ARSenic: a Free, Open-Source, Web-based Audience Response System

Instructor's Guide

Version 0.92
latest revision: July 26, 2014

Michael L. Richardson, M.D.

Professor of Radiology
University of Washington
Department of Radiology
Seattle, WA
mrich@uw.edu

Contents

Chapter 1

Purpose of ARSenic

Executive Summary

Audience response systems (ARS) have been available since the 1960s [1]. As prices have dropped and availability has increased, these systems have become increasing popular as pedagogical tools. ARS technology enables active learning by allowing real-time polling of student opinions and knowledge, as well as improving attentiveness and engagement in the classroom.

However, dissatisfaction with the cost and features of commercial ARS prompted me to create my own system — ARSenic — which polls students using personal web-based devices such as smart phones.

ARSenic is a free, open-source, web-based, ARS that can be installed on one's laptop or on a server. An instructor can control the polling using any web-capable device, such as a laptop, a smart phone or a tablet computer. Students type in their answers and view answer summaries using any web-capable device. This manual includes instructions for installing and using ARSenic on one's own computer. Full source code for this system is also included.

Rationale for creating ARSenic

Audience response systems (ARS) have been available since the 1960s [2]. As prices have dropped and availability has increased, these systems have become increasing popular as a pedagogical tool. ARS technology enables active learning by allowing real-time polling of student opinions and knowledge. Benefits of an ARS include [1]:

- improves attentiveness [3]
- increases student engagement in the classroom [4]
- increases knowledge retention [4, 5]
- allows anonymous polling
- gives immediate feedback
- promotes interactive learning
- insures that students understand key points before moving on
- collects data for reporting and analysis

There are many commercial ARS systems. Many of these use a combination of a student-held response unit (clicker) and an instructor-based receiver unit. This type of system usually requires several hundred dollars of hardware, as well as extra time to dispense, retrieve and maintain the clickers during teaching activities.

Since 2007, smartphones and other portable computing devices have become increasingly common among students and instructors, making it possible to use these not only as an ARS input device [6, 7], but as a receiving unit as well [7].

Most of the ARS systems currently available limit the instructor to specific types of questions, such as multiple choice questions (MCQ). While it is easy to grade MCQ and display their results in graphic form, they also have a number of disadvantages. Students can get correct answers by random guessing or through sheer "quizmanship". Writing well-designed MCQ is an acquired skill that can require considerable practice to master [8]. Another problem with MCQ is the "leading the witness" effect, in which a student picks the correct answer only because they see it listed among the possible answers.

MCQ are best adapted for testing well-defined or lower-order skills, while problem-solving and higher-order reasoning skills are better assessed through short-answer and essay tests [1]. For example, a diagnostic imaging study often yields a differential diagnosis, rather than one correct answer. An ARS that lets students to enter multiple sequential short answers allows an instructor to efficiently poll an entire class as to the most likely differential diagnosis (figure 1.1).

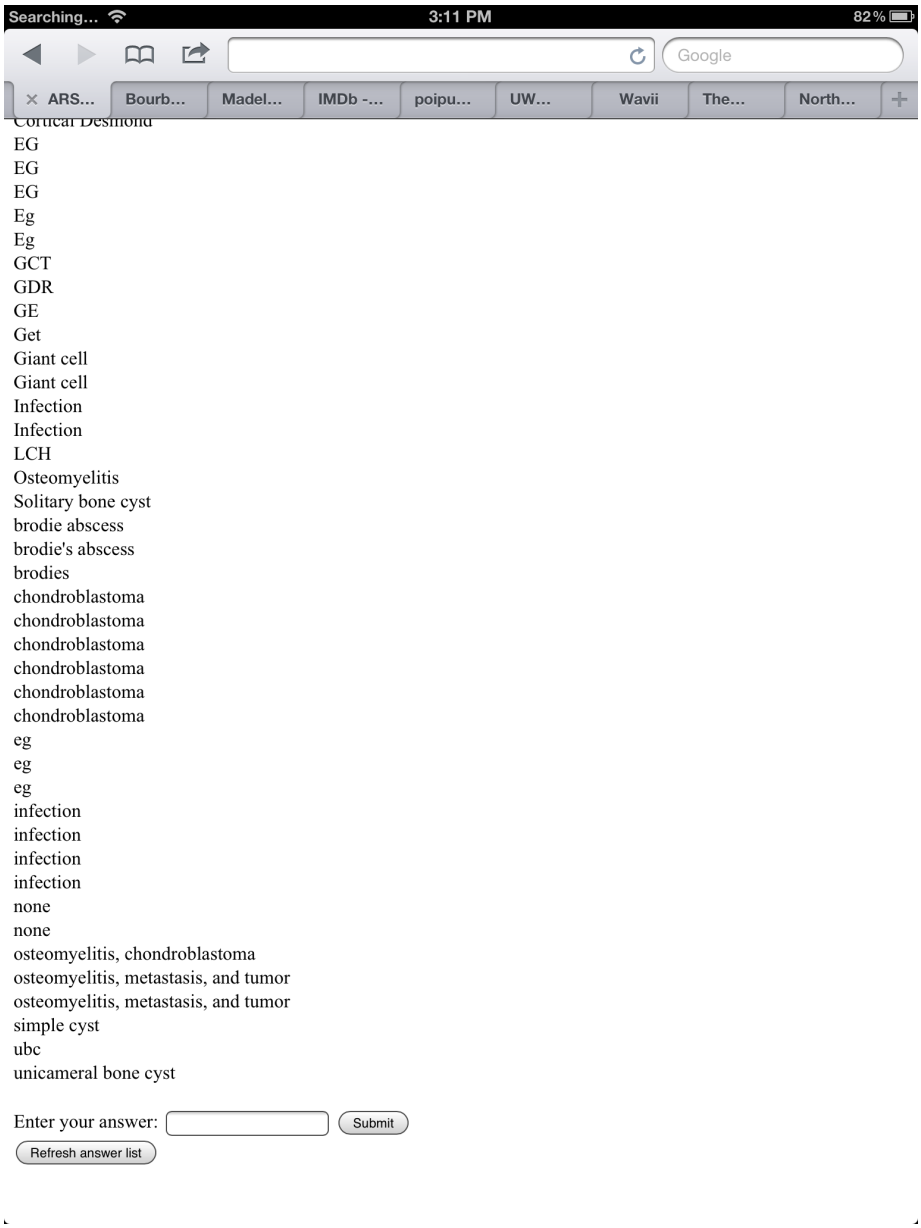

Figure 1.1: iPad screenshot from an ARSenic teaching session. I asked my radiology residents to sequentially enter their top 3 diagnoses for a particular lucent bone lesion.

It can take a *lot* of time to prepare well-written MCQ and embed them into a presentation. After spending hours constructing a high-quality presentation, one may well begrudge spending another 30-60 minutes writing ARS questions.

