

Abstract:

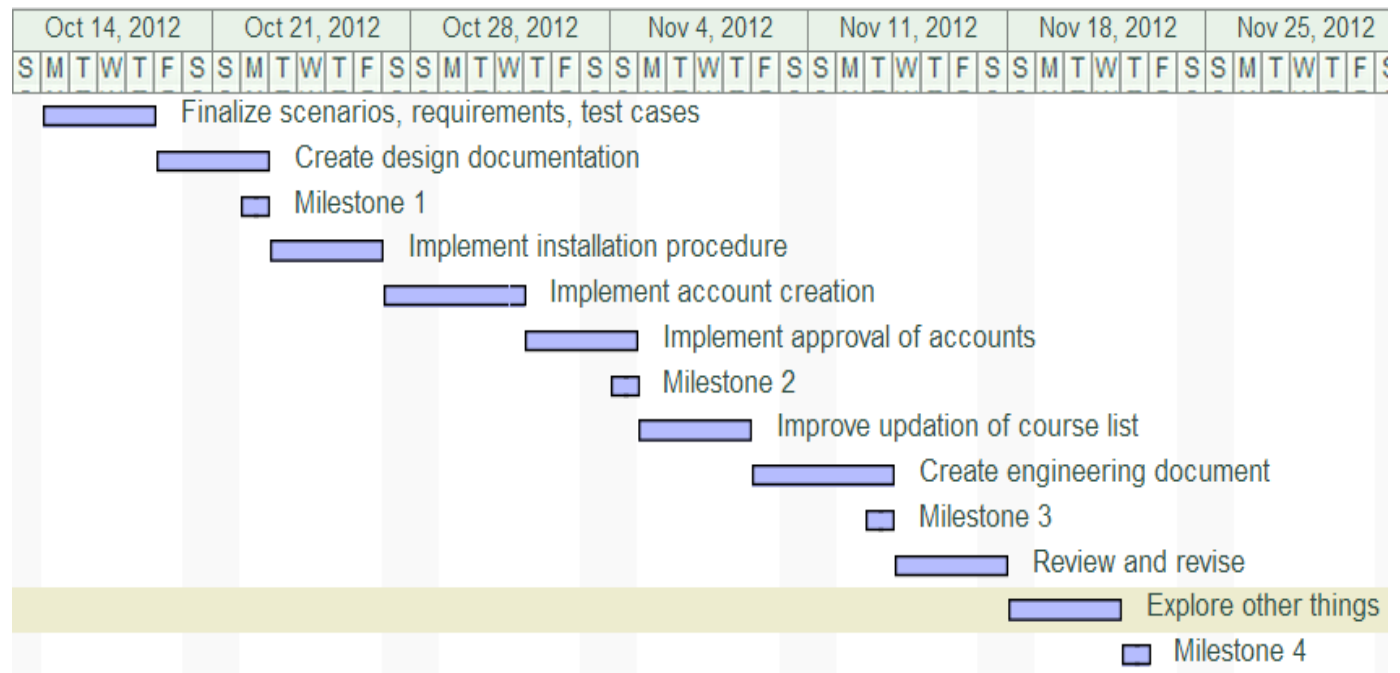
This project is intended for the CS411 student to develop his software engineering skills by working with an existing software product. The goals of this project is for the student to learn the source code of the product and be able to document it with explanations, as well as incorporating a few fixes themselves. Completing this project will allow the student to gain insight into the software development field and have experience with what is to be expected in possible future careers.

Introduction:

This project focuses on the software Quartz, which is a web application written in PHP that allows professors to create and manage their academic website. Its purpose is to create a consistent feel to all professors that use the application, because all their websites will be formatted the same way. Also the application lets the professors freely update their website after setting up an account with their email and password and getting approval from the database admin. The website will contain their photo, name, title, contact information, office hours, a short biography, course information, and supplemental information such as research, awards, and personal information. This application is very useful, except one of the main problems is the installation procedure, which is not user-friendly for the admin who manages the professors' accounts, and improving it is the main goal of this project. The end product is a fully function application with a user-friendly installation.

