

# Assignment 7 - PHP mini project

As part of project you have to design an website for sharing files. Various features that can be provided are categorized into headings:

**Required features :** You must implement these features in your mini project

**Recommended optional features :** You should try to include as many of these (preferably all) features in your design as possible. All of these are simple enough to implement and have huge practical value.

**Optional features :** You can try to include these features based on time / interest. These are not so important but are easy to implement. Try to implement at least selected few among these.

**Optional challenge features :** These are also optional but are very hard to implement. Implementing even one among these should be fun.

## Required features

- New users can register on website.
- Users can Login and upload files.
- Users can download files they have uploaded.
- For each uploaded file user can choose which other users are allowed to download files. For this list of all registered users is shown to each user while giving access.
- Users can download files uploaded by other users, for which the uploading user has given permission to particular user to download.

  For example if user $A$ uploads file `personal.txt` and gives permission to user $B$ to download file `personal.txt` then user $B$ should see option of downloading file `personal.txt` uploaded by user $A$ after he/she logs in.

## Recommended optional features

- Display strength of typed password using javascript while user is typing / choosing some password during registration. Here strength can be calculated trivially by looking at password length, whether it has digits and small letters and capital letters, whether it has special characters, number of different characters etc.

  Based on password strength word (weak, medium, strong, very strong) can be displayed or you can also display password strength meter using javascript.

- Verify email address of new users by sending them email with verification links. Users should be able to Login only after verifying their email address. It should not be possible for user to guess verification link without looking at / receiving email, else there is no point in having verification links.

- Ensure that file can only be downloaded after users login and if they have permission to download the file. No one should be able to download file just because they have URL to download the file. If this is not done registration / login processes are useless if users have links to files.

- Give options for removing already given access to a user for particular file. This is basic functionality user would expect. If permissions once granted remain forever then usability is severely affected.

- Give option for removing uploaded file. Users should have options of deleting files else useless files will occupy disk space and would also cause confusion.

- Do not store password of user as it is in database. Try to store hash of password, so that even if database is lost users passwords are still safe for a short duration. This is very important requirement these days.

- Implement quota for each user. You can keep fixed quota for all users or you can have separate quota limit for each user. This is important to ensure that same user cannot eat entire server space by uploading large number of files.

- Implement a admin interface via which any user can be disabled or deleted. Admin should also be able to delete any uploaded file / download file uploaded by any user. There is no limit to amount of power

that can be given to admin user. For example admin user can be given option of approving users for whom email verification is still pending. Admin user can be given option for modifying users personal details name, etc. Admin can reset / change password of any user.

This would be design exercise in choosing what options should be provided for admin. You would have to think which options are more important than others and implement important / useful ones before others. You also have to think on how admin user will change his/her password, will admin user also use same login interface as other users, can admin user also upload and share files with users, etc.

- Provide remember me functionality on login page so that if users use the website frequently from personal computer they do not have to login every time.

- Sanitize all output being displayed on screen to ensure that you are not printing javascript code injected by users. Similarly sanitize all queries going to database to ensure you are protected by SQL injection attacks.

- Implement basic image captchas so that users have to enter words / numbers displayed in image to prove that they are humans. This is very important to defeat automated registrations / file uploads / posts etc.

- Transfer sensitive information like password over HTTPS. At the same time avoid making entire website HTTPS which would have sever performance penalty.

- Do anti-virus scanning of uploaded file. You can use 'clamav' or any other anti-virus of your choice for scanning uploaded files for viruses.

- Implement basic logging so that we can trace which file was upload by which user on what date / time from which IP address. We should also be able to tell which users downloaded the uploaded file and when and from which IP address. We should also log when files were deleted, by which user (admin / uploading user) and from which IP address.

- Users should not be able to achieve anything malicious by executing library files directly by typing their URL. For example if you have mentioned most supporting code in '`common.php`' file then opening URL similar to http://.../common.php should not cause any security problems.

