APPENDIX A – SERVER CONFIGURATION

Server Configuration

MorphBank is intended to run on a two-server system. For security and reliability, the database should be running on a dedicated machine behind the organization's firewall and should only communicate to the web server that should be in the organization's DMZ. Please note that all hardware is minimum recommended systems.

Database Server

Hardware:

- Dual AMD Athlon 2400+ MP 2.0GHz Processors
- 1GB PC2100 Registered ECC DDR RAM
- 80GB 7200RPM Hard Drive
- 1024GB External hot-swappable RAID Device
- 100Mb Ethernet card

Software:

- Fedora Core 2
- MySQL (>3.23.54, not tested on earlier versions).

Web Server

Hardware:

- Dual AMD Athlon 2400+ MP 2.0GHz Processors
- 1GB PC2100 Registered ECC DDR RAM
- 250GB 7200RPM Hard Drive

- 100/1000Mb Ethernet card

Software:

- Fedora Core 2
- PHP (>4.3.0)
- Apache (1.3 or 2.0)
- Imagemagick (Latest version recommended).
- Ghostscript (GNU Ghostscript >7.06).
- Tomcat 5

Mirror Site

Mirror sites tend to attract less traffic and are not mission critical. Still, due to the nature of the application, they require significant processor capacity and disk space.

Hardware:

- AMD Athlon 2400+ MP 2.0 GHz Processor
- 1GB PC2100 Registered ECC DDR RAM
- 250GB 7200RPM Hard Disc Space
- 100Mb Ethernet Card

Software:

- Fedora Core 2
- Apache (1.3 or 2.0)
- MySQL (>3.23.54, not tested on earlier versions).
- PHP (>4.3.0)
- Imagemagick (Latest version recommended).

- Ghostscript (GNU Ghostscript >7.06).
- Tomcat 5

Note that this is a recommended minimum configuration. The database size is expected to exceed recommended disc space rapidly and provisions should be made to upgrade this feature regularly. Furthermore, although unsupported, other operating system and web server configurations are acceptable assuming they provide MySQL > 3.23.54 and PHP >4.3.0 support.

Install Required Components

It is assumed that the system administrator is able to install and configure the required software to build a web server. The following links and directions are provided as a summary checklist.

Apache

Download the latest version of Apache from http://www.apache.org, choose between version 1.3 or 2.0.

- httpd-2.0.50.tar.gz (Unix)
- httpd-2.0.50-win32-src.zip (Windows)

Install this on the computer that you intend to use for serving MorphBank. There is no need to alter any of the default settings with this.

A Unix step-by-step can be found here:

- http://httpd.apache.org/docs-2.0/install.html

A Windows step-by-step can be found here:

- http://httpd.apache.org/docs-2.0/platform/windows.html

Tomcat

Download the latest version of Tomcat from:

- http://jakarta.apache.org/site/binindex.cgi
- jakarta-tomcat-5.0.27.tar.gz (Unix)
- jakarta-tomcat-5.0.27.exe (Windows)

Install this on the computer you intend to use for serving MorphBank. A step-by-step guide for both Unix and Windows can be found here:

- http://jakarta.apache.org/tomcat/tomcat-5.0-doc/setup.html

PHP

Download the latest version of PHP from:

- http://www.php.net/downloads.php

Install this on the computer that you intend to use for serving MorphBank. A Unix step-by-step can be found here:

- http://www.php.net/manual/en/install.unix.php

A Windows step-by-step can be found here:

- http://www.php.net/manual/en/install.windows.php

MySQL

Download the latest version of MySQL from:

- http://dev.mysql.com/downloads/index.html

Install this on the computer that you intend to use for serving MorphBank. A Unix and Windows step-by-step can be found here:

- http://dev.mysql.com/doc/

We recommend you change the root password IMMEDIATELY. This is a potential source of attack for the website.

Imagemagick

Download the latest version of Imagemagik from:

- http://www.imagemagick.org/www/download.html?

Install this on the computer that you intend to use for serving MorphBank.

- Follow the readme file.

We recommend that you add extra mage types before commencing. Each image format supported will require additional mage types to enable conversions.

Ghostscript

Download the latest version of Ghostscript from:

- http://sourceforge.net/project/showfiles.php?group_id=1897

Install this on the computer that you intend to use for serving MorphBank.

- Follow the readme file.
- Install this and the provided fonts.

All the software should now be installed and ready to use.

Setting up the database

MySQL

Once installed the daemon has a default user with all privileges allowed. You MUST change this and add a password before commencing. Without doing this your database and other databases you put on the MySQL server could be open to attack from malicious users. The process for doing this is outlined above.

Once this has been done you can set up the tables using the ‘morphbank.sql’ file provided.

- Start the MySQL monitor from the command prompt (Assuming you have set the password you will have to type mysql –u root –p).
- Copy the file ‘morphbank.sql’ to the mysql/bin directory.
- Enter the following at the mysql prompt:
  - source morphbank.sql
- This should set up the tables as required.
- If you are setting up a new database go to step 6 else carry out the following
  - Copy the file with the sql data in it (probably dump.sql) to the mysql/bin/ directory
- Enter the following at the mysql prompt:
  - source dump.sql
- Next the ‘user.sql’ file must be altered to the desired details of you the administrator. Open the file and change the settings for password, username, e-mail, forename, surname, institution and address. These are the values between the square brackets (You should also remove the square brackets).
- Copy the file to mysql/bin/.
- Enter the following at the mysql prompt:
  - source user.sql

- This should have successfully setup the database with the administrator.

PHP Scripts

Copy the contents of the directory www to the web_root (or to the intended directory for MorphBank).

Open the file include/variables.inc and change the details within it (Full instructions are given in the file).

Open the file id/.htaccess and change the first line (again instructions are given)

Finished

Your database should now be setup and ready to use.

