April 2008

uSCRIPT: an AJAX-based Online Manuscript Transcription Service

Brian Cafferelli
Worcester Polytechnic Institute

Eric Bishop Sutman
Worcester Polytechnic Institute

Nathaniel Walker Piper
Worcester Polytechnic Institute

uSCRIPT: an AJAX-based Online Manuscript Transcription Service

A Major Qualifying Project Report:

Submitted to the Faculty of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

By:

Brian Cafferelli

Nathaniel Piper

Eric Sutman

Advisors:

Fabio Carrera

Stanley Selkow

April 23, 2008

http://www.uscript.org

Authorship Page

Statement from the team: *All team members shared equally in the creation of this report.*

Abstract

This project, a continuation of the ongoing Emergent Transcriptions Initiative, redesigned and implemented the Transcription Assistant software system. The system provides tools for archivists and transcribers in order to preserve ancient documents in a digital, searchable format. Specifically, it converted the system to a more appropriate web-based architecture. The software is composed of a web browser based user interface, a series of server-side services, and a set of databases for storing all the related transcription and user information. The project also established a web “home base” for the past, present, and ongoing project.

Table of Contents

1 Introduction

The history of the world is contained in handwritten manuscripts stored in archives across the globe. Individual documents may seem unimportant, documenting day-to-day affairs such as birth and death records, police reports, and other government acts. However, when collections of these manuscripts are considered as a whole, one can view histories of entire civilizations.

Unfortunately, environmental conditions such as moisture ensure that these manuscripts, which are typically written on paper or parchment, will not last forever. They yellow and grow moldy, eventually becoming unreadable or even disintegrating altogether. Historians are charged with the duty of transcribing these manuscripts to inform the public of the history they describe.

The most common and useful process for transcribing manuscripts is to create a digital transcription. Today, a historian will sit down at a computer with a manuscript, open up a word processor such as Microsoft Word, and start typing. There are many drawbacks to this method, including lack of support for manuscript-specific font-styles (such as erased text and words written in a different hand) and an absence of collaboration opportunity. After a historian has finished transcribing a manuscript and puts it back on the archive shelves, another person could walk in the next day and start transcribing that very same manuscript that has been done the previous day, with no way of recognizing the redundancy.

To deal with this very important issue, our project continues the work of the Emergent Transcriptions Initiative, which seeks to build a software system to aid historians in transcribing documents, as well as building an online community of transcribers cooperating to help build a cumulative, reliable and transparent process for transcribing manuscripts. It seeks to reduce the inherent redundancy of multiple, possibly several, people transcribing any one document from scratch, by allowing transcriptions to be shared and gradually improved, this expediting progress towards a complete computerized record of the information in the world's archives.

As a proposed answer to the problem of redundant transcription and dearth of manuscript-specific word processing, the uSCRIPT system employs the asynchronous JavaScript model (*AJAX*) with the help of the Google Web Toolkit (GWT) to establish a lightweight web application that responds to user input fast enough to emulate the look and feel of a traditional desktop program.

Chapter 2 provides a comprehensive review of the historical importance and uses of manuscripts along with a summary of previous work completed by project teams as part of the Emergent Transcriptions Initiative.

Chapter 3 explains the goals of our project and how we achieved each of them. This chapter contains implementation and architecture details for the uSCRIPT system.

Chapter 4 contains the recommendations for students working on a future project as part of the Emergent Transcriptions Initiative, as well as conclusions about the work we did as part of the initiative decided by the team at the end of our project.

Appendix A explains how to use sourceforge and subversion to access uSCRIPT source code.

Appendix B explains where to find database documentation files.

Appendix C displays a database structure proposed early on in the project.

Appendix D displays the current transcription database structure.

Appendix E provides a look at the structural hierarchy of the user interface.

2 Background

Development of a tool to help transcribers first requires an understanding of the research process when working with historical manuscripts. Our work is built on previous work done that has already examined this facet of the Emergent Transcriptions Initiative in great detail. This chapter is intended to provide the reader with an overview of historical research using manuscripts as well as a picture of the status of the Emergent Transcriptions Initiative at the beginning of our work.

2.1 Overview of Historical Research Using Manuscripts

At the core of the Emergent Transcriptions Initiative are historical manuscripts. These are ancient documents that usually were never intended to be published. The Emergent Transcriptions Initiative is intended to streamline the process of using manuscripts in historical research. It is therefore important to understand the importance of manuscripts, how they are cataloged, what transcriptions are and why they are important, and how manuscripts are stored digitally. This section explains all of these, and gives an overview of the process historians typically follow when doing historical research with manuscripts.

2.1.1 Importance of Manuscripts

The most important type of source of information in historical research is a "primary source". A primary source of information is a document, artifact or other source of information that was created close to or during the event or time period being studied. There are three types of primary sources used in historical research: (1) Oral traditions and oral histories, (2) Physical artifacts, and (3) Documents, both printed and handwritten (http://memory.loc.gov/learn/lessons/psources/analyze.html). Of these three types, the third is perhaps the most important and prolific.

The Library of Congress further divides the third category into published and unpublished works. This distinction is important because few copies exist of each unpublished work. These predominately handwritten documents are usually referred to as "manuscripts" and can consist of personal correspondence, town records, court proceedings, and many other first-hand accounts of historical events.

These manuscripts can be found in thousands of repositories across the globe, stored in archives which protect them as best as possible from physical deterioration, and attempt to catalogue them in a useful fashion. These tasks often prove difficult, as the collections are usually quite large—the works of a single author often total in the hundreds of thousands of pages (Rath et al., 2004).

Much of the information contained in these documents is relatively unimportant—minutiae of everyday life, well documented in other texts. However, often historians will "unearth" important data about significant events buried in these accounts. For this reason, these vast collections of historical documents are important because of their potential to reveal lost secrets about the past.

Figure 2.1: An Example of a Damaged Manuscript

Sifting through this data is a long and arduous process, and unfortunately the time we have to do so is limited. The materials these historical accounts were recorded on are fragile; they were never intended to last hundreds of years. Archives devote immense resources to protecting and preserving the documents in their care. Water, mold, and even light can damage manuscripts until they become unreadable. Archives must restrict access to their collections in order to reduce the number of hands that handle any given document; each viewing of a manuscript quickens its deterioration.

2.1.2 Transcriptions

The traditional solution to the conflicting needs to preserve and use historical documents is the creation of transcriptions. The creation of transcriptions is also typically one of the steps a historian takes in processing a document during research. Transcriptions have two primary benefits over original documents. First, they are not as fragile as the source document. Second, they are often easier to understand and read than the original. Note that a transcription is not to be confused with a translation, whose primary purpose is to transform the content of a document into a different language.

A transcription is a document which is intended to take the place of the original. It is more than a simple copy of the final text of a document; typically it also describes other features of the manuscript. Marks and notes in the margins, writing between lines, and stricken passages can all be as important to a historian as the main body of text (Finch, 1999).

Creating a transcription is a time-consuming and labor-intensive process (Rath et al., 2004). Manuscripts are often difficult to read because of damage and age, and difficult to understand because of archaic words, expressions, and constructions (Finch, 1999). Still, they are an essential part of the research process, especially because archives limit access to the original documents.

2.1.3 Digital Manuscripts

Recently, many archive collections have begun to be converted to digital format. This process allows for more frequent use of manuscripts without subjecting them to the handling that is involved in physical examination. This is good for both archives and researchers, as documents can be simultaneously better protected and more widely used.

Digitization of a document typically begins with scanning its pages, producing a digital image. Large collections of these manuscript images are difficult to use, however, because current technology for searching image contents is not very developed (Rath et al., 2004). The common solution to this issue is to manually attach metadata describing the document to the image in the database.

One organization that has attempted to provide a standard for representing digital versions of documents is the Text Encoding Initiative (TEI). The Guidelines published by the organization are designed to allow "a variety of literary and linguistic texts" to be represented for online study and preservation (http://www.tei-c.org/). The various XML and SGML elements that the organization provides are intended to be inserted by hand into plain-text transcriptions of texts in order to mark features such as chapter beginnings, headings, and lines of poetry.

2.1.4 Manuscript-Based Historical Research

In order to create a tool for studying historical manuscripts, the existing process for doing so must be understood. This facet of the Emergent Transcriptions Initiative was previously examined. What was found is that the current process suffered from major inefficiencies, and is time-consuming and labor-intensive. Their description of the process was based on the system in place at the Venetian Archives, but a similar procedure exists in other archives. Figure 2.2 shows an overview of the procedure.

The process begins with historians searching the archive's records of manuscripts they own. Unfortunately, information about the contents of the manuscript is limited, so the search must be based on the metadata the archive has on file. This means that historians' search results can yield a large number of possibly useful manuscripts.

Researchers then request a manuscript from the archive by collection set and series. Archive personnel check the archive's reserve to see if the requested document is already in reserve. If it is not, they must search the archive's stacks for the manuscript. When they find it they put it in reserve. Once the document is in reserve, it is available for researchers to study.

The delay between the request and receipt of a document can be as long as two weeks due to the size of the collection and inefficiencies in the cataloging system.

Once researchers are notified that the manuscript is available, they may view it in the archive's reading room. This is the only location in which people who do not work for the archive may view the original documents. Therefore, all work with the document must be done in the reading room. For this reason, transcription is an important part of the research process.

The actual transcription of the document is often a long and difficult task. The process often requires use of dictionaries of abbreviations and symbols and other reference texts. A researcher may require multiple trips to the archive in order to complete transcription of a single document. On each visit, the researcher must again have the manuscript brought to the reading room, and must locate the point at which they stopped work. Once the transcription process is complete, the researcher returns the original document, and keeps the transcription for personal use.

Two characteristics of the existing research process make it particularly inefficient. First, the methods for locating manuscripts are currently not precise enough to quickly search for manuscripts covering specific topics. Second, the transcriptions created by researchers are rarely shared with other members of the research community. Together, these two characteristics lead researchers to repeat work already completed by other historians, and to use well-known manuscripts rather than uncovering new information.

Currently, searches for manuscripts are largely restricted to the contents of the metadata for each document. Archives already spend significant time and resources keeping these records accurate (Rath et al., 2004). Unfortunately, this data is often not specific enough for researchers searching generic records for mention of specific people, places or events.

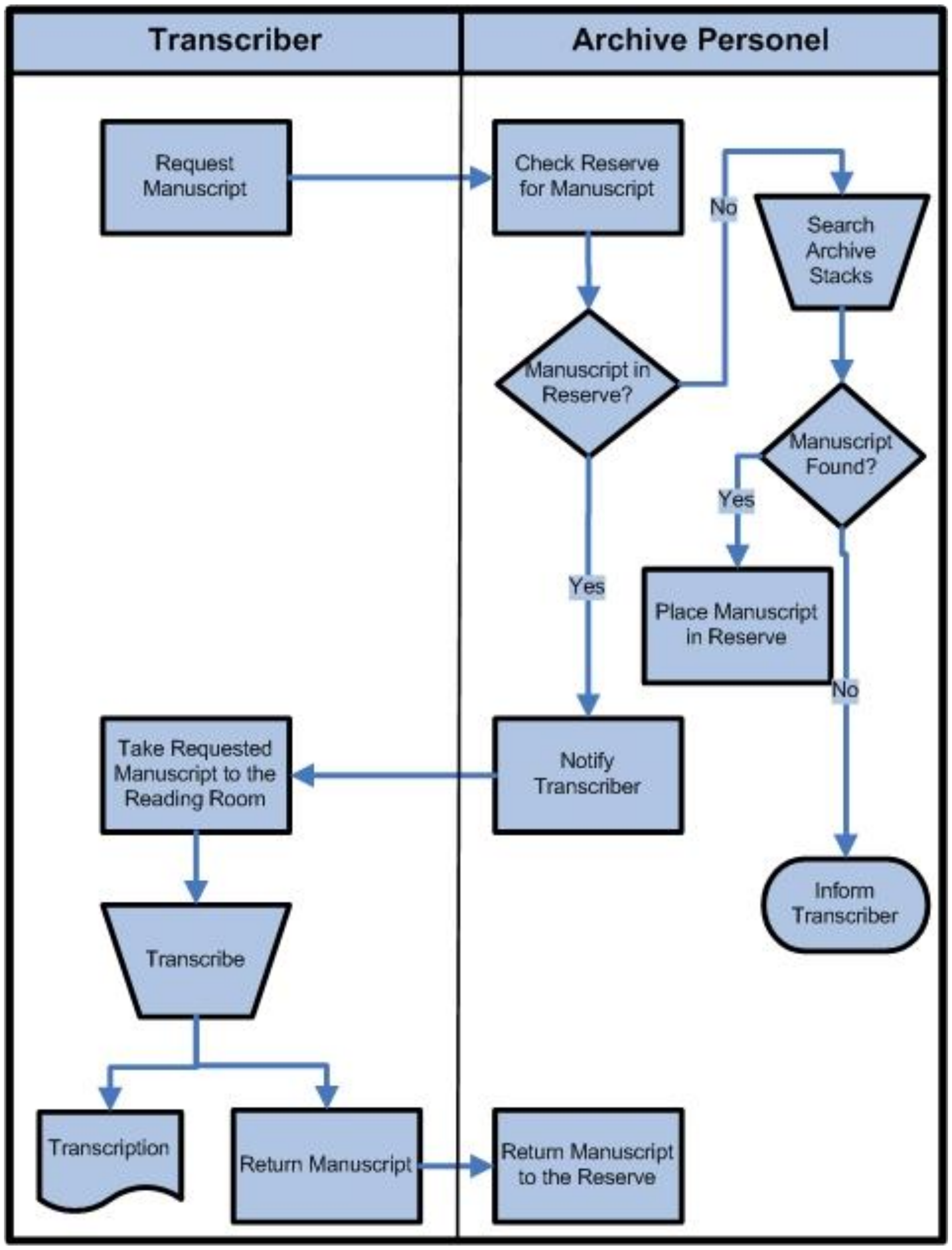


Figure 2.2: The Traditional Transcription Process

A study of history professionals in 2002 found that the most often used method of finding primary sources during research was finding leads in other printed sources, which 98 percent of respondents said they regularly used for research (Tibbo, 2002). The next three most commonly used methods (81 percent each) were printed bibliographies, printed finding aids, and printed repository guides. The first of these three is very similar to finding leads in printed sources, and the latter two must be compiled by the owners of collections (a time-consuming task). By far the most commonly used electronic method of searching was searching an OPAC database, which was only used by 78 percent of respondents.