Schedule:

Gantt Chart



Time Estimation

Finalize scenarios, requirements, test cases		
• Edit scenarios		1 hour
• Edit requirements		1 hour
• Edit test cases		1 hour
Create design documentation		
• Gather information		1 hour
• Draft it		1 hour
• Type it up		2 hours
Implement installation procedure		
• Write php code		1 hour
• Debug		2 hours
Implement account creation		
• Write php code		1 hour
• Debug		2 hours
Implement approval of accounts		
• Write php code		1 hour
• Debug		2 hours
Improve updation of course list		

<ul style="list-style-type: none"> • Write php code • Debug 	1 hour 2 hours
Create engineering document <ul style="list-style-type: none"> • Gather information • Draft it • Type it up 	1 hours 2 hours 2 hours
Review and revise <ul style="list-style-type: none"> • Read each portion • Editing and finalizing 	2 hours 2 hours
Explore other things <ul style="list-style-type: none"> • Find other possibilities 	1 hour

Total time = 25 hours

Unknowns:

- The number of bugs I will run into and the amount of time it will take to fix.
- Conflicts with future projects and assignments from other classes.

Possible Risks:

- Becoming ill and unable to work on project
- Hard drive failure, therefore losing all project work

Safety factor- 1.5x

Final time= around 37 hours

Gantt chart schedule takes unknowns and risks into account, allocating excessive time for debugging and assuming you can only spend 1-2 hours on the project every day.

Installation Procedure Design:

As previously mentioned, the installation procedure that Quartz had by default was not very user friendly for the admin. The following is a rundown of the process in a small scenario:

After downloading the Quartz code and extracting it into the Apache server directory, Admin Adam then edits the database variables of the code to match his server host and folder names. Then opens up the MySQL console and manually create the database, grant himself admin privileges to the database, and set a password, all with MySQL commands. Not only that, but he also runs the a MySQL script in phpMyAdmin to create tables for his account and professor accounts inside the database before he can login to the administration panel on the website manager.

As you can see, this process isn't very efficient even though it's only done once for each database admin. Thus an improvement was made which lets the admin install Quartz in one step besides downloading and placing it in the proper folder and the following is a scenario for the improved installation procedure:

After downloading the Quartz code and extracting it into the Apache server directory, Admin Adam opens up his browser to run a PHP script that displays a form. After filling out the form with the MySQL server name, the server's root password, the desired admin username and password, the desired name of the database, the url web server along with a port number if non-default, the name of the subfolder where Quartz was placed, and finally whether his web development platform is able to send emails via the PHP function mail(), Adam hits the 'submit' button and can now login to the administration panel on the website manager.

Functional Requirements and Test Cases:

1. The user has the ability to run a PHP script on his browser
 - a. Open up the browser and enter the script name. Verify that it opens.
 - b. If it fails to open, test is negative.
2. The user has the ability to enter required information in the script and submit it
 - a. After script opens, verify that all necessary information can be entered.
 - b. If a single piece of information cannot be entered, test is negative.
3. The system has the ability to display confirmation of installation.
 - a. After entering information and pressing 'submit', verify that confirmation text appears.
 - b. If confirmation text does not appear, test is negative.
4. The user has the ability to login after running the script.
 - a. After confirmation text appears, verify that user can login by entering 'username'@bu.edu and password at login screen.
 - b. If login fails, test is negative.

Nonfunctional Requirements and Test Cases:

1. Quartz is installed by a php script and ONLY a php script.
 - a. If the test cases for the functional requirements above all return positive without additional steps, then this requirement test case will also be positive.
2. Quartz's installation procedure works on all platforms (Windows, Mac, Linux)
 - a. Run installation script on Windows, Mac, and Linux
 - b. If installation fails on any test case in any of the platforms, test if negative.

Installation Structure:

Inside the Quartz Code is a file named 'install.php'. This script contains several things, including a html form for the admin to enter necessary information in order to setup Quartz.

After the 'submit' button is pressed, the information is stored in php variables. The variables and their values are then written to a new file 'dbvars.php', which is then placed in the same directory and used for running most of Quartz.

