Candidate Cancer Gene Database

Contents

Concept....5
- Who are the users?....5
- How long does the website need to last?....5
- What do you want the website to look like?....5

Platform Options....6

Licenses....6

Risks....7
- Source Data Becomes Outdated....7
- Source Data Format Changes....7
- Data Becomes Damaged During Update....7
- Gene Names Increasingly Become Case-Sensitive....7
- Application Environment Changes....7
- Malicious User Stages Attack....7
- University Requires Changes....8

Website....8
- Search Page....8
  - Banner....9
  - Navigation Bar....9
  - Search Form....10
  - Footer....11
  - University Elements....11
- About Page....11
- Download Page....12
- Help Page....12
- Tutorial Page....13
- Links Page....14
- Contact Page....15
- Search Results Page....16
  - Page....16
  - Export....16
  - Table....16
  - Navigation....17

Parameters....17
No Results....17
Spam....18
Timeout....18
Detail Page....18
Fields....18
No Results....19
Field Explanation Page....19
Fields....19
No Results....20
Study Explanation Page....20
Fields....20
No Results....20
Maintenance Page....21
Login Page....21
Upload Page....22
Access....22
Tables....22
File Field....23
Messages....24
Accessibility....25
Analytics....26
Google Analytics....26
Database Analytics....26
Security....27
Indexing....27
Error Messages....27
Forbidden Access....27
System Calls....27
Passwords....27
SQL....27
Scripting....27
Email....29
Database....29
Tables....29

Routines....29
Optimization....29
Test Environments....31
Production Environment....35
Provision....35
Platform....35
Partitions....35
Packages....35
Privileges....36
Email....36
Storage....37
Connection....37
VPN....37
SSH....39
SFTP....40
Configuration....41
Redirects....41
Indexing....41
Google....42
SSH....46
Apache....51
Permissions....52
MySQL....53
PHP....54
Subversion....54
WinSCP....56
Fugu....58
Cron....58
Strings....58
Sanger....59
PubMed....60
Gene and HomoloGene....61
Testing....62
Files....62
Web and Database Server....62

Google....65
File Upload....67
Update Scripts....69
Error Reporting....69
Accessibility....69
Scheduled Tasks....70
Launch....73
Public Configuration....73
Publicity....73
Indexing....73
Publication....74
Usage Statistics....74
MySQL Service Interruption....75
Sanger....77
NCBI....78
Maintenance....79
Email....79
Automatic Updates....79
Manual Updates....84
Study Information....84
Change Website Password....96
Change Database Password....97

Concept

Who are the users?

Users will be researchers looking to identify common mutations. Users are generally skilled enough to get around a webpage without much difficulty. They will probably be using Macs and (to a lesser extent) PCs. The user base may expand with public access in the future. At this time, there are no core users with special needs that must fundamentally shape the development of the user interface.

How long does the website need to last?

The website may be up for more than 3 years.

What do you want the website to look like?

Styles, colors, and graphics are a low priority at this point. Functionality is important. See www.sanger.ac.uk/genetics/CGP/Census/ and www.sanger.ac.uk/genetics/CGP/cosmic/.

People will want to:

1. look up information by submitting a mouse gene
2. view all information tied to a gene
3. view statistics that aggregate gene information
4. export information to a spreadsheet
5. link to outside URLs for additional information

Must-have features:

1. Input one gene and see if it showed up in any screens (e.g., APC in screens X, Y, Z)
2. Input a whole list of genes at one time and get lists of screens for each gene
3. Input a cancer and see the frequency with which each gene is mutated

Updates from Excel are good.

Platform Options

Surveys suggest that Unix is the most popular website operating system, that Apache is the most popular web server, that MySQL is the most popular web database, that PHP is the most popular server-side programming language, that JavaScript is the most popular client-side programming language, and that jQuery is the most popular JavaScript library, and that Google Analytics is a common analytics package. Basic HTML, CSS, and JavaScript technologies have all been out for more than a decade. Some platform decisions will be influenced by hosting constraints.

Licenses

Some of the HTML, CSS, and JavaScript for this site comes from elemisfreebies.com. The About page contains a link back in accordance with the elemisfreebies.com license. The site uses the jQuery JavaScript library, which is free under the MIT license. The site is managed using Dreamweaver, which is proprietary; however, finished code may be edited in other HTML WYSIWYG editors. The AnyConnect VPN client available from OIT is proprietary, but it is possible to configure VPN connections using native software on many operating systems. The PuTTY SSH client and TortoiseSVN client are also free.

I developed with WAMP stack, which is free for use under the GPL license. This included Apache, MySQL, PHP, phpMyAdmin, and webgrind. The MySQL installation included the MySQL connector and MySQL for Excel. ActiveState Perl, gzip, CollabNet Subversion, and GIMP were free. The LAMP stack contained free Linux variants of these programs running in an Ubuntu distribution. The Linux environment additionally contained Postfix, mailutils, and Samba.

Some sites require specific acknowledgements, citations, or links to disclaimers as shown below. The database currently uses only NCBI and Sanger sources but has links to Ensembl, GeneCards, and UCSC.

Site	URL Parameter Links	Data	Contact
Ensembl	Permitted.	Not used. Logo use permitted.	helpdesk@ensembl.org
FlyBase	Not used.	Permitted	Form at http://flybase.org/cgi-bin/mailto-fbhelp.html
GeneCards	Permitted.	Not used.	info.yeda@weizmann.ac.il
HUGO	Not used.	Permitted with specific citation.	hgnc@genenames.org
MGI	Not used.	Permitted with specific citation.	Form at http://www.informatics.jax.org/mgihome/support/mgi_inbox.shtml
NCBI Gene and HomoloGene	Not used.	Permitted with link to Disclaimer and Copyright Notice. Logo use permitted.	info@ncbi.nlm.nih.gov
NCBI PubMed	Permitted with link to Disclaimer and Copyright Notice.	Not used.	info@ncbi.nlm.nih.gov
Sanger CGC	Not used.	Permitted with specific acknowledgement. Logo use permitted.	copyright@sanger.ac.uk
Sanger COSMIC	Not used.	Permitted with specific acknowledgement.	copyright@sanger.ac.uk
SGD	Not used.	Permitted.	Form at http://www.yeastgenome.org/cgi-bin/suggestion
UCSC	Permitted.	Not used.	genome-www@soe.ucsc.edu
ZFIN	Not used.	Permitted.	zfinadmn@zfin.org

Risks

Source Data Becomes Outdated

Newly published information may render database information obsolete. This is largely mitigated by automatic update processes.

Source Data Format Changes

Format changes in external data may interfere with update processes. Risk is probably reduced by the likely low frequency of such reformatting. These events can probably be handled with additional website and database development. One exception would be the introduction of spaces into gene names; the current website depends upon the use of the space as a delimiter.

Data Becomes Damaged During Update

Data may become damaged during updates. This can be mitigated with backups and error-trapping code for update processes.

Gene Names Increasingly Become Case-Sensitive

A few gene symbols appear to be case-sensitive duplicates for the same case-insensitive string. This can be mitigated by collation changes to database tables, but there will be a usability cost.

Application Environment Changes

The website may become inoperable due to server environment changes (e.g., operating system, web server, MySQL server, PHP engine, Perl engine). Commands may become deprecated. This may require website and database development work.

One specific risk involves deprecation of `mysql` commands in PHP version 5.6, which will likely come out within a few years. It appears that PHP version 5.6 will replace `mysql` commands with `mysqli` commands. Unfortunately, it isn't possible to use `mysqli` commands and still retain capacity for record set modification using the Dreamweaver database feature set, which has the advantage of promoting security and reliability in record set construction. Should conversion to `mysqli` commands be necessary, there is a conversion tool available that can make the changes in a matter of seconds.

Another risk involves Adobe plans to drop database features from Dreamweaver. The PHP code in place would still execute properly but would not be as easily editable within Dreamweaver. There may be extensions available, but these may have a relatively short lifespan. Web development frameworks such as Yii and CodeIgniter also have a questionable lifespan.

The website could become inoperable due to client environment changes involving the use of Microsoft Excel. The website requires and accepts only uploaded spreadsheets in the Excel 1997-2003 `.xls` BIFF8 format; it does not accept spreadsheets in the Excel 2007 `.xlsx` XML format. The 1997-2003 format should remain readable for some time; even Excel 2013 supports reading and writing Excel 5.0 files from 1995. By the time a future version of Excel stops supporting `.xls` files, there may be a widely available PHP or Perl utility for parsing `.xlsx` files with tab-delimited `.txt` output. If not, future versions of Excel may still support manual output into tab-delimited `.txt` files.

Malicious User Stages Attack

Sanitization of user input is necessary due to potential for SQL injection and cross-site scripting attacks. This can involve statement preparation, tag-stripping, and HTML encoding. Errors need to be suppressed and access needs to be forbidden for sensitive files. Search engine indexing should exclude administrative assets. The webpage and database login and password should be reasonably secure. Scripts called with shell functions should feature hard-coded parameters. The server should provide some notification of brute force attacks.

Should an attacker gain access to the website or database, they should not discover any sensitive information such as medical or financial information; the website should not be gathering any such information.

University Requires Changes

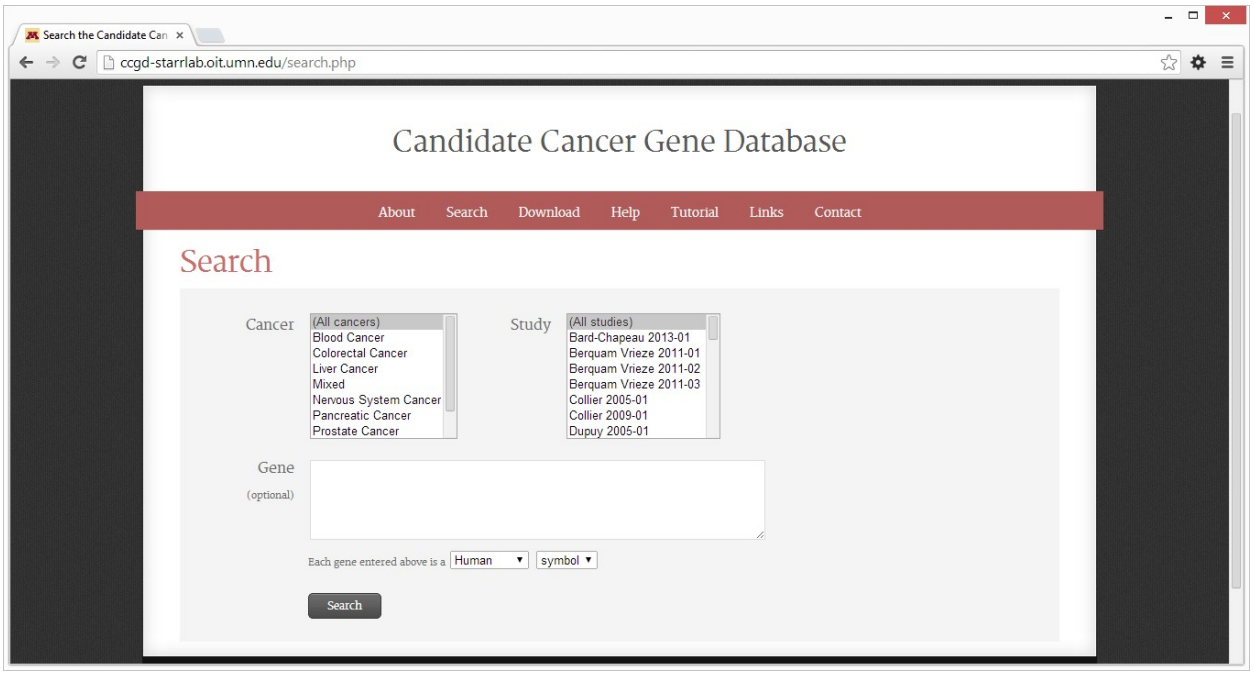
The University has requirements that may change from time to time and may be posted on the brand blog. This blog has a feed that can provide update notification. Some changes may require significant development.

University templates validate as XHTML documents, though XHTML was superseded by HTML5; should the University require XHTML doctypes, Dreamweaver can quickly make the conversions with the File > Convert menu item.

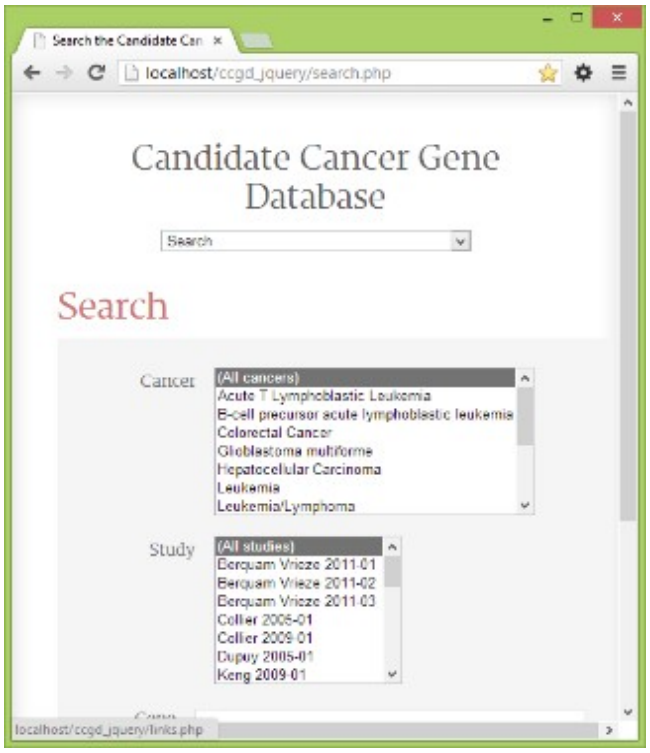
Website

Search Page

The search page contains a banner, navigation menu, and a search form. The search button is visible without scrolling in a window with 768-pixel height.



The search page responds to resizing.



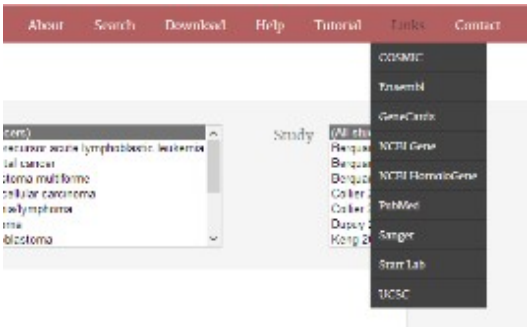
The search form container has a minimum width that prevents alignment problems associated with very small window sizes.

Banner

The search page includes a banner. The banner includes the website title. Changes to `includes/banner.php` ripple throughout the entire website.

Navigation Bar

The menu includes drop-down navigation. Top-level menu items replace the currently loaded page. Dropdown items open in a new browser window or tab.



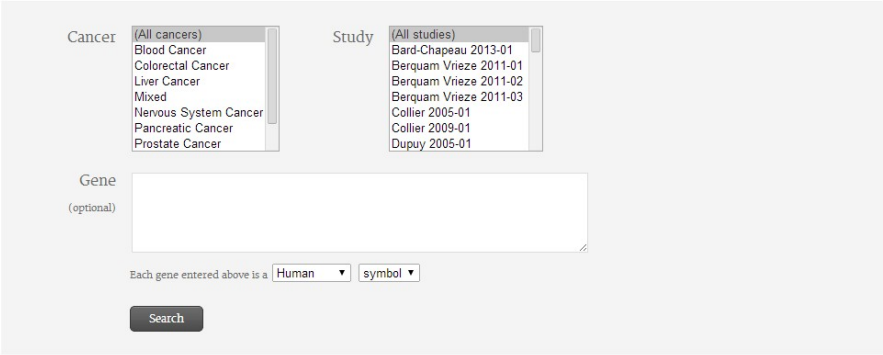
Changes to `includes/navbar.php` and `includes/navbar-links.php` ripple throughout the entire website.

A sticky “Back to Top” button appears at the bottom of the window when the user has scrolled down far enough to lose sight of the navigation bar.

Search Form

Multiple Selection and Dropdown Fields

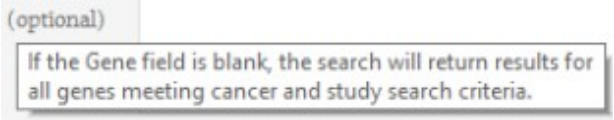
The server automatically generates field options using database queries. `Cancer` and `study` dropdowns have "All" options, but `organism` and `identifier` dropdowns do not. The `organism` dropdown defaults to "Human" but will be treated as "Mouse" unless text is entered into the gene field. The `organism` dropdown uses colloquial English names except for Drosophila. The `label_type` dropdown defaults to "symbol."



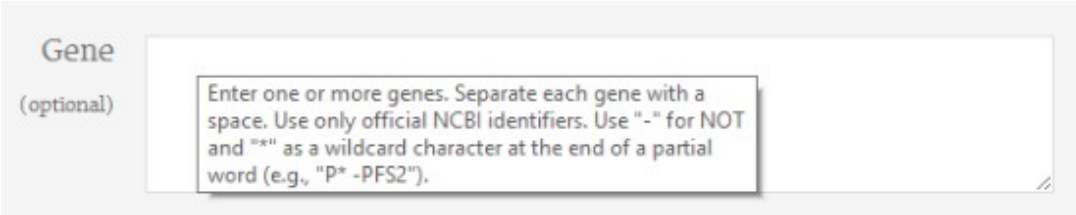
The `study` field filters values based upon the selections made in the `cancer` field. For this filtering to work properly, the sum of all lengths for all cancer names should not approach 2,000 characters. In development, a comma-separated list of all cancer names was only 167 characters long. This feature can be disabled by commenting out the `$('#study').html(var_study_options);` and `$("#study_concat").val("(All studies)");` lines in `search.php`.

Tooltips

The "(optional)" text beneath the gene field label has a tooltip which explains field function.



The `gene` field has a tooltip which explains the use of Boolean operators. Search results will require the inclusion of each term typed into the gene field, but the use of "-" will prohibit inclusion of a term in search results. The use of "*" as a wildcard after at least one character will return multiple results. The search field ignores a query consisting only of a "*" or "-" character.



The search form fields contain other tooltips.

- Choose one or more tumor types.
- Choose one or more studies.
- Choose an organism.
- Choose a gene identifier.

Query Processing

JavaScript code strips leading and trailing spaces, tabs, carriage returns, and newline characters before form submission. The form sends variables using the `POST` method instead of the `GET` method. The `GET` method would have allowed for bookmarking of results pages through URL variables, but this would have imposed a limit of approximately 2,000 characters for some browsers.

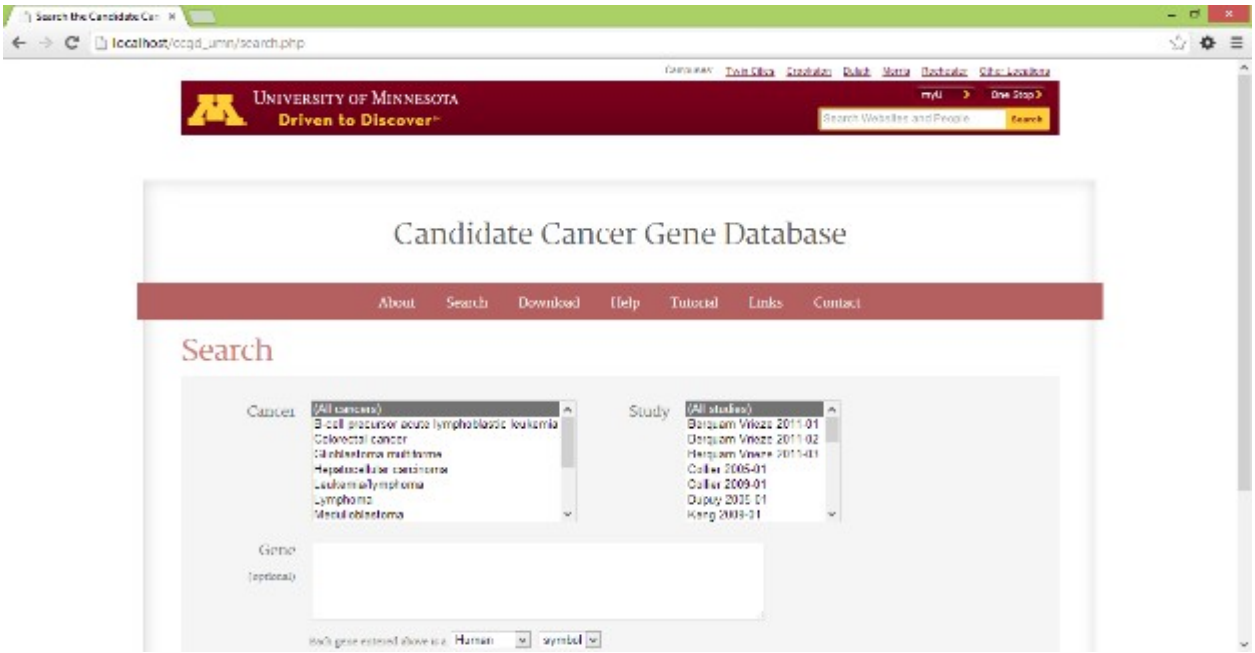
Footer

The footer contains a copyright notice with a year that automatically updates. The footer also contains a University hyperlink, a hyperlinked email address, and a hyperlink to a University privacy statement. The footer email address is obfuscated with JavaScript in order to reduce spamming.

Changes to `includes/footer.php` ripple throughout the entire website.

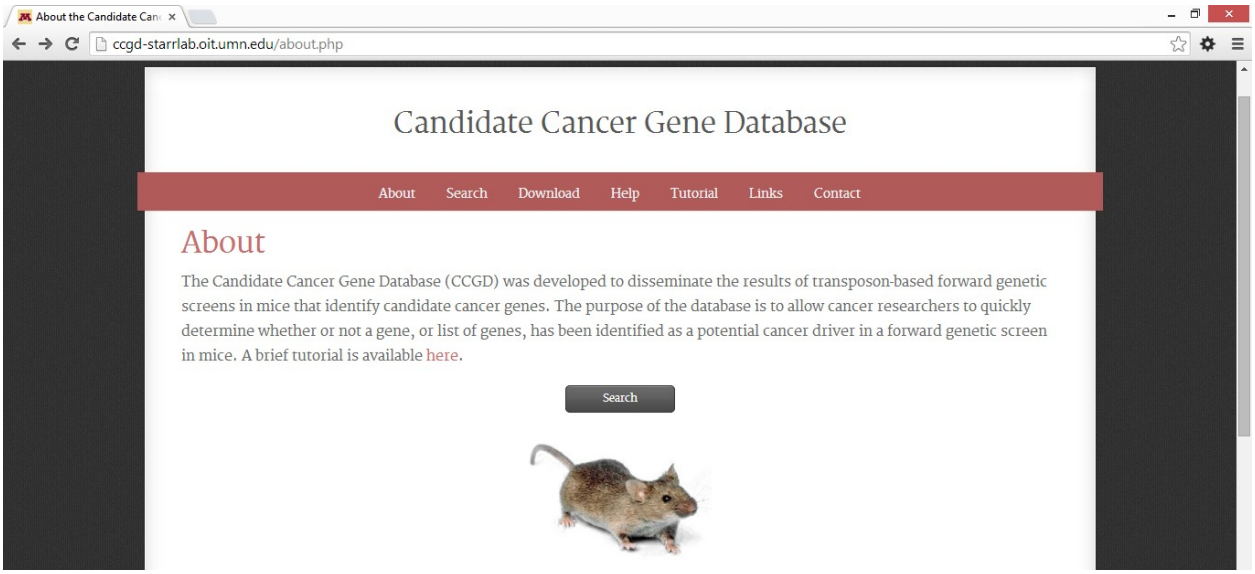
University Elements

The `includes/university_switch.php` file contains a `$university_header_footer` variable that can turn University headers and footers on or off. When using the University header and footer, the grey background will disappear. The University template automatically produces the correct copyright year.



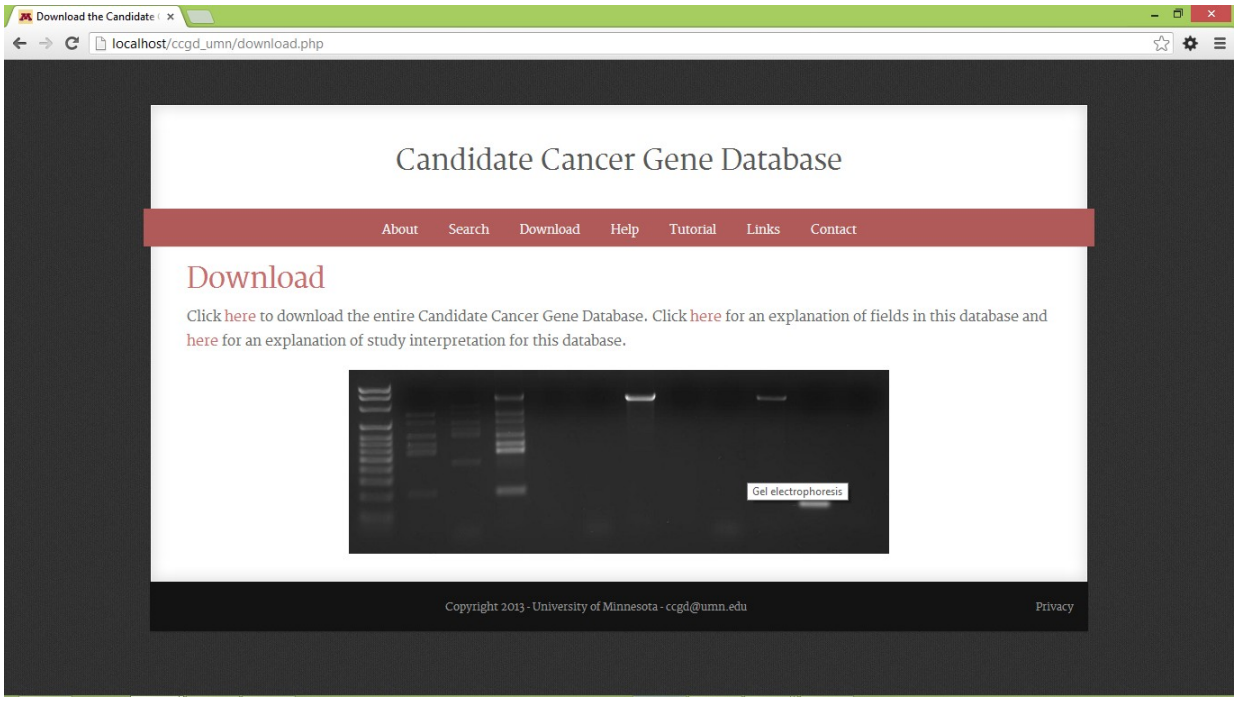
About Page

The about page is the default site webpage specified in the redirect in `index.php`. The about page contains an explanation of the database and a link to the search page. The about page also shows automatically updated information about update processes, mouse gene symbols, studies, tumor types, and queries. The about page includes an example reference citation and other required acknowledgements.



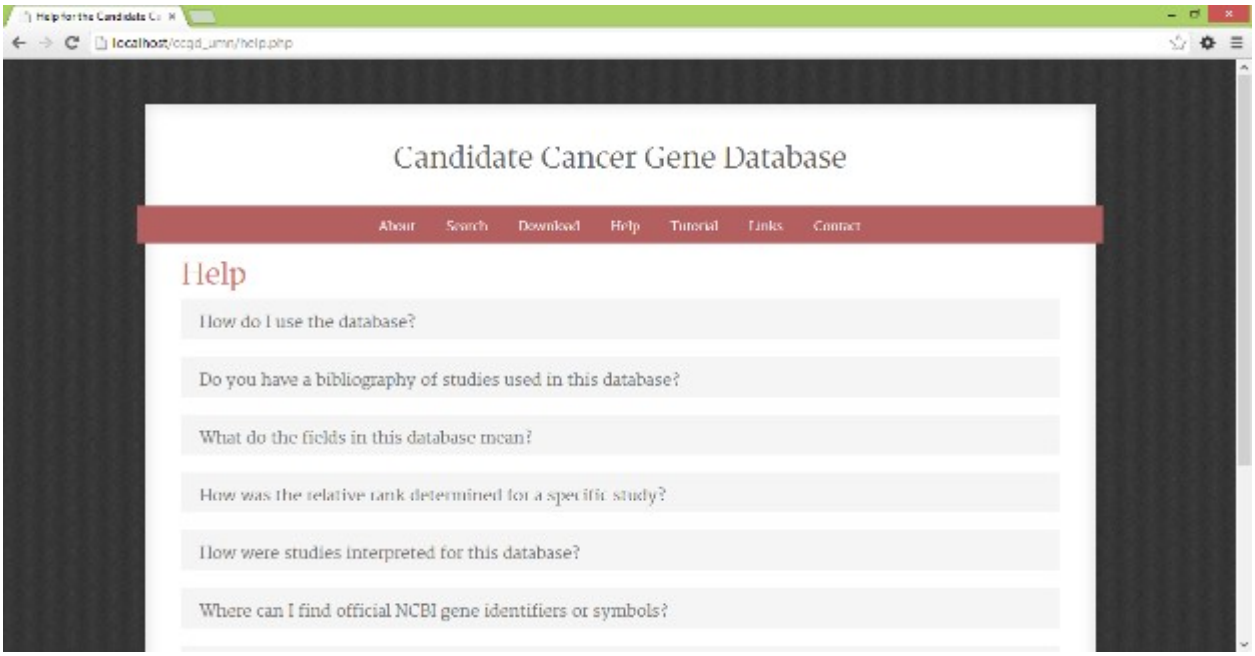
Download Page

The download page provides a link for downloading the entire database. The download page also contains links to field and study explanation. The download page includes an electrophoresis image for interest.



Help Page

The help page includes expandable tabbed panels which provide information on specific questions.



The panels expand with a click.

All the panels expand with a click on the “Help” heading.

Tutorial Page

The tutorial page includes tabs that trigger display of step-by-step guides.

Links Page

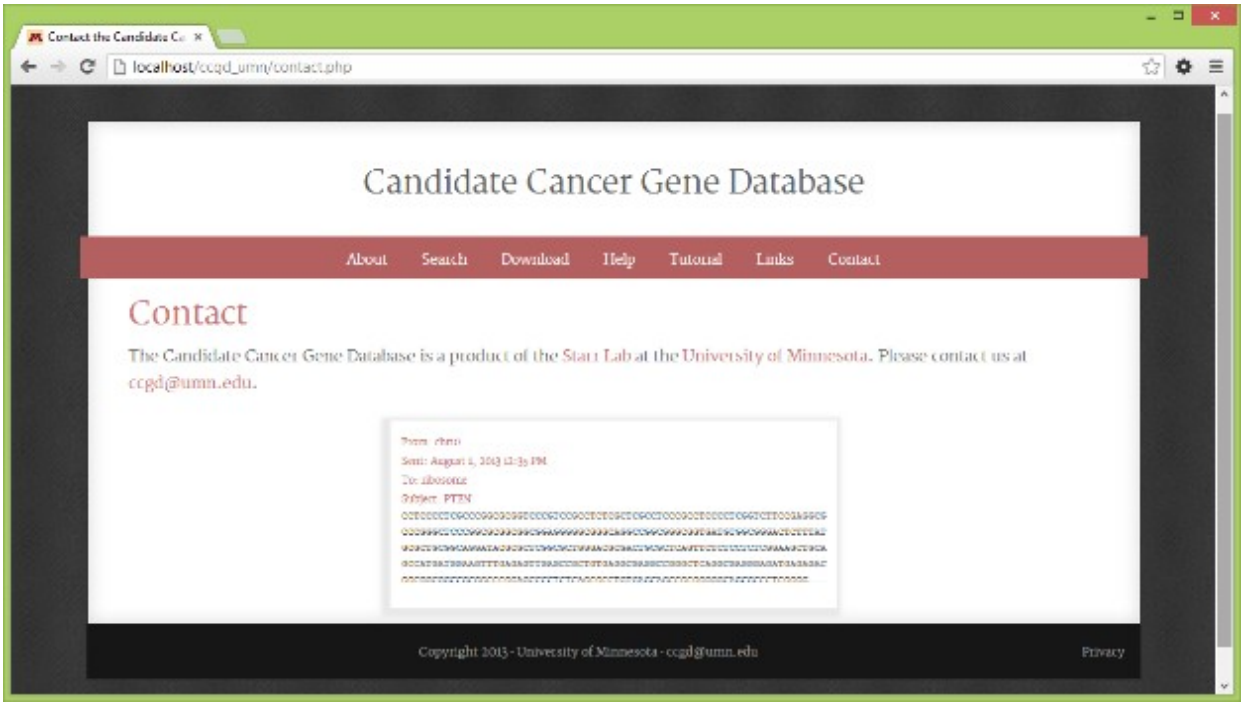
The links page provides images to accompany hyperlinks from the Links menu.

Images are 60 pixels tall except for the NCBI image. The NBCI image is taller than it is wide, and its font would be very small in an image only 60 pixels tall.

The University has requirements governing the use of website images and it is possible that someone will object to the use of third-party logos on this Links page. If this happens, it's possible to wrap most of content in `<!-- -->` comment tags and uncomment the PHP include which will echo items from the Links menu.

Contact Page

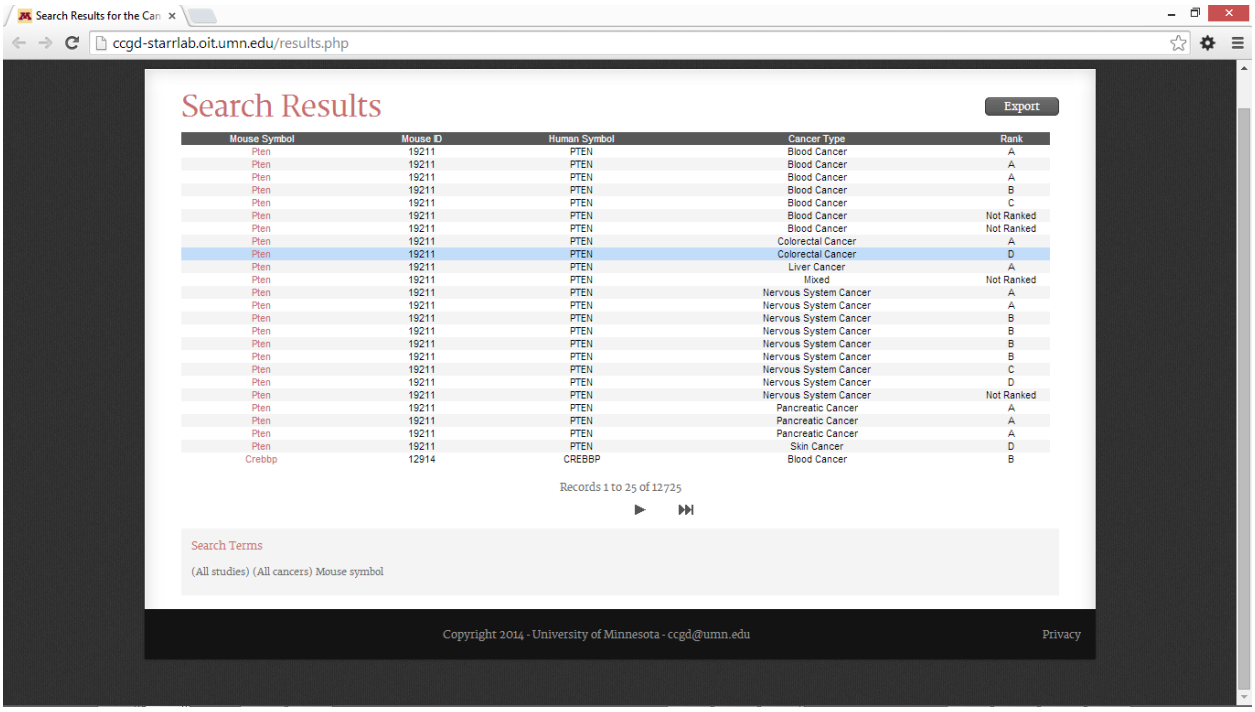
The contact page contains hyperlinks and an email address. The contact page includes part of the PTEN mRNA sequence for interest.



Search Results Page

Page

Search results open in a new tab. The search result page includes a table, record numbers, paging arrows, an export button, and search parameters. The width of the table area has a maximum that prevents some sizing problems and provides continuity for printing.



Export

The search result page includes an export button.

If the user clicks on the export button, two other buttons fade in to offer specific file formats.

The filenames are `CCGD_export.csv` and `CCGD_export.bed`. BED files can be used in genome viewers and CSV files open automatically in Excel. CSV files may also be parsed more easily than `.xls` spreadsheets or XML-based `.xlsx` spreadsheets.

Table

The table text appears in the Arial font. The Arial font is commonplace and may be easier to read than monospace fonts or fonts with serifs.

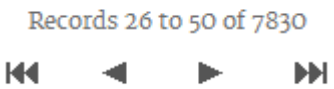
Columns include `mouse_symbol`, `mouse_id`, `human_symbol`, `cancer_type`, and `relative_rank`. These fields will generally fit into the table even when full of large characters (e.g., `M` or `W`). Any `mouse_symbol`, `mouse_id`, or `human_symbol` fields exceeding 15 characters are shown truncated at 12 characters with an ellipsis. Entries in the `cancer_type` field will be similarly truncated if they exceed 30 characters. The `relative_rank` field is small enough that it does not require truncation.

The table includes alternating striped rows to improve readability when traversing across the screen. The row with a mouse pointer highlights in light blue.

Each mouse symbol and table row hyperlinks to a gene detail page that opens in a new browser tab or window.

Navigation

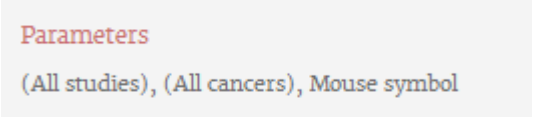
The search result page includes record numbers and page navigation buttons.



The first, previous, next, and last buttons are spaced to reduce the potential for errant clicks. The button image colors match other website styles.

Parameters

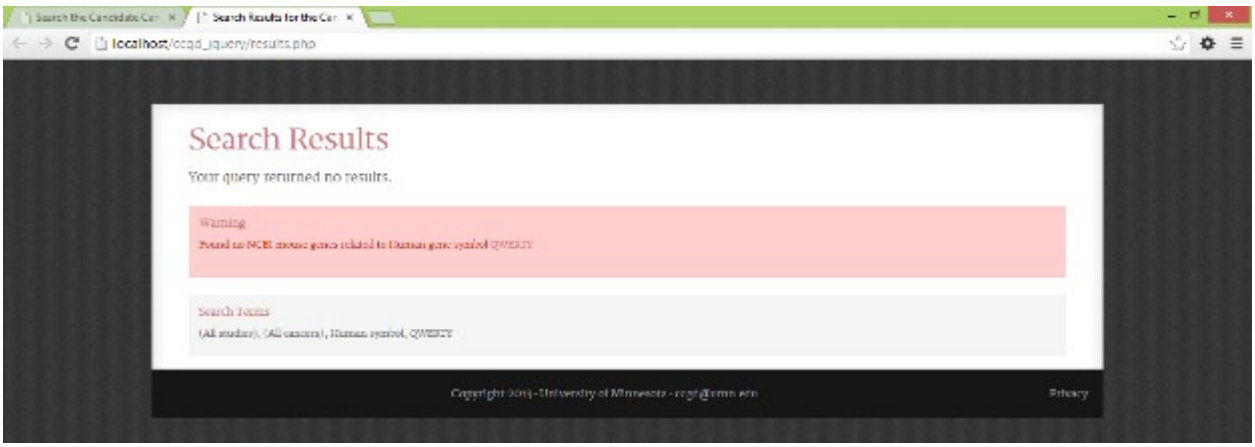
The search result page includes search parameters. Search parameters appear at the bottom of the page in order to minimize table navigation problems.

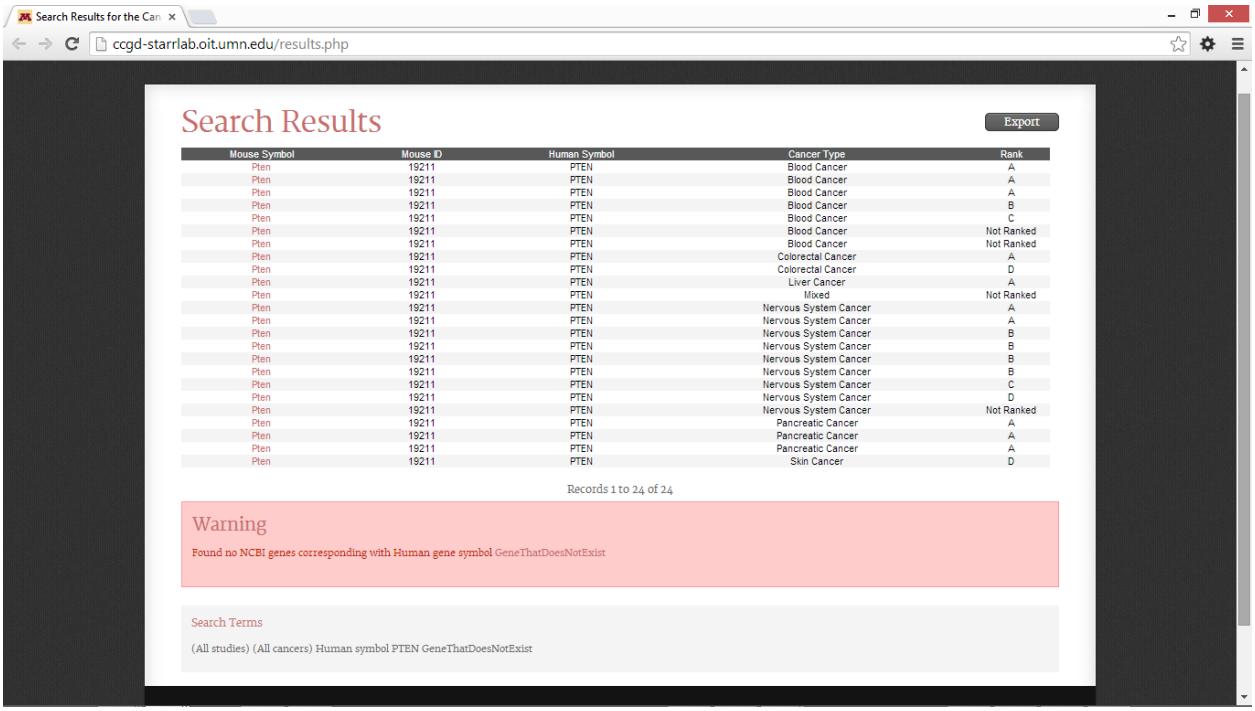


No Results

The results page notifies users if their query returns no results.

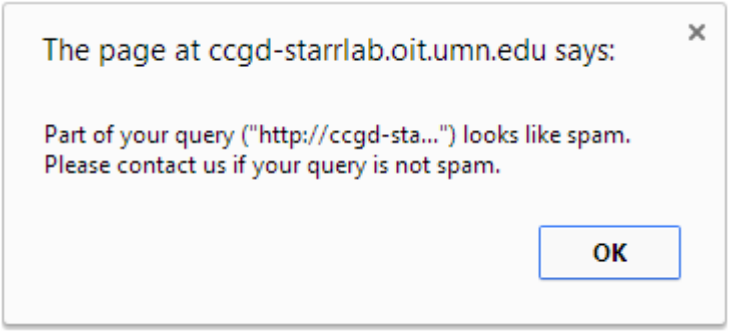
The results page notifies users if any of their query terms appears invalid. Each potentially invalid term hyperlinks to an NCBI search page.





Spam

The results page redirects to the search page if it detects `http://` in the query text. The results page provides a JavaScript popup for human users.



The results page does not record queries that appear to be spam.

