Weekly Report

MOOC Prototype

Summer Internship – 2013

P. Divya Shree

Summary :

Week 1 :

Visited various MOOC websites that already exist (Udacity, Coursera, and edX) and explored their features. It is necessary to version control any developing software. Learned to use git and created an account in www.github.com. Since I was totally unaware of web programming, started learning HTML5 and JavaScript from www.w3schools.com. I implemented all the features I came across in examples simultaneously whenever possible. I rushed through CSS and JQuery.

Week 2 :

Since we will be using Django in building MOOC, I wanted to learn Django as soon as possible. Django is based on Python. I learned the basics of Python from www.tutorialspoint.com and started a course on Internet History and Security on Coursera. We discussed the basic layout of the template that will be used in MOOC. I was given the task to design a login page with proper validation. I went through several websites' login pages and decided to use *Bootstrap's* features in the design. After several changes, the final page with proper indentation and validation was ready.

Week 3 :

I started a self-study course on database management on Coursera. I designed 'register' and 'forgot password' pages. The register page has instant validation. I tried to implement a modal for forgot password but some unknown error came up every time. It requires to be fixed. I started learning django from tutorials given in www.docs.djangoproject.com . To implement login via google account, I will be using *Django Social Auth*.

MOOC(Massive Online Open Course) :

Introduction :

The two fundamental components of MOOC are **open access** and the **ability to support large scale enrollments**. These courses are typically pitched to college level learning without offering credit. In some cases, certification is available (usually at a small cost). Although the companies offering these courses point to enrollments in the millions, the average course completion rate is under 10%. Currently the three biggest players in the MOOC field are Udacity, Coursera, and edX.

Why MOOCs?

The appeal to MOOCs is obvious. If one wants to learn anything apart from his college curriculum, in his own style and at his own pace, MOOCs offer a low-stakes oppurtunity to do so. If the teaching method in one course does not match your style, you can always move to another offering. They can be started at any time and we can select a course that matches our pace. MOOCs are built on efficiency of scale, giving access to the teaching of a world class professor to thousands of students at once.

Features of a MOOC :

1. **Open Access to all :** Most of the MOOCs are open to all. Anyone can register free of cost but some of them ask you to pay a little for credit.
2. **Scalability :** The courses are designed for an indefinite number of students.
3. **Pre-programmed course of study and assessments**
4. **Access to world-class faculty :** The lectures, assessments and activities for a course – especially an online course – and the expertise of the professor behind the content isn't cheap and, in many cases, is unique to a particular university. A MOOC throws open the door of the professor's classroom, allowing him to teach more than just a few dozen students at a time.
5. **Credentialing Flexibility :** MOOCs aren't bound by traditional university credentialing – they can be offered with or without a certificate or "badge" indicating that a student has completed the course. The credential can be separate from the class itself.
6. **Discussion Forums :** This gives the user an oppurtunity to present his views, doubts before a community that comprises people coming from various parts of the world. This encourages healthy debates and discussions.
7. **Evaluation of Assignments and Exams :** If an assignment/exam cannot be computer graded then peer-to-peer evaluation is being employed which also helps students acquire skills apart from the course.
8. **Subtitles and indexed transcripts with each video.**
9. **Offer a different granularity :** MOOCs provide a means of offering smaller chunks of courses. There is less pressure on them to be a model for the full degree in a free system. They operate as smaller units of learning which may be aggregated, or may not, appealing to leisure learners and professionals.
10. **Unconventional Curriculum :** They offer courses that are not in the conventinal curriculum thus filling the void not met by many institutes.
11. **Experimental :** They allow universities to experiment with different ways of teaching and students to experiment with various subjects without impacting the actual core degree.
12. **Student Dependent :** The amount of knowledge one takes away from a MOOC depends on his/her own motivation. This varies from person to person. Very few (only about 10% of all those enrolled) are motivated enough to complete the entire course.

Issues and Challenges for MOOCS :

1. Requirement of a good internet connection.
2. Students must be self-motivated to complete the courses. Not many are.
3. Digital/Online literacy among all other than internet facility for a better outreach.
4. Its use in economically less-developed countries.
5. Better peer-to-peer grading.
6. Something needs to be incorporated into the courses to increase the percentage of people completing the courses.

