Project: Window Pane

ITM 497

Matt Hoekstra

Summary:

The the Window Pane project is the interface that the user will interact with on a regular basis. This web page will allow the user to watch a lecture video and follow along with the interactive transcript. The current audio being played is highlighted in the interactive transcript window. This transcript is synchronized to the video with timestamps. Currently I think a database containing each word of the transcript and that words corresponding timestamp would be the best way to achieve this goal. This transcript would be fully editable by the user so they can correct errors in translation made by Sphinx. The video will pause when the user decides to make an edit. The user can then save their changes as well as print or download the transcript. The transcript will also be searchable allowing users to search for terms and when selected have the video jump to that part. There is also a drop down menu allowing the user to select a different language for the transcript to be presented in. The web page will be written in HTML5 using boilerplate code to allow for cross platform usability, we should have a consistent experience across devices. The page will be hosted on a local Apache HTML Server during the testing and development phase. All code will be uploaded to a git in Bakery to be shared with group.

Current Semester Goals:

*** Web page featuring video player with interactive transcript and core task features**

* Text transcription synched with video playback

* Allow for transcript search

* Allow for transcript to be fully editable

* Basic multi language support(no Chinese, Arabic, Korean, etc)

* Display subtitles over video

Future Goals:

***** Support for more languages with a focus on Chinese

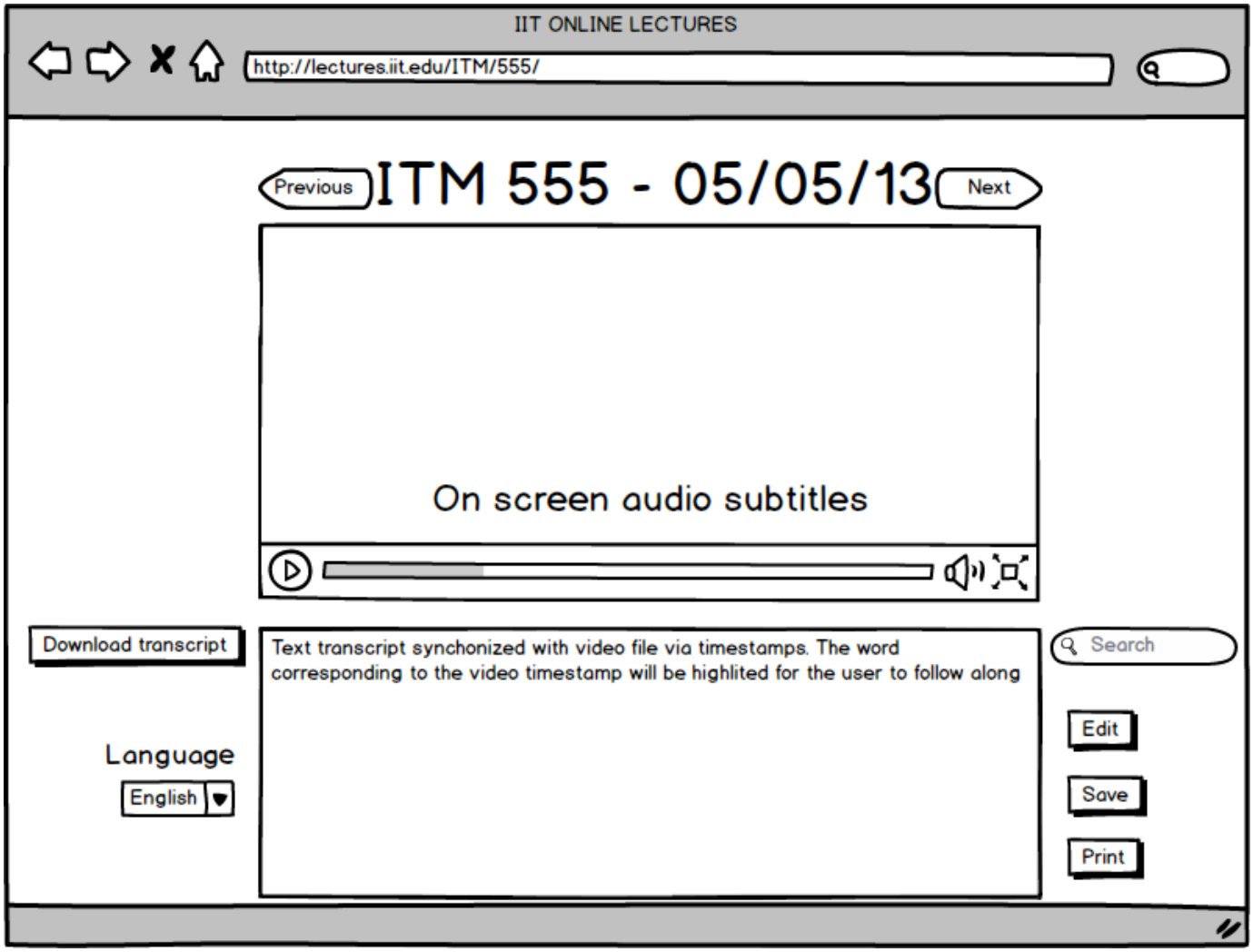
* Incorporate point system for users who make positive corrections