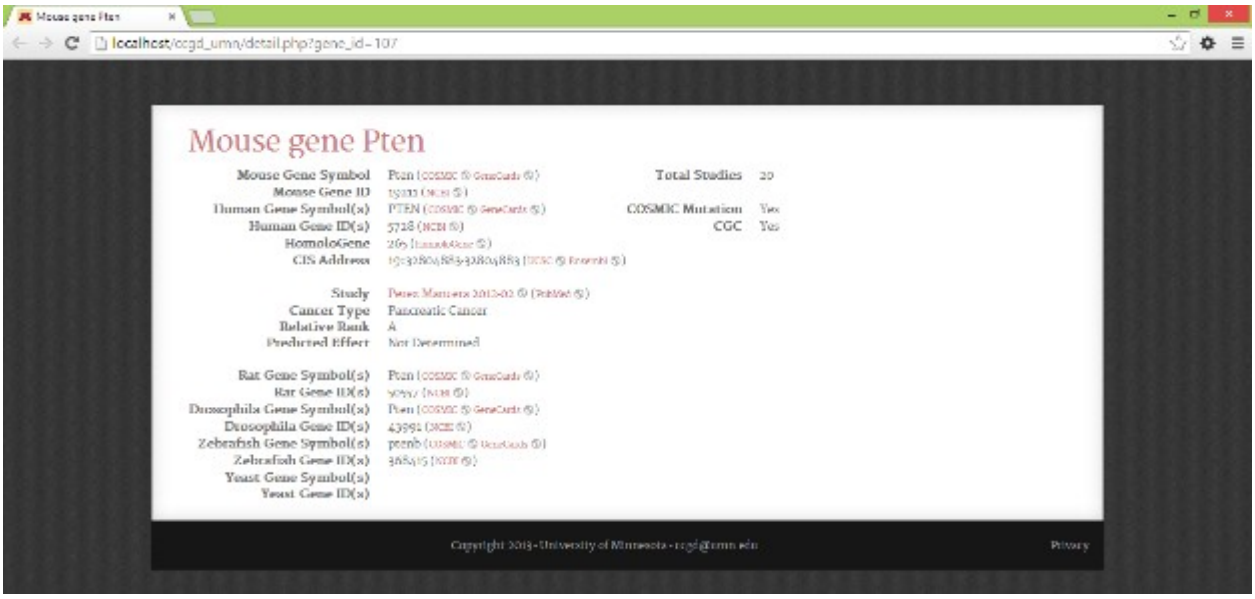
Timeout

The results page times out at 120 seconds. This is longer than the usual limit of 30 seconds and should allow for execution of large queries.

Detail Page

Fields

The gene detail page shows information not fitting on the search results page (e.g., concatenated `zebrafish_symbol` entries exceeding 2000 characters).



Genes with an NCBI ID hyperlink to COSMIC, GeneCards, and NCBI. Genes with a HomoloGene ID hyperlink to HomoloGene. Genes with a CIS address hyperlink to UCSC and Ensembl. Links to external sites appear with small globe icons. Each field label hyperlinks to a field explanation page.

No Results

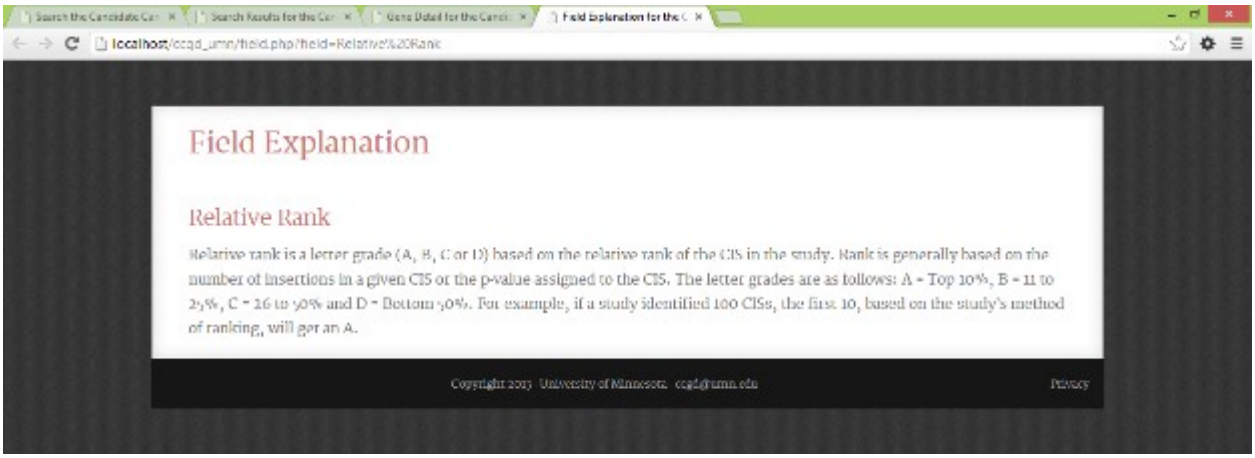
The detail page informs the user if their query returns no results.

The user should see this page only if they manually modify a gene URL in their browser window.

Field Explanation Page

Fields

The field explanation page provides descriptions for each field on the detail page.



No Results

The field explanation page informs the user if their query returns no results. The user should see this page only if they manually modify a field URL in their browser window.

Study Explanation Page

Fields

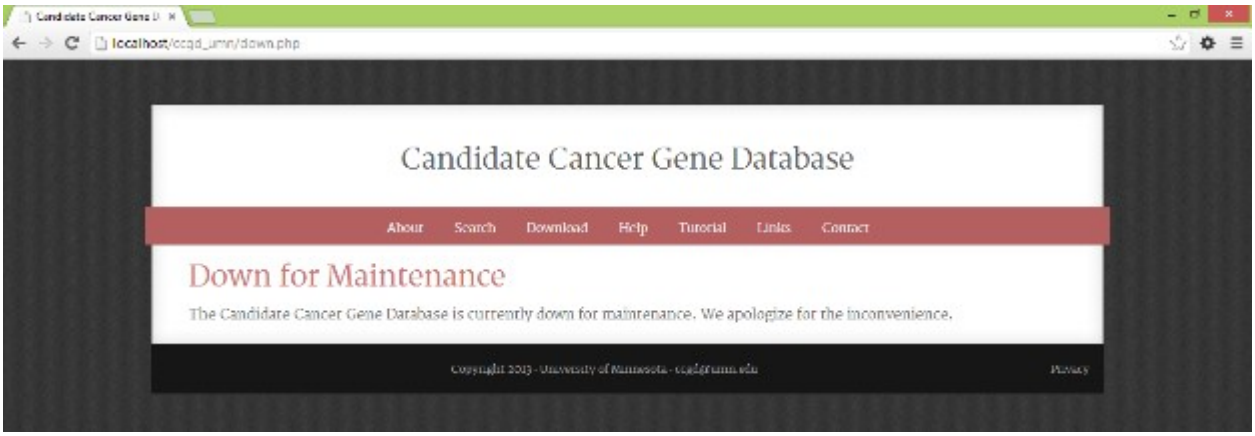
The study explanation page provides descriptions of study interpretation.

No Results

The study explanation informs the user if their query returns no results. The user should see this page only if they manually modify a study URL in their browser window.

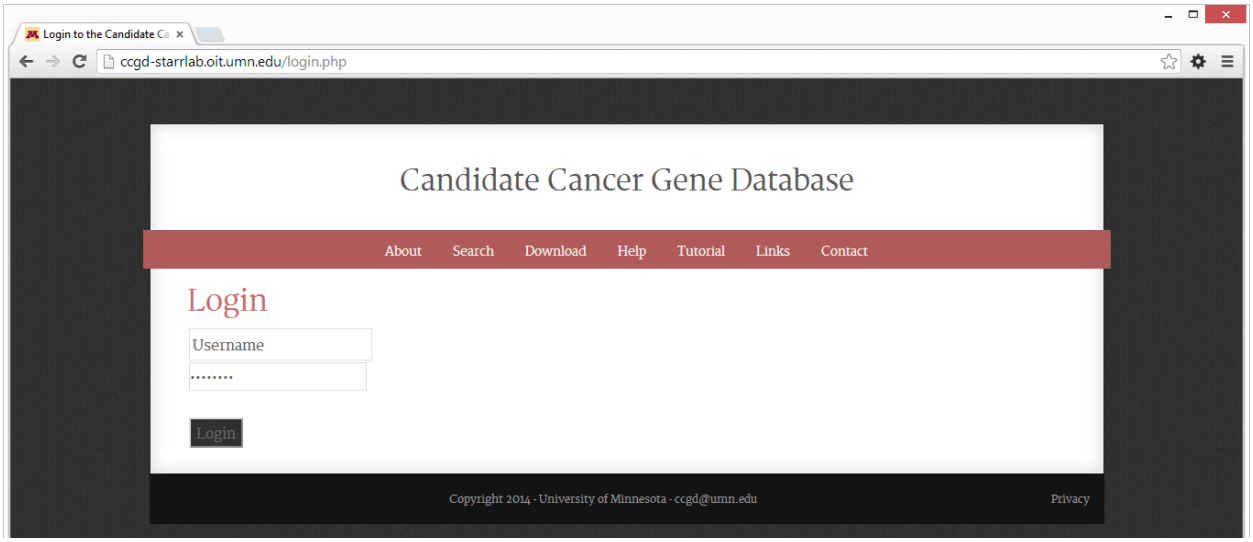
Maintenance Page

Uncommenting the `meta` tag in `includes/error.php` will force all pages to redirect to a maintenance page.

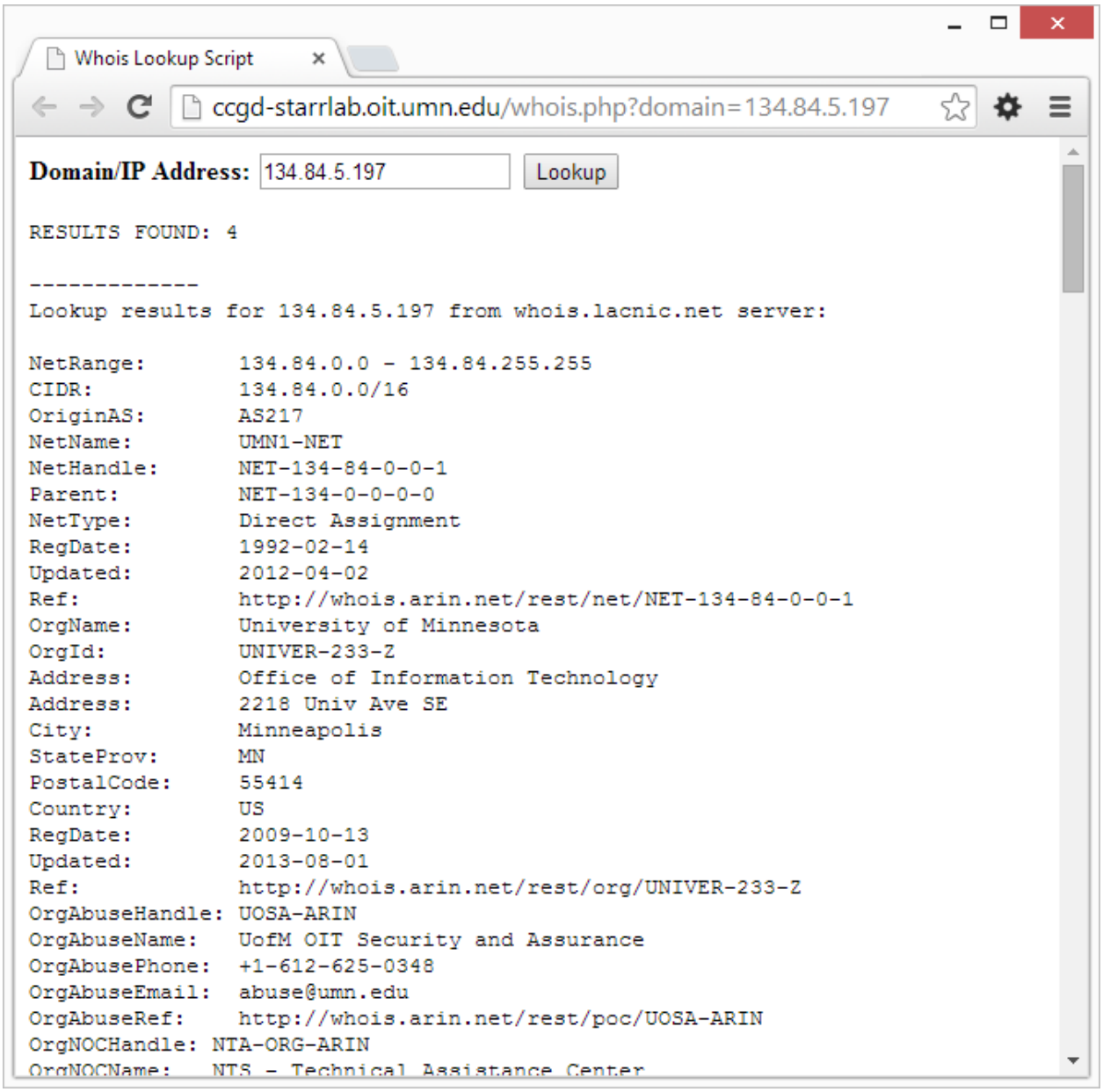


Login Page

The login page controls access to an upload page.



The login page is not accessible by hyperlink; the user must type `login.php` into the URL bar. If login succeeds, the server redirects the user to an upload page. If login fails, the website generates a notification email and sends the user to the site index page. The notification email includes a hyperlink to a `whois` domain lookup page.



The session variable for this login is destroyed as soon as the user closes their browser window. The login page (which controls access to the upload page) can be disabled by uncommenting the redirection at the beginning of `login.php`.

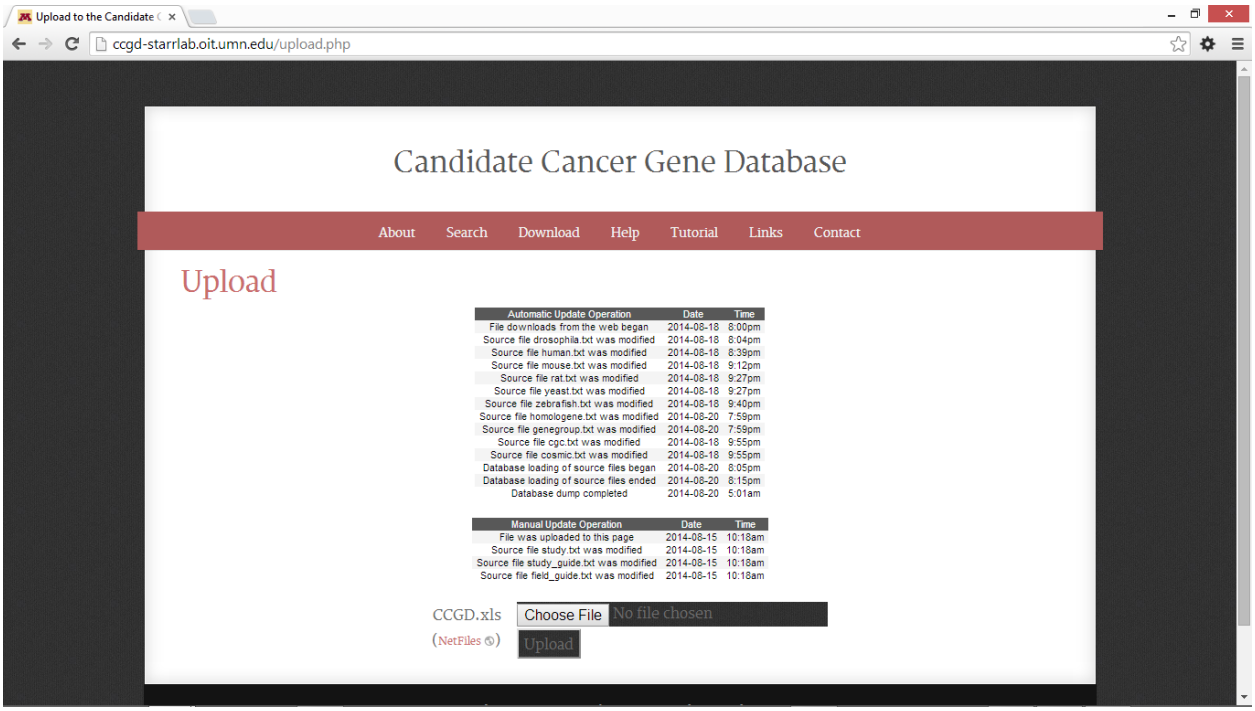
Upload Page

Access

There are no hyperlinks pointing to the upload page. The upload page accepts users logging in through `login.php` and redirects other users to the site index page.

Tables

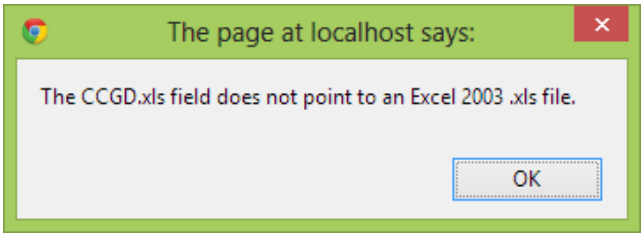
The upload page contains tables with diagnostic information about database updates.



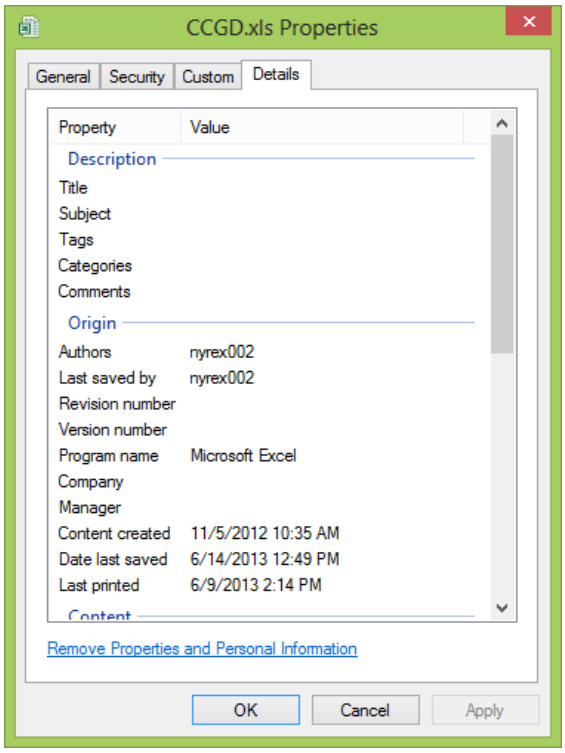
A date discrepancy in the first table not accounted for by a file upload indicates a malfunction with automatic updates. A date discrepancy in the second table indicates a problem with file upload.

File Field

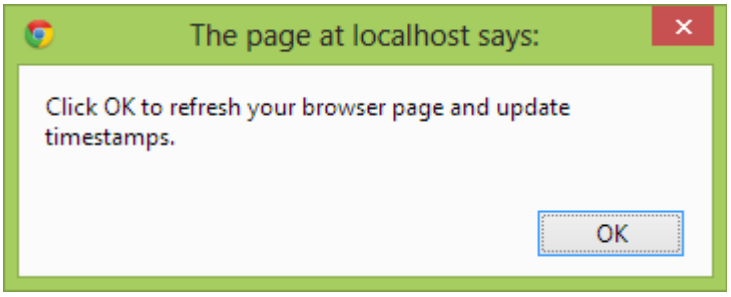
The space below the file upload field label contains a link to the CCGD NetFiles share. The file upload field generates a file browsing popup. The file upload field accepts only an Excel 2003 `.xls` file.



The uploaded file should bear author information.



The upload page logs events in `ccgd_log` and generates an email addressed to the persons specified in `email.php`. If the upload is successful, the upload page starts the `autoload.php` script described in the Automatic Updates section of this document. If the copy of the upload to NetFiles fails, the upload page generates an email. Finally, the upload page asks for a browser refresh in order to display updated timestamps.

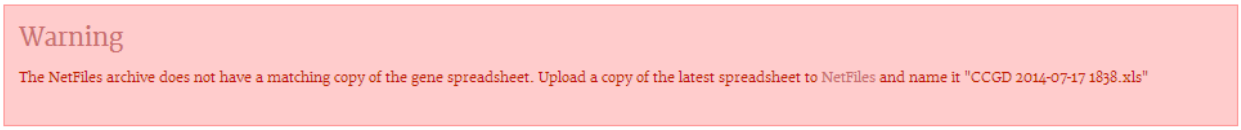


Messages

The upload page provides warning messages instead of a file field when tasks are already in progress.

- Wait until at least XX:XXam/pm to upload data. Web download processing takes about X hours.
- Wait until at least XX:XXam/pm to upload data. Database loading takes about X minutes.

The upload page provides a warning if the spreadsheet on the CCGD server doesn't have a copy in the NetFiles archive.



The website ignores gene symbols in the spreadsheet when a gene ID is present. However, the upload page also provides a warning when a symbol in the spreadsheet doesn't match the official symbol corresponding with a given gene ID.

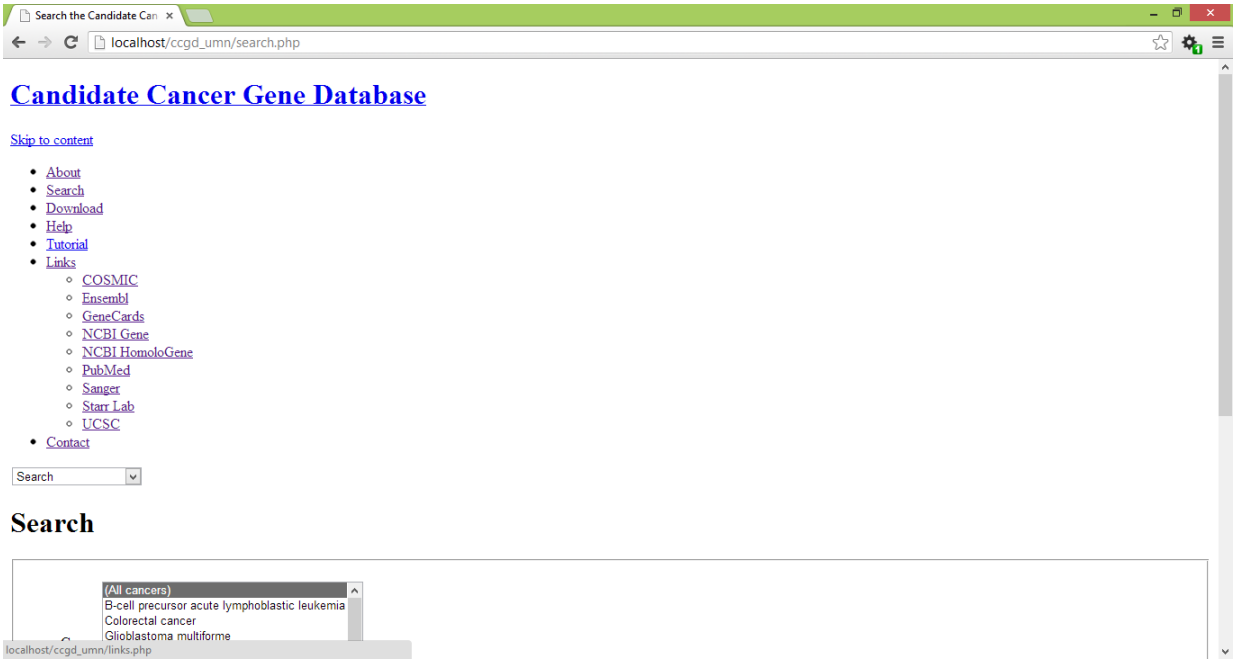
The upload page provides a similar warning if it cannot associate a study name with a PubMed ID or if it detects invalid entries for `predicted_effect` or `relative_rank`.

Accessibility

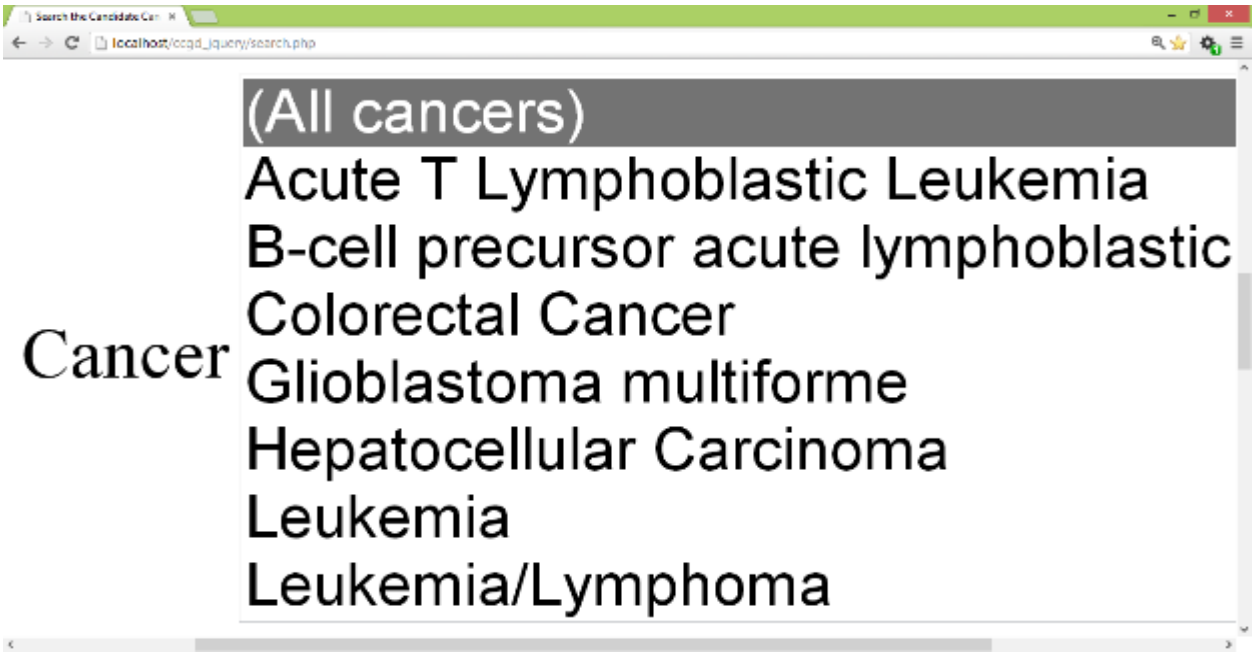
The HTML, CSS, and JavaScript base required some changes for compliance with University accessibility standards. Form fields, buttons, email addresses, and links required `title` attributes for tooltip production. New images required `alt` attributes. Navigation bar elements required `role="header"` attributes. The navigation bar required a hidden "Skip to content" link. Data tables required header tags and a `tabindex` attribute. Fonts required definition in terms of ems instead of pixels.

All clickable controls are activation are accessible with keystroke navigation.

The site remains functional with linked style sheets disabled.



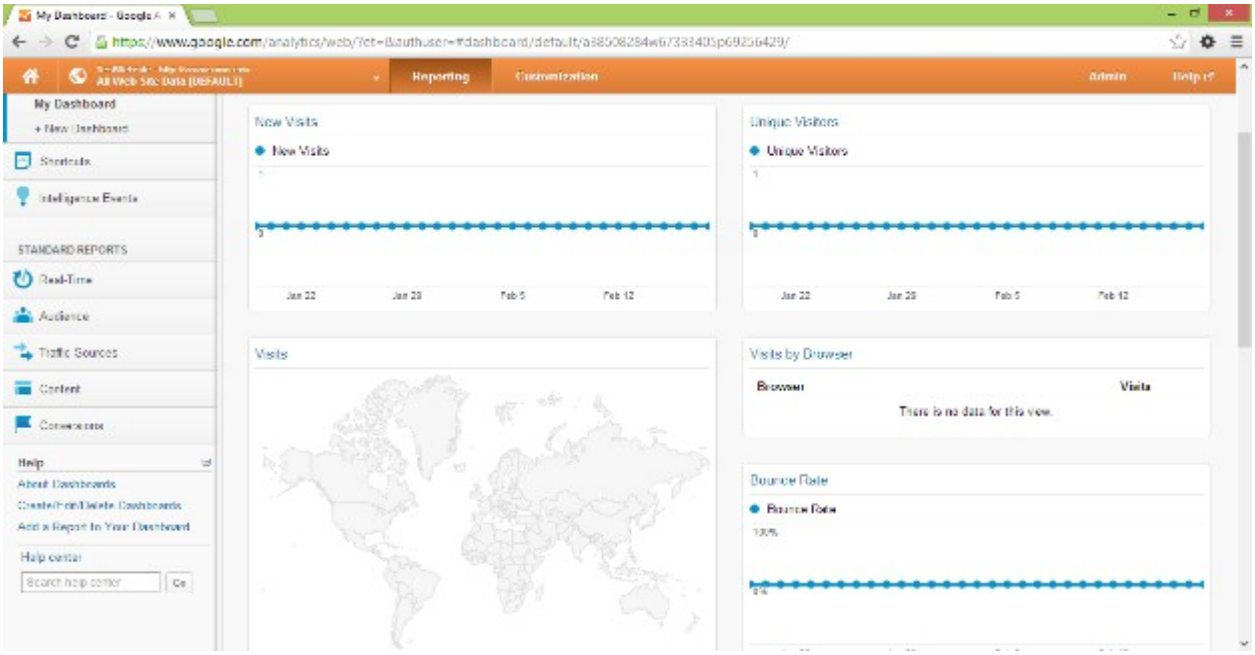
The site remains functional with linked style sheets disabled and 500% magnification.



Analytics

Google Analytics

Each page contains analytics tagging which triggers a call to Google Analytics.



Database Analytics

The database stores query information not captured by the database. The `ccgd_query` table stores timestamps, IP addresses, and session IDs as well as query text for cancers, studies, and genes. The `query_view_cancer`, `query_view_study`, and `query_view_gene` views show, in descending order, the most popular query text for cancers, studies, and genes. The `query_view_gene_length` view gives minimum, average, and maximum

character length for the gene query. The `query_view_pages` view gives minimum, average, and maximum results page number (obtained through use of First, Previous, Next, and Last buttons). The `query_view_ip` view gives distinct IP addresses. The record sets from `query_view_cancer`, `query_view_study`, and `query_view_gene` are not web-accessible because of the need to protect user privacy. The `query_view_gene_length` and `query_view_pages` views are not web-accessible because of their limited value when isolated from `query_view_cancer`, `query_view_study`, and `query_view_gene`. The `query_view_ip` view is not web-accessible because distribution of user IP, country, region, city, latitude, longitude, ISP, and domain would raise privacy concerns; much of this information is available through geolocation utilities such as http://software77.net/geo-ip/multi-lookup/ or http://www.ip2location.com/free.asp.

Security

Indexing

The login page and sensitive directories are excluded from search engine indexing through the use of a `robots.txt` file. There are no hyperlinks pointing to the login page.

Error Messages

PHP errors are suppressed in production to conceal information about server and database structure.

Forbidden Access

Through the use of `.htaccess` files, users are not be able to directly view information such as included PHP files containing database connection information or other information about application structure and function. The `.htaccess` file in the site root directory contains a bot blacklist from http://perishablepress.com/2013-user-agent-blacklist/.

System Calls

Scripts called using a shell function feature hard-coded parameters which are invulnerable to injection attacks.

Passwords

The MySQL server root password is a random string from www.grc.com/passwords.htm. The server provides email notification of unsuccessful login attempts to login.php. MySQL server and NetFiles passwords are stored only in server-side code.

SQL

The current database uses Adobe's method of statement preparation in order to reduce the probability of a SQL injection attack. No user-entered data is incorporated into a SQL statement unless it passes statement validation. The SQL Inject Me plugin for Firefox reports no problems with `search.php`. The SQL Inject Me plugin reports HTTP redirection failures for `login.php`, but the redirects are all intentional.

Scripting

No user-entered data is reproduced on a webpage unless it passes through `strip_tags` and `htmlentities` commands. The XSS Me add-on for Firefox reports no problems with `login.php`. The XSS Me add-on reports a DOM failure for `search.php` using an XML string.

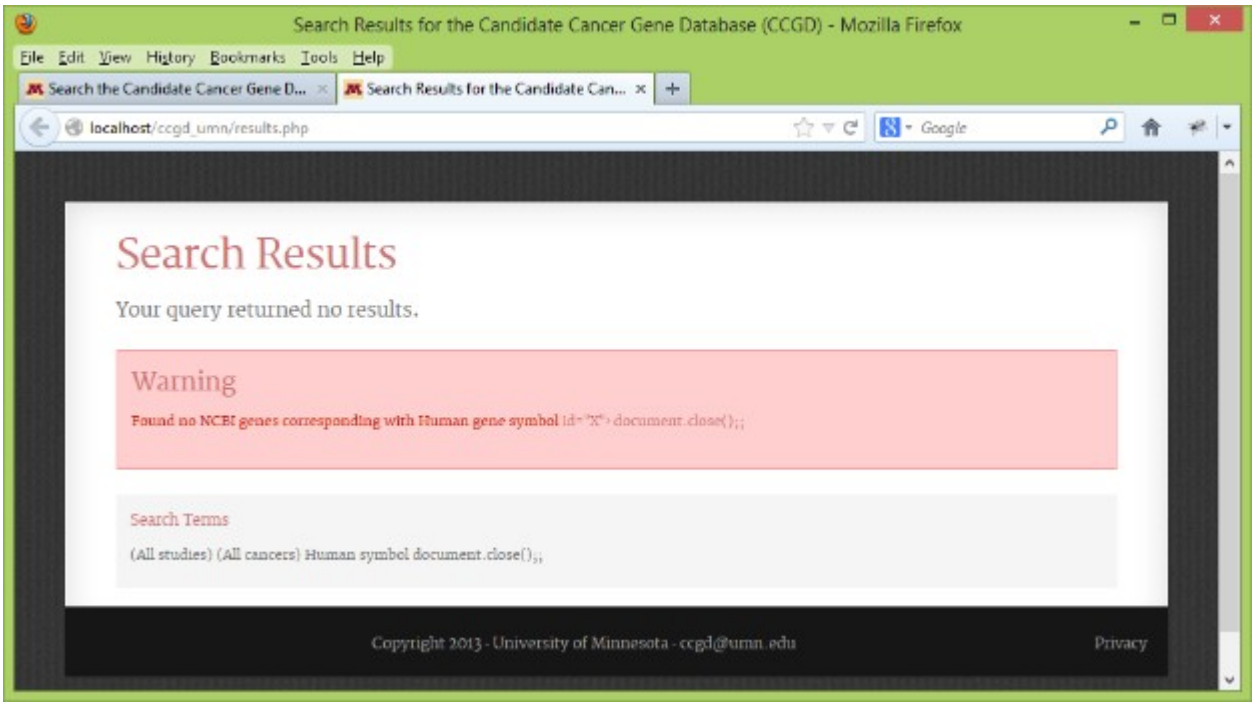
DOM was modified by attack string. Field appears to be very vulnerable to XSS String.
Tested value: <xml id="X"><a><script>document.vulnerable=true;</script>;</a></xml>

The unencoded attack string was found in the html of the document. Other browsers may be vulnerable to this XSS string.
Tested value: <xml id="X"><a><script>document.vulnerable=true;</script>;</a></xml>

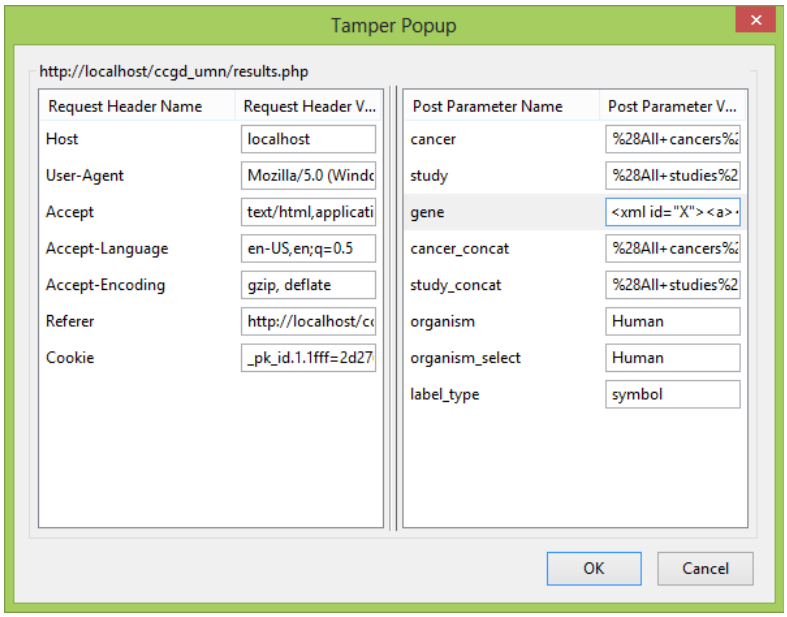
The string actually appears unencoded in `results.php` since it was subject to `htmlentities()` and `strip_tags()` commands.

```
id=&quot;X&quot;&gt;document.vulnerable=true;
```

The error messages are traceable to the echoes of `gene` values inside unmatched term and search term boxes at the end of `results.php`. Despite the error messages, JavaScript does not appear to execute.



The script fails to execute even if entered using the Tamper Data add-on for Firefox.



The risk of a cross-site scripting attack seems small, but there is likely some potential for exploitation any time a page is producing text derived from user input. Consultation with a web security expert might be in order if there is any significant discomfort with existing XSS security measures.

Email

The `email.php` script controls emailing for webpages and scheduled tasks. The script calls `gmail.pl` in the Windows environment and the `mail()` command in a Linux environment. The `gmail.pl` script uses `smtp.gmail.com` as an SMTP relay.

Database

Tables

Most database tables carry an `autoload_` or `work_` prefix. Tables serving information to webpages carry a `ccgd_` prefix. A routine named `import_process` uses `autoload_` tables to create match tables that, along with query tables, provide invalid search term detection. The routine creates mouse-based HomoloGene tables, adding processed information from CGC, COSMIC, and study tables. The routine creates a gene table that provides search term matching. There is also a log table. The model below summarizes database objects.

model.pdf

Routines

The `import_process` routine runs automatically but needs to be manually triggered if there are any direct changes to `autoload_` tables.

The `query_explode` routine splits space-delimited query strings into separate records.

Optimization

A webgrind report for a small query shows that `mysql_query` is the most costly call in `results.php`.

webgrind of C:\wamp\ww

localhost/webgrind/

webgrind v1.0
profiling in the browser

Show 100% of Auto (newest) in milliseconds update

Hide PHP functions

C:\wamp\www\ccgd_umn\results.php

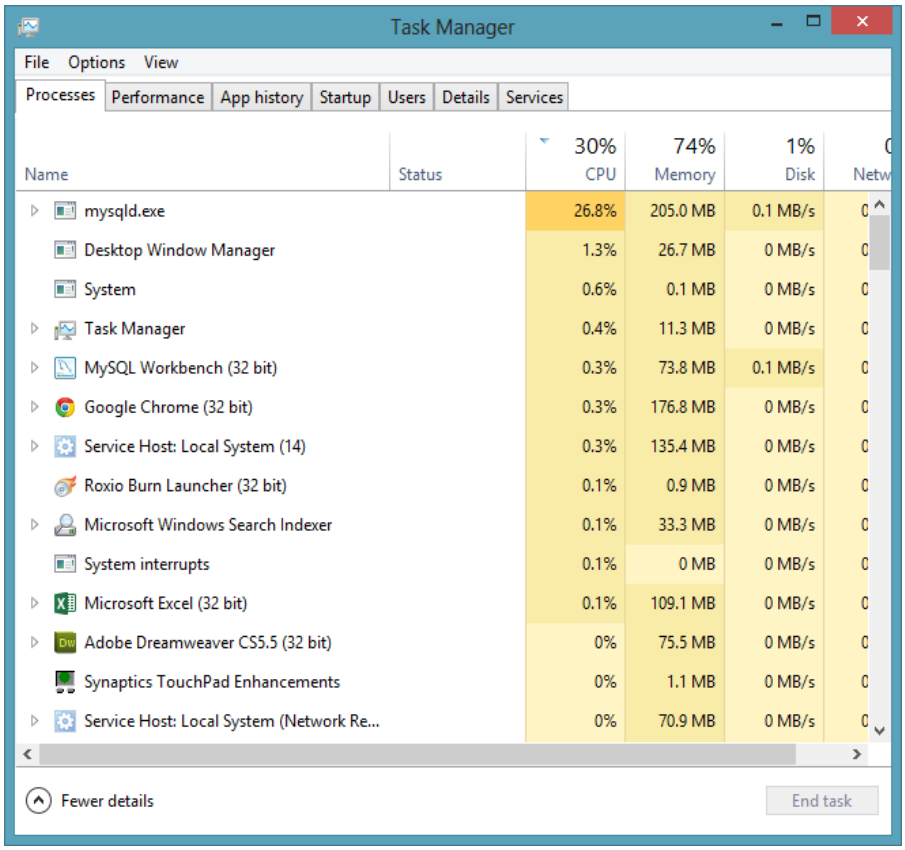
cachegrind out 1366480027.2720 @ 20-04-2013 17:47:08

32 different functions called in 666 milliseconds (1 runs, 32 shown)

Function	Invocation Count	Total Self Cost	Total Inclusive Cost
php::mysql_query	7	643	643
{main}	1	14	666
truncate	104	3	3
GetSQLValueString	19	1	1
php::mysql_fetch_assoc	29	1	1
php::session_start	1	1	1
php::mysql_select_db	2	1	1
include::C:\wamp\www\ccgd_umn\includes\footer.php	1	1	2
php::session_destroy	1	1	1
include::C:\wamp\www\ccgd_umn\includes\university_switch.php	1	0	0

Called From	Count	Total Call Cost
{main} @ 67	1	295
{main} @ 100	1	282
{main} @ 116	1	57
{main} @ 119	1	7
{main} @ 142	1	2
include::C:\wamp\www\ccgd_umn\includes\footer.php @ 36	1	0
{main} @ 95	1	0

No queries contain wildcards for table names. Large webpage queries contain no wildcards for field names. Fields subject to `JOIN` operations or `WHERE` clauses contain indexes. No webpage record sets feature `JOIN` operations. The unmatched terms record set in `results.php` draws from split tables. The query cache size is increased to reduce disk reads for identical queries. Most tables use the `InnoDB` storage engine, but two tables use the `MyISAM` engine in order to allow `FULLTEXT` indexing with keys exceeding 767 characters; one other table also uses the `MEMORY` engine. A test run of an enormous 500,000 gene (3.7 million character) query shows no disk I/O bottleneck.

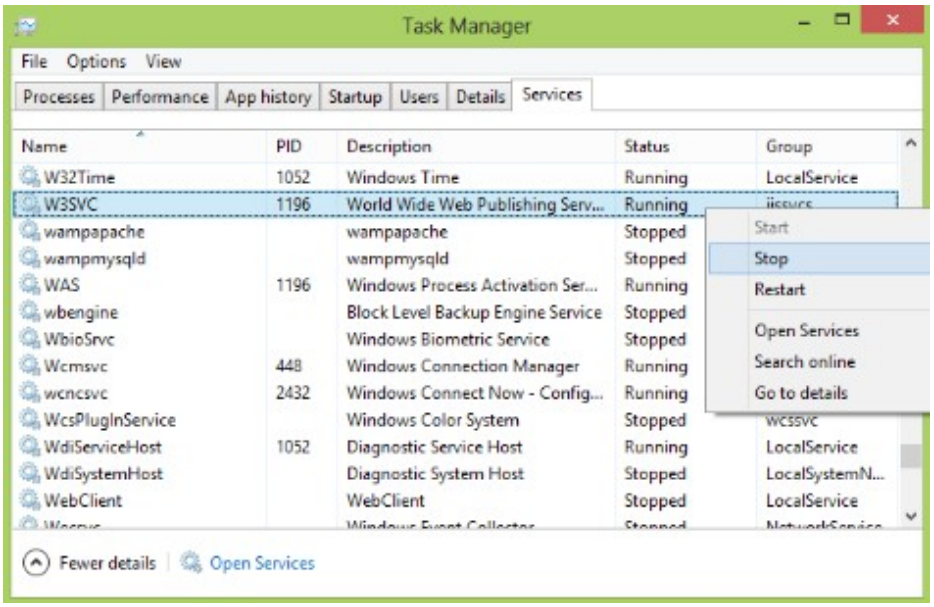


The 500,000 gene query caused a webpage timeout, but a 100,000 gene (740,000 character) query produced a complete webpage in approximately 9 seconds. This included 2.3 seconds for the `query_explode` call. If users gave up after 60 seconds, we might speculate that the website could handle 60/9 ≈ 6 users simultaneously searching for 100,000 genes. Queries of 10,000 genes are not slow enough to register in the `slow_query` log.