This information means that historians are using manuscripts that have already been analyzed by their peers far more often than they examine any of the incredible volume of manuscripts which have not been touched in years. This leaves a possible vast resource of information untapped. If more sophisticated methods for searching are not developed, more and more documents could be lost to deterioration before their value can be determined.

The other major problem with manuscript research as it currently stands is that the same work is often repeated many times by different researchers. When a transcriber completes the transcription process, he keeps the transcription for his own personal use. The archive takes back the original document, but usually does not even record that a transcription was made. When other researchers choose to use the same manuscript for their own work, they often end up repeating the transcription process.

While the archives are partially to blame because they do not keep records of transcriptions, the overall system is far more responsible for this gross inefficiency. As the current historical research process stands, there is no incentive for a transcriber of a manuscript to share his work with the community. Doing so can give others working on similar projects an advantage, and allow them to publish their findings first.

2.2 Goals of the Emergent Transcriptions Initiative

The overall goal of the Emergent Transcriptions initiative is to create a system that will leverage the work that is already being done by visitors to archives around the world. It seeks to create an ever-growing collection of transcriptions that are refined by the work of many researchers. In this way, it hopes to uncover important information hidden in thousands of still untapped manuscripts (Carrera, 2005).

The Emergent Transcriptions initiative is predicated upon five assumptions, according to Carrera (2005):

- "There are precious few researchers with the necessary paleographic skills who are able to produce reliable transcriptions of ancient manuscripts;
- "Despite this crucial bottleneck, these few capable individuals frequently duplicate efforts by re-transcribing the same exact manuscript that someone else has already worked on, often unbeknownst to each other;
- "The constant manipulation of the primary sources (parchments and the like) renders them less and less legible as time goes by;
- "Very few manuscript transcriptions are published verbatim;
- "The work put into transcriptions is subsumed into scholarly journal articles and books, thus it is rarely if ever seen or re-used by others." (Carrera, 2005)

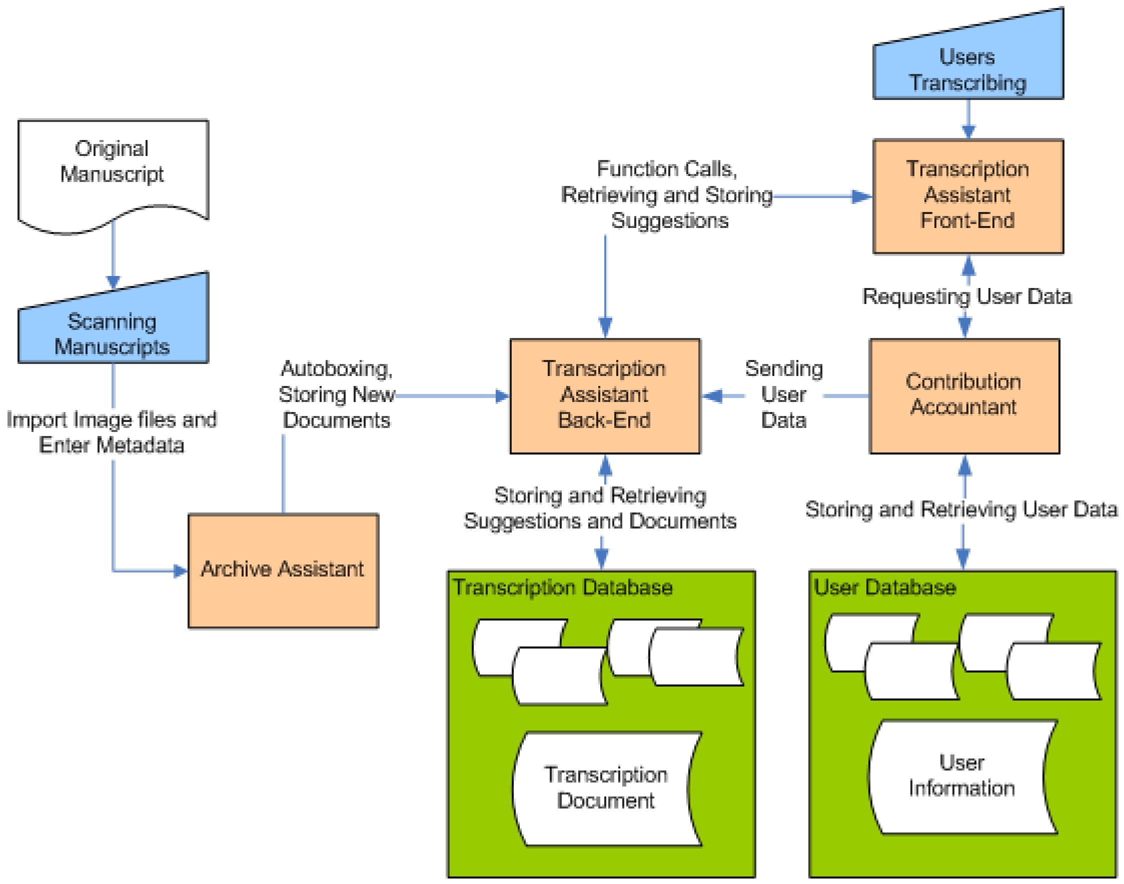
2.2.1 What is Emergence?

At the core of this project is the idea of *emergence*. In this case, it refers to the way in which high-quality transcriptions will emerge from the work of many different participants. The contribution of each researcher who contributes to transcriptions within the project will be added to the system's repository of information. This is not a new idea; similar projects, such as Wikipedia, have been built on similar principles.

The Emergent Transcriptions initiative builds upon that idea by adding the concept of credibility. As a given transcriber's work is viewed and modified by others, acceptance or modification of that work contributes to the original submitters' credibility score. As more people accept a transcriber's work as correct, the credibility score of that transcriber will increase. If others modify that transcriber's work, their credibility score will decrease. When deciding upon an accepted transcription out of the work of many different transcribers, the system will consider these credibility scores. (Freitas & Glajch, 2007)

2.3 Components of the Emergent Transcriptions Initiative

The final version of the Emergent Transcriptions initiative will consist of three related subsystems. These are the Transcription Assistant, the Archive Assistant, and the Contribution Accountant. The role of each of these will be described in the following sections. An overview of the various components and how they interact is shown in Figure 2.3.



Users Transcribing
Original Manuscript
Function Calls, Retrieving and Storing Suggestions
Transcription Assistant Front-End
Scanning Manuscripts
Requesting User Data
Autoboxing, Storing New Documents
Transcription Assistant Back-End
Sending User Data
Contribution Accountant
Import Image files and Enter Metadata
Storing and Retrieving Suggestions and Documents
Storing and Retrieving User Data
Archive Assistant
Transcription Database
User Database
Transcription Document
User Information

Figure 2.3: High-Level View of the System

2.3.1 Transcription Assistant

The most visible part of the Emergent Transcriptions initiative is the Transcription Assistant. This tool will be used by all transcribers while transcribing manuscript images that the archive has loaded into the Emergent Transcriptions initiative. It allows the transcriber to retrieve, review, and contribute to existing transcriptions and manuscripts.

The Transcription Assistant is particularly important to the project because it will serve as the primary incentive for researchers to use the Emergent Transcriptions initiative. It is intended to make the task of transcription easier in at least four ways. First, it makes it easier for a transcriber to find their place in the manuscript should they have to consult another reference. Second, it may in the future be able to give suggestions for words through sophisticated Optical Character Recognition techniques. Third, it will allow researchers to consult existing transcriptions in the Emergent Transcriptions database. Fourth, and perhaps most importantly, it will allow a researcher to retrieve a manuscript or transcription almost instantly from the server. It is important to ensure that transcribers want to use the Transcription Assistant, because it will be the primary incentive for them to contribute their transcription work to the larger community. (Freitas & Glajch, 2007)

2.3.2 Archive Assistant

The Archive Assistant portion of the Emergent Transcriptions initiative comprises a number of tools which assist the archive in managing the Emergent Transcriptions initiative. Most of these tools will make up the back end of the system. This portion of the system includes three major tools. First is the database for all uploaded manuscript images. Second is a tool for archivists to easily and quickly load manuscript images and metadata into the database. Third is the automatic manuscript processing tool, which automatically boxes, orders, and uses OCR techniques in order to index manuscript images which have not yet been transcribed. The Archive Assistant also includes a user database and manager for all active users. (Freitas & Glajch, 2007)

2.3.3 Contribution Accountant

The Contribution Accountant is the portion of the system that tracks the "reputation scores" for all known users. The contribution assistant will constantly monitor the transcriptions database, and update a user's credibility score as they contribute to a transcription. It will also use its records of credibility to determine the currently accepted text of the transcription.

Though the credibility system has not yet been designed, some basic features have already been decided. There will be two types of credibility scores: scores for individual users, and scores for each modifiable item in a transcription. Each time a user modifies a transcription, the portions of the transcription they changed are counted as "votes" for their own submission. A given user's vote would be weighted by their own credibility, and would modify the credibility score for all items they confirmed or changed. A user's credibility score would be increased when their submissions were confirmed, and decreased when their submissions were modified. The credibility scores of elements in transcriptions would be used to determine the “accepted text” of a transcription. Simply contributing to the project by submitting a transcription would increase a user’s credibility by some base amount, in order to encourage participation.

Another aspect of the credibility system that has been proposed is allowing access to more sophisticated features only to high-credibility users. For example, normal users might have access to a basic OCR library or a limited number of uses of the OCR system during transcription, while a higher credibility user could use author-specific OCR systems and could do so more frequently. The credibility system could also be used to encourage users to give back to the community by restricting the number of transcription downloads allowed in a given time period based on the credibility of a given user. (Freitas & Glajch, 2007)

2.3.4 Role of the Project in Manuscript Research

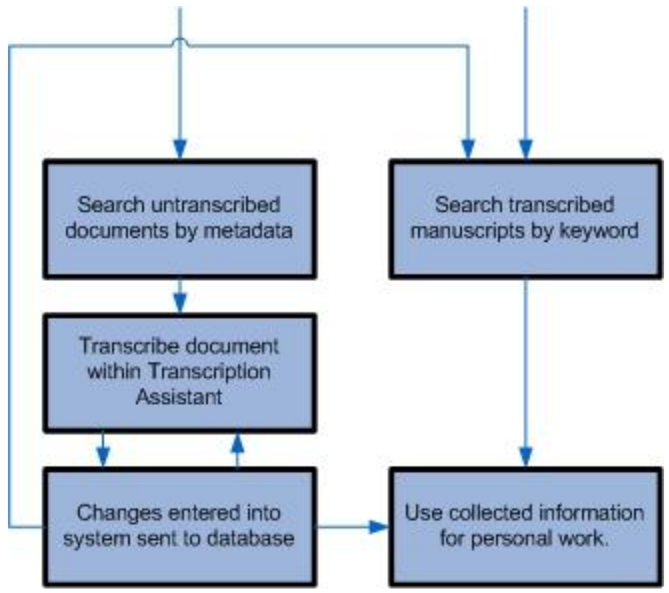

Search untranscribed documents by metadata
Search transcribed manuscripts by keyword
Transcribe document within Transcription Assistant
Changes entered into system sent to database
Use collected information for personal work.

Figure 2.4: Manuscript Workflow Using the Emergent Transcriptions System

Figure 2.4 demonstrates the way in which the manuscript research process will change once the Emergent Transcriptions initiative is used at an archive. There are now three parts to the process. The transcriber and the archive personnel still have significant roles in the process, but they no longer directly interact. Both work with the Emergent Transcriptions server to get things done.

