Forager : Project Analysis Report

Cycle 1

November 6, 2012

by



Matthew Powell

Robin Mays

Thomas Couture

Samuel Hall

A project report submitted for

SWE3613 Software Engineering Systems

Fall 2012

Department of Computer Science and Software Engineering

Southern Polytechnic State University

Marietta, Georgia

Table of Contents

[1. Introduction 4](#_Toc337502203)

[1.1 Executive Summary 4](#_Toc337502204)

[1.2 Project Goals 4](#_Toc337502205)

[1.3 Cycle Goals 4](#_Toc337502206)

[Sprint 1 5](#_Toc337502207)

[Sprint 2 5](#_Toc337502208)

[2. Requirements 6](#_Toc337502209)

[2.1 Project Environment, Technology, Hardware, Etc. 6](#_Toc337502210)

[2.2 User Stories 6](#_Toc337502211)

[3. Design 10](#_Toc337502212)

[3.1 System Architecture 10](#_Toc337502213)

[4. Management Plan 13](#_Toc337502214)

[4.1 Planned Assignments and Schedule for First Cycle 13](#_Toc337502215)

[Week 1: 13](#_Toc337502216)

[Week 2: 13](#_Toc337502217)

[Week 3: 13](#_Toc337502218)

[4.2 Actual Assignments and Schedule for FIRST Cycle 13](#_Toc337502219)

[Week 1: 14](#_Toc337502220)

[Week 2: 14](#_Toc337502221)

[Week 3: 14](#_Toc337502222)

[4.3 Planned Assignments and Schedule for CURRENT Cycle 14](#_Toc337502223)

[Week 1: 14](#_Toc337502224)

[Week 2: 14](#_Toc337502225)

[Week 3: 15](#_Toc337502226)

[4.4 Actual Assignments and Schedule for FIRST Cycle 15](#_Toc337502227)

[Week 1: 15](#_Toc337502228)

[Week 2: 15](#_Toc337502229)

[Week 3: 15](#_Toc337502230)

[5. The Honeycomb Features Walkthrough 17](#_Toc337502231)

[6. Cycle Post-Mortem Analysis 21](#_Toc337502232)

[6.1 Successes 21](#_Toc337502233)

[6.2 Failures 21](#_Toc337502234)

[6.3 Lessons Learned/Risk Mitigation 21](#_Toc337502235)

[7. Test Plan and Procedures 22](#_Toc337502236)

[7.1 Test Plan 22](#_Toc337502237)

[7.1.1 Introduction 22](#_Toc337502238)

[7.1.2 Test Items 22](#_Toc337502239)

[7.1.3 Software Risk Issues 22](#_Toc337502240)

[7.1.4 Approach 22](#_Toc337502241)

[8. Code 24](#_Toc337502242)

LIST OF ABBREVIATIONS

|  |  |
| --- | --- |
| HTML  JSON | Hypertext Markup Language  JavaScript Object Notation |
| OS | Operating System |
| SQL | Structured Query Language |
| UI | User Interface |

# Introduction

## Executive Summary

Group4 is the producer of the file sharing website analysis tool Forager. Forager will allow a systems administrator or webmaster to easily scan their site for broken links and missing resources, then offer an easy way to generate and compare reports.

Forager is user friendly and portable. Users are provided access to reports and scanning tools through a website produced using common web standards. This means that Forager is accessible from any PC, laptop, tablet, or mobile device regardless of the client OS.

## Project Goals

Forager is a web based website analysis tool. A user will login to the service and be able to start a variety of scans of their website. When the scans have completed, they will then be able to examine the results of the scans and compare them in various ways.

Forager will allow scans to be generated starting from the front page, limited to specific subdomains, or limited by time and distance from the front page. It will also allow the user to use a list of broken links from a previous scan instead of revisiting the entire website.

Forager will then allow users to sort their scans based on page load time, response time, errors, and the individual subdomains on which the page was found.

Group 4 plans to complete Forager’s features in two sprints. The project is expected to be complete by November 27, 2012.

## Cycle Goals

Group4’s goal is to complete Forager in two sprint cycles as outlined below.