Hardware is always a balance between price, performance and planned obsolescence. In business, core systems are typically expected to last two or three years before being replaced. In academia, a less stable, grant based funding model encouraging a nominally more expensive machine that has an expected service life closer to five years. Although MorphBank does not currently have sufficient traffic to justify this hardware, the projected database growth over the next two semesters and combined with an expectation that the biological community will embrace the service, we recommend the following server configuration.

Web Server:	$1,800
Database Server:	$1,700
RAID Device:	$4,900
RAID Spares:	$220
Mirror Server:	$1,600

APPENDIX B – GRAPHICAL ANNOTATION TECHNOLOGY

Purpose of Annotation Technology

The Annotation Technology group was assigned the task of creating the software to enable annotations of images on the MorphBank project in compliance with the software requirements and specifications.

Our group was told to first take Shayne Steele's existing code and add other functions so that it meets the specifications stated.

The Annotation Technology software's requirements include: being able to load images of type jpeg, gif, tif, bitmaps, and so on. In addition, the software must be able to capture the time and date the annotation when the annotation was posted, the author or contributor of the annotation, and the type of image being annotated (i.e. images, graphs, data, etc.). Furthermore, the software must be able to capture the x-coordinate and the y-coordinate of the pixel on which the mouse is pointed to, and store the annotation text on that location, thus allowing multiple annotations on every image. Importantly, the software must not alter the images loaded onto the site. It should be able to retain the original image and avoid any sort of distortion after the annotation is submitted.

Research for Annotation Technology

The general purpose of this research is to track other projects with similar goals and requirements as the MorphBank project. In a more particular note, this research will eventually aid the graphical annotation technology layout of the said project. We are to compare other annotation technology projects with that of Shayne Steele's current

Annotation Technology software, in terms of their annotation types, image annotation, data annotation, morphological annotation, and their use of searching and processing annotations. In addition, we are to focus on projects that have been successful with the new annotation technology or that are continuing to develop such innovation. Through these projects, we are hoping to extract a wide range of information that will pave the way to the development of the new graphical annotation technology for the MorphBank project.

Inote Annotation Technology

Description

"Inote is an image annotation tool written in Java, which allows the user to attach textual annotations to various regions in an image and then store those annotations and details in a separate text file." (http://www.iath.virginia.edu/inote) Inote is used as a stand-alone application.

It will read the image, and then automatically generate certain kinds of details. One can create several types of details, such as rectangle, polygon, circle, or a point and attach more than one annotation to these details. The details can then be organized into overlays, which is unlimited. These details can be moved from one overlay to another; they may also be resized, and copied from one to another. The annotations are then saved as an XML-format data.

By running Inote as an applet, Java-capable web browsers can then view the annotated images with the details and annotations that were created with them. To view the annotations, the user must click the mouse on the detail's region. If that region has

multiple annotations, a list will appear with a scroll-bar to its side which users may select from.

Distribution

Inote is distributed as Inote 6.0 version by the Board of Visitors of the University of Virginia, and is downloadable, free of charge for educational purposes at (http://www.iath.virginia.edu/inote/).

Requirements

To be able to run Inote 6.0 version, there are two Inote downloadable packets from their website—a Windows95 application with installer and with the Java Runtime Environment built in, and a tar file containing only the Inote class files. The Windows95 application package is recommended for download for Windows95 users, while other platform users are recommended to download the tar file.

Menu Options

The following menu options were copied from the Inote website (http://www.iath.virginia.edu/inote/help.html).

File Menu

Open Local Image — this option allows the user to load an image from your local file system: file selection is done with a standard dialogue box. Inote can only load JPEG and GIF images.

Open Image via HTTP — this option allows the user to open a JPEG or GIF image from any Web server, by giving the fully qualified URL of that image (including the server name, domains, path, and filename).

Remove New Overlay — this option allows the user to remove a new overlay from the image.

Load Local Overlays — this option allows the user to load overlays from a local file system through a standard dialog box.

Load Overlays via HTTP — this option allows the user to open an existing Inote overlay from any Web server, by giving the fully qualified URL of that overlay (including the server name, domains, path, and filename).

Save Local Overlays — this option allows the user to save current overlays to a local file system.

Save Overlays Via HTTP — this option is not functional at present.

Save Overlays Locally — this option saves overlays to the local file system.

Close Image — this option will discard Inote's currently loaded image and allow you to load another image into Inote from the local filesystem or via HTTP.

Exit — this option will exit the Inote program and close all Inote windows.

Frame Menu

Screen Size — this option resizes the Inote frame (but not the loaded image) to the total viewable area of your monitor.

Half Screen (Landscape) — this option will size the Inote frame to half the total viewable area of your monitor, in landscape orientation.

Half Screen (Portrait) — this option will size the Inote frame to half the total viewable area of your monitor, in portrait orientation.

Quarter Screen — this option will size the Inote frame to one quarter of the total viewable area of your monitor.

Fit to Image — this option will size the Inote frame to fit the loaded image.

Image Menu

Double Size — choosing this menu option will magnify the loaded image to twice its current size. See also plus/minus buttons, below.

Half Size — choosing this option will reduce the loaded image to half its current size. See also plus/minus buttons, below.

Fit to Frame — this option will fit the image inside the Inote frame; if the frame is a different shape than the image, you may see some gray space in the Inote frame around the image.

Actual-size Imaging — this option will return the image to its original size.

Mode Menu

Browse — this option allows the user to click on existing details and view associated annotations.

Edit — this option presents the user with a series of buttons controlling the editing, addition and removal of overlays, details and annotations

Detail — all mouse selections control choice of details as well as moving, resizing and deleting details. A mouse-click upon a detail produces a dialog with the single detail or a list of overlapping details upon which the user can select items to rename, delete or copy the annotations to a new overlay of auto-generate

new details. Holding the mouse button down while over a detail allows the user to move that detail within the image.

Annotation — mouse clicks select, create and edit annotations. A mouse click selects a detail and produces a dialog box displaying that detail or overlapping details and allows the user to select an annotation from those details. The user may then move the annotations to new detail, delete the annotation, edit it, copy it to a new detail, cancel the operation, or browse.

