2015

Creating Computer Availability Maps

Scott Bacon
Coastal Carolina University, sbacon@coastal.edu

Follow this and additional works at: https://digitalcommons.coastal.edu/lib-fac-pub
Part of the Computer Sciences Commons, and the Library and Information Science Commons

Recommended Citation

This Article is brought to you for free and open access by the Kimbel Library and Bryan Information Commons at CCU Digital Commons. It has been accepted for inclusion in Library Faculty Publications by an authorized administrator of CCU Digital Commons. For more information, please contact commons@coastal.edu.
Creating Computer Availability Maps

Scott Bacon

Kimbel Library, Coastal Carolina University

During the fall semester of 2012, the Kimbel Library and Bryan Information Commons conducted a LibQUAL+ (libqual.org) survey to gauge how patron needs were being met. Many survey respondents commented that there were not enough computers for use in the library buildings. This issue was remedied by the addition of over 70 percent more computer workstations in the new commons building, a move that was well received by patrons. However, these desktops still filled up quickly, especially during midterms and finals, so the library decided to create a dynamic system to show patrons which computers were available for use at any given time. This is how the Kimbel Library Computer Availability Map project (coastal.edu/library/maps/availability) was born.

To create this service the library needed to involve several departments and needed to tie several different data points together. After reading Kim Griggs’s Code4Lib Journal article “How to Build a Computer Availability Map,”¹ we decided that a computer availability map mashup would be the perfect solution. Our project used Griggs’s code as a foundation of the mapping system.

What Are Computer Availability Maps?

Simply stated, computer availability maps provide an interface that shows patrons the location of computer workstations and whether those computers are available or are in use. This is usually accomplished by
adding a programming script to the machines, which records the status of the computer ("available" or "in-use") and then sends this data to the database. The database is then queried to show different icons depending on the computer status, usually something like a green circle for available or red circle for in-use (Figure 14.1).

Computer availability maps can be purchased by using a proprietary mapping service such as LabMaps (labstats.com/products/labmaps). In this chapter, however, I will detail how to create an in-house solution using open source code and a little programming. I will outline the steps Kimbel Library took to create our mapping system and provide links to our code (coastal.edu/library/maps/availability/zipfiles/availabilityfiles.zip) so that anyone with a little coding and programming knowledge can create maps for their institution.

**Why Should You Make Computer Availability Maps?**

Librarians serve their patrons in many ways by providing access to resources and services. Maps help patrons orient themselves with your access points and provide a visual path to these points. Our patrons like Kimbel Library because it is a place to study, meet friends, and utilize technology. They also see our librarians as forward thinking, using technology to provide them with faster, more modern, and more integrated services such as these maps. Building
services for your patrons helps to establish relationship and forms a good impression of the library in their minds.

What Makes These Maps Mashups?

A broad definition of a mashup might read: creating a single online service by joining data together using other services. This is how our computer availability maps were built. We used our MySQL database to load computer IDs, pixel positions, and map locations. We used a Perl script on the computers to tell the database whether the computer was logged in or logged out. We designed a PHP- and HTML-based user interface that included the maps and computer icons. We designed a graphical user interface console that allowed library administrators to change icon locations through drag-and-drop functionality. Finally we developed several ways for patrons to access the maps: through the help desk monitors, the full website, and our mobile website. Our maps illustrate how several different data points and applications can be mashed up to create new functionality for patrons.

Working Together

The Kimbel Library Computer Availability Maps project involved not only several different library services working in concert, but also several different library staff members. The Web Services and Emerging Technologies Librarian was in charge of the project. He designed the library maps, expanded the original script to include both buildings, mobilized the maps, and created the dashboard script for the help desk monitors. A web design student assistant carried out the map and computer icon designs using graphic design software. A computer science student took some of the code provided by other sources and revised it to suit Kimbel Library’s needs. He also created and coded the administrative console and graphical user interface. Kimbel Library’s information resource consultant (IRC) embedded the Perl script on the machines and performed computer maintenance and troubleshooting operations.

This project may appear daunting to those unfamiliar with coding and programming, especially noting the variety of expertise of the library staff members involved. But a basic application of this project can be created as long as there is an amenable IT staff member
with access to the institution’s database and directory structure. The script will need to be loaded on all of the machines as well. Not much else is needed to create maps tailored to each institution. Some HTML and PHP experience will be helpful in understanding how all of the files fit together and knowing what code to plug in to what area. A bit of knowledge about Adobe Photoshop or Adobe Illustrator is helpful in creating maps and icons.