Finally, an MCQ-based ARS makes it difficult to add questions on the fly during a presentation.

Because of these considerations, the optimal ARS for my personal use has the following criteria:

- the ability to pose short answer questions
- minimal or no preparation time
- no password needed for students to access ARS
- no software installation required for students
- web-based system that works with any modern browser
- simple interface for students and instructor
- works with any presentation system (even a physical chalkboard)
- works on multiple platforms
- no extra hardware costs
- minimal expense
- works on laptop over local WiFi
- runs in background on laptop behind other programs
- can be installed on laptop or fixed-location server

Since I could not find a commercial ARS that met these criteria, I decided to create my own simple web-based ARS system, which I call ARSenic. It has met my needs quite well, and has been successfully deployed at several other sites by academic pals who have kindly helped me to beta-test this software.

Although I occasionally prepare questions ahead of time, I usually don't bother. Most radiology lectures or case conferences revolve around fairly obvious questions, such as, "What is the findings?" or "What is the most likely diagnosis?" or "What are the top 3 things on the differential diagnosis?".

This manual describes how to install and operate ARSenic on Macintosh, Windows or Linux computers. Full source code for this system is included for those who wish to adapt or further develop ARSenic.

Chapter 2

System requirements for ARSenic

While ARSenic is platform-independent, there are certain system requirements. The scripts that do most of the work are written in Python, version 2.7.1 [9]. However, these scripts should also work in most other recent versions of Python.

Your computer will also need to have some type of webserver software installed. My Macintosh Mac OS X computer has version version 2.2.22 of Apache [10] installed by default. My Ubuntu 11.04 system currently runs Apache 2.2.17.

Chapter 3

Installing ARSenic

Installing ARSenic on Mac OS X

Installation on a Mac OS X system is very simple.

1. download the following file:

`http://uwmsk.org/arsenic_distribution/ARSenic_0.75.pkg`

Double-click this installer package and follow the installer instructions. An administrator's password will be needed to to install the ARSenic files.

This installer will automatically install a folder called "arsenic" in the standard Mac OS X web document root – the /Library/WebServer/Documents folder. It will also install five Python scripts and one configuration file into the /Library/WebServer/CGI-Executables folder.

2. start the webserver on your computer (Mac OS X 10.7 (Lion) and earlier)

On Mac OS X Lion (10.7) and earlier, this is done by going to System Preferences, clicking on the "Sharing" preference pane, and then clicking the check box for "Web Sharing". This will start the Apache webserver built into the Mac OS X operating system.

3. install a web-sharing preference pane (Mac OS X 10.8 (Mountain Lion) and 10.9 (Mavericks))

When Apple updated the Mac operating system to 10.8, they removed the web sharing preference pane in System Preferences. However, Tyler Hall has written a preference pane to replace it. This can be downloaded at:

`http://clickontyler.com/blog/2012/02/web-sharing-mountain-lion/`

This preference pane can be installed by double-clicking on the downloaded file (figure 3.1).

Once installed, the web sharing preferences pane has only one control — an on/off switch (figure 3.2). I leave mine on all the time.

Figure 3.1: Web Sharing prefPane.

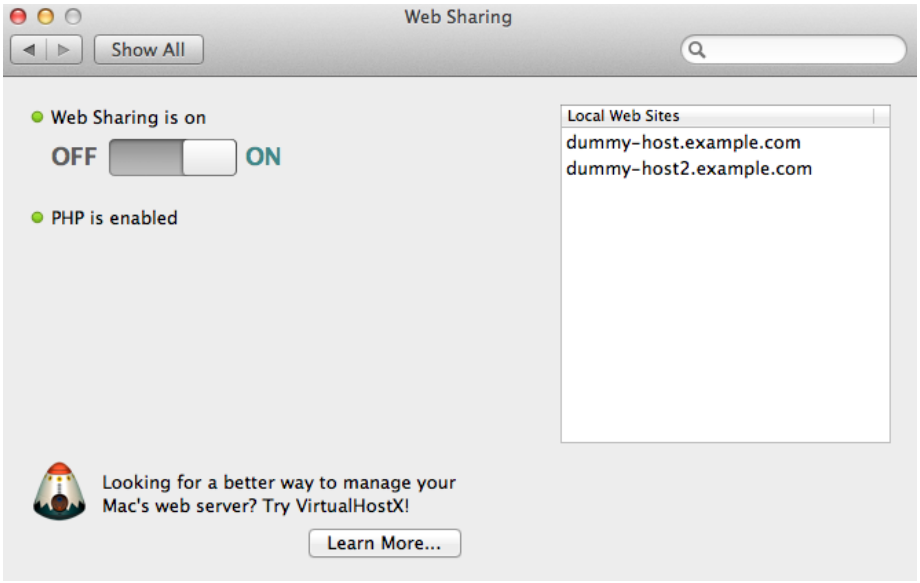

Figure 3.2: Web Sharing control panel.

Installing ARSenic on Linux or Windows

For non-Macintosh platforms, installing ARSenic requires the user to be somewhat comfortable with the command line interface on their computer.

1. download the ARSenic files at:

`http://uwmsk.org/arsenic_distribution/ARSenic_0.75.zip`

Upzip this file, which contains two folders of files that will need to be installed.

2. place the arsenic folder in your computer's web document root

Inside a folder called "put_the_arsenic_folder_in_web_doc_root", there is another folder, called "arsenic". The entire arsenic folder should be moved into the folder that is the web document root of one's computer. The precise location of this folder varies among platforms, webserver software and software versions. For a typical Macintosh Mac OS X installation, this is the /Library/WebServer/Documents folder. For my Ubuntu 11.04 system running Apache 2.2.17, this is the /var/www folder.

3. insure that the webserver has read and write permission for the file "answers.txt"

The "answers.txt" file lies within the arsenic folder. On a Unix-based platform such as Mac OS X or Linux, one can set these permissions by typing the following lines in the terminal program:

```
cd /Library/WebServer/Documents/arsenic
chmod 777 answers.txt
```

Depending on your system, you may have to type "sudo" in front of the "chmod" command, e.g.:

```
cd /Library/WebServer/Documents/arsenic
sudo chmod 777 answers.txt
```

4. place the Python scripts in your computer's cgi-bin folder

In a folder named "put_these_files_in_cgi_bin_folder", there are six files. All six of these files should be moved to the cgi-bin folder for your computer. The precise location of this folder varies among platforms, webserver software and software versions. For a typical Macintosh Mac OS X installation, this is the /Library/WebServer/CGI-Executables folder. For my Ubuntu 11.04 system running Apache 2.2.17, this is the /usr/lib/cgi-bin folder.

