Anoto Medical Image Annotator: Final Report

1. Member Names and Roles

Group Name: The Annototators

Anirudh Vemprala:	Code: Auto-save text commenting, thumbnail browser redesign.
Edward Karuna:	Code: Image export code
	Deliverables: Poster
Gene Zhang:	Code: Image resizing, stroke storage, code consolidation.
Robert Held:	Code: Paper interface redesign
	Deliverables: Wrote report and presentation.

2. Problem and Solution Overview

Radiologists and medical imaging researchers work in a field that demands constant, careful evaluation of complex images. The ability to accurately evaluate a medical image and detect important features, whether they are broken bones or metastasized tumors, is tantamount to success in radiology. Therefore, it is important for radiologists and researchers to have tools that complement their abilities by making it easy to record observations and share them with others. Currently, radiologists have limited on-screen tools to select regions of interest and share their observations with colleagues. Most people review medical images by opening the files on their respective computers, making personal notes, and then presenting their observations in a group setting. The solution to the current, inefficient system is an integrated approach that allows one to attach comments to a medical image and permit others to add their own observations. The spirit of collaboration could also be extended to include a student-teacher relationship. That is, students could indicate what they believe to be important features within an image, and then the teacher could review the students' work, indicate missed features, and provide tips for future analyses. Along the same lines, if the tool is quick and easy to use, it could be used in the classroom to facilitate audience participation. The solution should also include the ability to create the observations directly on a printout, allowing the user to use a pen to circle features and jot notes and avoid the relatively clumsy use of a mouse for the same actions. The *anoto* digital pen and its range of abilities are well-suited for such a solution. We have named our solution the "*Anoto* Medical Imaging Annotator (AMIA)".

3. Tasks

The following pilot study tasks outline the key functionality provided by our system.

Task 1 (Easy): Login and load previous annotations for viewing.

In order to facilitate the sharing features of the program, users will need to be uniquely identified. The user login will also be crucial for the teacher-student relationship, since students will not be able to view their peers' reviews of a given image if the program is being used to evaluate their performance. The teacher, on the other hand, would be given access to everyone's files in order to deliver his/her comments.
When the program loads up, a sign-in dialog box will appear and ask the user for his/her screen name and password. The task then demands that the user load a previous set of annotations, an action which will be necessary in order to add comments to other users' notes for a given set of medical images. The task should be a straightforward process.

Task 2 (Medium): Add text comments to another person's annotations

To collaborate on the review of patient's medical images, multiple users must be able to comment on the same images. The program includes a text box for providing additional comments on top of those already recorded using the *anoto* pen. The task involves selecting an image with pre-existing comments, selecting the text box, and adding a few lines of text. In the early prototype of the interface, a "Save Comment" button needed to be pressed to store the comments. After user feedback, we made the save function automatic.

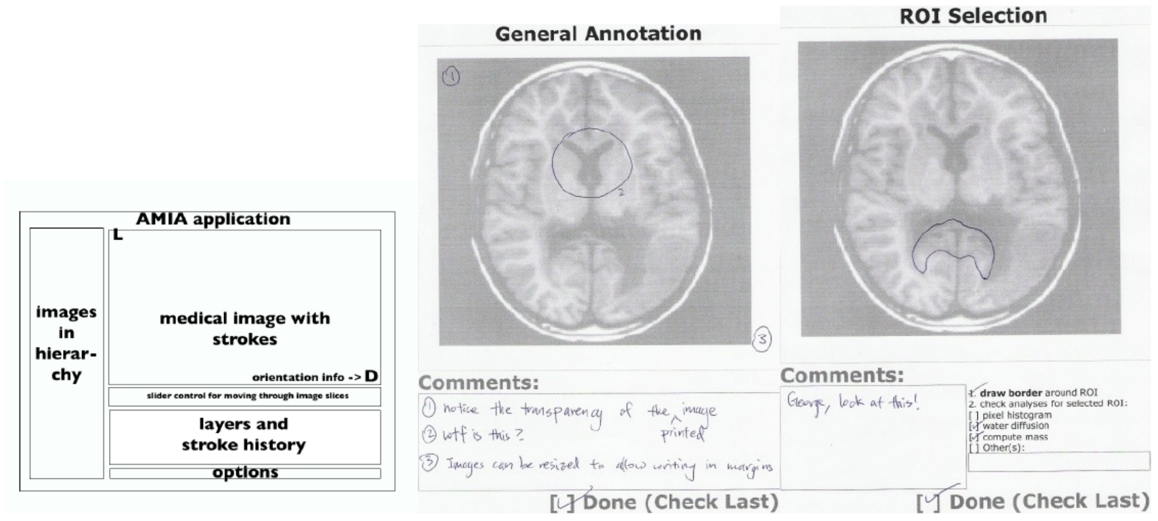
Task 3 (Hard): Print and annotate a set of image.

The core functionality of AMIA lies in its use of the *anoto* digital pen. Therefore, it was crucial to include a task that required the user to print out a set of medical images, add comments, and register those comments with the program. The task is accomplished by reviewing a medical image set, clicking the "Print" button, retrieving the printed images, using the pen to indicate comments and various shapes on the images, and then verifying that the strokes were loaded into the computer and associated with the correct images.