### Sprint 1

Sprint 2

# Requirements

## Project Environment, Technology, Hardware, Etc.

The Forager is a web application written in a combination of PHP5 and Python. PHP is a widely-used, general-purpose scripting language that is especially suited for Web development and can be embedded into HTML, the primary markup language for displaying web pages in a web browser. Python is a common scripting language that has a rich set of features for interacting with webservers and processing HTML data. Both the Python and PHP components communicate with a PostgreSQL 8.4 database back end and the user interface is served by the Apache 2.2 webserver. This selection of mature, multi-platform technologies will allow Forager to run with little modification on most modern operating systems. It is currently being developed and tested on a virtual machine running Debian Linux 6.0 (Squeeze) on VMWare ESXi 4.1.

Users of this application are not limited to Linux or any major web browser. Any computer capable of running a web browser that supports basic HTML and SSL, which includes all of the most popular mobile and PC browsers, will be able to access the application.

## User Stories

# Design

## System Architecture

The Forager system consists of two parts. The webcrawler is written in Python3, using the requests library (a wrapper around the native urllib3 HTTP client). It uses PsycoPg2, a PostgreSQL connector that supports the DB-API interface defined in PEP 249. The report viewer and user interface is written in PHP 5, which is served by an Apache 2.2 Web server. Scan results are stored in a PostgreSQL 8.4 database, and all components are deployed on a machine running Linux.

A user wishing to interact with the system will go to the login page and enter their authentication credentials. They will then be presented with an option to start a new scan or view the results of existing scans. Starting a scan will check the database for any scans not marked as completed and attempt to send a Continue (SIGCONT) signal to them. If any of these signals succeed, the user is alerted that a scan is already in progress, otherwise the crawler process is spawned. On startup, the crawler will initialize its database connections and logs, and then use a simplification of Richard Stevens’ daemon initialization algorithm to cleanly detach from the console. It will then create a scan record in the database, storing its process ID and the time that it was started.

The crawler uses a resource object to represent the URLs that it is given. When the crawler is initialized, it creates an object for the initial URL and stores a reference to it in a hashtable of existing resources and in a pending queue for objects that have not yet been retrieved. This object initially contains only the URL and a null link to its parent. The crawler then processes the pending queue until it is empty by retrieving the first element, using the resource’s fetch() method to visit the page and store relevant information, like the HTTP response code and the load time. If the resource is an HTML page, the HTML is parsed and a list of children is stored in the resource. The resource’s SQL\_Call() method is then used to store the resource in the database, whose row structure matches the object definition. If any children were found in the resource, the crawler will iterate over them, and any that do not already exist in the resource list have resources created for them and are placed in the pending queue and the resource list. When the pending queue is exhausted, all resources that meet the restrictions of the crawler and that are reachable from the first page will be stored in the database, and the parent records will describe a spanning tree of the site map. When the queue is exhausted, the crawler will store its exit time in the database and shut down.

Once a scan has been registered in the database, it will be visible from the scans page on the website. The data is retrieved from the database and converted into JSON (JavaScript Object Notation). This JSON data is loaded by jQuery and processed into a sortable table with the DataTables jQuery plugin. Each of the scans can then be clicked to retrieve a list of the URLs visited in a similar manner. The list will show and allow sorting based on the response time, response code and URL.

# Management Plan

## Planned Assignments and Schedule for First Cycle

Group 4 planned assignments for cycle 1 will breakdown as follows:

### Week 1:

The Team Assignments

Gather requirements

Tune User Stories

Assignment of tasks

Individual Assignments

Matthew Powell - Requirements

Robin Mays - Requirements

Thomas Couture - Requirements

Samuel Hall – Requirements, Set-up server

### Week 2:

The Team Assignments

Revise requirements

Tune User Stories

Weekly Status Report

Individual Assignments

Matthew Powell – Testing, Documentation

Robin Mays – Coding: Web UI, Documentation

Thomas Couture – Coding: Web UI, Documentation

Samuel Hall – Coding: Backend, Documentation

### Week 3:

The Team Assignments

Weekly Status Report

Project Analysis Report

Project Demo

Individual Assignments

Matthew Powell – Testing, Documentation

Robin Mays – Coding: Web UI, Documentation

Thomas Couture – Coding: Web UI, Documentation

Samuel Hall – Coding: Backend, Documentation

## Actual Assignments and Schedule for FIRST Cycle

### Week 1:

The Team Assignments

Gather requirements

Tune User Stories

Assignment of tasks

Individual Assignments

Matthew Powell - Requirements

Robin Mays - Requirements

Thomas Couture - Requirements

Samuel Hall – Requirements, Set-up server

### Week 2:

The Team Assignments

Revise requirements

Tune User Stories

Weekly Status Report

Individual Assignments

Matthew Powell – Testing, Documentation

Robin Mays – Coding: Web UI, Documentation

Thomas Couture – Coding: Web UI, Documentation

Samuel Hall – Coding: Backend, Documentation

### Week 3:

The Team Assignments

Weekly Status Report

Project Analysis Report

Project Demo

Individual Assignments

Matthew Powell – Testing, Documentation

Robin Mays – Coding: Web UI, Documentation

Thomas Couture – Coding: Web UI, Documentation

Samuel Hall – Coding: Backend, Documentation

## Planned Assignments and Schedule for CURRENT Cycle

### Week 1:

The Team Assignments

Gather requirements

Tune User Stories

Assignment of tasks

Individual Assignments

Matthew Powell – Requirements, Coding, Documentation

Robin Mays – Requirements, Coding, Documentation

Thomas Couture – Requirements, Coding, Documentation

Samuel Hall – Requirements, Coding, Documentation

### Week 2:

The Team Assignments

Revise requirements

Tune User Stories

Weekly Status Report

Individual Assignments

Matthew Powell – Testing, Documentation

Robin Mays – Coding: Web UI, Documentation

Thomas Couture – Coding: Web UI, Documentation

Samuel Hall – Coding: Backend, Documentation

### Week 3:

The Team Assignments

Weekly Status Report

Project Analysis Report

Project Demo

Individual Assignments

Matthew Powell – Testing, Documentation

Robin Mays – Coding: Web UI, Documentation

Thomas Couture – Coding: Web UI, Documentation

Samuel Hall – Coding: Backend, Documentation

## Actual Assignments and Schedule for FIRST Cycle

### Week 1:

The Team Assignments

Tune User Stories

Assignment of tasks

Individual Assignments

Matthew Powell –Documentation

Robin Mays – Documentation

Thomas Couture – Documentation

Samuel Hall – Coding (File backend), Documentation

### Week 2:

The Team Assignments

Weekly Status Report

Individual Assignments

Matthew Powell – Documentation

Robin Mays –Documentation

Thomas Couture –Documentation

Samuel Hall – Documentation

### Week 3:

The Team Assignments

Weekly Status Report

Project Analysis Report

Project Demo

Individual Assignments

Matthew Powell – Coding (Password Reset, Terms of Service), Testing

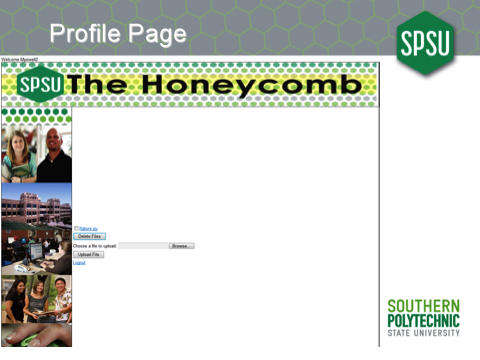
Robin Mays – Coding (Group Functionality), Documentation

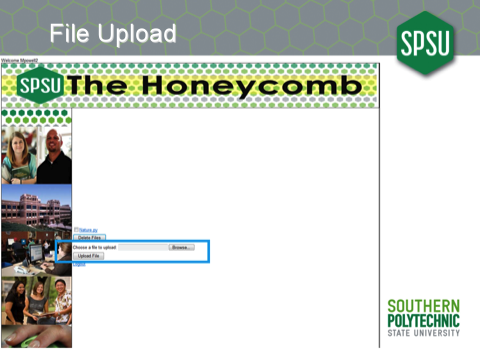
Thomas Couture – Coding (Admin Interface, Group Functionality), Documentation