5. insure that these scripts are all executable

For example, on a Unix-based platform such as Mac OS X or Linux, one can set the appropriate permission for the arsenic_poll.cgi script by typing the following lines in the terminal program:

```
cd /Library/WebServer/CGI-Executables
chmod 755 arsenic\_poll.cgi
```

Again, depending on your system, you may have to first type "sudo" in front of the "chmod" command, e.g.:

```
cd /Library/WebServer/CGI-Executables
sudo chmod 755 arsenic\_poll.cgi
```

6. for non-Macintosh platforms, use a text editor to change the path information in the file "config.py" to match the exact path to your web document root.

The default path in the config.py file assumes that the web document root is located on a Macintosh at:

```
/Library/WebServer/Documents
```

For other platforms, replace "/Library/WebServer/Documents" with the path to your computer's document root. For example, if your Ubuntu document root is /var/www, then replace "/Library/WebServer/Documents" with "/var/www" in config.py.

7. start the webserver on your computer

For Unix and Windows computers, this will depend on exactly which specific webserver software and which specific version is installed. For example, on my Ubuntu 11.04 server running Apache 2.2.17, I restart the Apache webserver with the following command:

```
/etc/init.d/apache2 restart
```

Chapter 4

Using ARSenic

How I Use ARSenic

I use ARSenic in two different scenarios. In the first scenario, ARSenic runs on a remote web-server. On the plus side, no configuration or installation is needed, and all I need to do is to point my students toward the URL for the ARSenic server. On the downside, I and my users all need to have internet access for this to work properly.

In the second scenario, I run ARSenic on my own Mac laptop, using its built-in webserver. The plus side here is that I don't need to install ARSenic on a remote server — just on my laptop. I and my students do need to be on the same WiFi network, although the WiFi router does not actually need to be connected to the internet.

Scenario 1 — server-based instance of ARSenic

I keep a copy of ARSenic running continuously on one of my servers. This system allows users anywhere on the internet to chime in with their answers. I have used this to poll my residents at multiple locations around Seattle and to poll users from around the globe during online presentations.

Test-drive a server-based instance of ARSenic

To test-drive a server-based version of ARSenic, please visit the ARSenic website at http://uwmsk.org/arsenic. There you will find links to a student's view and an instructor's view of a live ARSenic session.

Using ARSenic is fairly simple. The student can do one of two things: enter an answer or refresh the answer list. The instructor can do one additional thing: clear the answer list for fresh questions.

Scenario 2 — laptop-based instance of ARSenic over local WiFi

I keep the built-in webserver software (a.k.a Web-Sharing) running on my Mac laptop all the time. This software (Apache 2.2.22) runs quietly in the background, while I run my presentation software in the foreground. I devote my laptop screen to my presentation software, and administer ARSenic via my iPad. It is possible to simultaneously give a presentation and administer ARSenic on the same laptop, but most users find this awkward.

There is really no technical limit as to how long you can run an ARS session. You will be limited instead by practical endpoints, such as your students dropping in their tracks or (more likely) hounding you from the lecture room.

Using a laptop-based version of ARSenic is very similar to that of a server-based instance. A student can do one of two things: enter an answer or refresh the answer list. An instructor can do two additional things: start an ARSenic session and clear the answer list for fresh questions.

Starting an ARSenic session

1. Get everyone on the same WiFi network

Generally, most presentations will be given in a room with only one choice for the WiFi network. However, if only some of your users are able to connect to ARSenic, dueling WiFi networks could be the cause.

2. Start ARSenic on the laptop

To begin a session, the instructor should point her laptop's web browser to a initial webpage with a name known only to the instructor. In the default installation of ARSenic, this page is called "arsenic_admin.html". However, this filename can be changed to any other desired name without affecting ARSenic's operation. In the default installation of ARSenic, the instructor would type in the following address in the browser address field to bring up the opening screen for ARSenic:

http://localhost/arsenic/arsenic_admin.html

Instructor's view The opening screen contains a single button for the instructor to start the ARS session (figure 4.1).

Figure 4.1: Opening instructor screen.

Once the instructor clicks the "Start ARSenic" button, the program interrogates the local computer's WiFi network connection, and displays a local URL to be given to one's students (figure 4.2).

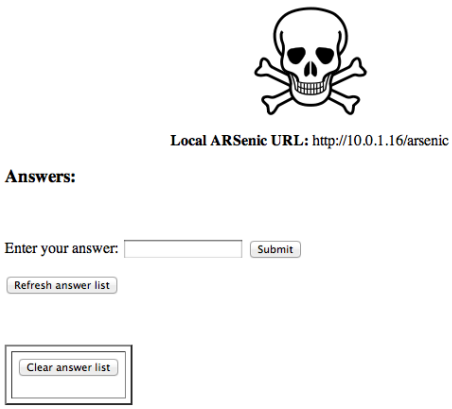

Figure 4.2: ARSenic instructor's screen, displaying the instructor controls as well as the local ARSenic URL to be given to students.

In the example above, when a student enters "http://10.0.1.16/arsenic" into the address field of their web browser, their browser will connect to ARSenic.

3. Enter admin URL into iPad

Once the local ARSenic URL has been determined, the administrative URL will be a slightly modified version of that URL. To control the session from my iPad, I type in this admin URL into the browser screen of my iPad. Using the example above where the local ARSenic URL is "http://10.0.1.16/arsenic", I type the following administrative URL into my iPad:

http://10.0.1.16/arsenic/arsenic_admin.html

4. Quiz the users at will

The instructor's screen should now be visible on the iPad. Besides the answer submission form, this screen includes buttons to refresh the answer list and to clear the answer list.

As students enter their answers, ARSenic collects the answers into a text file called "answers.txt", sorts them into alphabetical order, and displays them on the screen (figure 4.3).

Local ARSenic URL: http://10.0.1.16/arsenic

Answers:

ABC
ABC
EG
enchondroma
giant cell tumor
osteomyelitis
osteosarcoma

Enter your answer: Submit

Refresh answer list

Clear answer list

Figure 4.3: Instructor's view of the ARSenic screen once several answers have been entered by students.

Student view The student screen looks just like the instructor's screen, but without the ability to clear the answer list.

Scenario 3 — using a laptop as a WiFi router and ARSenic server

In some venues, you may find yourself at a venue with poor or no WiFi. Fear not, some laptops will allow you to establish your own local network, using the laptop's WiFi transceiver as a router. You do not need to be connected to the Internet to pull this off. You could therefore give this technique to hold an ARSenic session in a cave, or in a boat on the open sea, or on Mars. As long as your students are within range of your laptop's radio, Bob's your uncle.

I suppose that there must be some limit to the number of users a laptop's built-in WiFi can handle, but don't know what that number is. However, one ARSenic user has successfully used his Mac laptop to poll 30 students. I'd love to hear reports from the field of how well this works on other Macs and other systems.

Using a Mac laptop as a router

1. Creating a Network

Click the WiFi icon on your menu bar, and select "Create Network", as shown in figure 4.4.

Figure 4.4: Creating a personal wireless network on a Mac laptop.

