Team 1

Team 1 was given the task of creating a file upload system that was meant to in it’s first week of development fulfil a list of 17 requirements, the main requirements were:

* Login in system
* Upload a file (as lecturer)
* Approve/Reject/Delete file (as moderator)
* Download the files
* View approved files as student
* Log out

Team 1 first meeting started with two main questions: “How?” and “Where to?”. Firstly, considered using Git as Combines both as has commands to upload data and an online database to store it. The second solution was to use raw PHP code to upload file to a cloud/host server of choice. Considering both possibilities we agreed using PHP would be using less shortcuts without being that much harder and also that it would allow more flexibility when it comes to where the code should be uploaded.

Initial solutions proposed to solve the issue using this solution was to host server could be on a virtual machine (here Virtual box). We would then potentially use an image of this virtual server to put it on a real live one.

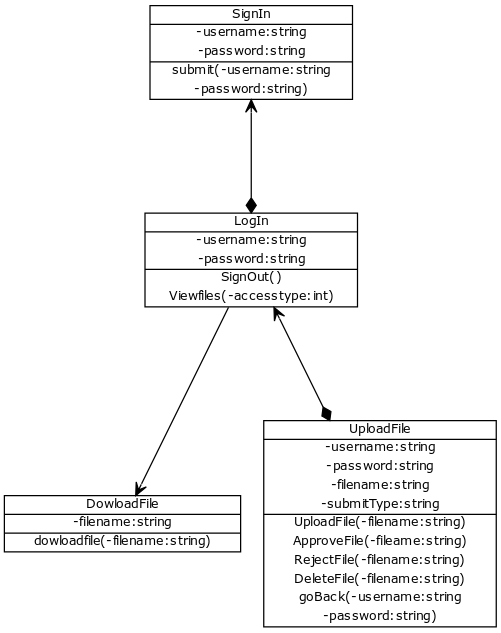
However, following difficulties manipulating the virtual machines it was decided it would be simpler to use a local server using (WAMP) and the project server made available by Cardiff University. The only rule that remained was that the system had to remain hosted in a way where it would be easy to move it on to another server. Logins were decided to be simply hardcode as although it was implied logins would come from a database at some point no database was put in place. We also decided that to approve/reject files they would simply be moved into a different registry all on the server and thus retrieved differently to keep things ordered (ex Student would only retrieve files from approved folder).

When it came to be solving the issue, during the initial team meeting it was decided that the upload file system would be separated into three main parts. A login system, an upload system and finally a system that would retrieve files from the server. For the login system we decided for 3 main login types, a student that could view approved files, a lecturer who could upload files and a moderator who could approved/reject uploaded files and delete them. As the retrieving part of the system needed a functioning upload system before work could be done on it the initial work was split into two parts and therefore the team was split into two.

We also noticed that many of our requirements were the same if not slight variations of the same process. For instance, the file upload was fairly similar to the move file function we would have to implement, and the download functionality was to be used several times. For this reason, we decided to have two main functions stored in a different php file. An upload function that would handle initial uploads, file transfers and deletions (eventually version control as well although this wasn’t initially planned) according to input arguments upon being called. And a download function that could be recycled whenever it was required.

The main page thus interacts with these two functions that allow all of the system functionality.

UML:

s

Robin and Will worked on creating a login system. The first assumption that was made was that usernames and passwords where to be retrieved from a database later in development. Therefore, username and password are initially hardcoded as a dictionary. We created a simple login page that was made up of a form submitting inputs for name and password when “Log In” is pressed. The form had for action another PHP page “login.php”.

This login page therefore had to display differently according to what kind of login type (lecturer, moderator) has logged in. This is done by comparing inputs to the hard-coded usernames and password. If username and password match, we set a variable access\_type that then links to one of three main IF statements in the code each displaying and allowing different things. If username or password are wrong a submit form with only a submit button “try again” comes up sending us back to the Sign In page.

Sean and Andrew worked on the upload system. They set out to design a system that allows lecturers to upload their proposed project files, data about the files would be stored in a database so that in the future we can sort through many files using SQL queries. What we first designed was a simpler, multipurpose system that would allow the user to select any local PDF file that would then be stored on the site’s server. We first made use of HTML’s inbuilt file selector tool (by creating an input element with type “file”). This then passes the file data through to a PHP script, via a post method, that can validate if the file is valid. Initially, for a file to be valid it would have to be a new, non-null PDF file, maximum 5MB size. Then, we made use of PHP’s inbuilt “move\_uploaded\_file” method, which stores the file into a specified directory on the server. One issue we ran into using this implementation of our system was that files could not be uploaded if there was already an existing file named the same thing. This was not necessarily a bad thing because it would stop the old file’s path being completely overwritten by the new ones. However, after having a follow up meeting with our group leader we came to a decision that we would design a way of incrementing file names of new files, which was then worked on by Robin.

The team met again and implemented the upload function as part of the lecturer functionality. The next part was then to retrieve the information, make the files downloadable. We used php to read through the list of files in the upload directory on the server. These could then be presented to the user as a list of links to download each file using PHP’s inbuilt “readfile” method.

One of the issues we encountered was the since the upload was stored in a different php file we needed to find a way to go back to the previous page after uploading without losing log in details. After considering the use of cookies we decided the simplest was for now was to keep posting the username and password as part of the upload forms. Therefore, when uploading a file, the form would also upload username and password and pressing the “go back/ upload more” submit button on the upload page would sent this information back too. Once this was done we worked on separating the files into three states “uploaded”, “approved” and “rejected”. We decided the simplest way of doing so was so sort them into three directories on the server. Code was therefore added to the upload function. Copying the file to the demanded directory and deleting it from its previous one. Retrieve function was also modified simply retrieving from three directories instead of one and displaying it accordingly into the three categories. Code was also added so that student would see files in the “approved” directory.

A suggestion was made to have an automated system differentiate files uploaded by the same name. An extension “(V\_x)” was added to a file if it was already uploaded and approved with x being the version of the file. X is the latest version of the file uploaded so if a file of the same name is uploaded and approved several times the latest one uploaded will have the biggest X. If a file is uploaded and has the same name as a file that isn’t yet approved, then this file overwrites it’s previous to keep things clear with the moderator.

An issue with this arose as if the moderator deleted one of the “middle version files” the code would no longer work correctly has the gap between for instance V4 and V6 would be filed as V5 by the latest uploaded file. Therefore, to fix this it was decided that every time an approved version file is deleted all the version superior to it are decremented to keep everything in the correct order.

The program is thus flexible while satisfying all the basics requirements: a login system that links to an upload system that allow users to approve/reject files. Files are then retrieved and displayed differently according to login and are also downloadable. It is built thinking forwards and is set to be the base for the entire project as code will be recycled to satisfy other criteria’s such as marking. The team worked together well meeting up often to brainstorm and organise tasks, then splitting the tasks to sub teams or individuals according to said team/individual strengths was an efficient way of solving the problem. Thinking forward the program’s lack in security is its main flaw and with the newly implemented database it is the program’s next step forward in development. The login system will also need to be linked to more functionalities than the upload system, for instance the group making and the upload system will be slightly modified and recycled to be put to use in a grading system.s