4. Design Interface

Contextual Inquiry

We learned from the contextual inquiry assignment that our target users (radiologists and medical imaging researchers) would like a quick, intuitive way to choose regions of interest (ROI's). Our target users work with many medical images on a regular basis, so any device or system that can improve efficiency and reduce the time spent on each image would be welcomed. Additionally, a system for the simultaneous review of resident sample analyses of images would enhance an instructing radiologist's ability to assess their findings and return helpful comments. One interviewee emphasized her desire for a quicker, more direct way to annotate medical images in general. We also learned that radiologists typically assess images in a dedicating reading room that includes a handful of computer stations. They also share their findings with their patients in their offices, and may verbally share findings in group meetings. Researchers process medical images at the point of acquisition, at lab workstations, at their desktop computers, and even on their laptops. Sharing of written observations was therefore lacking. Based on the preceding information, the interface we had in mind was one that resembled a multi-document image viewer. We thought it would be wise to include a preview pane, would allow quick selection of image sets from within a given directory. The main document space would present its user with the medical images under consideration, along with representations of layers that store the annotations made to the images. A slidebar at the bottom of the screen would allow the user to rapidly scan through the slices within the imaging set. For a given image, a set of layer thumbnails would be displayed near the bottom of the screen. Each layer would represent comments from a different user. By selecting one of the thumbnails, one could place that layer over the original image in the main window. A "Comment" button would then create a dialog box where the user can add his/her comments about the currently viewed layer. These comments would then be attached to that layer's metadata, and would be viewable by the original creator of that layer. Additionally, a "Print" button would be included. After selecting the print button, the user would be prompted to print either an ROI printout or a commenting printout on Anoto



Comments:
① notice the transparency of the printed image
② wtf is this ?
③ Images can be resized to allow writing in margins

[✓] Done (Check Last)

Comments:
George, look at this!

[✓] 1. **draw border** around ROI
2. check analyses for selected ROI:
[ ] pixel histogram
[✓] water diffusion
[✓] compute mass
[ ] Other(s):

[✓] Done (Check Last)

Figure 1: Left to right: Rough sketch of computer interface, paper UI for general annotation, and paper UI for ROI selection.

paper. A commenting printout would only include the original image and a "Done" check box. An ROI printout would include the original image and "Done" box, along with check boxes for running certain types of analysis, such as pixel histograms, once the *anoto* strokes are uploaded to the computer. Figure 1 indicates our initial concepts.

Lo-Fi Prototype: Design

Following the contextual inquiry, we created a pen, paper, and tape low-fidelity implementation of our interface. Figure 2 shows an example screen, while Figure 3 shows the lo-fi paper interface for general comments.

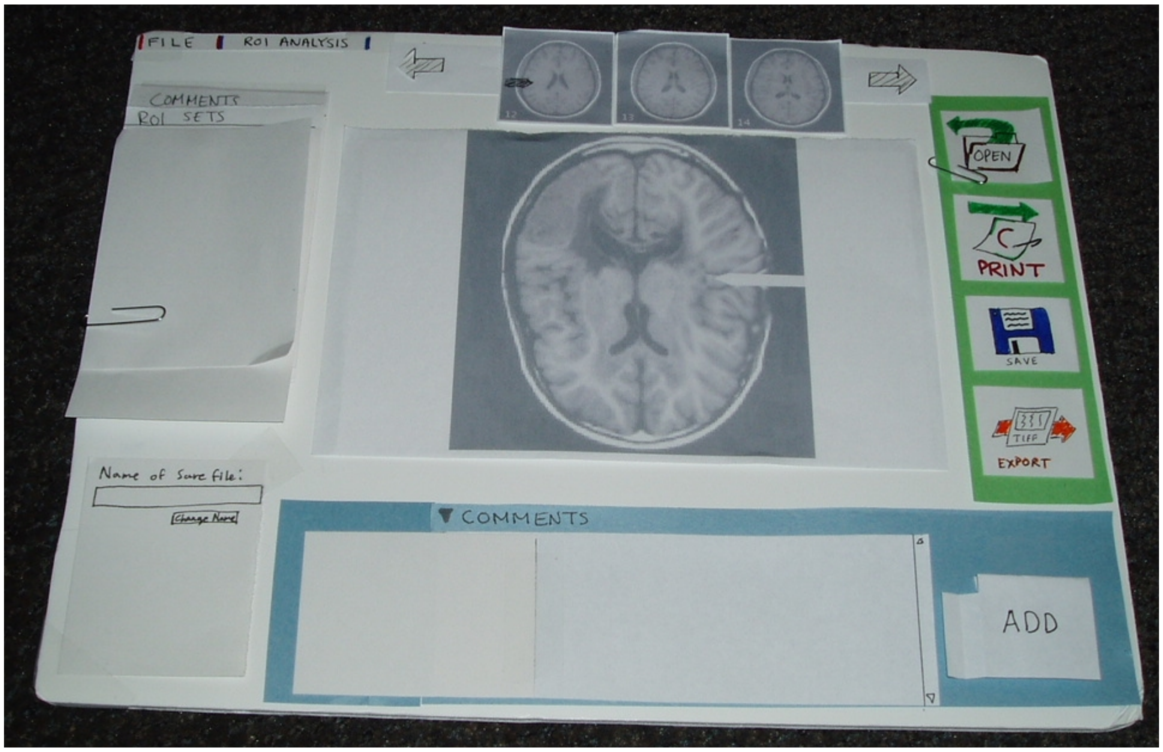


Figure 2: Lo-fi computer interface.

As seen above, we kept our ideas for a comment text box, a main window showing the currently selected image, a thumbnail slider for images, and button shortcuts for actions such as printing, saving, exporting, and opening. An “Add” button was included to commit recently typed text comments to the current slice. The left side of the screen also included collapsible lists (collapsed in the presented view) that allowed the user to toggle various comment layers and ROI sets associated with the currently viewed image. Below the collapsible list was a box for constantly displaying the name of the current annotation set. The concept was to remind the user of the name of the currently loaded set, as well as provide a quick way to rename it. The filename box was added on-the-fly during one of the interviews in response to a user request.