This will bring up another small window, which will allow you to name your new network, select your preferred WiFi channel, and decide what kind of security you want (figure 4.5). I usually just use the system defaults.

3. Start ARSenic on the laptop

Starting ARSenic via a personal network is exactly the same process followed when using an existing WiFi network.

To begin a session, the instructor will point her laptop's web browser to a initial webpage with a name known only to the instructor. Again, in the default installation of ARSenic, this page is called

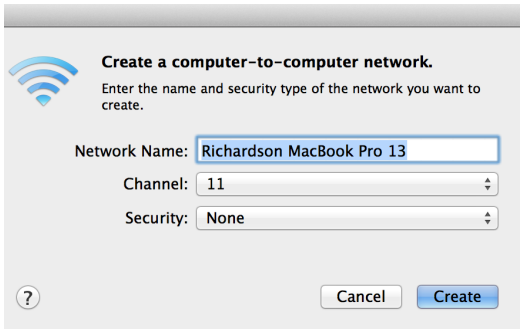

Figure 4.5: Configuring your personal network.

"arsenic_admin.html". However, this filename can be changed to any other desired name without affecting ARSenic's operation. In the default installation of ARSenic, the instructor would type in the following address in the browser address field to bring up the opening screen for ARSenic:

http://localhost/arsenic/arsenic_admin.html

Instructor's view The opening screen contains a single button for the instructor to start the ARS session (figure 4.6).

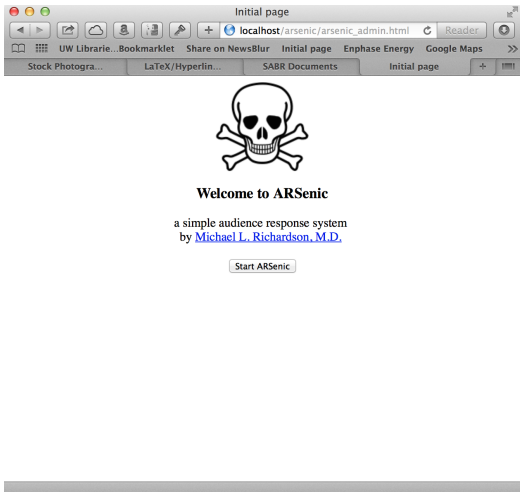

Figure 4.6: Opening the instructor screen.

Once the instructor clicks the "Start ARSenic" button, the program interrogates your personal computer to learn the details of the new personal network, and displays a local URL to be given to one's students (figure 4.7). In this example, the local ARSenic URL listed under the skull and crossbones is "http://169.254.107.5/arsenic". This is the URL that will be given to the students to enter into the address field of their web browser.

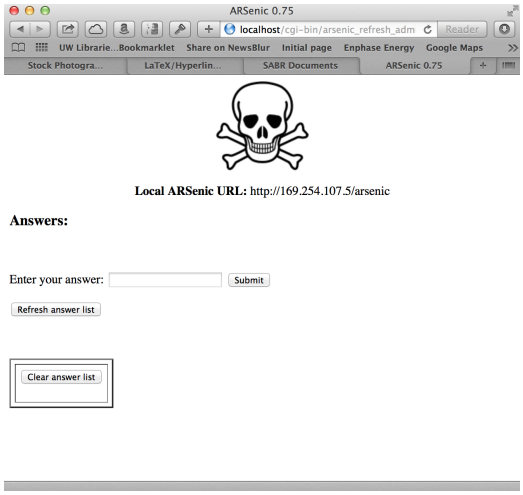

Figure 4.7: ARSenic instructor's screen, displaying the instructor controls as well as the personal network ARSenic URL to be given to students.

4. Connect to Personal Network

It's now time to get any personal internet devices used by you and your students connected to the personal network. To connect an iPhone to the personal network, go to the WiFi setting screen on my iPhone and look for the personal network. In this example, our new network — "Richardson MacBook Pro 13" — is shown listed under "Devices" (figure 4.8).

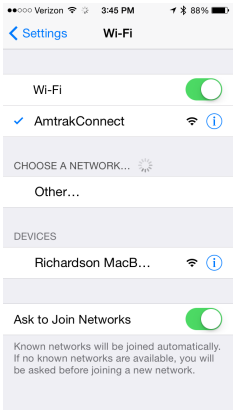

Figure 4.8: iPhone WiFi setting screen showing the new personal network listed under "Devices".

When selecting this new network, the iPhone presents a message, warning us that the laptop is not connected to the internet. In this scenario, we don't care about that, so we select "Join Anyway" (figure 4.9).

The WiFi settings page now shows that the iPhone is connected to the new personal network.

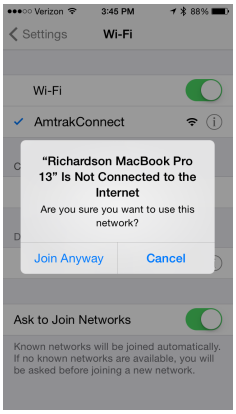

Figure 4.9: iPhone warning us that the laptop is not connected to the internet.

5. Entering the ARSenic URL on the personal network

The final step in getting your students connected to ARSenic is to give them the URL we discovered back in step 3 (figure 4.7). For this example, they would now enter "169.254.107.5 arsenic" into the address field of their iPhone browser. The iPhone should now display the familiar skull and crossbones icon of ARSenic (figure 4.10).

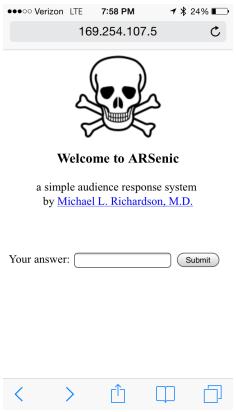

Figure 4.10: Success!! The iPhone is now connected to ARSenic via the personal WiFi network that we just set up.

In a similar manner, you should now be able to connect any other web-enabled device to our new personal network.

6. Disconnecting from the personal network

When you're done with your ARSenic session, don't forget to disconnect your Mac from its personal network. This is again done via the WiFi icon on the menu bar. For this example, we will select "Disconnect from Richardson MacBook Pro 13" (figure 4.11).

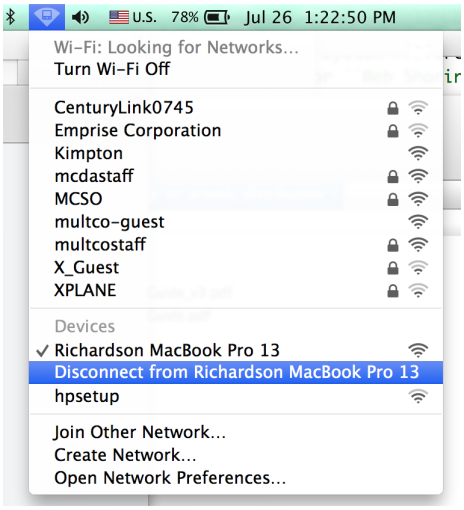

Figure 4.11: Disconnecting the laptop from the personal network we created.