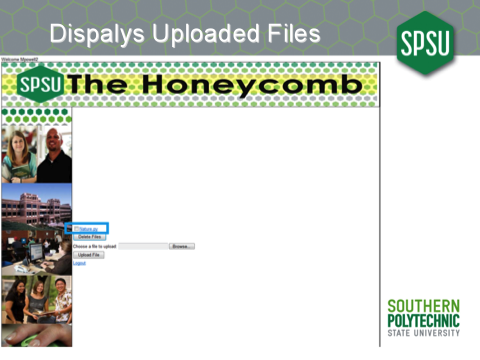
Samuel Hall – Testing, Documentation

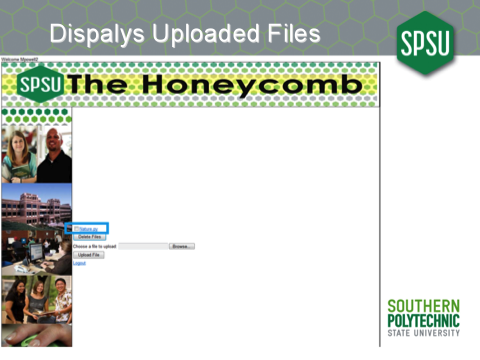
# The Honeycomb Features Walkthrough

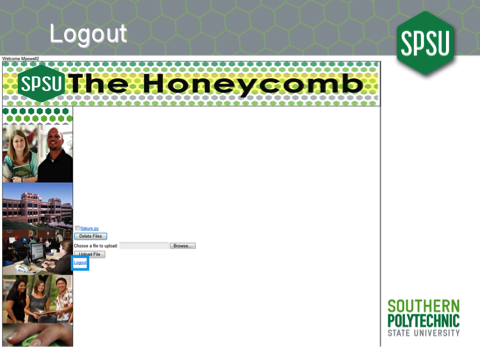














# Cycle Post-Mortem Analysis

## Successes

Group 4 welcomed many successes during the course of Cycle 1. They were:

We were able to create complete, concise, and workable user stories.

Creation of the log in page that blocked the user’s password from sight.

Creation of a Registration page that only accepted unique emails.

Allowed users to upload to their directory.

Users can download and delete files from their directory.

## Failures

Things that went poorly in Cycle 1 were:

Group 4 failed to fully research the bugs that in PHP that would affect the way we implemented our project.

There also was a bug affecting the ability to upload some types of files. This came from the PHP error and miscommunication about the size limit for file uploads that was a self-imposed limit.

There was also a communication issue with GitHub that delayed us a day.

## Lessons Learned/Risk Mitigation

In the future, we will use better debugging techniques. We lost too much time assuming that a problem was related to the browser being used for testing and not looking at other underlying similarities between the test cases that cause the bug to occur.

# Test Plan and Procedures

## Test Plan

### Introduction

The test plan for the Honeycomb project will involve offering inputs to the server that should produce both successes and error conditions in an effort to ensure that the software behaves as expected in response to use, misuse, and abuse. We would also like to ensure that any information leakage from the application is accounted for, as well as cataloging outright error conditions.

### Test Items

Tests will be designed around the each user story. The application will then be examined for any possible user interaction that was not covered under a user story, so that tests may be designed for those cases as well. Comprehensive load testing will also be needed, but it may be left off until the next cycle, depending on rollout projections. As we do not currently have any maintenance processes or other systems that would cause the system to act outside of user interaction, we can ignore a large slew of testing scenarios. Similarly, we are not currently doing any client-side validation of user input, so most malicious input can be tested directly from the user interface, instead of having to test both the client-side code, and then bypass it to test server-side code (Exceptions to this will be covered in the individual test procedures).

### Software Risk Issues

As the service is not currently in production, there are few risks in the testing scenario. We do not have any customer data in the system that can be lost or compromised, and any outages that are created will not be customer visible. As the testing machine does share hardware and bandwidth with other production services, when it occurs, we will need to plan our load testing carefully to ensure that Quality of Service guarantees of both bandwidth and cycles are met throughout testing.

### Approach

Testing of User Interface (UI) elements will be done with the 3 most common web browsers: Google’s Chrome, Microsoft’s Internet Explorer, and the Mozilla Foundation’s Firefox. Malicious non-UI input will be tested with the TamperData extension to Firefox and direct access via telnet. Items will be considered passing if the application behaves as expected. In response to legitimate user input, the application should either take the action requested, provide the user clear instructions on how to proceed, or notify the user and the system administrator of an uncorrectable failure of the application. In response to abusive input that cannot be accomplished directly from the user interface, the application should refuse to leak information. Useful information may be provided when it does not leak information, but generic failure messages are also acceptable.

# Code

Headers and Helpers

<?php /∗

∗ ∗ ∗ ∗ ∗ ∗/

config.php  
−Lee Hall Thu 06 Sep 2012 10:10:03 PM EDT

This f i l e opens the database connection and provides some useful global variables to the project

$conn\_str="user=apache dbname=honeycomb"; $conn= pg\_connect($conn\_str);

if (!$conn)  
die("Unable to connect to database .");

$URL\_BASE=" h t t p s : / / $\_SERVER [ SERVER\_NAME ] " ; $FILE\_STORE="/var/www/honeycomb/file\_store "; $FILE\_URL=$URL\_BASE . "/ file\_store ";