## Optional features

- Implement different types of users:
  - One type of users who can just login and download files shared by other users.
  - One type of users who cannot share files but can only download files uploaded by themselves.
  - One type of admin users who can just download files uploaded by any user and nothing else
  - One type of admin users who can change user details, delete files uploaded by any user, reset user password, set user quota etc.
  - Super admin users who can create and delete admin users. These type of users can change type of users from normal users to admin and vice-versa.
  - Backup users. These can download entire website data (database dump) and files for backup. These users cannot do any change in data, they just download existing data as it is for backup.
- Implement detailed logging for accountability, traceability and legal obligations. The logging mentioned in 'Recommended optional features' section above is very basic / minimal logging. Detailed logging can include information about file like file size, mime type, output of running '`file`' command on given file.

  Every user registration, email verification, login, logout should be logged. Reset of password, disabling of user, deletion of user etc. should be logged. Basically every action howsoever small must be logged.

  When logging is done this extensively using databases, logging becomes expensive both CPU and storage wise. Since no one would be querying log regularly and logs are never updated (else how can they be trusted) we can use files for logging instead of using databases. But if files are used how parallel logs from so many differnet users will be handled without causing race conditions, dead-locks, file corruption, loss of logs, etc.

  Whether you use tables or flat files, logs also need rotation. One cannot keep adding logs to same table / same file forever. Log files are rotated automatically periodically. Again period of log rotation, every day / every week can be configurable parameter.

If logs are stored on same machine as website then they can be destroyed / altered if website is compromised / server is hacked. Hence logs should ideally go to some other server where logs can only be appended but never modified / deleted by website.

- Implement better test for detecting humans using tests like copy / paste, identify cats, solve equations, fill in the blank, distorted image captchas in comparison to simple image captchas.

- Integrate your website with Single-Sign on used by IIIT courses / mess websites so that there is mapping between your site users and IIIT email address. If someone has already logged in on courses website, he/she should not be required to login on your website again.

  You can also provide login via Gmail, Facebook, Twitter etc. This is called openID where login of user on one website is used to identify and associate user on other websites. Look at OAuth protocol to understand how users must permit which website can collect which details about particular user from main authenticating website like Gmail, Facebook etc.

- Verify mobile numbers by sending SMS with verification code. Mobiles are more and more important these days. Not being able to write applications that can be run on mobile or which cannot send alerts to mobile is huge limitation.

  You need SMS gateway to send SMS. Since that costs you can either share a gateway among group of friends or you can use websites like 160by2.com, way2sms.com. When using website you would have to register with your own mobile number. It is good idea to have a spare mobile for such registrations as once you give mobile number you are likely to receive many advertisement SMS on given number irrespective of privacy policy of given website.

  Just registering on way2sms.com or 160by2.com is not enough. You would have to check which website allows sending of SMS after login without any captcha (human verification) and then write code to send SMS by logging in on website.

  *I wrote code for obtaining AIEEE details of student - Name, Fathers name, Branch, Category, Physics marks, Chemistry marks, Maths marks, AIR, etc. based on AIEEE roll number. That program is used during admissions for obtaining details of each student to verify whether student gave correct rank or not.*

*This was possible only because AIEEE website does not verifies human users when someone tries to check results for given roll number. It is considerably fun to automate such tasks as AIEEE result website is not designed to be used by other programs. So one has to parse the HTML output generated by result website and extract relevant fields.*

*Hence writing code that can send SMS via free SMS websites is possible provided such websites do not verify whether user is human or not.*

*Once you can learn to use functionality provided by websites, you would have whole range of things that you can do by code. There are websites that can convert currency as per latest exchange rates, tell current time in any time zone, convert time specified in one time zone to other, do complex maths (wolframalpha.com), compare given image with similar images on web (Google image search), directions from one location to another (Google maps, wikimapia) etc.*

*You would be able to do all this by just using the functionality provided by one website using your website. Offcourse you would have to read terms and conditions for websites that you are using to ensure that you are not breaking law in the process.*

## Optional challenge features

- Most popular websites provide APIs so that users can write applications which work with those websites. For example facebook will provide API so that user can login via code, check posts, get friends list, get status updates etc. These APIs are then used by various application developers to provide facebook application on mobile phones, desktop clients etc.

  Try to provide API for your website. This is again design exercise. What features will you provide using your APIs? How will authentication be handled? Which languages will you support? What will be format for exchange of information? (as in this case two programs would be interaction with each other without human intervention. Remember XML.)

- Provide support for plugins for your site. Popular websites like facebook and popular browsers like firefox, popular email clients like thunderbird, popular chat clients like pidgin (Internet messenger) provide plugin support. These plugins allow users to extend functionality of website, browser, chat client, email client without modifying code of ac-

tual website but by creating plugins using specified interface for writing plugins.