**Architecture and Implementation**

The following sections describe in greater detail the system design and architecture of the availability maps and the steps involved in its implementation. The mapping system runs in a standard LAMP stack development environment (Linux, Apache, MySQL, PHP). We programmed the application to link to the university’s MySQL database, which displayed information about each computer involved in the mapping system. The most important bit of information in the database was the login/logout Perl script, which switched from 0 to 1 to denote whether machines were available or in-use. PHP was used to display this computer status through the application programming interface (API) and to allow administrative functions through the administrative console. The front-end interfaces were coded using PHP, HTML, and JavaScript. This mashup of web programming, API, database, and scripting resulted in a user interface where patrons could see whether computers were available by finding green icons throughout the library buildings.

To track the operations of each computer, we created a MySQL database that contained each computer’s name, status, type, positioning (relative to the top left of each map), the time of the last update, and location. You will need to create a database with at least these values in order to recreate our availability maps.

Figure 14.2 shows a sample of the structure of the MySQL database. We can see that Mac workstation 10883 has a status of 1 (in-use). Its icon will be located on the image map 609 pixels from the left and 394 pixels from the top. The Bldg column shows that the computer is located on the second floor of the Bryan Information Commons (BIC2).

**Loading Perl Script on Computers**

In order to track the status of each computer, a Perl script was loaded onto each machine’s login and logout scripts. The Perl script sends
HTTP requests to a PHP script, changing when the computer is logged on or off by recording a value of 1 or 0, respectively. These values are stored in the database’s status column, which is queried by the API to determine which icon to show on the map. A red icon shows if the status is 1 (in-use); a green icon shows if the status is 0 (available). These scripts were used verbatim from the Griggs article.

PHP and HTML Coding

Once the scripts were loaded on the computers, we set out to code the functionality of the application. The following section details each of the files created to make the maps work.

First, we needed to establish a database connection so the login/logout scripts could talk to the database and record each computer’s availability status. The _sql_header.php file helped establish a connection to the database. The statuschanger.php file queried the database to set the computer status. The database values were in a constant state of flux, so we also made sure to include a JavaScript refresh on the maps to give the most up-to-date information to patrons. We chose a 2-minute refresh in our map system in order to provide a balance between having real-time data and easing the load of updates to our servers.

User interface functions were created to make for a better user experience. For the interface, we determined that we needed differing floor locations, system metrics, a timestamp, and a symbol key. We identified four different locations for which we needed to create maps for our system: first floor library, first floor commons, second floor library, and second floor commons. A drop-down menu was created to enable switching between these locations; the _maps.php file contains this drop-down code. The Bldg column of our database was queried to assign computers to the correct map, as is seen in the

<table>
<thead>
<tr>
<th>computer_name</th>
<th>status</th>
<th>computer_type</th>
<th>left_pos</th>
<th>top_pos</th>
<th>updated_at</th>
<th>bldg</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIC10883.COASTAL.EDU</td>
<td>1</td>
<td>mac</td>
<td>609</td>
<td>394</td>
<td>2013-07-02 11:14:22</td>
<td>BIC2</td>
</tr>
<tr>
<td>BIC10879.COASTAL.EDU</td>
<td>0</td>
<td>mac</td>
<td>655</td>
<td>341</td>
<td>2013-07-01 09:43:00</td>
<td>BIC2</td>
</tr>
<tr>
<td>BIC10883.COASTAL.EDU</td>
<td>0</td>
<td>mac</td>
<td>575</td>
<td>314</td>
<td>2013-06-24 09:39:22</td>
<td>BIC2</td>
</tr>
<tr>
<td>BIC10885.COASTAL.EDU</td>
<td>0</td>
<td>mac</td>
<td>609</td>
<td>284</td>
<td>2013-06-26 17:59:44</td>
<td>BIC2</td>
</tr>
<tr>
<td>BIC10886.COASTAL.EDU</td>
<td>1</td>
<td>mac</td>
<td>656</td>
<td>230</td>
<td>2013-06-18 16:48:38</td>
<td>BIC2</td>
</tr>
</tbody>
</table>

Figure 14.2  Database table sample
change map .php file. The metrics section shows users the real-time availability of Macs and PCs for each location. The timestamp shows users when the page was last updated. The symbol key shows what each icon represents.