?>

<?php /∗

∗ ∗ ∗ ∗ ∗ ∗/

session .php  
−Lee Hall Thu 06 Sep 2012 10:13:49 PM EDT

Check that a session exists .  
If not , bounce them to the login page and die .

session\_start ();  
if (! array\_key\_exists(’user\_id’, $\_SESSION) ||

! i s s e t ( $\_SESSION [ ’ u s e r \_ i d ’ ] ) ) { header( "location : login .php"); die("User not logged in");

}

?>

<?php /∗

∗ index.php  
∗ −Lee Hall Thu 20 Sep 2012 01:03:13 PM EDT ∗/

//Check if the user is logged in. require\_once("include/session .php");

// I f they are , send them to the main page header("Location : profilePage .php");

?>

2 Login

<?php /∗

∗ ∗ ∗ ∗ ∗/

i f

login.php  
−Lee Hall Thu 06 Sep 2012 10:23:45 PM EDT

Allow the user to login  
r e q u i r e \_ o n c e ( ’ i n c l u d e / c o n f . php ’ ) ; //Is there a user trying to log in?

} }

i f

( array\_key\_exists ( ’ logout ’ , $\_POST)){ session\_destroy ();  
header("Location: $\_SERVER[PHP\_SELF]");

( array\_key\_exists ( ’ login ’ , $\_POST)){

i f ( ! array\_key\_exists ( ’ user\_name ’ , $\_POST) | |  
! array\_key\_exists ( ’ password ’ , $\_POST) ){

die("User or password not set. How did you get here?"); }

// Get user info from database. Only retrieve users who have authenticated // their accounts .  
$sql="SELECT user\_id , password FROM users

WHERE user\_name=$1 AND auth\_hash IS NULL; " ;  
$params=array ($\_POST[ ’ user\_name ’ ] ) ;  
$results=pg\_query\_params($conn , $sql , $params );  
assert ( ’pg\_num\_rows( $results ) <= 1 /∗uniqueness violation in database ∗/ ’);

//Bail and reload the page if we didn’t find a user $row=pg\_fetch\_array( $results );

if (! $row){  
header("Location: $\_SERVER[PHP\_SELF]?msg=Unknown User"); die("User not found.");

}

//Does the password match?  
if (md5($\_POST[ ’password ’]) == $row[ ’password ’]){

session\_start ();  
$\_SESSION[ ’user\_name’]=$\_POST[ ’user\_name ’]; $\_SESSION[’user\_id’]=$row[’user\_id ’];  
$\_SESSION[ ’ user\_dir\_fs ’]=$FILE\_STORE . $row [ ’ user\_id ’ ] ; $\_SESSION[ ’ user\_dir\_url ’]=$FILE\_URL . $row [ ’ user\_id ’ ] ;

header("Location : profilePage .php");

die("Done loading user ."); }else{

header("Location: $\_SERVER[PHP\_SELF]?msg=Bad Password");

//  This leaks information about whether or not a user exists on the

//  system . The ease of use is a net positive , however .

//  This problem can be alleviated with rate limiting on the login. die("Bad password.");

2

die("Reloading login page."); }

?> <HTML> <HEAD>

<TITLE>Honeycomb Login</TITLE>  
<link href="include/yui /2.8.2 r1/build/fonts/fonts−min. css"

rel="stylesheet" type="text/css">  
<link href="include/yui/2.8.2r1/build/treeview/assets/skins/sam/treeview.css"

rel="stylesheet" type="text/css"> </HEAD>

<BODY>  
<table cellspacing="1" cellpadding="0" border="0"

id="shell" height="471" width="1168"> <tr height="50">

<td height="83" colspan="2" bgcolor="white"> <table title="Banner" id="banner" border="0">

<tr><td width="1195"></a></td></tr> </table>

<img src="images/Honeycomb Logo 2. jpg"  
width="1221" height="137" alt="Honeycomb Logo 2">

</tr>  
<tr height="200">

</td>

<td width="219"><a href="register .php">"Register a new account</a></td> <td width="946" bgcolor="white">

<tr height="200">  
<td width="260" bgcolor="white">

<table id="navigation" title="Navigation" border="0">

<tr><td>  
<table border="0" cellspacing="0" cellpadding="0">

<tr>

</tr>

<link href="include/yui /2.8.2 r1/build/fonts/fonts−min. css" rel="stylesheet" type="text/css">

<link href="include/yui/2.8.2r1/build/treeview/assets/skins/sam/treeview.css" rel="stylesheet" type="text/css">