Users entering hundreds of thousands of genes will only have to wait a fraction of a second for their browser to run JavaScript that processes delimiters in pasted text; however, the pasting operation that precedes this may require several seconds. This delay should reduce the risk of a single user submitting a series of very large queries that bring service for other users to a halt.

Test Environments

I built Windows and Linux test environments. I generated most screenshots for this document in the Windows environment. For the WAMP stack to run in Windows, I had to disable `W3SVC` and `ReportServer` services in Task Manager.



The Linux environment ran on a virtual machine in Hyper-V, the hypervisor that ships with Windows 8 Professional. I gave the virtual machine a virtual switch for internet connectivity. I worked mostly in the terminal but could also have worked in the GUI if launching programs with `sudo` or root privileges. I loaded Ubuntu 12.04 with `apt-get install` commands for `lamp-server^`, `mysql-workbench`, `phpmyadmin`, `postfix`, `mailutils`, `samba`, `subversion`, and `libapache2-svn` packages. I set up a Samba share of the website root and directly transferred files from the Windows environment. I installed `libnet-ssleay-perl`, `libcrypt-ssleay-perl`, `libnet-https-any-perl`, and `libssl-dev` packages for `excel2txt`. I used the CPAN shell to install `Spreadsheet::ParseExcel`, `Readonly`, `YAML`, `Email::Sender::Transport::SMTP::TLS`, and `excel2txt` modules. I made changes to `/etc/mysql/my.cnf`, `/etc/php5/apache2/php.ini`, and `/etc/postfix/main.cf`, reconciling these with the following mixed list of Windows and Linux settings:

MySQL

a) `ft_min_word_len` from `3` to `1`
b) `ft_stopword_file` from a filename to `""`
c) `group_concat_max_len` from `1024` to `4096`
d) `query_cache_size` from `0` to `10485760`
e) `key_buffer_size` from `0` to `134217728`
f) `myisam_sort_buffer_size` from `0` to `134217728`
g) `long_query_time` from `10` to `1`
h) `slow_query_log` from `OFF` to `ON`
i) `slow_query_log_file` from nothing to a filename
j) addition of `expire_logs_days = 10` to the replication section
k) `max_allowed_packet` from `1M` to `10M`

PHP

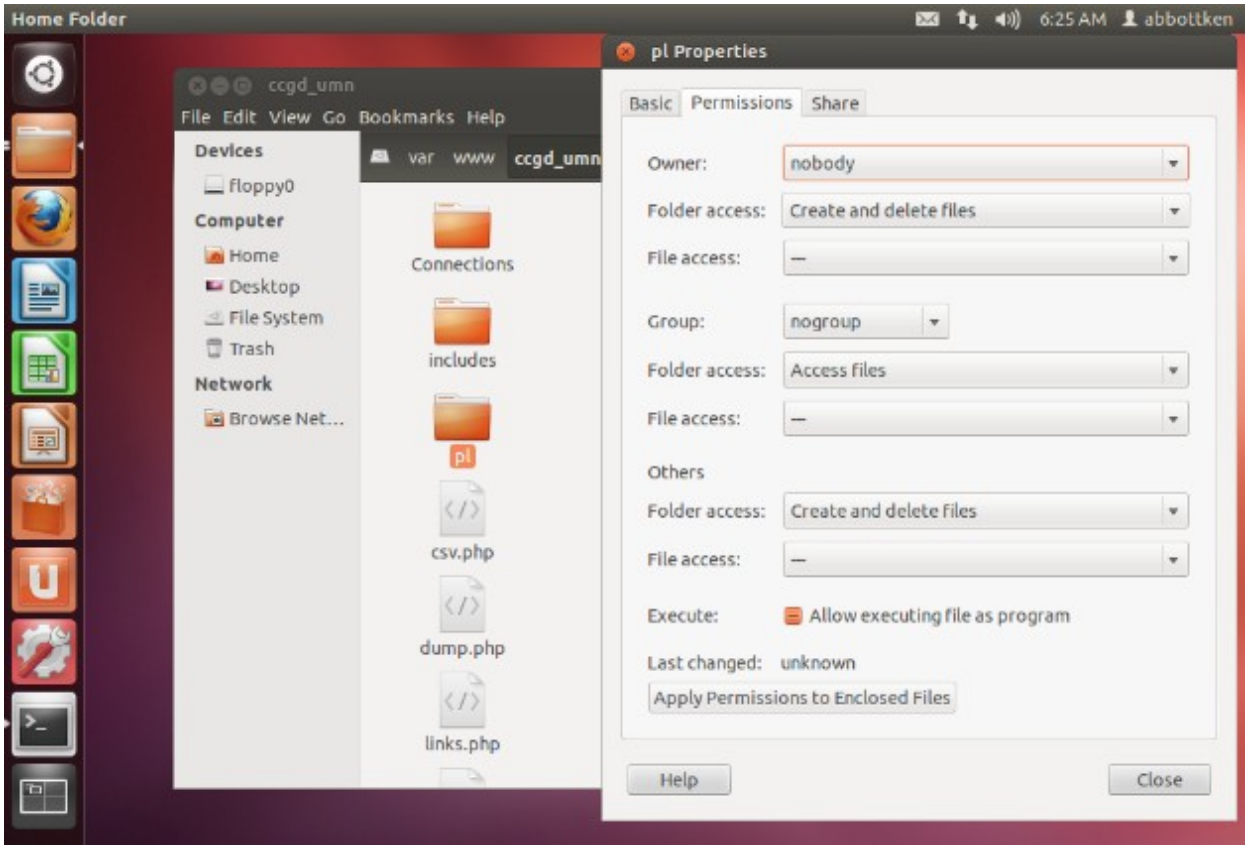
a) `memory_limit` from `128M` to `1024M`
b) `upload_max_filesize` from `2M` to `50M`
c) `xdebug.profiler_enable` from `off` to on
d) `SMTP = localhost`
e) `smtp_port = 25`
f) `sendmail_from = name@domain.com`

g) `sendmail_path = "C:\wamp\sendmail\sendmail.exe -t"`

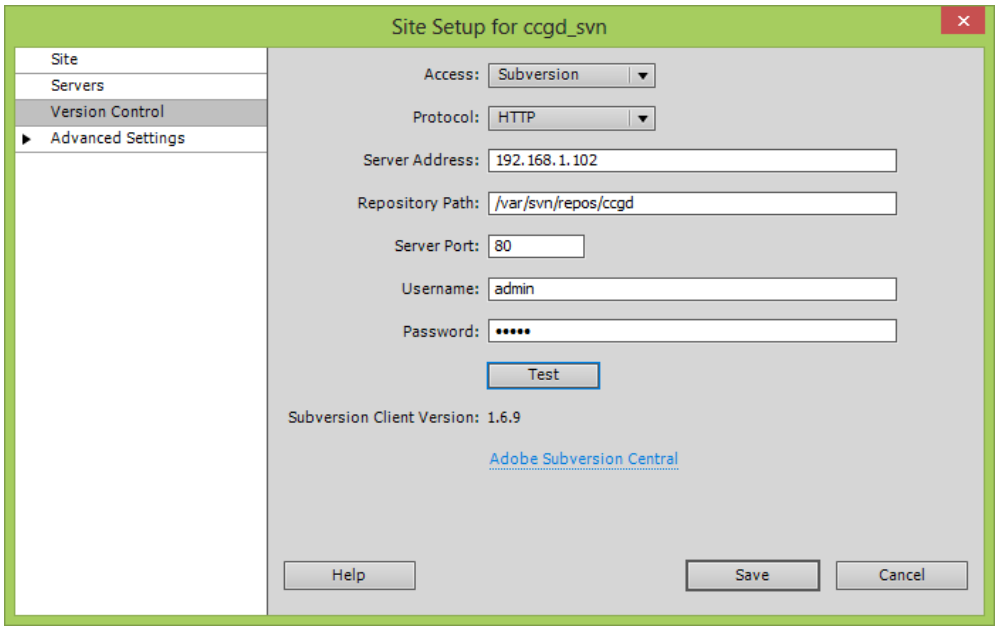
SMTP relay

a) `relayhost = [smtp.gmail.com]:587`
b) `smtp_use_tls = yes`
c) `smtp_sasl_auth_enable = yes`
d) `smtp_sasl_password_maps = hash:/etc/postfix/sasl/sasl_password` (requires addition of `[smtp.gmail.com]:587 mailer.ccgd@gmail.com:password` to `/etc/postfix/sasl/sasl_password`)
e) `smtp_tls_CAfile = /etc/ssl/certa/ca-certificates/crt`
f) `smtp_sasl_security_options =`

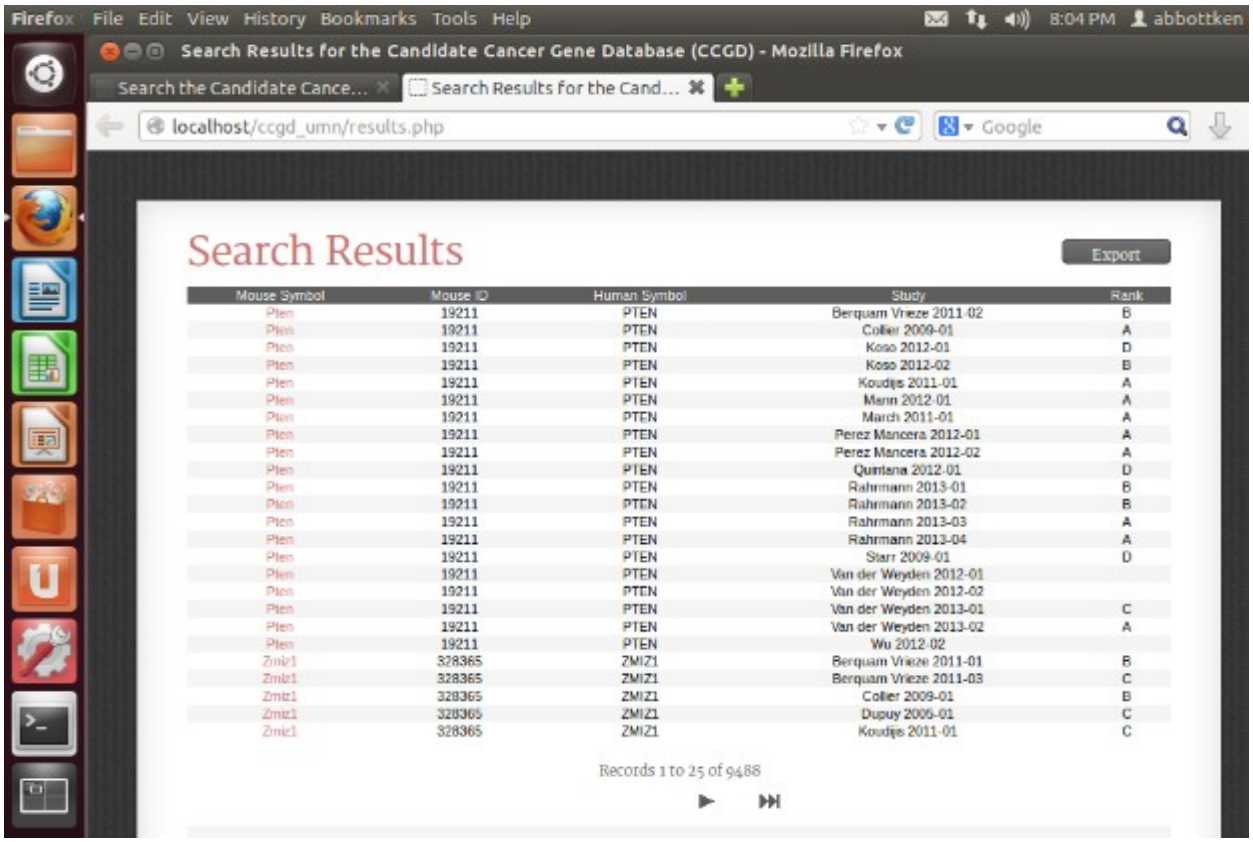
I restarted web, database, and email servers using `service apache2 restart`, `service mysql restart`, and `service postfix restart`. I used `chmod o=rwx pl` to grant privileges to the `/pl` directory.



I scheduled crontab tasks and used `less /var/log/syslog` to check for evidence of task activity. I used `mail` to review messages. I set up Subversion, using `htpasswd -c /etc/subversion/password` to create a Subversion user and configuring Dreamweaver to connect to the repository.



I launched the application and verified that it worked in the new environment.



Downloads succeeded on 6 of 7 nights.

Production Environment

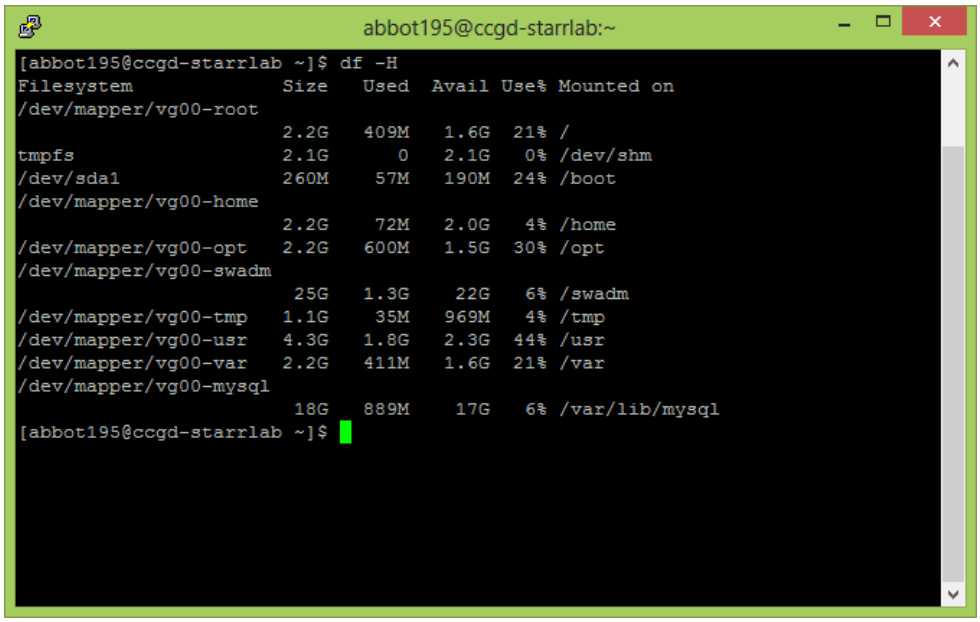
Provision

Platform

Linux Server Hosting by OIT is described on this How-to page. OIT (oialinux@umn.edu) provided a virtual machine running Red Hat Enterprise Linux (see quick-start guide and wiki). A `uname –a` command returned `Linux ccgd-starrlab.oit.umn.edu 2.6.32-358.11.1.el6.x86_64 #1 SMP Wed May 15 10:48:38 EDT 2013 x86_64 x86_64 x86_64 GNU/Linux`. A `perl -v` command returned `v5.10.1`.

Partitions

OIT allocated space as shown below. The MySQL partition required expansion because it was initially only 500MB.



```
abbot195@ccgd-starrlab:~
[abbot195@ccgd-starrlab ~]$ df -H
Filesystem             Size   Used  Avail Use% Mounted on
/dev/mapper/vg00-root
                       2.2G   409M   1.6G  21% /
tmpfs                  2.1G      0   2.1G   0% /dev/shm
/dev/sda1              260M    57M   190M  24% /boot
/dev/mapper/vg00-home
                       2.2G    72M   2.0G   4% /home
/dev/mapper/vg00-opt   2.2G   600M   1.5G  30% /opt
/dev/mapper/vg00-swadm
                        25G   1.3G    22G   6% /swadm
/dev/mapper/vg00-tmp   1.1G    35M   969M   4% /tmp
/dev/mapper/vg00-usr   4.3G   1.8G   2.3G  44% /usr
/dev/mapper/vg00-var   2.2G   411M   1.6G  21% /var
/dev/mapper/vg00-mysql
                        18G   889M    17G   6% /var/lib/mysql
[abbot195@ccgd-starrlab ~]$
```

Packages

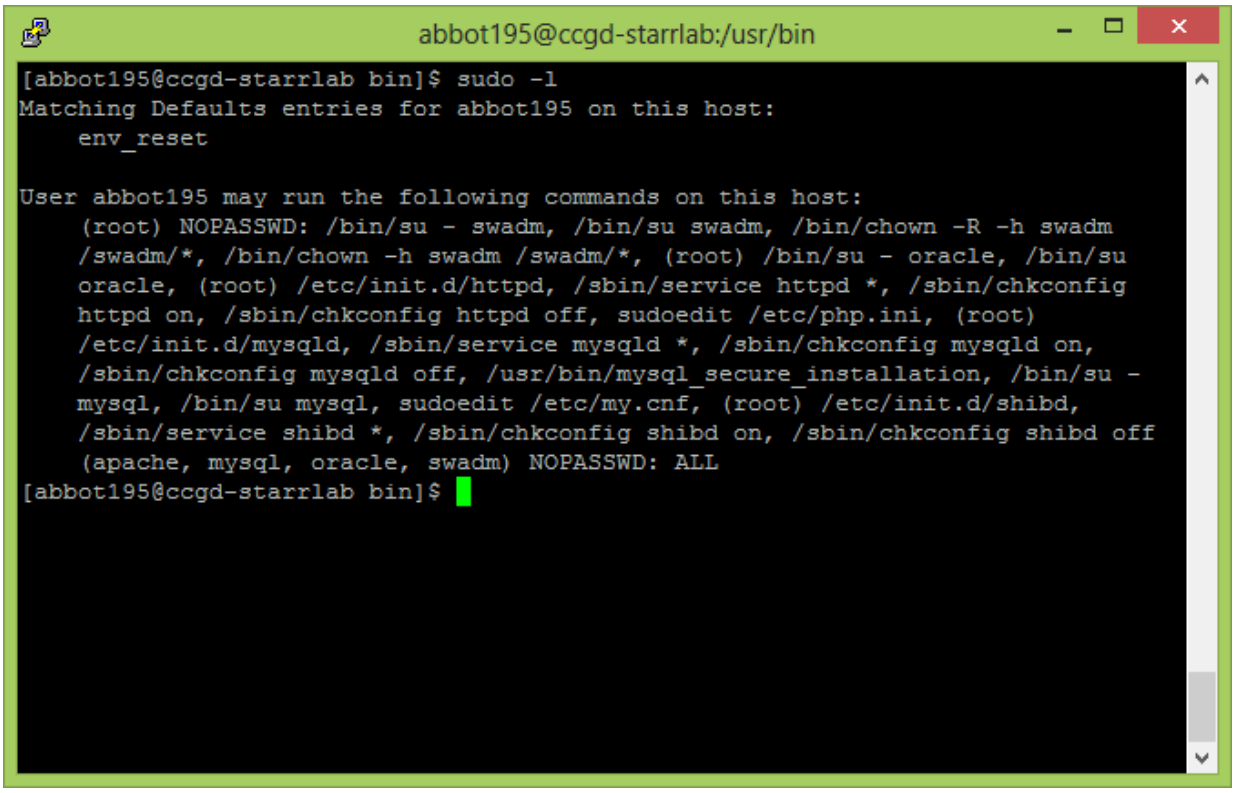
OIT installed Linux packages and their dependencies.

1. subversion-1.6.11-9.el6_4.x86_64
2. gzip-1.3.12-19.el6_4.x86_64
3. postfix-2.6.6-2.2.el6_1.x86_64
4. httpd-2.2.15-28.el6_4.x86_64
5. php-5.3.3-22.el6.x86_64
6. mysql-server-5.1.69-1.el6_4.x86_64
7. perl-Net-SSLeay.x86_64 0:1.35-9.el6
8. perl-Crypt-SSLeay.x86_64 0:0.57-16.el6
9. perl-AnyEvent-HTTP.noarch 0:1.46-1.el6
10. perl-AnyEvent.noarch 0:5.27-1.el6 perl-Curses.x86_64 0:1.28-1.el6
11. perl-Event.x86_64 0:1.13-1.el6 perl-Event-Lib.x86_64 0:1.03-8.el6
12. perl-Glib.x86_64 0:1.223-1.el6 perl-Heap.noarch 0:0.80-4.el6
13. perl-IO-Async.noarch 0:0.29-3.el6 perl-IO-Tty.x86_64 0:1.08-4.el6
14. perl-POE.noarch 0:1.354-1.el6 perl-Socket-GetAddrInfo.x86_64 0:0.12-1.el6
15. perl-Socket6.x86_64 0:0.23-3.el6 perl-TermReadKey.x86_64 0:2.30-13.el6
16. perl-Time-HiRes.x86_64 4:1.9721-131.el6_4

17. perl-WWW-Mechanize.noarch 0:1:62-2.el6
18. Readonly (possibly perl-Readonly-XS-1.05-3.el6.x86_64)
19. uw-imap-utils.x86_64 0:2007e-14.el6
20. libc-client.x86_64 0:2007e-11.el6
21. expect.x86_64 0:5.44.1.15-5.el6_4
22. expect-devel.x86_64 0:5.44.1.15-5.el6_4
23. perl-Expect.noarch 0:1.21-3.el6

Privileges

OIT established the privileges for my user id which were visible with the command `sudo -l`. I used the command `sudo su swadm` to switch to the `swadm` user.



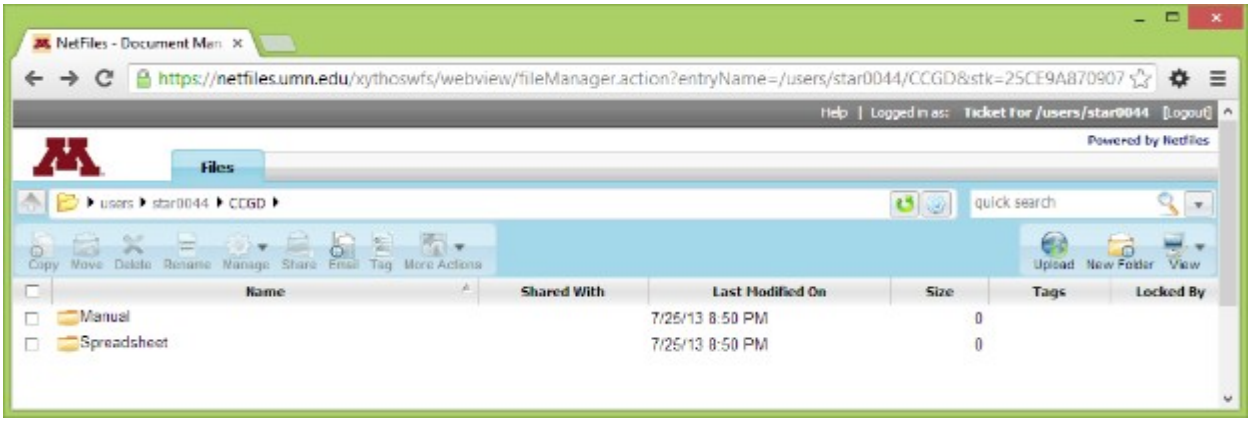
OIT also granted read access for other users to `/var/log/mysqld.log`; this file was alternately readable with `su` to the `mysql` user.

Email

Upon request, OIT provided an email account named ccgd@umn.edu with conversion to a Google Groups email site. As described below in the Google Groups section, this departmental account was reverted from a Google Groups departmental account to a sponsored non-group Google account. OIT also granted permission for abbot195@umn.edu sponsored account conversion to a Google account.

Storage

Tim Starr established a NetFiles share of a folder in his own NetFiles space. The share was not accessible while logged into Gophermail or while using any other centralized Shibboleth authentication.



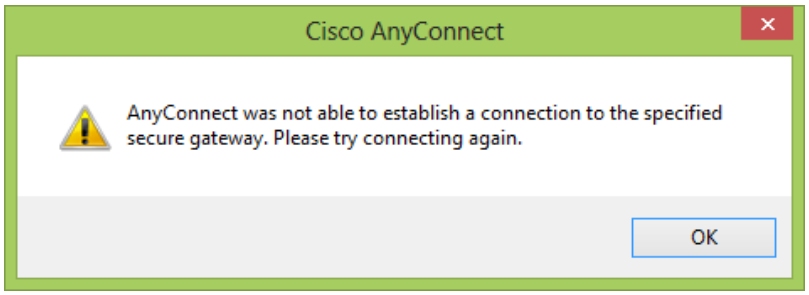
Connection

OIT required server connection through VPN and then SSH. OIT created accounts on ale03.oit.umn.edu and ccgd-starrlab.oit.umn.edu, specifying that I could:

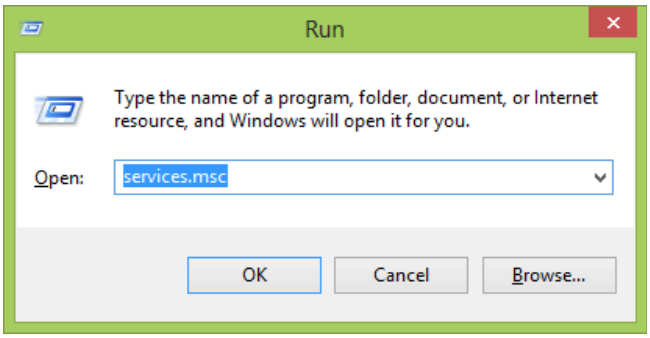
- SSH to ale03.oit.umn.edu and then SSH to ccgd-starrlab.oit.umn.edu
- establish a VPN connection and then SSH to ccgd-starrlab.oit.umn.edu
- establish a VPN connection and use Dreamweaver SFTP functionality to connect to ccgd-starrlab.oit.umn.edu.

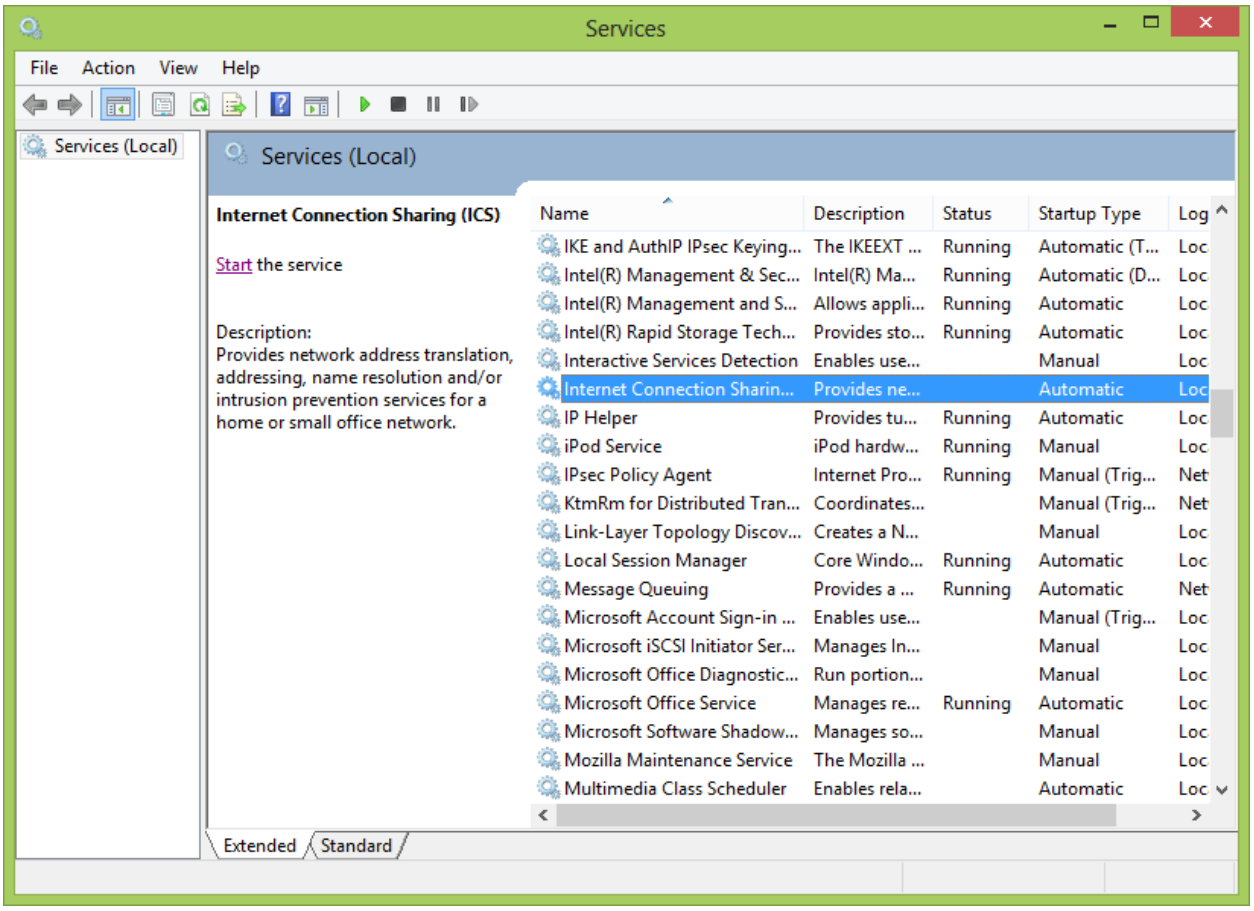
VPN

I initially received an error message when attempting to use the VPN client.

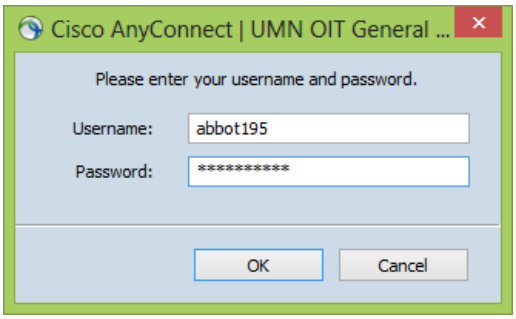


I found out that VPN connection was not possible without disabling internet connection sharing in `services.msc`. This meant that I could not simultaneously use the VPN client and use a network bridge to feed internet connectivity to the virtualized Linux machine in Hyper-V.

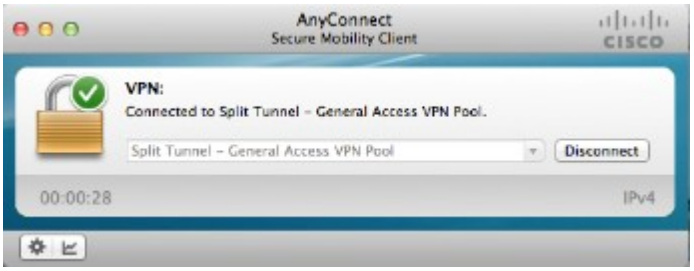




I used the AnyConnect VPN client from the [OIT VPN download page]() to establish the VPN connection. The VPN connection did not require an Mkey or Duo password.

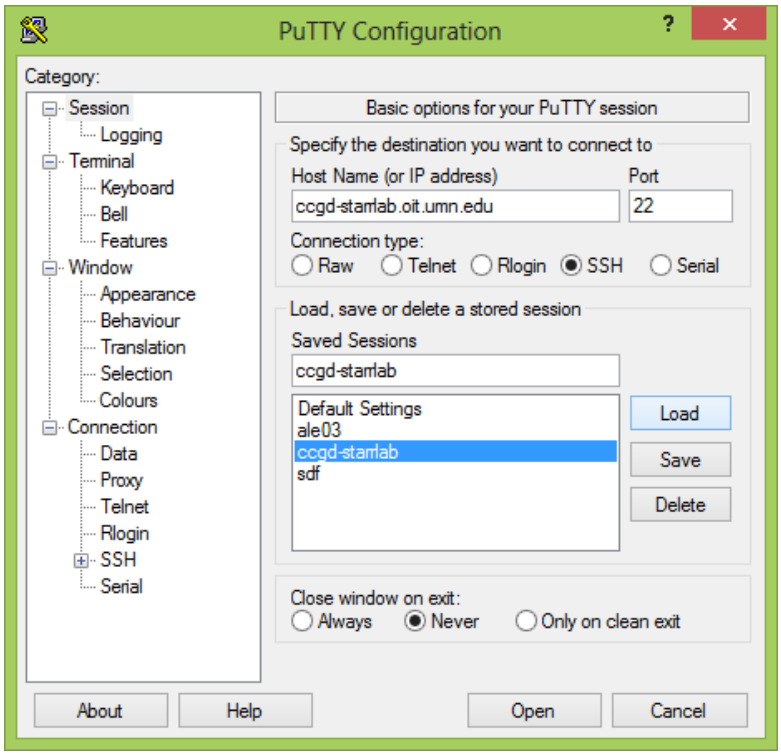


The AnyConnect client also worked on a Mac.



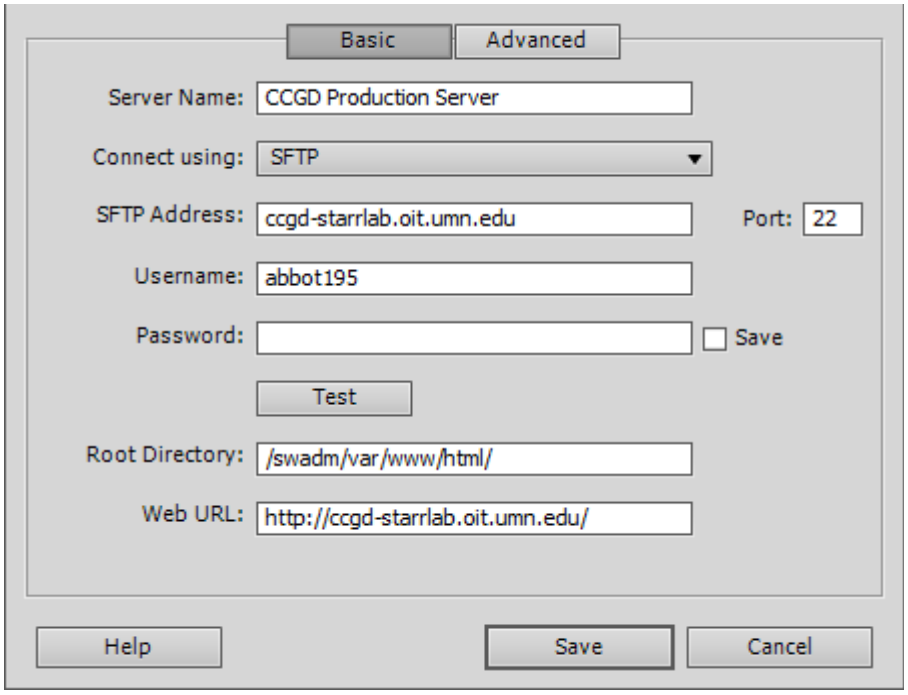
SSH

Having established a VPN connection, I used the PuTTY client to establish an SSH connection to ccgd-starrlab.oit.umn.edu. The SSH connection initially required an Mkey password (later a Duo password) instead of the regular login.

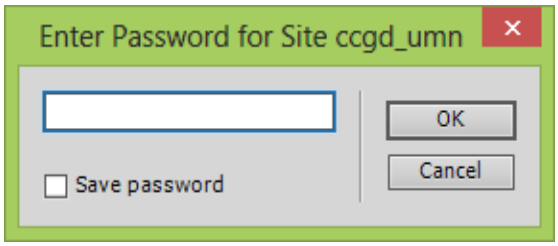


SFTP

I configured the SFTP production server connection in Dreamweaver.



I did not bother saving a password because SFTP and SSH connections required Mkey use, unlike the VPN connection. This meant that I would need to enter the password each time I connected to the remote server.



The SFTP connection from Dreamweaver stopped working when OIT abandoned Mkey authentication in favor of Duo authentication. This is because Dreamweaver's password dialog gave me no way of answering Duo's authentication method question. The authentication question had to be handled in an interactive PuTTY session.

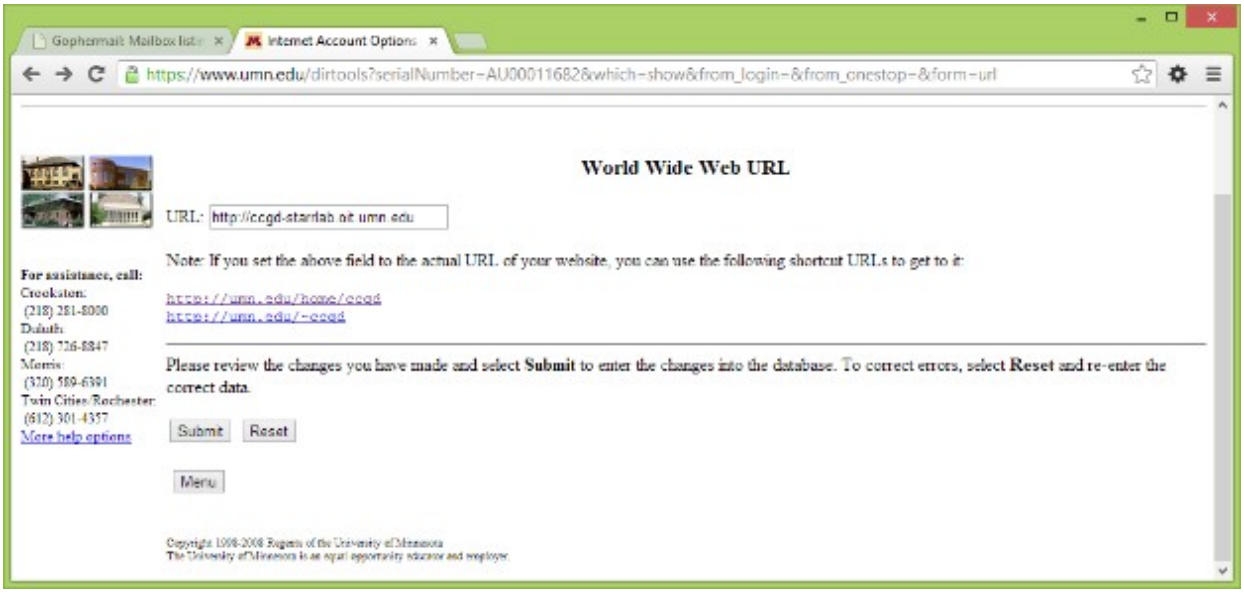
```
Duo two-factor login for abbot195

Enter a passcode or select one of the following options:
1. Duo Push to XXX-XXX-1345
2. Phone call to XXX-XXX-1345
3. SMS passcodes to XXX-XXX-1345

Passcode or option (1-3):
```

Unfortunately, OIT chose not to support an append mode allowing entry of keys or authentication methods in the above Dreamweaver dialog box (e.g., `Password,123456` or `Password,push`). OIT also chose not to support a configuration option which would have allowed for the selection of a default message, bypassing the interactive question entirely. To transfer files to and from my development machine in (`GET` and `PUT` operations), I had to use the WinSCP client. I describe the WinSCP client setup later in this document.

Configuration

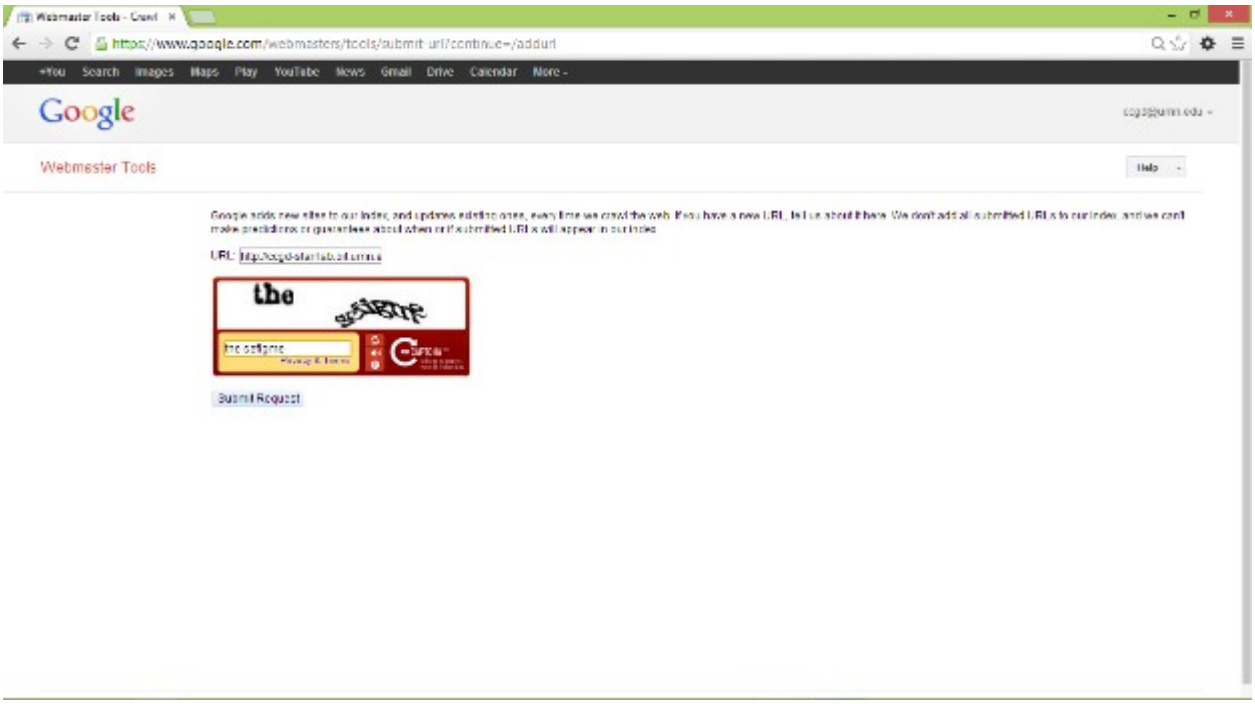
Redirects

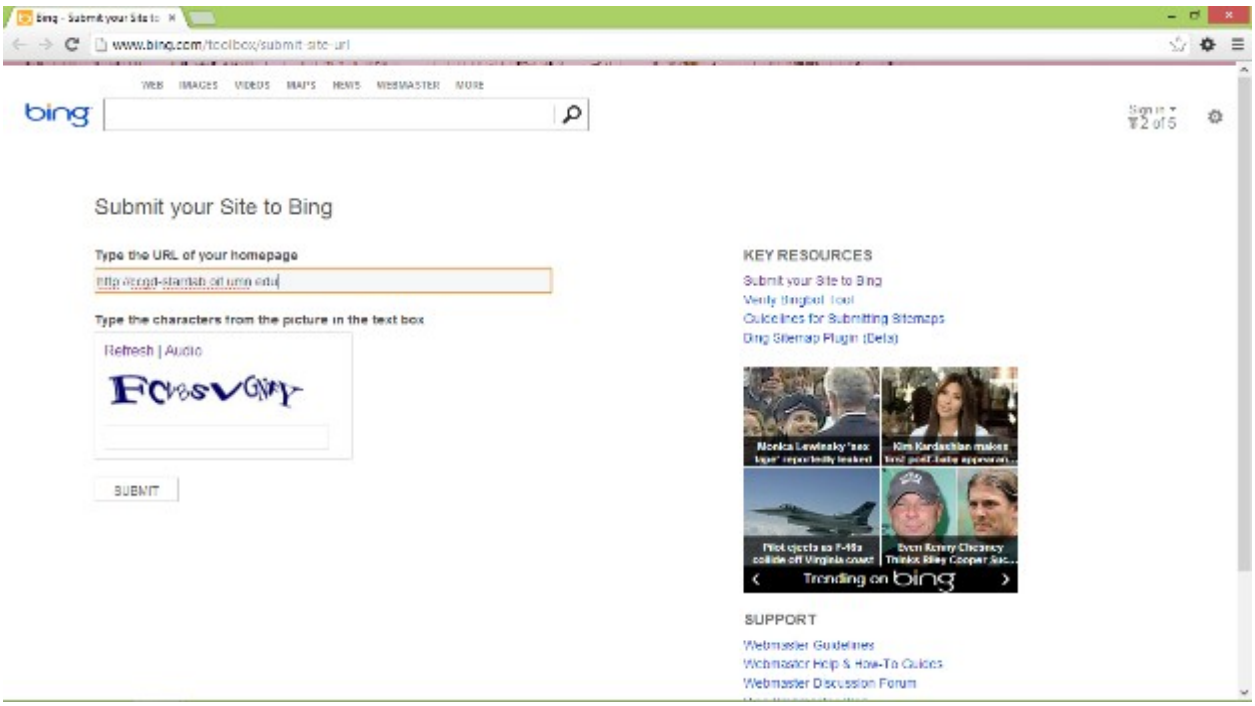
I enabled the redirects in the Gophermail client.