The rest of the script are solely MySQL queries which does all the server connecting, creates a new database, grant a user privileges and sets a password, and sets up all the tables inside the database.

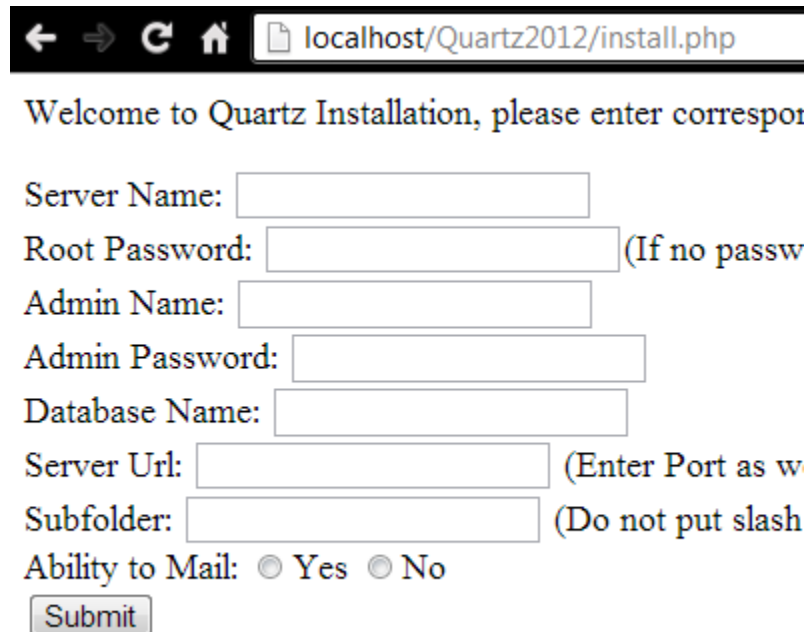
All these steps are done within seconds and within one script, which is much faster than the previous process.

Side Note: I did not consider approving professor accounts and sending them emails as part of the "installation" process, hence these will be covered in the appendices.

Installation instructions:

1. Download Quartz source code.
2. Extract source code into WAMP/MAMP/LAMP server directory.
3. Open up your internet browser after WAMP/MAMP/LAMP server is online.
4. Run installation script by entering server url / name of subfolder Quartz was extracted into / install.php into the address bar. A form should appear.

For example:



The screenshot shows a web browser window with the address bar displaying 'localhost/Quartz2012/install.php'. The page content includes a welcome message and a form with the following fields and options:

- Server Name:
- Root Password: (If no passw
- Admin Name:
- Admin Password:
- Database Name:
- Server Url: (Enter Port as w
- Subfolder: (Do not put slash
- Ability to Mail: ☐ Yes ☐ No
-

5. Complete the WHOLE form following the requested formats and click submit.
6. Make sure all fields are entered accurately, in particular the password fields as they are blocked from view once typed. But if a mistake was made, one can always re-submit the form with a different database name
7. If everything went well, a message similar to the following should display, otherwise it will return errors:

Subfolder: (Do not put slash in t

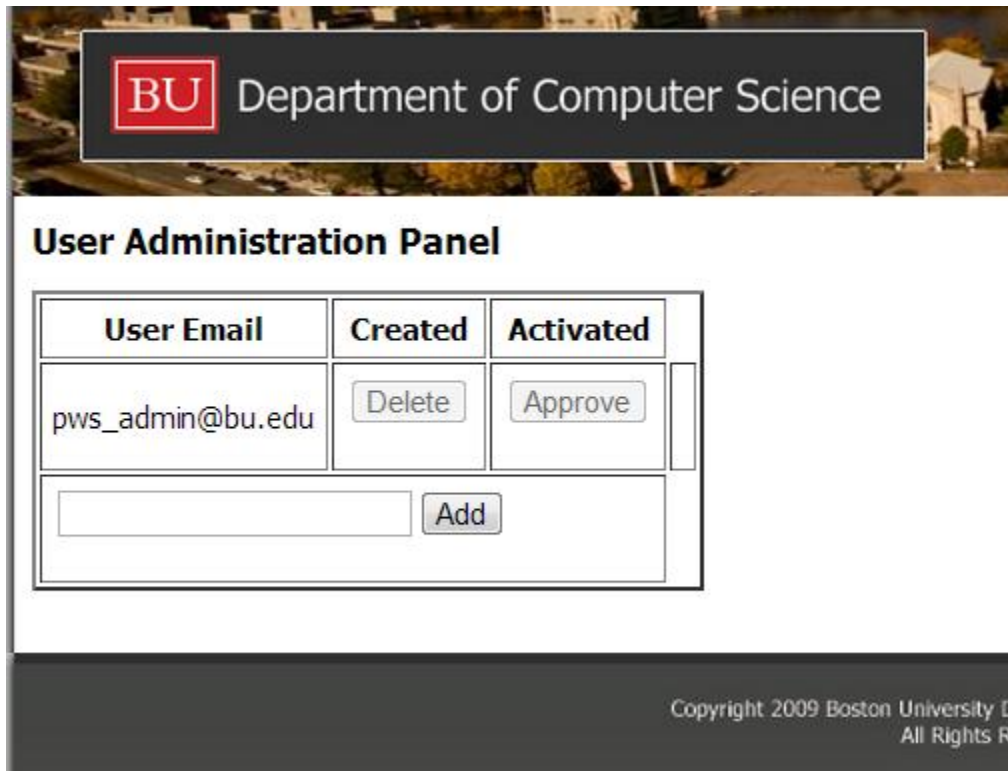
Ability to Mail: ☐ Yes ☐ No

Database created.

PHP successfully connected to profwebsitedb2012!

Tables created. Installation Complete.

8. You should now be able to login with the admin account and access this panel:



The screenshot shows a web interface for the Department of Computer Science at Boston University. At the top is a banner with the BU logo and the text "Department of Computer Science". Below this is the title "User Administration Panel". The main content area contains a table with three columns: "User Email", "Created", and "Activated". The first row shows the email "pws_admin@bu.edu" with "Delete" and "Approve" buttons in the "Created" and "Activated" columns respectively. Below the table is an "Add" button next to an empty text input field. At the bottom right, there is a footer with the text "Copyright 2009 Boston University De" and "All Rights Re".

User Email	Created	Activated
pws_admin@bu.edu	<input type="button" value="Delete"/>	<input type="button" value="Approve"/>
<input type="text"/> <input type="button" value="Add"/>		

Copyright 2009 Boston University De
All Rights Re

Quartz System Design:

A programmer will find the following 8 questions and answers useful for understanding the original Quartz:

Question 1: What kind of system is Quartz?

Quartz is a web-based application.

Question 2: How does one install and run Quartz (for the old system)?

Step 1: Have WAMP/MAMP/LAMP or similar web development platform installed.

Step 2: Obtain a copy of Quartz source code from the web.

Step 3: Extract this code into server directory of your development platform.

Step 4: Edit variable values in 'dbvars.php' to correspond to user's platform.

Step 5: Run MySQL console to create and setup database and its tables.

Step 6: Run Quartz by entering server and folder name into address bar of browser.

Question 3: What does Quartz do?

Quartz allows a professor to create and manage his own website for academic uses and allows admins to manage approval of these professor accounts.

Questions 4,5, and 6: What is the structure of Quartz and what are its components and the interfaces to each component?

Since Quartz is written in PHP with a MySQL back end, it's structured by nested php files, along with a set of images and tables in a MySQL database to store information, both of which are used to display the professor's website.

The php files can be categorized by the modules they require to run. Some files, such as 'header.php', 'footer.php', and 'dbvars.php' are used in almost every file since they are needed to display interfaces and access the database tables. There is a template folder which every professor's directory will be based on.

The 'index.php' file in the main folder is the core of the code and the website and determines what is displayed for most of the following modules.

The account creation module displays the registration screens, using the files, 'register.php', 'regform.php' and the database tables.

Create Account

Name:
Please enter your first name.