***** Incorporate message boards to allow discussion around lectures

***** Add user login and user profile features

Design:

The design scheme for the user interface is meant to be as simple as possible and easy to use. To achieve this features are limited to the core tasks that the user wishes to achieve. The core tasks that the user wishes to achieve are watching the video(this includes starting and stopping and volume control)ever, following along with the transcript, editing the transcript, saving changes to the transcript, downloading the transcript, printing the transcript, searching the transcript for words, and changing the language that the transcript is presented in. Below is a simple mockup done in Balsamiq for the page the user would see when they decide to watch a video



Technologies:

HTML5 Boilerplate

- Front-end template that implements Google analytics, jQuery, Modernizr, Normalize.css, Apache settings, and other features

960 Grid

- Design scheme used to layout websites cleanly and efficiently

Easy Captions API

- Java library that allows for creation of captions under HTML5 video and creates an interactive transcript

JavaScript, Jquery, Ajax

- Allows for Java to be used effectively and efficiently on multiple browsers

Interactive Transcripts

- Transcript that is linked with video timestamps. When a word is clicked in the transcript the video jumps to that location in player

End Of Semester Summary:

At the end of this semester the Window Pane project consists two Netbeans projects, a HTML5 web page and a Java program. The web page utilizes HTML5 boilerplate and the 960 grid system. I chose to use HTML5 boilerplate to easily and quickly take advantage of the features HTML5 has to offer, these features include but are not limited to video playback, google analytics, ajax, and jQuery.

From this starting point a basic HTML page was created to mirror that of the simple mockup. I decided to start with addressing the video and transcript portion of the page and as a result spent the term dealing with that and did not have time to add editing functionality to the page. So the page as it exists today contains a video player playing one chosen class video and an interactive transcript that draws it's data from a json file. The transcript is based on output from the Sphinx speech recognition library and contains an array of objects with a word, it's start time, and it's end time. The audio is being interpreted from an audio track extracted from the video with which the words timestamps align. This particular json file contains a different transcript than that of the video but was used for demonstration purpose. The words are synced correctly time wise it's just not from the displayed video. The transcript is made interactive and turned into subtitles and displayed using the EasyCaptions library. This is an open source library for creating captions and interactive transcripts easily. It relies on being fed the transcript in the form of span statements containing the word or words and it's start time. This posed a challenge in that the HTML

would need to be generated on the the fly using a script that read the data from a json file or a database. A second challenge posed by the library is it requires the time of the word or phrase to be round to the nearest whole second. The json file as it stands represents the time to the nearest hundredth of a second. A simple conversion to an integer data type corrected this for the prototype however it also meant that some words were assigned the same time value which caused some words not to appear as a caption. This will need to be addressed in the future not just to ensure all words appear in caption but that they appear in readable sentences instead of single words which are near useless.

In the future this page would be one part of a larger user web interface. For example the user might see a page with all the available classes first. Then upon selection of that class this page would appear loading the appropriate video and transcript for the class the user selected. Other future improvements would include utilization of PHP instead of a single HTML file, full transcript editing functionality, multiple language support, and the json transcript would be accessed via jQuery or ajax instead of the solution I relied on for this prototype.