Rectangle Drawing Tool — to use this tool the user selects a corner and drags the mouse to create a rectangle by enclosing the desired region.

Polygon Drawing Tool — a series of mouse clicks create a polygon by enclosing a space.

Circle Drawing Tool — to create a circle the user selects a center point with a mouse click and drags the mouse to determine the circle's radius. Release of the mouse button produces a circle enclosing the desired region.

Point Drawing Tool — a single mouse click creates a new point in the top overlay.

Overlay — this option presents a dialog box. The user's first option is "new," which produces a new overlay. The "delete" option removes a selected overlay. The "copy" option reproduces the contents of a selected layer to another overlay. The "move" option relocates a selected overlay to the top layer.

Select Overlay — selection of this option allows the user to determine the uppermost overlay in use.

Overlay Color — selection of this option allows the user to change the color used to accent details in the overlay in use.

Overlay Visibility — this option allows the user to display and hide overlays.

Help Menu

Contents — choosing this option will display the URL for this document.

About Inote — choosing this option will bring up a pop-up window with information about Inote's programmers, download and documentation, copyright, licensing, and liability.

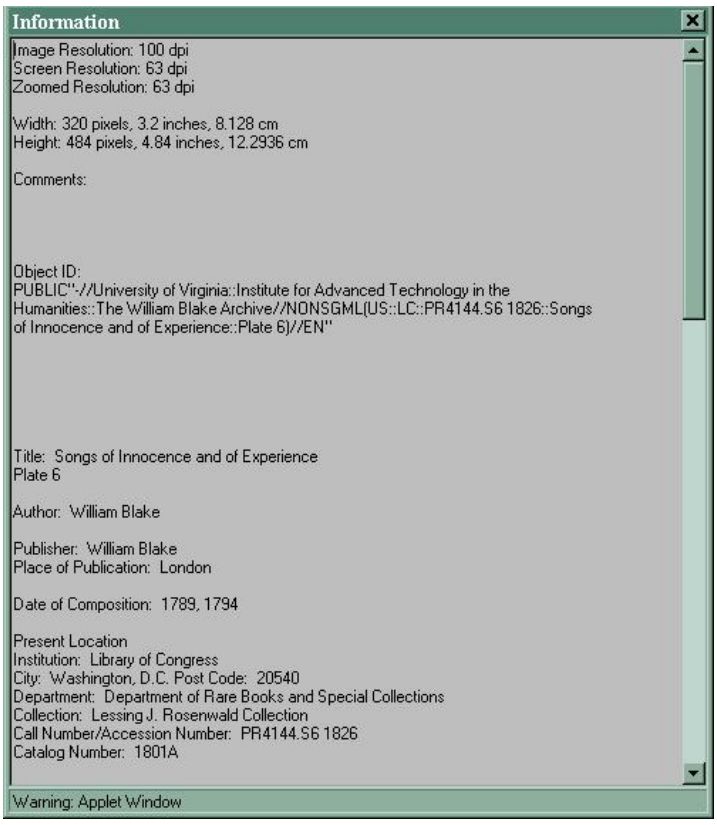
Show/Hide Panner

Show Panner will pop up Inote's panner window, a navigational device for images larger than the Inote frame. When the panner appears, you will see a blue box representing the area of the Inote frame relative to the image as a whole: by clicking your mouse on that box and holding the mouse-button down, you can drag the panner box around the image, which will move the image inside the Inote frame. NB: Inote will allow you to drag the panner outside of the area occupied by the loaded image: if, by doing this, you lose the image in the Inote and/or panner window, clicking on a scrollbar in the Inote window will recenter the image for you.

Hide Panner will make the panner window disappear. Scrollbars are also available as an alternative navigational device.

Info Button

This button (available in both Inote and the ImageSizer applet) will display information about the image loaded, including any textual information embedded in the image header.


Information
Image Resolution: 100 dpi
Screen Resolution: 63 dpi
Zoomed Resolution: 63 dpi
Width: 320 pixels, 3.2 inches, 8.128 cm
Height: 484 pixels, 4.84 inches, 12.2936 cm
Comments:
Object ID:
PUBLIC"-//University of Virginia::Institute for Advanced Technology in the Humanities::The William Blake Archive//NONSGML(US::LC::PR4144.S6 1826::Songs of Innocence and of Experience::Plate 6)//EN"
Title: Songs of Innocence and of Experience
Plate 6
Author: William Blake
Publisher: William Blake
Place of Publication: London
Date of Composition: 1789, 1794
Present Location
Institution: Library of Congress
City: Washington, D.C. Post Code: 20540
Department: Department of Rare Books and Special Collections
Collection: Lessing J. Rosenwald Collection
Call Number/Accession Number: PR4144.S6 1826
Catalog Number: 1801A
Warning: Applet Window

Plus/Minus Buttons

These buttons magnify or reduce the size of the loaded image by 5% for each click.

Status Bar

At the bottom left of the Inote window you will see status messages for the Inote software, such as "Loading Data" or "Ready".

Running Inote as an Applet

Button Method — this allows the user to launch Inote from a single applet button, which the user can then open Inote to an entire image or zoon to a preselected detail.

Autoload Method — this load Inote automatically as soon the page containing the applet tag is loaded in a Web browser. In general, this method is not recommended because it takes extra time to load the image.

Utilities

The Inote DTD — all overlay information, the coordinates of detail boundaries, the labels assigned to those details as well as the contents of text annotations are all stored in a text file, which is written in XML.

The ImageSizer Applet — this can be used independently because it is not related to Inote in any way.

Limitations

There are a few limitations to Inote 6.0 as far as images are concerned. This program only allows JPEG and GIF images. Furthermore, Inote is not available as a

stand-alone applications on the Mac, nut will run on a Mac through a Web browser. Lastly, Inote does not save overlays by http.