The process begins with the archive personnel entering a manuscript into the system. This will be an ongoing process until the archive's entire collection has been added to the Emergent Transcriptions initiative. The archivist will scan the manuscript, attach some basic metadata to it, and upload it to the database, all with the help of the Archive Assistant. As a particular archive will eventually accumulate a huge amount of data, the system will be expanded such that each archive participating in the initiative will operate its own storage server, and grant access to the system. A portion of the Archive Assistant on a central server will then index the manuscript and enter the correct data into the database. The archivist can then continue to upload additional manuscripts.

Once manuscripts have been uploaded, the transcriber can request to view manuscripts using the Transcription Assistant or an associated web application. Transcribers will be able to search the manuscripts using a variety of criteria, including full text of a transcription, and any of the standard metadata elements. The server will respond almost instantly, retrieve the requested manuscripts, and make them available to the transcriber. This will allow transcribers to quickly browse through the results until they find one that interests them.

At this point, the transcription process begins. The Transcription Assistant will make this step of the process much easier, but it will likely still involve a significant amount of work. As the transcriber works with the transcription, their contributions are dynamically updated on the server.

As each contribution is made, the Contribution Accountant will examine the changes the transcriber made, and update the credibility information in the database accordingly. Any changes the transcriber made will decrease the credibility scores of both the changed elements and the corresponding transcribers. If the change the transcriber made was to a value a previous transcriber entered, that transcriber's credibility will increase. All changes in credibility will be weighted by the current transcriber's own credibility.

There is one additional portion to this process: automatic processing that the Archive Assistant does when it has free cycles. There are currently two proposed activities which the Archive Assistant could engage in when idle: Automatically indexing manuscripts that have not been transcribed using OCR methods, and optimizing OCR and auto-boxing for specific classes of manuscripts. These two processes should allow un-transcribed manuscripts to be found in searches, thereby increasing the likelihood that more and more manuscripts will be examined. (Freitas & Glajch, 2007)

2.4 Previous Work on the Emergent Transcriptions Initiative

The Emergent Transcriptions initiative has been ongoing for some time, with the most recent work in the 2006-07 academic year. This section gives a brief overview of the status of the initiative at the beginning of the 2007-08 academic years when our contribution began.

2.4.1 Transcription Assistant

A team of WPI students created the first major version of the Transcription Assistant in 2003, which is the tool to help someone transcribe an existing manuscript. Their work featured a boxing system on top of the manuscript image that allowed the user to work with both the original image they were boxing, and the transcription text side by side. The user would draw a box around a word, and then double click on the box to edit the text value associated with the box. That text shows up on the adjacent panel in the same position where the box was in the image. A screenshot of the system at the beginning of our project can be seen in Figure 2.5. Though this system was updated during another project in 2004, it still was not a complete working program, and several of the components were either not completely implemented, or had bugs. Still, the general interface for editing the boxes was present, so we decided to work off of and redesign this program. (Freitas & Glajch, 2007)

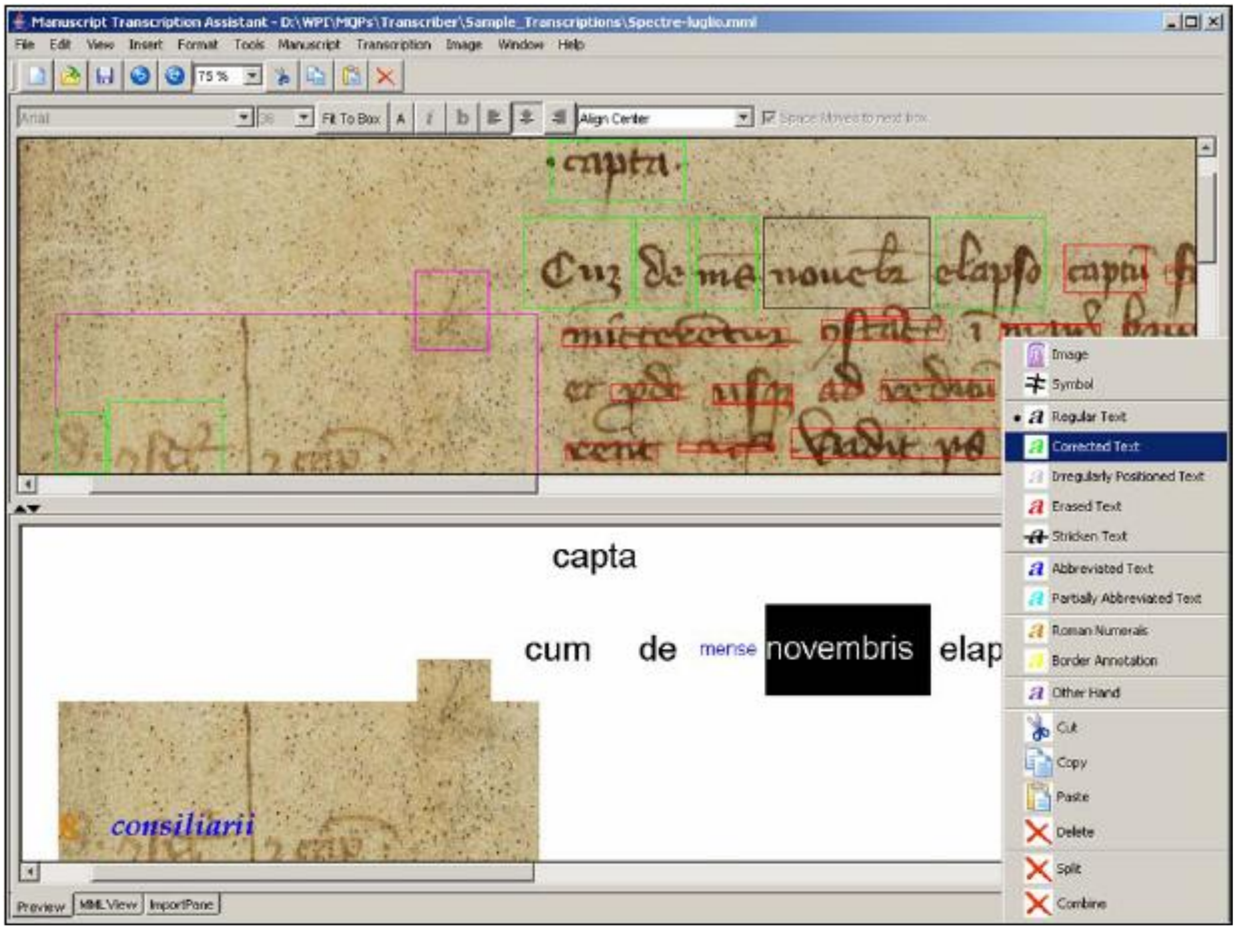

Manuscript Transcription Assistant - D:\WPI\MQPs\Transcriber\Sample_Transcriptions\Spectre-luglio.mml
File Edit View Insert Format Tools Manuscript Transcription Image Window Help
Fit To Box
Align Center
Space Moves to next box
capta
cum de mense novembris elap
consiliarii
Image
Symbol
Regular Text
Corrected Text
Irregularly Positioned Text
Erased Text
Stricken Text
Abbreviated Text
Partially Abbreviated Text
Roman Numerals
Border Annotation
Other Hand
Cut
Copy
Paste
Delete
Split
Combine
Preview
MML View
ImportPane

Figure 2.5: Previous Completed Work on the Transcription Assistant

2.4.2 Manuscript Markup Language (MML)

A team of WPI students in 2003 had devised a basic structure for storing the transcription information for a particulate manuscript that they labeled as MML, or Manuscript Markup Language. This was the data structure implemented in the previous Transcription Assistant program. While the MML version was not fully XML compatible, it worked in a similar fashion. It was basically a list of unordered box tags. Each tag has five attributes, the x and y pixel locations for the top right and bottom left corners of the box, as well as a type field, which indicated whether the box was a text box or image box. Within each tag was the actual word or words that the user input as the transcription for the word or words boxed, along with some simple HTML-style text-markup identifiers, such as bold tags (Ho et al., 2003, pages 40-42). While the old projects did store metadata for each transcription entered by the user, it was not stored in the MML. Another WPI student team in 2007 decided to restructure this MML into a new fully XML compliant structure which allowed for a more descriptive document representation. (Freitas & Glajch, 2007) The redesigned MML allows for the expression of the full document history, integrates metadata, enables transcriber notes, and maintains a specified word order.

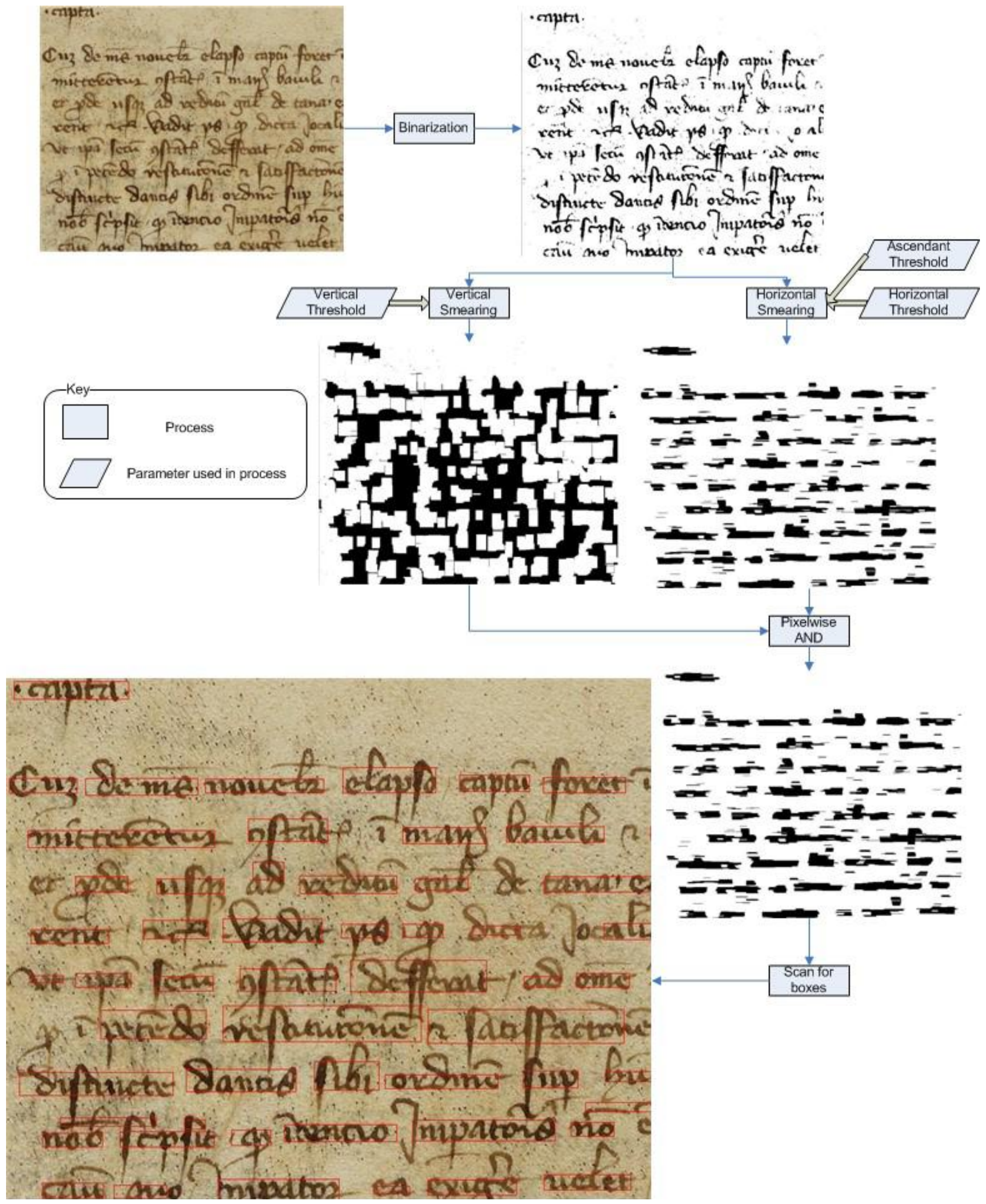


Figure 2.6: The Smear Automatic Boxing Process (Freitas & Glajch, 2007)

2.4.3 Automatic Boxing

A team of WPI students in 2004 extended the Transcription Assistant to include an algorithm for automatically creating boxes around words in a manuscript, and then displaying those boxes on the screen. Though this algorithm places no order on the boxes, and did not do a complete job of boxing every word correctly, it worked well enough that the 2007 WPI student team based their auto-boxing research off of the existing tool.