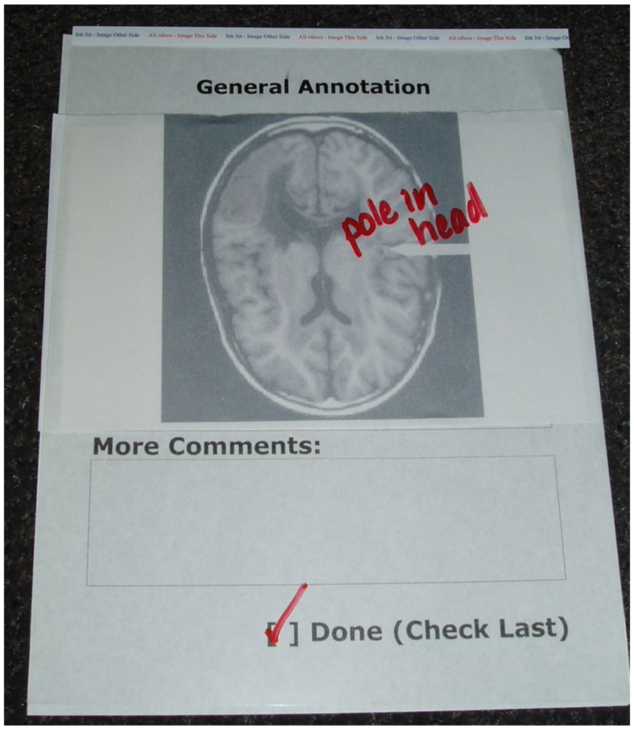


Figure 3: Lo-fi general comments paper UI.

The paper UI seen in Figure 3 shows the more simple general annotation layout. The original image slice was presented, along with a special comments box and done checkbox. During the lo-fi interviews, the image slice printouts were sandwiched between the paper UI and a transparency, allowing the user to add comments without leaving permanent marks on the slice or the UI.

Lo-Fi Prototype: Reactions and Adjustments

We had the users accomplish three tasks: jot down comments on a general comments printout, run ROI analysis using an ROI printout, and add comments to another user's annotations using the comments text box. Figures 4-6 show the storyboards for the tasks.

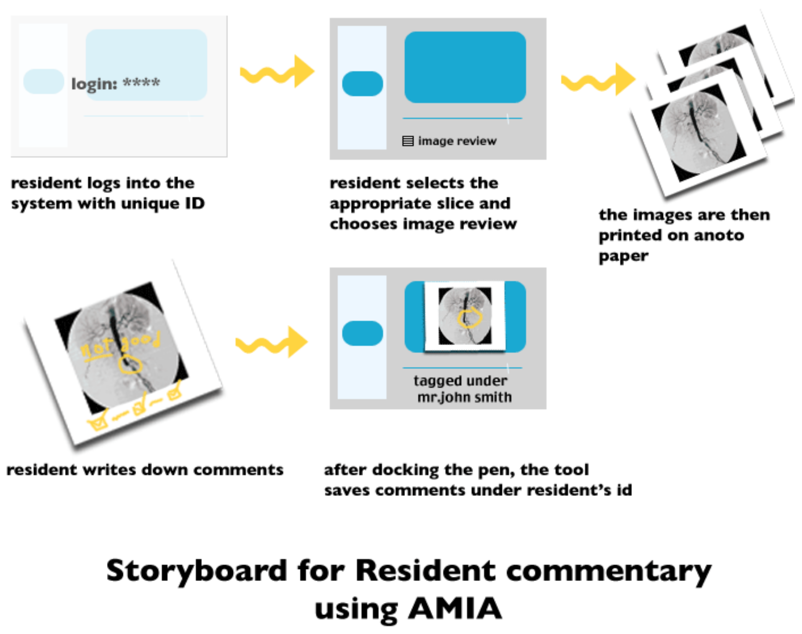

login: ****
resident logs into the system with unique ID
image review
resident selects the appropriate slice and chooses image review
the images are then printed on anoto paper
resident writes down comments
tagged under mr.john smith
after docking the pen, the tool saves comments under resident's id
Storyboard for Resident commentary using AMIA

Figure 4: General annotation task.

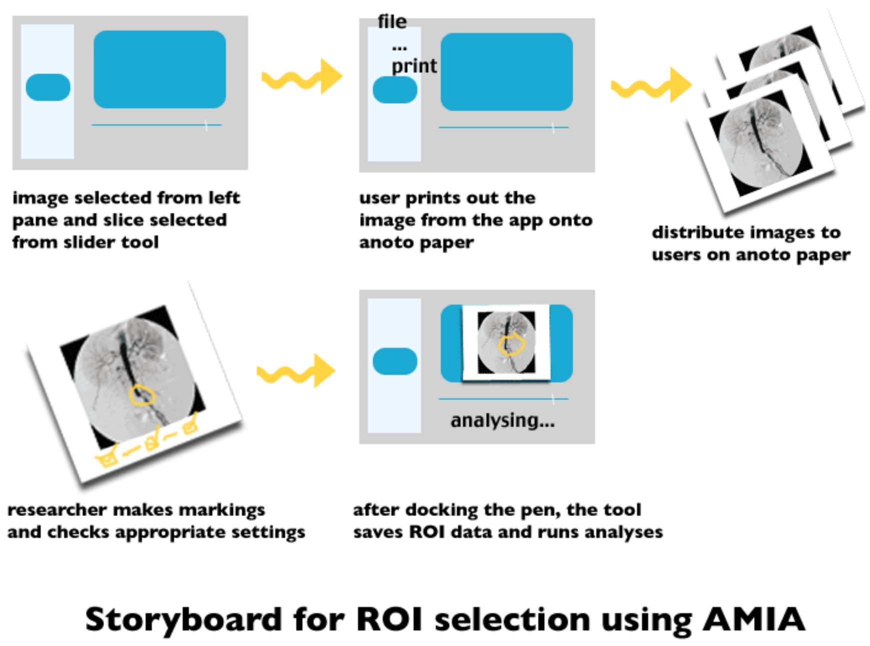


image selected from left pane and slice selected from slider tool
file
...
print
user prints out the image from the app onto anoto paper
distribute images to users on anoto paper
researcher makes markings and checks appropriate settings
analysing...
after docking the pen, the tool saves ROI data and runs analyses
Storyboard for ROI selection using AMIA

Figure 5: ROI selection task.