Indexing

I asked Google and Bing (which also handles Yahoo indexing) to crawl and index the site.

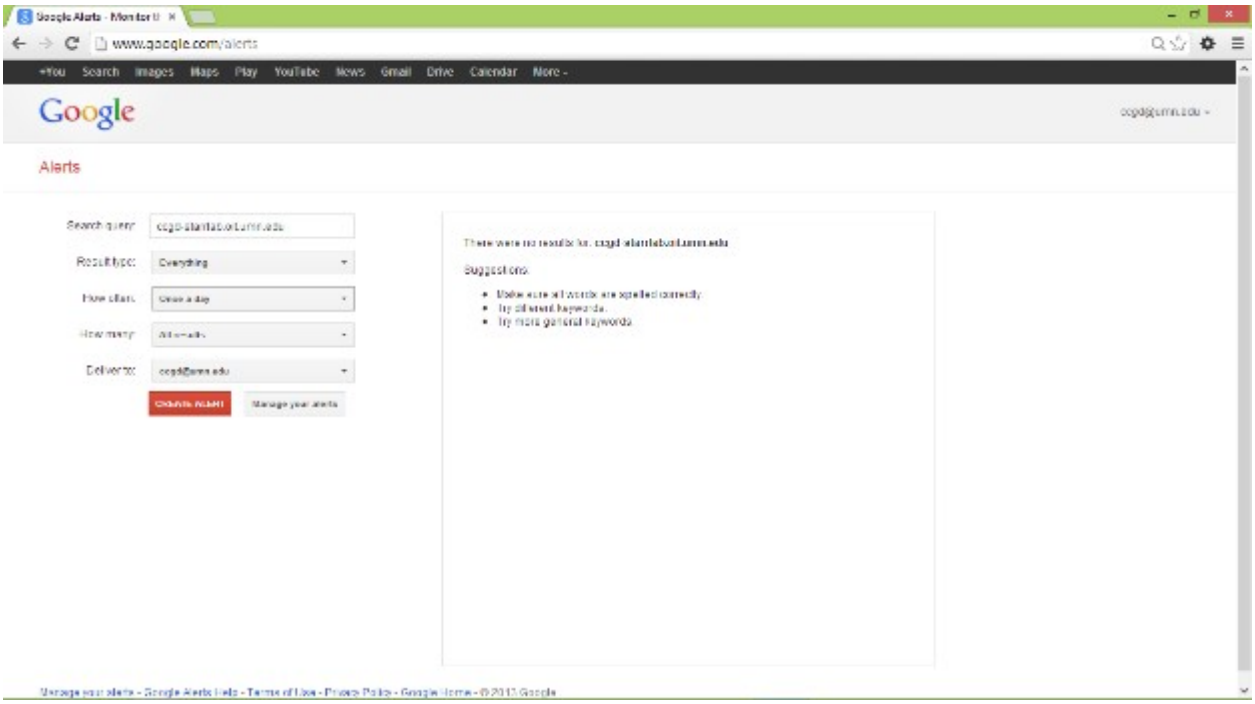




Google

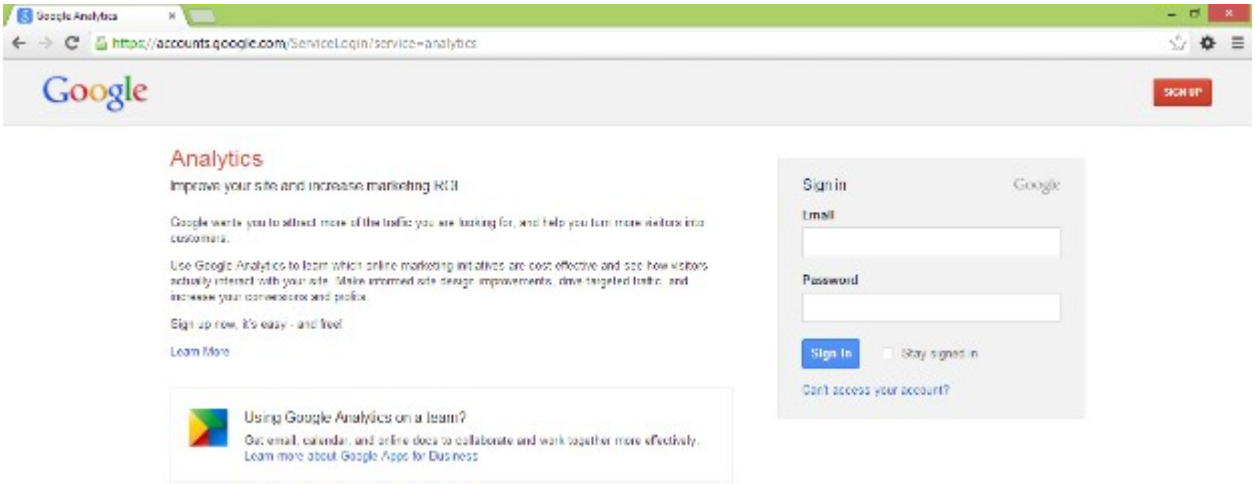
Search Agent

I signed ccgd@umn.edu up for email alerts from Google.

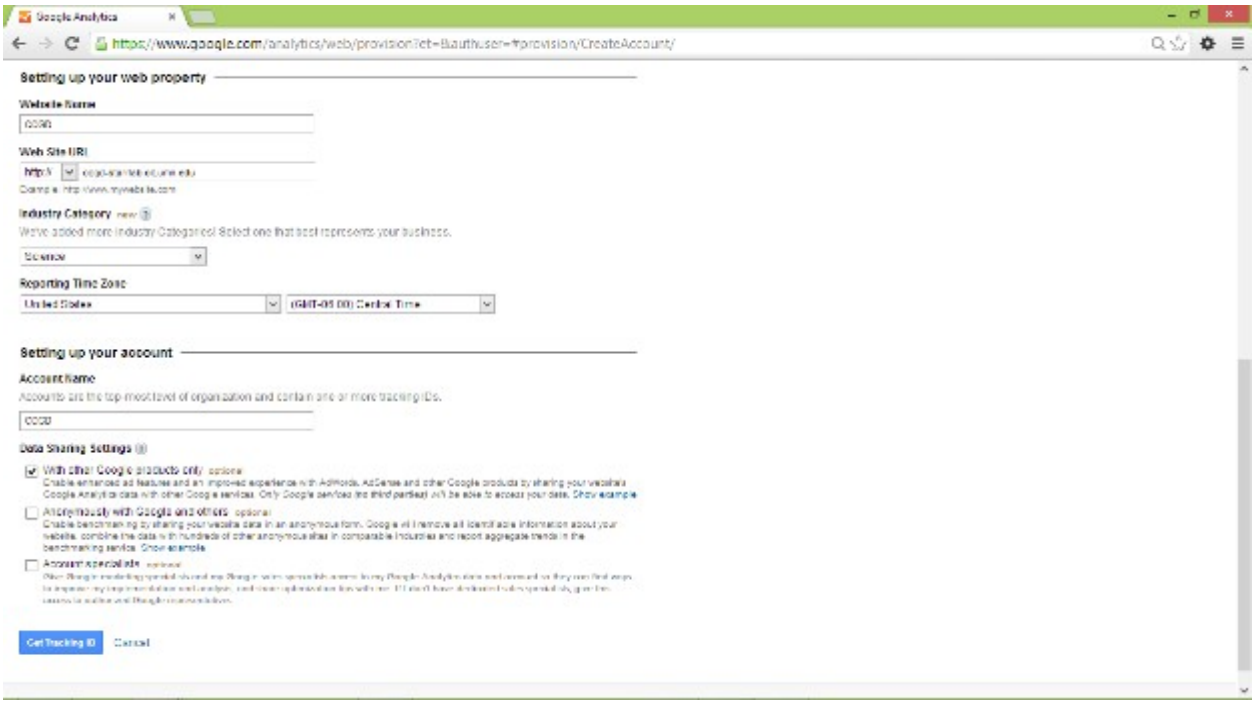


Analytics

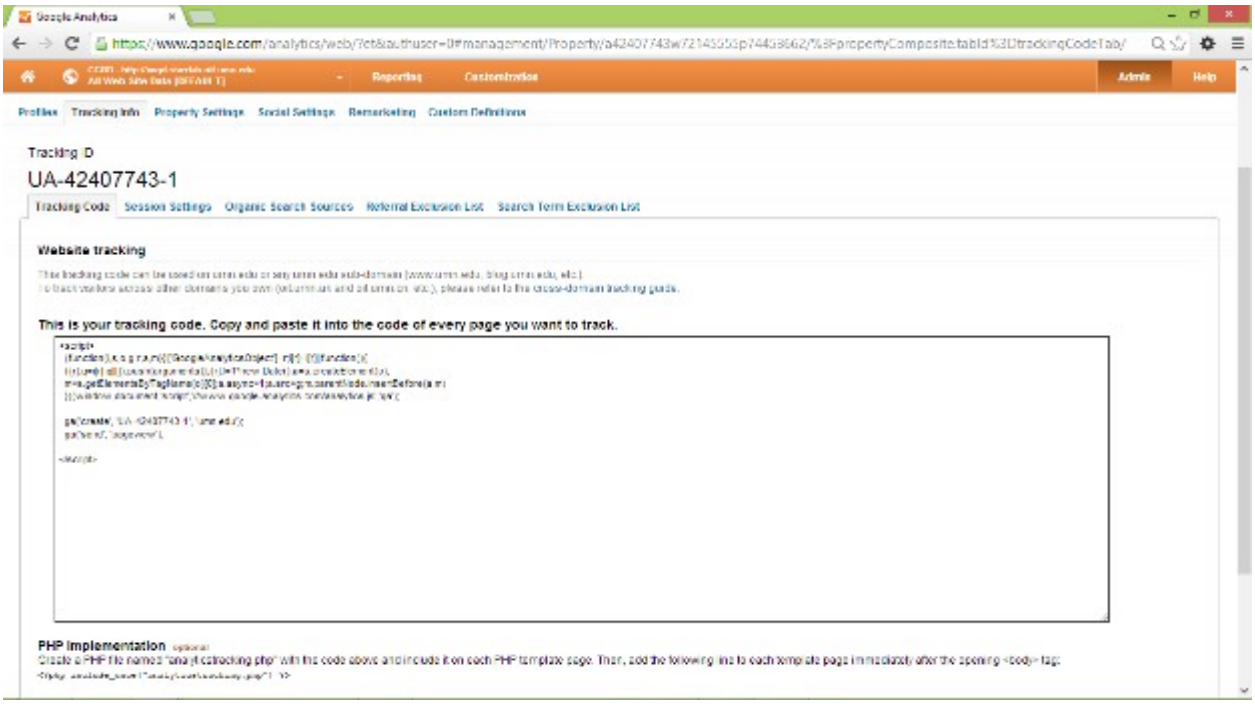
Following the instructions at http://www.oit.umn.edu/web-stats/, I signed up for a new Google Analytics account at https://accounts.google.com/ServiceLogin?service=analytics/.



Google required email verification through ccgd@umn.edu. After activating the account I opened https://www.google.com/analytics/web/provision?et=&authuser=#provision/SignUp/ and configured tracking.



Google generated a tracking ID and code snippet which I placed into `includes/analytics.php`.

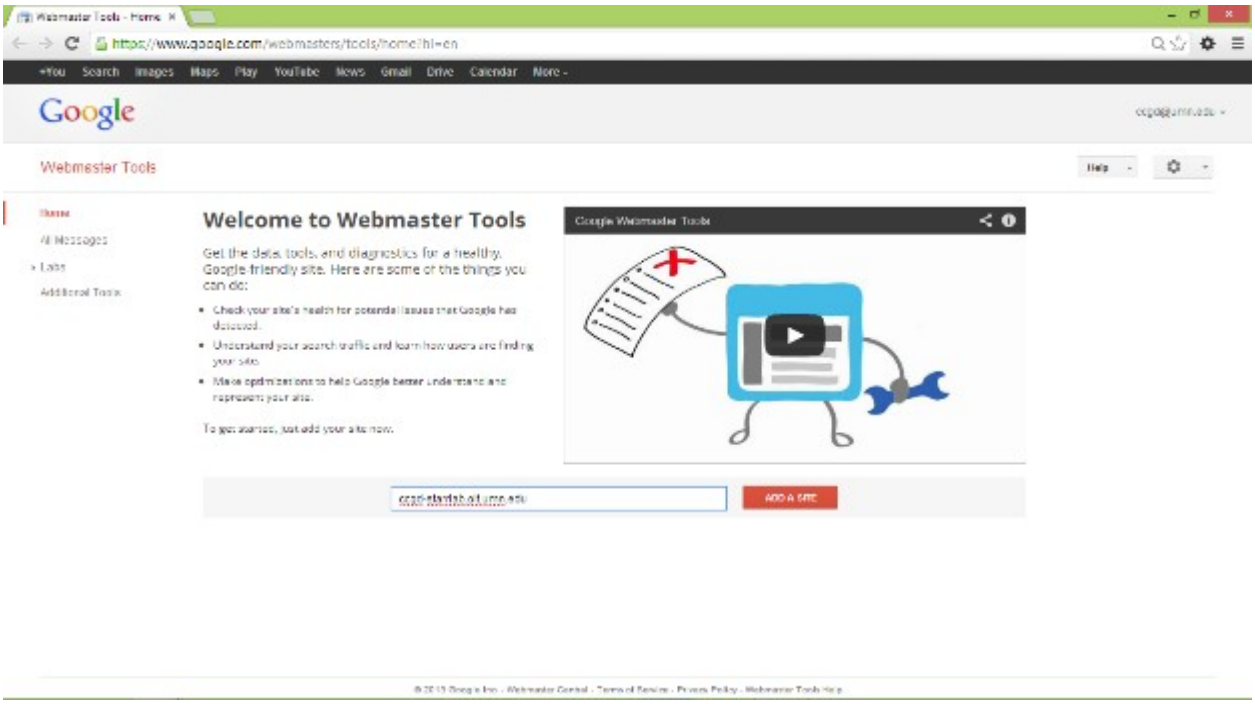


I added a shared report of visits by date, weekday, and hour.

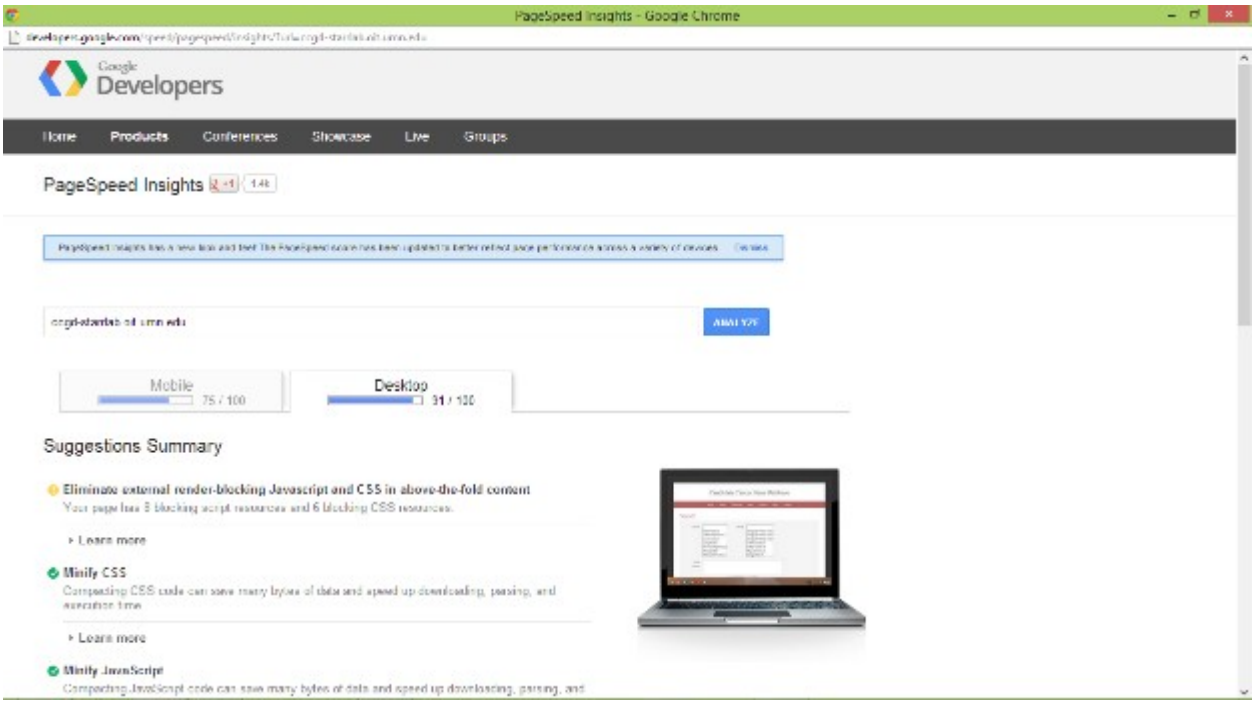
As described in the Google Groups section below, the Google Analytics account was later switched to ccgd.starrlab@gmail.com.

Webmaster Tools

I added the site to Webmaster Tools.



I reviewed Google's recommendations for site optimization.



I chose not to use inline styles and scripts because the style and script code is critical to site function and much too large to be inlined. Caching should also diminish page load times.

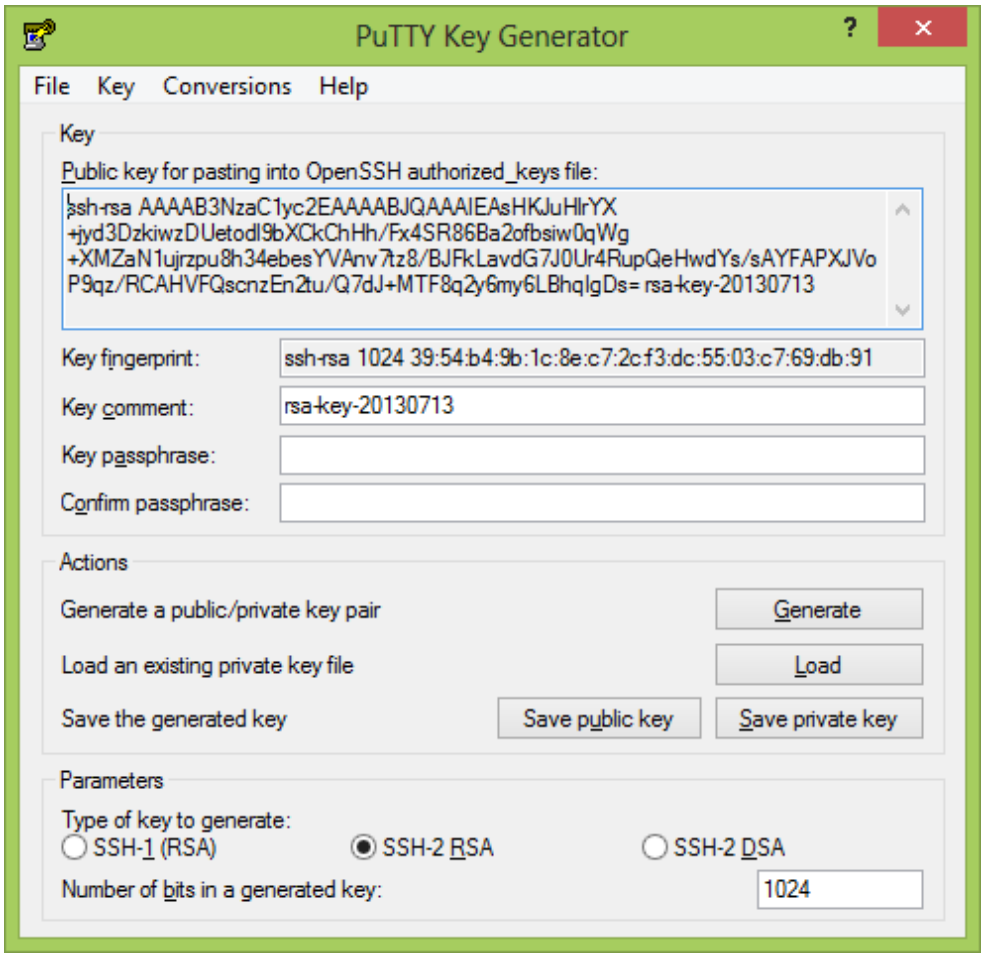
Groups

OIT converted the ccgd@umn.edu email account into a Google Group account. Google then required conversion of the account for Analytics and Webmaster Tools; I chose ccgd.starrlab@gmail.com. The group forwarded all ccgd@umn.edu emails to the group owner but not to group members. The departmental account ID

ccgd@umn.edu was reassigned to a sponsored account; while this resulted in loss of Google Group features, it allowed for NetFiles access.

SSH

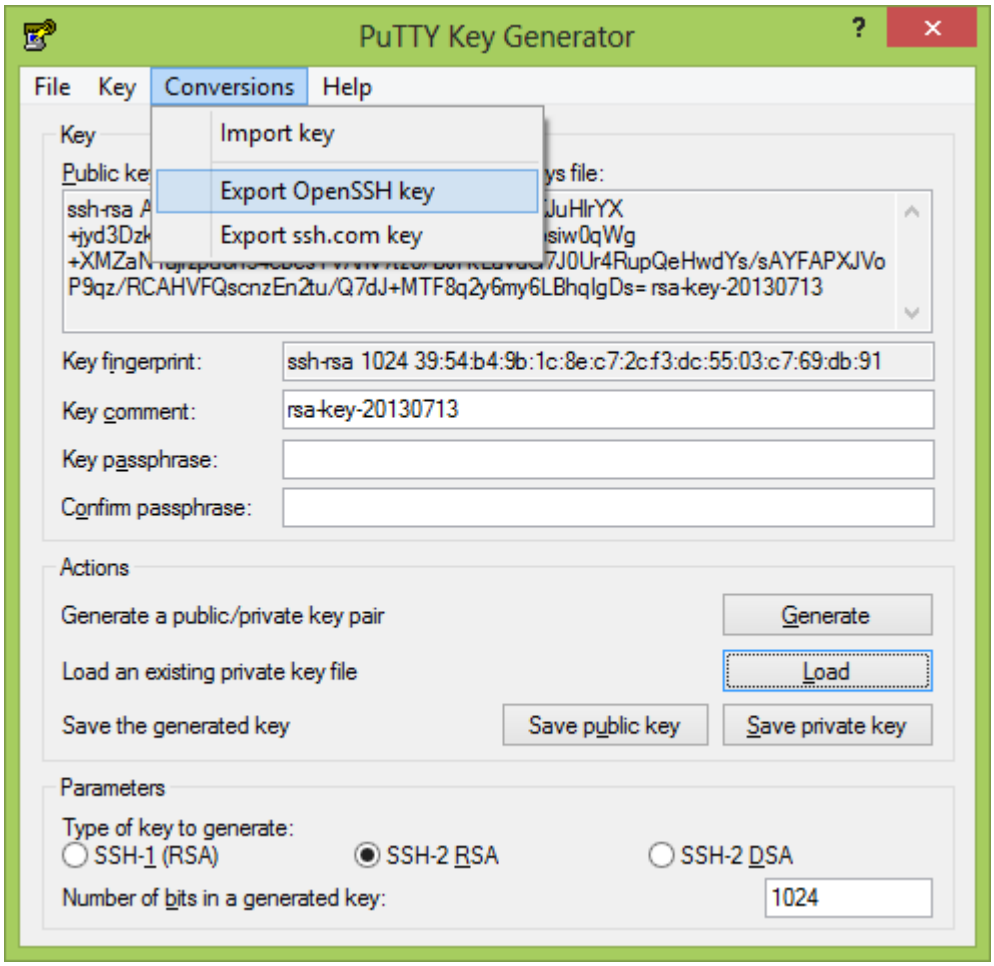
To enable future Subversion use I created an SSH key with PuTTYgen.



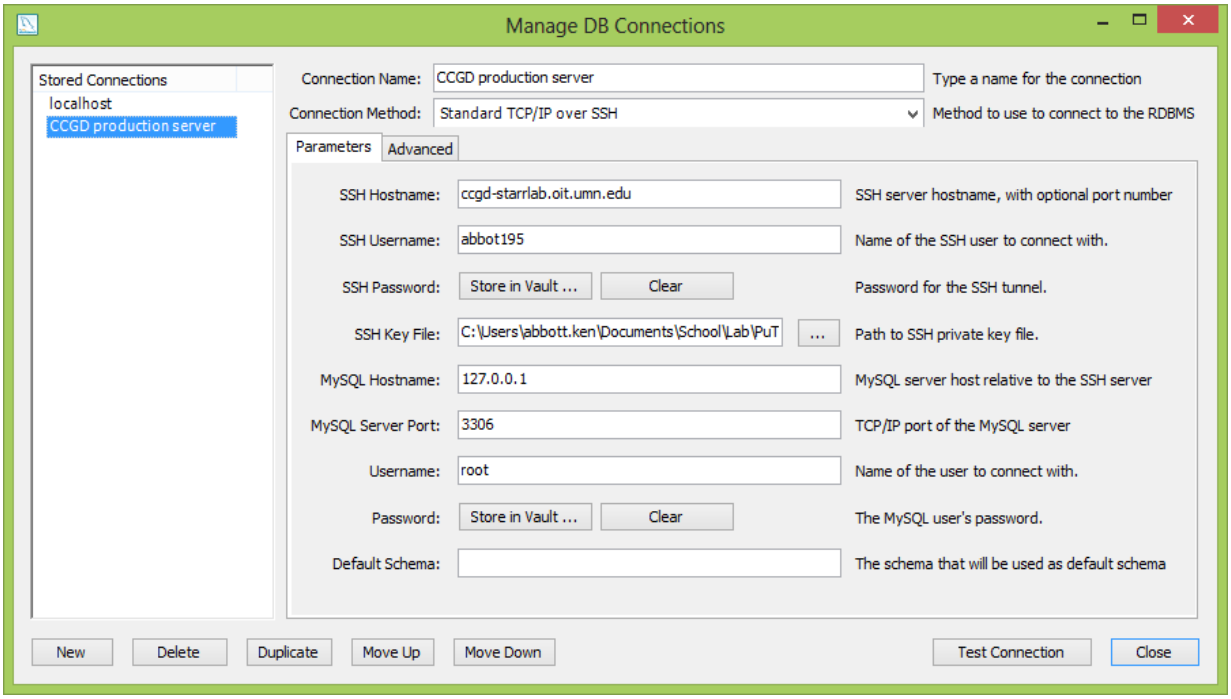
On my computer I saved the private key as `puttygen.ppk`.

In `/home/abbot195`, I used `mkdir .ssh` and `cd .ssh` to create and enter an SSH subdirectory. I used `vim authorized_keys` to create a key file into which I pasted the public key. I then used `chmod 600 authorized_keys` to set permissions for the `authorized_keys` file.

I used the conversion feature in PuTTYgen to create an OpenSSH version of the private key.



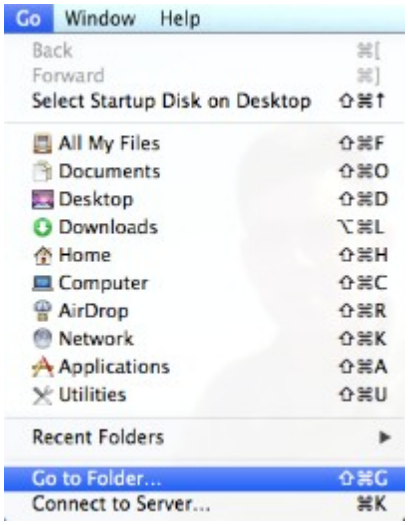
I then pointed MySQL Workbench to the key.

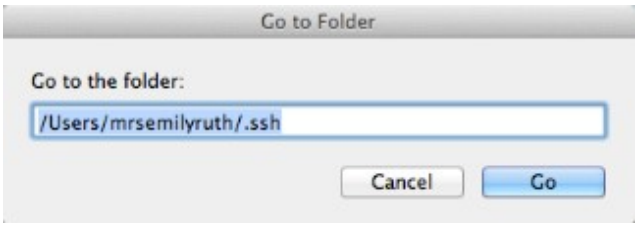


I repeated the SSH key generation process on a Mac, opening the Terminal application and entering `ssh-keygen –t rsa –b 1024`.

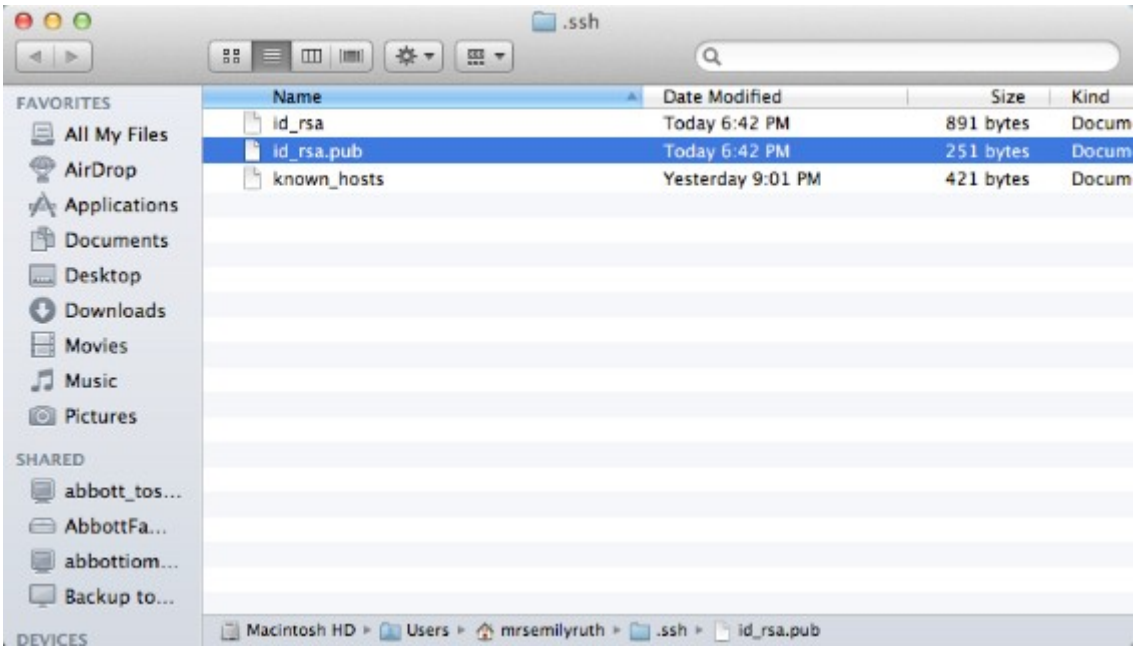
```
mrsemilyruth — bash — 80×24
Emilys-MacBook-Pro:~ mrsemilyruth$ ssh-keygen -t rsa -b 1024
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/mrsemilyruth/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/mrsemilyruth/.ssh/id_rsa.
Your public key has been saved in /Users/mrsemilyruth/.ssh/id_rsa.pub.
The key fingerprint is:
f6:66:0a:17:f3:68:05:98:1d:c8:1a:58:79:bd:2e:4c mrsemilyruth@Emilys-MacBook-Pro.
local
The key's randomart image is:
+--[ RSA 1024]----+
|   o.o o.        |
|  . o ++..       |
|     +o o.       |
|    . E ..       |
|     o .S .      |
|      o..B       |
|      ..+ =      |
|       + +       |
|        .        |
+-----------------+
Emilys-MacBook-Pro:~ mrsemilyruth$
```

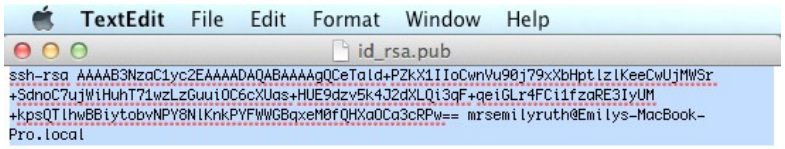
I then navigated directly to the folder shown in the Terminal application.



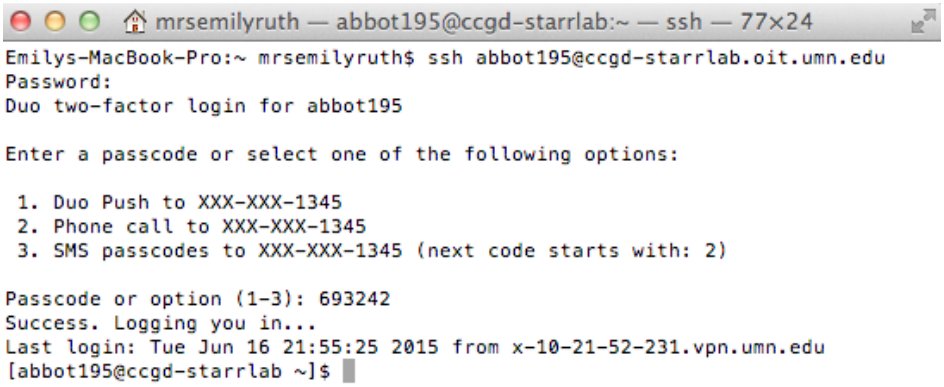


The `.ssh` folder contained a private key file and a public key file with a `.pub` extension. I opened the public key file in TextEdit, copying its contents to the clipboard.

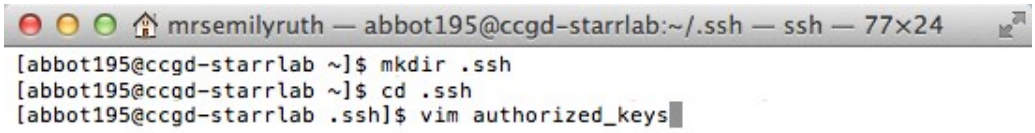




I opened a second Terminal window and connected to the CCGD, using a `ssh username@ccgd-starrlab.oit.umn.edu` format, entering my regular password, and then entering a Duo password.



I made a `.ssh` directory, entered the `.ssh` directory, and started a file named `authorized_keys`.



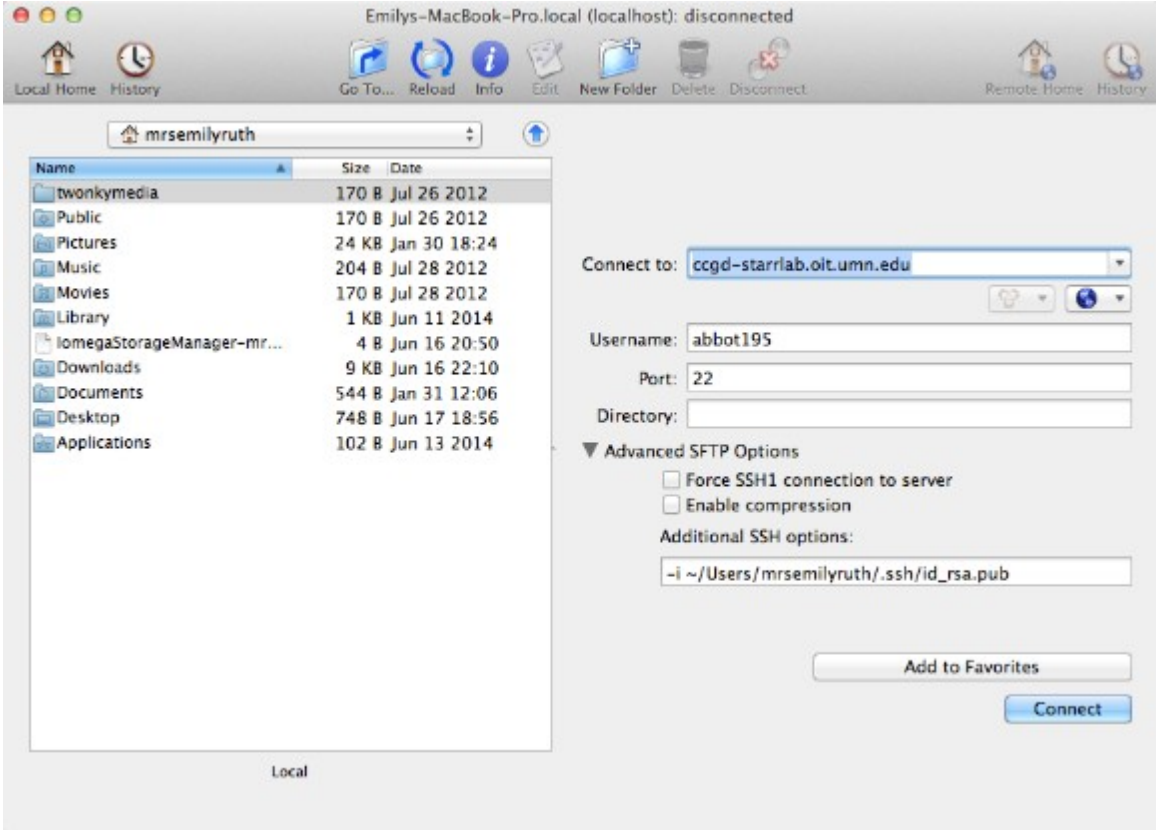
I pressed the `i` key to switch to insert mode and pasted the public key text from my clipboard into the editor.

```
mrsemilyruth — abbot195@ccgd-starrlab:~/.ssh — ssh — 77×24
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQCeTald+PZkX1IIoCwnVu90j79xXbHptlzlKeeCw
UjMWSr+SdnoC7ujWiHuhT71wzLzGuuiOC6cXUas+HUE9dzv5k4J2dXLQi3qF+qeiGLr4FCi1fzaRE
3IyUM+kpsQTlhwBBiytobvNPY8NlKnkPYFWWGBqxeM0fQHXaOCa3cRPw== mrsemilyruth@Emily
s-MacBook-Pro.local
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                          1,251         All
```

I exited insert mode with `Esc`, typing `wq` to write the file to disk and quit the editor. Then, I used `chmod` to change the `authorized_keys` file permissions to `600`.

```
mrsemilyruth — abbot195@ccgd-starrlab:~/.ssh — ssh — 77×24
[abbot195@ccgd-starrlab ~]$ mkdir .ssh
[abbot195@ccgd-starrlab ~]$ cd .ssh
[abbot195@ccgd-starrlab .ssh]$ vim authorized_keys
[abbot195@ccgd-starrlab .ssh]$ chmod 600 authorized_keys
[abbot195@ccgd-starrlab .ssh]$
```

By specifying an `-i` command line option for SSH, I was able to connect Fugu connection to the server without using Duo authentication.



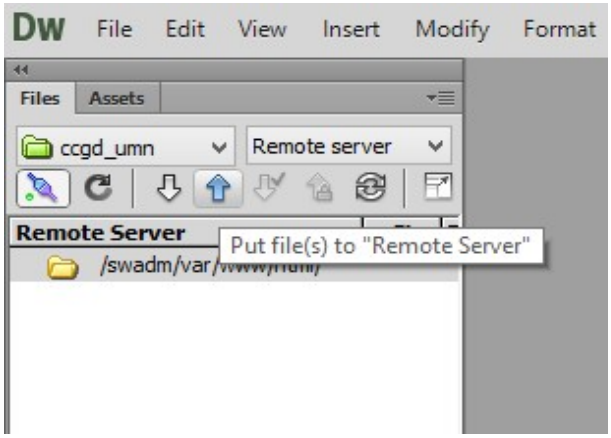
Apache

I followed configuration instructions on the OIT Linux wiki. I used `/usr/local/bin/swadm-httpd_initialize` to build directories in `/swadm`. I changed directories with `cd /swadm/etc/httpd/conf.d` and used `mv ssl.conf ssl.conf.renamed` to stop Apache from using certificates; HTTPS would have broadcasted the existence of some secure functionality and seemed like overkill for only one user who would only occasionally authenticate through `login.php` and only be allowed to upload one non-executable file. I used `vim /swadm/etc/httpd/conf/httpd.conf` to update the `<Directory "/var/www/html">` tag, changing it to `<Directory "/swadm/var/www/html">`; inside this tag I changed the `AllowOverride` option from `None` to `All`. I also added the lines below to the end of the file.

- `# Set expiration for fonts, images, and scripts that are unlikely to change`
- `<IfModule mod_expires.c>`
- ` ExpiresActive On`
- ` ExpiresByType application/vnd.ms-fontobject "access plus 1 year"`
- ` ExpiresByType application/x-font-ttf "access plus 1 year"`
- ` ExpiresByType application/x-font-opentype "access plus 1 year"`
- ` ExpiresByType application/x-font-woff "access plus 1 year"`
- ` ExpiresByType image/svg+xml "access plus 1 year"`
- ` ExpiresByType image/gif "access plus 30 days"`
- ` ExpiresByType image/jpeg "access plus 30 days"`
- ` ExpiresByType image/jpg "access plus 30 days"`
- ` ExpiresByType image/png "access plus 30 days"`
- ` ExpiresByType image/x-icon "access plus 30 days"`

```
 ExpiresByType text/javascript "access plus 30 days"
 ExpiresByType text/css "access plus 30 days"
 <Files style-mods.css>
  ExpiresDefault "access plus 1 hours"
 </Files>
</IfModule>
# Set deflation
<IfModule mod_deflate.c>
 SetOutputFilter DEFLATE
 AddOutputFilterByType DEFLATE text/plain
 AddOutputFilterByType DEFLATE text/html
 AddOutputFilterByType DEFLATE text/css
 AddOutputFilterByType DEFLATE application/xhtml+xml
 AddOutputFilterByType DEFLATE application/javascript
 AddOutputFilterByType DEFLATE application/x-javascript
 BrowserMatch ^Mozilla/4 gzip-only-text/html
 BrowserMatch ^Mozilla/4\.0[678] no-gzip
 BrowserMatch \bMSIE !no-gzip !gzip-only-text/html
</IfModule>
```

I used `sudo /sbin/chkconfig httpd on` to have Apache start upon boot. I then used Dreamweaver to send all files into the website directory.



This included copies of some very important files in the `/downloads` directory, including a mysqli conversion utility, the original website template, the original University template, and the original NCBI script. This directory should also contain copyright permissions for images in use by the website.

I started HTTP service using the command `sudo /etc/init.d/httpd start`.