BU e-mail:
Please enter your BU email address eg. john@bu.edu.


Confirm e-mail:
Confirm your email address.

Password:
Your password should have atleast 2 special characters.

Confirm password:
Re-type your password.

BU ID number:
Please enter your 9 digit BUID.

The account management module displays the admin panel for the admin for approving professor accounts, using the files 'approve.php', 'manage.php', and the database tables.

 Department of Computer Science

User Administration Panel

User Email	Created	Activated
pws_admin@bu.edu	<input type="button" value="Delete"/>	<input type="button" value="Approve"/>
<input type="text"/>	<input type="button" value="Add"/>	

Copyright 2009 Boston University Department of Computer Science
All Rights Reserved

The content editing module displays the interface in which the professor edits how his website is displays, using the files 'manage.php', and the database tables.

Apply

Picture:

Picture Placeholder

Your picture should be approximately 230x350 pixels.

Choose a picture to upload:

Choose File

 No file chosen

Upload Picture

General Information:

Tip : Use HTML tags for special formatting.

Display Name :

Title. Name M. Last

Job Title :

Job Title Here

Office Address :

#XXX Street Name, BID-RMN
 Boston, MA 0

Tel :

(XXX) XXX-XXXX

Fax :

(XXX) XXX-XXXX

Office hours :

Day TT:TT - TT:TT
 Day TT:TT - TT:TT

 (Use this format [DAY TT:TT - TT:TT])

The login module uses the files 'login.php' and 'forgot.php', and 'reset.php' which displays the login screen.

Login

Username:

Password:

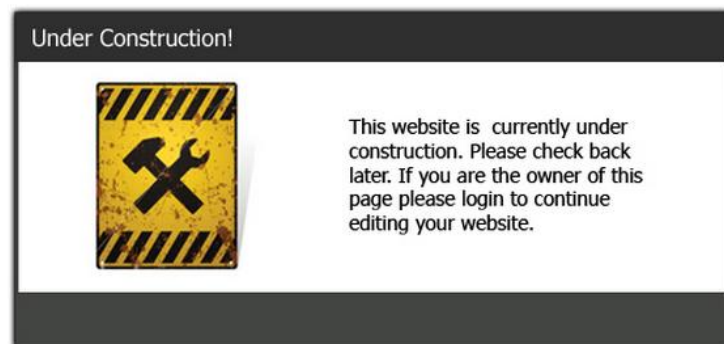
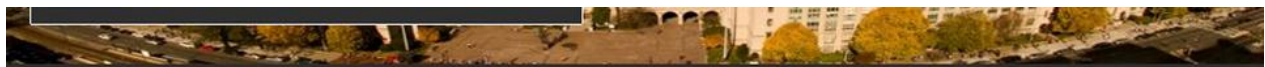
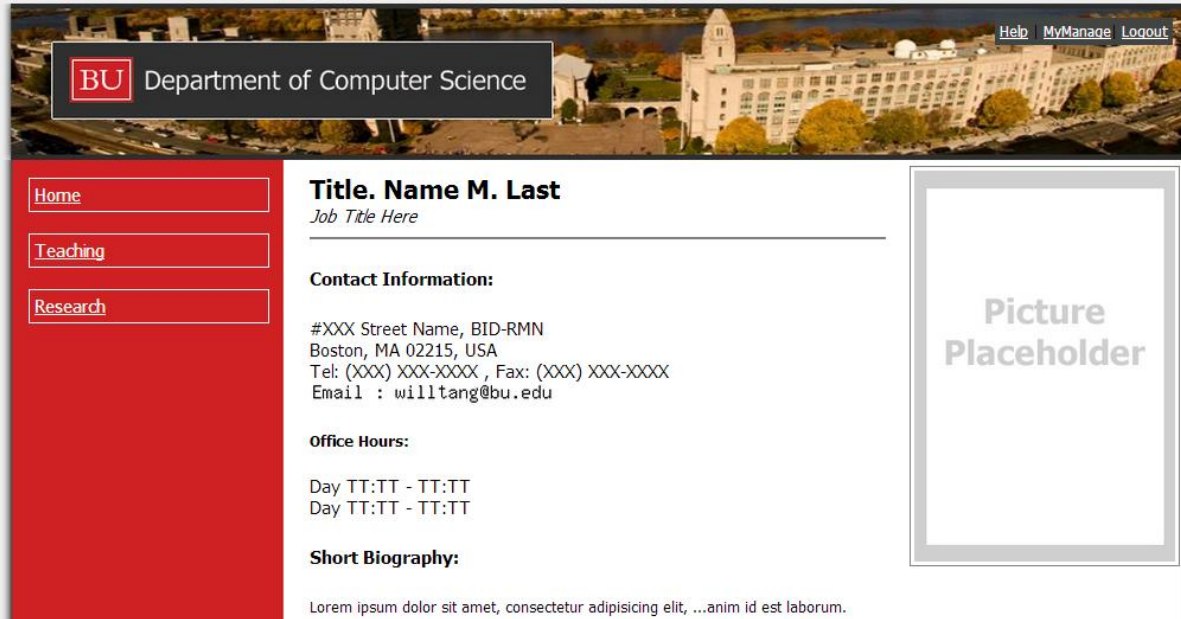
[Forgot Password?](#)