SID (Specimen Image Database)

Description

This particular project is more similar to the MorphBank project than any other projects being developed at the moment. According to their website, http://darwin.zoology.gla.ac.uk/~sid, it is described as "a searchable database of high-resolution images for phylogenetic and biodiversity research." Like the MorphBank project, each image can be searched by querying on certain words from the annotation text database and each registered user is authorized to submit images. Unlike the MorphBank project, however, only registered and credible users are allowed to submit annotations.

Distribution

SID is designed as a Standalone package or a Taxonomy Dependent version. These packages can be downloaded free of charge from the website: http://darwin.zoology.gla.ac.uk/~sid/host.php.

Requirements

- A web server compatible with PHP
- PHP (>4.3.0, slight modifications allow it to work with earlier versions. Instructions included)
- MySQL (>3.23.54, not tested on earlier versions)
- Imagemagick (latest version recommended)

- Ghostscript (GNU Ghostscript >7.06)
- Web service access to the Glasgow Taxonomy Name Server

Capabilities:

- Web upload/download of images (supports multiple image formats)
- Bulk and single image annotation via web forms (permits quick, simple, and detailed annotation of images)
- Extensive browse and search options (includes advanced taxonomic search options)
- Web service facility (allows external websites to display images and annotations stored by SID)
- Web utility to label specific image features (displays optional labels overlaying an image, useful for highlighting key features)
- Taxonomy served and validated independently by the Glasgow Taxonomy Name Server (maintains integrity of the data and helps "future-proof" the taxonomy)
- Alias addresses for images by accession number (permits rapid citation of images in publications e.g. http://darwin.zoology.gla.ac.uk/~sid/id/15)
- Freeware - available for download from the "Host SID" link (allows anyone to set up the database and serve their own images)

ESRI (Environmental Systems Research Institute)

Description of the Company

ESRI is a privately owned company that develops GIS, which stands for geographic information systems, specifically tailored to not only place annotations on the

map but to be able to analyze information as well. Their software allows users to draw a complex analysis from this information. ESRI is a consulting firm and is one of the largest research and development organization that is dedicated to GIS.

Description of the Software (ArcInfo software)

This particular software is stand-alone, which means it can be run on any Windows2000 and newer Windows Operating Systems, though it is mainly a desktop product. It can also be run in any Windows2000 and Windows2003 servers.

Requirements

This software requires 800 MHz of processing power on a desktop and 256 RAM of memory. It requires 700 MB of disk space and Internet Explorer 6.0 version or newer.

Distribution

To acquire the software, we must contact ESRI at 1-800-447-9778. The price of the software is not stated on the company's site.

Capabilities

This software has the ability to create multiple annotations on multiple layers. In essence, the user will be able to link annotations together and use these annotations to make an analysis of the information contained. It also has the ability to convert data into different formats and it possesses extensive printing options.

Drawbacks

The Software requires training, which will require more money spent. In addition to this, the software is massive. Since it is proprietary, we will not be able to edit any implementations to make it work with pursuant to the requirements.

Advantages

Despite the drawbacks, one thing this software guarantees is its quality. Since this is a proprietary software, the numerous testings have already been conducted on the product. Furthermore, in lieu of having questions about the software, there are experts available to answer questions. As an added plus, the software will be well-maintained, since we have the ability to purchase the updates.

Shayne Steele's Annotation Technology Software

Description

Shayne Steele's software currently has limited capabilities. It allows loading of images on the site and has basic functionalities. It is not developed to run as a stand-alone application as of yet. Our group spent three weeks to get the applet running as stand-alone, this was done through numerous visits to Shayne Steele's office and e-mails. After that, we spent more time trying to run the applet on other machines. The only machines allowed to run the applet are the Florida State University Computer Science machines, due to permission issues.

Requirements:

This software requires the installation of the Java 1.4 plug-in to run the applet. In addition to this, a requirement of at least 100 MB of memory is recommended. A server unprotected by firewall is preferred.

Capabilities

This particular software has limited capabilities as previously stated. It allows loading of images. It also captures the contributor of the image, the time the image is

loaded as well as the date, and the type of the images being loaded. As far as the annotation is concerned, each image is its own class type, consisting of a link list of annotations with the author-stamp for each annotation, the date and time stamps, the type of the image loaded, and the x & y coordinates of the pixels. Annotation can be added by pointing the mouse to a certain location on the image, capturing the x and y coordinates of the pixels and storing the text on that location. Thus searches will be queried through the annotation text body. It also enables zooming in and out of the images.

Drawbacks

There are several drawbacks on this software. One of them is that it does not run as a stand-alone application. Also, the Java 1.4 plug-in needed to run the applet takes a long time to download and requires at least 100 MB of memory. Furthermore, the applet will not run on servers protected by firewalls.

Advantages

The positive feedback of this software is that it is possible to make it run as stand-alone. And since the software is developed locally, by our very own Shayne Steele, problems with the applet can be easily reported by visiting Shayne Steele's office or by e-mailing him. Another good feedback is that additional requirements can be easily met since Shayne Steele is accessible around campus.

Group Opinion and Recommendation

As a group, we have decided through research and experience working with the above software products, that the Inote software is a winner. Although it is mentioned that the Inote software has some drawbacks, it appears to have the most capability and

conforms to the project's requirements more than the other software products we have researched. There are more advanced features on the Inote software: i.e. allowing different styles of overlays such as oval, circular, rectangular, and so on. The user can also change the colors on the annotation text. It also interacts with the server better than Shayne Steele's software. Most importantly, it does create the XML document that meets the requirements and specifications of the project.

However, the Inote 6.0 does require more work. It has not been updated since its release in the year 1998, but with permission granted, and more time given on this project, one can take the existing code for the Inote 6.0 software and add more code to meet the exact specifications of the MorphBank Project.

APPENDIX C – DATABASE SCHEMA AND LOW LEVEL SOFTWARE LIBRARY

PHP and MySQL

Read me file for PHP and MySQL code for MorphBank system.