Overview :

- **Not a threat :** The MOOCs are *“disruptive extensions”* rather than threats to traditional universities. MOOCs will not replace the existing campus-based education model, but represent a huge opportunity to create a completely new, and much larger market that universities couldn’t serve previously due to the physical limitations of the campus.
- **Capture Market Value :** Different business models will emerge for MOOCs in the future. The unbundling of learning, credentials, social interaction, facility access, and assessment will make it possible for institutions to establish new business models.
- **Keep-on learning :** The traditional education model needs to be transformed to a more iterative and on-demand model. MOOCs enable people to easily learn any new topics whenever needed, and to stay up to date during their entire career.
- **Customization :** MOOCs enable you to optimize the mix various educational offerings irrespective of location, time, age, and provider.
- **Meet and learn :**Learners will pair up locally in coffee shops or conference rooms, study the online courses together and have face-to-face discussions.

<u>Git</u> :

Introduction:

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. *Distributed Version Control Systems* (DVCS) are better than *Centralized Version Control Systems* (CVCS) since clients don’t just check out the latest snapshot of the files: they fully mirror the repository. Thus if any server dies, and these systems were collaborating via it, any of the client repositories can be copied back up to the server to restore it. Git is a DVCS.

Advantages of git over other DVCS :

1. Every time you commit, if files have not changed, Git doesn’t store the file again - just a link to the previous identical file it has already stored.
2. Almost every operation is local.
3. It has a lot of integrity.
4. It only adds data. Therefore, every task is undo-able.

A quick guide to use git :

- To install git in ubuntu, use command *sudo apt-get install git-core.*
- Go to the directory where you want the git repository and use the command – *git init*. This will create a .git file in the directory where all your files will be saved.
- Create the files that you want to add and add them using *git add file_name*.
- After adding all the files to git, commit by using *git commit* and add an appropriate commit message.
- To connect your local repository to your GitHub account, you will need to set a remote for your repository and push your commits to it using the below given commands.
  git remote add origin https://github.com/username/Hello-World.git
  # Creates a remote named "origin" pointing at your GitHub repository
  git push origin master
  #Sends your commits in the "master" branch to GitHub
- Now if you look at your repository on GitHub, you will see your files have been added to it.
- To clone a repository into your directory, use the command *git clone required_link.*
- To change proxy settings in git, go to *.gitconfig file in* home directory and add:

```
[http]
        proxy= http://username:password@proxy-server:port-number
```

Python :

Introduction :

Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes. Python combines remarkable power with very clear syntax. It has interfaces to many system calls and libraries, as well as to various window systems, and is extensible in C or C++. It is also usable as an extension language for applications that need a programmable interface. Finally, Python is portable: it runs on many Unix variants, on the Mac, and on PCs under MS-DOS, Windows, Windows NT, and OS/2.

Features :

1. It can be used as a scripting language or can be converted to byte-code.
2. Very high-level dynamic data types and enables dynamic type checking.
3. It can be very easily integrated with many other programming languages.
4. It supports both functional and object-oriented programming.
5. Easy to learn.
6. Easy to read.
7. Easy to maintain.
8. Interactive mode : Enables interactive testing and debugging of code.
9. It has a very broad standard library.
10. It provides interfaces to almost all the commercially used databases.
11. It supports GUI applications.

All the above mentioned features make python a very interesting language to learn and a very easy language to use.

Few points to remember :

- Python is both dynamically typed (because it doesn't use explicit datatype declarations) and strongly typed (because once a variable has a datatype, it actually matters).
- Functions are declared using the 'def' keyword as follows:
  def function_name(arguments). Note that the keyword def starts the function declaration, followed by the function name, followed by the arguments in parentheses. Multiple arguments (not shown here) are separated with commas. Also note that the function doesn't define a return datatype.
- You can document a Python function by giving it a *doc* string.
- Everything, including function is an object in Python.
- Python functions have no explicit begin or end, and no curly braces to mark where the function code starts and stops. The only delimiter is a colon (:) and the indentation of the code itself.
- Python modules are objects and have several useful attributes. You can use this to easily test your modules as you write them.

<u>Bootstrap</u> :

What is Bootsrap?

Bootstrap is a front-end toolkit for rapidly developing web applications. It is a collection of CSS and HTML conventions. At its core, Bootstrap is just CSS, but it's built with Less, a flexible pre-processor that offers much more power and flexibility than regular CSS. Bootstrap remains very easy to implement; just drop it in your code and go. Compiling Less can be accomplished via Javascript, an unofficial Mac application, or via Node.js. Bootstrap looks great on tablets and smartphones as well because of its responsive CSS.