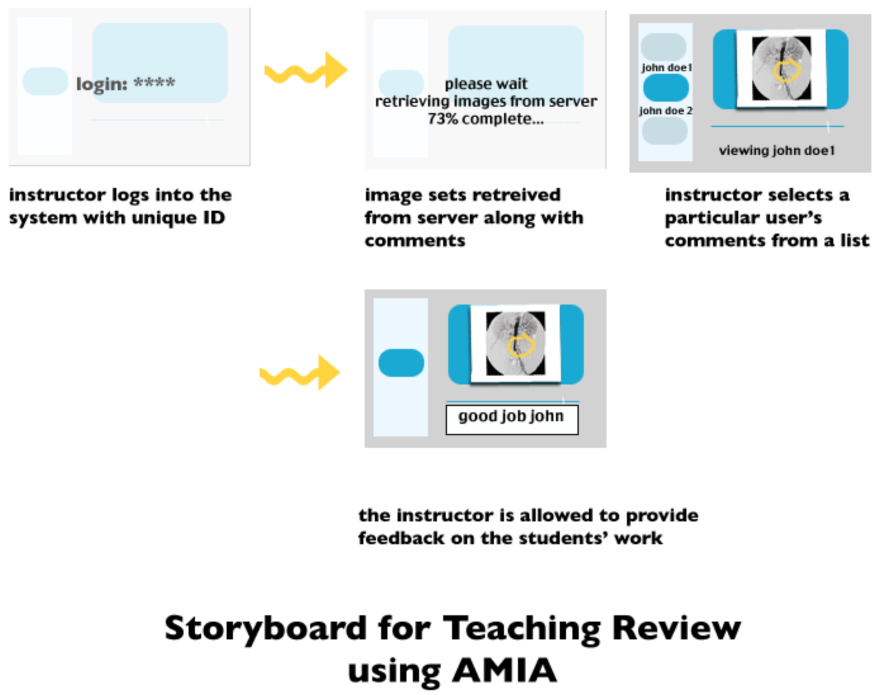

login: ****
please wait
retrieving images from server
73% complete...
john doe1
john doe 2
viewing john doe1
good job john
instructor logs into the system with unique ID
image sets retreived from server along with comments
instructor selects a particular user's comments from a list
the instructor is allowed to provide feedback on the students' work
Storyboard for Teaching Review using AMIA

Figure 6: Teacher comments task.

After testing the lo-fi prototype with three target users, we gained valuable insight into the advantages and disadvantages of AMIA. First, we learned that the ROI selection functionality should be removed. Upon examination of our users' interaction with this feature, we learned that not only were they having some difficulty achieving the required tasks, but they also found the use of paper-based medium for ROI selection rather unhelpful. Given that there are existing tools that can do ROI selection in an automated fashion with more sophistication than AMIA, it was decided to focus on the collaborative/instructive aspect of the program, as there are no tools that provide such functionality in the market today.

Additionally, users found the "Add" button (meant to represent Add Comment for the onscreen text box) rather confusing. We decided to rename it to "Save Comment". Some users expressed concerns with the way file I/O was conducted through the UI. The prototype had a persistent "save file" pane which was confusing to users. Some suggested moving this into a more traditional File->Save or Save As action. Users also recommended not launching the application with an "Open File" dialog, but rather allow them to manually open files. To that end, features like thumbnails in the file dialog boxes would be helpful as our application deals with images. However, we considered these to be a bit too advanced for our interactive prototype and have left the file open process as mostly a "wizard of oz" feature for the time being.

Medical image orientation was considered important to the interviewees. All of our examples include MR coronal cross-sections of the brain, when in reality the user would be able to view sagittal, coronal, and transverse sections of the same patient. We decided not to address the issue in the interactive prototype, as coronal images are sufficient to test the program's functionality.

Finally, the lo-fi prototype of AMIA required the user to print out slices of an image one at a time. Based on the prototype test, we learned that it would be easier for the user to select multiple slices and print them at once on Anoto paper.

Interactive Prototype: Design

For the interactive prototype, we attempted to implement image loading, thumbnail browsing, paper UI printing, stroke syncing, text commenting, and image exportation. The Java Swing interface that we created follows in Figure 7.