Permissions

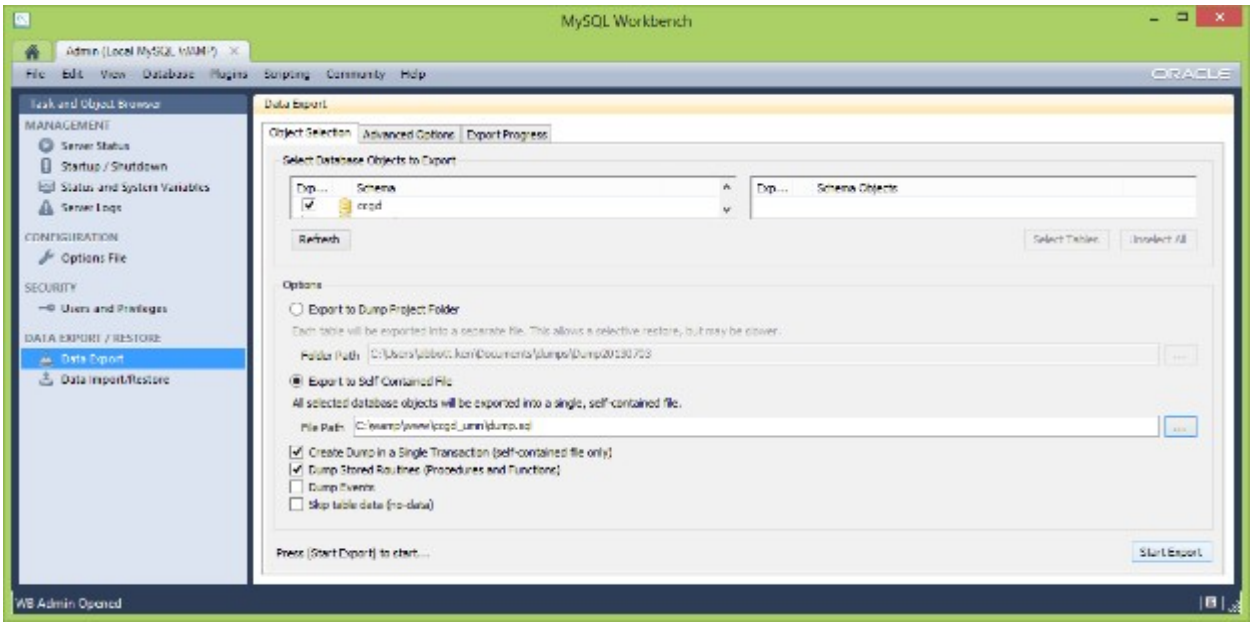
In the SSH client I used `chmod g=rwx /swadm -R` to recursively add write ability for `swadm` group users in `/swadm` following Apache configuration. I used `chmod u=rwx /swadm -R` to recursively add read and execute ability for owner `swadm` following Apache configuration. I used `chmod o=rwx /swadm/var/www/html/pl -R` to recursively add read, write, and execute ability for `apache` (and other maintainers with SFTP access) to the Perl update directory. I used `chmod o=rwx /swadm/var/log` to add read, write, and execute ability for MySQL to the log directory. I used `touch /swadm/var/log/mysql_slow_query.log` and `chmod o=rw`

`/swadm/var/log/mysql_slow_query.log` to allow read and write ability for MySQL to the log file. I used `ls –l` to verify permissions.

It may be necessary to run `chmod` or `chown` commands in the future since files sent through SSH connections may show the sender as owner and may have more restrictive permissions.

MySQL

I prepared a dump of the MySQL database using MySQL Workbench.



I saved this to the local website root and uploaded it to the side as described in the Apache configuration section.

I followed configuration instructions on the OIT Linux wiki. In the SSH client I started the MySQL server with `sudo /sbin/service mysqld start`. I then ran `/usr/bin/mysql_secure_installation`, which included setting a root password. I used `sudo /sbin/chkconfig mysqld on` to have MySQL start upon boot. I launched the MySQL shell using a `mysql –u root –p`. In the MySQL shell I used `CREATE DATABASE ccgd; USE ccgd; SOURCE /swadm/var/www/html/dump.sql` to create and load the database. I later used `mv /swadm/var/www/html/downloads/dump.sql /swadm/var/dump.sql` to transfer the MySQL dump outside of the website directory.

I used `sudoedit etc/mysql/my.cnf` to make MySQL configuration changes:

1. `# CCGD additions start`
2. `ft_min_word_len = 1`
3. `ft_stopword_file = ""`
4. `group_concat_max_len = 4096`
5. `query_cache_size = 10485760`
6. `key_buffer_size = 134217728`
7. `myisam_sort_buffer_size = 134217728`
8. `long_query_time = 1`
9. `slow_query_log = 1`
10. `slow_query_log_file = /swadm/var/log/mysql_slow_query.log`
11. `expire_logs_days = 10`
12. `max_allowed_packet` from `1M` to `10M`

13. `# CCGD additions end`

I then used `sudo service mysqld restart`.

PHP

I used `sudoedit etc/php.ini` to make PHP configuration changes:

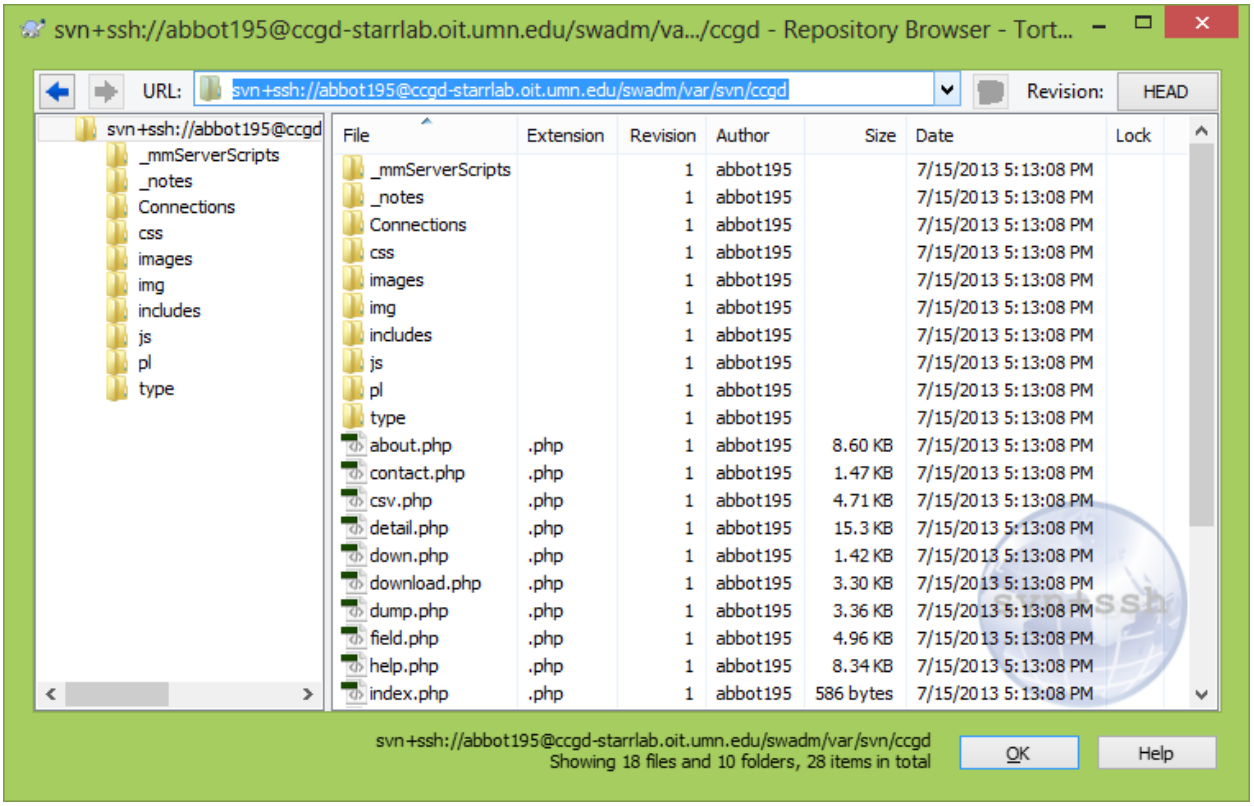
1. `memory_limit` from `128M` to `1024M`
2. `upload_max_filesize` from `2M` to `50M`
3. `sendmail_path = /usr/sbin/sendmail -F CCGD -f ccgd@umn.edu -t -i`

I preceded each of these changes with `; CCGD change` because the changes were located throughout the file. I restarted Apache with `sudo /sbin/service httpd restart`.

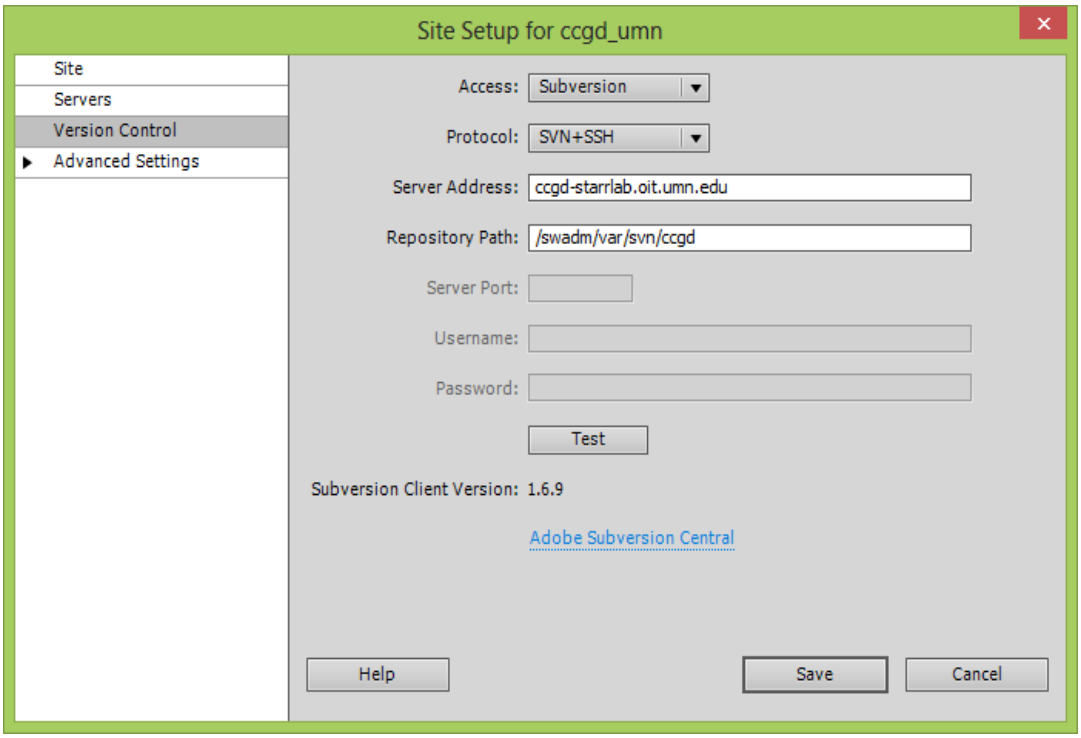
Subversion

In the SSH client I used `mkdir /swadm/var/svn` and `mkdir /swadm/var/svn/ccgd` to create directories for the Subversion repository. I used the command `svnadmin create /swadm/var/svn/ccgd` to add basic Subversion files to the repository directory.

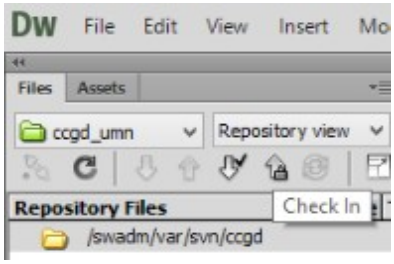
I installed the TortoiseSVN Subversion client in Windows, adding `ssh = $SVN_SSH C:/Program Files/TortoiseSVN/bin/TortoisePlink.exe -i C:/Users/abbott.ken/Documents/School/Lab/PuTTY/puttygen.ppk` to the `[tunnels]` section of `C:\Users\abbott.ken\AppData\Roaming\Subversion\config`. I verified that I could access the repository in the TortoiseSVN client.

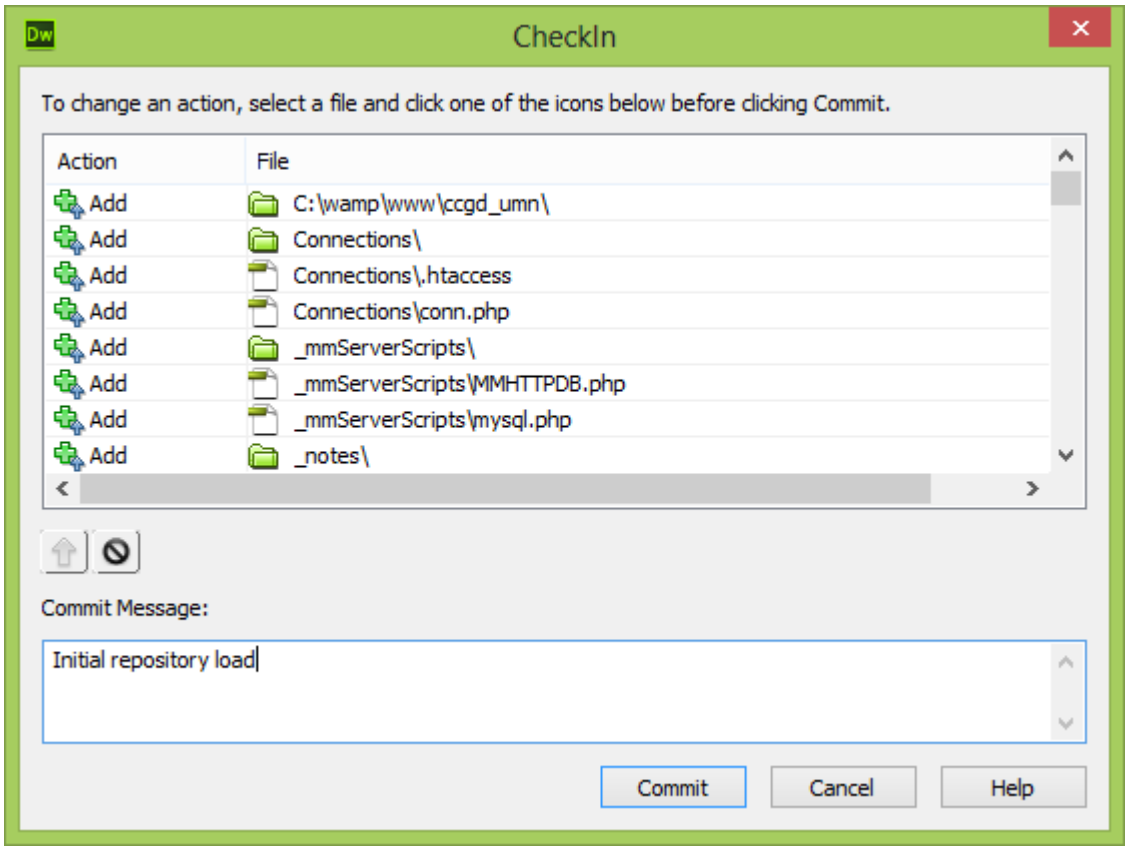


I configured the Subversion repository connection in Dreamweaver.

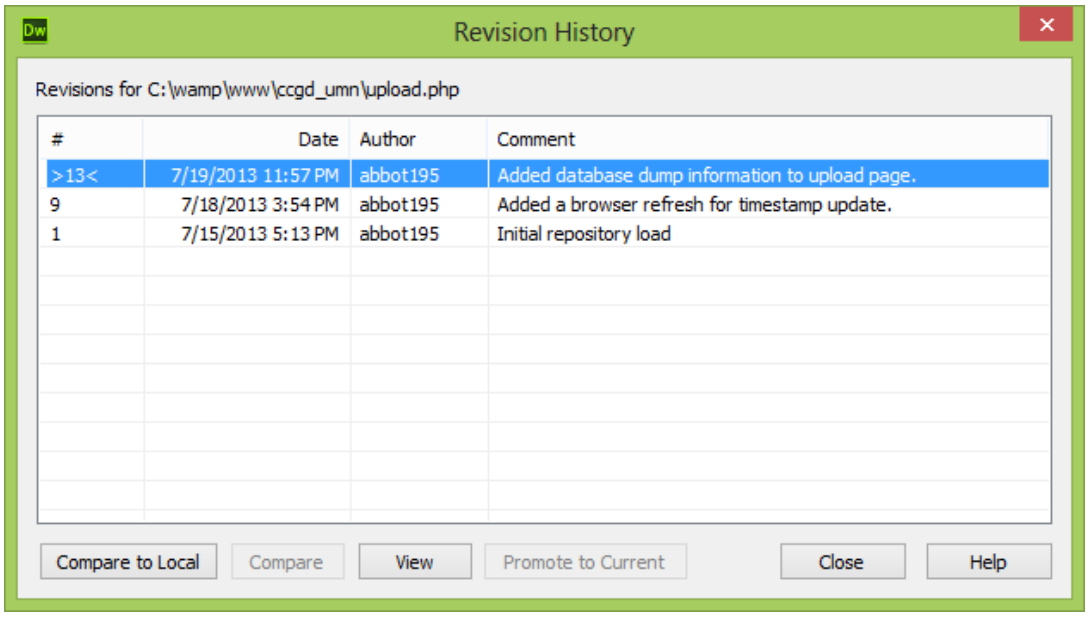


I used the CheckIn button to commit the entire local website to the repository.



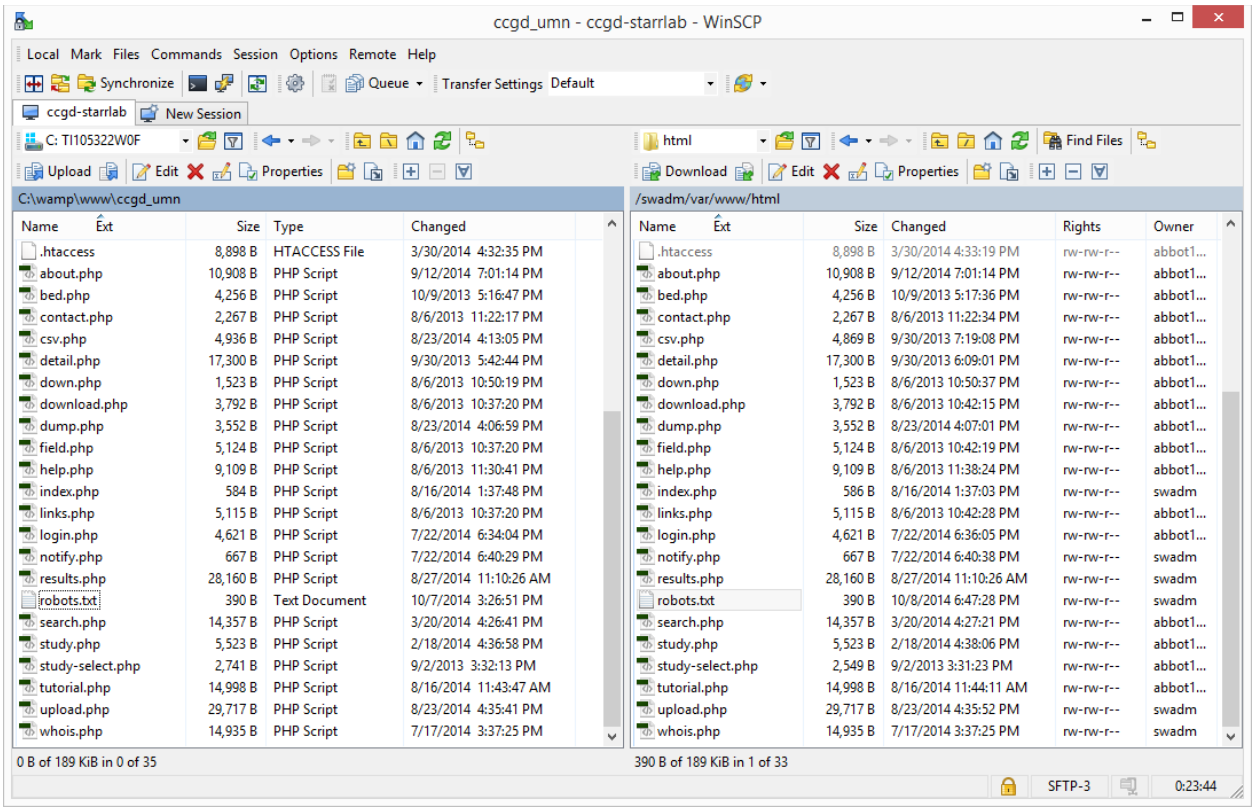


I made some changes and verified ability to revert to previous versions.

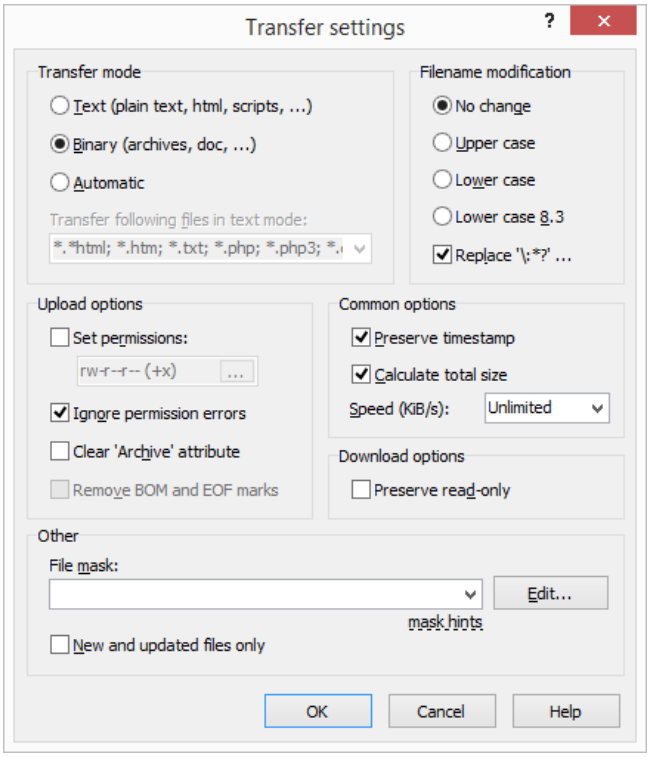


WinSCP

I installed the WinSCP client on my development machine.



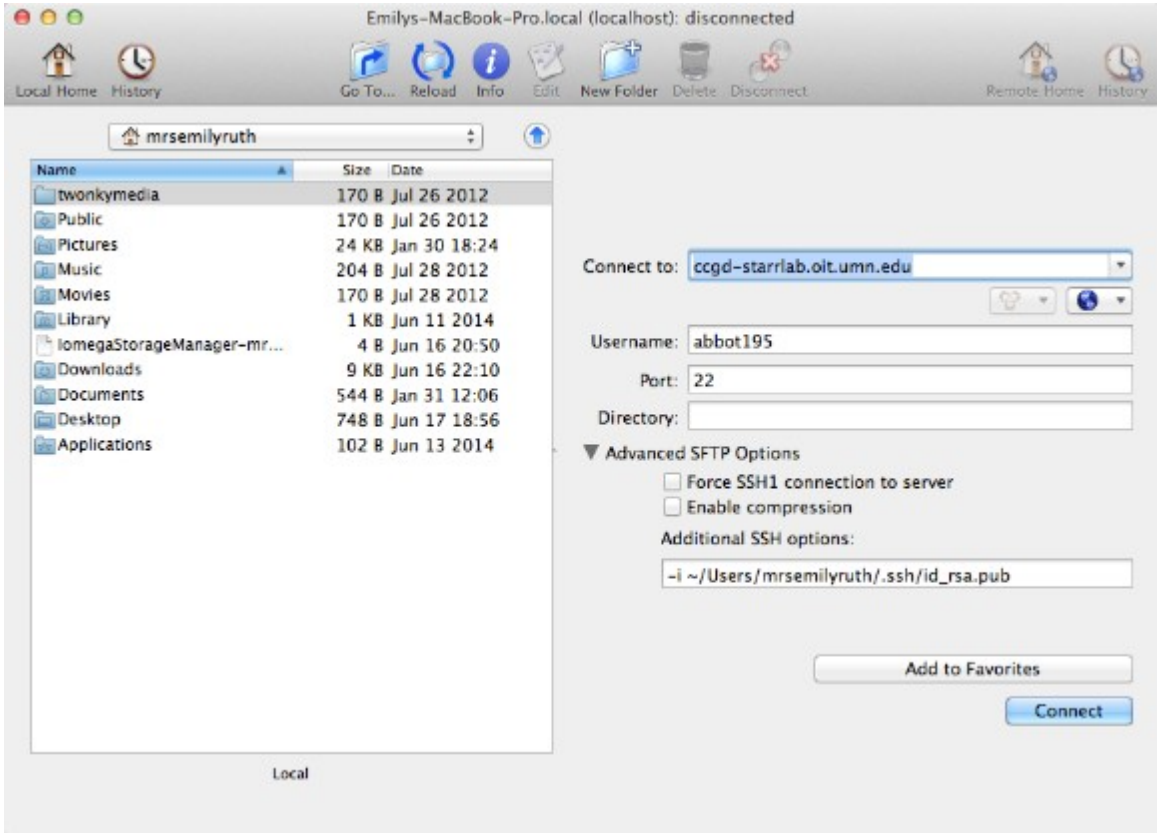
The WinSCP installation detected my PuTTY session data, including pointers to the SSH keys I used in order to sidestep Mkey or Duo authentication. I set WinSCP to ignore permissions errors.



I then used WinSCP to transfer files from my development machine to the production website. However, I still used Dreamweaver to check files in and out of the Subversion repository.

Fugu

On a Mac, I configured the Fugu SFTP client to use my SSH key.



Cron

I used `sudo su swadm` to switch to the `swadm` user before running `crontab -e` and making the following entries:

Had I run `crontab` under my own username, none of these entries would have appeared. These entries kicked off an update process at 2000 which would be verified (and possibly retried at 2300 and 0200) before an email report at 0500. The entries overwrite and then add to `cron.log`, which shows all terminal output from the previous night's update process. The email report coincided with a link check and a MySQL dump.

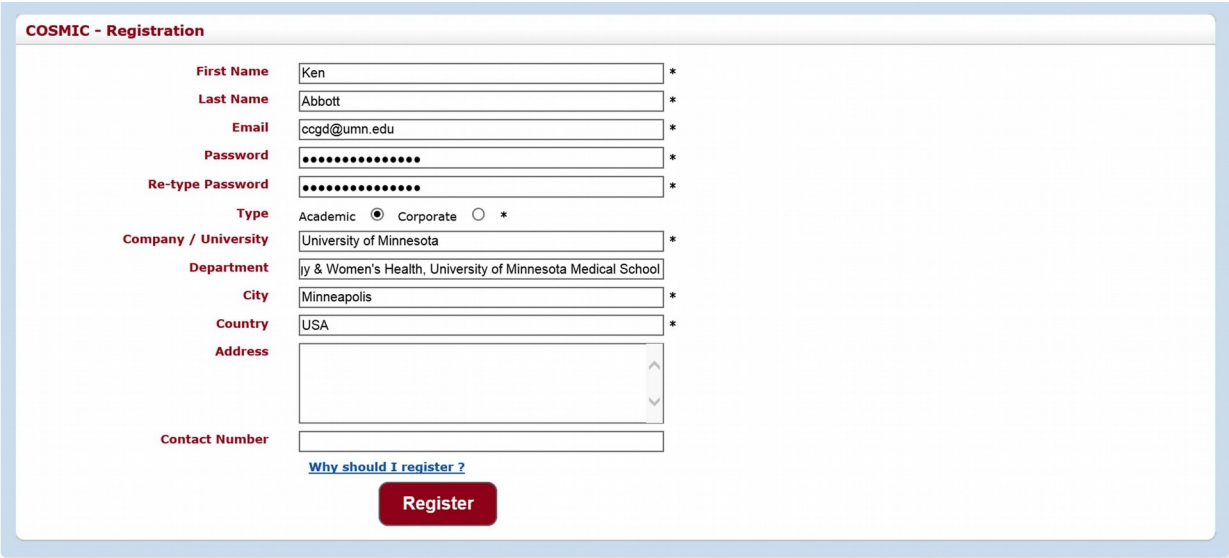
Strings

I verified that `includes/email.php` both used ccgd@umn.edu as an email address. I verified that `pl/email.php` included ccgd@umn.edu, abbot195@umn.edu, and star0044@umn.edu email addresses. I verified

that `pl/autoload.php` and `upload.php` both contained the appropriate `/swadm` path. I verified that `pl/ncbi.pl`, `pl/update.pl`, `pl/gmail.pl`, and `pl/excel2txt` all contained the Perl path returned by `which perl`. I verified that `Connections/conn.php` and `pl/update.pl` both contained the correct MySQL login and password. I verified that `ccgd_users` contained the correct administrative username and password. I verified that `pl/functions.php` and `upload.php` contained a login and password for NetFiles uploads; the NetFiles share owner must invite the user by email address and OIT must enable NetFiles access for any sponsored accounts. I verified that `pl/sanger.pl` and `pl/sanger.exp` contained a Sanger email address and password.

Sanger

I used the ccgd@umn.edu email address to register for Sanger downloads at https://cancer.sanger.ac.uk/cosmic/register.



This required email activation.

From: no-reply-cosmic@sanger.ac.uk
Date: August 9, 2014 at 11:31:42 AM CDT
To: ccgd@umn.edu
Subject: Cosmic-Registration

Cosmic-Registration

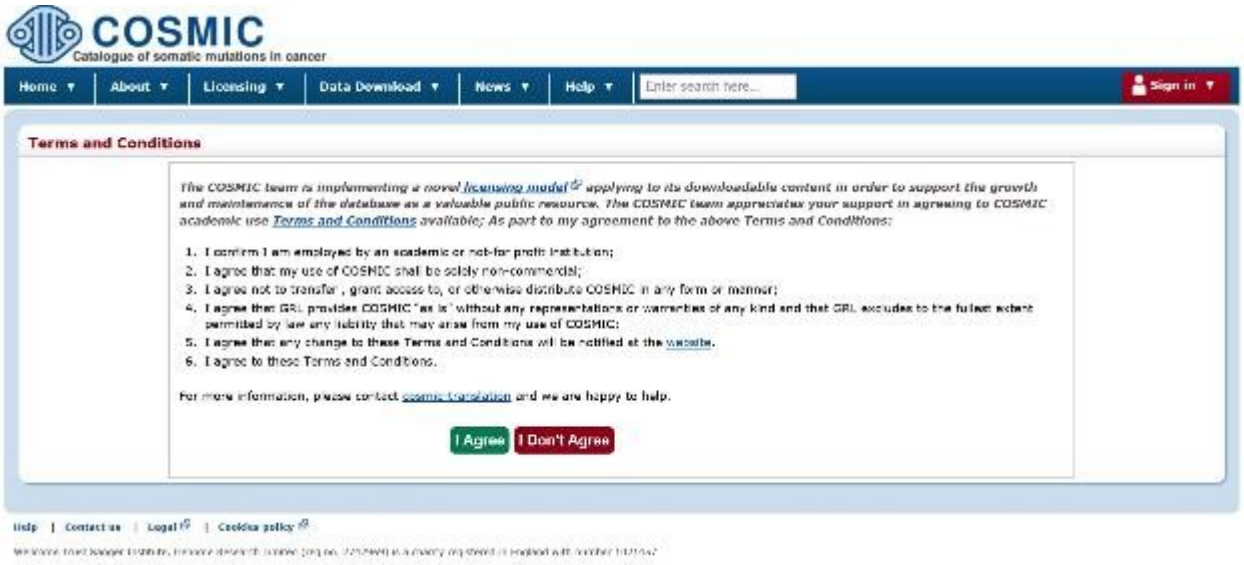
Dear Ken Abbott,

Thanks for showing interest in the cosmic data.

Your account has been approved. Please click this link to activate your account.

or alternatively, please go this url:
https://cancer.sanger.ac.uk/cosmic/user/activate9mrlSJ8E2594ie38AjyH

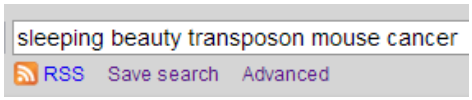
This later required acceptance of license terms and account reactivation.

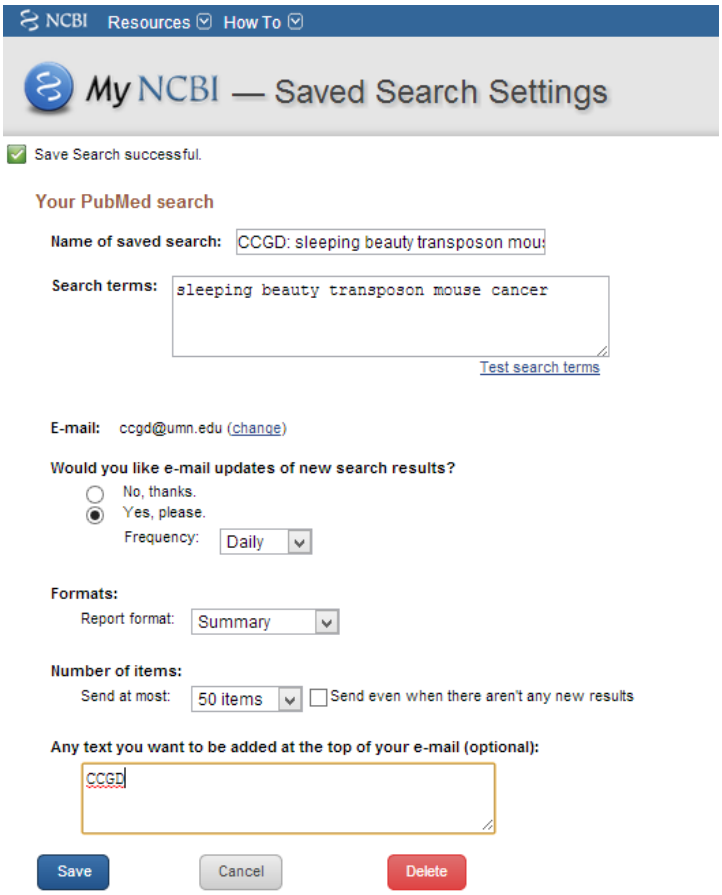


I made an initial connection to the Sanger server in order to allow for future automation.

PubMed

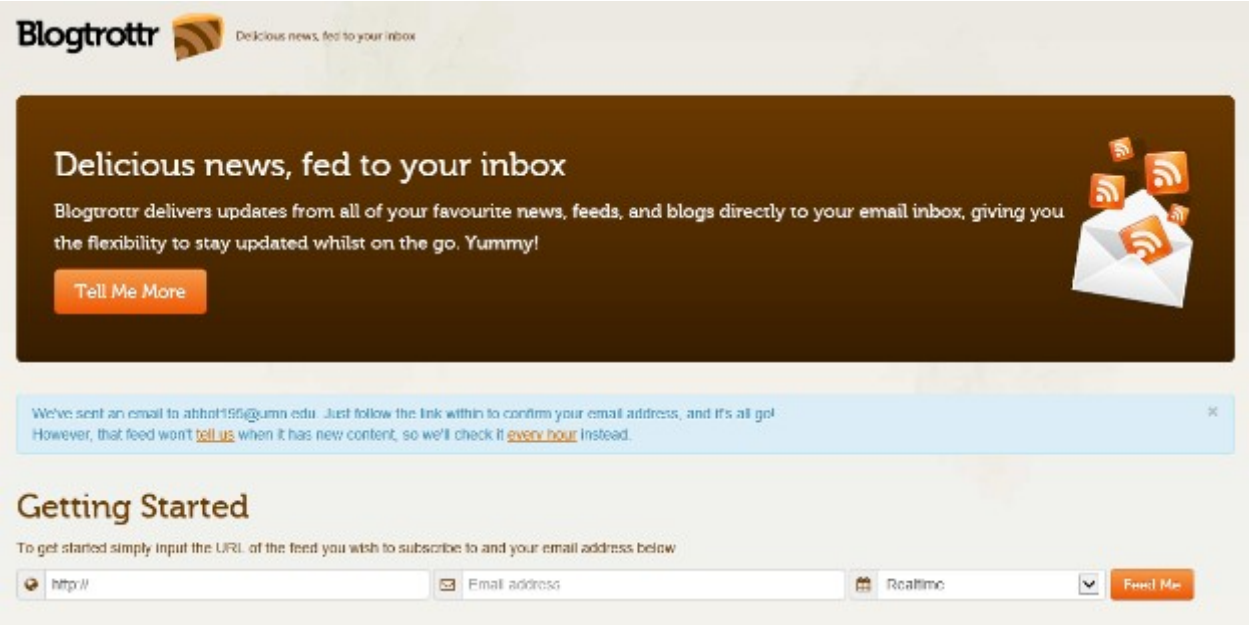
I used the ccgd@umn.edu email address to subscribe to NCBI updates through an advanced search feature.





Gene and HomoloGene

I signed up for email forwards of Gene and HomoloGene RSS feeds using blogtrottr.



Testing

Files

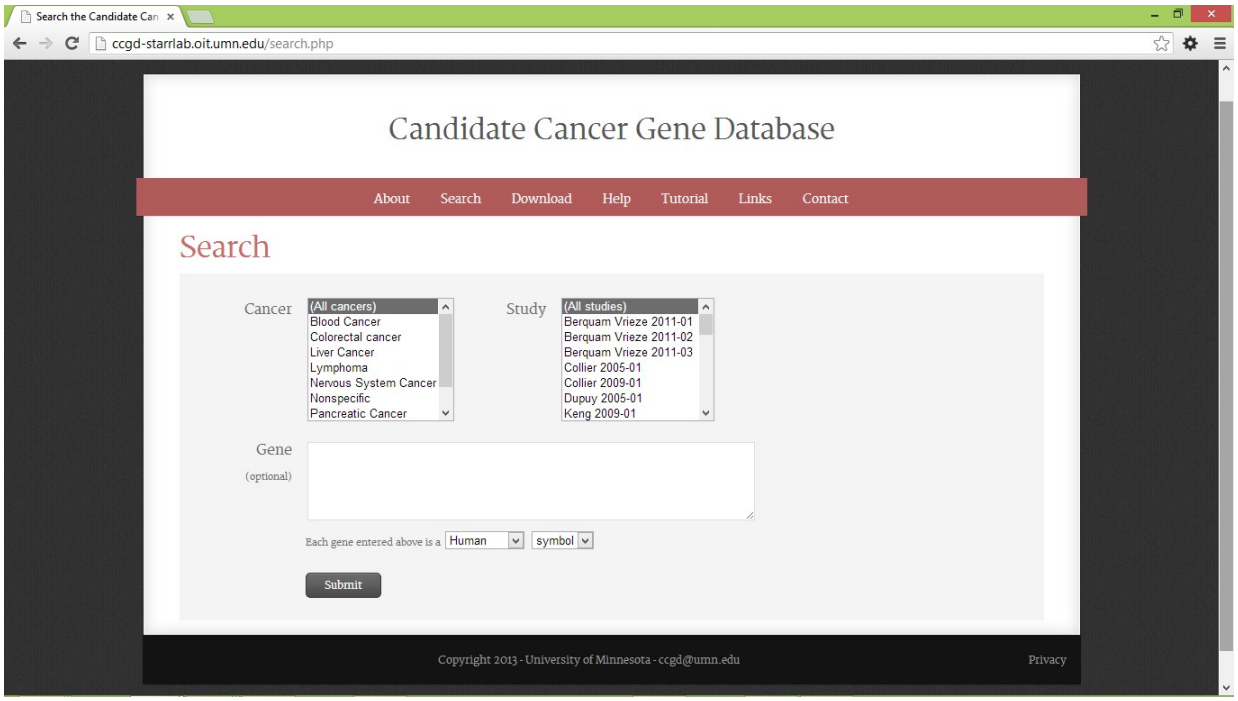
I verified that file uploads completed successfully. The text files below contain file names, file sizes, and directory structure for the entire website and database. The DOS command to create this file is `dir *.* /s > directory.txt`. The equivalent Linux command `ls –laR > directory.txt`.

directory.txt

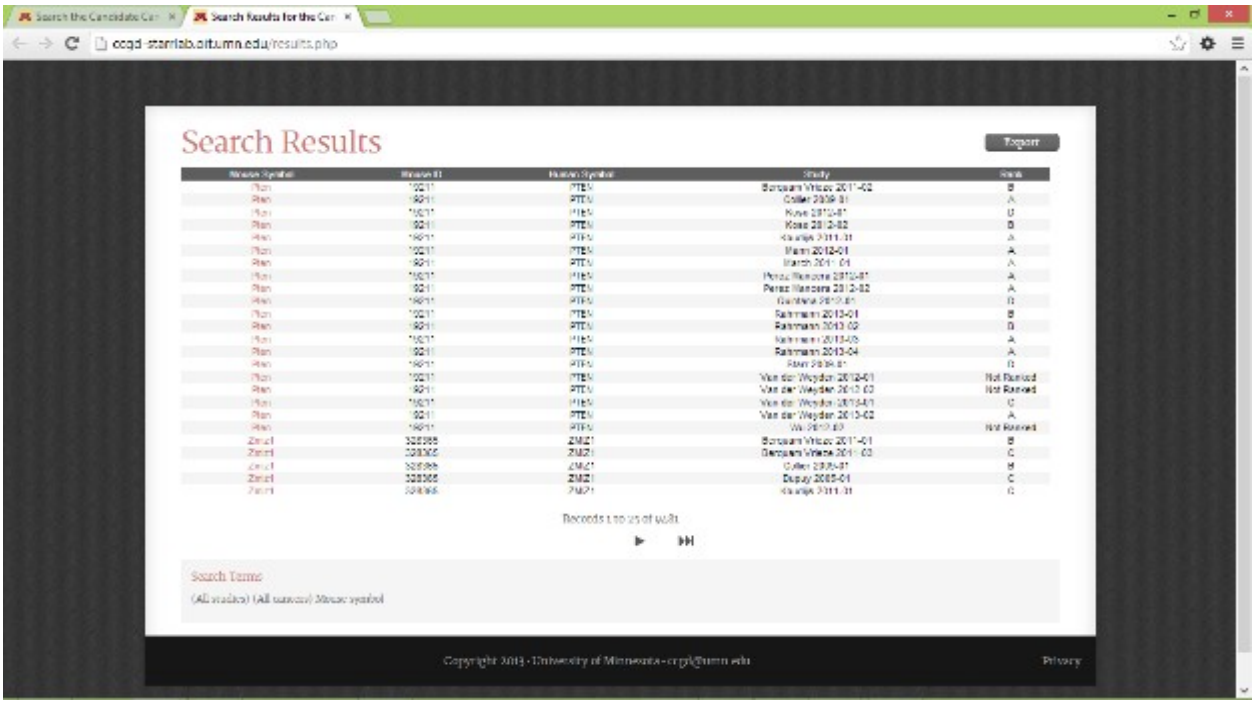
I used commands like `find . -name '*.php' | xargs wc -l` to get line counts for custom `.php` (3,789), `.pl` (447), and `.css` (403) files.

Web and Database Server

I pointed my browser to http://ccgd-starrlab.umn.edu/. I verified that the search page loaded completely. This proved that the webpages were connecting successfully to the database server in order to load field options. I clicked on links throughout the site to verify function. The screenshot below shows the first complete search page ever produced by the live site.



I submitted a query and verified that the results page loaded. This proved that the database was executing the `query_explode` routine.



I submitted small and large queries. The 100,000 gene (740,000 character) query that took 9 seconds in the development environment required approximately 26 seconds over my home internet connection. Before moving on to other testing, I flushed these queries from `ccgd_log`.