[Create a new account.](#)

Reset

Login

The content viewing module either displays the professors website, fetching data from the database tables and using 'index.php' from the professor's folder or the construction page with 'construction.php' if his website is not ready yet, or under management. The email is fetched from separate file inside the professor's folder, which will be explained later.



Question 7: What data is involved and how is it organized?

Almost all of the data involved is stored in the MySQL database's three tables: loginset, nlogin, and webdata. Image files are stored in the professor's folder in Quartz.

loginset contains the email of an account and whether the account is approved.

nLogin contains the email of an account, a hashed password, the name and BU ID of the professor, and whether or not the account is active.

webdata contains most of the data which is displayed on the webpage including the professor's email, name, biography, phone and fax numbers, office address and time available, whether the professor's website is selected to be online, the professor's research and courses, the professor's awards and projects, personal information, etc.

Question 8: How does feature x work?

(1)

Issue: How does login work?

Solution: Quartz has a login form in the login module where user enters username and password, if either is incorrect after checking with the content editing module and the databases, will display an error.

Link: [Quartz2012.0001] (in login.php)

(2)

Issue: How is invalid login information determined to be invalid?

Solution: Quartz first checks if either username or password boxes are empty, then checks if username and password BOTH matches what is stored in the database.

Link: [Quartz2012.0002] (in index.php)

(3)

Issue: What happens when a user requests an account?

Solution: After checking if the username is already used, Quartz adds the user's credentials into the database tables, after hashing the password with MD5, then copies the template into a new directory created for the user. The improved version also adds the user to the content editing module for admin's.

Link: [Quartz2012.0003] (in register.php)

(4)

Issue: What happens when an admin approves an account?

Solution: A email will be sent to the user either through the php server or through the admin's email client (depending on the admin's settings) with the activation link, which the user has to click.

Link: [Quartz2012.0004] (in manage.php)

(5)

Issue: How does the system know if someone is logged in, and how does it know whether the person logged in is an admin or a professor?

Solution: There are three user types. Quartz checks for the type of the current user, if there is no type or the first type, then no one is logged in. The second type means its a professor. The third type means its an admin.

Link: [Quartz2012.0005] (in index.php and manage.php)

(6)

Issue: How does the system know whether to display the content editing UI module, or the content viewing UI module?

Solution: Quartz shows the content editing module by default. If the use clicks on MySite on the header, it will display the content viewing module. Clicking on MyManage will bring him back.



Link: [Quartz2012.0006] (in header.php)

(7)

Issue: How does the system store information about the courses that a professor teaches?

Solution: The name, link, and description is concatenated and stored as a single string and inserted into the corresponding table row MySQL database.

Link: [Quartz2012.0007] (in manage.php)

(8)

Issue: How does the system add a new course to the professor's course list?

