Course Hand-in pages:
Set-up, Inner-working & Utilization

Created by: Raghuveer Mukkamalla
Graphics by: Min Xiao
Supervised by: Dr Katy Borner
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>Basic set-up</td>
<td>7</td>
</tr>
<tr>
<td>Adding new students</td>
<td>8</td>
</tr>
<tr>
<td>Setting-up new projects</td>
<td>8</td>
</tr>
<tr>
<td>Setting up comments</td>
<td>9</td>
</tr>
<tr>
<td>Counting comments</td>
<td>10</td>
</tr>
<tr>
<td>Future work</td>
<td>10</td>
</tr>
<tr>
<td>Appendix</td>
<td>10</td>
</tr>
<tr>
<td>login.cgi</td>
<td>10</td>
</tr>
<tr>
<td>browse.cgi</td>
<td>11</td>
</tr>
<tr>
<td>submit.cgi</td>
<td>11</td>
</tr>
<tr>
<td>comments.cgi</td>
<td>12</td>
</tr>
<tr>
<td>studentsetup.cgi</td>
<td>12</td>
</tr>
<tr>
<td>projectsetup.cgi</td>
<td>12</td>
</tr>
<tr>
<td>commentcount.cgi</td>
<td>13</td>
</tr>
</tbody>
</table>
Introduction

Please read the document completely before starting anything.

1. The hand-in pages serve 3 purposes.
   i) To let the instructor organize students and projects of his/her class
   ii) To let students of a class submit their project work online
   iii) To let students browse projects and leave comments

2. The basic assumption is that all the student projects are links to the websites they created on the WWW. If the projects are offline, they cannot be submitted through this system.

3. This is how the pages work, in brief:

   The class instructor assigns the students login ids and passwords. These passwords are NOT the IU network passwords and the students need to be told that these passwords are visible to the instructor. They can be assigned/changed by the instructor alone and it has been kept this way deliberately. If the students are given the freedom to change, they'd probably change the assigned password to their network password and this is not desirable, as the instructor can see it.

   Once the login is successful, the students are presented with 2 options: Browse or Submit.
Browse:

If the user hits the browse button, s/he is taken to a page where s/he sees the list of all projects with the images of the students.

The list of projects page is a table with the students name, picture and his links in each row and the list of projects as the column headings. If the project is not due (and has not been set-up) or if the student/team has not yet submitted their work, the words 'No entry' will appear in the column. In case a project has been submitted, an image, which is a link, will be present in each cell. Clicking this would open the project page in a new window.

Submit:

To use the submit button, the user needs to enter a valid link and select his/her project. The only constraint on the link is that it should start with 'http://'. If the submission is successful, s/he is taken to a page with a list of all projects (same as the browse screen with the updated link s/he submitted).

Commenting:

If the user needs to make a comment or read the existing comments, s/he can click on the Comments button under each link. This would take him to a page where s/he can see the existing comments (with details of who, what and when) and a small text box where s/he can enter his own comments and click the submit button. An empty text box is not considered for submission; any other comments are valid. Each submitted comment appears immediately and is also sent as a mail to the relevant person/team.
While displaying comments, the link for which a person commented is also mentioned. The reason is that, students may submit interim links in the course of their work and the users need to know which comments pertain to which page.

4. Katy commented on this page at 4/30, 6:47P: 
Well-organized, designed, and informative site – fun and educational to browse and study. Good navigation guidance through a large and diverse set of information.
See project printout for more comments.

4. An admin can enter students for the class and set-up project teams and also check the number of comments each person entered (indicating class participation). Upon logging in, s/he will be presented a screen with these options and s/he can choose the required option.

5. To take a quick tour before dipping into the details, go to: http://ella.slis.indiana.edu/classes/handin/handin-tutorial/login.html

This page has been set-up with students added; they also submitted 'presentation' and 'final project'. The final project is a team submission and it has comments associated with it.

You have 3 options to login:
i) guest (no password but you can only browse and you will not be able to see the provision to submit links or to make comments).

ii) Student (Id: ‘rdoshi’ passwd: ‘handin’). I trust you not to do any submissions though!

iii) Admin (Id: ‘admin’ for passwd: ‘tolkienisgod’). Here you can see the existing students, modify the project set-up and see the number of comments made by each user).

6. Along with the README file, find the following in this packet. The scripts/data files have been set-up for the example given above.