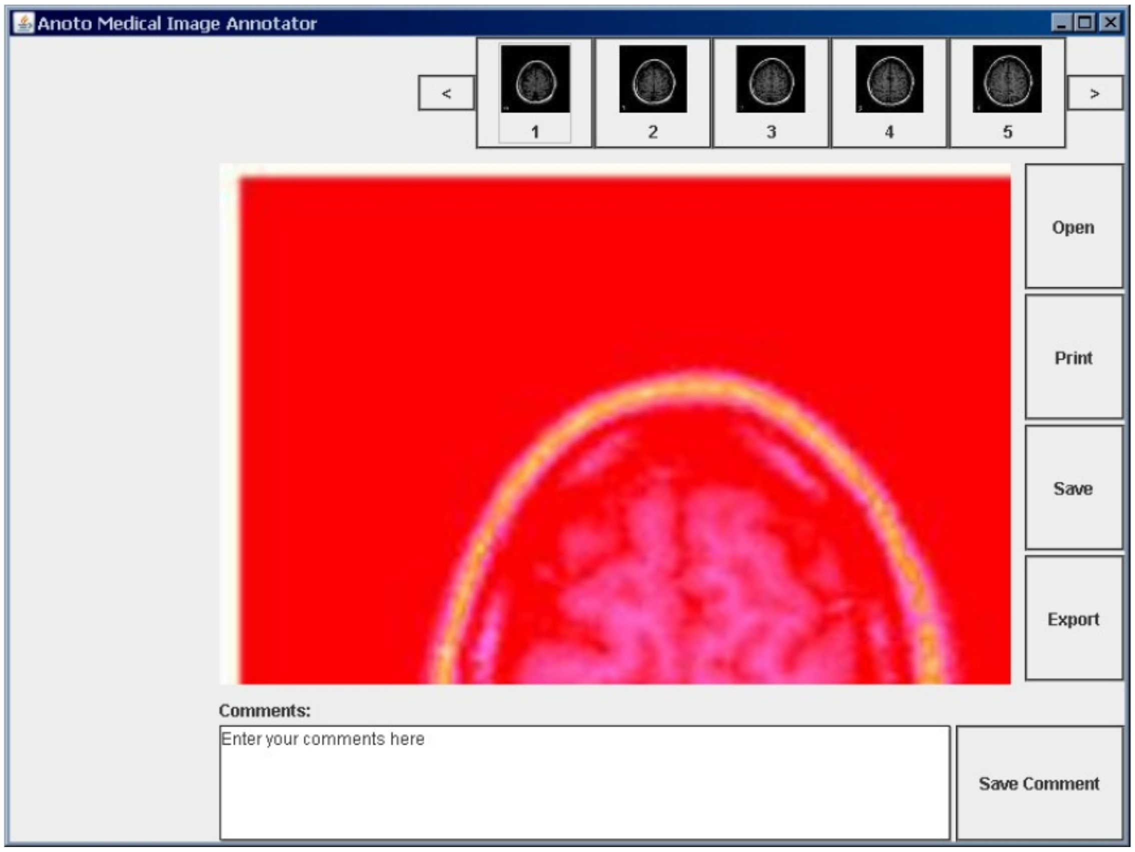


Figure 7: Screenshot of early interface as it was presented for in class. The main window displays the current image, while the thumbnail browser across the top can be used to switch between images within a set. The buttons along the right will allow the user to open, save, print, and export image. The text box on the bottom provides a way for the user to enter text comments to be associated with the current image.

Due to time constraints, we unfortunately could only finish the image loading and comment input for the prototype presentation. Image loading was implemented to provide the user with a directory selection dialog. Following the selection of a folder, all images therein were loaded into the program. Clicking on one of the thumbnails along the top of the screen loaded the corresponding image into the main window. Unfortunately, as seen above, the images were not scaled for full visibility within the main window. However, before the pilot studies began the following week, the image resizing was fixed. Most importantly, we implemented stroke streaming.

The program was written to automatically create .pdf files of the paper UI's for each of the loaded image slices. Since the printing process took a considerable amount of time for each .pdf (~5-10 minutes), we decided to print several sample sheets ahead of time. In the original implementation completed prior to the in-class presentation, we attempted to use the R3 toolkit's "bundle" methods to create series of paper UI's with unique dot patterns. We hoped that we could then program the computer to recognize each sheet as the user annotated them, and automatically associated the strokes with the correct files. However, the bundle methods were not fully implemented and we could not get the program to recognize the sheets. To fix the problem for the pilot studies, we went back to the original, single sheet .pdf generator methods. This meant that each printed sheet had the same dot pattern. As a result, it was up to the use to be sure to have the correct image selected on the computer to ensure that the hand-written notes were recorded to the correct slice. It should also be noted that only the dot pattern over the image was recognized. The "Comments" and "Check here when finished" patterns were not given any functionality in the prototype.

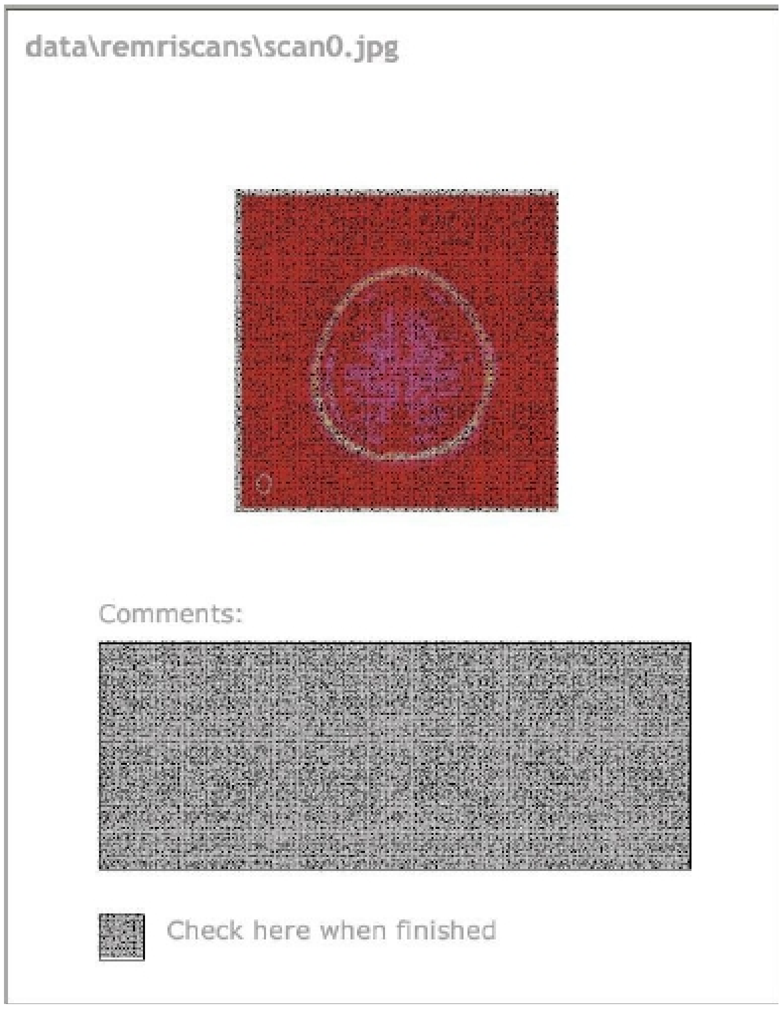


Figure 8: Example paper user interface. The anoto dot pattern is printed over the medical image to facilitate annotation. The comment box is used to indicate additional notes. The check box is used to tell the system that the current annotation session is over.