An ApacheBench test for 100 requests (maximum 10 concurrent) showed that the server typically produced `search.php` in less than a second.

```
C:\WINDOWS\system32\cmd.exe

C:\wamp\www\ccgd_umn>ab -n 100 -c 10 http://ccgd-starrlab.oit.umn.edu/search.php

This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking ccgd-starrlab.oit.umn.edu (be patient).....done


Server Software:        Apache/2.2.15
Server Hostname:        ccgd-starrlab.oit.umn.edu
Server Port:            80

Document Path:          /search.php
Document Length:        20153 bytes

Concurrency Level:      10
Time taken for tests:   30.795 seconds
Complete requests:      100
Failed requests:        0
Write errors:           0
Total transferred:      2032600 bytes
HTML transferred:       2015300 bytes
Requests per second:    3.25 [#/sec] (mean)
Time per request:       3079.495 [ms] (mean)
Time per request:       307.950 [ms] (mean, across all concurrent requests)
Transfer rate:          64.46 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       41  305 2398.4     59   24048
Processing:   129  572 100.2    584     693
Waiting:       49  254 140.8    231     631
Total:        205  877 2406.1    647   24677

Percentage of the requests served within a certain time (ms)
  50%    647
  66%    672
  75%    684
  80%    692
  90%    729
  95%    754
  98%    762
  99%  24677
 100%  24677 (longest request)

C:\wamp\www\ccgd_umn>_
```

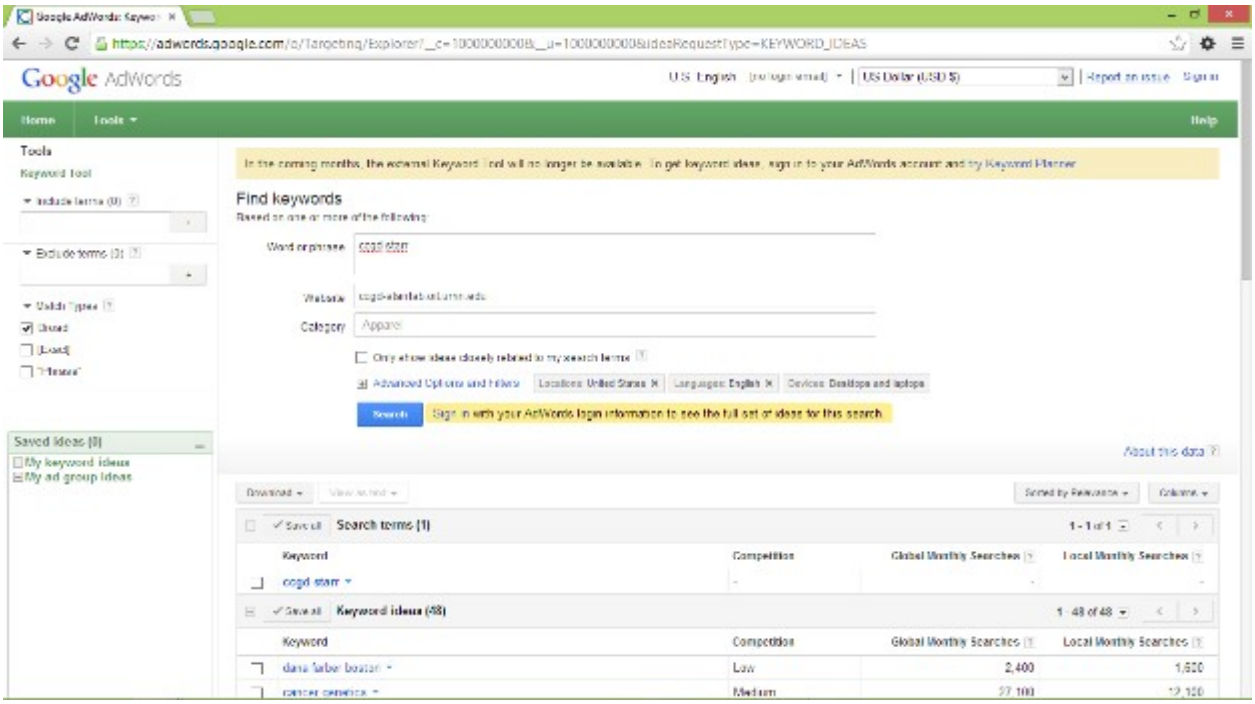
Another ApacheBench test for 1000 requests (maximum 100 concurrent) showed that the server needed several seconds to produce `search.php`.

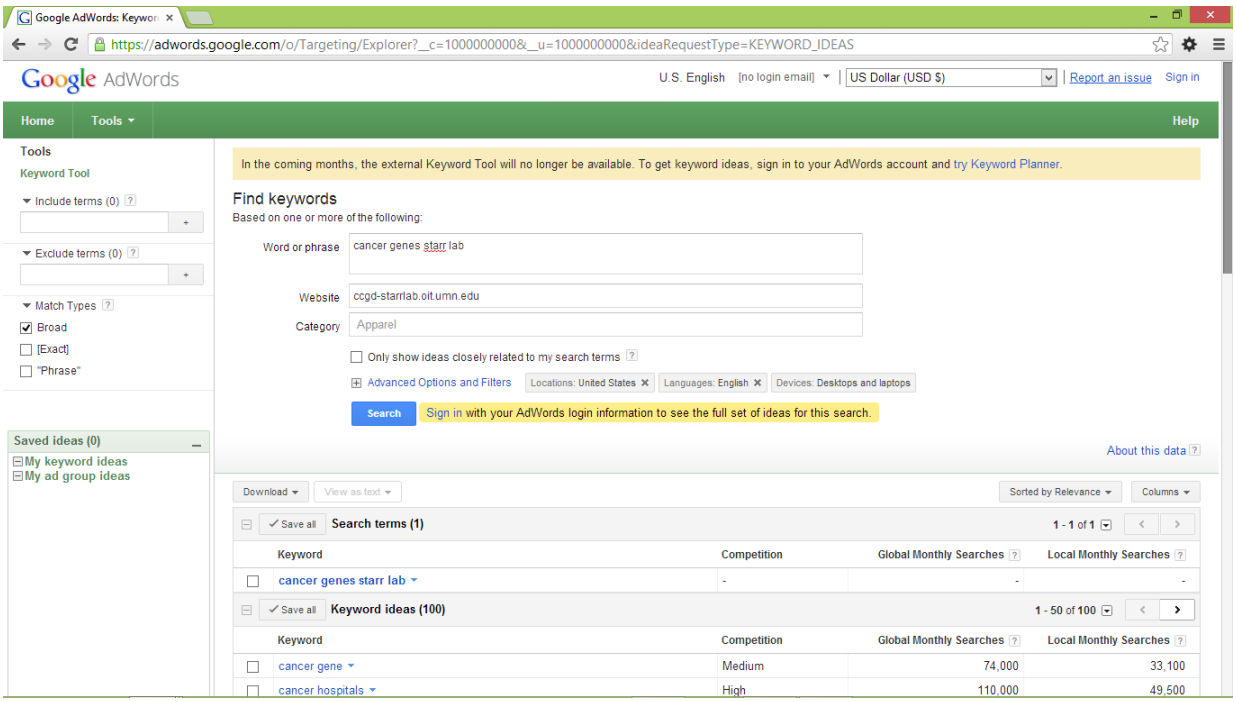
```
C:\WINDOWS\system32\cmd.exe
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking ccgd-starrlab.oit.umn.edu (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:        Apache/2.2.15
Server Hostname:        ccgd-starrlab.oit.umn.edu
Server Port:            80

Document Path:          /search.php
Document Length:        20153 bytes

Concurrency Level:      100
Time taken for tests:   71.616 seconds
Complete requests:      1000
Failed requests:        0
Write errors:           0
Total transferred:      20326000 bytes
HTML transferred:       20153000 bytes
Requests per second:    13.96 [#/sec] (mean)
Time per request:       7161.621 [ms] (mean)
Time per request:       71.616 [ms] (mean, across all concurrent requests)
Transfer rate:          277.17 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       36   70  39.6     62    1071
Processing:   120 6738 1399.4   6897   10901
Waiting:       55 3473 2073.3   3474    7823
Total:        182 6808 1401.2   6967   10942

Percentage of the requests served within a certain time (ms)
  50%   6967
  66%   7085
  75%   7170
  80%   7248
  90%   7910
  95%   8401
  98%   9176
  99%   9203
 100%  10942 (longest request)

C:\wamp\www\ccgd_umn>_
```

Google

AdWords

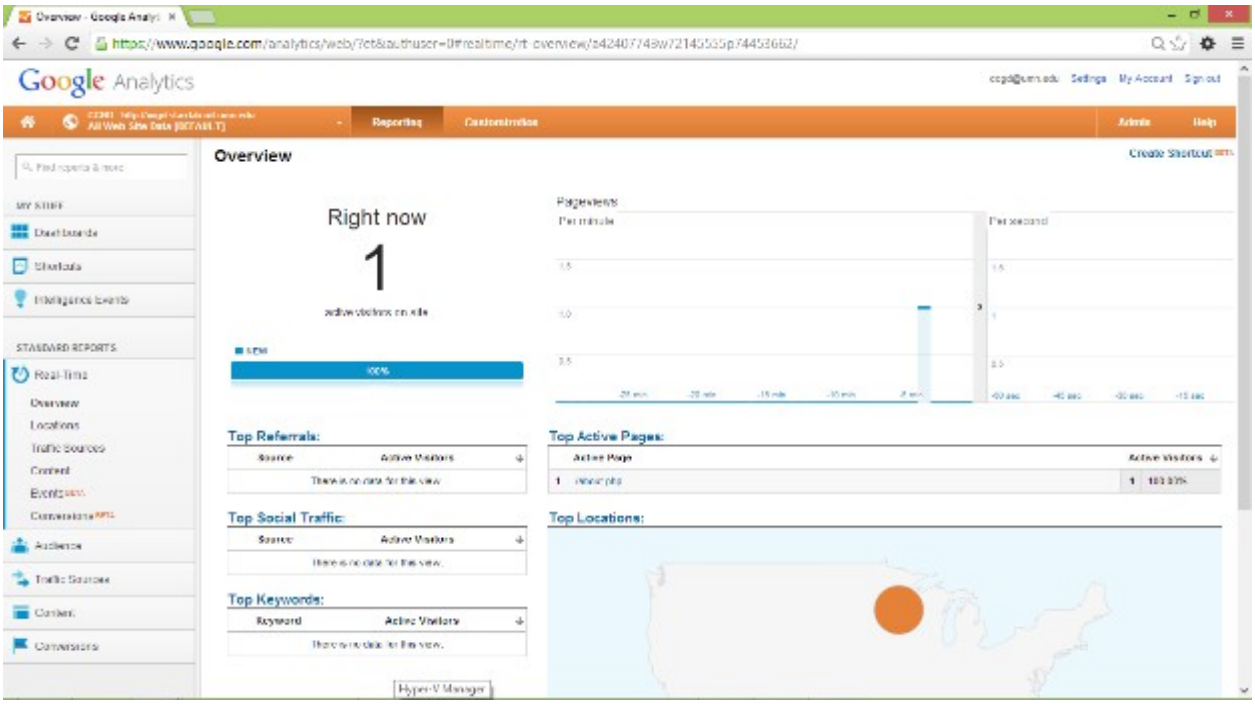
I noted no significant competition for phrases such as `ccgd starr` and `cancer genes starr lab`.





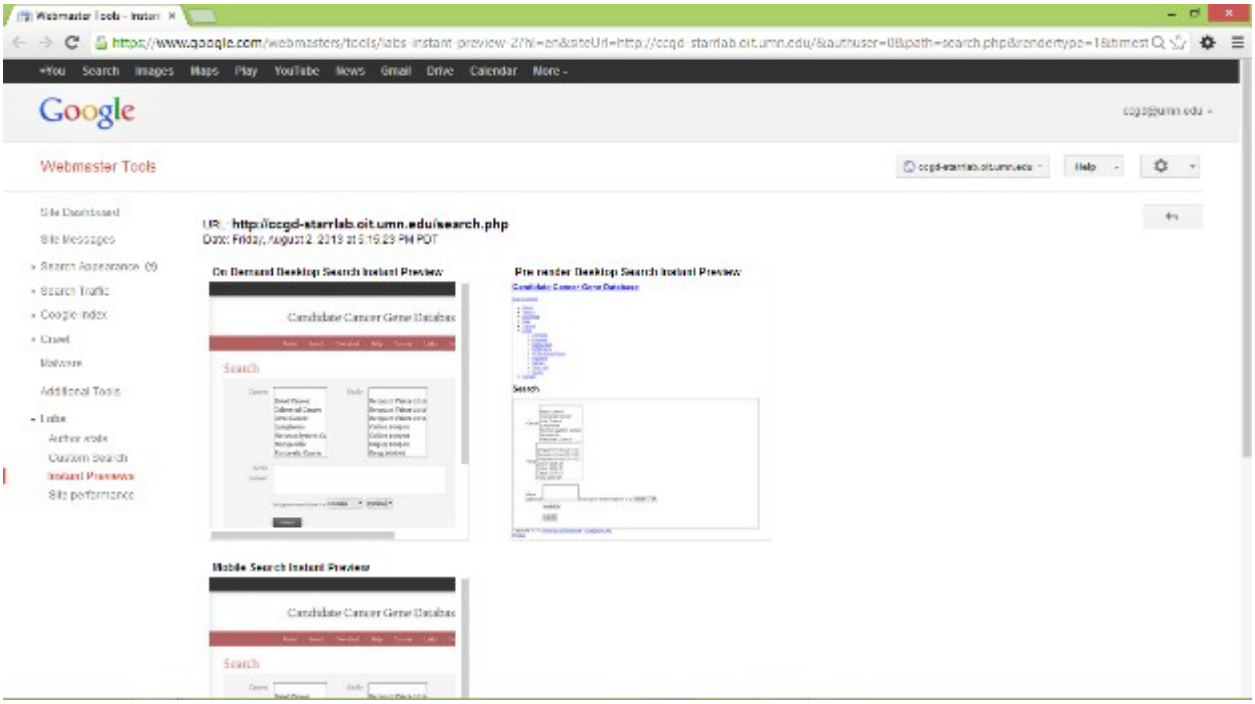
Analytics

I verified that my browsing activity registered in Google Analytics.



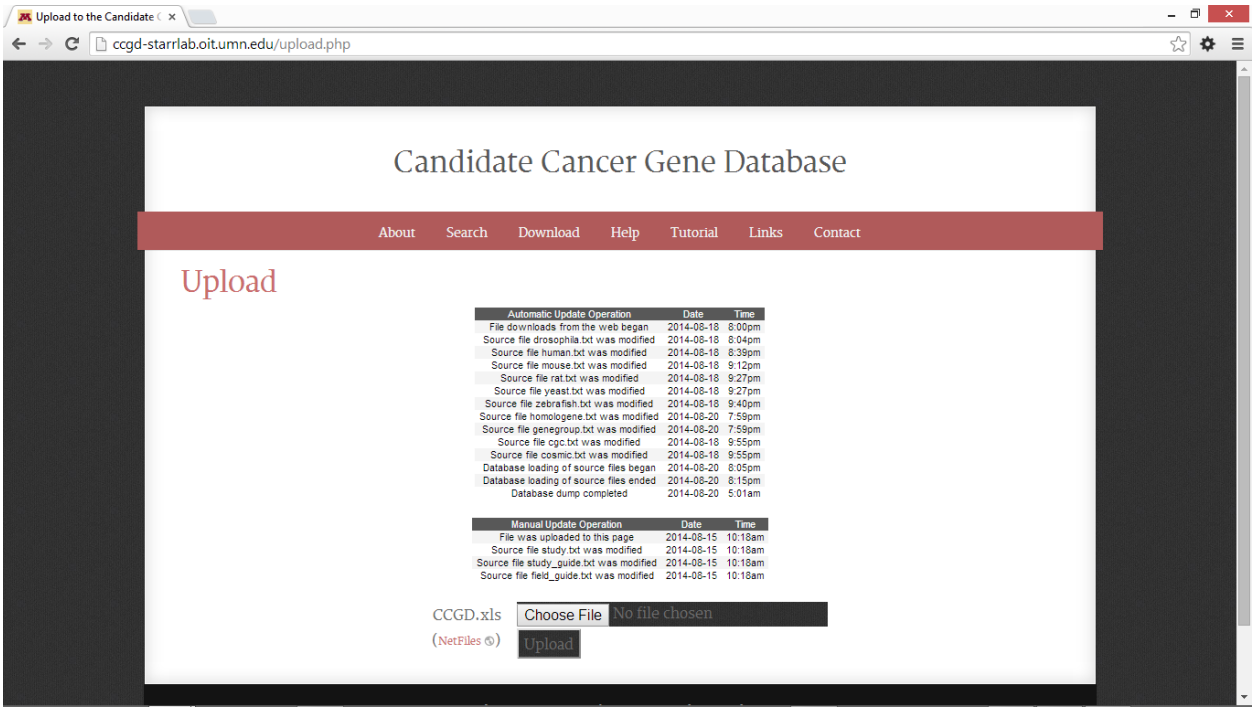
Webmaster Tools

I verified that Google could see the site.

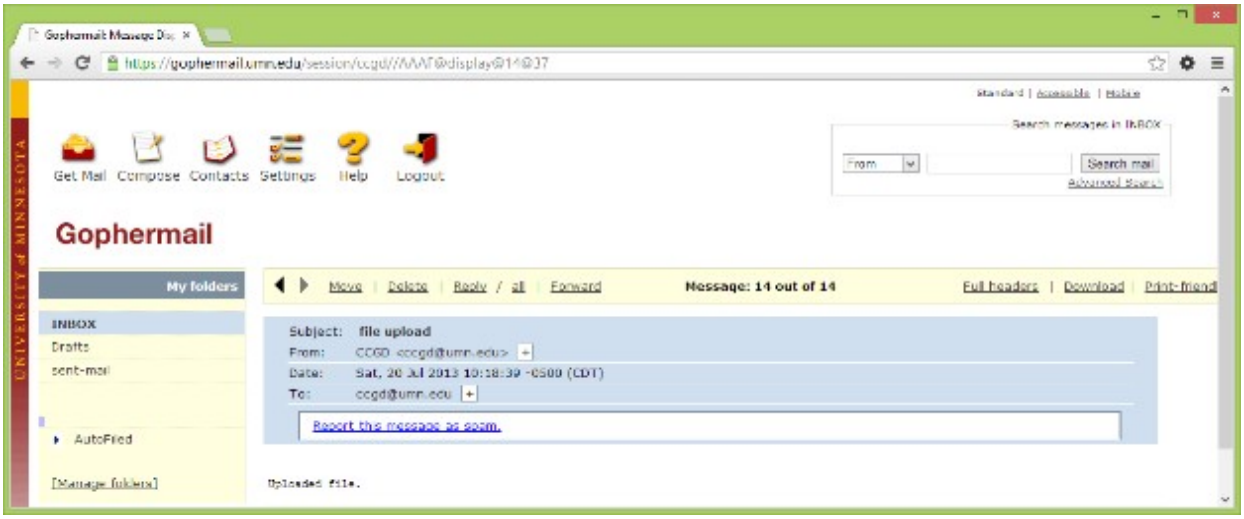


File Upload

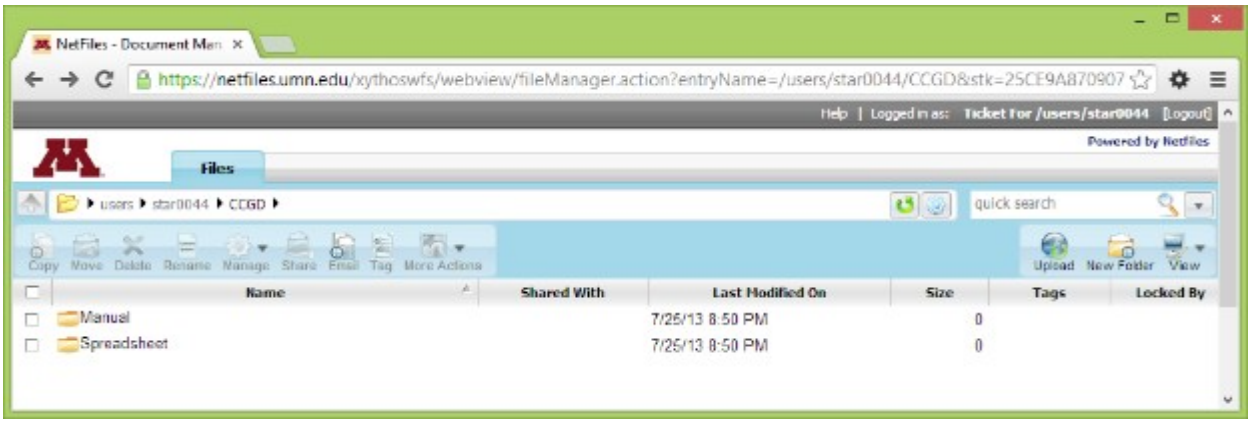
I tested file upload, noting timestamp evidence of successful Excel spreadsheet parsing and processing.



I also noted that the server sent a file upload email to the CCGD mailbox.

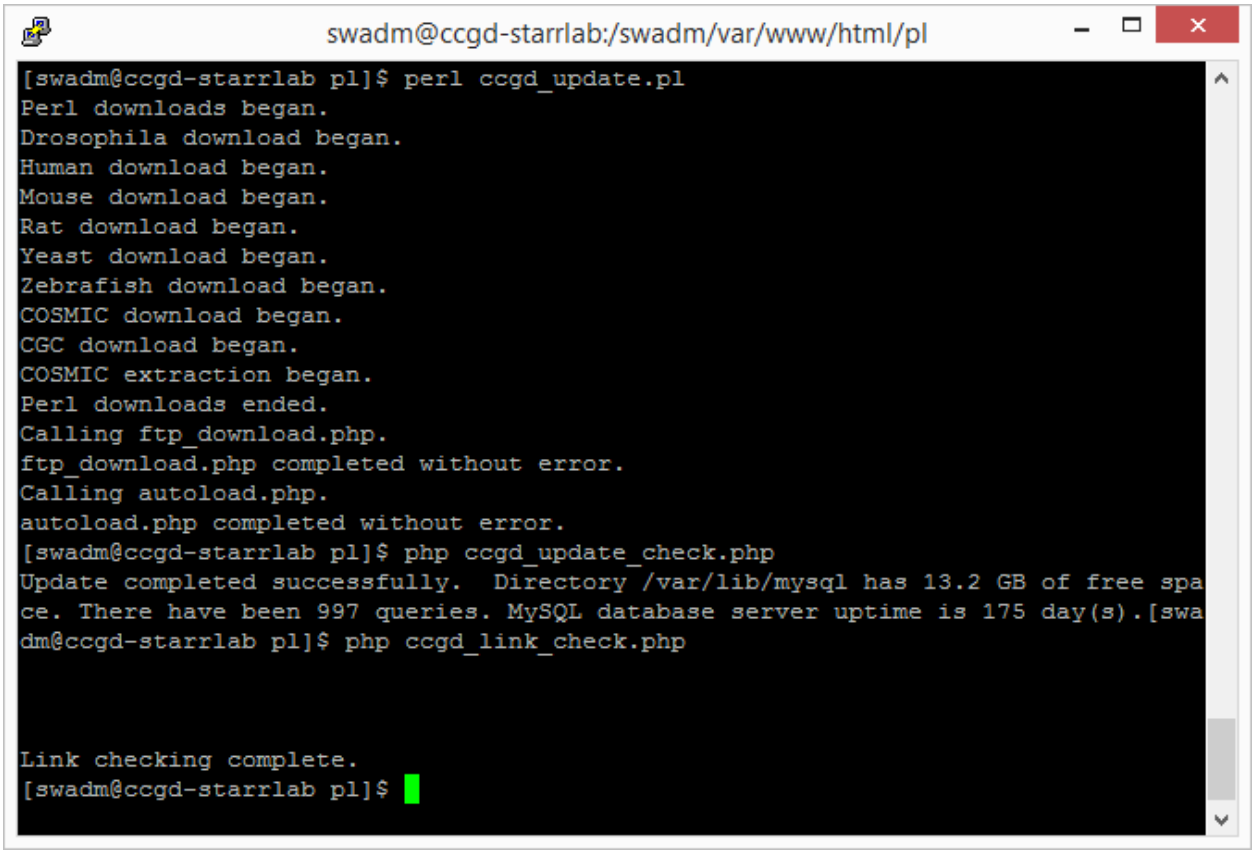


I sent the upload-ready file to the NetFiles share.



Update Scripts

I tested update tasks by manually triggering them in `/swadm/var/www/html/pl`, verifying that none of the scripts produced any error messages.



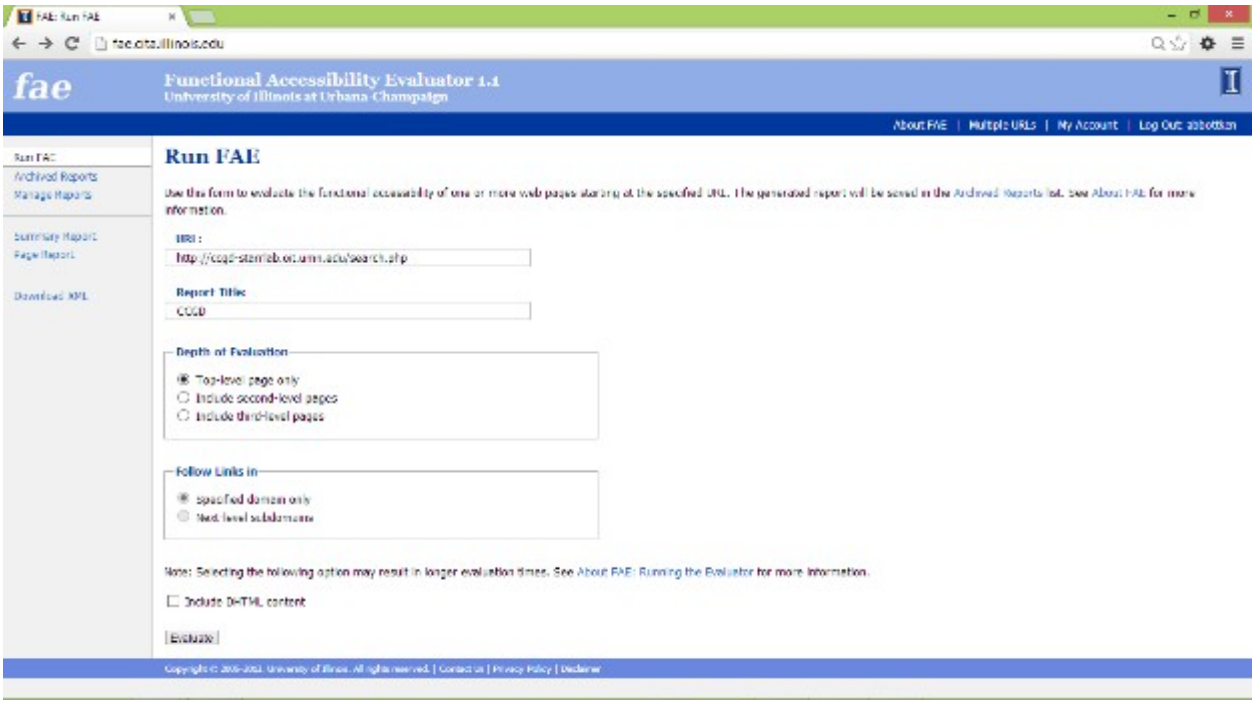
I used `ls -l` to verify that each regular and backup copy of each downloaded file showed a current timestamp.

Error Reporting

I turned on error message suppression by uncommenting `error_reporting(0);` in `includes/error.php`.

Accessibility

I used the Functional Accessibility Evaluator to verify that all pages passed accessibility tests.



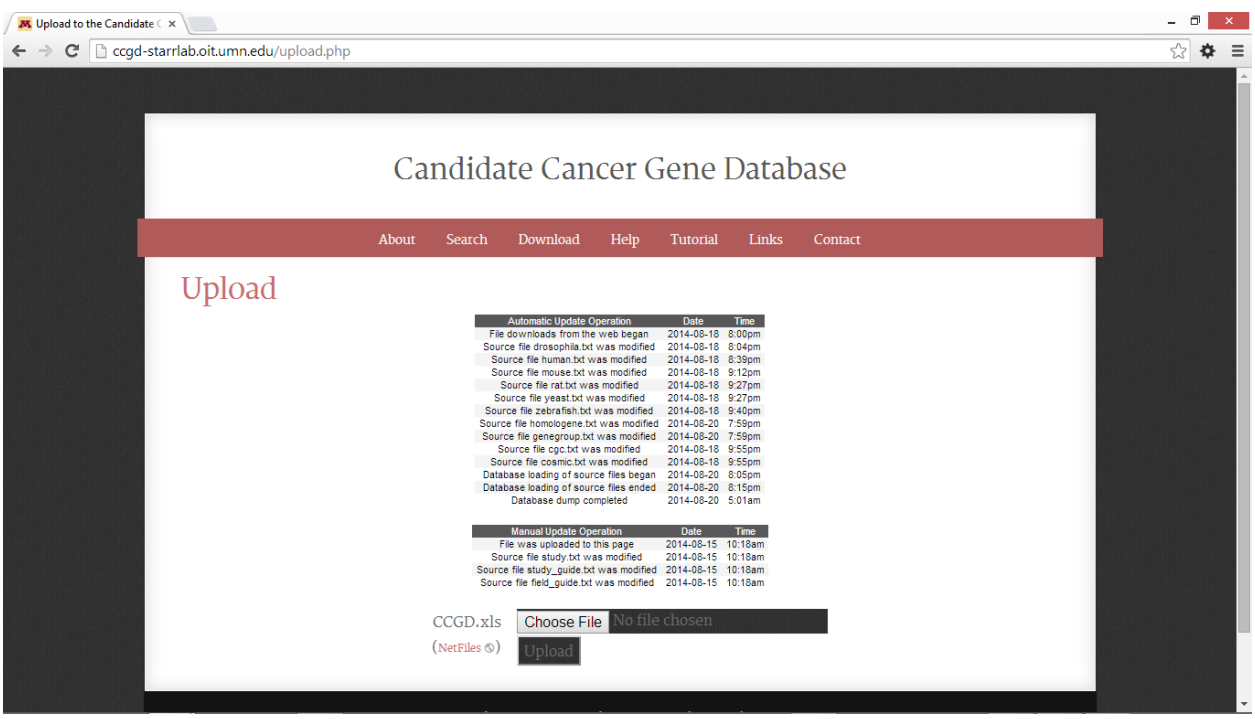
The tool complained about the absence of `h2` headings before the navigation bar, but the first line in the navigation bar file is a skip-to-content hyperlink. The tool reported no failures.

Scheduled Tasks

I let the updates run as scheduled tasks for several days. Updates succeeded on 8 of 9 days; on day 5, the update failed because `mouse.txt` was too small.

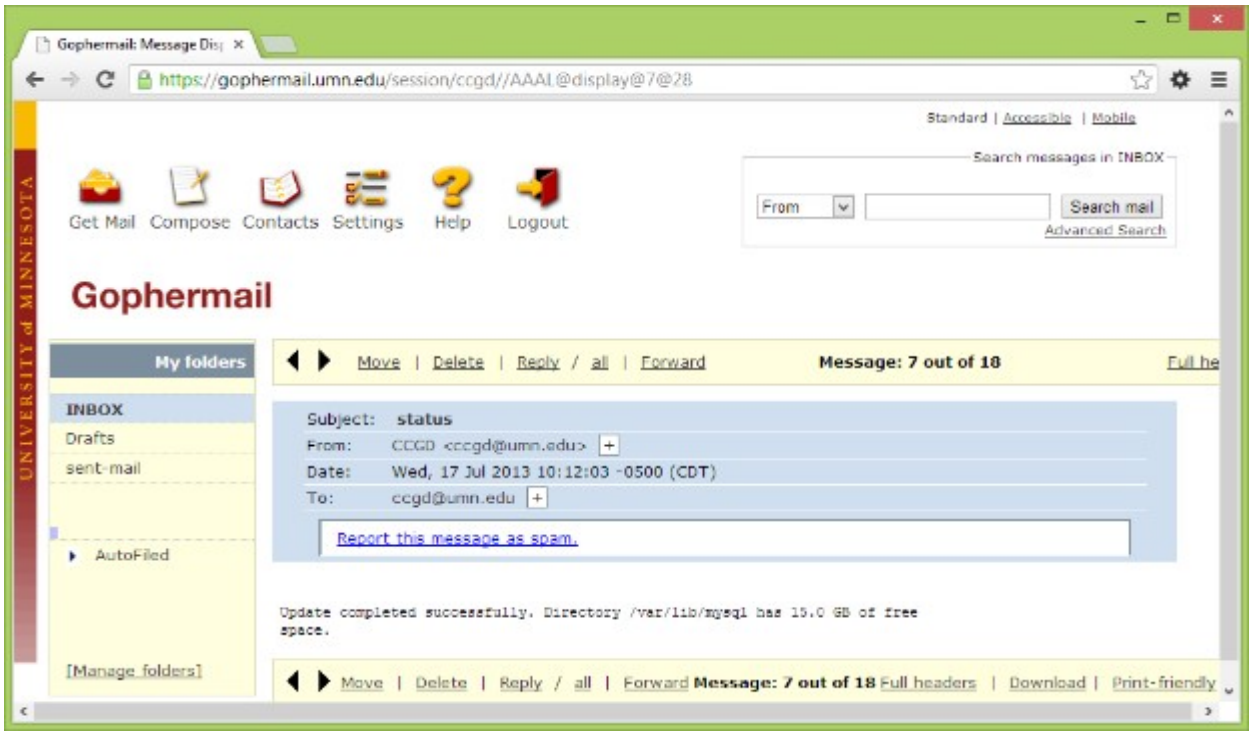
Update Process

I verified that the upload page showed current timestamps for all automatic update files.



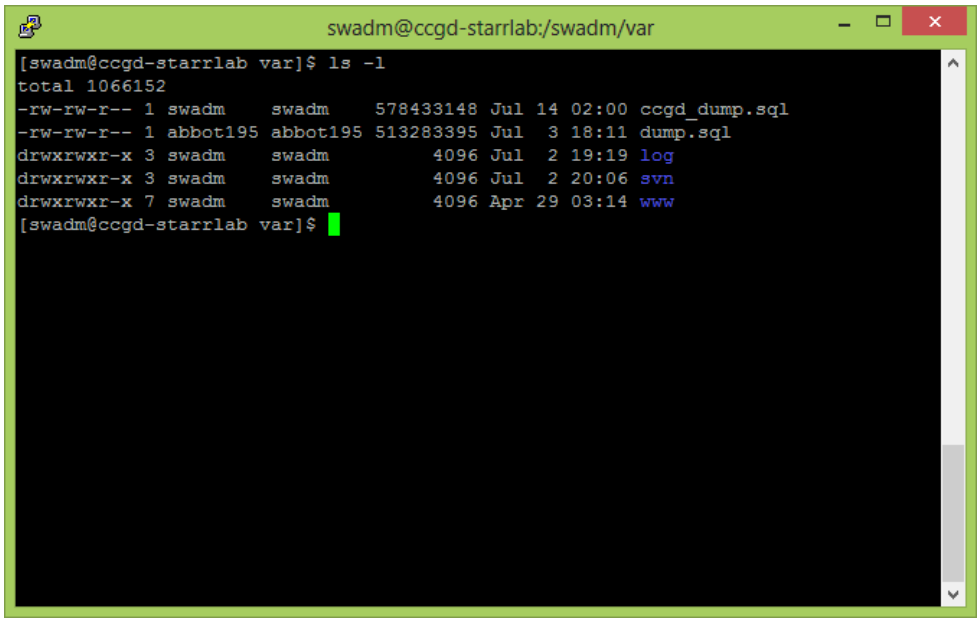
Update Check

I verified that there were no reported errors for the update check task.



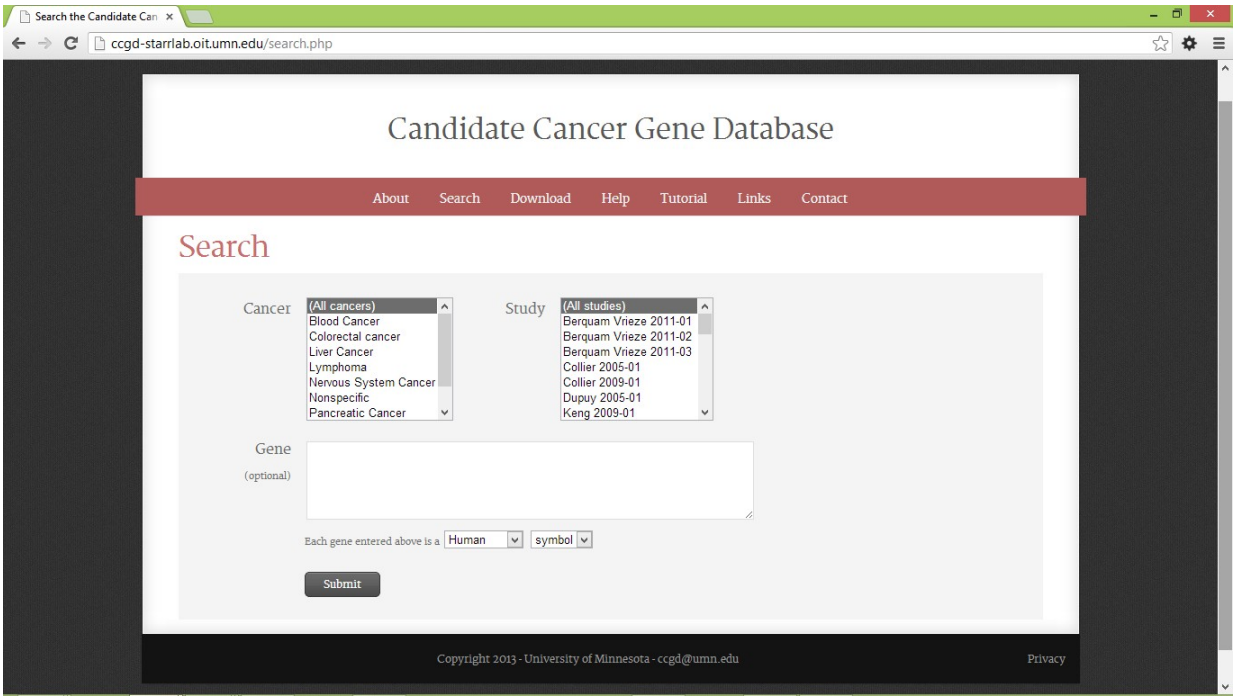
Database Dump

I verified that `/swadm/var/ccgd_dump.sql` showed a current timestamp.

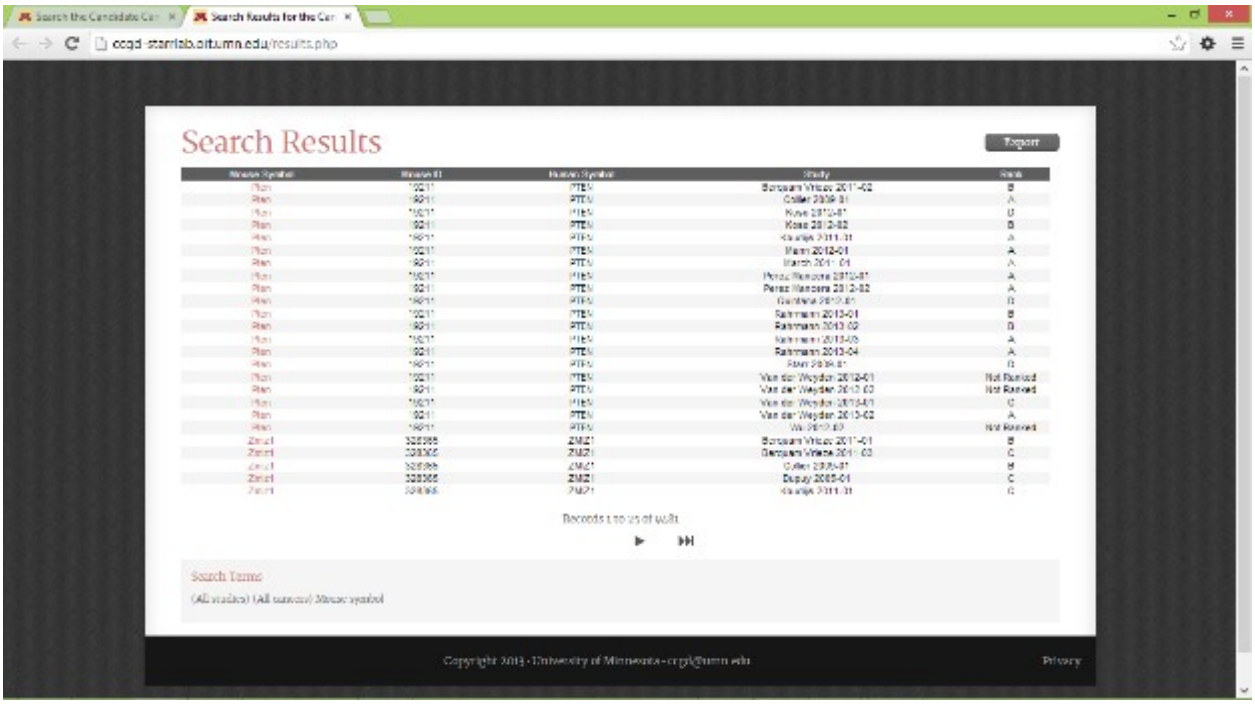


Webpage Function

I verified that the search page contained complete selection fields from the database.



I verified that the search results page produced a large result set.



Launch

Public Configuration

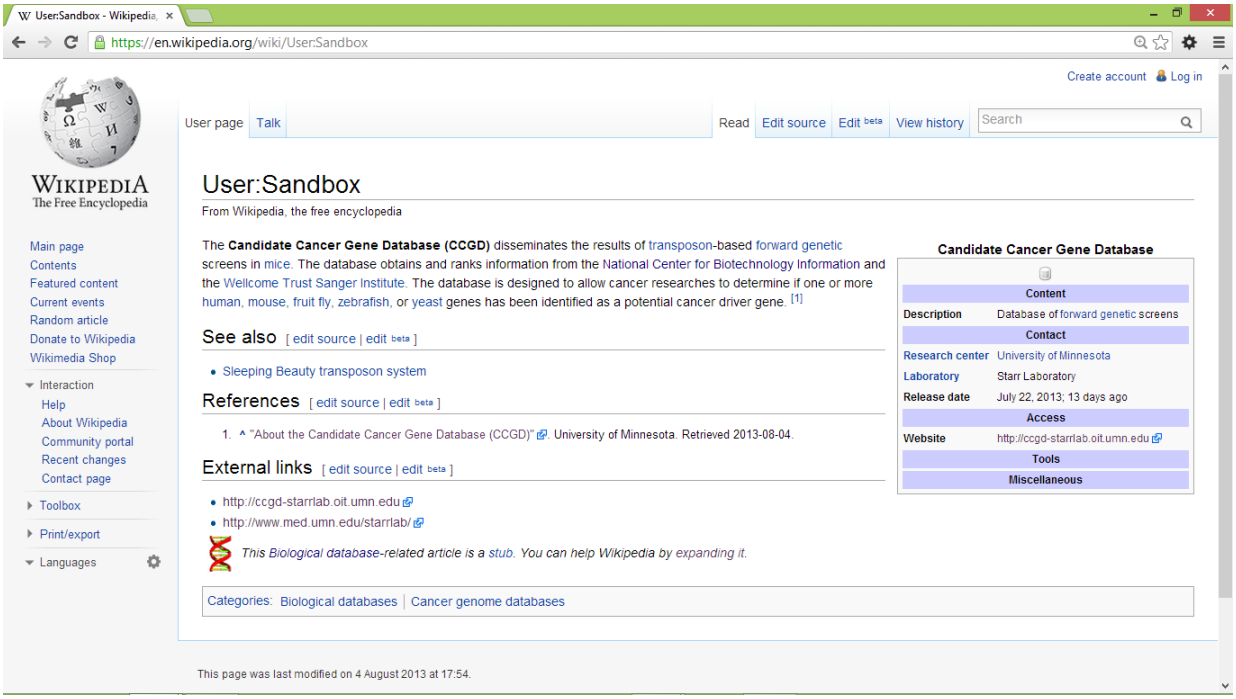
When I was satisfied that the website was functioning well, I turned off slow query logging in `/etc/my.cnf` by changing `slow_query_log` from `1` to `0`. The website was originally available only through the VPN connection; upon request, OIT opened HTTP port 80 to the world. The site went live at approximately 8:20am on 7/22/2013. I verified that the site produced a results page and that the upload process completed successfully. On the following day I verified that overnight updates completed successfully.

Publicity

Tim Starr announced the website through emails to lab members, a signature block hyperlink, and Starr Lab website hyperlinks.