Our part in this project was exploring the technology of PHP scripting with MySQL, and to create some examples to demonstrate what you can do with it in order to save time by lowering the learning curve for the future coders. I was paired up in this task with Robert Worley who has been an invaluable, dedicated partner, and together we have succeeded in our task. The whole of our work I have tied together in a couple of web pages which allow you to add to, update, and delete from an example table in the database we created. They are examples to how these functions are performed on a database through PHP/MySQL scripting. This web interfacing was very easy through PHP/MySQL's seamless integration with HTML scripting. The web-page to view this work is http://www.cs.fsu.edu/~logan/index.php.

Rob and I first began with tackling the task of getting the functions to work on the example database and checking it with the "show tables" command at the mysql command line. This was probably the most arduous of the work to be done because we had to start from scratch knowing nothing of PHP/MySQL. However, like most programming, we found that once you get started things start to sink in and make more sense, and thus it was easier as we went along. I personally undertook the update command. I created several textboxes for a user to enter the data that they wanted to change in the database. Then using HTML forms, I passed the data to a PHP page which

processed the data and executed the proper MySQL command to fulfill the user's wishes.

Once we had all of the functions working, I decided we needed an easier interface for a user who didn't have access to our accounts, and thus couldn't use the command line to see the results of executing our commands on the database. I accomplished this through our main web page to which I provided a link to each function. But more so through printing out the data contained in our example table on the HTML page for the user to see. Then when the user executed a command, the page would take them back to the main index page where they could now see the change their command execution had. Printing out the data was rather tricky but I did it by storing the table's data in arrays, then printing out each columns data under its proper title using a for loop.

All in all I am very satisfied with our groups performance, which is bolstered through the praise of our director. Knowing just the little bit I have learned of PHP/MySQL, I feel it would be a wonderful choice to use in the MorphBank project if simply for the reasons of its' seamless integration with HTML and low learning curve. Another reason I would recommend the use of this language is because it will allow us to create an interface which the biologists are already familiar with and want, which the rival of PHP/MySQL, WebSphere, I am not sure will be consistent with in that it, to my understanding, scans a database then automatically creates an interface for the user. If the database's underlying structure is very well organized and normalized, then web services will create better queries, and overall perform seamlessly with the system. I would also

recommend PHP/MySQL because it is open source which also conforms to the requirements set forth by Dr. Ronquist.

Code Detail

Requirements

There are a few requirements needed to run the PHP code.

1. Most current release of PHP
2. Most current release of MySQL
3. Most current release of Apache

For PHP download: http://www.php.net/downloads.php

For MySQL download: http://dev.mysql.com/downloads/mysql/4.0.html

For Apache download: http://httpd.apache.org/download.cgi

Apache is web server software that is needed to interact between PHP and MySQL.

Note: All this software is FREE.

PHP

PHP is simple code that can be placed within HTML code to produce dynamic web pages. PHP is powerful in the fact that it is compatible with many different types of databases (Microsoft Access and Oracle). PHP was used with the MorphBank system mainly because of the compatibility reason and that it is open source software. Another powerful advantage with PHP is that it is well documented (many examples can be found via the internet and books). The most helpful web site is probably: http://us4.php.net/manual/en/ this site is basically a manual for all the built in PHP

functions and all additional plug-ins. Note: In certain areas of this document it is assumed that you will have minor programming experience.

Notes on PHP code for MorphBank

Most of the code was written with the idea that it would be used in the future (the real MorphBank system). The most time was spend on add_to_table, add_to_table2, input_data, and functions; these files were written to automatically detect the number of tables in an database, the number of fields, identifies required fields, and does some minor error checking. A simple search file was created that will search one table (species) for a specific familyname. This file was written solely for demonstration purposes but may also provide some good example for future use. The delete and update files work hand-in-hand with the search file. Once a user access the code and sees it visually, this will all make sense.

For future use: One of the most difficult problems was trying to pass variables from one web page to another. All the files used in the MorphBank system manually pass the variables from one web page to another (having to declare most of them again on every page) which became a pain. There is an easier way: use the built in PHP function session_start, it can be found on the web site cited in part one of this document. This would have been used (trust me) if I would have known about it in advance. Also, code should be written to validate the data entered into the database, such as correct date, type of familyname, and so on. The final recommendation would be to remove the global.php file that was created for security reasons. The global file contains global variables used in each file which I found to be dangerous when reading about security treats to a system.

An easy way around this would be to use the first recommendation, the session_start function.

Description of PHP files

Here will be a description of all the PHP files used in the MorphBank system. This section of documentation will cover areas of the source code that would not be easily understood. All the source code will be provided for you to review and study. Note: most of these files work with each other (so in other words, you need all the files). Note: Every PHP statement must end with a semicolon.

Connect.php

The connect file was written to connect to a specific database. The reason it is specific is that there are a few certain parameters that must be hard coded into this file such as the database server, database name, user name, and user password. If used properly, the connect file can connect to any database as long as the correct information (listed above) entered into it. For example: in the connect code there are lines that read:

```
$dataBaseServer='dbsrv.cs.fsu.edu';
$dataBaseUser='worley';
$dataBasePassword='robert';
$dataBaseName='worleydb';
```

The dollar sign in front of the name declares that as a variable or in other words creates memory to store the name provided to it. You can see here that these variables are hard coded, by changing what is in the single quotes you could connect to any database so

long that database exists and the data entered is correct. Once the data is entered there are a couple of built in PHP functions that must be used to connect to the database, they are:

mysql_connect(server, user, password)

mysql_select_db(database name)

The function mysql_connect takes three parameters, the server name, user name, and user password. This function makes the actual connection to the database server. Once the connection has been made, the correct database must be selected (the database server could have several databases on it). The function mysql_select_db only takes on parameter, database name and makes a connection to that specific database.