Using other laptops as routers

I have no personal experience doing this with systems other than Mac laptops. In theory, this should be possible on any make of laptop. However, as the saying goes:

> *In theory, theory and practice are the same. In practice, they're not.*

If you are a Windows or Linux user and know how to do this, I'd be happy to include that information here. Please send me a list of steps and appropriate screenshots and I'll include your how-to in this instructor's guide.

Chapter 5

Limitations of ARSenic

To meet the criteria for an optimal ARS mentioned in Chapter 1, I spent a great deal of time throwing out potential features when designing ARSenic.

I didn't want students to have to bother with signing in to my ARS system. Therefore, users wishing to use an ARS for taking attendance or for administering exams should consider a commercial system that includes these features.

ARSenic has no built-in security — any idiot with a web browser can submit an answer via this system. This has not been a problem for me so far. However, widespread use of this system could make it a target for abuse. I therefore mention several practical ways of securing ARSenic in Chapter 6.

Users hoping for colorful bar charts and pie charts will not find them here. For one thing, I regard them as needless chart junk [11]. A better reason is that the fill-in-the-blank questions used by ARSenic do not lend themselves easily to summary bar charts or pie charts. Instead, ARSenic summarizes student-supplied answers in an alphabetized list.

Running a completely anonymized ARS system with fill-in-the-blank answers is not for the faint of heart, as students quickly realize that they can enter almost anything into the system. In my large ARS sessions (over 50 students), students have felt anonymous enough to include gentle badinage with me and the other students along with their answers (figure 5.1).

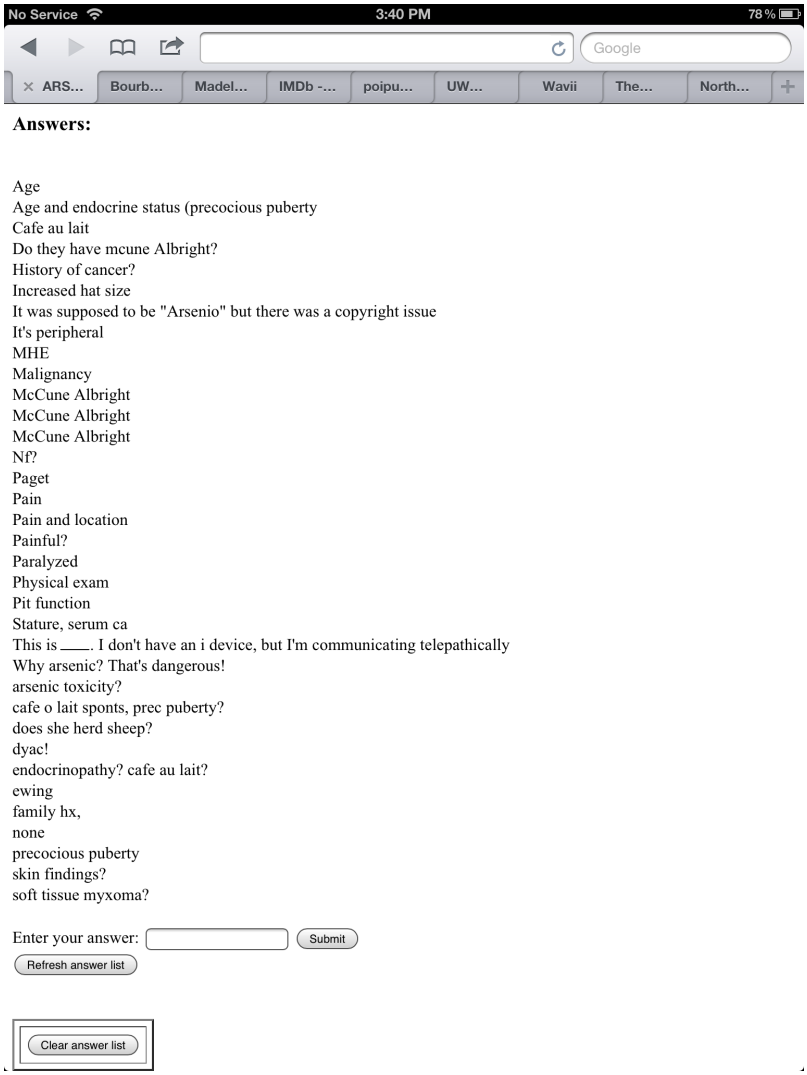

Figure 5.1: I showed my residents a radiograph of the pelvis of a patient with McCune-Albright syndrome. I asked them: "What question would you ask this patient?" This iPad screenshot contains a mixture of their answers and quips..

One clever student figured out how to insert valid HTML image tags into the answer page . This was the source of some merriment, but did not significantly detract from our teaching session. Personally, I was delighted that this unknown student was engaged enough in the ARS process to figure out how to do this. To date I have not seen actual profanity inserted into the ARS feed – one hopes that most students will act like grownups. If my audiences ever do push things a bit too far, it won't be hard to write a "nanny" filter that will match answers against banned words in its dictionary and excise them from the answer list.

Chapter 6

Security considerations

At present, the main security for ARSenic is its obscurity. In typical usage, ARSenic will be temporarily accessible only to those students who are on the same WiFi router as the instructor and only at the URL given by the instructor. Once the instructor leaves the vicinity of that router, the URL given to the students expires. For laptop-based usage during occasional lectures, this type of security should be more than sufficient.

If one is running ARSenic on a webserver that is constantly connected to the Internet, other security measures should be considered. This is the case for a number of my presentations, which I give to multiple remote sites using screen-sharing software (`http://www.gotomeeting.com`). For this purpose, I have installed a copy of ARSenic on our section webserver, allowing attendees at remote sites to express their opinions about cases.

One simple security measure for an Apache-based server would be to add a .htaccess file into the arsenic folder [12]. Adding this file requires all users to first supply a password before being allowed access to the ARSenic page. If more security is required, most webservers can be configured to limit access to ARSenic files to a specific range of internet protocol (IP) addresses, such as only to those located within one's medical school.

ARSenic may not work everywhere, due to the restrictive firewall [13] configurations at some sites. This is currently true for both our local pediatric specialty hospital and our local Veterans Affairs hospital.

Chapter 7

How ARSenic Works

This ARS was written in a combination of HyperText Markup Language (HTML) [14] and Python, version 2.7.1 [9]. The ARSenic logo was adapted from a public domain vector image file hosted at Wikipedia [15].

The student portion of ARSenic is controlled by one HTML file and two Common Gateway Interface (CGI) [16] scripts written in Python (table 7.1).

Table 7.1: These three files control the student portion of ARSenic.