<script src="include/yui/2.8.2r1/build/yahoo−dom−event/yahoo−dom−event. js" type="text/javascript"></script>

<script src="include/yui/2.8.2r1/build/treeview/treeview−min.js" type="text/javascript"></script>

<script type="text/xml"> <!−−  
<oa : widgets>

<oa : widget wid="2444522" binding="#OAWidget" /> </oa : widgets>

3

−−> </script>

</td> </tr>

<tr height="200">  
<td width="260" bgcolor="white">

<table id="navigation" title="Navigation" border="0">

</table>

</td><td width="397" bgcolor="white">

<table title="Content" id="content" border="0"> <tr><td>

</td></tr> </table>

</td> </tr>

</table> <style>

} </style>

<?php

. ygtvitem {  
font−family : Verdana ,

Geneva , sans−s e r i f ;

<tr><td><form action="<?php echo $\_SERVER[ ’PHP\_SELF’ ] ; ?>" method="post" id="login">

<table> <tr>

<td>User Name:</td>

<td><input name="user\_name" type="text"></td> </tr>

<tr>  
<td>Password:</td>  
<td><input name="password" type="password"></td>

</tr> <tr>

<td><input name="login" type="hidden"</td>

<td><input value="Login" type="submit"></td> </tr>

<tr> <td></td>

<td>  
if (array\_key\_exists(’msg’, $\_GET)){

4

echo "$\_GET[msg]"; }

</td> </tr>

?>

</tr> </table>

<style>

} </style>

</table> </BODY> </HTML>

</table> </form></td></tr>

</table>  
<img src="images/bigbox . jpg" width="432" height="432"> </td> </td>

. ygtvitem {  
font−family : Verdana ,

Geneva ,

sans−s e r i f ;

3 Main

<?php  
//Display a user profile , list the files , and do most of the heavy lifting

?> <HTML> <HEAD>

<TITLE>User </HEAD>

<BODY> <?php

P r o f i l e </TITLE>

r e q u i r e \_ o n c e ( ’ i n c l u d e / s e s s i o n . php ’ ) ;  
r e q u i r e \_ o n c e ( ’ i n c l u d e / c o n f . php ’ ) ;  
// Where the file is going to be placed $target\_path = "$FILE\_STORE/$\_SESSION[user\_name]/";

?> <?php

i f

( array\_key\_exists ( ’ uploadedfile ’ , $\_FILES)){  
/∗ Add the original filename to our target path.

Result is "uploads/filename . extension" ∗/  
$target\_path = $target\_path . basename( $\_FILES[’uploadedfile ’][ ’name’]);

if (move\_uploaded\_file($\_FILES[ ’ uploadedfile ’][ ’tmp\_name’] , $target\_path)) { echo "The file ". basename( $\_FILES[’uploadedfile ’][ ’name’]).  
" has been uploaded ";

} else{  
trigger\_error("Failed to save file as $target\_path");  
echo "There was an error uploading the file , please try again !";

} }

i f

} ?>

<?php  
p r i n t

?>

( array\_key\_exists ( ’ delete ’ , $\_POST)){ foreach ($\_POST[ ’ filelist ’] as $myfile){

if (strstr(’/’, $myfile)){  
die("Please don’t be rude. That’s not a filename I gave you.");

}  
if (! unlink("$FILE\_STORE/$\_SESSION[user\_name]/$myfile")) {

} }

die("Unable to delete

$myfile .");

"Welcome

$\_SESSION [ user\_name ] " ;

<link href="include/yui /2.8.2 r1/build/fonts/fonts−min. css" rel="stylesheet" type="text/css">

<link href="include/yui/2.8.2r1/build/treeview/assets/skins/sam/treeview.css" rel="stylesheet" type="text/css">

<script src="include/yui/2.8.2r1/build/yahoo−dom−event/yahoo−dom−event. js" type="text/javascript"></script>

6

<script src="include/yui/2.8.2r1/build/treeview/treeview−min.js" type="text/javascript"></script>

<script type="text/xml"> <!−−  
<oa : widgets>

<oa : widget wid="2444522" binding="#OAWidget" /> </oa : widgets>

−−>  
</script>  
<table cellspacing="1" cellpadding="0" border="0"

bgcolor="black" id="shell" height="639" width="1168"> <tr height="50">

<td height="83" colspan="2" bgcolor="white"> <img src="images/Honeycomb Logo 2. jpg"

width="1157" height="137" alt="Honeycomb Logo 2"></td>

</tr>  
<tr height="200">

<td width="216" bgcolor="white">  
<table id="navigation" title="Navigation" border="0">