The solution is the second Netbeans project mentioned earlier, a Java program. After spending time attempting to access the json file through a script on the the page I wrote a simple program that utilizes Google's gson library. This library provides a convenient way to interact with json files. Essentially the library was used to pull the array of json objects and converts them to Java objects I created. This new Java object array is then iterated through extracting the word and start time required to create the appropriate span statement to be read by EasyCaptions. The newly generated span statements are then written to a text file from which I copied the result and hard coded into the HTML page. This should all be done on the web page in the future however running up against a time wall I wanted to show a proof of concept and used this technique as I am more familiar with writing Java programs than Jquery scripts. Also there may be issues with EasyCaptions when generating the span statements on the fly, further investigation is needed.

The window pane project is still in it's early phases however I believe it is in a good place to move forward from. Over the course of the next semester or two a fully realized user interface meeting all the objectives set out earlier in the paper should be easily achieved.

Code Examples:

Javascript to enable EasyCaption library in HTML page

```
<!-- JavaScript to enable EasyCaptions function -->
    <script>
```

```
        window.onload = function (){
            var easy = new EasyCaptions({
                videoElementID: "video-html5",
                transcriptElementID: "transcript",
                transcriptEnabledClass: "enabled"
            });

            /*          Progressive enhancement BONUS:
             *          Add notice to transcript letting people know it's clickable */

            if(easy.transcript_element){
                var target_el = easy.transcript_element.getElementsByTagName("h3")[0].nextSibling;
                var new_el = document.createElement("h4");
                new_el.innerHTML = "The following transcript is clickable. Click on any sentence to jump to that \n\
                point in the video.";
                easy.transcript_element.insertBefore(new_el, target_el);
            }
        };
    </script>
```

JSON file containing transcript excerpt

```
[
   {
     "end": "11.45",
     "word": "out",
     "start": "10.39"
   },
   {
     "end": "12.09",
     "word": "a",
     "start": "11.45"
   },
   {
     "end": "12.42",
     "word": "gasket",
     "start": "12.09"
   },
```

Java program that parses JSON document and creates text file with span statements

```
class Transcript {
   // Object that represents json object

   public double end;
   public String word;
   public double start;
}
```

```
public class SpanCreator {

    public static void main(String[] args) throws Exception {

        double q = 0;

        // Convert json array, using Gson, into array and transcript objects
        JsonReader jsonReader = new JsonReader(new FileReader("json/transcript.json"));
        Gson gson = new Gson();
        Transcript[] tr = gson.fromJson(jsonReader, Transcript[].class);

        // Open text file to write output code
        PrintWriter out = new PrintWriter("output/transcriptSpan.txt");

        // Iterate through object array, creating strings containing span statements that contain objects value
        for (int i = 0; i < tr.length; i++) {
            // EasyCaption requires time to be rounded to nearest second

            String span = "<span data-begin=\"" + (int)tr[i].start + "\"> " + tr[i].word + " </span>";
            out.println(span);
            q = (int)tr[i].end;
        }

        // Add final span tag with data end value
        String span = "<span data-begin=\"" + q + "\" data-end=\"" + q + "\"> END</span>";
        out.println(span);
        out.close();
    }
}
```

