courses. All students have to do to follow along a live lecture is log in to their LectureScribe account and select their course and lecture. Professors can use any listening device they want. One example of a good listening device is a wireless lapel microphone, such as the Revolabs xTag Wireless Mic [9] that we tested and is described in section 4.1.2.

1.3. Limitations and Unimplemented

As with any project, there is always more work we would love to have time for. We accomplished our major goals, but the following items are limitations or features we did not get to implement. More detail available in section 12.2

- Email notification to students once a professor invites them
- Transcription text download to local machine for students

1.4. Outline of the Report

The major sections of the document following this start with section “2. System Requirements”. Sections 3 and 4 cover how the web based application is actually used with “3. Scenarios for the Web-Based UI” and “4. Using the System”. After that, the next four sections discuss the more technical side of the project. “5. System Architecture” and “6. Voice Recognition System” dive into the important technical details of our system and of the voice recognition systems that we researched and chose from. “6. Networking Design and Implementation” and “7. Database Design and Implementation” detail what you would expect. The document finishes with reports on testing and on future work and uses of the application.

2. System Requirements

The requirements for this project are in the form of user stories. This section contains the user stories for both the professors and the students using this application.

Professor User Stories

1. As a professor, I can create an account so I can use the lecture capturing software with administrative purposes.

2. As a professor, I can create a new course in my profile, so that students can register for that course.
3. As a professor, I can send out emails to my students, so they can have viewing access of the different lectures for that course.
4. As a professor, I can start a new lecture for a course, so that my students can have another way to view my lectures.
5. As a professor, I can pause or resume the voice to text transcription during a lecture, so that only valuable lecture information is transcribed to all of the students.
6. As a professor, I can edit or delete past lectures, so that I can manage what my students can see and access.

Student User Stories

1. As a student, I can receive the email from my professor and create an account, so that I can access the courses that my professor has granted me permission to.
2. As a student, I can view one of the professor's lectures in real time, so that I can have a visual transcript of the lecture.
3. As a student, I can edit my copy of the professor's lecture in real time, so that I can add personal notes to the transcript.
4. As a student, I can download one of the professor's lectures, so that I can have a local copy of the visual transcript.

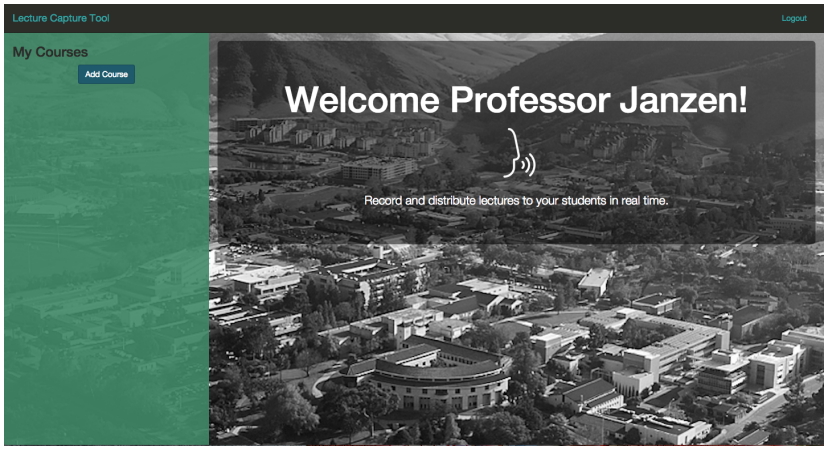
3. Scenario of Use for the Web-Based UI

3.1 Overview

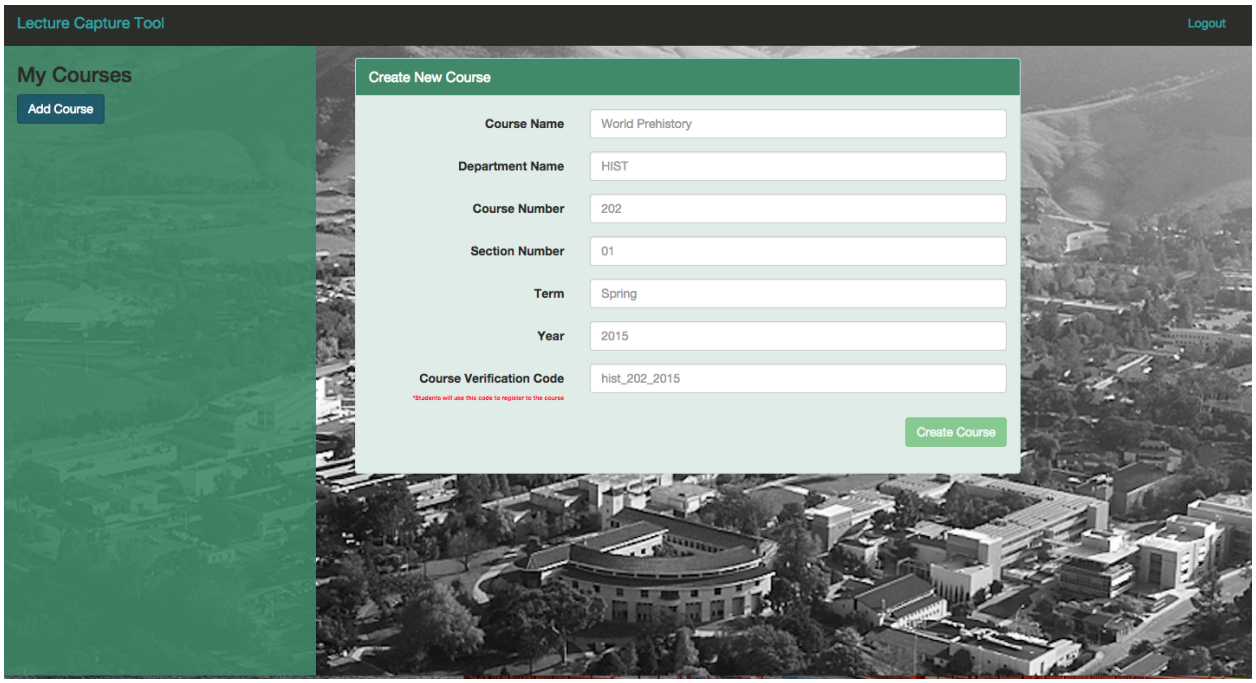
Below is the description of the process a professor using the LectureScribe service goes through to set up a typical user scenario. The professor creates a class, and a lecture in that class. Then the professor can navigate to that lecture and press the record button to start transcribing his lecture.