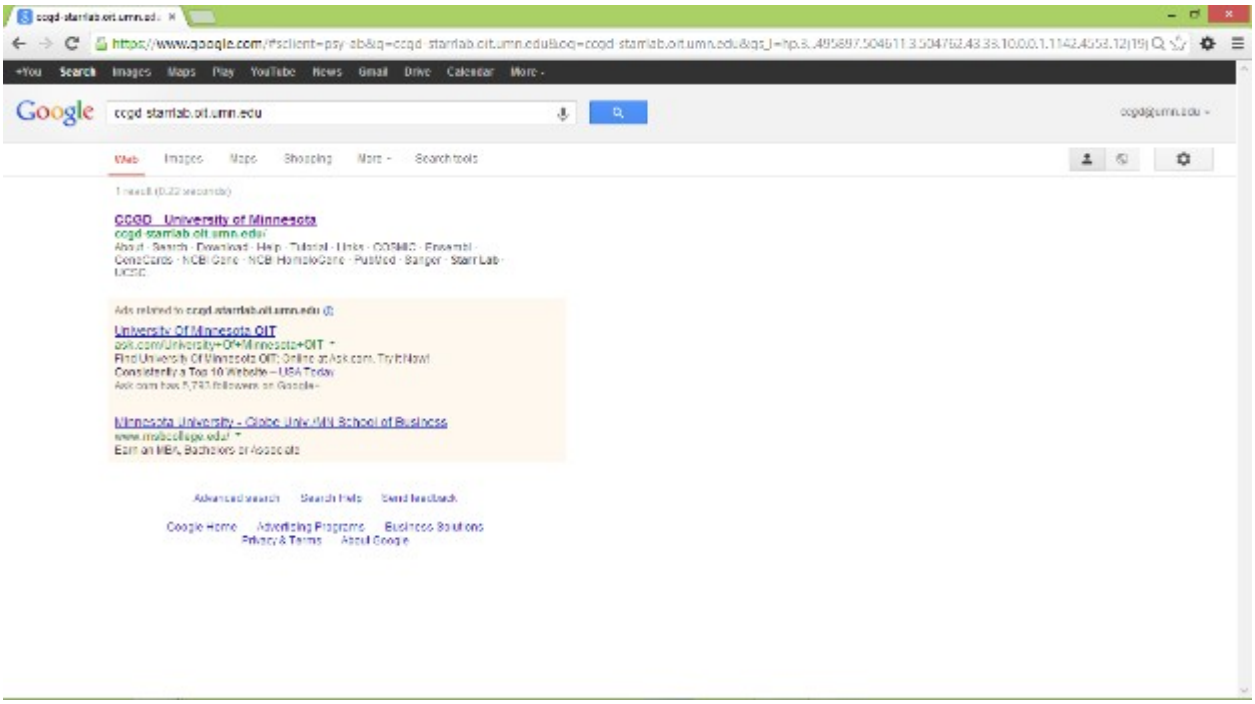
I drafted potential Wikipedia markup.

Wikipedia_code.txt



Indexing

Google detected CCGD hyperlinks after 7 days and indexed the site after 11 days. I experimented with search terms.



Publication

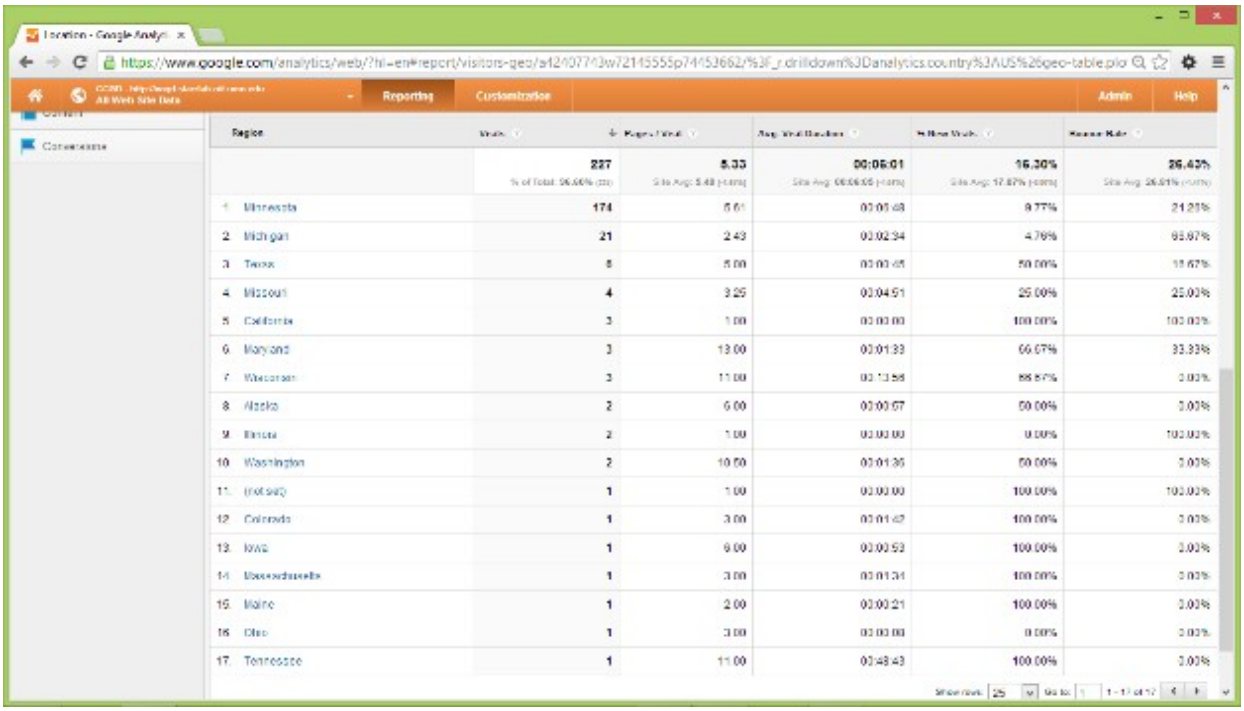
The manuscript for the website was published online on September 4, 2014.

Usage Statistics

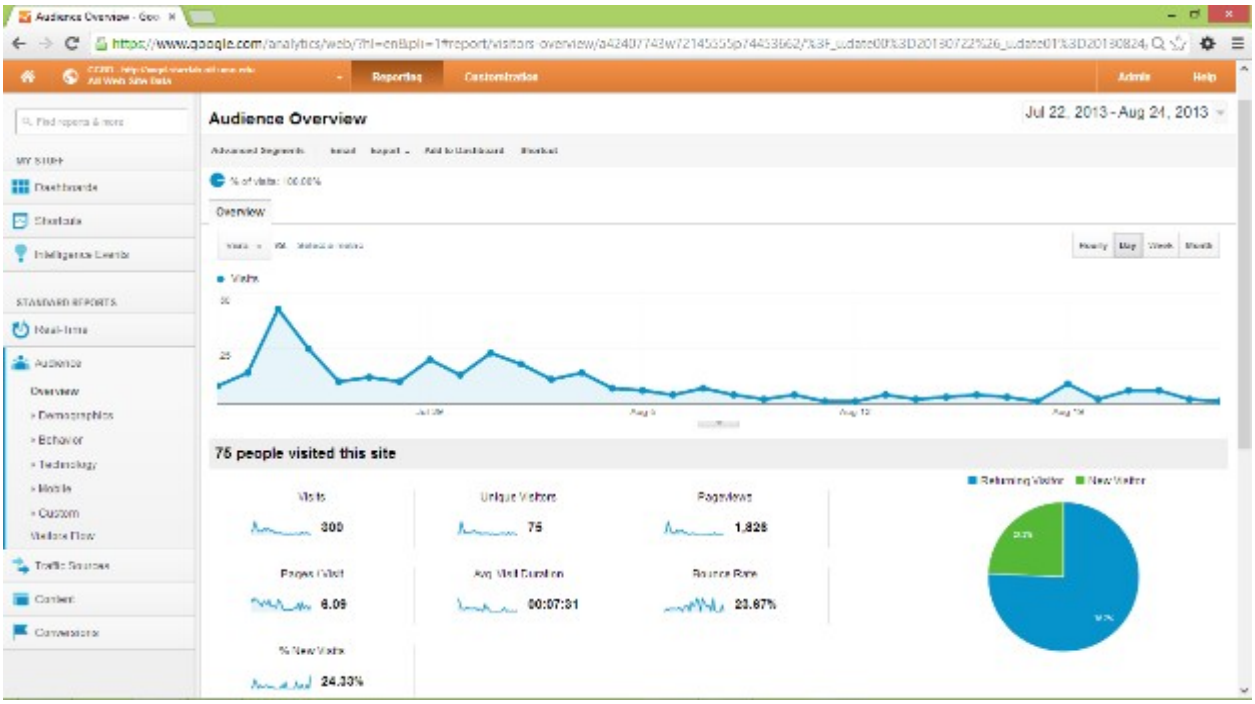
After approximately one month, Google Analytics showed a little site usage outside of Minnesota and outside of the United States.

MySQL Service Interruption

After approximately three months, the nightly update scripts began to fail. A series of `mysqladmin status` commands showed that database server uptime consistently zeroed out at 100-second intervals. The `mysql` log file showed hundreds of thousands of entries beginning on 0412 on 11/21/13, coinciding with thousands of `dracut` and `up2date` log entries describing installation of modules, keymaps, and fonts. The `/etc/init.d/mysqld restart` command failed to restart service and the `/etc/init.d/mysqld status` command showed that MySQL was dead but that the subsystem was locked. A `ps –A | grep mysql` command showed no `mysqld` process, but there was a visible lock file at `/var/lock/subsys/mysqld`, but it was only removable by users with root privileges. The `reboot` command also required root privileges. A ticket opened 11/23/13 received only moderate priority, leading to approximately two days' worth of service interruption over a weekend. OIT restarted database service and concurred that the MySQL collapse was related to automatic updates. The nightly email now includes a line reporting MySQL server uptime.



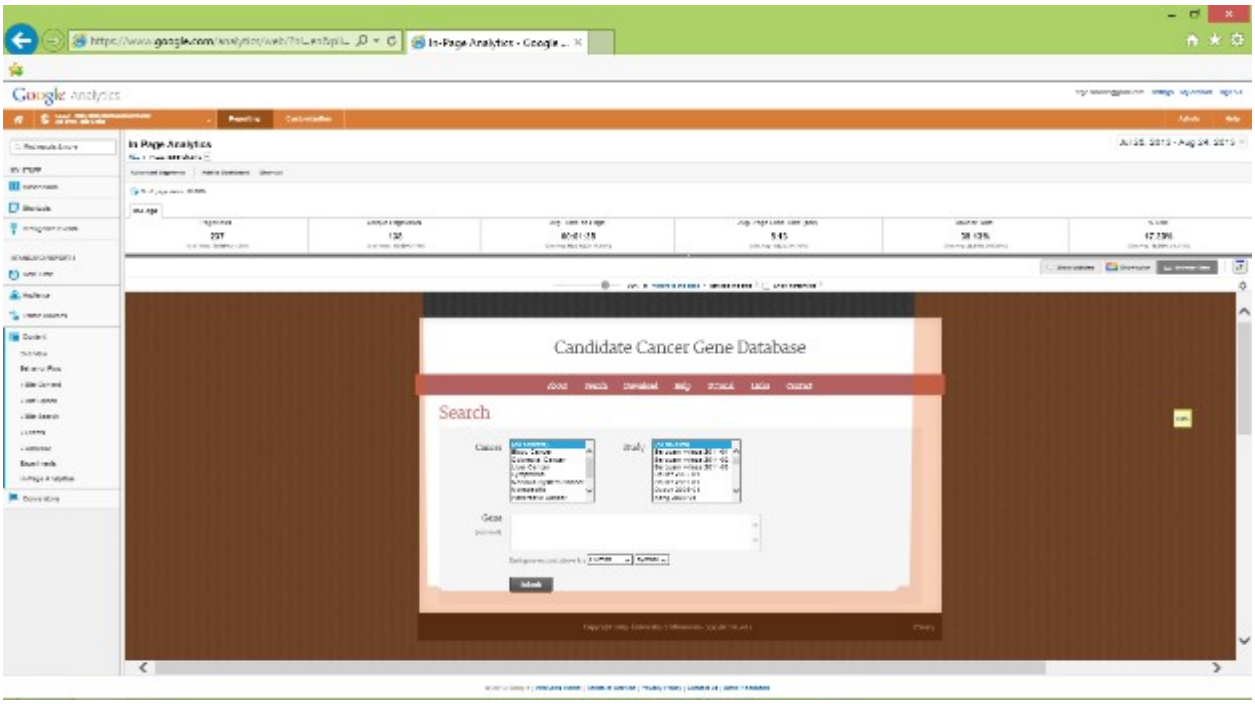
Most of the activity occurred shortly after launch.



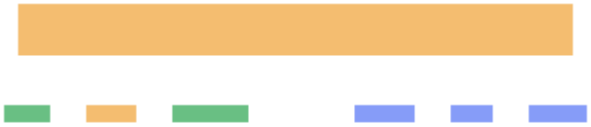
The database received the most traffic around 4 pm CDT, corresponding with website announcement.

The database logged 315 distinct queries during this time. A little over half of the queries featured something typed into the gene field, and most of these queries involved searches for human gene symbols. The average length of a typed gene query was only about 9 characters (about one gene), and this was somewhat inflated by a 471-character gene query. None of the gene query entries appeared suggestive of an attempted injection attack. There was rarely more than one selection in the cancer or study field.

Google's in-page analytics tool worked in Internet Explorer but not in Chrome. The tool suggested that most users were able to view the entire search form.



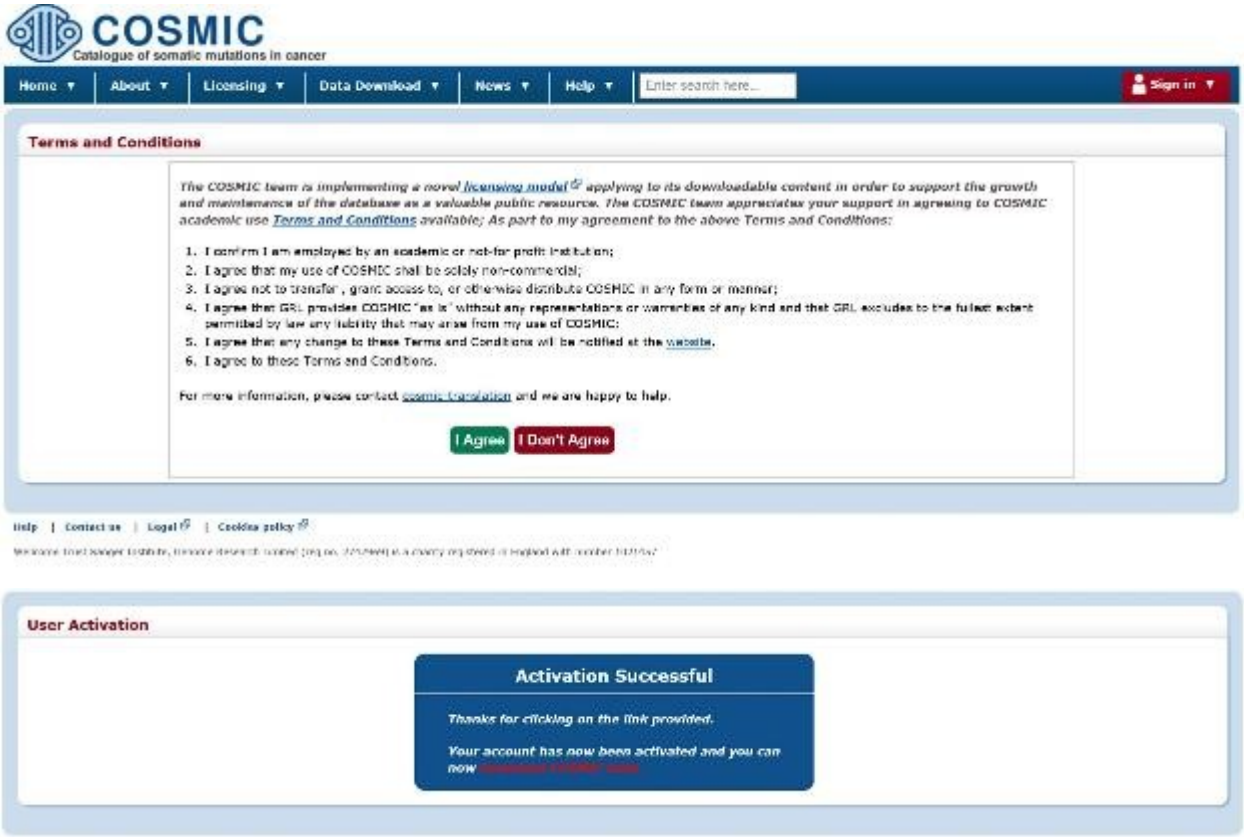
A heat map of the banner and navigation bar suggested that the banner hyperlink (to `search.php`) and the Search link (also to `search.php`) received many clicks.



Sanger

Approximately one year after launch, Sanger moved their downloadable files off of their FTP site. This required creation of a bot to complete login and download.

Approximately two years after launch, Sanger moved their downloadable files to an SFTP server. This required creation of an SFTP download script and acceptance of new license terms.



NCBI

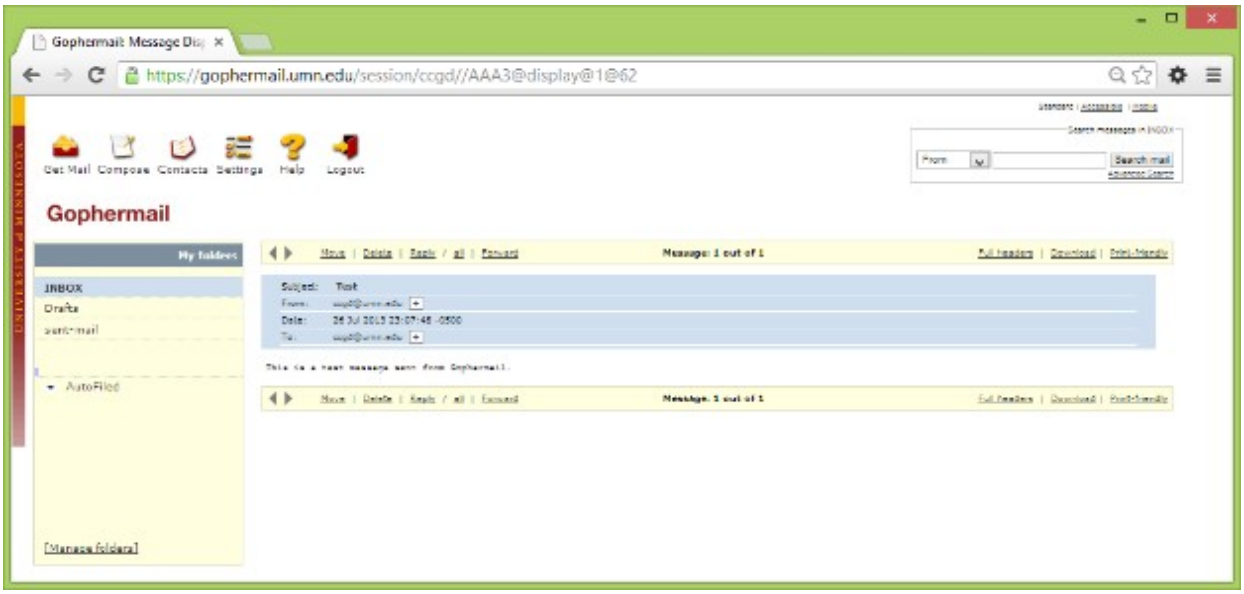
Approximately two years after launch, NCBI restructured their XML for Eutilities responses. This required changes to `ncbi.pl`, including temporary inclusion of the `version=1.0` variable followed by permanent changes to capture information from the new tags.

Maintenance

Email

Make no POP or IMAP connections to the CCGD mailbox. A small mistake in email client configuration on a Mac, Windows, iPhone, or other smartphone could cause the server to destroy email messages. The email client might pull messages from the server to the client mail program, rendering the messages unreadable on other devices. The server might destroy messages that have been filed or deleted by an email client. This may result in the loss of historical data on CCGD update processes, disk space warnings, bug reports, and intrusion attempts.

Access email through the University's Gophermail webpage client.



The forwarding feature in the Gophermail system does not allow for copies of CCGD mailbox messages to be sent to another address; it simply sends emails to the specified address instead of the CCGD mailbox. To receive copies of CCGD messages, add email addresses to `email.php`.

Automatic Updates

The `update.pl` script serves as a wrapper for a number of scripts. First are a series of downloads involving a modified version of an NCBI-authored E-utilities Perl script which complies with E-utility requirements and produces tab-separated `.txt` files. The E-utilities script is necessary because it captures inactivated genes appearing in the sporadically updated HomoloGene download but not in the frequently updated `gene_info.gz` archives on the NCBI FTP site. Next are Sanger downloads. Following this is an FTP download in `ftp.php` for records from NCBI HomoloGene and a supplementary NCBI ortholog file. Next comes the `autoload.php` script, which tests the file size of each download. If a file is not within 99.97% of its expected size, `autoload.php` terminates. The 99.97% figure should allow a little room for variation but still catch even one dropped packet of 200 genes consuming at least 1398 bytes in a file size as large as 3676975 bytes. The expected file sizes hard-coded into `autoload.php` can be changed if necessary. If all files meet minimum size requirements, `autoload.php` copies them into a backup directory and loads the files into the MySQL database. The script then tests the size of newly loaded tables; if any table is not within 99.97% of its expected row size, `autoload.php` terminates. The script then tests for empty columns in order to detect delimiter problems; if the last column of any table is completely filled with null values, `autoload.php` terminates. If the `autoload.php` script detects no errors before or after loading, the script initiates processing of downloaded data by calling the `import_process`

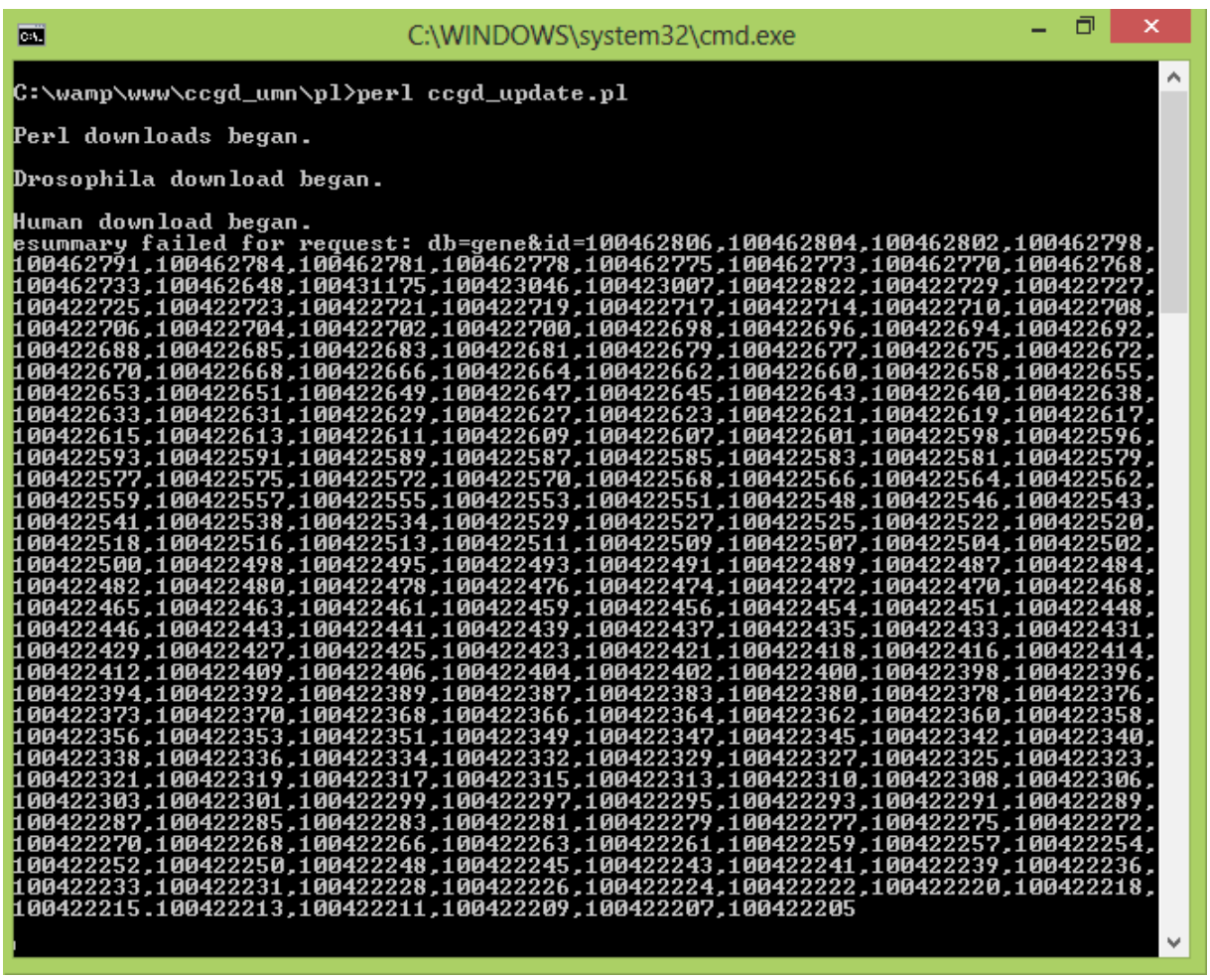
procedure in the MySQL database. Downloads require about 1.5 hours for completion; loading and processing should complete within a few minutes. Note that the tables queried by webpages will remain unchanged until the final call to `import_process`, a call that does not occur until the end of `autoload.php`.

The `ftp.php`, `autoload.php`, and `import_process` scripts log record their progress in the `ccgd_log` database table. The table below shows the normal message progression, along with typical duration.

Duration	Category	Message
0:00:00	Perl	Perl downloads began.
0:04:26	Perl	Drosophila download began.
0:32:38	Perl	Human download began.
0:30:04	Perl	Mouse download began.
0:10:55	Perl	Rat download began.
0:00:05	Perl	Yeast download began.
0:10:23	Perl	Zebrafish download began.
0:01:29	Perl	COSMIC download began.
0:00:03	Perl	CGC download began.
0:00:03	Perl	COSMIC extraction began.
0:00:02	Perl	Perl downloads ended.
0:00:00	FTP	FTP downloads began.
0:00:33	FTP	HomoloGene download began.
0:00:02	FTP	Ortholog supplement download began.
0:00:02	FTP	Ortholog supplement extraction began.
0:00:02	FTP	FTP downloads ended.
0:00:00	Load	Automatic loading began.
0:00:00	Load	Loading tests began.
0:00:00	Load	Found download drosophila.txt with size of 483853 bytes.
0:00:00	Load	Found download human.txt with size of 3863080 bytes.
0:00:00	Load	Found download mouse.txt with size of 3563984 bytes.
0:00:00	Load	Found download rat.txt with size of 1282493 bytes.
0:00:00	Load	Found download yeast.txt with size of 11758 bytes.
0:00:00	Load	Found download zebrafish.txt with size of 1326416 bytes.
0:00:00	Load	Found download homologene.txt with size of 12847548 bytes.
0:00:00	Load	Found download genegroup.txt with size of 53622167 bytes.
0:00:00	Load	Found download cgc.txt with size of 59348 bytes.
0:00:00	Load	Found download cosmic.txt with size of 216097516 bytes.
0:00:00	Load	Found download study.txt with size of 910215 bytes.
0:00:00	Load	Found download study_guide.txt with size of 8396 bytes.
0:00:00	Load	Found download field_guide.txt with size of 2748 bytes.
0:00:01	Load	Backups began.
0:00:15	Load	Loading began.
0:00:00	Load	Loaded 26320 rows into autoload_drosophila.
0:00:25	Load	Found values in the organism_chr column of autoload_drosophila.
0:00:01	Load	Loaded 190987 rows into autoload_human.
0:00:22	Load	Found values in the organism_chr column of autoload_human.
0:00:00	Load	Loaded 179154 rows into autoload_mouse.
0:00:00	Load	Found values in the organism_chr column of autoload_mouse.
0:00:00	Load	Loaded 65080 rows into autoload_rat.
0:00:21	Load	Found values in the organism_chr column of autoload_rat.
0:00:00	Load	Loaded 600 rows into autoload_yeast.
0:00:08	Load	Found values in the organism_chr column of autoload_yeast.
0:00:00	Load	Loaded 62203 rows into autoload_zebrafish.
0:00:21	Load	Found values in the organism_chr column of autoload_zebrafish.

0:00:00	Load	Loaded 259130 rows into autoload_homologene.
0:00:00	Load	Found values in the reference column of autoload_homologene.
0:02:23	Load	Loaded 1507208 rows into autoload_genegroup.
0:00:00	Load	Found values in the gene_id column of autoload_genegroup.
0:00:00	Load	Loaded 488 rows into autoload_cgc.
0:01:13	Load	Found values in the other_syndrome column of autoload_cgc.
0:00:11	Load	Loaded 912816 rows into autoload_cosmic.
0:00:02	Load	Found values in the comments column of autoload_cosmic.
0:00:00	Load	Loaded 8508 rows into autoload_study.
0:00:00	Load	Found values in the cancer_type column of autoload_study.
0:00:00	Load	Loaded 28 rows into autoload_study_guide.
0:00:00	Load	Found values in the description column of autoload_study_guide.
0:00:00	Load	Loaded 23 rows into autoload_field_guide.
0:00:00	Load	Found values in the description column of autoload_field_guide.
0:00:00	Load	Header removal began.
0:00:01	Load	Automatic loading ended.
0:04:01	Processing	The import_process procedure began.
0:00:00	Processing	The import_process procedure ended.

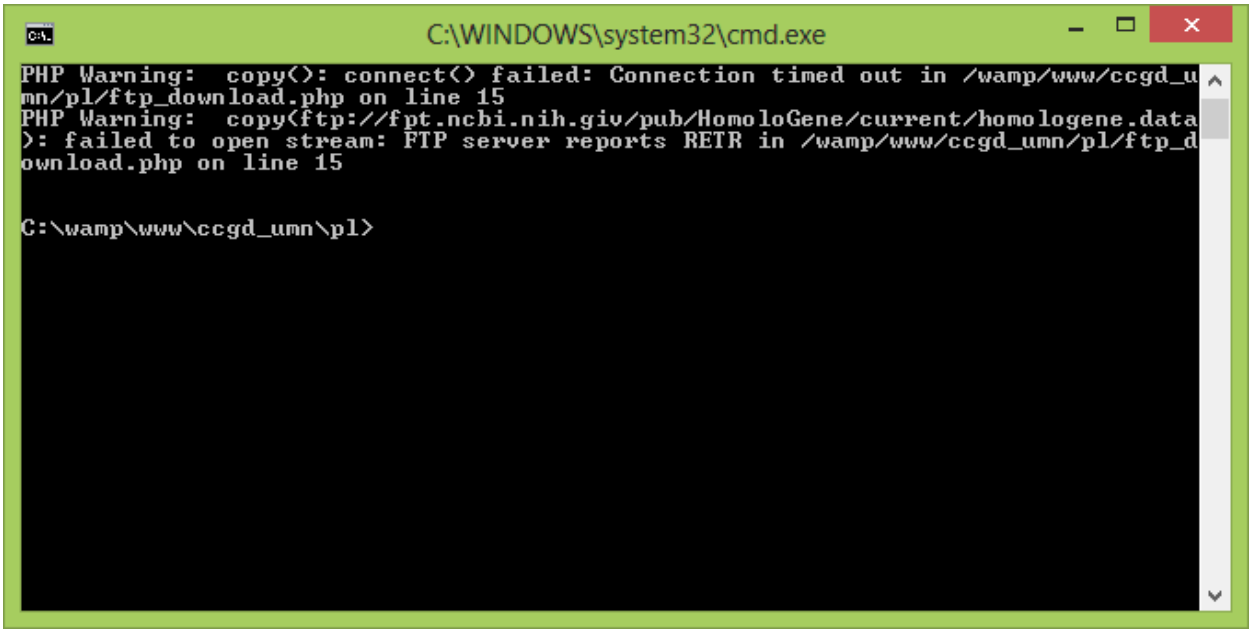
The NCBI search may occasionally fail for a limited number of gene IDs when URL variables become too numerous.



The NCBI backend may also fail.

```
C:\WINDOWS\system32\cmd.exe
C:\wamp\www\ccgd_jquery\pl>perl ccgd_update.pl

Perl downloads began.

Drosophila download began.

Human download began.
<?xml version="1.0" ?>
<!DOCTYPE eSearchResult PUBLIC "-//NLM//DTD eSearchResult, 11 May 2002//EN" "htt
p://www.ncbi.nlm.nih.gov/entrez/query/DTD/eSearch_020511.dtd">
<eSearchResult>
        <ERROR>Search Backend failed:
</ERROR>
</eSearchResult>
 did not contain expected <Count>,
 for request db=gene&retstart=97000&retmax=250&term=9606[taxid]+NOT+source_mitoc
hondrion[properties]+NOT+srcdb_refseq_model[properties]

Mouse download began.
<?xml version="1.0" ?>
<!DOCTYPE eSearchResult PUBLIC "-//NLM//DTD eSearchResult, 11 May 2002//EN" "htt
p://www.ncbi.nlm.nih.gov/entrez/query/DTD/eSearch_020511.dtd">
<eSearchResult>
        <ERROR>Search Backend failed:
</ERROR>
</eSearchResult>
 did not contain expected <Count>,
 for request db=gene&retstart=0&retmax=250&term=10090[taxid]+NOT+source_mitochon
drion[properties]+NOT+srcdb_refseq_model[properties]

Yeast download began.
<?xml version="1.0" ?>
<!DOCTYPE eSearchResult PUBLIC "-//NLM//DTD eSearchResult, 11 May 2002//EN" "htt
p://www.ncbi.nlm.nih.gov/entrez/query/DTD/eSearch_020511.dtd">
<eSearchResult>
        <ERROR>Search Backend failed:
</ERROR>
</eSearchResult>
 did not contain expected <Count>,
 for request db=gene&retstart=0&retmax=250&term=4932[taxid]+NOT+source_mitochond
rion[properties]+NOT+srcdb_refseq_model[properties]

Zebrafish download began.
<?xml version="1.0" ?>
<!DOCTYPE eSearchResult PUBLIC "-//NLM//DTD eSearchResult, 11 May 2002//EN" "htt
p://www.ncbi.nlm.nih.gov/entrez/query/DTD/eSearch_020511.dtd">
<eSearchResult>
        <ERROR>Search Backend failed:
</ERROR>
</eSearchResult>
 did not contain expected <Count>,
 for request db=gene&retstart=0&retmax=250&term=7955[taxid]+NOT+source_mitochond
rion[properties]+NOT+srcdb_refseq_model[properties]
```

FTP downloads may fail (especially for HomoloGene).



The `ftp.php` script tries to complete FTP downloads up to 20 times. Similarly, the `sanger.pl` script tries to complete its own downloads up to 20 times, choosing to execute either the `sanger.exp` or dynamically generated `sanger.ftp` SFTP script. The `retry.php` script may try to rerun `update.pl` up to 2 times in order to secure a complete set of Perl and FTP downloads. The `check.php` script emails a progress report to the address specified in `email.php`. The `check` email includes a report of free disk space in the MySQL directory since that directory could expand with the addition of records to `ccgd_study`, `ccgd_log`, and `ccgd_query`. The `check` email also recommends pursuit of additional disk space when free disk space does not exceed 2 GB. Both `retry.php` and `check.php` make decisions based upon the last entry in `ccgd_log`.

A script named `sites.php` verifies that all of the links entered into `ccgd_links` produce readable files. Any website changes involving external hyperlinks should be reflected in the `ccgd_links` table. The `ccgd_links` table contains some examples of dynamically generated hyperlinks that wouldn't be tested by a static link checker. It's possible to script more in-depth testing of content obtained with dynamically generated URLs, but this approach is not very tolerant of design changes on target sites. Scripting cannot guarantee that information returned by a target site is useful; for example, a "No results" page may contain all of the markup needed to pass validation tests. There is ultimately no substitute for human use and inbox monitoring. If `sites.php` detects any errors, it generates a notification email to the address specified in `email.php`.

If a user initiates processing through `upload.php`, the messages below may appear in `ccgd_log` before the Load entries generated by `autoload.php`.

- Uploaded file.
- Uploads completed successfully.

Example error messages include:

- Error: zebrafish.txt has an expected size of 1326416 bytes and an observed size of 11758 byte(s).
- Error: autoload_drosophila has an expected size of 26551 rows and an observed size of 1 row(s). The new line character(s) may be incorrect.
- Error: autoload_cgc has a completely null other_syndrome column. The .txt file may not be tab-delimited.

These errors appear in messages sent to the email address in `email.php`.

Manual Updates

Study Information

PubMed Notification

Check the CCGD email inbox for PubMed messages about new hits for the query “sleeping beauty transposon mouse cancer.”

Update the Study Spreadsheet

Download a copy of the current spreadsheet from the NetFiles share. Note that the share may not be accessible while logged into Gophermail or while using any other centralized Shibboleth authentication.

Limit data in fields to alphanumeric characters when possible and try to avoid commas, spaces, or quotation marks. Unfortunately, some gene names will contain non-alphanumeric characters (e.g., `beta'Cop`, `CIS10:36034542-36035273`, `Epb4.1`, `Gt(ROSA)26Sor`, `NM_134232, O<hv>`, `and su(w[a])`, `Del(7)Tyr<c-25H>`). The last example happens to be one of about 20 mouse symbols that are not going to be searchable in the database because they contain encoded `< >` symbols that are treated as malicious HTML. Do not change the data type for any cells to anything other than Text, because Excel might begin making assumptions about information that it thinks could represent dates, times, or currency.

Keep in mind the following when updating `study` worksheet data:

- Add records to the end of the table instead of inserting rows between existing records.
- All columns should be formatted as "text" except for the first column, `mouse_id`, which is formatted as "general". If you are pasting data from another worksheet, use Paste Special and Values to avoid changing any column formatting.
- To remove a study, delete the rows containing the study data. If you will be replacing with new data, add the data at the end.
- The `mouse_id` column should have entries for all gene symbols, except CISs labelled CISX:XXXX-XXXX. If you are missing a large number of gene IDs, try using the MGI Batch Conversion Tool.
- Ensure that any single study only contains a list of unique `mouse_symbol` entries. Duplicate entries compromise database calculations. Use Find and Sort functions in Excel to verify this.
- Use the following format for the `study` column: first author’s name (without hyphens), a space, four-digit year, and 2-digit study number (including a leading zero if needed). This cannot exceed 30 characters. Example: `Starr 2011-01`
- Ensure that the `predicted_effect` column includes only `Loss`, `Not Determined`, or `Gain`.
- Ensure that the `relative_rank` column includes only `A`, `B`, `C`, `D`, or `Not Ranked` entries. Records in this field having more than 13 characters will be truncated.
- Avoid abbreviations for the `cancer_type` column and ensure that no entries consist of a string that is wholly included in another `cancer_type`; for example, a cancer type of `Leukemia` is included within the tumor type of `Acute T Lymphoblastic Leukemia`. This cannot exceed 50 characters. Use consistent capitalization for words such as `Cancer`.
- Verify you have not lost any records by comparing the number of records you add to the spreadsheet to the number of records appearing in untouched study data.

Keep in mind the following when updating `study_guide` worksheet data:

- The `pubmed_id` column must not have blank entries and must have a match for each entry on the `study` worksheet.
- The `added` column should contain only dates in MM/DD/YYYY format. It will appear on webpages in YYYY-MM-DD format.

- The description column should not contain any double quotation marks, superscript, or subscript; it should be plain text.

Keep in mind the following updating `study_guide` worksheet data:

- The description column should not contain any double quotation marks, superscript, or subscript; it should be plain text.

Do not add worksheets to the Excel workbook. The workbook should only contain `study`, `study_guide`, and `field_guide` worksheets. Other worksheets might pile up as `.txt` files in the `pl/` directory. The upload page should automatically archive your spreadsheet in the NetFiles share.

Update CIS Addresses (rare)

Rarely, it will be necessary to update CIS addresses in the study spreadsheet. Use the LiftOver tool at http://genome.ucsc.edu/cgi-bin/hgLiftOver.

Save a backup copy of the old study spreadsheet. The process for updating CIS addresses is prone to error.

Create a helper column in the study spreadsheet which will add the letters `chr` to the chromosome number.

	CIS Start	CIS End
="chr"&C2	32770143	32808349
	32757497	32826160
	32757497	32826160

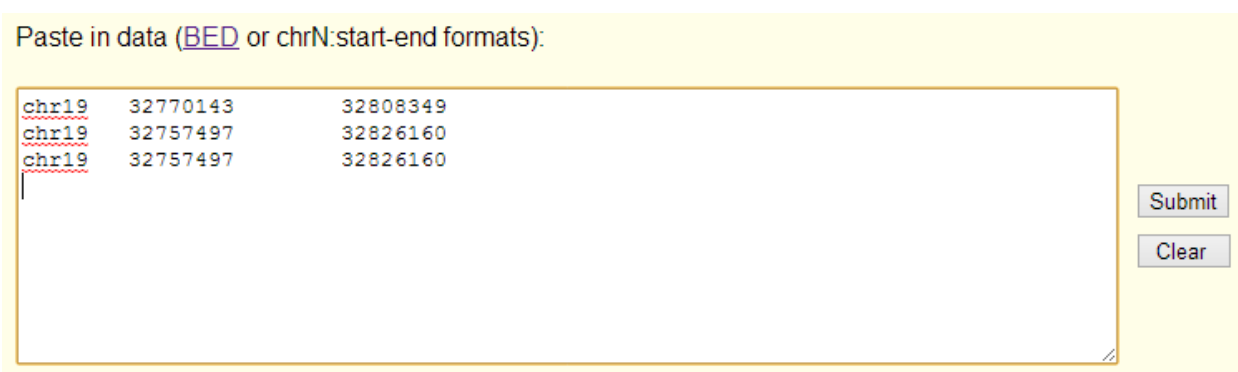
Fill the entire column with your formula.

chr19	32770143	32808349
chr19	32757497	32826160
chr19	32757497	32826160

Select all of the chromosome, `CIS Start`, and `CIS End` values but do not include the column headers in your selection. Copy these to the clipboard.

	CIS Start	CIS End
chr19	32770143	32808349
chr19	32757497	32826160
chr19	32757497	32826160

Navigate to http://genome.ucsc.edu/cgi-bin/hgLiftOver. Paste the copied values into the conversion field. The values should appear separated by tab characters.



Click on the Submit button and then find the link to view results.

Results

Successfully converted 3 records: View Conversions

Click on the View Conversions link. Your conversions should appear in a new browser window.

```
chr19    32695653        32733859
chr19    32683007        32751670
chr19    32683007        32751670
```

Unfortunately, the conversions do not appear alongside original values. Be careful about copying and pasting these new values back into the spreadsheet. The tool will not skip rows that it fails to convert. If there is a failure to convert a row, you will see an extra link to a failure file.