The algorithm converted the color image temporarily into a grayscale image. Then using the Java Advanced Imaging's (JAI) existing conversion tools, it binarized the image into a black and white image. This black and white image was passed through an algorithm which goes through the image horizontally and vertically looking for runs of white pixels smaller than a certain threshold, and turns the small white pixels runs to black pixels. The result is that the image looks smeared horizontally and vertically. These two images are combined by taking the cross-section of the black smeared areas, to create an image with sections of black smears over each word. Then boxes are created around these words, and finally boxes that are too small or overlapping are combined. This algorithm has some thresholds which can be manipulated to produce better performance in different styles of manuscripts. (Freitas & Glajch, 2007)

2.4.4 Optical Character Recognition (OCR)

A key piece of the final Emergent Transcriptions initiative will be a sophisticated Optical Character Recognition (OCR) system. Research on OCR algorithms revealed that at this point in the Emergent Transcriptions initiative, it was not practical to begin work on the OCR system. The extent of the work that has gone into OCR development is the automatic word boxing algorithm.

2.4.5 Genetic Algorithms

There are many advanced algorithms for pattern recognition and image manipulation that are planned for inclusion in the Emergent Transcriptions initiative, and each of these will likely be difficult to optimize. The 2007 project realized that optimization can be made easier and faster through the use of a “Genetic Algorithm” (GA). A GA uses a method which mimics evolution in order to find a near-optimal solution to a problem. Without doing the complex, long, and often unfeasible task of analysis required to derive the optimal solution, a GA can find an excellent solution in a fraction of the time.

At the core of the GA is a “chromosome”, which is a series of bits that represents a unique solution to the problem. Typically, these are made up of all of the different parameters that determine how a process functions. Chromosomes are the elements in the GA that evolve and improve over time. The purpose of the GA is to find the chromosome that represents the best solution to the larger problem.

The chromosomes in the GA are created in “generations”. Each generation is a collection of chromosomes that are created and evaluated at the same time. All of the generations in a given GA typically contain the same number of chromosomes. Each chromosome in the first generation is randomly generated. Each chromosome in subsequent generations is produced by “reproduction” between pairs of chromosomes in the previous generation. Reproduction pairs are selected based on a “fitness value” that is calculated for every chromosome. (Freitas & Glajch, 2007)

Both the processes of reproduction and selection in a GA are discussed further in the 2007 report.

2.4.6 Text Line Detection

An important set of information that hadn’t yet been accounted for was the order of the boxes (and therefore words) on a page. This information is necessary for making logical sense of the transcriptions on a page. The 2007 project designed a line detection subsystem in an effort to order the words.

A proof of concept for this technique was developed using a line detection algorithm based on what is called the “Vertical Projection Profile”. The algorithm examines fixed-width columns of the entire image, and locates Partial Segmentation Lines (PSLs) at locations likely to be the bottoms of words. After finding all the PSLs for all of the columns, the algorithm joins PSLs across the entire page into word line borders. (Freitas & Glajch, 2007) This process is described in greater detail in the 2007 report. The advantage of the algorithm was that it could be optimized with the same genetic algorithm that could optimize the automatic boxing system. Although it was functional it was not efficient or accurate, and needed significantly more development.

The 2007 student project team realized that their desktop application, which saved transcriptions on a client’s machine in an MML file, did not sufficiently fulfill their goal to provide opportunity for historians to collaborate. Since uploading a saved transcription would’ve been an extra step for the user and there was at that time no incentive to do so, the team concluded that they had to accept that people would typically not bother submitting their work to share with others. Though it was never fully functional, a java applet version of the Transcription Assistant was developed in an effort to more effectively build an online community of transcribers. The 2007 team also pioneered a single-table mySQL database stored on WPI’s database servers to store transcription data. Metadata was stored directly in the database, with pointers to both the manuscript image and MML file for each document.

3 Discussion

At the outset of this iteration of the Emergent Transcriptions project, we determined that the overall initiative required a thorough evaluation. Past projects resulted in software with various levels of functionality, but the initiative was yet to produce fully functional and useful software. There was useful code, but too much of it was interlaced with inefficient, non-working, or useless code. Many of the past groups had spent considerable amounts of time sorting through previous projects trying to make sense of the code. This was an immense waste of time that could have been better utilized if the projects were better organized and documented.

Realizing that our work would be another step in the initiative, and that future groups would again be using our work as a building block, we decided that our primary goal would be to build a functional framework for the system that would be easily understood and easily extensible. In order to achieve this, a complete system re-design was necessary.

We took this opportunity to reanalyze the requirements for the project, and gauge whether the initiative was being fueled by the end-user requirements, or if it was steering off course. It was easy to see how an ongoing long-term project such as ours might have drifted off course due to continually compromised requirements. We realized that continually refreshing the requirements is necessary to this project's success. Many of the past projects dealt largely with researching user requirements, and were our primary source for determining the current requirements.

3.1 Software Redesign

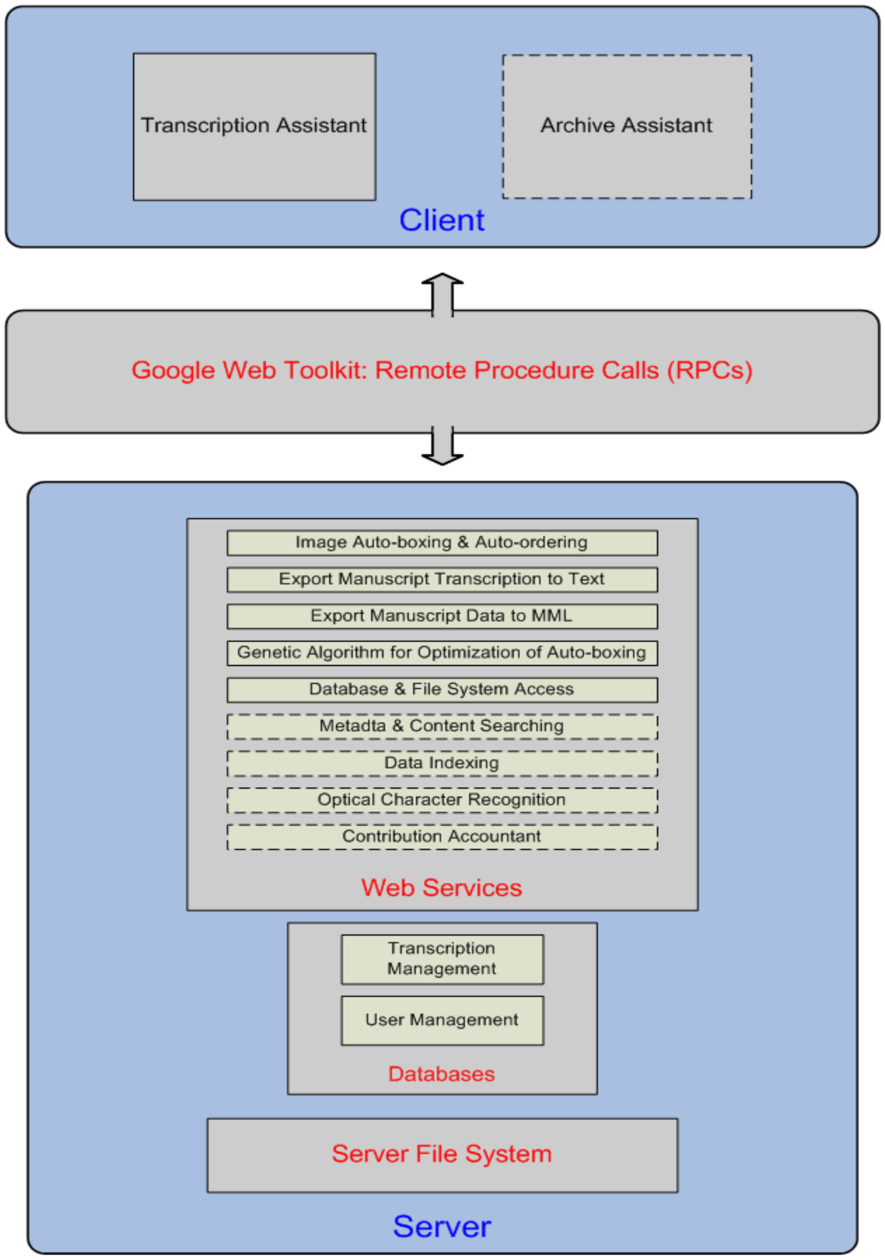
The major update that we determined was needed was the conversion of the software to an online system. Collaboration within the system is dependent on connectivity, and previous stand-alone applications never reached a point where collaboration was actually a possibility. There was also concern about the technical aptitude of our potential users. Having to download, install, or configure anything may pose a problem. In addition, public computers typically have restrictions on downloading and installing software. We determined that an online system, properly designed, could have the same user feel as a desktop application, without requiring any downloads or installations, and increase the feasibility for seamless collaboration.

The software redesign was an interesting experiment in restraint. We had an existing legacy code base that was fairly expansive, with working code and design decisions already made. It would have been tempting to start our new design with the assumption that we could reuse much of the previous code, but we realized that if we considered the legacy code a "starting point", we would compromise our design in order to facilitate easier coding. We knew that we needed to design from the ground up.

We decided to adopt a web-based service architecture for the new software. It incorporates three major components: a client-side graphical user interface, a set of server-side functionality, and a set of databases for centralized storage. The decision to move some of the functionality to server-side web services was made because of the heavier, processor-intensive Transcription Assistant functions such as auto-boxing, auto-ordering, running the genetic algorithm, and parsing data to/from the older MML data format. The legacy code base was largely adaptable into web services for the TA. The difficulty with extricating useful code from previous versions of the software was that it was largely tied into the specific implementation of the software. By removing unnecessary dependencies within the code, we were able to isolate and streamline the processes that we were interested in reusing. Rather than being intrinsically tied to a specific implementation, the services are now available through a simple interface from anywhere within our software. The Google Web Toolkit provides a Remote Procedure Call specification which allows this communication between server and client.

One modification was done to the auto-boxing mechanism when converting it to a service. Rather than boxing an entire page by default, we realized that it would likely be more accurate if the auto-boxer were given a particular set of boundaries (referred to as a "text field") to box. Doing so will eliminate extraneous boxes by reducing the amount of white space, marginalia, miscellaneous marks, unwanted text and images. It will also significantly improve the speed at which the auto-boxer runs, as it will minimize the number of pixels that are processed.

The Transcription Assistant maintains the feel of a desktop application because all of the communication between client and server is asynchronous. Typically when working in a browser, any update of information requires that an entire page is reloaded. Through GWT's RPC calls, the browser is able to pass and receive information and requests behind the scenes, eliminating the need to refresh the page.

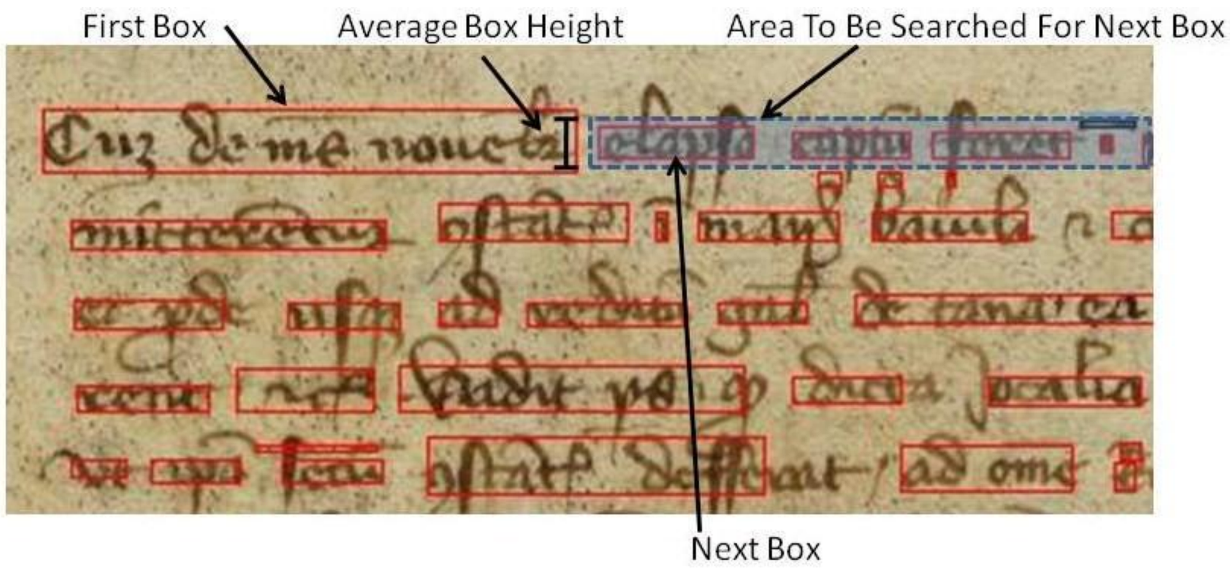

Transcription Assistant
Archive Assistant
Client
Google Web Toolkit: Remote Procedure Calls (RPCs)
Image Auto-boxing & Auto-ordering
Export Manuscript Transcription to Text
Export Manuscript Data to MML
Genetic Algorithm for Optimization of Auto-boxing
Database & File System Access
Metadta & Content Searching
Data Indexing
Optical Character Recognition
Contribution Accountant
Web Services
Transcription Management
User Management
Databases
Server File System
Server

3.1.1 New auto-ordering algorithm

The previous algorithm for determining where a line of text on a page lies was slow and inaccurate. Knowing where lines of text lie on a page is integral to the software, because it dictates the order in which the boxes are stored, and consequently, the order of the words. The previous algorithm, like the algorithm for auto-boxing, depended on an analysis of the image data. Rather than try to analyze the image, we suggested a new algorithm that depends on the location of boxes generated by the auto-boxing algorithm. In preliminary testing, this new algorithm is more accurate and significantly more efficient.