<tr><td> </td> </tr> </table>

<img src="images/Side Bar Pics . jpg" width="216" height="864" /> </td> <td width="959" bgcolor="white">

<form enctype="multipart/form−data"  
action="<?php echo "$\_SERVER[PHP\_SELF]";? >" method="POST">

<table title="FileList" id="FileList" border="0"> <?php  
$mydir = "$FILE\_STORE/$\_SESSION[user\_name]"; $myurl = "$FILE\_URL/$\_SESSION[user\_name]";

$file\_dir = opendir($mydir);  
if (! $file\_dir) die("Can’t see directory ."); $id = 0;  
while($myfile = readdir($file\_dir)){

if ($myfile == "." || $myfile == "..") continue;

printf (’<tr><td><input type="checkbox" value="%s" name="filelist[]"/><a href=" }

?>  
<tr><td><input type="submit" name=’delete ’ value="Delete Files" /></td></tr>

</table>  
<table title="Content" id="content" border="0">

<tr>  
<td><input type="hidden" name="MAX\_FILE\_SIZE" value="100000" />

Choose a file to upload: </td>  
<td><input name="uploadedfile" type="file" /></td>

</tr> <tr>

<td><input type="submit" value="Upload File" /></td> </tr>

<tr>  
<td><a href="<?php echo "$URL\_BASE/login .php?logout "; ?>">Logout</a></td>

</tr> </table>

</form></td><td width="4" bgcolor="white"> <form enctype="multipart/form−data"

7

</form> </td>

</tr> </table>

<style>  
. ygtvitem {

font−family : Verdana ,

} </style>

</BODY> </HTML>

Geneva , sans−s e r i f ;

action="<?php echo "$\_SERVER[PHP\_SELF]";? >" method="POST">

4 Registration

<?php /∗

∗ ∗ ∗ ∗ ∗ ∗/

register .php  
−Lee Hall Sat 08 Sep 2012 06:05:55 PM EDT

Allow the user to create a new account, and verify that account

// Yes, this makes a connection when we don’t neccesarily need it. It’s better // than repeating the include inside multiple branches , though  
r e q u i r e \_ o n c e ( ’ i n c l u d e / c o n f . php ’ ) ;

$MAIL\_SUBJECT="[Honeycomb] Registration ";  
$MAIL\_TEXT="Please click the following link to finish the registration" .

" process . $URL\_BASE$\_SERVER[PHP\_SELF]? verify =";

//Is there a user trying to register?  
if (array\_key\_exists(’register ’, $\_POST)){

i f

}

}

//Get user info from database  
$token=md5(mt\_rand() . $\_POST[ ’user\_name ’]);  
$sql="INSERT INTO users (user\_name , password , email , auth\_hash) VALUES

($1, $2, $3, $4);";  
$params=array ($\_POST[ ’ user\_name ’ ] , md5($\_POST[ ’ password ’ ] ) ,

$\_POST[ ’ email ’ ] , $token ) ; $results=pg\_query\_params($conn , $sql , $params );  
if (!$results || pg\_affected\_rows($results) != 1) {

//There has to be a more elegant way to do this. $error=pg\_last\_error ();  
if (strpos($error , ’user\_name\_key’) !== False){

$msg="That username is already in use .";  
} else if (strpos($error , ’email\_key ’) !== False) {

$msg="That email address is already in use."; }else{

die("Unknown error ."); }

header("Location: $\_SERVER[PHP\_SELF]?msg=$msg"); }else{

( ! array\_key\_exists ( ’ user\_name ’ , $\_POST) | | ! array\_key\_exists ( ’ password ’ , $\_POST) | | !array\_key\_exists(’email’, $\_POST) ){

die ("User , password or email not set . How did you

//Make sure that no one ’ s doing anything tricksey //since it gets used as a filename for the moment

get here ?");

with the username ,

i f ( ! ctype\_alnum ($\_POST[ ’ user\_name ’ ] ) ) {  
$msg="Please only use letters and numbers in the username.";

header("Location: $\_SERVER[PHP\_SELF]?msg=$msg"); die("Illegal characters in username.");

9

mail ($\_POST[ ’ email ’ ] , $MAIL\_SUBJECT, $MAIL\_TEXT . $token ) ; die("Account created succesfully ." .