Pilot Study Reactions and Adjustments

We interviewed a neuroscience post doc, a radiology graduate student, and a radiology professor. Each person was asked to perform the three tasks listed in section 3 of the report.

We learned that the red color of the printouts made it difficult to differentiate between gray and white matter and the anatomy of the brain in general. The shortcoming could affect the teaching setting, since it would make it more difficult for students to recognize features. However, the red color was necessary in order for the *anoto* pend to recognize the overlaid black dot pattern. A compromise would be to always display the black and white image on the screen, but print out the red version for use with the anoto dots. In general, HIPAA standards would prevent one from ever printing out patient data and bringing it to a café or on a plane. Thus, one would need to annotate images in one's office or reading room. As a result, we changed our code to present the original, black and white image in the main window of the program.

During the second task, the requirement of the user to press the "save comment" button was not intuitive. One interviewee recommended an automatic saving routine for the text comments. Also, she mentioned that the anoto strokes must be saved to each slice, which was planned originally but not implemented in time for the interview. We have implemented the stroke saving and text comment auto-save features.

We were told that the comment box would be better for all written comments, not just those added later by another person. In other words, the paper UI should be set up only to afford shape drawings and very brief comments. The user should be expected to enter most text on the computer to maintain legibility. The comment was kept in mind as we rebuilt the paper UI. It now only includes the title of the image file, the image placed within a large anoto dot field, and a brief note instructing the user where to add annotations. The new UI is much simpler and less cluttered than the first implementation.

It was suggested that we replace the directory open dialog with a directory tree. We looked into the tree structure for an open-file dialog, but could not find a solution that our time frame could accommodate, so we stayed with the implementation in our interactive prototype.

In general, the subjects were impressed by our interactive prototype. In particular, the live streaming nature of our implementation was appreciated. In one of the interviewee's words, it "made it easy to get immediate feedback from the computer" to verify that one's strokes are being registered in the correct location. In general, she felt batched mode would be less useful, since HIPAA standards would prevent travel with patient data. However, one subject stressed that the system would not offer a big advantage over current diagnostic systems. Nonetheless, he saw definite potential in its application to a teaching setting. He emphasized that the "gimmick" nature of the device would be a strong selling point. That is, the novel idea of a digital pen for student interaction would like grab their attention and promote participation in class discussions. As the interview concluded, the interviewee made sure to point out that very stringent requirements are in place for monitors that are used for medical image-based diagnostics. These include contrast, brightness, and resolution criteria that could not be provided by a paper printout. We decided to leave these items unchanged in our final implementation, since the interviewer agreed such improvements would not be necessary for a basic teaching aid.

Usefulness of Evaluations

All of the interviews were instrumental to our project's development. However, the lo-fi prototype was the most useful interview technique, followed by the contextual inquiry, and lastly the pilot study. While contextual inquiry helped us become better acquainted with the needs of our target user group, it was difficult to refine our focus. Said differently, it provided us with a daunting amount of data that was hard to pick apart as we created our first draft interface. The lo-fi prototype, however, gave us very specific, useful feedback. Once the users were interviewed within the context of our interface, they were able to rapidly list several features they would expect. Their comments were vital to the creation of the interactive prototype. Finally, the pilot study gave us feedback about the finer details of our interface, but as we had already used the preceding interviews to refine the interface, we did not gain any exciting new insights.

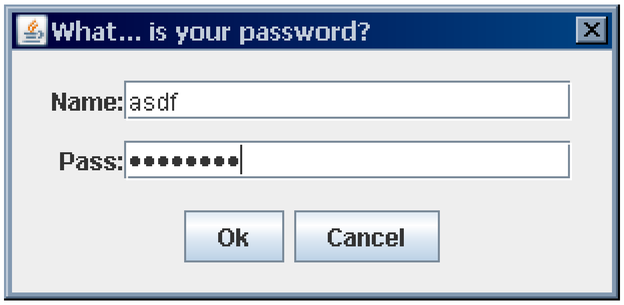
5. Final Interface

Overview

The final version of the software allows the user to login using a preset user name and password, open entire folders of images, automatically create paper UI's from those images, and add comments to the images. The user can either use the *anoto* digital pen to add annotations directly to the paper UI's, or else add text comments using a designated box in the program interface. All comments, handwritten or typed, are attached to a specific image and can be viewed at the user's leisure.