Functions

This document contains several functions that are used in many of the PHP files. Basic definition of a function is a piece of code that is written once but can be used several times. The functions in this file are:

get_tables() - Gets the name of the tables.

get_num_tables() - Gets the number of tables.

get_number_fields()- Gets the number of fields.

get_f() - Gets the name of each field.

get_field_length() - Gets the length of each field.

get_required_fields() – Gets the required fields.

Each function is well documented in the functions file but I will go over one of them here for explanation purposes. The get_number_fields() functions looks like this:

```
1       function get_number_fields($dbn, $tba, $i) {
```

```
2       $num_fields_array = array();
3       for($i1 = 0; $i1 < $i; $i1++) {
4       $query = "SELECT * FROM " . $tba[$i1];
5       $result = mysql_query($query);
6               if(!$result) {          //If error
7               echo "Failed Query";
8               exit;
9               }
10      $num_fields = mysql_num_fields($result);
11      $num_fields_array[$i1] = $num_fields;
12      }
13      mysql_free_result($result);
14      return $num_fields_array;
15      }
```

Starting at 1, the keyword function tells PHP that the lines of code between the opening { and closing } are a function. This function takes three parameters $dbn, $tba, and $i. These are just shortened names for database name (dbn), table array (tba), and ‘i’ is the number of tables in the database. At line 2, an array is declared to store how many fields are in each table. Line 3, a for loop is declared that goes through all the tables in the database. Line 4 has a MySQL query statement is used to query a table. The order and form of the query statement is important or else many errors will be encountered. The “ “SELECT * FROM “ . $tba[%i1] “ part tells the query to select everything from the

current table being accessed. The period is used to append variables onto the end of text (similar to the plus in java). Line 5 uses a PHP built in function called mysql_query, it accesses the table and performs the query defined. Lines 6 through 9 do simple error checking to see if the query succeeded or not. Line 10 also uses a PHP built in function called mysql_nem_fields, this function takes the result from the query performed and counts the number of fields in the result. Line 11 stores that number into the array. Line 12 is the end of the while loop. Line 13 frees the query performed on the database tables. This should be done because if there are several people accessing the database large amounts of memory will be used if the result is not released (so you're basically freeing memory). Line 14 returns the array and Line 15 ends the function. Note: Please make sure you understand the explanation of the function; most of the functions are similar. If you understand how get_number_fields works, the rest of the functions will be very easy to follow.

Add_To_Table and Add_To_Table2

These files allow a user to enter data into the database. This code was written to work with any database; it makes use of the PHP files connect global, and function. The add_to_table files use the keywords include and require, these keywords allow a user to include another file or require that one be included or executed. This line require_once 'Connect.php'; requires that connect be run once which will connect to the database. The lines include 'functions.php'; and include 'global.php'; make those two files accessible by the add_to_table file. The add_to_table file makes use of all the function in the function file (it makes a call to each one). Once all the data has been collected by the function

calls, the file uses that data to create several text boxes for the user to enter data into. To create text boxes a form must be used, it is a embedding in HTML. To use HTML commands inside PHP code you must use either the echo command or the printf command. Note: I found the printf command to be much better and useful. Once the user enters all the data in the text fields and clicks the “next table” button, all the data they entered is automatically sent to the input_data file. Below is a screen shot of what the web page produced by this code looks like. This could be easily modified to be put into a HTML template file.

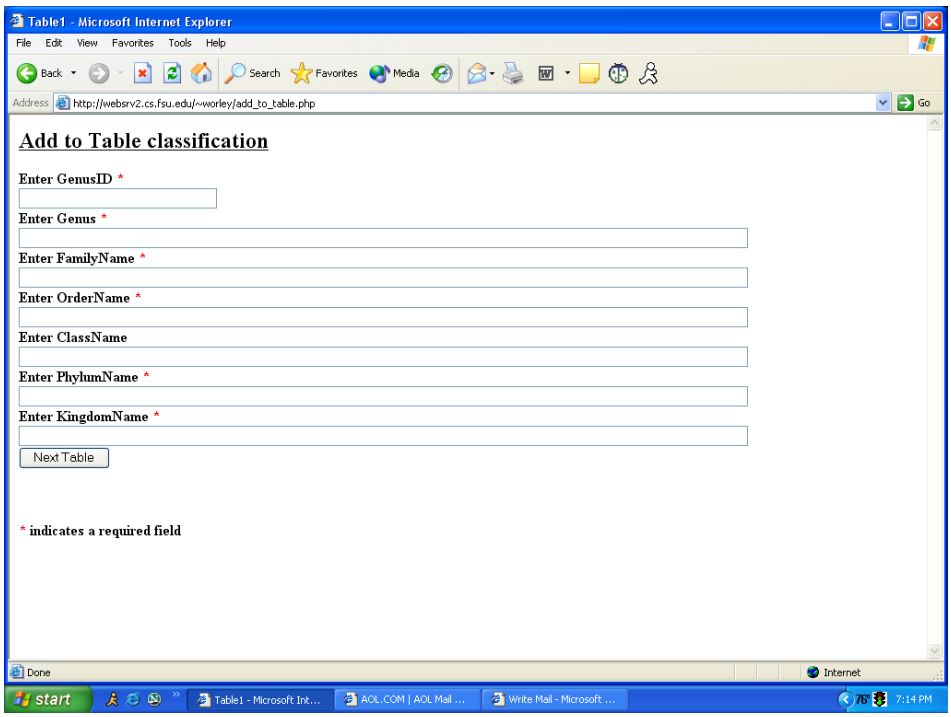


Table1 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://websrv2.cs.fsu.edu/~worley/add_to_table.php
Go
Add to Table classification
Enter GenusID *
Enter Genus *
Enter FamilyName *
Enter OrderName *
Enter ClassName
Enter PhylumName *
Enter KingdomName *
Next Table
* indicates a required field
Done
Internet
start
Table1 - Microsoft Int...
AOL.COM | AOL Mail ...
Write Mail - Microsoft ...
7:14 PM