Administrative console functionality was the next component to be coded. The administrative console is an API that was created using AJAX and the jQuery UI library to enable employees to login and drag-and-drop each computer icon to its proper location throughout each map (Figure 14.3). This functionality was designed to avoid much back-end work within the database and allow those without programming skills to change the computer locations in the event that computers needed to be moved to different tables or different rooms. Functionality such as this is also helpful due to the fact that different browsers “read” pixels in different ways. Shifting icons slightly becomes necessary if a browser used in your institution reads the icon positioning slightly different than other browsers. The positioning of the icons on the maps is coded in the file update_map_.php and is reflected in the database columns left_pos and top_pos in real-time.

Other important administrative console functions are contained in the following files:
index.php contains, among other things, the code that links the correct icon to the correct map.

_links.php allows users to log in to the console.

_clear_msgs.php clears messages upon session start or shows an error message if there is a login error.

_protected.php contains authentication code, which lets authenticated users into the console or re-directs unauthenticated users to _links.php options.

admin.php file is the code for the user login screen, which allows users to authenticate to gain access to the console.

login_.php checks the username and password entered in the admin.php page in order to allow entry into the console.

logout_.php allows logout functionality.

create_user.php and user_create_.php allow the site administrator to create new users while also storing username and password information in the database.

Making the Maps

The next step in the project was to create the maps. Test maps were originally created by taking screenshots of building blueprints. The maps were then conceptualized and re-designed by hand for the website pages. At this point, Kimbel Library’s web design student assistant designed the maps in Illustrator. She also designed the icons, incorporating Apple and Microsoft symbols in the icon design to allow users to differentiate between these machines. Icons were placed atop the map using CSS.

Several design decisions were made in the making of these maps. Since the maps were going to be published to a variety of different devices in a variety of sizes, we needed to provide visual markers that would help map viewers orient themselves relative to each location. We ended up using the library help desk, circulation desk, and equipment checkout room as orientation points on the maps. Other important practical objects that we included on the maps were elevators, stairs, vending machines, and restrooms. Maps were positioned to match as
near to true north as was appropriate design-wise. In general, the maps were broader and less detailed design-wise in order to mitigate potential design updates. Room numbers were left blank and non-essential building features were not included.

**Publishing and Display**

Before proceeding with the coding and design of the maps project, it is important to decide in what form the maps will eventually be presented to the public. The Kimbel Library maps were styled for the web and linked to the library website for use on desktop, laptop, and tablet devices. Maps were displayed on monitors above the main library help desk so that patrons could quickly find an open computer when they arrived at the library. We also mobilized the maps for display on mobile devices, so patrons could check the maps before coming to the library to determine if it was too busy to visit. We chose to use the same map for all three types of publication and display: web use, help desk monitors, and mobile devices.

**Webpages**

Patrons can access the API from our library website pages. A map 800 pixels wide by 872 pixels high was created for each floor. Positioning was measured in pixels from the top left of the map. This created a matrix of that same size where the computer icons could be placed. This consistency aided us in using one master map and one master database table to display the same map matrix on several different devices.

**Help Desk Monitors**

Patrons in the library can see computer availability on monitors at the main help desk. We have two LCD monitors above and behind this central information desk, located at the intersection of the library and information commons buildings. The Web Services Librarian created a new dashboard index page for each building, which consists of a JavaScript dashboard script that calls the two main HTML map pages. The script cycles through the floors at 7-second intervals, allowing patrons to get a full view of all computers on both floors in a relatively short time.4

Maps were displayed on the monitors by linking each monitor to a PC through a VGA cable. We had to use the “display full screen”
function and had to decrease the browser window zoom level to 85 percent to enable optimal display on these monitors. Institutions could have the opportunity to choose monitors with the appropriate screen size and resolution based on the size and shape of their specific availability map system.

Mobile Pages
The maps were mobilized by utilizing several mobile optimization steps. The header was not called in the mobile pages. We added style sheet elements to use mobile tabs for navigation as opposed to dropdown functionality. Users could therefore quickly see which computers were available without having to interact with any unnecessary bells and whistles (Figure 14.4). We optimized the site for mobile

![Map optimized for mobile devices](image)

Figure 14.4 Map optimized for mobile devices
devices by linking to a mobile CSS style sheet. CSS media queries seemed like a good idea until we encountered the difficulty of representing pixels differently for the same computer on three different screens. In order to show three different map sizes, we would have needed to build three different database tables to pull data from top and left positions, which was not optimal for our environment. We did use the viewport metatag\(^5\) to control the layout on mobile browsers, specifying a viewport width of 850 in order to maintain consistency across devices. During testing we found that the icons and maps were large enough even in mobile devices with screen widths of 480 pixels, which we felt was acceptable for our needs.