Script name	**Action**
index.html	student entry point into ARSenic system
arsenic_poll.cgi	allows student to enter answers
arsenic_refresh.cgi	allows student to refresh the answer list

Interactions between the HTML file and the CGI scripts are shown in figure 7.1.

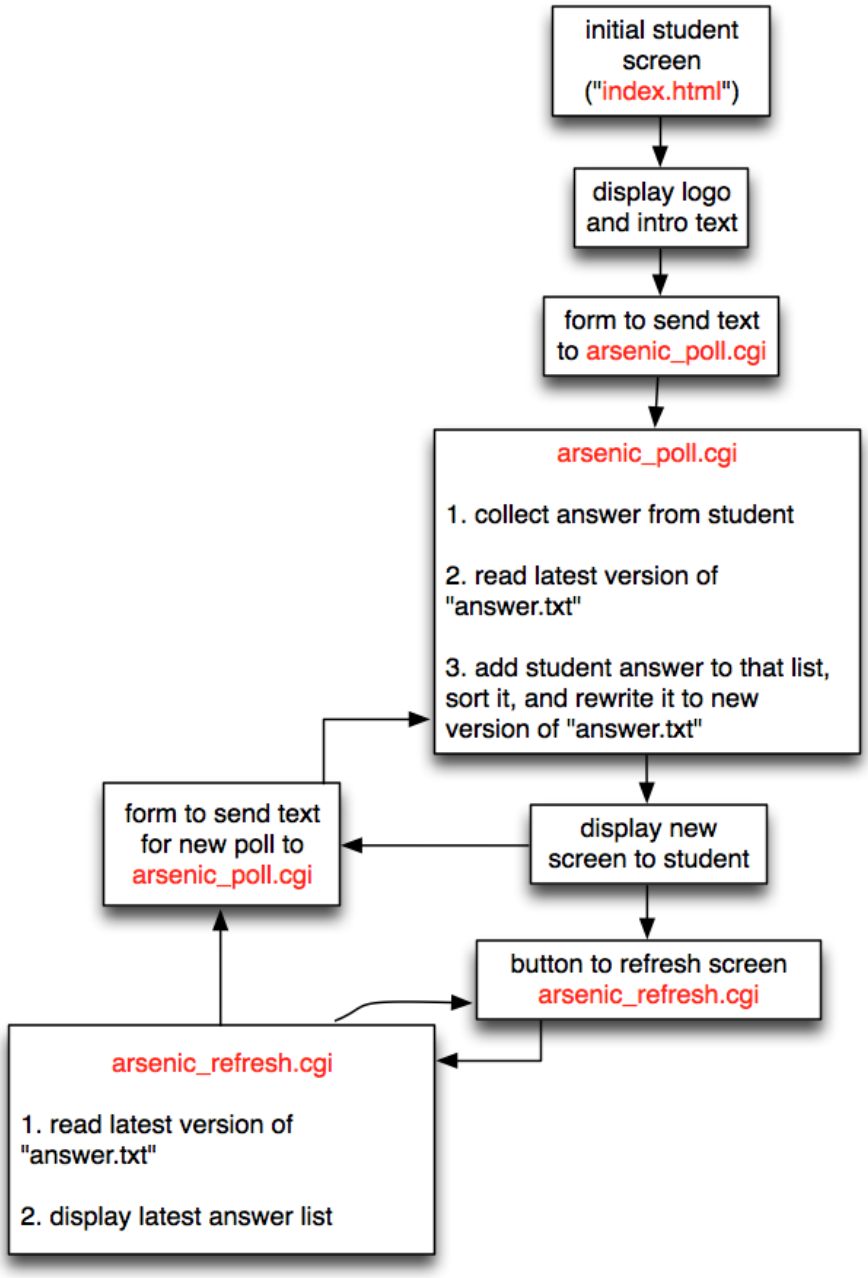

Figure 7.1: Flowchart describing the interaction of the HTML file and the CGI scripts running the student portion of ARSenic.

The instructor portion of ARSenic is controlled by one HTML file and three CGI scripts written in Python (table 7.2).

Table 7.2: These four files control the instructor portion of ARSenic.

Script name	**Action**
arsenic_admin.html	causes Python to interrogate local network and display local ARSenic URL
arsenic_poll_admin.cgi	allows instructor to enter answers
arsenic_refresh_admin.cgi	allows instructor to refresh the answer list
arsenic_clear_answers.cgi	allows instructor to clear the answer list

Interactions between the HTML file and the CGI scripts running the instructor portion of ARSenic are shown in figure 7.2.