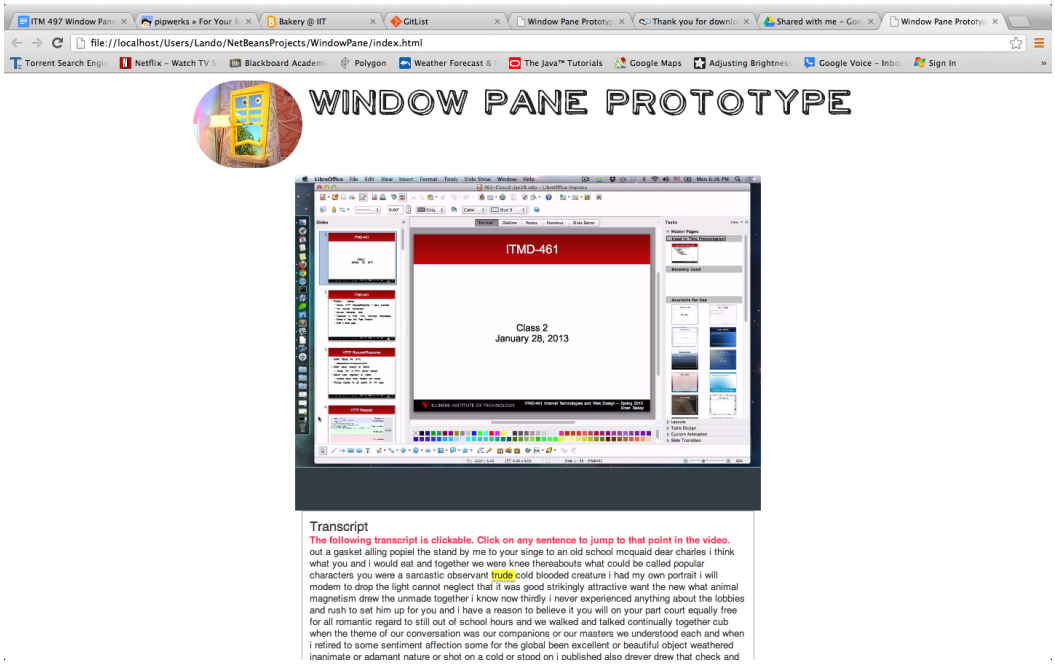
Output file containing generated span statements

```
<span data-begin="10"> out </span>
<span data-begin="11"> a </span>
<span data-begin="12"> gasket </span>
<span data-begin="12"> alling </span>
<span data-begin="12"> popiel </span>
<span data-begin="12"> the </span>
<span data-begin="13"> stand </span>
<span data-begin="13"> by </span>
<span data-begin="14"> me </span>
<span data-begin="14"> to </span>
<span data-begin="14"> your </span>
```

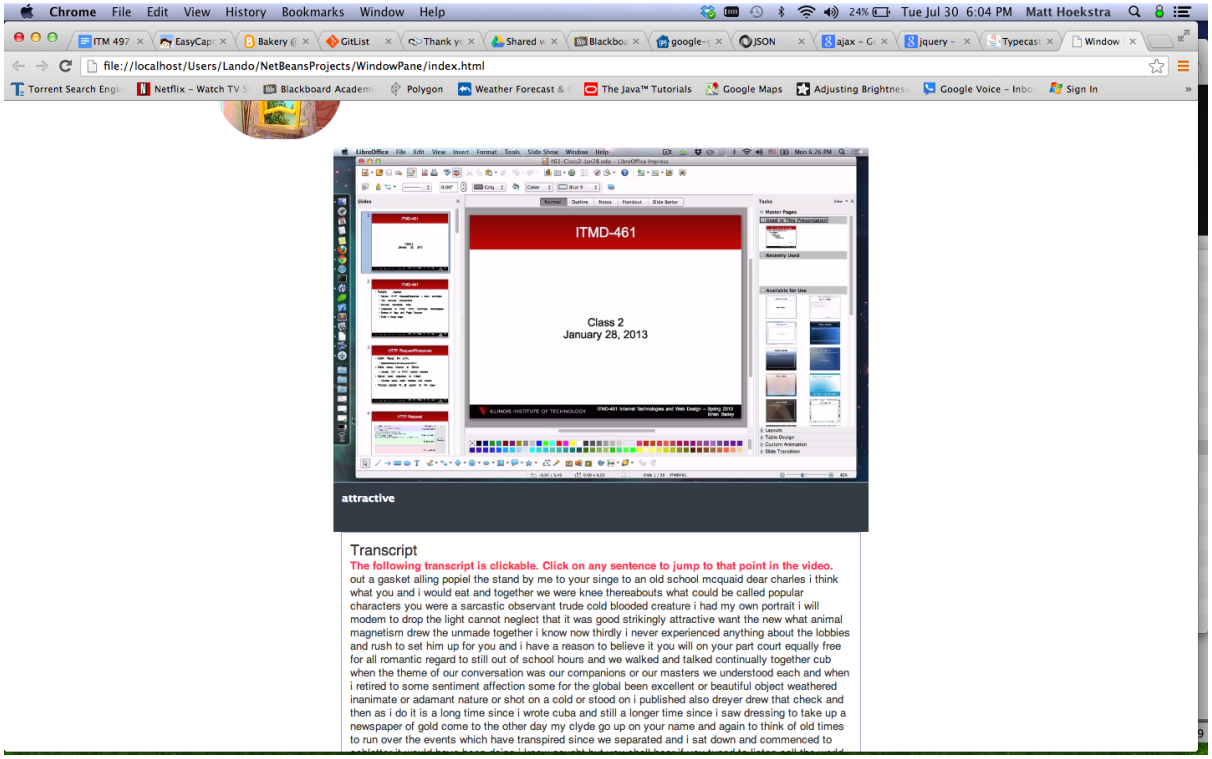
Example span statement that allows for caption display

```
<span data-begin="12">One way to change our genes is to make new ones,</span>
```