The new algorithm depends on a new piece of data that is very useful to analyze an image: the average height (in pixels) of the boxes generated by the auto-boxing algorithm. It may sound like a trivial piece of information, but it describes the average height of the text on the page. Because manuscripts have no uniform “font size”, and different scanning resolutions make words look different sizes to an image analyzer, this data gives us a better understanding of the relative size of the words on the page.

The ordering algorithm starts by searching the list of boxes on a page to find the top left-most box. As the boxes are stored by coordinates, it finds the box with the lowest x and y values (x,y). It assumes that this is logically the first word on the page. It removes that box from the “to search” list, and adds it as the first box in the ordered list. To determine which box is the next box, it searches the list of boxes for the box with the lowest **x** value with a **y-midpoint** value within the range of **y-midpoint** +/- half of the average box height.

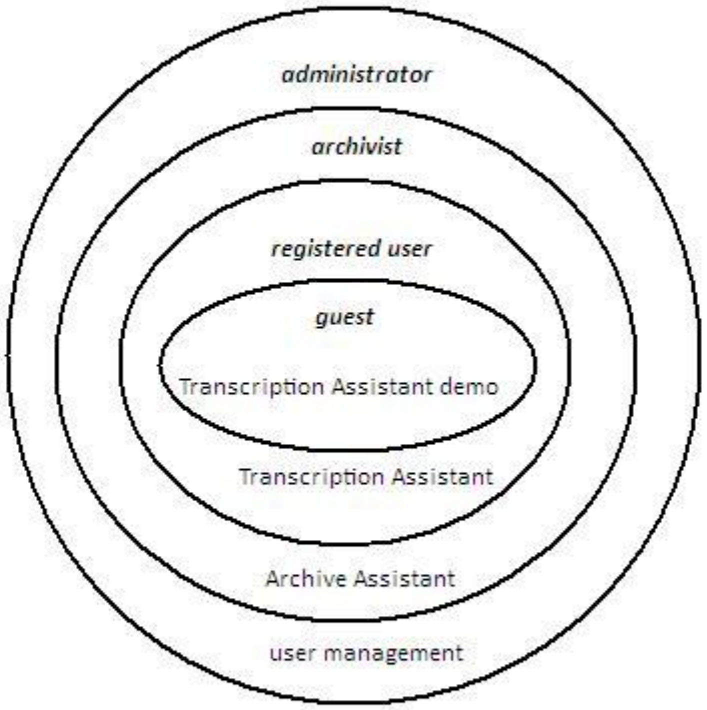


Logically, this finds the next word that appears to be on the same line as the previous word. Once again, it removes that box from the "to search" list, and appends it as the next box in the ordered list. The algorithm determines that it has found the end of a line when there is no box that meets the above criterion. It then finds a new top-left box, and repeats the steps. When the "to search" list is empty, the algorithm is finished.

3.2 Database Design

As is the case with most modern web applications, the Transcription Assistant is three-tiered. The lowest tier is the data tier and is comprised of two relational, mySQL databases. The first stores information related to the system's users and the second stores all the data that comprise the transcriptions themselves along with their metadata. Manuscript images are stored on an external file system on the server, with pointers to them stored in the transcriptions database.

The user database employs a simple, single-table design to store user data**:** login credentials, user access level (normal user, archivist, administrator), and the number of transcriptions they have worked on, among other details. Guests to our site have access to the demonstration version of the Transcription Assistant (seen on the uSCRIPT website on the *Transcribe* tab) to allow them to get a feel for what our system can offer. Registered users will have basic access to the Transcription Assistant so that they can save their work, therefore contributing to those emergent documents along with the ability to export their transcriptions for personal use. Archivists will additionally have access to the Archive Assistant, allowing them to upload and auto-box scanned manuscript images. System administrators have the ability to ban problematic users to ensure the operation of the system runs smoothly. Please note that the user access schema is set up such that each type of user has all of the access rights of all user levels lower than theirs:



The transcriptions database houses all of the data and metadata of each manuscript transcription, and has undergone multiple design changes over the course of our project.

Using the initial design, the transcription database contained only 12 tables. So all suggested words in the transcriptions in the system are stored in the same table, which becomes unreasonably large very quickly.

The second design dealt with the problem of very large tables, but it ended up overcompensating. Using this design, each box had its own tables for suggested words, votes for those words, suggested coordinates, and votes for those coordinates. The number of tables used to represent data for a single page of average length approaches one-thousand. Each table is very small. However, the number of tables becomes highly prohibitive in terms of executing database operations quickly enough to provide a smooth user experience from the perspective of the web interface.

In order to decrease the number of tables needed to represent pages in the transcription, the third design focuses on the page as a storage unit. Rather than having each box having its own table for suggested words, all suggested words on a given page are stored in the same table. With this setup each page is represented by twelve tables in the database. Each of these tables is fairly small, containing no more than a few hundred rows. For more details, see Appendix C.

After considering the above design, we found that there were still too many tables, so the focus was moved to the entire manuscript as a unit of storage. With this design, all suggested words in a given manuscript are listed in the same table, allowing a single manuscript to be represented by twelve tables.

In the end we came to the realization that the initial design, in which all transcription data and metadata in the system are stored in twelve tables, is the best for uSCRIPT as it minimizes the number of tables in the system. The fact that each table is quite large in size isn't a remarkable drawback because database tables by nature are meant to store enormous amounts of data.

While a manuscript is likely to be lacking a proper title, there has to be a way of uniquely identifying them in our system. One could argue that a reference code would serve as the unique identifier, but that is not an optimal solution. Firstly, references codes vary from archive to archive, so there's a chance that there are two manuscripts in our system with the same reference code. Also, reference codes tend to err on the unintelligible side in terms of end-users looking at them, as they were meant for internal archive use in keep the different manuscripts in order. Even if a title has to be contrived by an archivist based on a few other metadata fields (such as reference code and subject), it is our best bet for uniquely identifying documents in the uSCRIPT system.

3.2.1 Database Structure

The structure of the transcriptions database mirrors the Manuscript Markup Language (MML) file structure, which is an XML-based system devised by a previous project team to store transcription data. While MML files are no longer used for storing data in the uSCRIPT system, the same structure is used in the transcriptions database. Each time a transcription was opened in the Transcription Assistant its MML file would be sent to the client machine and its data loaded in the application. When a user was finished with a document they would save it, sending the updated MML file back to the server. The sending of MML back and forth is both unwieldy and time consuming, and the AJAX model compensates for MML's drawbacks. The vision for the web-based Transcription Assistant built on the framework we've provided only transmits data to the client machine from the server and vice versa exactly when it is needed. For example, when a user types a word into a box, that word will immediately be sent to its proper place in the transcriptions database. The end result is that a user's transcriptions will be saved automatically as they work, eliminating the need both to wait for all of the updates to be saved at the end of a session and developers needing to worry about data being lost if a user is disconnected in the middle of a session.

Each transcription contains a set of metadata, including the title, author, and subject matter. Due to the age of many of these manuscripts, many pieces of metadata may be unavailable, so the archivist who is uploading a manuscript will simply enter as many pieces of metadata as they can. As for the structure of transcription data itself, each document contains a series of one or more pages, which are labeled with a number for recordkeeping purposes, even if the printed page numbers on the manuscript don't match up exactly. Each page contains a series of one or more page elements, which can represent a number of different things, but are most commonly either a paragraph of text or an image. If a page element is a text paragraph, it contains a series of one or more boxes. Boxes are a specific element that are drawn around words, and are comprised of a number of elements:

- a list of suggested transcriptions for that word
- a list of votes for the suggested transcriptions for that word
- a list of suggested sets of coordinates for the location of the box

- a list of votes for the sets of coordinates for the location of the box
- the text style of the word in the box

Users are allowed to suggest new locations for the boxes because it makes documents more emergent, and it recognizes the reality that archivists are likely not going to have the time to tweak the locations of boxes on a manuscript they've just auto-boxed. They're probably going to consider the boxing good enough for transcription to begin, upload the boxed image, and move on to the next image they want to upload. However, there's a much better chance that transcribers will, from time to time, take a moment to suggest a new set of coordinates for a box in order to improve the clarity of boxes.

Traditionally, a scripting language such as Perl or PHP is used to allow the front-end of a web application to communicate with and manipulate databases. However, in our three-tiered design we have a middle tier running on the server side in the form of a Java application. To fit into this system we opted to make use of Java's Database Communications (JDBC) library . This allows us to format and process the raw data being received from one of the databases before sending it to the TA web interface on the client machine, thereby minimizing the amount of data that needs to be sent between the client and server machines.

3.2.2 Indexing

One method that we looked into using for the purpose of improving database performance is the use of indexes. Indexes are typically used on larger tables to decrease the amount of records a database has to sift through in order to formulate the result of a query. However, along with their potential for significant speed increases, indexes actually slow down processing on queries that modify fields in a database, rather than simply reading fields. The most common database operation in our system, unfortunately, is a modification resulting from a user typing a suggested word on a transcription page. Because our databases will be serving more update (write) queries than read queries, the use of indexes in our transcription database would decrease overall query processing speed.

3.3 Front End Design

The front-end component of our application represents a very important aspect of our overall project. As the primary end-user product, the front-end is responsible for the usability of the entire system. Since our application is not being designed for other computer scientists, and is not being programmed only as a proof-of-concept, we must take special care to create a program with an intuitive interface that allows users to work effectively, and to work efficiently. We constructed an interface that largely resembles desktop applications while running within a web browser, taking into consideration and developing elements such as toolbars, work areas, and an infrastructure for multi-language support,

3.3.1 AJAX

One of our early decisions was to implement our online system using AJAX, which stands for Asynchronous JavaScript and XML. As its name implies, AJAX is not a language or platform, but a system of techniques for developing rich Internet applications (RIAs). The primary element to AJAX applications is the use of the XMLHttpRequest object or IFrame objects. These allows web browsers to make dynamic data requests without having to reload the entire page and application, making data exchange much faster and easier to handle. The second commonly found trait in AJAX applications is the use of XHTML and CSS for the styling of elements. While this is not particularly uncommon, it is noteworthy for aiding the speed with which elements can be developed, as XHTML and CSS are both common, easily learned and quick to implement. Next, the interface and Document Object Model (DOM) calls of an AJAX application are typically coded in JavaScript or a related scripting language. Lastly, AJAX applications most commonly use XML for the file format of exchanged data, but HTML, JavaScript Object Notation (JSON), or plain text are all also commonly used. The culmination of these four steps is an elegant method for putting together a browser-based application with a well-balanced client-server relationship and large potential for functionality and usability. That said, AJAX is not the only technique for creating rich Internet applications.

The specific reasons we chose AJAX for our system center around our user requirements. Our target audience, historians and historical scholars, are by no means guaranteed to be particularly tech savvy, or to have extensive computer resources. We therefore need our application to make as few demands upon them as possible. Our system would need to be easily accessible, intuitive to use, and light on system requirements, AJAX was the simplest solution which addressed all of these issues. First, all a user would need to access an application in AJAX is a connection to the Internet, and a web browser with JavaScript enabled. With other technologies, such as desktop applications, installation might become a hindrance, as the users, especially those who may work off of computers owned by a company or university, might not have the appropriate permissions. Another benefit is that all system upgrades or updates would be automatic, requiring no patches or re-installation. With regard to usability, AJAX applications offer essentially the same functionality as would a desktop application. While development of some elements might be faster in Visual Basic, for example, as was the original 1998 Emergent Transcriptions project, the same features are nevertheless available with AJAX, with the added benefits of an online application. Finally, because much of the work of the application is done locally by the browser, with major functions being run by the server through remote procedure calls (RPCs), the application is not intensive from either a processing or networking standpoint.

There are, however, some weaknesses to using AJAX. For one, our application runs the risk of having latency problems when loading large amounts of data from the database is loaded. Also, since the application is typically coded in JavaScript, problems can arise where different browsers use different standards for interpreting JavaScript.

3.3.2 Google Web Toolkit (GWT)

After deciding upon AJAX as a method of implementation for our application, w looked into various available technologies and libraries available for creating JavaScript front-ends, and eventually decided upon the Google Web Toolkit (GWT). GWT launched version 1.0 on May 17, 2006 to much fanfare and has since received much praise from a variety of sources on web development. Its primary feature is its Java-to-JavaScript Compiler, which allows a developer to write their application in Java design their application according to standard object oriented programming conventions. GWT also features a large class library of widgets (ie, interface elements), that can be used to build interfaces in a similar fashion to other common Java widget toolkits like SWING or AWT. Since the JavaScript produced is essentially an object code of sorts, it is not only more compact than most JavaScript applications, but much more difficult to read through; the latter property means that using GWT naturally creates a low-level security for the proprietary work in the application. Finally, GWT offers easy implementation of RPCs, speeding up time spent connecting the back- and front-ends of our application.