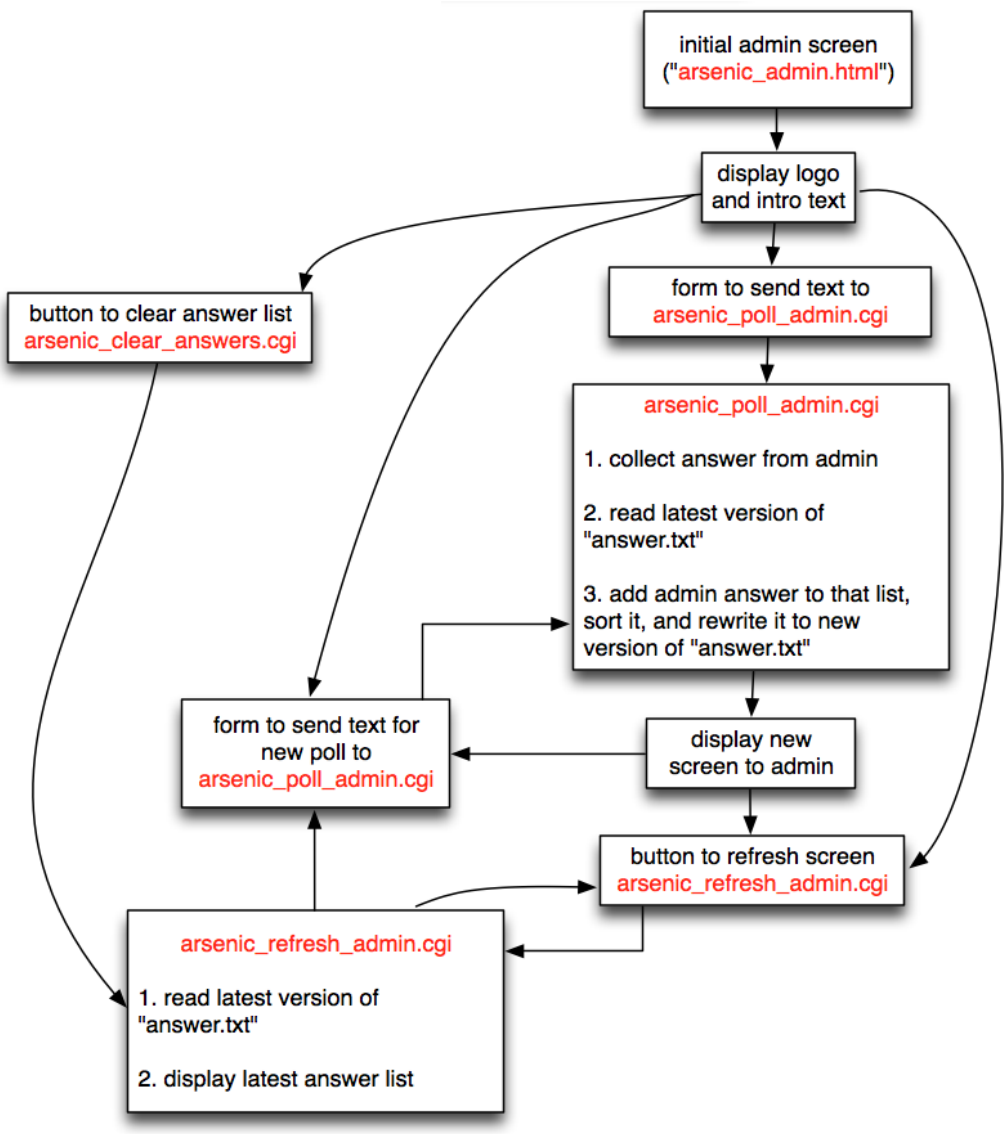

Figure 7.2: Flowchart describing the interaction of the HTML file and the CGI scripts running the instructor portion of ARSenic.

Chapter 8

ARSenic Source Code

Files to put in the webserver document root folder

index.html

```
<html>
  <head>
    <title>ARSenic</title>
        <meta name="viewport" content="user-scalable=yes, width=device-width" />
  </head>
  <body> <center>
                <img src ="120px-Skull\_and\_crossbones.png">
                <h3>Welcome to ARSenic</h3>
                a simple audience response system <br>by
               <a href="mailto:mrich@uw.edu">Michael L. Richardson, M.D.</a>
          </center>
          <br><br><br>
    <form action = '/cgi-bin/arsenic\_poll.cgi'>
    Your answer: <input type='text' name = 'name' />
    <input type = 'submit' />
    </form>
  </body>
</html>
```

arsenic_admin.html

```
<html>
  <head>
    <title>Initial page</title>
        <meta name="viewport" content="user-scalable=yes, width=device-width" />
  </head>
  <body>
         <center>
                <img src ="120px-Skull_and_crossbones.png">
                <h3>Welcome to ARSenic</h3>
                a simple audience response system <br>by
               <a href="mailto:mrich@uw.edu">Michael L. Richardson, M.D.</a>
               <br><br>
                  <form action = '/cgi-bin/arsenic_refresh_admin.cgi'>
     <input type = 'submit' name = 'refresh' value = 'Start ARSenic'/>
    </form>
         </center>
  </body>
</html>
```

answers.txt

```
<br>ABC<br>enchondroma
```

Files to put in the webserver cgi-bin folder

config.py

```
####################################
# Configuration file for ARSenic
#
# contains current document root
# for ARSenic
#
# and current version number
# of ARSenic
#
####################################

# The default path variable is set for a standard Mac OS X system.
# For other systems, change this to the path to the web document
# root on your computer.

path = '/Library/WebServer/Documents/'

# This variable contains the current version number of ARSenic.

version = '0.75'
```

arsenic_poll.cgi

```
#!/usr/bin/python

import cgitb; cgitb.enable()
import cgi
import config

form = cgi.FieldStorage()

name = form.getvalue('name', 'none')

# grab answers given so far
f = open(config.path + 'arsenic/answers.txt', 'r')
answers_old = f.read()
f.close()

# add user's answer to end of list of other answers and write it to disk
# answers_new = answers_old + '<br>' + name

# add user's answer to end of the text file containing other answers
# then split that file by '<br>' into a list
# then sort that list, join the elements by new '<br>'

answers_new = '<br>'.join(sorted((answers_old + '<br>' + name).split('<br>')))

# now write the new sorted list of answers to disk

f = open(config.path + 'arsenic/answers.txt', 'w')
f.write(answers_new)
f.close()

# now print out page with latest answers

print """Content-type: text/html

<html>
  <head>
    <title>ARSenic %s</title>
     <meta name="viewport" content="user-scalable=yes, width=device-width" />
  </head>
  <body>
    <h3>Answers:</h3>
    %s
<br><br>
    <form action = '/cgi-bin/arsenic_poll.cgi'>
```

```
    Enter your answer: <input type='text' name = 'name' />
    <input type = 'submit' />
    </form>
    <br>
    <form action = '/cgi-bin/arsenic_refresh.cgi'>
      <input type = 'submit' name = 'refresh' value = 'Refresh answer list'/>
    </form>
  </body>
</html>
""" % (config.version, answers_new)
```

arsenic_poll_admin.cgi

```
#!/usr/bin/python

import cgitb; cgitb.enable()
import cgi
import socket
import config

form = cgi.FieldStorage()

name = form.getvalue('name', 'none')

# grab answers given so far
f = open(config.path + 'arsenic/answers.txt', 'r')
answers_old = f.read()
f.close()

# add user's answer to end of list of other answers and write it to disk
# answers_new = answers_old + '<br>' + name

# add user's answer to end of the text file containing other answers
# then split that file by '<br>' into a list
# then sort that list, join the elements by new '<br>'

answers_new = '<br>'.join(sorted((answers_old + '<br>' + name).split('<br>')))

# now write the new sorted list of answers to disk

f = open(config.path + 'arsenic/answers.txt', 'w')
f.write(answers_new)
f.close()

myIP = socket.gethostbyname(socket.gethostname())

# now print out page with latest answers

print """Content-type: text/html

<html>
  <head>
    <title>ARSenic %s</title>
    <meta name="viewport" content="user-scalable=yes, width=device-width" />
  </head>
  <body>
  <center>
```

```
       <img src ="/arsenic/120px-Skull_and_crossbones.png"><br><br>
  <b>Local ARSenic URL:</b> http://%s/arsenic
  </center>

    <h3>Answers:</h3>
    %s
    <br><br>
    <form action = '/cgi-bin/arsenic_poll_admin.cgi'>
    Enter your answer: <input type='text' name = 'name' />
    <input type = 'submit' />
    </form>
    <br>
    <form action = '/cgi-bin/arsenic_refresh_admin.cgi'>
      <input type = 'submit' name = 'refresh' value = 'Refresh answer list'/>
    </form>
    <br> <br><br>
    <table border="2" cellspacing="5" cellpadding="5">
        <tr><td>
        <form action = '/cgi-bin/arsenic_clear_answers.cgi'>
         <input type = 'submit' name = 'clear' value = 'Clear answer list'/>
    </form>
    </td></tr>
    </table>
    <br>

</html>
""" % (config.version, myIP, answers_new)
```