Results

Successfully converted 2 records: View Conversions

Conversion failed on 1 record. Display failure file Explain failure messages

Failed input regions:

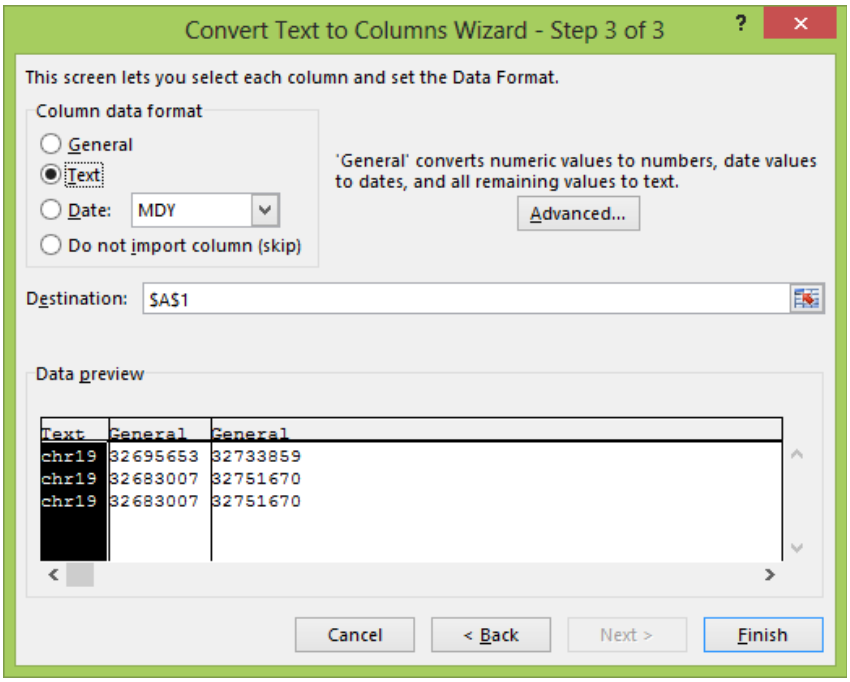
Select all data on the webpage.

```
chr19    32695653        32733859
chr19    32683007        32751670
chr19    32683007        32751670
```

Paste the data into a spreadsheet.

chr19	32	95653	32733859
chr19	32	83007	32751670
chr19	32	83007	32751670

Use the Text to Columns utility to separate the merged columns. You can specify the delimiter of Space. Always choose Text data types instead of General data types so that Excel does not modify the appearance of values that it thinks may represent dates or numbers.



Paste the columns of new values into empty columns on the study spreadsheet.

			CIS Start	CIS End
chr19	32695653	32733859	32770143	32808349
chr19	32683007	32751670	32757497	32826160
chr19	32683007	32751670	32757497	32826160

You may want to create additional helper columns to check the distance between old and new CIS addresses. Very large differences in these numbers might suggest an error.

			CIS Start	CIS End
=W2-U2		32733859	32770143	32808349
		32751670	32757497	32826160
		32751670	32757497	32826160

When you have verified the integrity of all CIS addresses, copy and paste the new values into the CIS columns on the study spreadsheet. Delete any helper columns you may have created. The spreadsheet should be ready for upload. You may need to update the genome build information as described in the next section.

Update Genome Build Information (rare)

Update the genome build information in `about.php` (e.g., "Human GRCh37/hg19 and Mouse GRCm38/mm10").

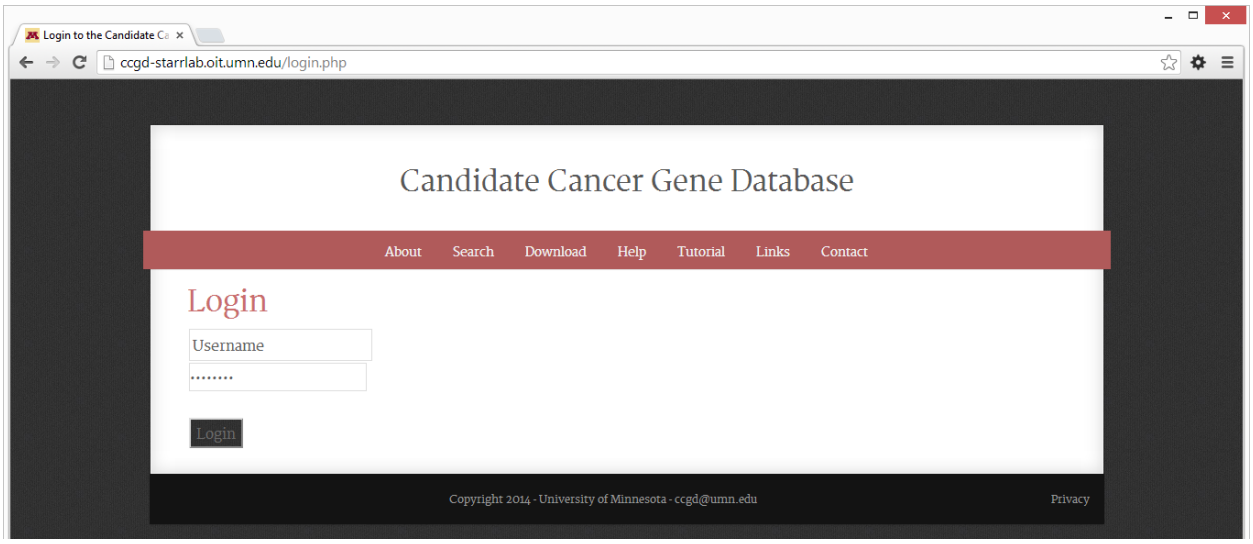
Load Option 1: Webpage

Login

Login to the website.

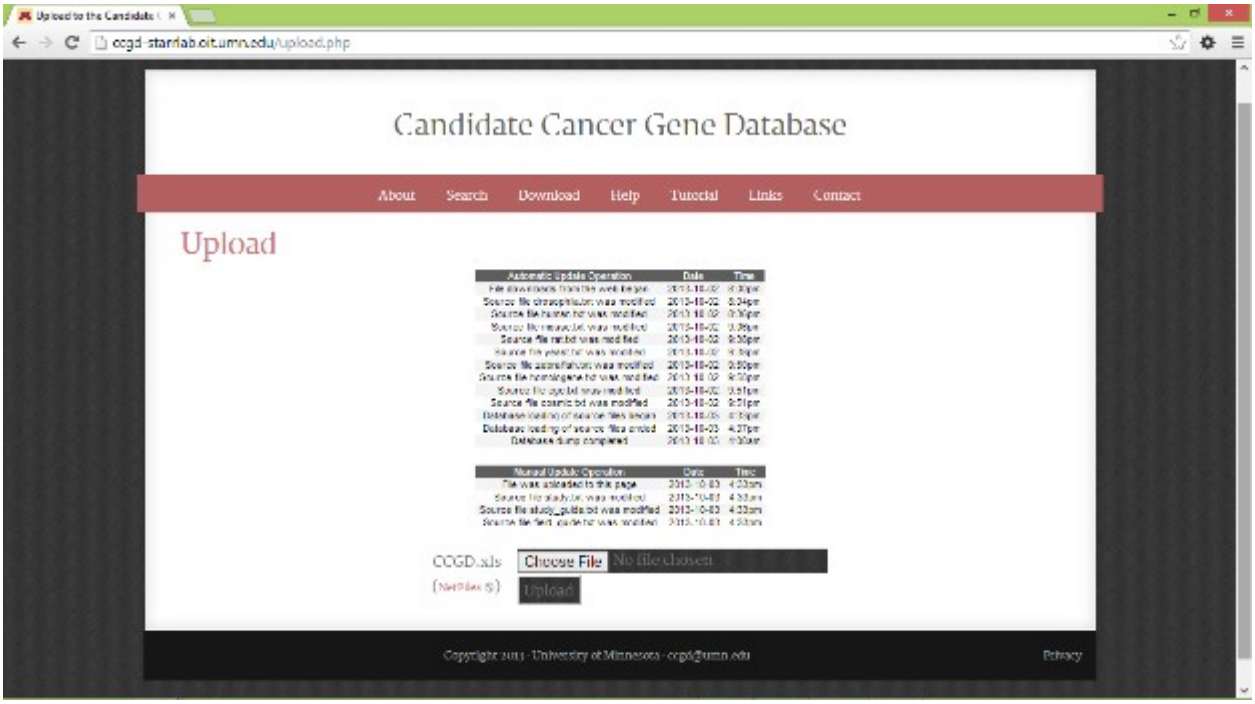
Username - ccgd

Password – Candidate!Genes



Upload

Upload the CCGD spreadsheet. The spreadsheet should remain in Excel 2003 `.xls` format (not Excel 2007's `.xlsx` XML format) so that it can appropriately be parsed by the Perl module that creates tab-delimited text files. Do not click the Upload button more than one time.



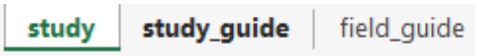
Verify Processing with Testing

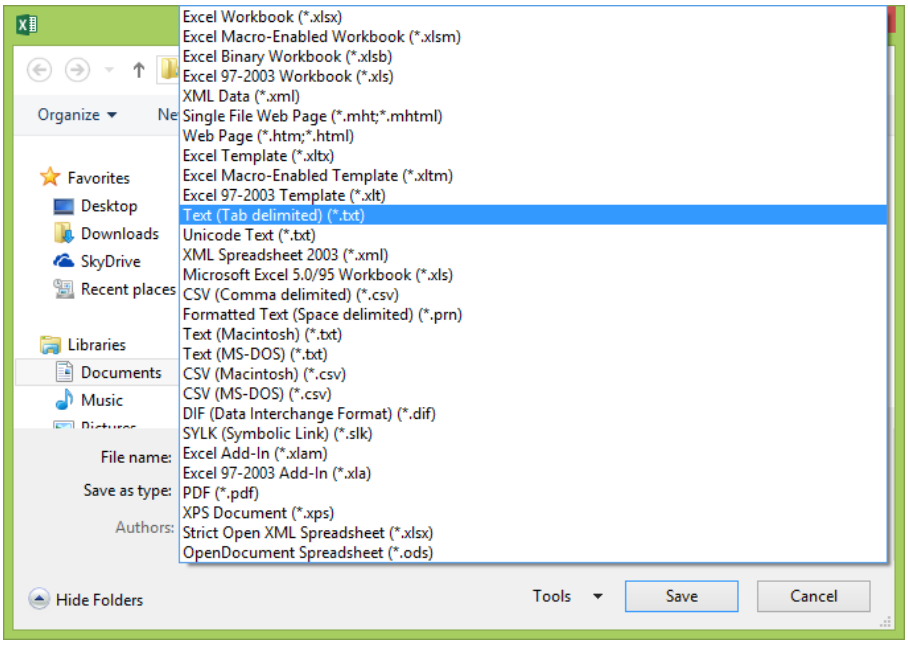
Perform some test searches to verify successful processing. If pages appear blank due to a PHP error, temporarily turn off error suppression by commenting out the command that appears in `error.php`. This file is included on the first line of website pages, meaning that error reporting can be toggled for the entire website with a change to only one file. When you have eliminated errors, un-comment the command in `error.php`.

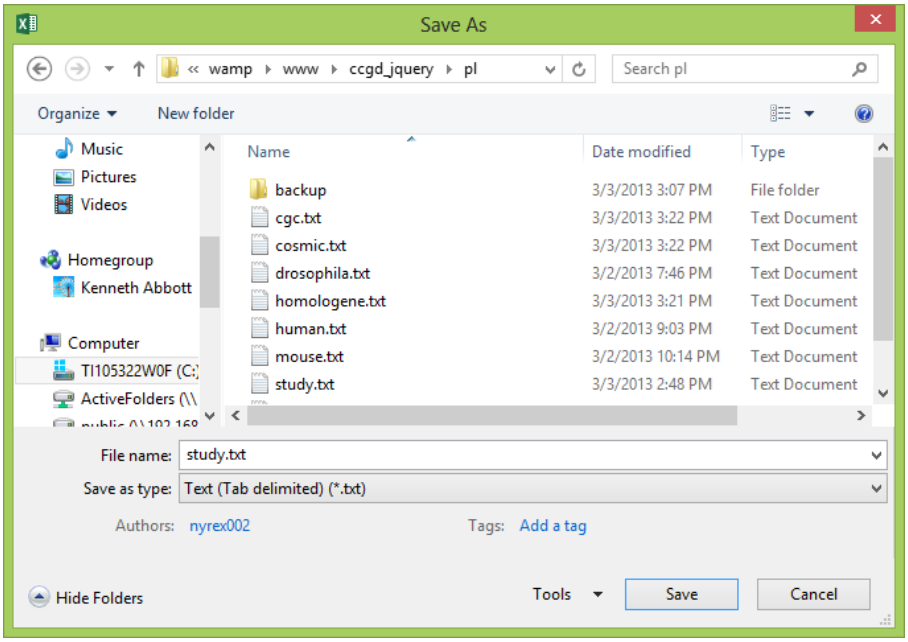
Load Option 2: Web Server Connection

Convert to `.txt`

Save each worksheet of the Excel spreadsheet in tab-delimited `.txt` format; note that this is one of multiple `.txt` formats. You will only be able to save one worksheet at a time, so save the `study` spreadsheet as `study.txt`, the `study_guide` spreadsheet as `study_guide.txt`, and the `field_guide` spreadsheet as `field_guide.txt`.

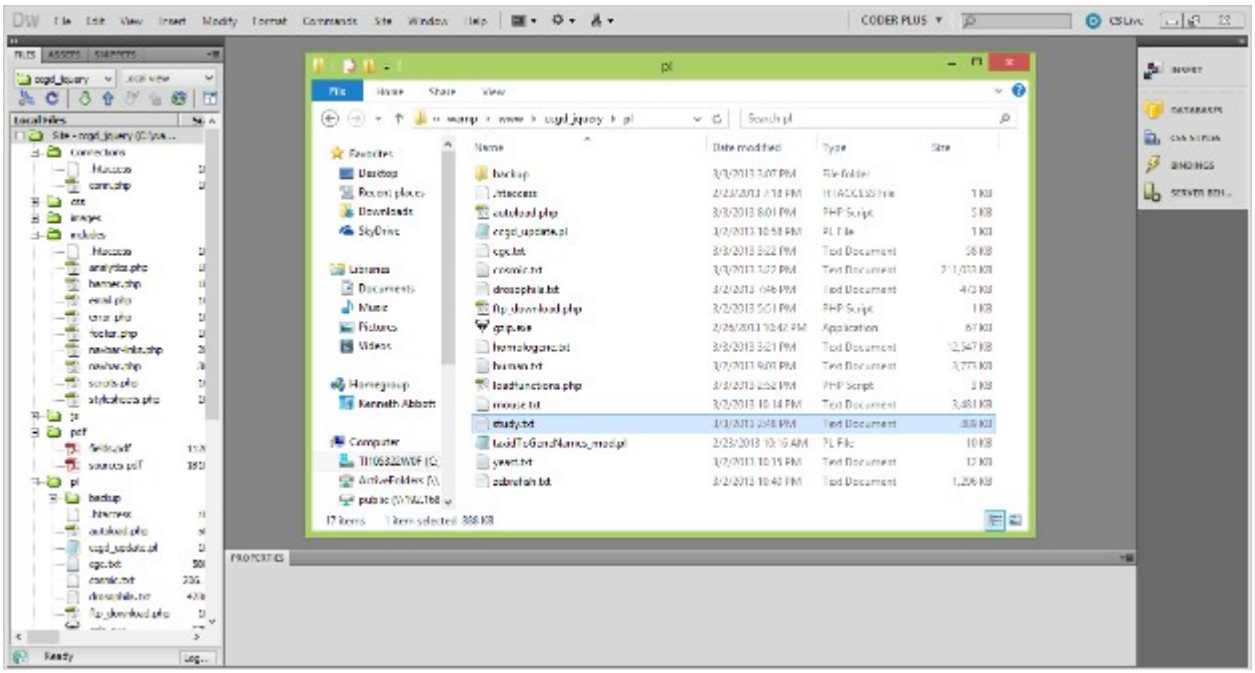






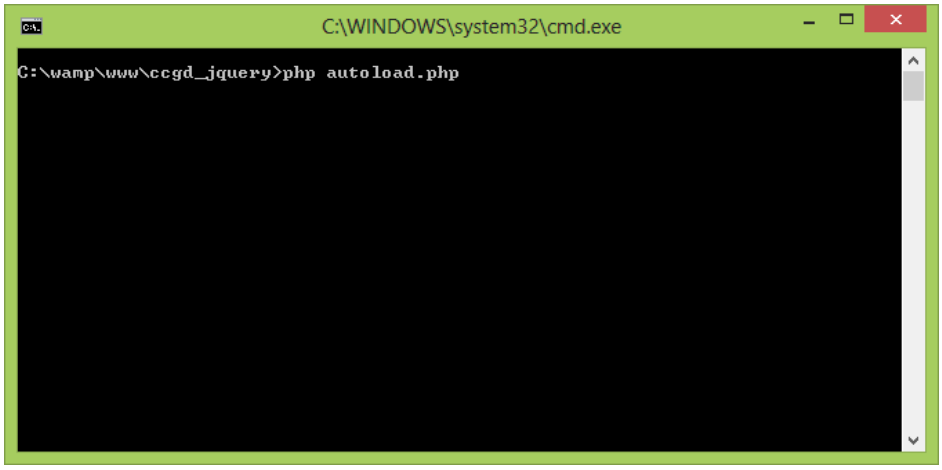
Send to Server

Send this file to the server to replace the currently published version of `pl/study.txt`.



Execute pl/autoload.php

Execute `pl/autoload.php`; this script tests file size, makes a backup, loads records, tests table sizes, and initiates processing.



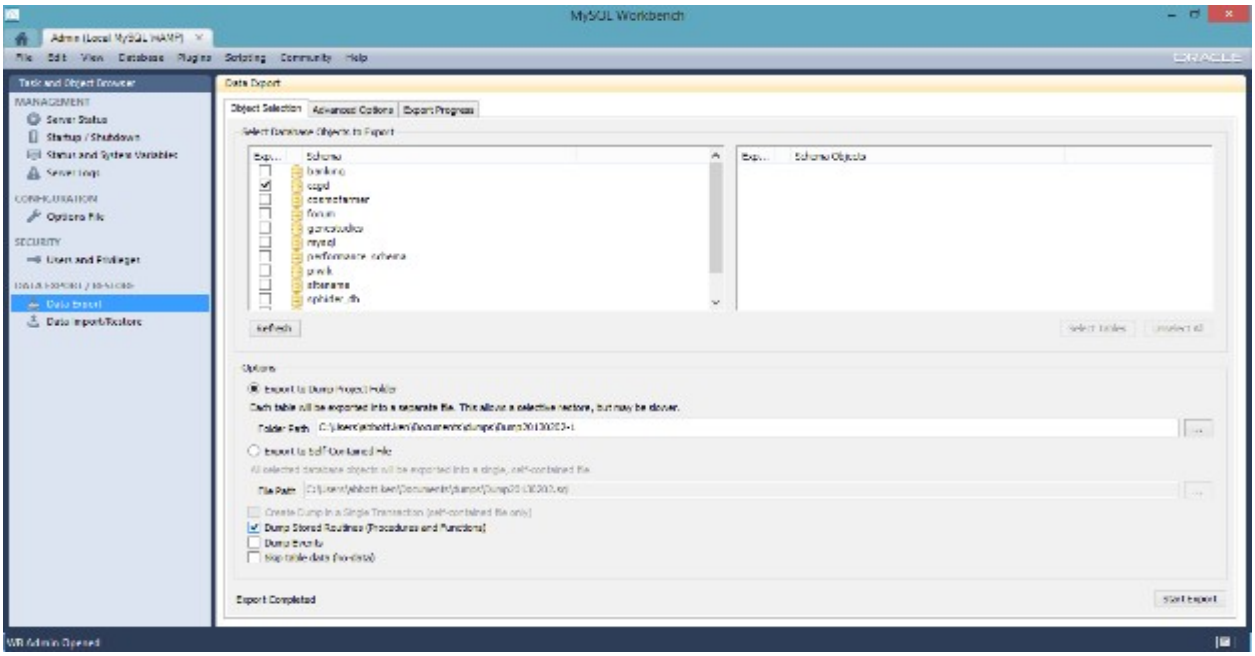
Verify Processing with Testing

Perform some test searches to verify successful processing. If pages appear blank due to a PHP error, temporarily turn off error suppression by commenting out the command that appears in `error.php`. This file is included on the first line of website pages, meaning that error reporting can be toggled for the entire website with a change to only one file. When you have eliminated errors, un-comment the command in `error.php`.

Load Option 3: MySQL for Excel Connection

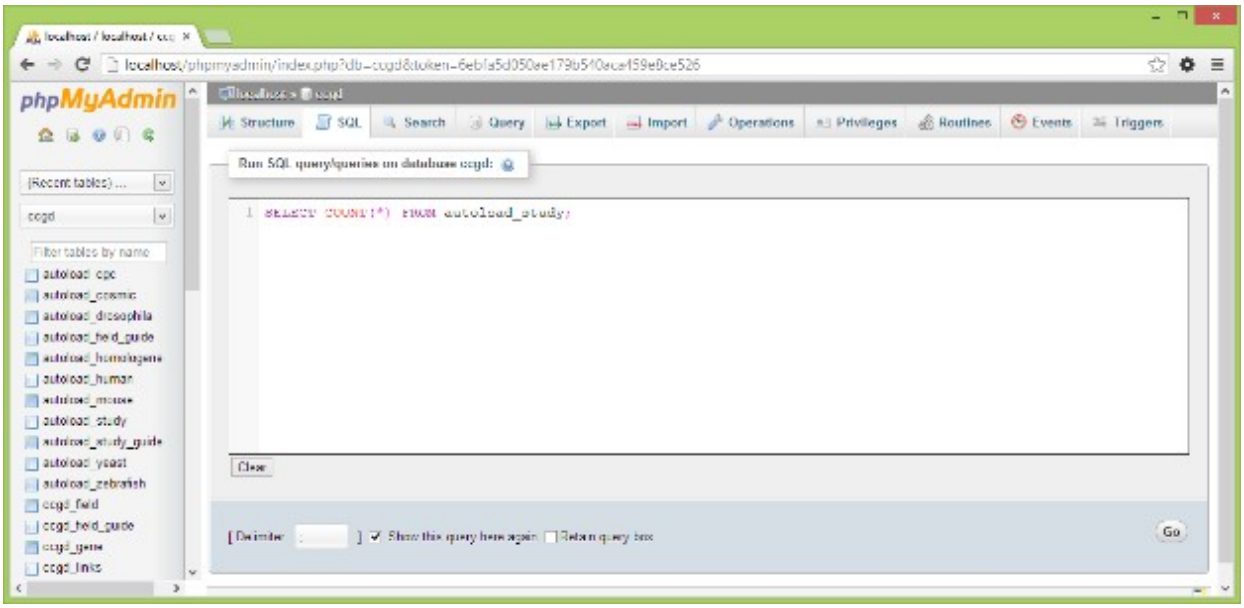
Backup the Database

Ensure that you have access to recent backup copies of database tables. MySQL Workbench contains an interface for running `mysqldump`, and this will quickly generate `.sql` files of database tables. Make sure you select the option to include exports of stored procedures.



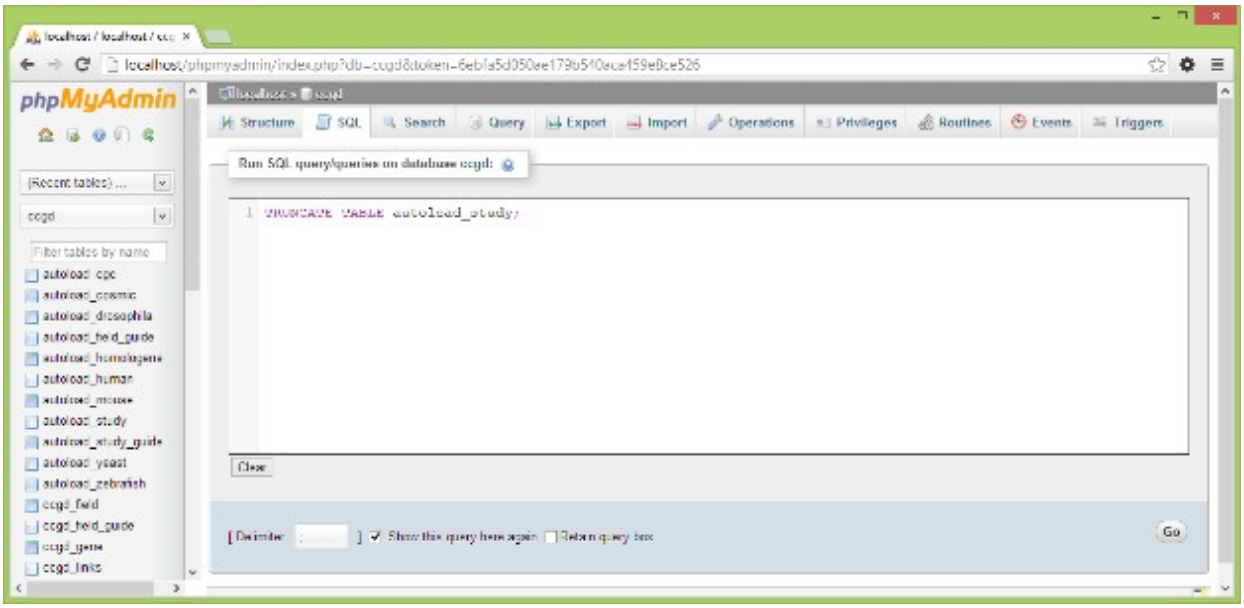
Obtain Baseline Table Size

Use a utility such as phpMyAdmin or MySQL Workbench to connect to the MySQL database. Select the `ccgd` database through mouse clicks or a `USE ccgd;` command. Obtain a record count by using a `SELECT` statement (e.g., `SELECT COUNT(*) FROM autoload_study;`).



Empty MySQL Table

Use a utility such as phpMyAdmin or MySQL Workbench to connect to the MySQL database. Select the `ccgd` database through mouse clicks or a `USE ccgd;` command. Empty an individual table using the `TRUNCATE` command (e.g., `TRUNCATE TABLE autoload_study;`).

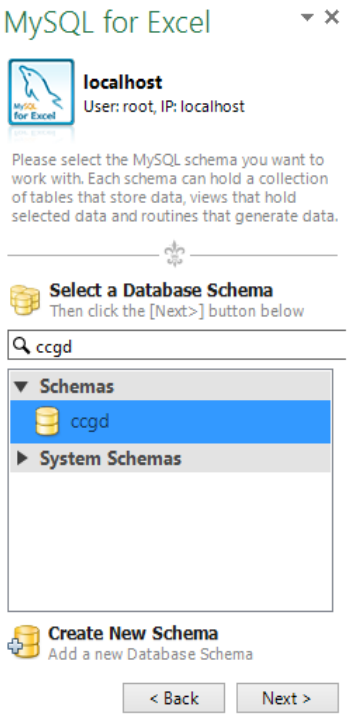


Upload Spreadsheets

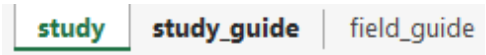
The MySQL for Excel plugin (Windows) easily maps Excel columns to MySQL table fields. This program is available from http://dev.mysql.com/downloads/installer/ as part of the MySQL installation package.

To update a MySQL table, find the MySQL for Excel button under the Data tab and pick a database connection.

Continue by picking a database.



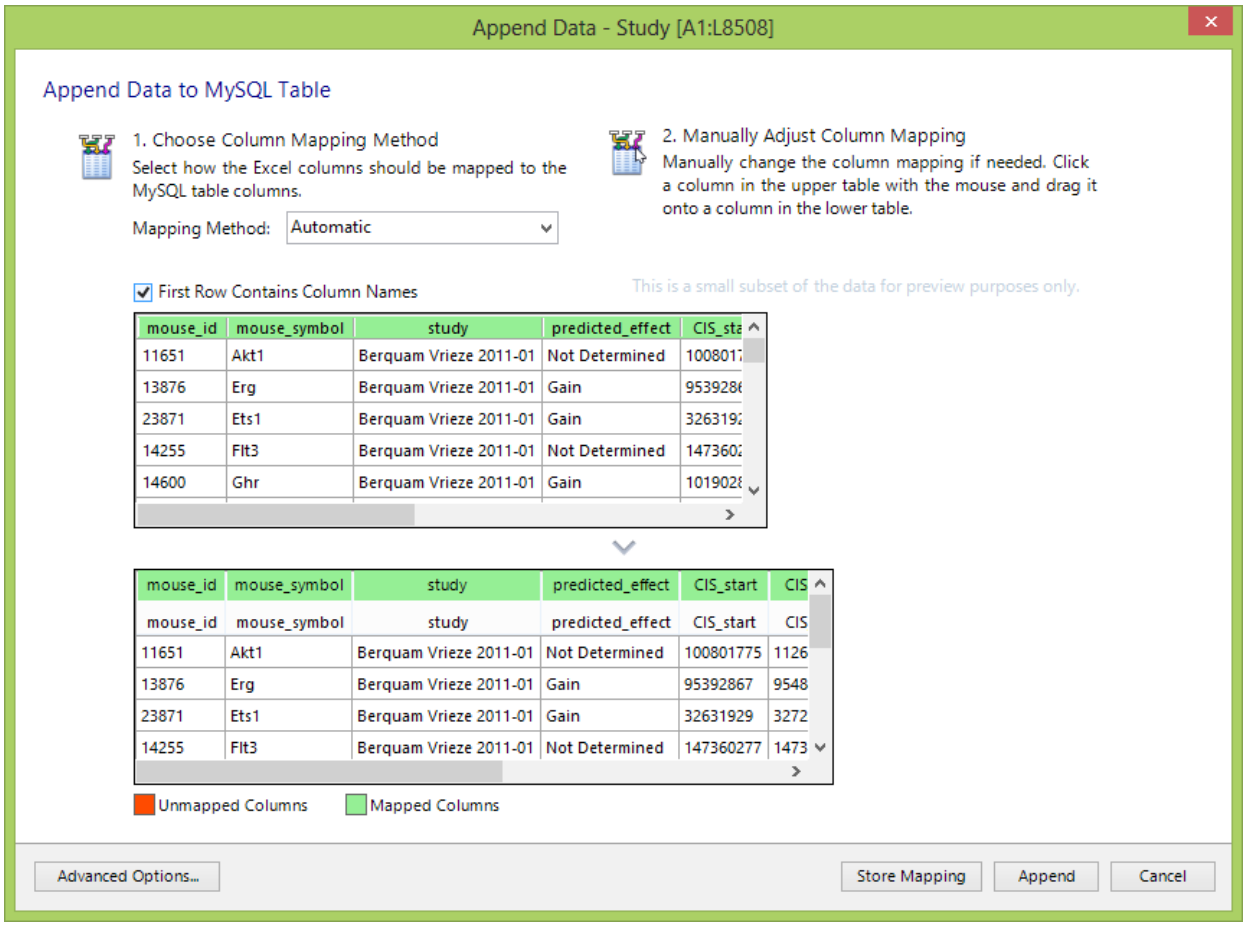
Note the name of the worksheet that you are working with. The `study` worksheet will be loaded into `autoload_study`. The `study_guide` worksheet will be loaded into `autoload_study_guide`. The `field_guide` worksheet will be loaded into `autoload_field_guide`.



Ensure that you have the whole table selected in the worksheet. Click on the Append Excel Data to Table link.

Click on the First Row Contains Column Names box; this should help map spreadsheet column names to database field names. If a field does not map properly, you can map it manually by clicking and dragging it from the top

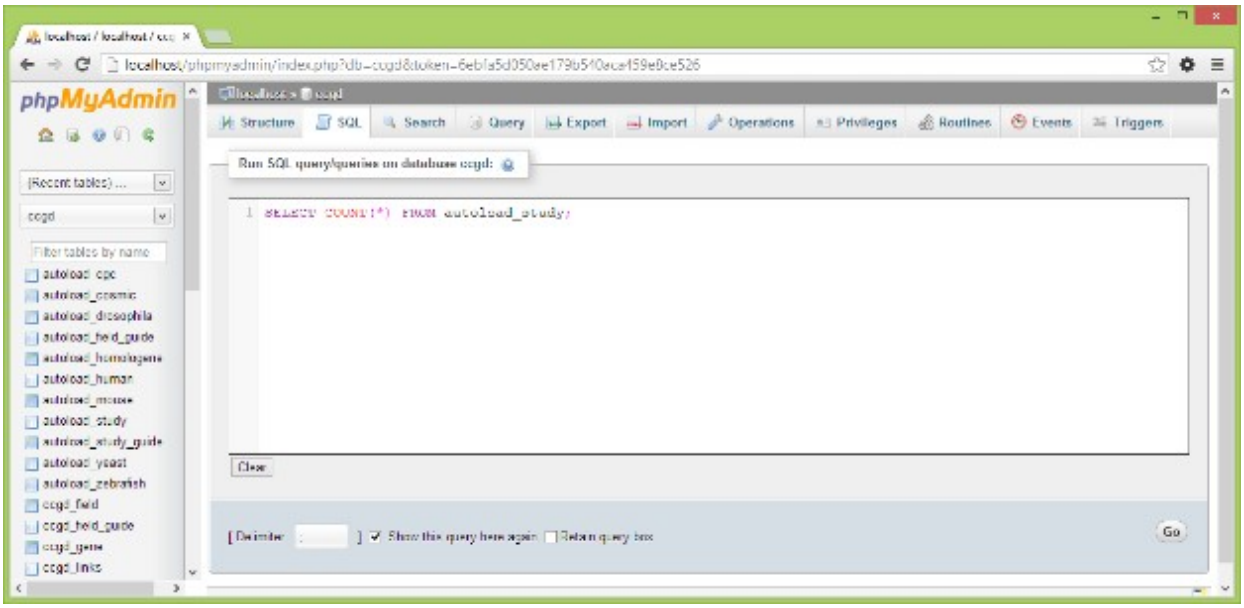
window onto a field in the bottom window. Note that the first column in the database table may be a key column which has no counterpart in the spreadsheet and should not be mapped to any spreadsheet column.



Click on the Append link to complete the append operation. Watch for any warnings that the utility provides, such as truncated fields.

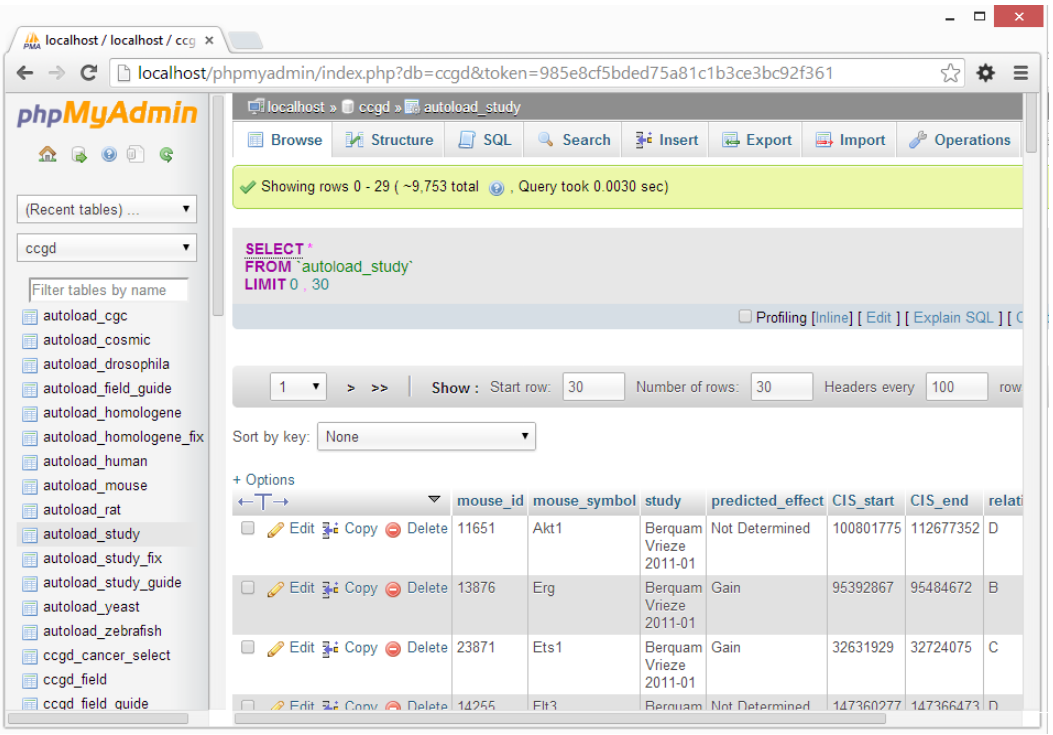
Verify New Table Size

Use a utility such as phpMyAdmin or MySQL Workbench to connect to the MySQL database. Select the `ccgd` database through mouse clicks or a `USE ccgd;` command. Obtain a record count by using a `SELECT` statement (e.g., `SELECT COUNT(*) FROM autoload_study;`).



Verify Absence of Null Columns

Use a utility such as phpMyAdmin or MySQL Workbench to connect to the MySQL database. Browse through the table and ensure that none of the columns has NULL values for every record.



Process Imported Data

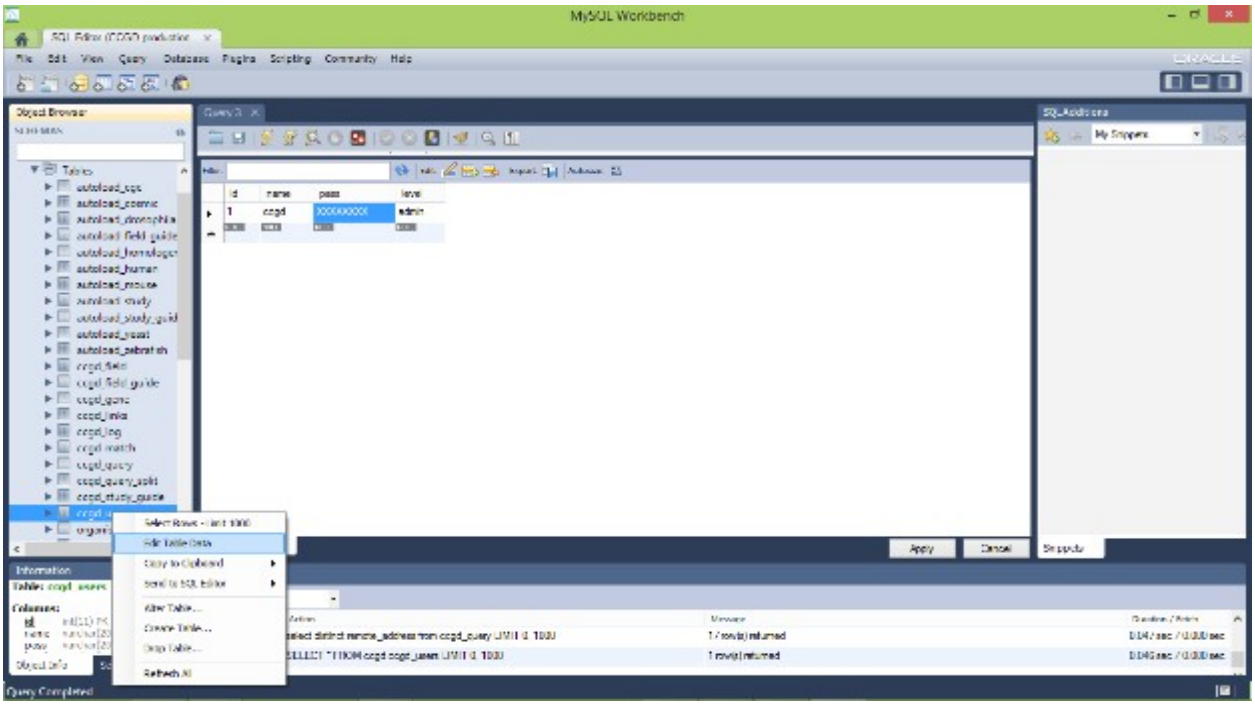
Use a utility such as phpMyAdmin again to regenerate `ccgd_gene`, `ccgd_field`, and `ccgd_study`. The command to use is `CALL import_process`. Note that MySQL Workbench may not work for this when in safe mode because the `import_process` routine includes some `UPDATE` statements that do not include a key.

Verify Processing with Testing

Perform some test searches to verify successful processing. If pages appear blank due to a PHP error, temporarily turn off error suppression by commenting out the command that appears in `error.php`. This file is included on the first line of website pages, meaning that error reporting can be toggled for the entire website with a change to only one file. When you have eliminated errors, un-comment the command in `error.php`.

Change Website Password

The CCGD password is stored in the `ccgd_users` table in the MySQL database. To change this password, you could use MySQL Workbench or the MySQL shell. Either option would require a successful SSH connection to http://ccgd-starrlab.oit.umn.edu followed by a successful login to the MySQL server as `root`.

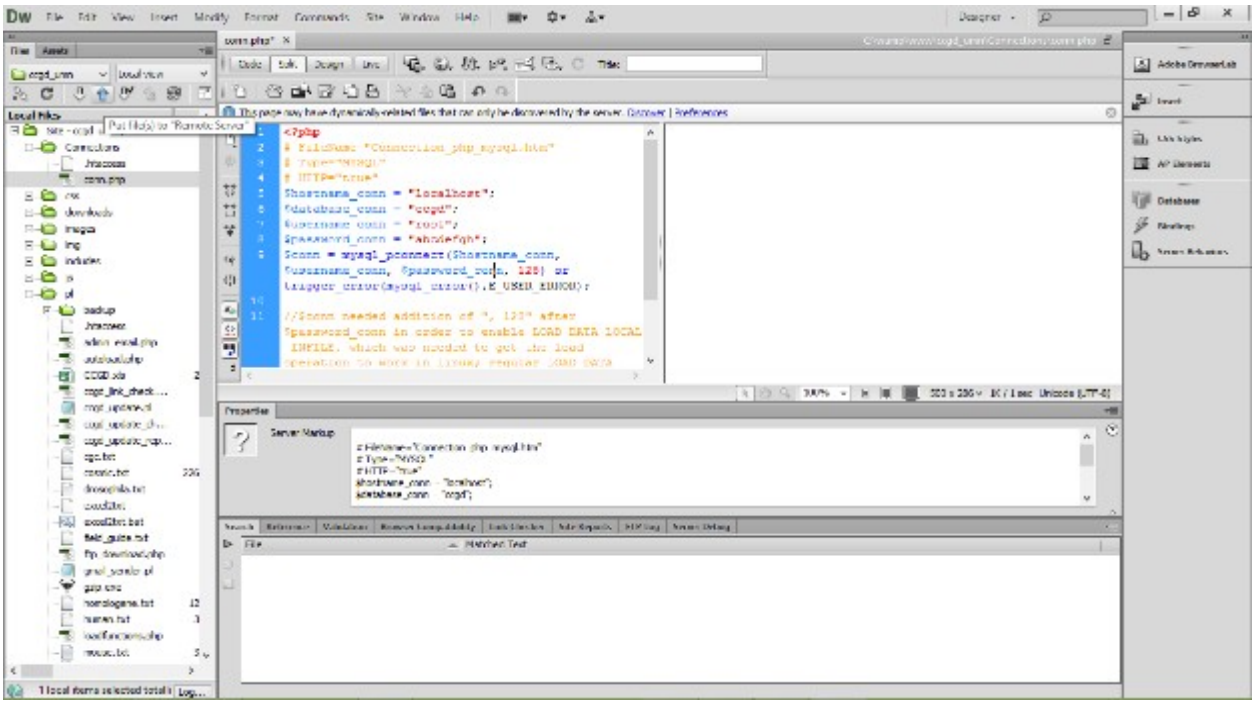


```
abbot195@ccgd-starrlab:~
[abbot195@ccgd-starrlab ~]$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 257
Server version: 5.1.69-log Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> USE ccgd;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> UPDATE ccgd_users SET pass='XXXXXXXX' WHERE id=1;
```

Change Database Password

It should be rare, but it's possible to change the MySQL database password. This password must be strong because of the high risk for damage in the event of unauthorized access. Any changes made to the MySQL password must be reflected in `Connections/conn.php` and `pl/update.pl`; the website will break if these passwords do not match.

```
abbot195@ccgd-starrlab:~
[abbot195@ccgd-starrlab ~]$ mysqladmin -u root -p'oldpass' password newpass
```


conn.php*
<?php
$hostname_conn = "localhost";
$database_conn = "cogd";
$username_conn = "root";
$password_conn = "ahrdefgh";
$conn = mysql_pconnect($hostname_conn,
$username_conn, $password_conn, 128) or
trigger_error(mysql_error(),E_USER_ERROR);
//$conn needed addition of ", 128" after
$password_conn in order to enable LOAD DATA LOCAL
INFILE, which was needed to get the load
operation to work in Linux; regular LOAD DATA
Server Markup
$hostname_conn = "localhost";
$database_conn = "cogd";