Features of Bootstrap that have been used :

Bootstrap comes equipped with HTML, CSS and JS to do all sorts of things. You just have to download it and include the files in your header and you are good to go. It comes with certain features which make it necessary to use HTML5 doctype.

Grid System :
Bootstrap utilizes 12-column grid system making a 940px wide container without responsive CSS activated. With responsive CSS, the grid adapts depending on the viewport. For a simple column layout, create a division of class type *row* and include divisions of class type *span* which must always add upto 12.

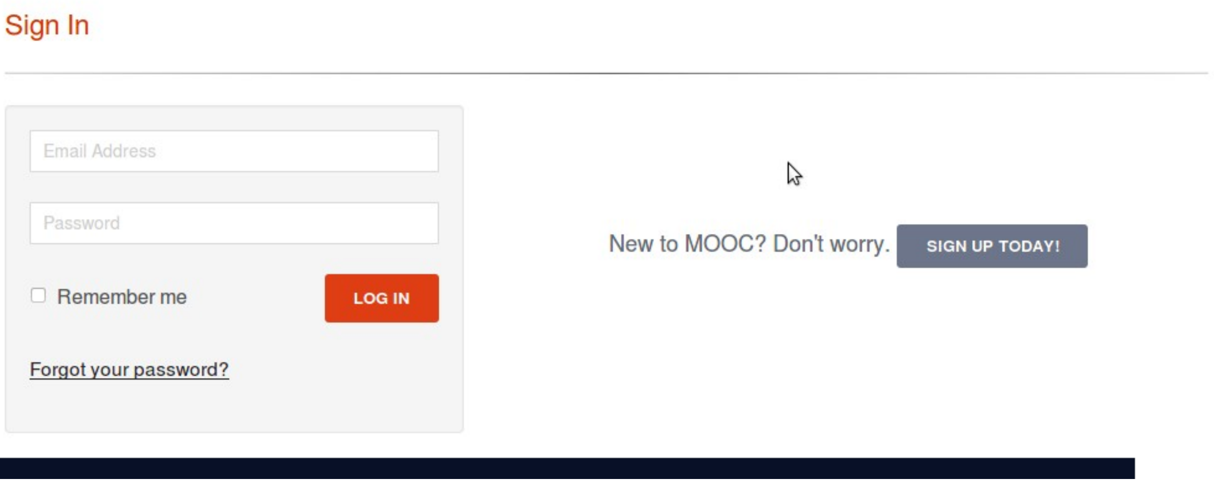
Base CSS :
To create forms, several classes have been used. The *well* is the classname for an 'in-set' container. Has a padding of 19 px and a ready made rounded corners. I used it in the login form. Other classes like *input, btn, btn-primary, controls,* etc have been used as and when required. One can use button drop-downs too.

Components :
Navbar, labels and alerts have been used wherever necessary.

Javascript :
I did not use Bootstrap's JavaScript features much in the design of the following pages. The JavaScript used there has been implemented as per the requirement for validation.

Page Designs :

To design the body of the pages, bootstrap's features have been used. Orange is the primary color. The fields email and password are required for you to proceed further in the login page. There are options for 'forgot password' and to 'register'. In the register page, alerts appear as soon as the field loses focus, incase the input is inappropriate. If the user overlooks this and hits *submit,* the page redirects the user to the same page asking him/her to change certain fields before submission.