3.2 Professor Records Lecture

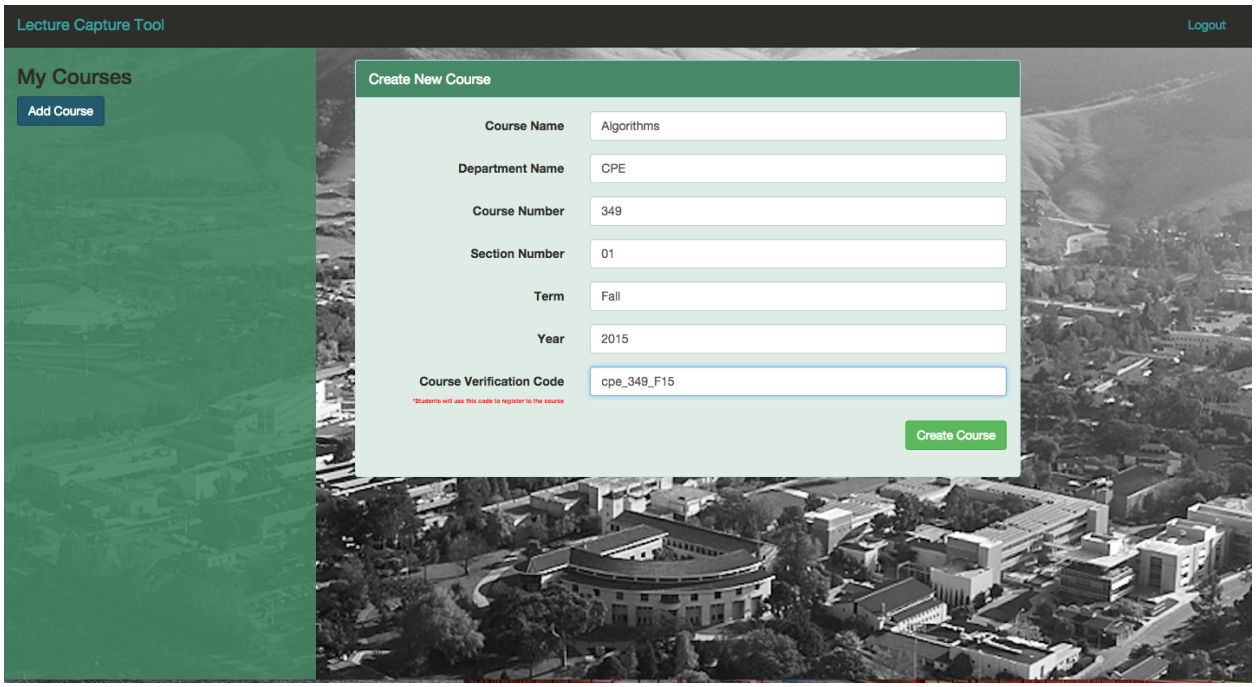
The first time a professor logs in, they will have no courses listed, and this is the screen they are greeted with.



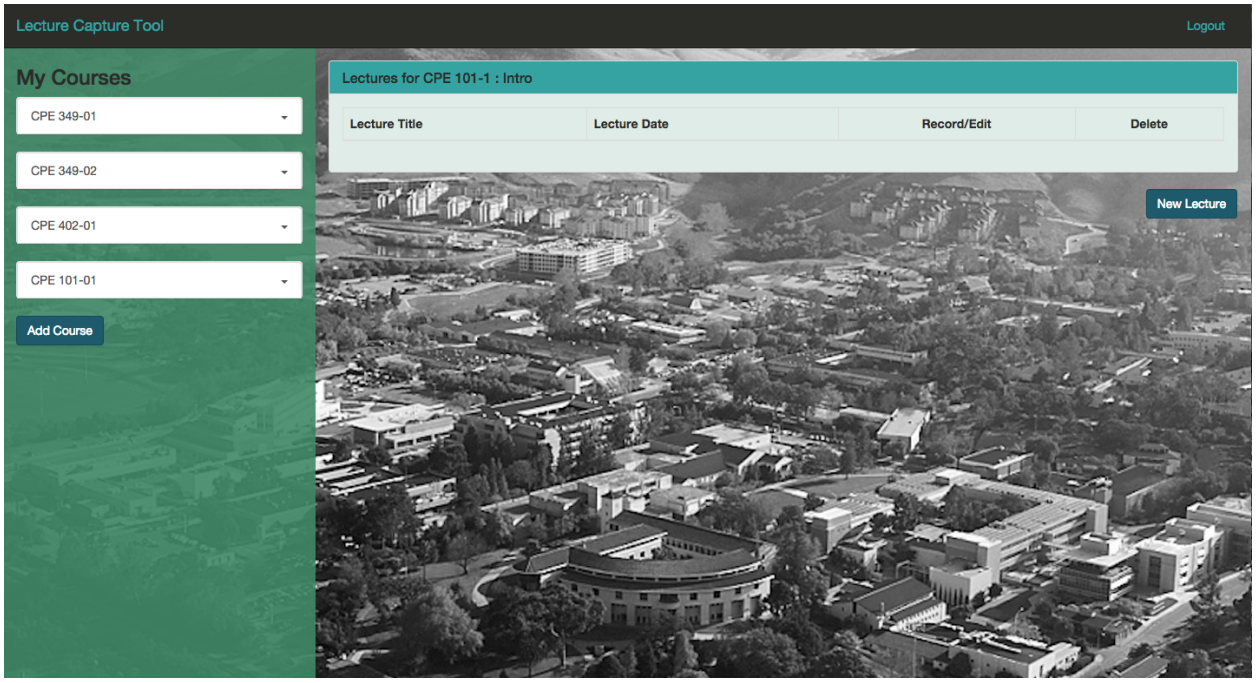
They can add a course on the following screen.