"Please follow the directions in your email."); }

}  
//Is someone trying to get their account verified?

i f

( array\_key\_exists ( ’ verify ’ , $\_GET)){  
$sql="SELECT user\_id,user\_name FROM users WHERE auth\_hash=$1;"; $params=array($\_GET[’verify ’]);  
$results=pg\_query\_params($conn , $sql , $params );  
if (pg\_num\_rows($results) != 1){

$msg="Account not found . " ;  
header("Location: $\_SERVER[PHP\_SELF]?msg=$msg"); die($msg);

}  
$row=pg\_fetch\_array( $results );

// We can still fail after this , but we’ve finished authentication , so it ’s // safe to authorize the user here.  
$\_SESSION[’user\_id’]=$row[’user\_id ’];  
$\_SESSION[ ’user\_name’]=$row[ ’user\_name ’];

if (!mkdir("$FILE\_STORE/$\_SESSION[user\_name]")) { die("Unable to create user file system.");

}  
$sql="UPDATE users SET auth\_hash=NULL WHERE user\_id=$1 ;"; $params=array($row[’user\_id ’]); $results=pg\_query\_params($conn , $sql , $params );  
if (!$results || pg\_affected\_rows($results) != 1){

die("Database error verifying user ."); }

header("Location : profilePage .php");

die("User validated ."); }

?>

<HTML>  
<HEAD>  
<TITLE>Register </TITLE>  
</HEAD>  
<BODY>  
<link href="include/yui /2.8.2 r1/build/fonts/fonts−min. css"

rel="stylesheet" type="text/css">  
<link href="include/yui/2.8.2r1/build/treeview/assets/skins/sam/treeview.css"

rel="stylesheet" type="text/css">  
<script src="include/yui/2.8.2r1/build/yahoo−dom−event/yahoo−dom−event. js"

type="text/javascript"></script>  
<script src="include/yui/2.8.2r1/build/treeview/treeview−min.js"

type="text/javascript"></script> <script type="text/xml">

<!−−  
<oa : widgets>

<oa : widget wid="2444522" binding="#OAWidget" /> 10

</oa : widgets> −−>

</script>  
<table cellspacing="1" cellpadding="0" border="0"

id="shell" height="639" width="1168"> <tr height="50">

<td height="83" colspan="2" bgcolor="white"> <table title="Banner" id="banner" border="0">

<tr>  
<td width="1195">

<img

</td> </tr>

</table> </td>

src="images/Honeycomb Logo 2. jpg"  
width="1157" height="137" alt="Honeycomb Logo 2" />

</tr>  
<tr height="200">

<td width="218">  
<img src="images/Side Bar Pics . jpg" alt="sidebanner"

width="216" height="864" /> </td>

<td>  
<form action="<?php echo $\_SERVER[ ’PHP\_SELF’ ] ; ?>"

method="post" id="register">  
<table title="Content" id="content" border="0">

<tr>  
<td>Username : </td>  
<td><input type="text" name="user\_name"></td>

</tr> <tr>

<td>Password : </td>

<td><input </tr>

<tr> <td>Email : <td><input

</tr> <tr>

<td><input

<td><input </tr>

<tr> <td></td> <td>

type="password" name="password"></td>

</td>  
type="text" name="email"></td>

type="hidden" name="register"></td> type="submit" value="Register"></td>

<?php  
if (array\_key\_exists(’msg’, $\_GET)){

echo "$\_GET[msg]"; }

?>

</td> </tr>

</table> </form>

11

</td> </tr>

</table> <style>

. ygtvitem { font−family : Verdana ,

} </style>

</BODY> </HTML>

Geneva ,

sans−s e r i f ;

5 Schema

/∗  
∗ honeycomb . s q l  
∗ −Lee Hall Wed 05 Sep 2012 10:55:06 PM EDT ∗/

SET ROLE honeycomb; BEGIN ;

DROP TABLE IF EXISTS u s e r s CASCADE; CREATE TABLE users (

);

user\_id user\_name password auth\_hash email first\_name last\_name quota

SERIAL PRIMARY KEY, varchar UNIQUE NOT NULL, varchar ,  
varchar UNIQUE,  
varchar UNIQUE NOT NULL, varchar ,  
varchar ,  
integer DEFAULT 0

COMMENT ON COLUMN users . password IS ’ sha1 hash of password ’ ; COMMENT ON COLUMN users . auth\_hash IS

’onetime key used for registration and lost passwords’; COMMIT;

Correspondence

\*\*\*PDF ONLY\*\*\*

Complete PowerPoints