Solution: Quartz simply adds another string to the table row of the corresponding MySQL database.

Link: [Quartz2012.0008] (in manage.php)

(9)

Issue: How does the system store and display the information displayed on a professor's "Personal" page?

Solution: Quartz stores the information as a string in the content editing module by adding it to the corresponding row of the professor. In content viewing module the string is then fetched and displayed.

Link: [Quartz2012.0009] (in manage.php and template's index.php)

(10)

Issue: How does the system make the email address displayed on the "Home" page of the content viewing UI module not selectable as text?

Solution: Quartz writes the text into a .gif image, and displays the image instead of the email text.

Title. Name M. Last

Job Title Here

Contact Information:

#XXX Street Name, BID-RMN
Boston, MA 02215, USA
Tel: (XXX) XXX-XXXX, Fax: (XXX) XXX-XXXX
Email : willtang@bu.edu

Office Hours:

Email : willtang@bu.edu

Link: [Quartz2012.0010] (in template's index.php and email_img.php)

Appendices:

Log of group discussions and meetings:

I created a facebook group which includes only the four people in my team. We asked each other occasional questions.

Log of work on project:

November 24 - 6 hours - Installation script

November 25 - 2 hours - Account creation improvements

November 26 - 3 hours - Fixing installation script and account creation bugs

November 27 - 2 hours - Documentation of abstract, introduction, schedule.

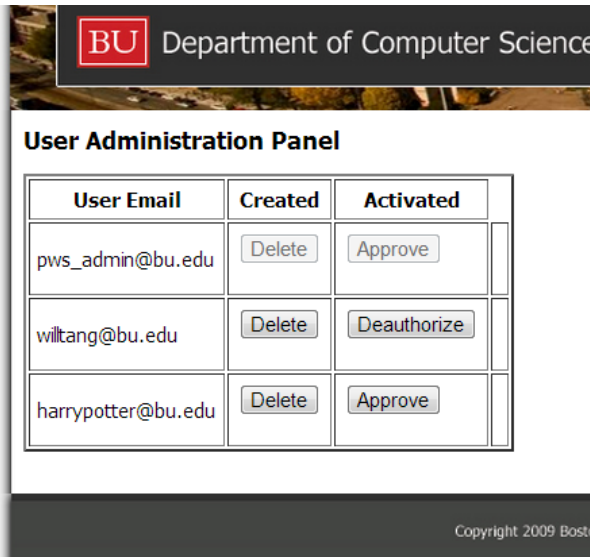
November 28 - 3 hours - Documentation of installation procedure

November 29 - 3 hours - Documentation of system design

November 30 - 2 hours - Documentation revising and finalization.

Completed improvements:

- Changed all implicit start of php code `<?` into explicit start `<?php` to avoid bugs with some versions of the Apache server
- The administrator no longer has to add the professor's email to the admin panel, it will be there once a professor registers. This was done by adding one line to 'register.php'. I also removed the ability to add a user to avoid confusion.
 - Link: [Quartz2012.0011] (in register.php)



- Instead of copying and pasting the text hidden under the banner in the admin panel to obtain activation url, the admin merely has to click Approve and a new page will pop up depending on whether or not the php server can use the mail() function.

- If it can, the activation email is immediately sent and 'link.php' is shown:

Activation email has been sent to the professor.

[Return to Admin panel](#)

- If it cannot a hyperlink is created to allow the admin to email the user with his own email client. In this case, 'link2.php' is shown:

[Send Approval](#)

[Return to Admin panel](#)

- Your email client should now pop up and look something like this, hit Send to send:

User Administration Panel

User Email	Created	Activated	
pws_admin@bu.edu	<input type="button" value="Delete"/>	<input type="button" value="Approve"/>	<input type="checkbox"/>
wiltang@bu.edu	<input type="button" value="Delete"/>	<input type="button" value="Deauthorize"/>	<input type="checkbox"/>
harrypotter@bu.edu	<input type="button" value="Delete"/>	<input type="button" value="Deauthorize"/>	<input type="checkbox"/>

Link: [Quartz2012.0012] (in manage.php)