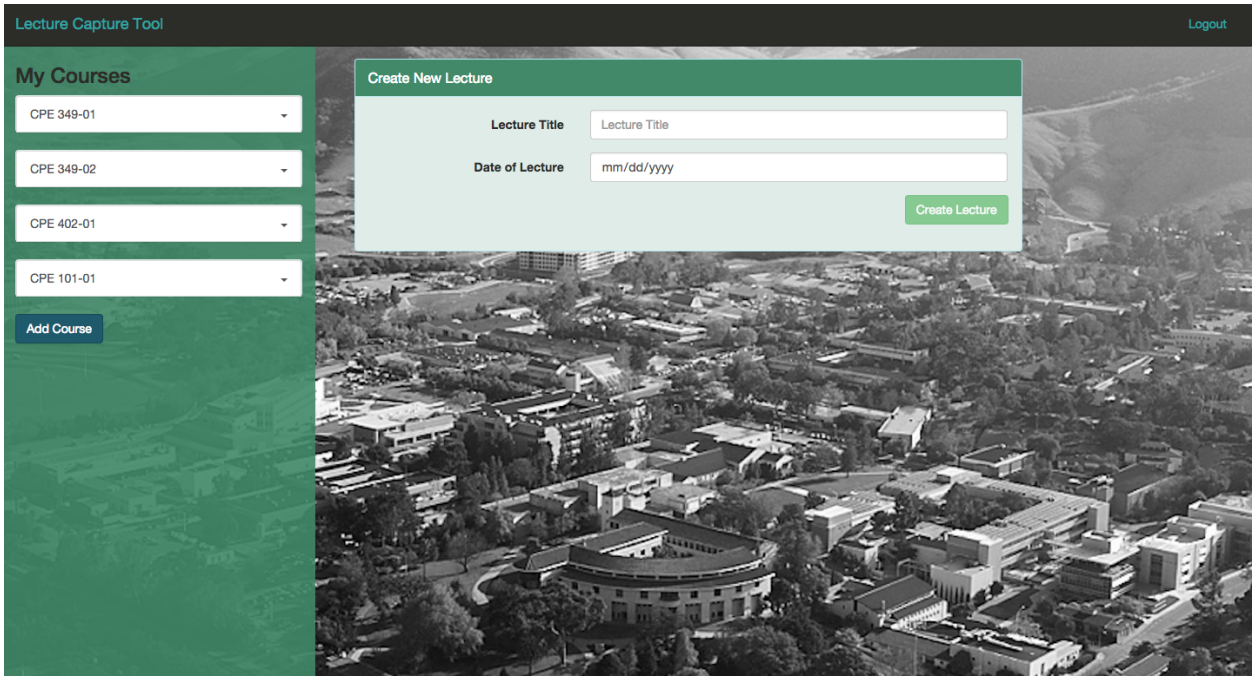
Here it is with information filled in.



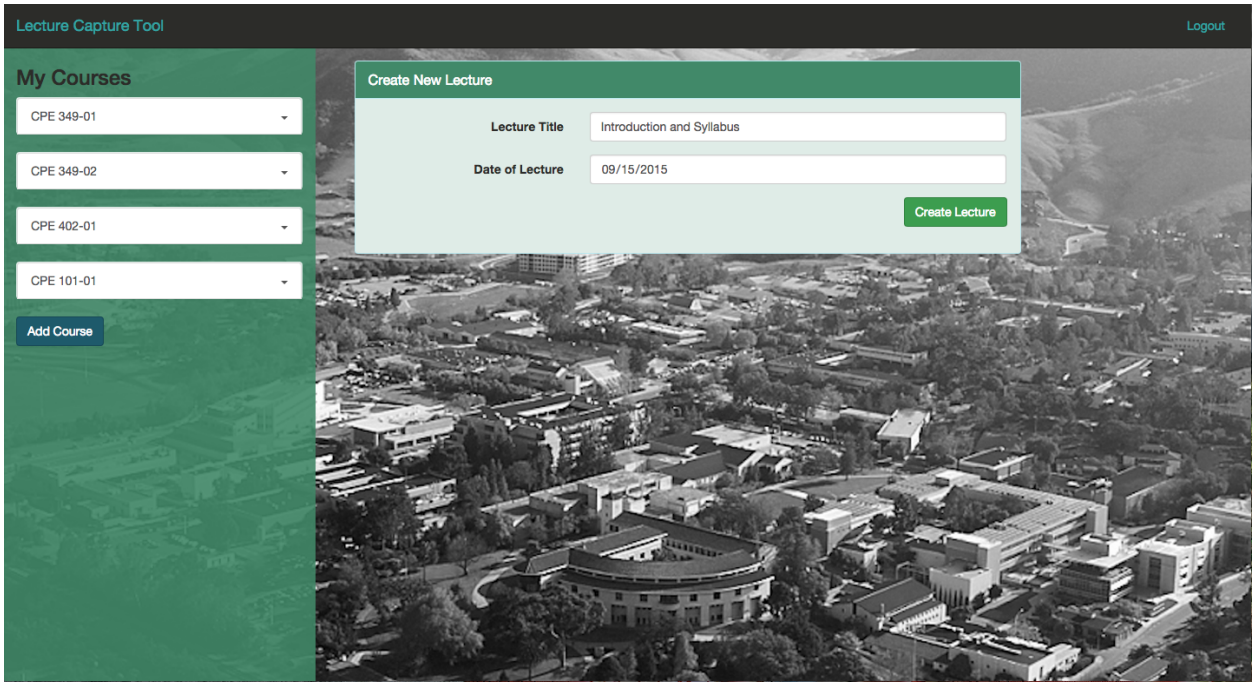
After courses have been created, the professor can create lectures for a course.



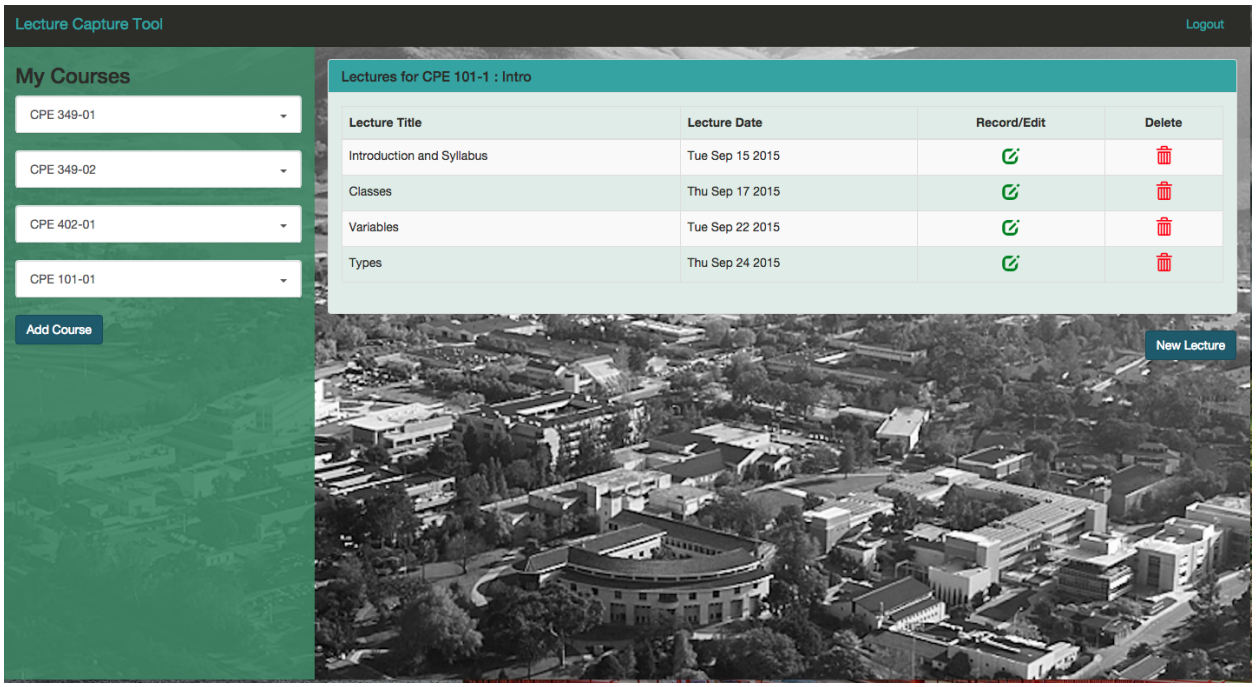
Professors can then create a lecture with a title and a date.