Design and Functionality

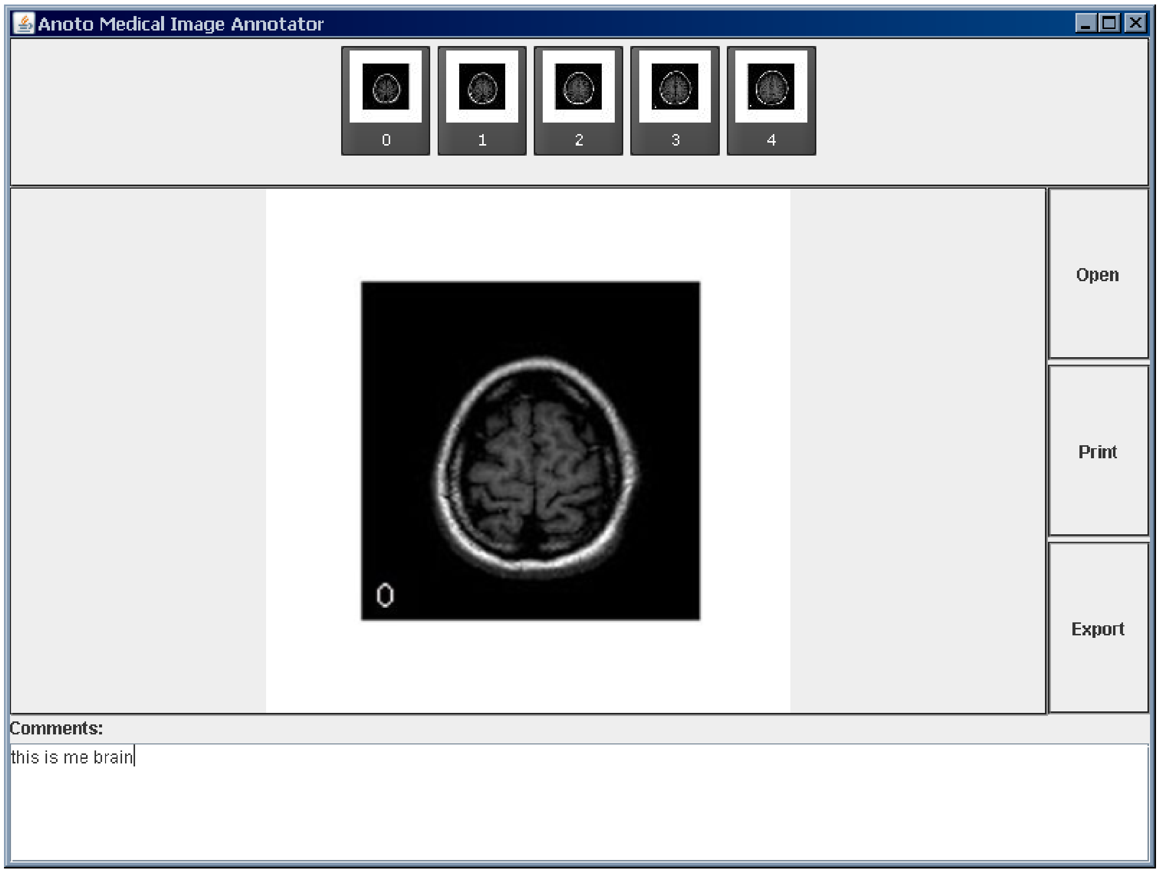
When the program is executed, the user is first greeted with a login screen. After successfully entering a username and password (Figure 9), the main interface opens (Figure 10).


What... is your password?

Name: asdf

Pass: •••••••••

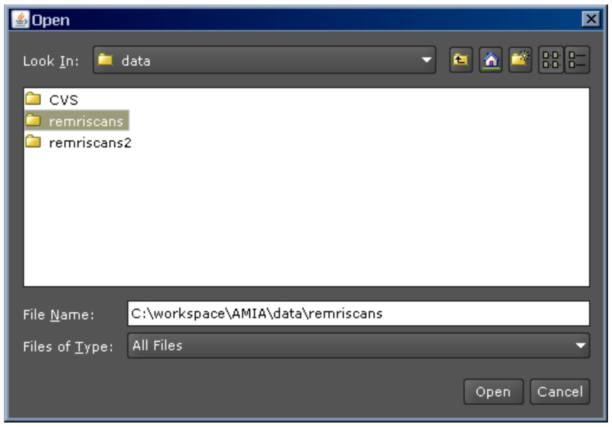
Figure 9: AMIA login screen.



this is me brain

Figure 10: AMIA main screen, shown with a loaded image set.

To access a set of images, the user must click the "Open" button on the right side of the screen. A dialog box then prompts the user to select a folder (Figure 11).



File Name: C:\workspace\AMIA\data\remriscans

Figure 11: Folder opening dialog box.

After selecting a file and pressing "Open," the program all the images within the folder, creates paper UI .pdf's, and loads the images into thumbnails placed along the top of the

screen (refer to Figure 10 again). Clicking on a thumbnail loads the full-sized version of the image into the main window, along with the associated strokes and text comments. Any *anoto* strokes made on a printout appears on the currently selected image.
Clicking the "Print" button hypothetically allows the user to print out a set of medical images on individual pages. In the current implementation, the button only shows a dialog box stating that image has been printed. We chose to print the .pdf files using the Foxit .pdf reader due to the inconvenient amount of time taken to print each image (5-10 minutes). Each printout (Figure 12) includes *anoto* dot patterns that allow the user to draw shapes and brief comments on the image.