- login.html (html file)
- cgi
  - browse.cgi
  - commentcount.cgi
  - comments.cgi
  - login.cgi
  - projectsetup.cgi
  - studentsetup.cgi
  - submit.cgi
- data
  - presentation.txt
  - project4.txt
  - project4comments.txt
  - students.txt
- images
  - bullet.gif
  - header_r1_c1.jpg
  - header_r1_c2.jpg
  - header_r2_c2.jpg
  - header_r2_c3.jpg
  - header_r3_c1.jpg
  - header_r3_c2.jpg
  - spacer.gif
  - team1.jpg
  - team2.jpg
  - team3.jpg
  - team4.jpg
  - team5.jpg
  (Plus all the student images)

- imagestore
  (This is a repository created by Min Xiao from which you can choose from 3 sets of images needed to set up the pages for three different classes.)
Basic set-up

1. The set-up should work under any server; on Ella, under your www directory, the directory structure would be as follows:

   www
   - cgi   (directory for all cgi scripts)
   - data  (directory for all related data files)
   - images (directory for all images for the webpages as well & pictures of the students)
   - login.html (file for initial login)

2. Copy all the cgi scripts and images from the same directories as given with this attachment (see part 1).

3. After copying the files into the above structure, you have to make the scripts Unix compatible as well as executables. Ensure the following:

   i) If you make change to the cgi files using any editor other than notepad and upload it to a Unix server, you have to do a 'dos2unix' on that file before running it; for example, if you made changes to browse.cgi and uploaded it, then at the command prompt, do: $dos2unix browse.cgi browse.cgi

   ii) You also have to make the cgi file an executable. To make browse.cgi an executable, you would do this:
       $chmod u+x browse.cgi

   iii) You must understand Unix directory/file permissions and set them up appropriately. As a hint, all the cgi scripts should have 755 and the students.txt must have 644 on it.

   iv) Take care of the case-sensitivity of the filenames. As a rule of thumb, use small letters for everything.

   v) Make sure that all the files into which entries have to be made have write permission on them.

4. I have retained the headings 'L542 Human Computer Interaction' so that it will be easy for you to change. Go through the login page and all the cgi scripts and change it to the class name. Also change my email id (rmukkama) to yours everywhere.

5. In the login.cgi script, in case of invalid entry, the user is given a link to go back to the main handin page. Check that the link is valid.

6. As you go through the following steps, remember that extra testing is always beneficial. After setting up anything, use a student id and go through all the steps yourself.
7. Also, you would do well to look at how the data is being stored currently to understand the process better. Look at the data attachments while you study the code. You have to understand the code, otherwise it will be tough to troubleshoot or make changes.

8. Finally, make sure you change the current admin password (hard-coded in login.cgi) to your own. You have to change it inside login.cgi. This way, if someone else is also setting-up the handin pages, they should not be able to guess your admin password.

**Adding new students**

1. Make sure you have gone through the Basic set-up.

2. The related script is studentsetup.cgi. Inside the script, change the class name and the link.
   ```
   $link = 'http://ella.slis.indiana.edu/~rmukkama/login.html';
   (The above link should point to where the students should login).
   $class = 'L542';
   (This should be the class name).
   ```

3. Create a text file called students.txt under the data directory. Make sure the permission on this is 644.

4. Go to the login.html page on www. Login as ('admin', 'tolkienisgod') and you come to a page giving you three options. Click on the 'Setup students' button. Here follow the instructions. After you add each student, a mail will be sent to him/her with the valid link and password.

   Supervisor refers to the class teacher; you would know the list of auditors from her. Everyone else is a student. Carefully assign the types.

5. If you make any mistake or need to delete a user who dropped a class, just enter the id and press delete.

6. The students must be asked to send their picture as 'jpg' (and 'jpg' only!). After they do, name the picture as 'userid.jpg' (where userid is that students IU id) and store it under images.

**Setting-up new projects**

1. Make sure you have gone through the Basic set-up and the Adding new students sections.

2. If it is a single person project, go to step 6.
3. For a team project, you have to know the list of people present in each project and their team number. Otherwise, you cannot make entries. Even two people working together are a team.

4. After you get the required information, create an empty file with the correct title under data directory.

5. The related script is projectsetup.cgi. In this, change all references to L542. Currently, it is set-up for just project 4. Look at the code mentioning project4.txt and make additions with a suitable else statement.

Now go to the admin page, click on project set-up and make entries following the notes there.

Go to step 7.

6. For a single person project, you just need to create an empty text file under the data directory. For example, for project 1a, create project1a.txt under data.

7. Make the additional option in login.cgi in the pull-down menu so that users can submit valid links for this project.

8. Make relevant changes in browse.cgi and submit.cgi. Either the single_person_project or the team_project sub-section would get an additional file open and an else statement with the existing projects. In display_all_tables, one 'No entry' section will go away. Just replicate the existing code for either presentation.txt or project4.txt. Be careful about the variable names.

**Setting up comments**

![Image of a form with options to submit a new link and a dropdown menu for selecting project. The form includes fields for New Link and For project. Below the form, there is a note to Got questions? Mail Raghu.](image-url)
1. Check the existing code in submit.cgi or browse.cgi and check how information is passed to the comments.cgi script. Under each display in the tables you are creating, there has to be a comment button.

2. Create an empty text file under the data directory. For example for collecting comments for project 2, you should create project2comments.txt.