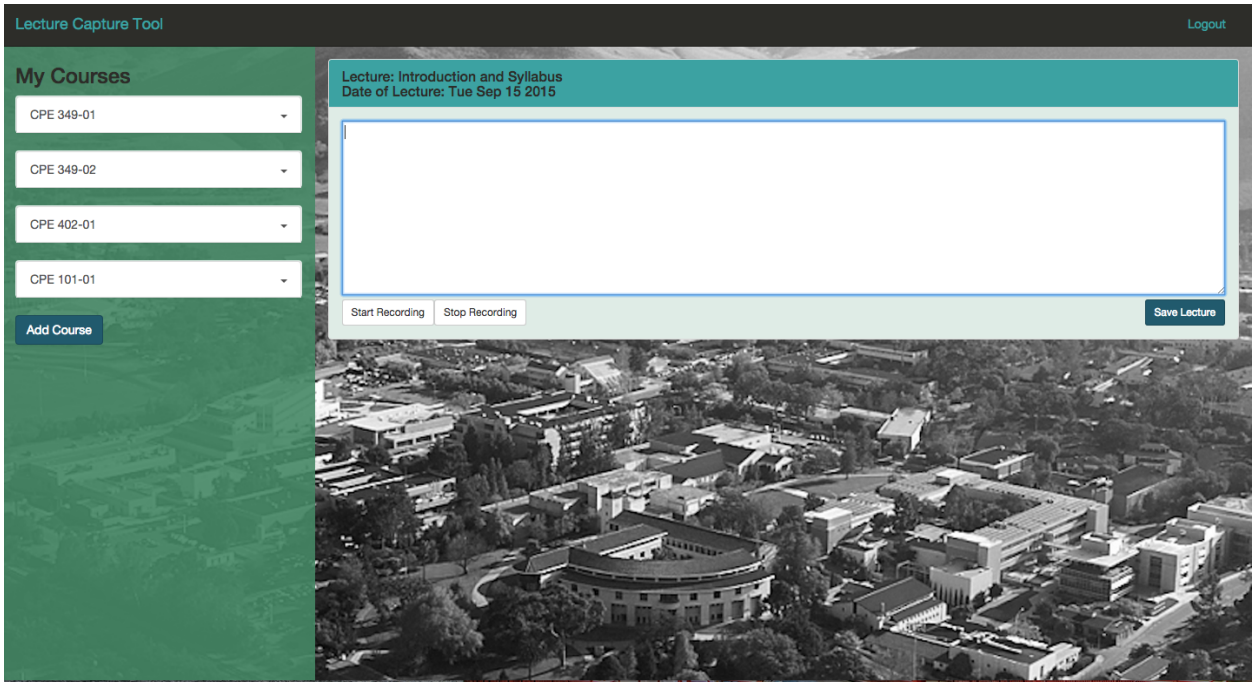
Here it is with data filled in.



The professor can create multiple lectures.



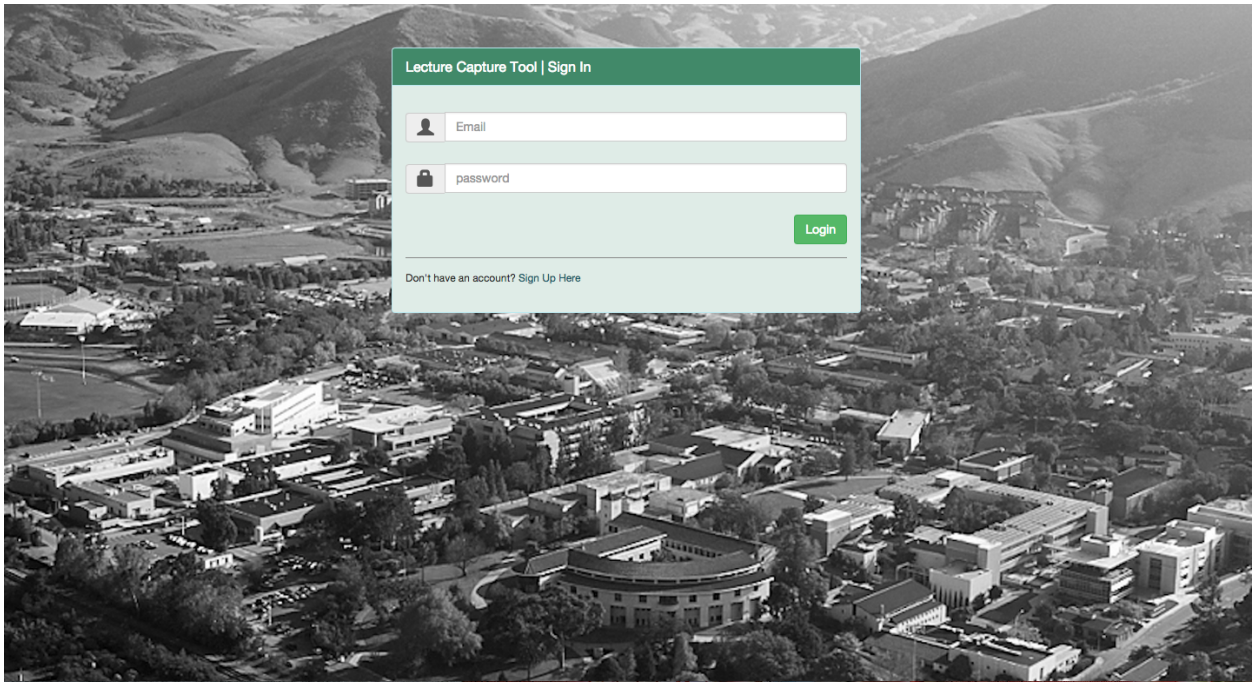
The professors can then record their lecture.