In addition to these features, we also chose GWT for some of its other strengths. As mentioned before, JavaScript and related ECMAScript scripting languages are inconsistently implemented by different browsers. As a result, developers can spend considerable time providing checks and fixes to applications to correct cross-browser problems. GWT has a good reputation for dealing with this problem, and, although we later ran into a problem with cross-browser support, this aspect was nevertheless a big selling point for us. Another feature of GWT is support for internationalization, wherein the text in the application may appear in multiple languages for users in different parts of the world. Since our application is initially being developed for use by an Italian archive, with an eye to expand to historical archives all over the world, multilingual support is an imperative feature for our usability. Essentially, GWT offers more rapid development in a familiar language and format as well as support for features that would be much more time-consuming to implement otherwise.

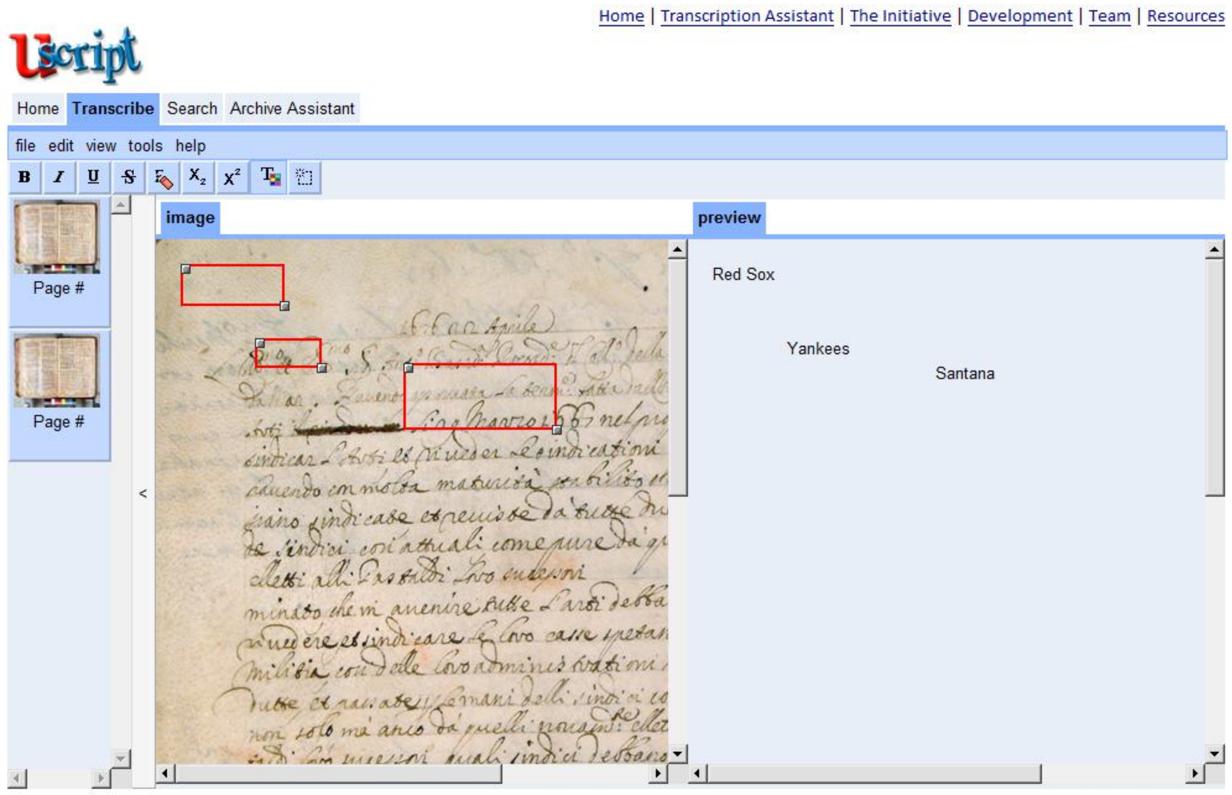
However, there are a couple of caveats to developing in GWT. In our experience, there are sometimes inconsistencies between the documentation and the functionality of widgets. This can be frustrating and can lead to many redesigns and work-arounds. Similarly, the widget library is still somewhat limited, as are the currently implemented widgets. As an example, the AbsolutePanel class is the only panel which can position widgets placed on top of it absolutely (according to a set pixel coordinate value). However if you want those widgets to be able to be focusable, you'll need to wrap them in a FocusPanel, the only panel which brings focus to the widgets it carries. This led to considerable layering of panels in our application, and will probably continue to do so in later iterations. Thankfully, these problems are very likely to be addressed by Google developers as GWT comes into maturity.

3.3.3 Interface design

When we began designing the user interface for our new application, we began by looking at previous iterations. We used a generally similar layout to the Java application that we worked with at the beginning. The primary feature of this layout is the use of two large windows in the primary workspace, in our version called the Image View and the Text View, from which the user can view the manuscript and a preview of the current transcription respectively.

The Image View comprises not only the scanned image of the current manuscript, but also the boxes in which transcription suggestions are stored, superimposed over the words they represent. The Image View is currently housed within a tabbed panel (TabPanel) with the idea being that in future iterations other views of a manuscript (such as under ultraviolet light) will be available.

The Text View is less directly interactive with the user, but is still an important component of the workspace. Since the boxes are superimposed over words on the manuscript, it would be problematic to place the transcription suggestion inside the box itself, since it would render both versions of the word illegible. Therefore, a second view in which only the digital text appears is necessary. Words in the Text View appear in the same place on the field as they do in the Image View, and the two views' scrollbars are locked together so that they will move together.


Home | Transcription Assistant | The Initiative | Development | Team | Resources
Uscript
Home
Transcribe
Search
Archive Assistant
file edit view tools help
Page #
Page #
image
preview
Red Sox
Yankees
Santana

The last major component to the workspace, which is new to our version, is the Preview Panel. This panel resides on the left hand side of the workspace, and is roughly similar to preview panels in more common programs like Microsoft PowerPoint or Adobe Acrobat. When a user has multiple transcription pages open, each will appear as a selectable button on the Preview Panel. Each button will feature a title and thumbnail view of the page.

Besides the layout of the application, we also needed to design the ways in which the user would input data into and interact with the transcription suggestion boxes. Moving and resizing the boxes is controlled by the use of two handles, located at the top-left and bottom-right of a box respectively. We had initially planned to make the sides of the boxes themselves editable, but ran into problems with resizing in the left and up directions. In order to enter text, the user simply clicks on the box, which prompts a pop-up box into which the user enters a word. It is important to note that the box does not appear directly over top of the box, because that would obscure the view of the word being transcribed. In order to submit a typed suggestion, the user may either press the spacebar to enter the suggestion and advance to the next box, or press the return key to enter the suggestion without moving their position. The pop-up's text field is itself a suggestion box (SuggestBox), meaning that, as the user types, previously entered suggestions that match what has been typed so far will appear below the pop-up. These words can be selected with either the mouse or the keyboard. Finally, on the right side of the pop-up is a button which expands the pop-up to a now mostly deprecated view. This secondary pop-up is a stack panel containing a drop-down for previously entered suggestions, a text field for entering new suggestions, a similar drop-down for coordinates, and a button to begin editing box coordinates under the initial system involving dragging box edges.

Besides the Transcription Assistant we worked on creating the preliminary layout for the Archive Assistant. The Archive Assistant, the tool which archivists use to import images into the database, is divided into two main sections. On the left is a file upload panel where the user can create manuscripts and pages and place them onto a tree structure before submission. On the right it contains a tabbed panel which contains a section for entering meta-data and a section for editing boxing in the file.

4 Recommendations and Conclusions

A major goal of this project is to provide future groups with a comprehensive view of the status of the project, where we envision the initiative progressing from here, and how we imagine that might be done.

4.1 Front-End Functionality

4.1.1 Adding a Throbber/Loading Bar

In the current implementation of the Transcription Assistant, there is no way to monitor the status of any ongoing loading or processing. As a result, users waiting for the program to load a large element or finish an intensive task are left without any indication that the application is still in fact functioning. This is addressed in browsers with elements like loading bars and throbbers. Throbbers are icons located typically in the top-right corner of a browser's screen which run a simple animation while the browser is busy. This is a simple method for reassuring the user that all is well and prevent them from mistakenly closing out the application. While our current state of implementation does not necessitate a throbber or a loading bar, almost any extension of functionality would be greatly aided by the presence of such a device.

4.1.2 Browser History Support

One of GWT's commonly touted features is its browser history support. Since the browser history is tied to interpreted HTML, JavaScript developers often do not or are unable to utilize it. That means that when users move between pages in such applications, they cannot rely upon the browser's built-in navigation tools. GWT allows the creation of new history items that can be placed on a browser's history stack . While this is not a core feature, it does not appear prohibitively difficult to implement compared to its benefits in usability.

4.1.3 User Projects

Realistically, users are not going to simply look through one or two documents for the whole of their research. Instead, they are going to collect lists of documents that they reference and it would be very useful for the Transcription Assistant to facilitate that. The idea of a project, then, would be to maintain a collection of links to relevant files. Since the project file is simply a list of references, it could easily be exportable to and importable from local files, and could easily be used by multiple users.

4.1.4 Splitting/Merging Boxes

There is a problematic case in emergent transcription where there is disagreement over whether a section of text is one or multiple words. Worse still, such contested sections may overlap and be interwoven. Our proposed solution would be to create tools for splitting and merging boxes. These splits and merges would themselves need to be suggestions, and as such, would be stored separately from the individual suggestions. Essentially a section that was originally marked as two words could be merged together, creating a third suggestion-storing object for when the boxes were viewed as one. The user would then need some visual indication in order to switch between the two views of the wording. Splitting would work similarly in the other direction.

4.1.5 Expanding Support for Other Languages and Other Archives

Currently, our project only has a language entry present for English, and has no specialization for different archives. Since we would like to see this project implemented in various countries by several different archives, it will be important for future groups to spend time working on implementing new language support and making meta-data friendly to differing formats. The current application contains the groundwork for implementing internationalization. Also thankfully, most archives primarily use a reference number which contains all of the information they need, but changes such as specialized forms must nevertheless be anticipated.

4.1.6 Straight-to-text

A straight-to-text tool would be fairly simple to write, and is very necessary in the next release of the software. This tool would take the top ranked suggestion for each word (or the word that the current user has chosen) and output the words in sequence (as defined by the user with the help of the auto-ordering algorithm). The output could go to an output file to be saved locally, or to a secondary “Preview” tab.

4.2 Website Expansion

4.2.1 User Forums

Another useful addition to the uSCRIPT system that a future project work should implement would be a set of user forums. These would allow interaction between users beyond simply assisting one another in transcribing manuscripts in the way of offering suggested transcriptions and giving an up-or-down vote on existing suggestions. Users would be able to have more open discussions with one another about transcribing, the uSCRIPT system in general. Along with providing for a much stronger sense of community and cooperation, user forums could serve as a mechanism with which to collect feedback about the usability of our system and suggestions for improvement.

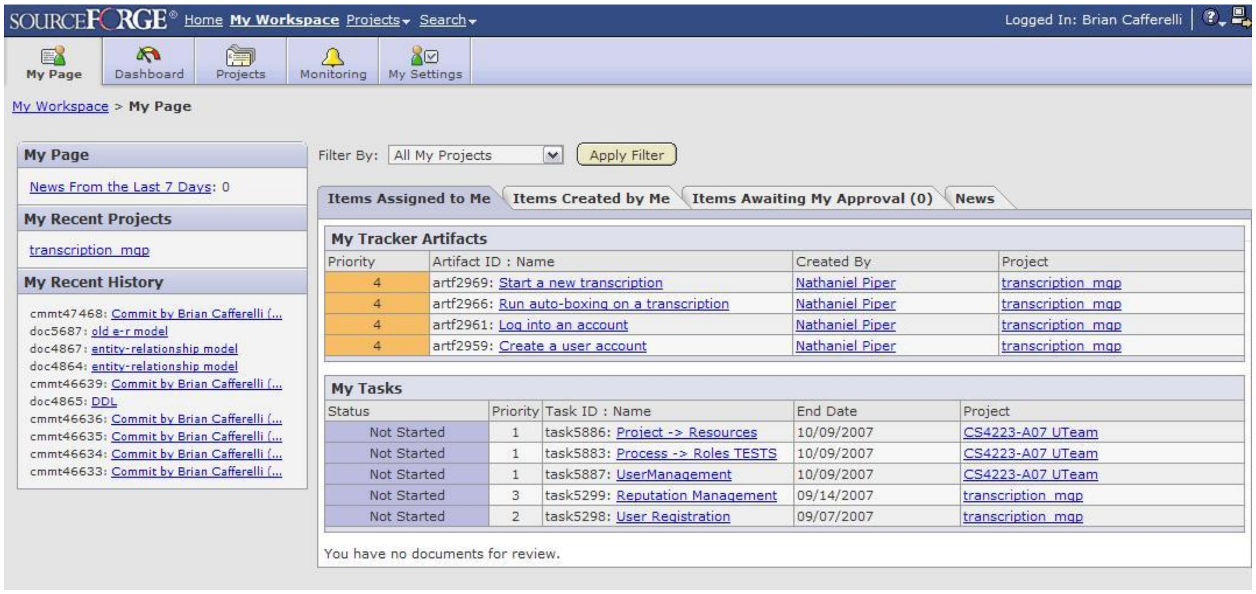
4.2.2 User Management

Another feature that would be nice to see in future iterations of the TA is user management. The possibility of malicious users working within the system should be recognized as a potential problem and some protection scheme put in place to combat behavior that is clearly subversive in nature. One simple way this could be accomplished is by allowing users to report other users who they suspect are causing trouble.

