A REPORT ON: À LA SHARE

By: Abhilash Nanda
M.Tech CSE
Reg: 201105615

Introduction:

The goal of a CMS must be easing development, lowering learning curve, fast development by avoiding repeatition of boiler-plate codes. A la share is not a CMS, though it boasts of many basic features of a CMS like:

- MVC architecture (in this case VC and preferably TV)
- Custom urls
- Modular development
- Central settings
- Template inheritance (possible through ti.php)

Design Choices:

1. *TV architecture (T = V, V= C)*:
   Initially it was in plans to build a MTV (model-template-view) architecture with capabilities of a dynamic ORM and form handling. However, due to constraint of time, both have been dropped. Nonetheless, we are left with a TV architecture left with direct interation with the database via php. It would have always taken a lot of time, however once the foundation is laid, development becomes fast.

   Here we call the view as template and controller as view cause we go by the following analogy, controller decides what should be viewed i.e. each controller function is response for a different view. Thus, we call the file view and view as template cause that is the file that is displayed.
   View can be found in directory **“views”** and templates in directory **“templates”**.

2. *URL mapping*:
   Since we are rewriting the URL, we would like it to to mapped to particulr functions in the views. For this in php a nested array approach has been followed.
   This gives us the power to have custom, clean and beautiful urls.

3. *Central controling*:
   A settings file has been provided with many default fields which are most commonly needed for a project to start. Custom settings can also be written on it.

4. *Upload directory*:
   Each user gets its own dedicated upload directory which is named after its hashed username.

5. *File resolution*:
   Suppose two files are uploaded with the same name by the same user, this will result in a conflict which on OS level will either needed to be replaced or the current file upload cancelled. To prevent such a situation we have a function called **“unique_filename”** that appends a timestamp as a prefix to the filename.
   Function **“clean_name”** removes it.

6. *Password hashing*:
   All passwords are stored in the database only after hashing them with sha256 algorithm. It

prevents the cracker from guessing the password very easily.

7. *Filenames in URLs*:
   Any filename in the URL are passed as base64 encode.

UI choices:

1. *Login page*:
   Login page has two forms: login form and register form.
   This approach saves extra bandwidth and unnecessary clicks for users in order to signup.
   Registration details are as minimal as possible so as to prevent user from tiring out (people are lazy, none wants to fill forms).
   On mistake, proper errors are conveyed.

2. *Confirmation URL*:
   On proper registration, a confirmation URL is sent to the users email ID. On clicking the URL, account of the user is activated. This process prevents submission of fake email IDs.

3. *First login*:
   On first login user is faced with a Profile form so as to gather some information about the user.

4. *Home page*:
   The interface is tabbed and the hidden tabs are not loaded until they are clicked. This makes the page fast and dynamic.
   Tabbed interface ensures user don't have to leave the home page to perform various tasks such as finding users and downloading shared files.

5. *Tiles display*:
   The files and users are displayed as tiles so as to make better use of space.
   The tiles in tab “My Files” represent various files the user has uploaded. On hovering on a tile, we are provided with three controls:
   1. Download
   2. Share
   3. Delete

   In these links, on clicking share, a modal dialog opens with all the users available. To share the file with a user, just click on the users tile.
   Blue - shared
   Grey – Not shared

   This interface reduces clutter and provides the user with a easy and intutive way to share his/her file.
   Same is with shared files, except we only get the download option on hover.

6. *Edit profile*:
   Click on the link displaying the users name, it will open a page to edit the profile.

From here:

Due to modular nature of the code various important features can be added into it with relative ease. Some like:

1. A plugin architecture that provides an additional tab for a new plugin.
2. A ORM that handles transactions with database and create tables on the fly.
3. A form library for better form handling.

As time passed:

Initially it took some pain to get settled with an architecture, but after some time, development speed became faster and leaner.
I believe, this framwork with some more coding can be used by anyone to create they own web app.

Conclusion:

We got a basic TV architecture that follows best programming practices to cut down development effort and separates the codes by logic.