3.3 Student Follows Along with Lecture

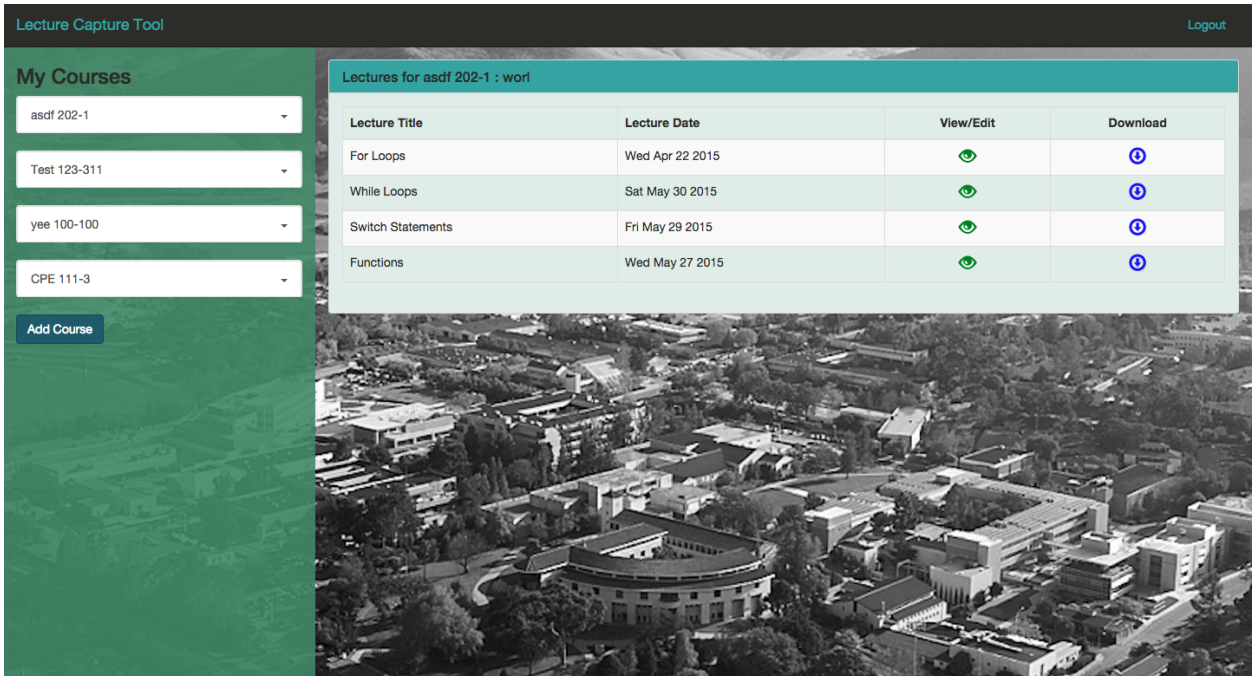
The following screenshots show how a student uses the system. This starts with the login and then follows a few easy steps for them to get to a live lecture.

The student logs in.

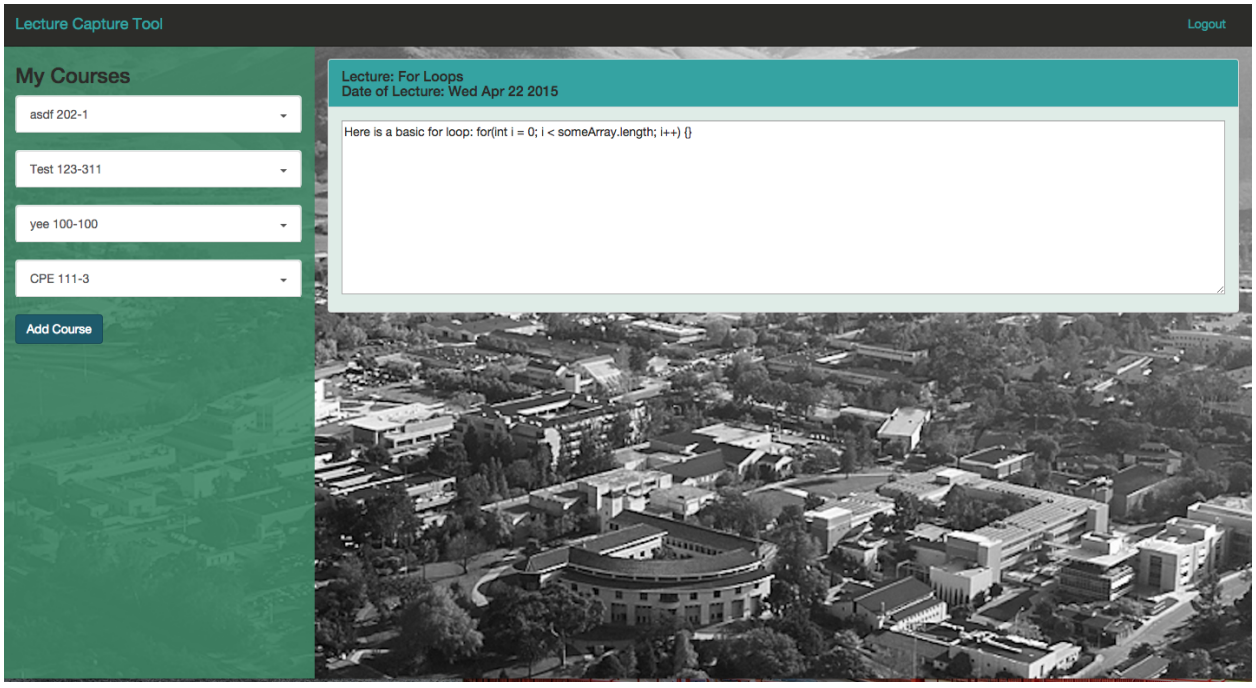


Once the student is logged in, they are greeted with this home screen. This lists the courses they are enrolled in.

Once they select a course, they are presented with a list of lectures in that course.



The student then selects a lecture. If the lecture is from a previous meeting, it will show the transcribed text. If the lecture is the current one, it will show the live text feed while the professor is lecturing. Here is the very beginning of a lecture.



4. Using the System

4.1 Overview

This section details the efficiency and limitations of the transcription service of our platform.