Reputation and credibility management also present an opportunity both to improve the usability of the Transcription Assistant and help offset the efforts of malicious users. While a simple reputation system exists in the current system, a future team should take the time to devise a more sophisticated scheme.

4.2.3 User pages

Many well-established, successful websites have user pages that act as a sort of customized headquarters to make browsing that site easier for each user. A user homepage on the uSCRIPT website should list transcriptions they have worked on recently, transcriptions that are similar in content to manuscripts that user has worked on, and similar transcriptions listed by era or location. Below you can see a good example of a user page laid out neatly at sourceforge.net:



My Page

News From the Last 7 Days: 0

My Recent Projects

transcription_mqp

My Recent History

cmmt47468: Commit by Brian Cafferelli (...
doc5687: old e-r model
doc4867: entity-relationship model
doc4864: entity-relationship model
cmmt46639: Commit by Brian Cafferelli (...
doc4865: DDL
cmmt46636: Commit by Brian Cafferelli (...
cmmt46635: Commit by Brian Cafferelli (...
cmmt46634: Commit by Brian Cafferelli (...
cmmt46633: Commit by Brian Cafferelli (...

My Tracker Artifacts

Priority	Artifact ID : Name	Created By	Project
4	artf2969: Start a new transcription	Nathaniel Piper	transcription_mqp
4	artf2966: Run auto-boxing on a transcription	Nathaniel Piper	transcription_mqp
4	artf2961: Log into an account	Nathaniel Piper	transcription_mqp
4	artf2959: Create a user account	Nathaniel Piper	transcription_mqp

My Tasks

Status	Priority	Task ID : Name	End Date	Project
Not Started	1	task5886: Project -> Resources	10/09/2007	CS4223-A07 UTeam
Not Started	1	task5883: Process -> Roles TESTS	10/09/2007	CS4223-A07 UTeam
Not Started	1	task5887: UserManagement	10/09/2007	CS4223-A07 UTeam
Not Started	3	task5299: Reputation Management	09/14/2007	transcription_mqp
Not Started	2	task5298: User Registration	09/07/2007	transcription_mqp

You have no documents for review.

4.3 Overall System Improvements

4.3.1 Transcription Searching

One area of the system that we'd like to see built upon in future projects is transcription searching. As searching in a web application is analogous to opening a file in a traditional desktop application, powerful searching capabilities make it easier for historians and archivists to use our system. The current uSCRIPT system provides the framework for taking a set of metadata obtained from the Venice State Archive and allowing users to search for existing transcriptions with those parameters.

Ideally, users will be able to seek out transcriptions on a given topic by searching the body of text in a transcription in addition to external metadata. This will increase the amount of related material users are able to discover to further their research.

4.3.2 Enhanced Optical Character Recognition

Enhanced optical character recognition has long been a goal of the initiative. As it currently stands, the image is analyzed in order to figure out where individual words lie within the page (auto-boxing). This is commonly the first step taken in modern OCR algorithms. We would like to see this taken a step further into analyzing the content of the boxes that are generated. (As a side note, success of this step would be dependent on a reliable auto-boxing of the image. Please see the section on algorithm optimization for more on this.) The method for analysis is yet to be researched, but some ideas have been proposed. Typical methods are unlikely to succeed in for this project, as the intricacy and variability of the text varies so widely. It wouldn't be appropriate to compare the word images to any standard set (or sets) of data in order to discover individual letters and proceed to build words. A more feasible option might be to utilize previously transcribed words, paired with some sort of quantitative analysis of the digital image. If, once the words on a page were boxed, the image data within each box could be extricated and stored, individual words could be compared to each other. If patterns matched, the system could use the user recommended transcriptions for one word in order to make suggestions for the other unknown word. There would be an interesting algorithmic problem to solve in how to analyze, quantify, and compare each word image. It would be extremely processor intensive to compare, pixel by pixel, one image to another, and would also be very difficult to determine whether it was a valid "match". One possible solution that has been mentioned is the possibility of quantifying the pen strokes within a word image to make for easier pattern matching. Other possible optimizations of this algorithm might take into account the metadata for the manuscript that a word is taken from, such as time period, author, or language. Clearly a match is more likely with word images that are known to be from the same author.

4.3.3 Importing Options

Expanding options for importing a manuscript into the system through the Archive Assistant is a necessary improvement. When importing a manuscript, the archivist should be required to enter a specific set of metadata about the manuscript. A preliminary boxing of each page of the manuscript should also be required. There should be some options presented to the archivist about how the documents should be boxed. Some options might include manually boxing the page(s), manually setting threshold values for the auto-boxing algorithm, displaying some sample manuscript images with preset threshold values for similar manuscripts, or the option to manually box a single page of a multi-page manuscript in order to have the genetic algorithm find optimal threshold values for the rest of the pages.

4.3.4 Business Plan

One of the long-term goals stated early on in the initiative was the desire for some sort of incentive program for system users. This could include but is not limited to monetary credit for providing transcriptions, to be funded by yearly service subscription fees. This could also be pushed along by providing higher credit to users with higher transcription reputations, and by charging credit for access for advanced features such as an optical character recognition algorithm or for access to view a transcription.

4.3.5 Algorithm Optimization

Most of the server-side processes have somewhat sophisticated algorithms, and most of them can and should be optimized to run more efficiently. The auto-boxing algorithm uses a smearing algorithm that may not be the most accurate choice. The auto-lining algorithm could be written to run more efficiently. The genetic algorithm was written to run infinitely, or until halted by a user. Realistically, it needs to have some way to decide when it has found a satisfactory solution. The inner workings of the genetic algorithm and auto-boxing are discussed in depth in previous reports, and the current auto-ordering algorithm is explained within the code.

4.4 Conclusion

As this has been an ongoing project for quite some time now, one might assume that the final software would be more sophisticated and fully functional than it is in its current state. Based on our research and analysis of past projects, we determined that the overarching reasons for lack of progress were that there was a lack of clear direction, guidance, and documentation on the part of the group members when passing the project from one group to the next and too often, emphasis was placed on less than integral components of the system. Thus, much time was spent trying to understand the background and existing work, and it typically resulted in an even more convoluted problem for the next team.

Our initial goal was to solve this problem by providing a system built on the appropriate platform (the web) that is particularly easy to understand, easily extensible, and practical for the problem. In addition, another goal was to provide clear and concise guidance and recommendations for future teams, through comments within the code, and sections in this report. We were also able to provide a central hub for the initiative, through the uscript.org web site. It provides a comprehensive background of the initiative, as well as a look into the current and future work. This is a valuable resource for community members interested in the ongoing project as well as for future groups. Although we were not able to expand the functionality of the system as far as we would have liked to, we have provided a valuable service to the Emergent Transcriptions Initiative. We believe that our work toward this project will enable the initiative to grow further and faster than it has in the past, and soon to reach its eventual goal of common use throughout the archive and transcription community.

5 Bibliography

Carrera, Fabio. (2005). "Making History: an Emergent System for the Systematic Accrual of Transcriptions of Historic Manuscripts" in proceedings of IEEE's **8th International Conference on Document Analysis and Recognition**. Seoul, South Korea: August 29-September 1, 2005.

Finch, Nathan. "Venetian Transcription Software," 1999. WPI Major Qualifying Projects Collection.

Fritas, Paul and Scott Glajch. "Web-Based Emergent Manuscript Transcriptions," 2007. WPI Major Qualifying Projects Collection.

Ho, Oliver, Chirag Patel, Ravi Patel, Ricardo Kligman. "Manuscript Transcription Assistant", 2003. WPI Major Qualifying Projects Collection.

Rath, Toni M., R. Manmatha, Victor Lavrenko. "A Search Engine for Historical Manuscripts." Published by the ACM, 2004. Retrieved 4/25/08 from http://ciir.cs.umass.edu/pubfiles/mm-341.pdf

Tibbo, Helen R. "Primarily History: Historians and the Search for Primary Source Materials". *International Conference on Digital Libraries*. 2002. pp. 1-10.

6 Appendix A: Uscript Codebase Access

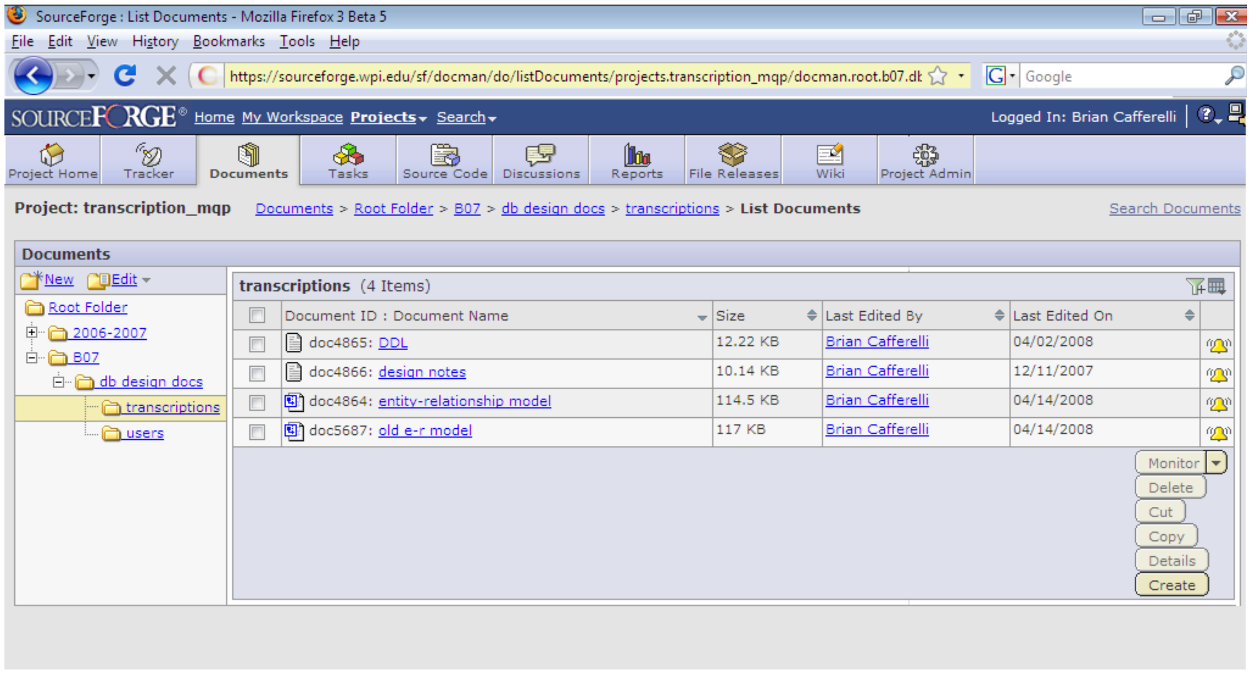
https://sourceforge.wpi.edu/sf/projects/transcription_mqp

The Codebase for our application is currently stored in a subversion repository on sourceforge.wpi.edu at the above URL. The current iteration of source code may be found in the directory “emergent_transcriptions” in the source code section of the Sourceforge project. The “transcription_mqp” directory next to it houses deprecated versions of the software at various different stages. The code may be checked out from the repository using the following command:

```
“svn checkout --username [USERNAME]
https://sourceforge.wpi.edu/svn/repos/emergent_transcriptions”
```

7 Appendix B: Database Documentation Access

https://sourceforge.wpi.edu/sf/docman/do/listDocuments/projects.transcription_mqp/docman.root.b07.db_design_docs.transcriptions


SourceForge : List Documents - Mozilla Firefox 3 Beta 5
File Edit View History Bookmarks Tools Help
https://sourceforge.wpi.edu/sf/docman/do/listDocuments/projects.transcription_mqp/docman.root.b07.d
Home My Workspace Projects Search
Logged In: Brian Cafferelli
Project Home Tracker Documents Tasks Source Code Discussions Reports File Releases Wiki Project Admin
Search Documents
New Edit
Root Folder
2006-2007
B07
db design docs
transcriptions
users
Monitor Delete Cut Copy Details Create