3. Change the comments.cgi script and add the relevant sub-section while looking at how project4comments is set-up. Pay attention, the comments go out as mails too. Currently the send mail section is in a loop because all the team members have to get the relevant mail. If you are setting up Comments for a single person project, then the looping has to be removed.

**Counting comments**

1. Go to admin page and click on the comment count button.

2. The script to be changed is commentcount.cgi. Currently only project4 comments are being counted. The required additions should be simple to comprehend.

**Future work**

*Note:* At first glance, the cgi scripts, 'browse.cgi' and 'submit.cgi' can be merged into one script with a slight change in logic in the previous screen. However, I made it this way deliberately, because if any new functionality were added to either, it would be easy to make changes.

1) Using JavaScript features, the back button logic can be handled better.

2) All the functionality can be implemented using a relational database model and SQL. At the time of development, however, I felt using a database to be overkill. Also text storage for data makes the application work much faster.

3) When the users submit their comments on a project, a radio button can be provided through which they can rank the work on a scale of 1-5.

4) Using the available data, participation can be visualized.

**Appendix**

**The Scripts**

**login.cgi**

This gets the login parameters from the login.html file.
If the user is guest, password is not checked. If the user is ‘admin’ and the password is ‘tolkienisgod’, then the login is fine. For all other id’s, the students.txt file is opened and the userid, passwords are verified. For each user, the type is fetched. Within the main subroutine, the variable $caller takes care of the condition when a user returns to this page after browsing the inner pages.

For all invalid accesses, an error message is displayed giving a link back to the main page.

Depending on the user type, the appropriate display is done; for guest and supervisor, only the browse button is shown. For admin, only set-up buttons are shown. For all other users, both the browse, submit and comment buttons are shown.

**browse.cgi**

For a valid entry, (a browser id and browser first name is passed from the previous screen), data from each project file is read into a separate hash. (In the current script, presentation and project4 data are being read.) For a single person project the data is stored in the form of user id, link and submission time. For a team project, the data is stored as user id, team number, project title, link and submission time.

Now, the student file is read and for each entry, the appropriate link is taken from the hash tables and displayed in html. If links do not exist or if a project has not been set-up, ‘No entry’ should appear in the table.

If a project has comments associated with it (project 4 in the current example), a form button should be present and the correct variables need to be associated with it to pass to the comments.cgi script.

**submit.cgi**

An entry is valid if a userid is passed from the previous screen. If the user did not submit a link, or did not select a project or has submitted a link that does not start with http://, an error message is displayed.

Depending on whether the project is a single person creation or a team based one, the appropriate sub-routine is called. This is hard-coded in the present script; presentation is a single person project and anything else is a team project. As you set-up new projects, you should add suitable conditions to this ‘if’ logic.

For a single person project, the data file is written to directly with the current time. If the student already submitted a link, it is overwritten.

For a team project, the team the student belongs to is determined first. The new link is updated for all students belonging to that team. The rest of the entries are written as is.
A message that submission was successful is displayed and the display process is the same as in the browse script.

**comments.cgi**

For a valid access, it is checked which project number is passed for comments and then that project comments file is read and the passed link is matched with any existing link already present. If yes, then these are formatted and displayed.

If the user is not a guest, a comment box appears at the end where the users can enter their feedback and press ‘submit’. The same script is called and the logic in the first section takes care of an empty submission.

For each valid submission, the line breaks are modified appropriately while being written to the data file and the comments are also sent as a mail to the owner of the link. For each team project, the send mail option is invoked in a loop after determining the team number so that mails are sent to all members of a team. The current script takes care of project4.

The key for a team based project is the team number and for a single person project, the key is the user-id of the owner of the link.

**studentsetup.cgi**

This is for admin only. After setting up each student, every valid submission invokes the same script and adds the user to the students.txt file (which must exist) and also sends him a mail giving the correct link to login. Upon deletion, a user is informed as well.

At the end all the existing students are displayed in a table as shown below.

**projectsetup.cgi**

This is for admin only and allows the setting up of teams.

To add: Up to 6 members can be present in each team. After checking that all the students are valid for this class, the relevant project file is read. All the students that do not match the entered students are written out into an array.

Now all the entered students are made stringed using the updated title and are appended to the above array.

Finally, the array is read separately and all students with the entered team number are updated with the new title. This ensures that a student always gets the latest title which is shown as a tool tip in the browse/submit screens.
To delete an entry, the entered user id is checked for validity. Then the team number is determined from the file and the file is written out skipping those entries with the entered team number.

Finally, the selected project details are displayed in the display routine. Currently only project 4 with 5 entries is being displayed and you can add to this.

**commentcount.cgi**

All the project comment files are opened and for each student, the number of comments is totaled (by comparing his user-id with the user-id in the comments file).

Currently only project 4 comments are being counted and you have to modify it to include others.