4.2 Voice Recognition Issues

Speech or voice recognition is an extremely high cost and complex operation to perform. A typical voice recognition system contains a significant amount of data models that assist in the identification of utterances. Because of the variance in languages and speakers, speech recognition technologies have some accuracy troubles in general. An issue that we encountered with Google Web Speech API [10] were mild accuracy errors. Examples of accuracy errors encountered consist of incorrect word identification. We found that incorrect word identification is caused mostly by homonyms or poorly enunciated words in our system. In addition, the quality of the microphone used in correlation with our system had an effect on the results of the speech recognition process.

4.2.1 API Speech Recognition Accuracy

After some research, Google Web Speech API proved to be the most accurate and reliable speech to text API that is also free to use. Speech to text transcription technologies are far from perfect in general. Google Web Speech API can achieve an average correct word recognition rate of 74%. [1] This sometimes alters the structure or meaning of a sentence, but sentences prove to be comprehensive enough to understand the majority of the time.

An empirical simulation-based study was conducted using Google Web Speech API by two international professors [2]. The unit used in this study's analysis is the utterance. An utterance is a continuous piece of speech beginning and ending with a clear pause. Depending on the language, utterances can range from 5 to 30 words in length. The results of the study consist of a 81% accuracy rate for utterances in Italian and a 75% accuracy rate for utterances in Brazilian Portuguese. Accuracy rate is strong for speakers that enunciate clearly. Accuracy may drop slightly depending on speaker and utterance differences. For example, the most accurate Italian speaker in the study achieved an 88% accuracy rate while the lowest Brazilian accuracy rate recorded was 68%. Compared to other free API's, Google's proved to be the most accurate and sufficient enough for this application.

4.2.2 Quality of Microphone

The quality of the microphone used with our application and Google Web Speech API had a noticeable affect on the accuracy of the transcription. We completed two processes testing the effect of the microphone quality. We began by selecting a ten minute lecture transcript from Clinton Staley's CPE 357 class. We then read this transcript into our application, comparing the results our application produced to the actual transcript. In our first run-through, we used the default microphone hardware on a 2014 Apple MacBook Pro. With this microphone, we achieved an overall accuracy rate of 71%. Overall, the lecture was legible and comprehensive enough to understand, although there were some noticeable errors.

Our second run-through consisted of us using a much higher quality microphone. This microphone, Revolabs xTag Wireless Mic, achieved an accuracy rate of 77%. With this text, the transcription resulting text from the Revolabs microphone was comprehensive enough to understand the lecture completely. This microphone has extremely powerful audio hardware to handle audio retrieval from the speaker. In addition, this microphone excludes external GSM and background noise interference. [7] Based on the improved results with a high quality microphone, we recommend that users of our system utilize a microphone of similar caliber.

4.2.3 Punctuation of Transcribed Speech

Although Google Web Speech API is sufficient in correctly identifying spoken words, it has trouble identifying punctuation. Google Web Speech API will not transcribe commas, periods, or other various punctuation on its own. The speaker has to specifically say "period" in order for the API to transcribe a period. Because professors do not say these things explicitly when speaking, our lecture transcripts often lack punctuation. To overcome this weakness, our web application gives students the ability to edit the transcribed text as it is received by their client connection.

5. System Architecture

5.1 Overview

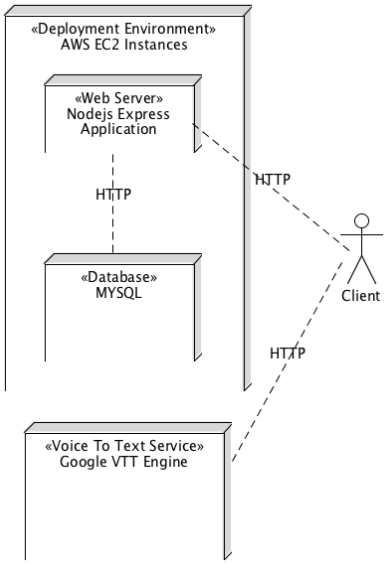
The LectureScribe platform is a web application that can be accessed through any web browser. The web application is deployed on an Amazon Web Service Elastic Beanstalk [11] instance. The deployment consists of two different components: the Nodejs [12] web server, and the MYSQL [13] database. Also, the client communicates with Google's voice to text engine for text transcription, which is separate from our servers.

5.2 Flow of Operation

When a professor is recording a lecture, he/she first talks into the microphone. His/her voice is sent to google's voice to text engine using HTTP. The transcription of the voice is returned, and displayed in the web application. Once the text is received by the web application, sockets are used to transmit the text for the students to view in their browsers who are also in the web application. The data is transmitted using sockets.

5.3 Deployment Diagram

Below is a deployment diagram showing the environments of the different components of LectureScribe.



6. Voice Recognition Systems

6.1 Google Web Speech API

Google Web Speech API provides a JavaScript interface for speech analysis and speech synthesis to web applications. This API is an open sourced project initially developed by the W3C Speech API Community Group. Google and Openstream are currently the primary drivers in its development and maintenance. The Web Speech API supports both speech analysis (speech to text) and speech synthesis (text to speech). The API is globalization friendly and supports many languages. For improved speech recognition, the API can be given a grammar and utilize these constraints.

Our application uses this API by first accepting a consistent audio input stream from the client. Our application then sends this audio data via a HTTPS POST request to the speech recognition web services owned by Google. The transcribed speech is then returned to the client from Google's servers. The text result is sent to our server and directed to the valid clients, its final destination.

6.1.1 Google Web Speech API Advantages

A large driving factor in our decision to use the Google Web Speech API is that it is a free-to-use API. In addition, Google's API proved to be the easiest to integrate into our application. The API uses Google servers to execute all of the speech recognition processes. By using this API, we avoided having to deploy any additional SDK's or images onto our remote server. Instead, we now have the ability to delegate the recognition process to Google's servers. In turn, our server is able to handle responses from Google's servers seamlessly.

Finally, Google Web Speech API showed to be one of the most accurate speech recognition libraries currently available. Google has an entire team devoted to speech recognition research. This team focuses on large scale computing to "rethink the architecture and algorithms of speech recognition, and experiment with the kind of methods that have in the past been considered prohibitively expensive." [6] This team is in charge of Google Web Speech API and have incorporated cluster computing and parallelism into their recognition servers. As a result, Google Web Speech API is an extremely robust and fast speech recognition API that satisfies the requirements of our application.

6.1.2 Google Web Speech API Disadvantages

Google Web Speech API is free to use and as a result has a daily server call limit. This limit has the potential to restrict the scalability of our application. If our application ever grew to the global level, Google Web Speech API usage would be limited and we may be forced to look into alternative speech recognition options.