Project: transcription_mqp   Documents > Root Folder > B07 > db design docs > transcriptions > List Documents

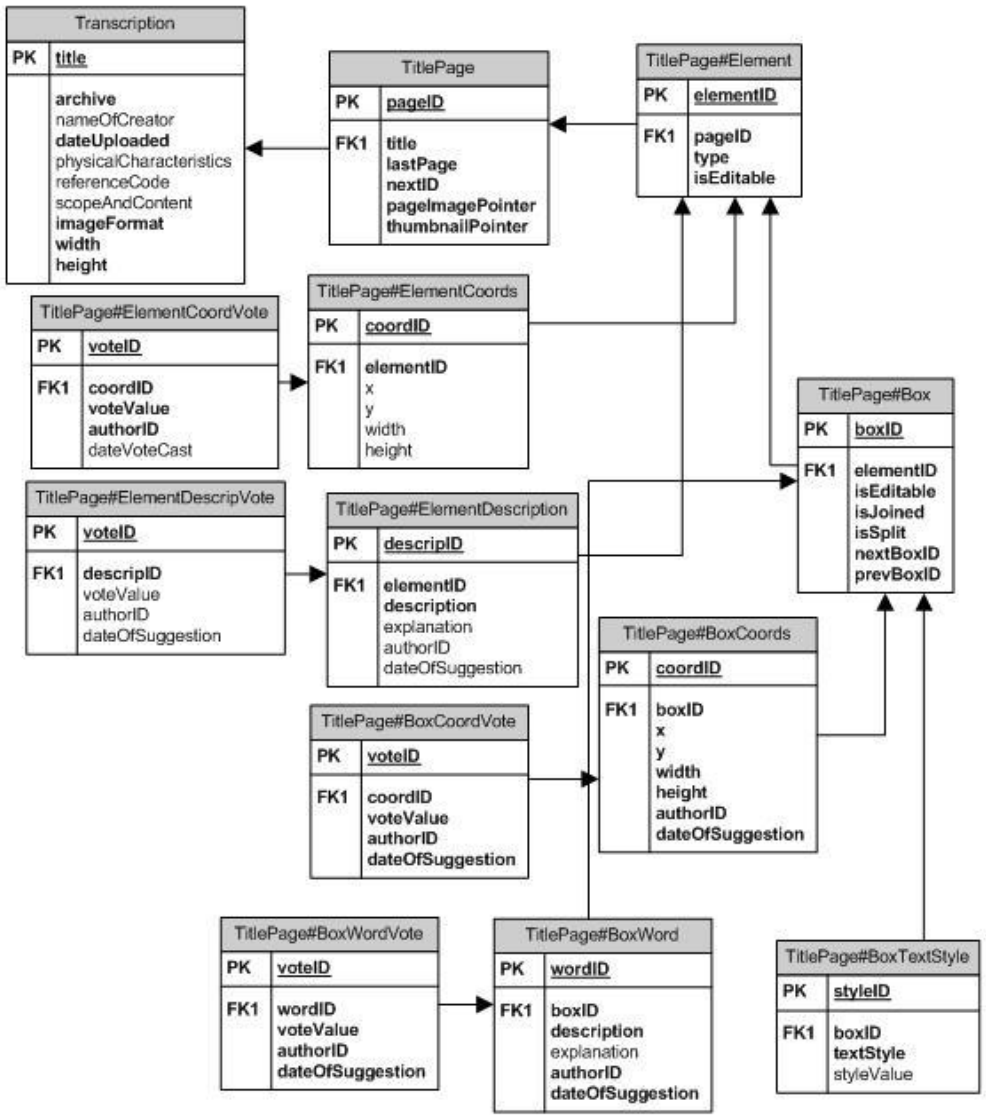
Documents

transcriptions (4 Items)

Document ID : Document Name	Size	Last Edited By	Last Edited On
doc4865: DDL	12.22 KB	Brian Cafferelli	04/02/2008
doc4866: design notes	10.14 KB	Brian Cafferelli	12/11/2007
doc4864: entity-relationship model	114.5 KB	Brian Cafferelli	04/14/2008
doc5687: old e-r model	117 KB	Brian Cafferelli	04/14/2008

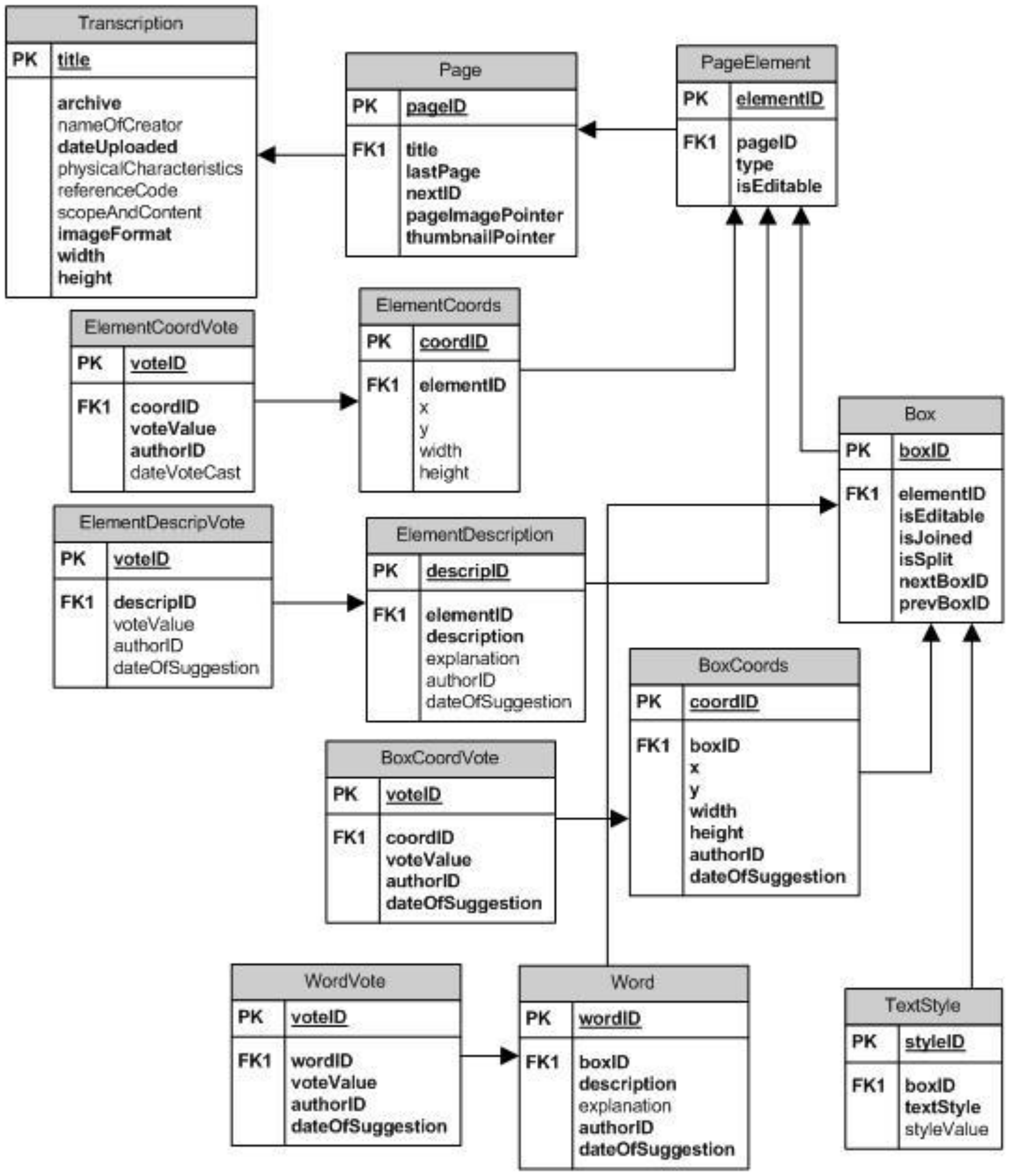
Future project teams can find SQL Database Definition Language files, entity-relationship diagrams, and field description files for both the transcription and user databases on the *transcription_mqp* project on on sourceforge.wpi.edu. After logging in, one can navigate to the Documents tab and open B07 > db design docs on the left panel to view the files.

8 Appendix C: Proposed Transcription Database Structure



Please note that the above structure, with the exception of the Transcription table, displays the tables that are needed under this model for each page in a transcription. For example, page 1 of a manuscript called "Police Report 456" would use tables called *PoliceReport456Page1*, *PoliceReport456Page1Element*, *PoliceReport456Page1Box*, etc. The fact that table names needed to be generated was a factor in simplifying the structure.

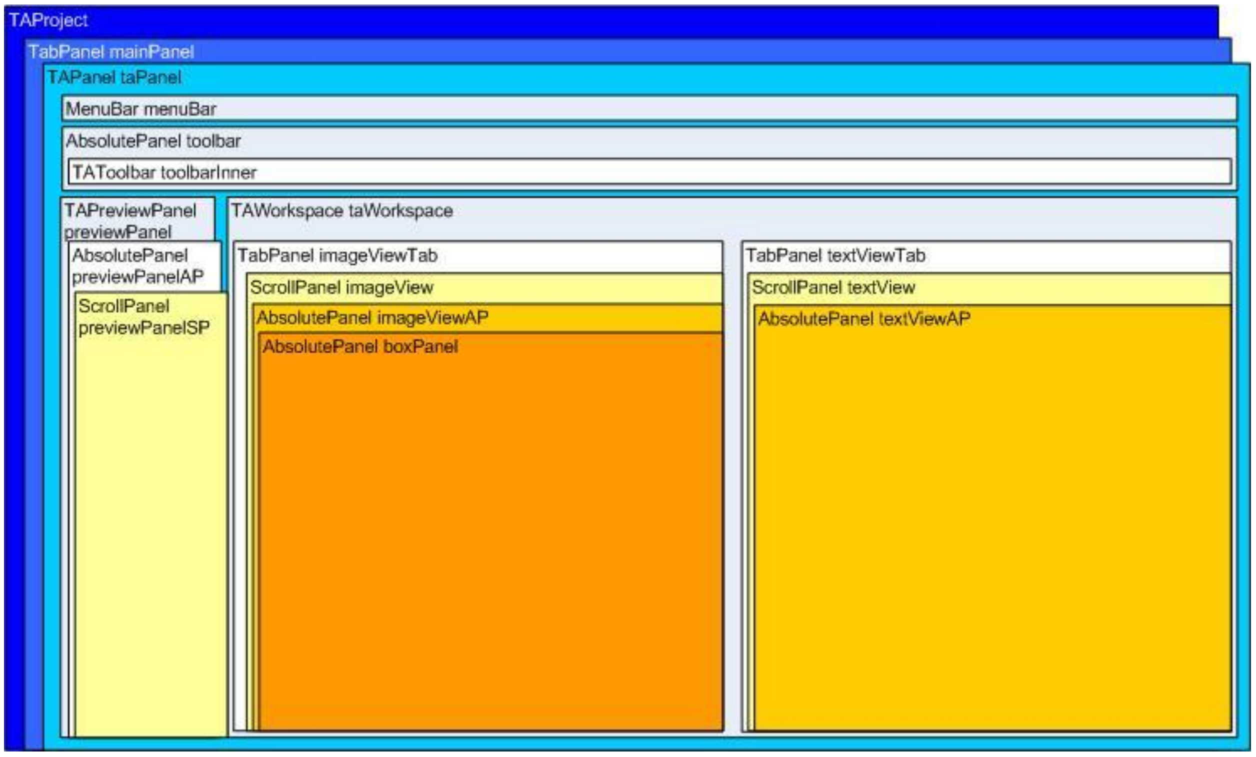
9 Appendix D: Final Transcription Database Structure



Unlike the diagram in Appendix C, the above diagram shows all tables in the entire database.

10 Appendix E: Front-End Panel Structure

The following diagram shows the skeletal hierarchy of the main panels in the Transcription Assistant layout. Each box represents an instance of some type of panel, which in GWT means a container widget on top which other widgets may be placed. A panel can be seen as a sort of platform, serving no purpose than to provide a context for other widgets. Each panel which appears “within” another was added as a widget to the second panel. This results in situations where panels can be stacked upon stacks of other panels in order to form composite panels with more complex behaviors than that of predefined panel classes. It should also be noted that the following types of panels in the diagram were defined as part of this project: TAPanel (extends DockPanel), TAToolbar (extends AbsolutePanel), and TAPreviewPanel (extends AbsolutePanel).


TAProject
TabPanel mainPanel
TAPanel taPanel
MenuBar menuBar
AbsolutePanel toolbar
TAToolbar toolbarInner
TAPreviewPanel previewPanel
AbsolutePanel previewPanelAP
ScrollPanel previewPanelSP
TAWorkspace taWorkspace
TabPanel imageViewTab
ScrollPanel imageView
AbsolutePanel imageViewAP
AbsolutePanel boxPanel
TabPanel textViewTab
ScrollPanel textView
AbsolutePanel textViewAP