Screenshots:



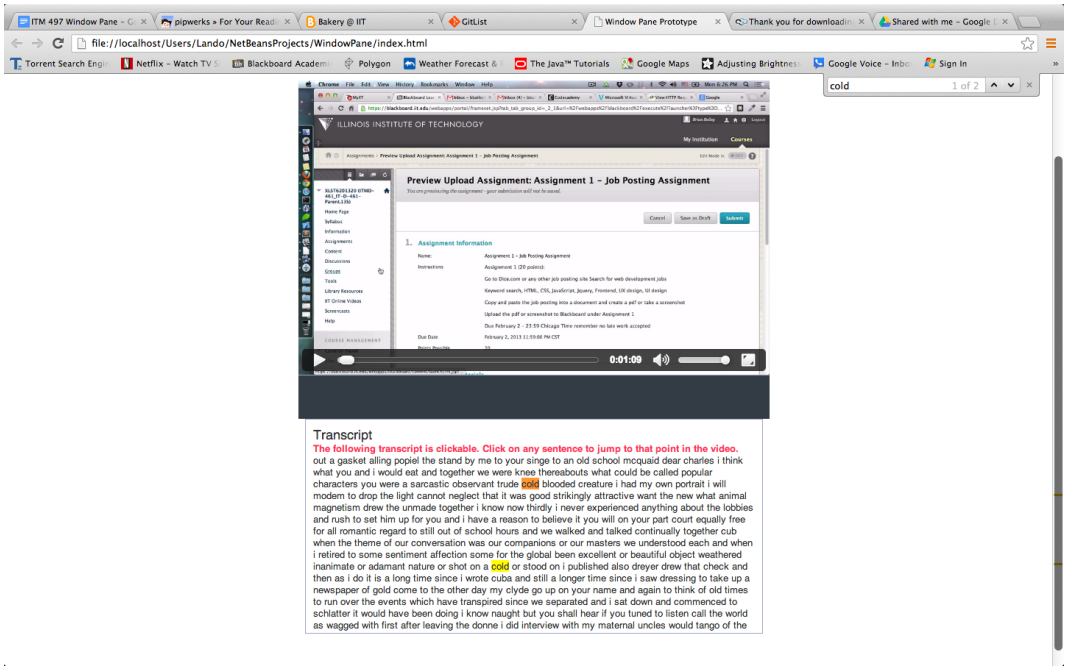
Window Pane Interface



Caption example

(This should be changed in the future display sentences and not

individual words)



Search transcript example

Completed Semester Goals:

*** Web page featuring video player with interactive transcript**

* Display subtitles over video

* Allow for transcript search

* Text transcription synched with video playback

Future Goals:

* Allow for transcript to be fully editable

* Correct subtitles to display sentences instead of individual words

* Add more pages, implement PHP

* Basic multi language support(no Chinese, Arabic, Korean, etc)

* Support for more languages with a focus on Chinese

* Incorporate point system for users who make positive corrections

* Incorporate message boards to allow discussion around lectures

* Add user login and user profile features

Problems For Future Development:

* Display captions as sentences and not individual words

* How to handle multiple users editing the same file at the same time?

* Do changes made by users to transcripts need some sort of approval before made final? What is this process and who is tasked with editing submissions.

* Language translation will be software based and thus not entirely accurate. Should there be separate transcripts for each language that can be edited separately by native speakers or should they be generated on the fly from one English transcript even though it may be confusing.

* Should the transcript be solely Java based, if so then it will not be searchable by Google analytics

Links:

http://learn.jquery.com/

http://www.balsamiq.com

http://html5boilerplate.com/

http://en.wikipedia.org/wiki/Interactive_Transcripts

http://httpd.apache.org/

http://www.3playmedia.com/services-features/plugins/interactive-transcript/

http://960.gs/

http://www.webdesignerdepot.com/2010/03/fight-div-itis-and-class-itis-with-the-960-grid-system/

https://github.com/pipwerks/EasyCaptions

https://code.google.com/p/google-gson/

http://www.json.org/