6.1.3 Google Web Speech API Conclusion

After researching and experimenting with Google Web Speech API, we decided to adopt it as our primary speech recognition engine. The primary motivation behind this choice was that the Web Speech API is free to use. This lines up with one of our initial motivations behind the project, a free alternative to existing lecture transcribers. Another motivation behind this choice includes a high accuracy conversion rate producing clear transcription results. Finally, Google Web Speech API was appealing because it handles all server power for handling the speech recognition process. As a result, we have our own server dedicated to serving the web application data to and from clients.

6.2 Dragon NaturallySpeaking SDK Server Edition

Dragon NaturallySpeaking SDK Server Edition provides a set of tools and interfaces giving developers the opportunity to build and integrate automatic speech recognition dictation into their server. The SDK contain ActiveX controls for easy integration into various server code bases. The digital dictation system in this SDK retrieves and audio stream from the client and converts the data into a .wav file on the server. [4] Dragon NaturallySpeaking then transcribes the dictation, corrects it against various data models, and finally makes the transcription available via .txt file. [5]

6.2.1 Dragon NaturallySpeaking Advantages

This SDK is much more customizable than the Google Web Speech API. It accepts acoustic models and data sets that “train” the system. In turn, the voice-to-text transcription is much more accurate. For example, our application could improve the accuracy of transcription by uploading and categorizing lecture transcriptions to the SDK. In addition, this SDK would be directly on our server only requiring one server call compared to Google Web Speech API requiring an extra.

6.2.2 Dragon NaturallySpeaking Disadvantages

Due to time constraints, using this SDK proved to be too time intensive. The interfaces provided by Dragon’s ActiveX controls allow integration with compiled and interpreted languages including C++, C#, Visual Basic, Python, Perl, etc. [4] Because our server is build on Node.JS, integrating with the NaturallySpeaking interface proved to be too difficult of a task. In addition, this SDK is not a free product and was therefore not appropriate for the scope of this project.

7. Networking Design and Implementation

7.1 WebSocket Library

Our system uses a WebSocket protocol JavaScript library called Socket.IO [3]. This library allows real time communication between web clients and servers. The library has two components, a client-side fragment that runs in the browser, and a server-side fragment that fit extremely well with our Node.JS tech-stack.

7.1.2 Socket.IO And Its Advandantages

The use of this library proved to be beneficial in the long run. First of all, this library handles much of the network connections transparently, so we only had to obtain Socket.IO knowledge rather than WebSocket knowledge. Socket.IO has the ability to effectively manage an unrestricted number of socket connections depending on the strength of the server. This library allows developers to construct specific URL endpoints for socket connection and communication handling. [3] Appropriate server-side code allows developers to handle communications to and from these endpoints. This library feature gave our application support for multiple, concurrent professor-to-student voice-to-text transcription streaming sessions.

7.1.3 Socket.IO And Its Disadvandantages

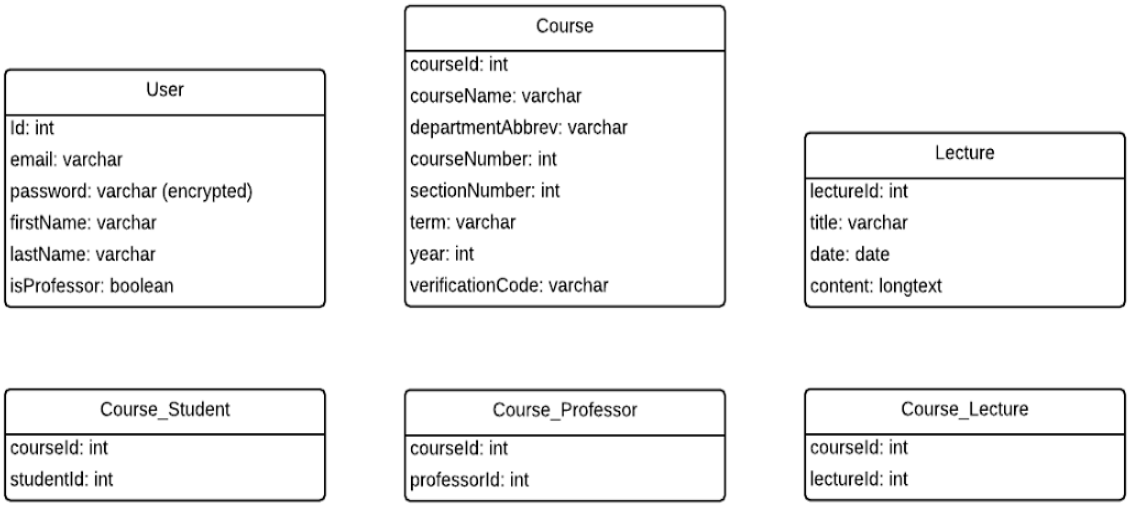
Socket.IO does bring a disadvantage to our tech-stack, however. Clients supporting standard WebSocket will not be able to communicate with a server supporting Socket.IO. The inverse is true as well: a client implementing Socket.IO will not be able to talk to a non-Socket.IO based WebSocket server. This strong client and server dependency makes our application less dynamic. Any future client or server side library or code-base changes and swaps could prove to be extremely difficult with this dependency.

8. Database Design and Implementation

8.1 Design

This section describes the database design and implementation we used to store data for our platform. The core entities in our model are users, courses, and lectures. Users can be students or professors. The *Course_Student* and *Course_Professor* entities join courses and users. The *Course_Lecture* entity joins courses and lectures. This simple model allows all of the information and relationships that are necessary.

8.2 Schema



8.3 Implementation
The database is a MySQL database, which is the most widely used open source relational database management system.

9. System Testing
System testing for the LectureScribe application was a manual process where each feature would be implemented and then tweaked accordingly to meet the requirements by the process of trial and error. Although a test suite would have been more of a productive process, there wasn't enough time to implement a large test suite and implement the majority of the features in the time allotted.

The process of "using our application as a means of testing" deemed to be an acceptable means of testing, because the testing process would have been very difficult. The voice to text transcription as well as the use of sockets would have been a large time commitment to get a test harness around.

10. Related Work
10.1 Overview
Currently there is no platform that both transcribes and distributes lectures in real time. However, there are existing technologies that do one or the other well.