If you provide plugin support, users can add features like forum, wiki, etc. to your website. Users can add options for viewing pdf files directly in browser. They can add options for sending documents / excel files to Google documents, etc. Rather than original developers trying to add all these features, having support for plugins encourages shared / parallel development, leading to development of thousands of applications for successful products in very short time.

## Rules and guidelines

This is individual project. Team work is not allowed. You are not suppose to borrow code / database designs from colleagues. You are free to borrow code from Internet, class examples with proper references mentioned in code comments and also in final report.

You have to submit project report explaining features that you have implemented. You can optionally mention problems faced and how you resolved them. You can explain various design choices you have made and try to justify why are those choices good.

Apart from report you have to submit entire PHP code, database designs in SQL and read-me file that helps in understanding various files and how to use code.

You are free to use any database of your choice SQLite, MySQL or PostgreSQL. You can try to justify your choice in report.

Code should be properly indented and commented. You should use good variable names. Code must be divided into various files and functions. Try to reuse code by implementing common functions in library files and by including those functions everywhere (similar to mytrash problem implemented using shell scripts). Proper documentation comments that can be used by doxygen or any other automated documentation generated tool should be present.

You can implement features not suggested in this document, if you feel they are important / useful. You can also use technologies like Flash, AJAX, HTML 5 to some extent while doing the project, but most / majority of work should be done using PHP. You can use any other language of your choice for providing support for anti-virus scanning, SMS, email etc. features.

Code should not be hard-coded for some particular database name, user name etc. Most of such parameters should be defined in one single PHP file.

You can name this file 'config.php' or something similar. Avoid magic values everywhere, you should use constants / parameters as much as possible.

# Delieverables

1. Report with justification of design choices
2. Read-me with detailed information on various files and on how to use code
3. SQL statements for table / database initialization. You can get away with this if your code can deal with empty databases.
4. Well commented and indented code with good variable names and no magic values.
5. Doxygen generated HTML documentation of project code.

# Food for thought

## Courses website

You can consider courses website as a very special case of above mini-project. In courses website all students can upload assignments and they can download only their own assignments. TAs / instructors can download assignments of any student.

TAs/instructors can uploaded resources / assignments which can be downloaded by any students, other TAs and instructors. Instructors / TAs can delete uploaded resources / assignment questions.

No-one can download any resource unless they have logged in. Just having link to resource is not enough to download it. One must login and have access before one can use links for download.

Courses website also has other features like threads / posts. It also helps in managing TAs. There is also option of mailing to TAs etc. But most of those are easy / small in comparison to assignment upload / download feature. Courses website developers also implemented backup systems to take regular backup of complete data / code.

## Generic CMS, Google docs or reservation websites

If above mini-project looks complex if try to implement all Optional features, then imagine complexity of generic CMS like jhoomla, moodle or drupal that allow creating of many different types of websites, just by change of configuration. These CMS also allow plugin of user code / pages at appropriate pages when functionality required cannot be achieved using CMS configuration. Most CMS also use URL rewrite to display friendlier URL and do caching to increase speeds.

Google docs allow parallel editing of documents on-line by many users from any ordinary browser. The user is not required to install any flash plugin or run java applet to be able to edit documents along with collaborates spread across the globe in parallel. One should be able to appreciate information exchange that allows such parallelism.

Reservation websites deal with on-line transactions from different banks, different payment methods like credit / debit cards, etc. They cater to thousands of users in parallel and have to be very accurate. The information should exist simultaneously in different geographic locations so that even if one entire city / state is out of power, suffering from natural calamity like earth quakes, under attack etc. reservation systems are up and running. Thus multiple consistent copies of reservation of thousands of tickets have to be maintained to provide reliable ticket reservation systems.

Imagine complexity of systems that are required to store unique information about each person, his / her voter ID etc. Also complexity of making identification / criminal databases on-line so that legal record of each individual can be tracked. Add complexity of associating this identification / personal records with various bank accounts / vehicle registration / land ownership registration records of same person.

One of the goals of this course is to make students understand complexity of existing already deployed systems, so that they know what is required to build such systems and hopefully contribute in development of such systems in future.