Input_Data

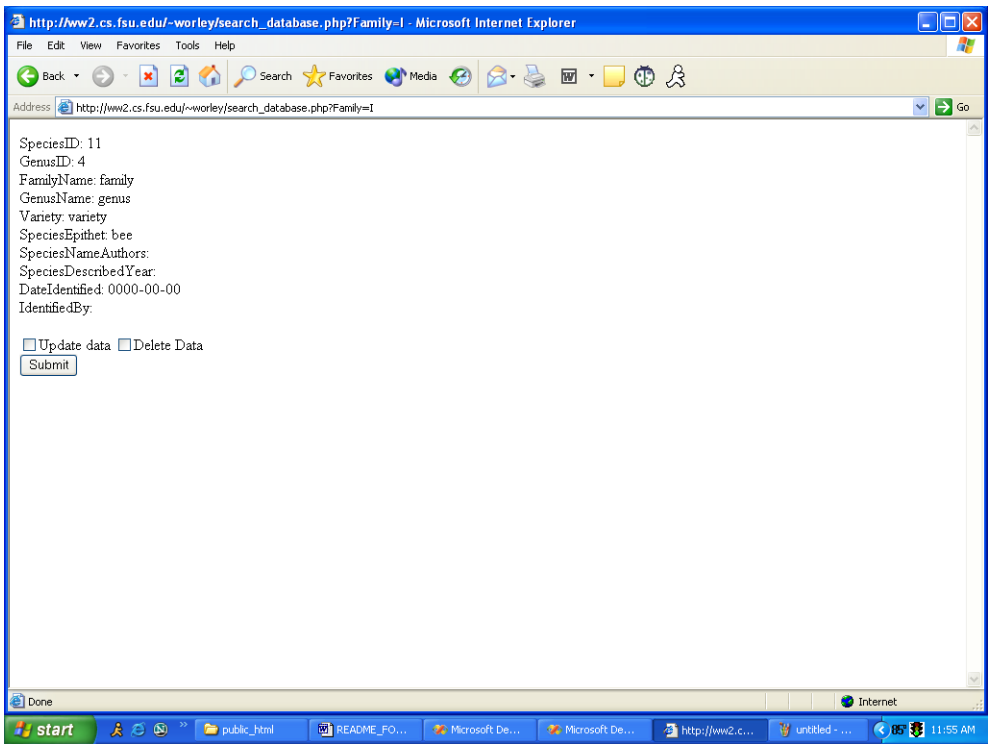
This file was one of the most difficult to write so it is well documented in the code. In short, the input_data file receives the data from the add_to_table file and stores it into an array. For there it checks to see if the required fields were entered, if not error. It then proceeds to build a query using the received data. A while loop was used to create the query which was a little difficult but is explained in detail in the code, please look it over! The query adds the data to the database then the user is automatically refreshed to the next table or back to the homepage.

Global

This is an extremely small file; it just contains a few global variables and initializes a few of them. The PHP keyword global is used here; this tells PHP that this variable is global.

Search_Database and Search

This is a very simple search created just demonstration purposes. It allows a user to search a table by the familyname and returns all the data found by that familyname. It then provides a list of that data and two checkbox options: update the data or delete the data. It was setup this way with the idea that this is how it would be in the real system (User friendly). The user simply has to click which option they would like and the data will either be deleted or they will be taken to another web page taking them through the update process. Below is a screen shot of what this all looks like:



Delete

This file gets a few parameters from the search_database file and performs a query on the specified table to delete the data. Notice the how the query is formed:

$query = "delete from $table_list[$increment_table] where $column=$value LIMIT 1";

Take notice of the LIMIT 1 part of the query, this is important; it tells MySQL to only delete one data entrée with the specified data. If LIMIT 1 is not in the query statement it will delete all rows.

Update_Delete and Update_Input

These two files are the same concept as the rest of the files. It takes data from the user on which particular field they want updated and performs the query. The files use a form to pass the data and the query looks like:

```
$query = "UPDATE $Name SET $Field = '$Value' WHERE SpeciesID= '$SpeciesID'
LIMIT 1";
```

Notice this has the LIMIT 1 as well, the same concept here; more than one data item will be updated if this is not in the query statement.

If you do not understand something please refer to the code, most of the code has comments explaining what is happening at each step.

Java Implementation

Documentation for PHP/MySQL and Java code for MorphBank©

What we did?

What the programmers of CEN4010 attempted to accomplish during the tenure of the class was to create enough coded material and documentation for future programmers and analysts to use for the completion of the system and all of its components. The result…a usable system that shows all aspects needed to mold a useful database for the MorphBank system. The code was divided up among members of the class, tasks were delegated, and code was written. The members accomplished the tasks of setting up

prototyped code and documented examples for future programmers to utilizes and study for the completion of MorphBank.

How we did it?

Using PHP, which is a HTML friendly language, tables were formed and a basic database was created to add information to the entire MorphBank system. The PHP code is essential for website input into the MySQL database. Once the user has entered the information through the website, the PHP sets it up into a basic MySQL database for temporary usage. That is where the Java code will be implemented.

The Java aspect of the program is the most essential part of the system since it is the most powerful code in the program and is the most useful. The Java code interprets the MySQL tables created by the PHP and transfers the data into the MorphBank database. This is done table-by-table and processed for errors and malicious data. Once the Java has migrated all the database information, the database is saved and the system is updated.

The convenient aspect of using PHP rather than only using Java is that PHP can create tables from an online input screen easily and effectively for temporary purposes. Where as Java would create many large tables to accommodate first hand input and run through many error checks before finally saving it to the large system. Another benefit of using Java and/or PHP is that, both are compatible with any database system that the programmer wishes to utilize (i.e. Oracle, MySQL, Access, etc). This could come in handy for temporary databases that have already been created and need to be transferred to the new MorphBank system.

Why we did it?

We constructed the MorphBank code in order to give future programmers of the system ample examples and prototypes to actually construct the entire program. The coding consists of numerous parts including code programmed in PHP, Java, and MySQL. All of these programming languages work together to create a system that is fast, efficient, and can be used on multiple platforms without compatibility issues.