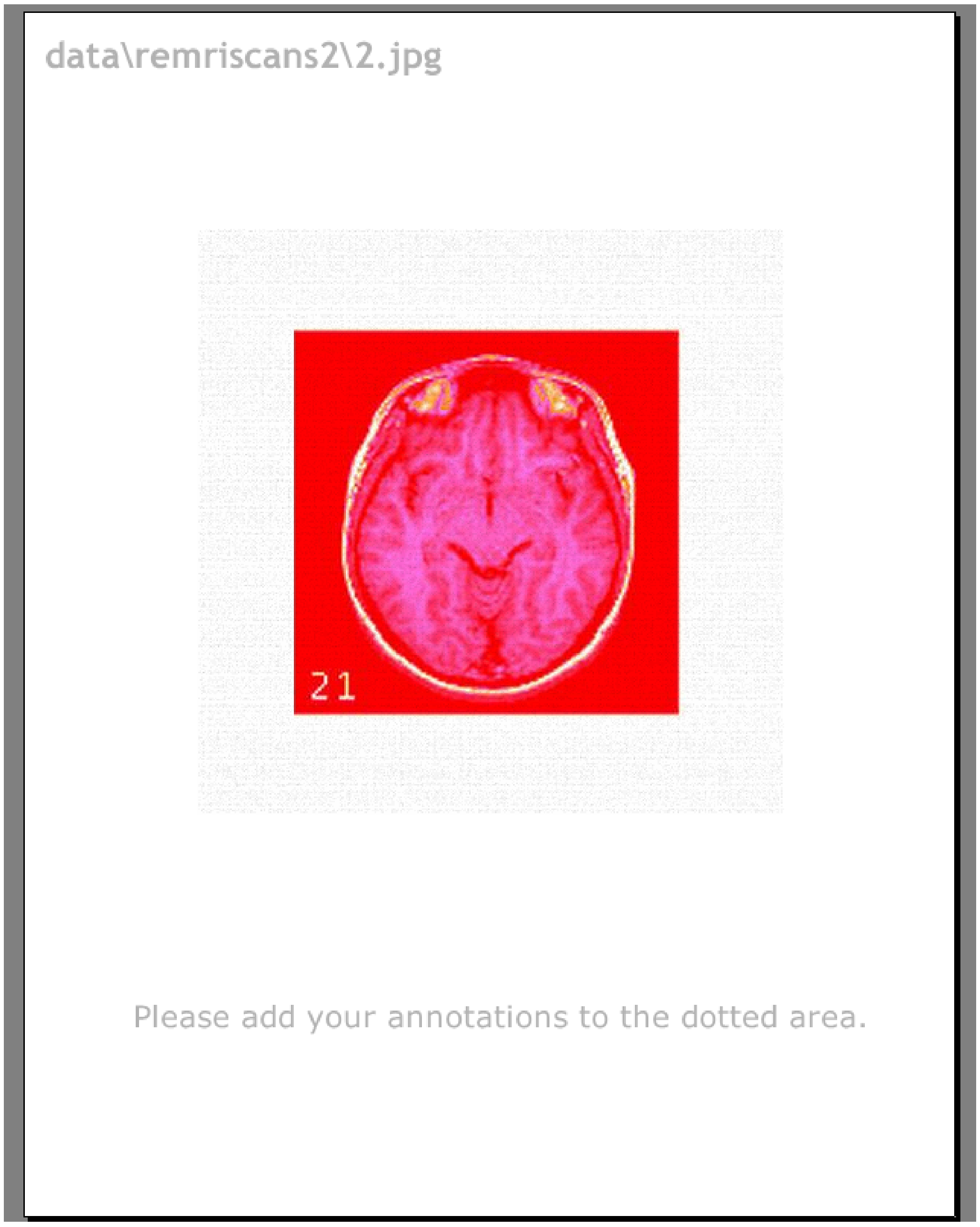


Figure 12: Example paper user interface using a transverse MRI image slice of the brain.

The user can also select the text box at the bottom of the screen in Figure 10 and add text comments to be associated with the image. The comments and *anoto* strokes are automatically saved, so the user can switch to a different thumbnail, then come back and be assured the old comments will remain. The "Export" button would allow the user to export the currently displayed image and its hand-drawn annotations to a .tiff image, but it could not be implemented in time for the final presentation.

Difficult Aspects of the Implementation

The trickiest part of the implementation was the generation of the paper UI's and subsequent syncing with the on-screen images. At first, we tried to use the "Bundle" methods found in the R3 toolkit. However, the associated methods were not completed at the time, so we necessarily had to abandon the bundles. Instead, the program individually creates a paper UI with anoto dot regions for each medical image slice. Due to the R3 implementation, each dot pattern is actually identical. So it the user's responsibility to make sure the correct image is loaded on the screen when s/he begins annotating a printout. At first, we encountered difficulty registering the strokes in the program. However, once we abandoned the Bundle approach, the program had no problem recognizing the strokes and displaying them on the screen. Additionally, we ran into several problems as we tried to use R3 to combine the *anoto* strokes with the underlying image in an export .tif image. As a result, we could not finish the exportation feature in time for the deadline. A repeating theme through our development of the project was the difficulty in using R3. We were able to code the SWING, user name/password, and other elements relatively quickly. However, we spent roughly 40% of our time struggling to set up the anoto system on each of our computers and debugging errors within the R3 code. A more complete, less buggy toolkit would have greatly enhanced our experience. Also, the fact that we had only one pen to share made it extremely difficult for everyone to write individual pieces of code. Specifically, only one person at a time could perform any debugging.

Omitted Features and Wizard of Oz Replacements

Most of the omitted features have already been discussed previously in the report. We originally wanted to have unique dot patterns on each printout, so the program would be able to automatically display the corresponding image as soon as the anoto pen started writing. However, we ran into problem with the R3 implementation and instead had to print the same dot pattern on each printout, leaving it to the user to load the correct image on the computer before adding any annotations. Although exportation of .tif images could not be completed on time, it was not considered a large loss as it had a minimal effect on the functionality of the interface. Also, we wanted to have a tree structure for the directory selection routine, but could not implement it time for the assignment due date.
Our key Wizard of Oz (WoZ) feature lies within the "Print" button. Printing each paper user interface takes roughly 5-10 minutes. So we decided to print the pages ahead of time using the freeware Foxit .pdf reader. As a result, the demonstrations of the program were not held up by long printing times and we were able to focus the users' attention on the functionality of the interface, rather than its speed

Addendum: Presentation Group Members

Unfortunately, Ani Vemprala was unable to make it to the final presentation due to a conflict with a full-time job interview. Due to the long weekend and other issues, it was difficult to reschedule the interview to a later time.