arsenic_refresh.cgi

```
#!/usr/bin/python

import cgitb; cgitb.enable()
import cgi
import config

form = cgi.FieldStorage()

name = form.getvalue('name', 'none')

# grab answers given so far
f = open(config.path + 'arsenic/answers.txt', 'r')
answers_new = f.read()
f.close()

# now print out page with latest answers

print """Content-type: text/html

<html>
  <head>
    <title>ARSenic %s</title>
    <meta name="viewport" content="user-scalable=yes, width=device-width" />
  </head>
  <body>
    <h3>Answers:</h3>
    %s
    <br><br>
    <form action = '/cgi-bin/arsenic_poll.cgi'>
    Enter your answer: <input type='text' name = 'name' />
    <input type = 'submit' />
    </form>

    <form action = '/cgi-bin/arsenic_refresh.cgi'>
      <input type = 'submit' name = 'refresh' value = 'Refresh answer list'/>
    </form>

  </body>
</html>
""" % (config.version, answers_new)
```

arsenic_refresh_admin.cgi

```
#!/usr/bin/python

import cgitb; cgitb.enable()
import cgi
import socket
import config

form = cgi.FieldStorage()

name = form.getvalue('name', 'none')

# grab answers given so far
f = open(config.path + 'arsenic/answers.txt', 'r')
answers_new = f.read()
f.close()

myIP = socket.gethostbyname(socket.gethostname())

# now print out page with latest answers

print """Content-type: text/html

<html>
  <head>
    <title>ARSenic %s</title>
    <meta name="viewport" content="user-scalable=yes, width=device-width" />
  </head>
  <body>
  <center>
       <img src ="/arsenic/120px-Skull_and_crossbones.png"><br><br>
<b>Local ARSenic URL:</b> http://%s/arsenic
</center>

    <h3>Answers:</h3>
    %s
    <br><br>
    <form action = '/cgi-bin/arsenic_poll_admin.cgi'>
    Enter your answer: <input type='text' name = 'name' />
    <input type = 'submit' />
    </form>

    <form action = '/cgi-bin/arsenic_refresh_admin.cgi'>
      <input type = 'submit' name = 'refresh' value = 'Refresh answer list'/>
    </form>
```

```
    <br> <br>
    <table border="2" cellspacing="5" cellpadding="5">
         <tr><td>
         <form action = '/cgi-bin/arsenic_clear_answers.cgi'>
          <input type = 'submit' name = 'clear' value = 'Clear answer list'/>
    </form>
    </td></tr>
    </table>
    <br>
</html>
""" % (config.version, myIP, answers_new)
```

arsenic_clear_answers.cgi

```
#!/usr/bin/python

import cgitb; cgitb.enable()
import cgi
import socket
import config

form = cgi.FieldStorage()

name = form.getvalue('name', 'none')

# grab answers given so far
f = open(config.path + 'arsenic/answers.txt', 'r')
answers_old = f.read()
f.close()

# replace them with a <br> and nothing else.

answers_new = ''

# now write the newly cleansed answer file to disk

f = open(config.path + 'arsenic/answers.txt', 'w')
f.write(answers_new)
f.close()

myIP = socket.gethostbyname(socket.gethostname())

# now print out page with latest answers

print """Content-type: text/html

<html>
  <head>
    <title>ARSenic %s</title>
    <meta name="viewport" content="user-scalable=yes, width=device-width" />
  </head>
  <body>
  <center>
       <img src ="/arsenic/120px-Skull_and_crossbones.png"><br><br>
  <b>Local ARSenic URL:</b> http://%s/arsenic
  </center>

    <h3>Answers:</h3>
```

```
    %s
    <br><br>
    <form action = '/cgi-bin/arsenic_poll_admin.cgi'>
    Enter your answer: <input type='text' name = 'name' />
    <input type = 'submit' />
    </form>

    <form action = '/cgi-bin/arsenic_refresh_admin.cgi'>
      <input type = 'submit' name = 'refresh' value = 'Refresh answer list'/>
    </form>
    <br> <br>

    <table border="2" cellspacing="5" cellpadding="5">
         <tr><td>
         <form action = '/cgi-bin/arsenic_clear_answers.cgi'>
          <input type = 'submit' name = 'clear' value = 'Clear answer list'/>
    </form>
    </td></tr>
    </table>

  </body>
</html>
""" % (config.version, myIP, answers_new)
```

Bibliography

[1] Wikipedia: **Audience response — Wikipedia, The Free Encyclopedia** 2013, http://en.wikipedia.org/w/index.php?title=Audience_response&oldid=546717924. [[Online; accessed 26-March-2013]].

[2] **Audience response** http://en.wikipedia.org/wiki/Audience_response.

[3] Miller RG, Ashar BH, Getz KJ: **Evaluation of an audience response system for the continuing education of health professionals**. *J Contin Educ Health Prof* 2003, **23**(2):109–15.

[4] Miller RG, Ashar BH, Getz KJ: **Evaluation of an audience response system for the continuing education of health professionals.** *Journal of Continuing Education in the Health Professions* 2003, **23**(2):109–115.

[5] Crouch CH, Mazur E: **Peer Instruction: Ten years of experience and results.** *Am J Phys* 2001, **69**(9):970.

[6] **Response Technology for Mobile Devices** http://www.turningtechnologies.com/.

[7] **Eclicker Presenter for iOS** http://www.eclicker.com/.

[8] **Item Writing Manual, National Board of Medical Examiners** http://www.nbme.org/publications/item-writing-manual-download.html.

[9] **Python Programming Language** http://www.python.org/.

[10] **The Apache Software Foundation** http://www.apache.org/.

[11] Wikipedia: **Chartjunk — Wikipedia, The Free Encyclopedia** 2013, http://en.wikipedia.org/w/index.php?title=Chartjunk&oldid=544569255. [[Online; accessed 26-March-2013]].

[12] Wikipedia: **.htaccess — Wikipedia, The Free Encyclopedia** 2013, http://en.wikipedia.org/w/index.php?title=.htaccess&oldid=539640664. [[Online; accessed 26-March-2013]].

[13] Wikipedia: **Firewall (computing) — Wikipedia, The Free Encyclopedia** 2013, http://en.wikipedia.org/w/index.php?title=Firewall_(computing)&oldid=546937955. [[Online; accessed 26-March-2013]].

[14] Wikipedia: **HTML — Wikipedia, The Free Encyclopedia** 2013, `http://en.wikipedia.org/w/index.php?title=HTML&oldid=546889505`. [[Online; accessed 26-March-2013]].

[15] **Skull and crossbones** `http://en.wikipedia.org/wiki/File:Skull_and_crossbones.svg`.

[16] Wikipedia: **Common Gateway Interface — Wikipedia, The Free Encyclopedia** 2013, `http://en.wikipedia.org/w/index.php?title=Common_Gateway_Interface&oldid=546334037`. [[Online; accessed 26-March-2013]].