The code takes data that was stored in a previous database, migrates it, organizes it, and places it into its respective tables inside the new database. This is the programs goal at the end of the program but so far the actual coded material is only in prototype form and is not operational at this time.

APPENDIX D – TRADEMARK AND COPYRIGHT

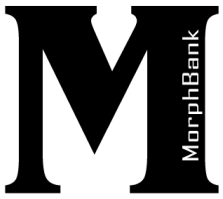
Overview

The Trademark and Copyright team were assigned to create a trademark in which MorphBank could be recognized by the users of their website and/or interested parties.

The overview of this project is to create a trademark, which is a word, phrase, symbol or design, or a combination of words, phrases, symbols or designs, which identifies and distinguishes the source of the goods of one party from those of others.

Once the trademark is approved by the MorphBank Team, it will be incorporated with the webpage, which is being designed by the WSRD project team.

Proposed Trademarks

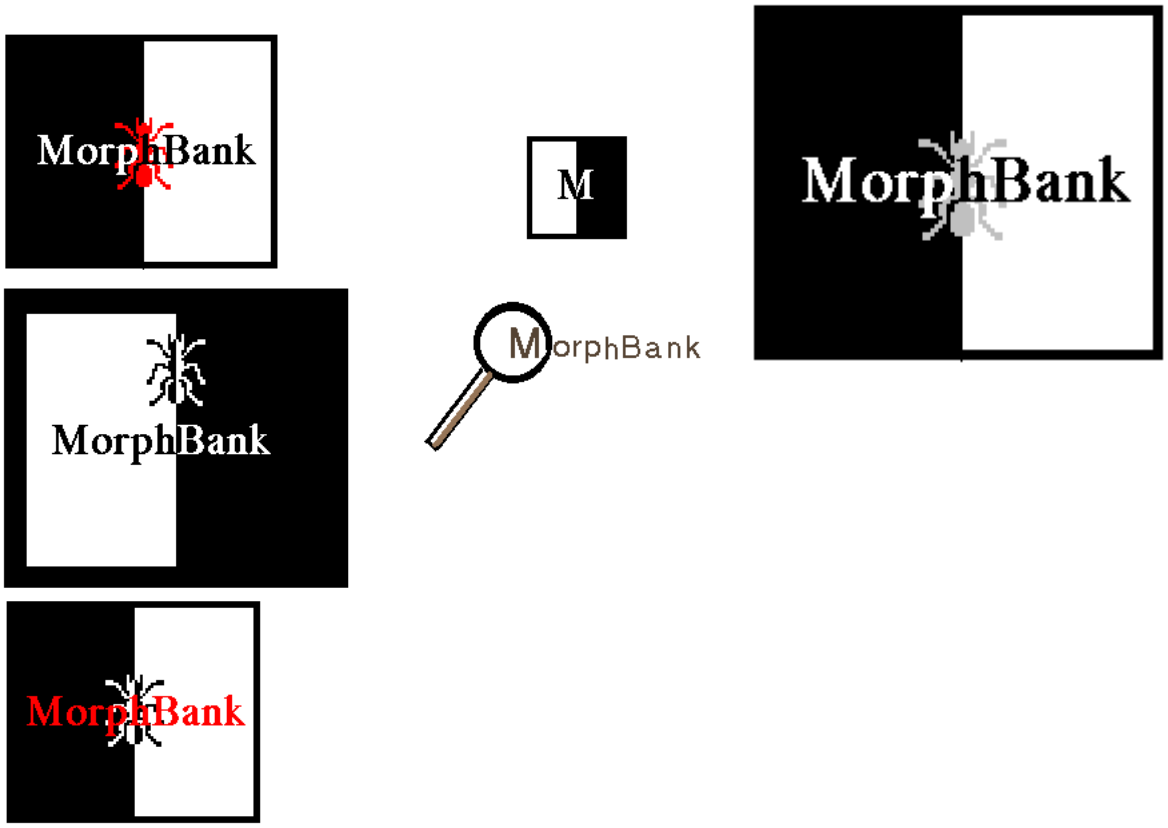



MorphBank


MorphBank


MORPHBANK


MORPHBANK


MorphBank
M
MorphBank
MorphBank
MorphBank
MorphBank

TEAM MEMBERS

Team I — MorphBank Web Site Requirements and Design

Demetrius Brown – Annotations/Image search and web search layout
Richard Cook – Web site template design and web functionality requirements
Duane Griffiths – Specimen data display and web site search layout
David Kopicki – Specimen search page and web site search layout
Antoinette S. Mulai – Taxonomy search page and web site search layout
Yuval Peress – Advanced search page and search results page

Team II — MorphBank Database Schema and Low Level Software Library Development

Thomas Bonfield – Documentation and Java implementation
Michael Jason – New database schema and data validation
Gabriel Logan – PHP, MySQL and presentation
Christi Shirley – New database schema and data validation
Robert Worley – PHP, MySQL and documentation
Keith Zenoz – Java implementation and documentation

Team III — Graphical Annotation Technology

Christopher Albritton – ERSI research and XML coding of Steele's GAT
Nikki Brown – Research other existing GAT
Kerstin Galutera – Full documentation for GAT and research on GAT
Kowit Jitpraphai – Implementation of Inote GAT
Johan Martinez – Implementation of Steele's GAT

Team IV — MorphBank Operational Specifications

Joe Barrett – Access control
Daniel Beech – Mirror site configuration
Justin Christofoli – Data modifications
Jesse Levier - Security
Michael Lind - Backup
Saif Mazhar – Operational server configuration

Team V — MorphBank Trademark and Copyright

Erica Bourne – Trademark research
Luisa Pleger – Copyright research

Team VI — Server and Software Configuration

Bruce Bayha – System process validation
Massa James – Hardware configuration

Document Configuration Management

Theresa Pace – Documentation