If we had a chance to do this project again, we might have created a series of three maps as others have done: one for large display (900 pixels wide), one for desktop display (600 pixels wide), and one for mobile device display (300 pixels wide). Many institutions mobilize their computer availability maps by porting text-only descriptions to the device screen, but we made a decision that we wanted to replicate the web experience as much as possible across all devices. Our icons were large enough when tested in many iOS and Android devices that we were satisfied with our basic web map design.

**Testing**

Several tests should be performed during a project of this nature. The login/logout script should be tested once the scripts are loaded onto the machines, to ensure that the database is recording and displaying 1s and 0s accordingly. Script values should be reflected in two places: in the database’s status column, which switches to 1 if available (logged-out) or 0 if in-use (logged-in); and in the front-end map interface, which should show the green PC or Mac icon if the status value reads 1 and the red PC or Mac icon if it reads 0.

API functionality should be tested by logging into the administrative console, moving icons with a cursor, checking that the database is recording, and displaying the updated positioning. This should be reflected in the database by the left_pos and top_pos values and in the maps by changing icon positions.

Cross-browser testing should be performed, taking into consideration the technology utilized by each institution’s population. Analytics should be used early in the project to determine which operating systems and browsers will be tested.\(^6\) Mobile device testing
should be performed to ensure users can access the mobile optimized service without too much scrolling, pinching, and squinting.\footnote{7}

Conclusion

Various issues that we encountered during the implementation of this project can be avoided to some extent with diligence and maintenance. Maintenance should be performed on the mashup components at regular intervals. Consult your IRC or systems librarian to set up a predetermined maintenance schedule in order to ensure that the scripts are running properly. The IRC should also check the script functionality whenever computers are re-imaged, for example.

Databases should be optimized, backed up, and free of any inconsistencies such as duplicate records. Refresh scripts should be checked to ensure that pages are not catching, pausing, or freezing during the cycle. Updates to operating systems, browsers, and software should be performed on a regular basis.

Many institutions have a wide variety of computers, which contain different operating systems, browsers, browser versions, software versions, antivirus, spyware blockers, and so on. A myriad of technology issues can cause machines not to send the login/logout status correctly. Scheduled updates and a diligent IT staff can help to mitigate many of these issues.

One issue that Kimbel Library has experienced deals with patron login/logout functions. The library does not require authentication for patrons to use computers. Since patrons do not need to log in to use computers, they do not need to log out either. The default logout event on our computers (when the screensaver takes over) is triggered by a script after 10 minutes of inactivity. Changing this metric may cause problems with students who are sitting at a computer taking notes; they are still at the computer but have not moved the mouse or used the keyboard. Forcing a logout may cause them to lose their unsaved work. The library is still actively working to find solutions to this issue and is contemplating several workarounds due to the fact that we are not authorized to add authentication on any library computers at this time. Each project team must work with their IRC or IT department to figure out the best workflow for their organization’s technology environment and plan their availability map project accordingly.
Maps are helpful because they provide a visual representation to patrons. They help patrons locate library services by visualizing those services. Helping patrons save time by offering services that are accessible online from multiple devices further serves to endear them to the library. Tying several different data points and services together to create a computer availability map mashup is not only fun, but can also contribute to patron happiness and appreciation. Create a mashup at your institution and you should observe a noticeable rise in patron satisfaction.

***

I would like to thank the major coding and design contributors in this project: Joshua Carlson-Purcell, Jordan Ratti, and Jason Leary. Their contributions make up the bulk of this chapter, and anything of worth is most likely theirs. Any mistakes are my own.

Endnotes


2. This moving and shifting of computers actually happened several times during project development. Be sure to save computer locations frequently while in the administrative console. Otherwise the page resets upon refresh, losing any unsaved computer movement.

3. Many institutions include an “offline” designation for computers, which allows the icon to stay in position on the map while signaling to the user that the computer is unavailable, yet is also not in use. We chose to forego using the offline signifier (we have not experienced many offline problems, as we have an extremely responsive IRC dedicated to quickly fixing library computers).

4. See index_dashboard_bic.html and index_dashboard_klib.html in the Kimbel Library code files. Both files contain the full dashboard script.

5. We placed the viewport meta tag code within the head tag, <meta name="viewport" content="width=850" />

6. We used Google Analytics to determine which operating systems and browsers to test on. We also used analytics to track map usage by using a PHP include statement to call an analytics file into several of our map pages.

7. Cross-browser testing was performed on the majority of operating systems and browsers used by our patrons. We tested for mobile functionality using COWEMO’s mobile phone emulator, www.mobilephoneemulator.com