10.2 Existing Transcription Services
There are a few existing platforms that focus on transcribing voice to text. One of the leading platforms in this area of expertise is Dragon NaturallySpeaking. This is the service that the Cal Poly Disability Resource Center uses to transcribe lectures. Dragon NaturallySpeaking [14] is a continuous transcription service that can be "trained" to learn new vocabulary. A document that has specific vocabulary can be inputted into the platform to make the words recognizable to the voice to text engine. This allows the NaturallySpeaking platform to identify certain vocabulary words that are not common, which is useful in academia where a wide variety of vocabulary is used. Other voice to text transcription services include IVR with Speech Recognition [15], Speech Assistant [16], TalkText [17], and more.

10.3 Existing Real Time Sharing Services
The most widely used real time sharing and collaboration serviced used is Google Docs [18]. Google Docs allows multiple parties to collaboratively edit or view a document in real time. Even though Google Docs is a powerful tool, its technology completely align with our problem space, because the overhead and extra functionality would detract from the main purpose of our

platform. Other online real time collaboration options include EtherPad [19], Microsoft Office Live [20], and ThinkFree [21].

11. Conclusions and Future Work

11.1 Overview

Creating LectureScribe was a great project for learning about the software development process. We went through the different stages of building our application, such as: understanding the requirements, creating prototypes to refine the requirements, designing the architecture, implementing, testing, and deploying the application. Through this process we were able to learn about new technologies, as well as get real-world experience making important design and architecture decisions for our application that we would have to implement.

11.2 Critique of Final Product

Looking at the final product, the majority of the main functionality is implemented and working properly. The authentication system is working properly where both students and professors can make accounts and log in and out of them. The creation process is working where the professor can create different classes, lectures, and invite students to them. And most importantly, a professor can speak during a lecture and the student will see the text from the lecture in real time.

However, there are still a few small features that are not complete. We did not complete the following features:

- Application is not deployed to an **easily** accessible URL. We were hoping to deploy the application to lecturescribe.com. Right now, it is deployed to the URL http://lecturescribe-env-ujcre3narj.elasticbeanstalk.com/, which is not very intuitive for the user to find
- The application does not have the functionality where once the professor invites the students, it sends the students an email notification, either letting them know they are enrolled, or prompting them to sign up.
- The student cannot download the transcript of the lecture at this point

11.3 Contributions

The work for this project was divided between the three team members: Erik Owen, Kevin Backers, and Kevin Feutz. Here is a summary of the major contributions to each team member.

Erik Owen:

- Set up the development environment
- Implemented features on the web application

Kevin Backers:

- Tested the application and the accuracy of the voice recognition system
- Designed the database schema

Kevin Feutz:

- Implemented features on the web application
- Researched numerous outside resources and technologies related to our project

11.4 Challenges

Technical challenges in the project consisted of setting up the authentication system, and using sockets to implement the real time distribution of the text from the lecture. Both of these features were technically complex, and took a few weeks to implement.

Challenges that prevented our application from working as intended mainly consisted of limitations with the voice to text engine. Google's voice to text engine would stop functioning after five or so minutes of transcription. Also, the voice to text transcription wasn't as accurate as we would have hoped it to be.

11.5 Future Usage And Plans

The majority of our application is implemented, and can be used by anyone to help professors distribute text (with or without using the voice to text functionality). If professors intend to use our service, they are more than welcome to. We will be leaving the web server up and running. The voice to text transcription isn't working well enough to have our application as functional as we intended, but the rest of the application works fine. Also, we plan on the voice to text technology becoming more and more stable in the future years.

Due to time constraints and busy schedules, the developers of this project won't have significant time to keep contributing to this project. However, some of the developers do plan on contributing bits and pieces to the project when they have time allotted. Also, the developers might plan on open-sourcing the project at a later time.

References

[1] http://home.in.tum.de/~adorf/pub/web-speech-api.pdf
[2]http://delivery.acm.org.ezproxy.lib.calpoly.edu/10.1145/2660000/2652537/a56-calefato.pdf?ip=1
29.65.23.208&id=2652537&acc=ACTIVE%20SERVICE&key=F26C2ADAC1542D74%2E2870C5A035FC0FDB%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=517462023&CFTOKEN=71133295&__acm__=1433648485_295da3a82e3129436596654896b77ffc
[3] http://socket.io/docs/
[4] http://www.nuance.com/for-developers/dragon/server-sdk/index.htm
[5]http://www.nuance.com/ucmprod/groups/dragon/@web-enus/documents/webasset/nc_008810.pdf
[6] http://research.google.com/pubs/SpeechProcessing.html
[7] http://www.revolabs.com/products/product-line/xtag-usb
[8] http://www.typewell.com/
[9] http://www.revolabs.com/products/product-line/xtag-usb
[10] https://www.google.com/intl/en/chrome/demos/speech.html
[11] http://aws.amazon.com/elasticbeanstalk/
[12] https://nodejs.org/
[13] https://www.mysql.com/
[14] http://www.nuance.com/dragon/index.htm
[15] https://www.ispeech.org/ivr/
[16] https://play.google.com/store/apps/details?id=nl.asoft.speechassistant&hl=en
[17] https://play.google.com/store/apps/details?id=com.ktix007.talk&hl=en
[18] https://www.google.com/docs/about/
[19] http://etherpad.org/
[20] https://office.live.com/start/Word.aspx
[21] http://www.thinkfree.com/thinkfree/thinkfreeMain.jsp

Appendix A: Sample of Voice To Text Output

A.1 Explanation

The Speech Herb Brooks gave to the 1980 U.S. hockey team before their game against Russia was used to test the accuracy of Google's voice to text engine. Below is the actual speech, and below that is the speech recorded using Google's voice to text engine. A microphone was used to facilitate the process, and the voice saying the transcript is a middle aged male.

A.2 Actual Transcript

Great moments... are born from great opportunity. And that's what you have here, tonight, boys. That's what you've earned here tonight. One game. If we played 'em ten times, they might win nine. But not this game. Not tonight. Tonight, we skate with them. Tonight, we stay with them. And we shut them down because we can! Tonight, WE are the greatest hockey team in the world. You were born to be hockey players. Every one of you. And you were meant to be here tonight. This is your time. Their time is done. It's over. I'm sick and tired of hearing about what a great hockey team the Soviets have. Screw 'em. This is your time. Now go out there and take it.

A.3 Transcribed Text of Transcript

great moments are born from great opportunity and that's what you have here tonight boys that's what you've learned here tonight one game ever played them 10 times that might win 9 but not this game not tonight tonight we skate with them tonight we stay with them and we shut them down because he can tonight we are the greatest hockey team in the world your appointed the IQ players every one of you and me were meant to be here tonight this is your time their time is done it's over I'm sick and tired of hearing about what a great hockey team the Soviets have screw on this is your time now go out there and take it