Sign In

Email Address

Password

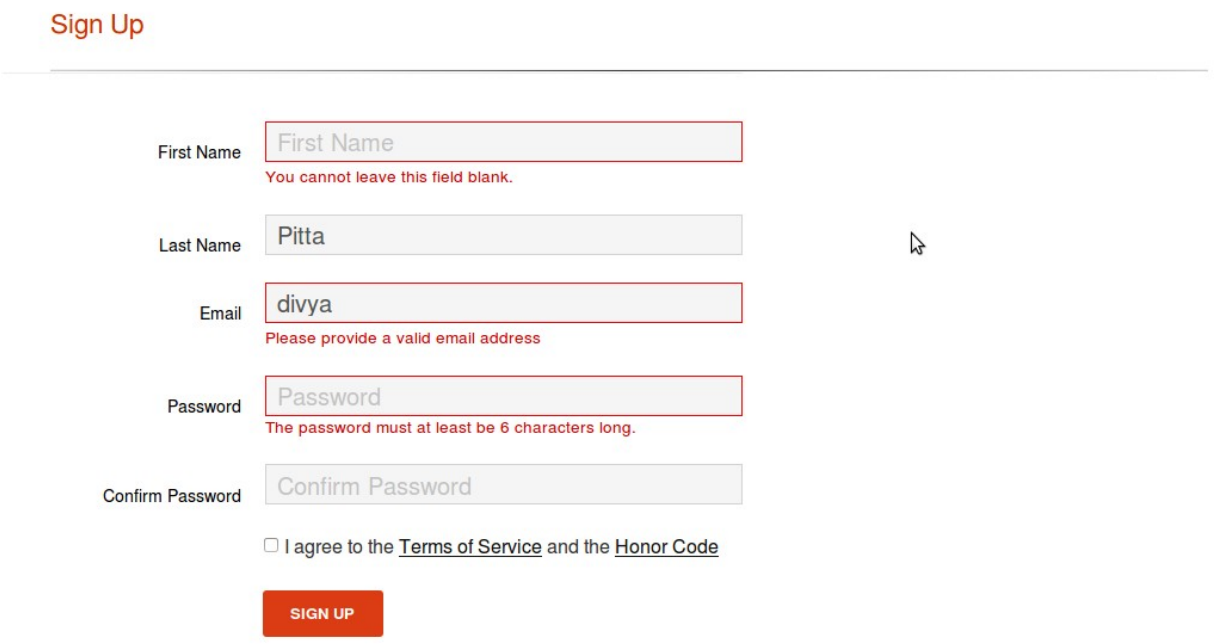
Remember me

LOG IN

Forgot your password?

New to MOOC? Don't worry. SIGN UP TODAY!

Login Page

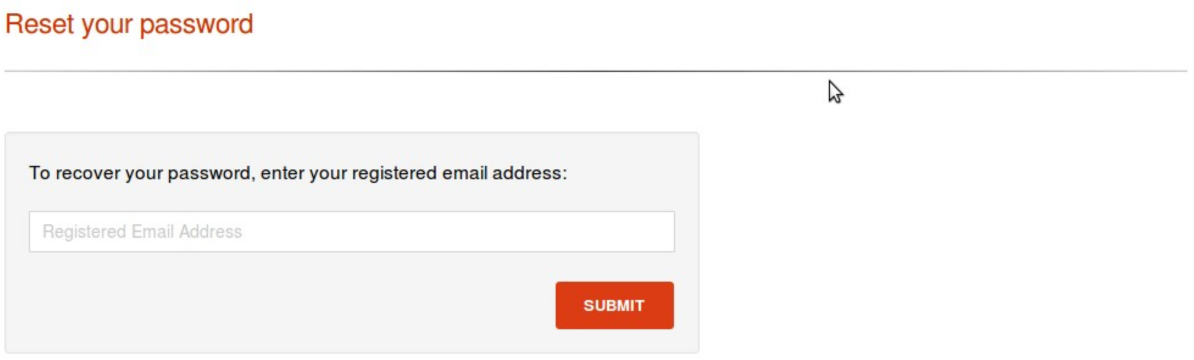

Sign Up

First Name	First Name You cannot leave this field blank.
Last Name	Pitta
Email	divya Please provide a valid email address
Password	Password The password must at least be 6 characters long.
Confirm Password	Confirm Password

I agree to the Terms of Service and the Honor Code

SIGN UP

SignUp Page


Reset your password

To recover your password, enter your registered email address:

Registered Email Address

SUBMIT

ForgotPassword Page

Features yet to be implemented :

1. In the login page, there will be an option to login through gmail/google account to prevent the user from remembering several logins and passwords. This will be implemented using Django Social Auth or OpenID.
2. A modal will appear on clicking forgotpassword link in the login page. I implemented this method using Bootstrap but the desired output could not be seen. I tried to modify the code but nothing worked. Some other procedure will have to be used.
3. Further customizing its look and feel.

Django :

Introduction :

Django is a high-level Python web framework that follows the **model-view-controller pattern**. It's primary aim is rapid development and clean design. It emphazises the principle of **reusability** and proper utilization of each and every component. It separates the representation of information and user's interaction with it. In django, **models** are nothing but *Python classes* which are mapped to relational databases being used. A **view** is equivalent to a system for processing requests with a *web templating system* and a **controller** is equivalent to regular expression based *URL dispatcher*.

Installing Django :

On your ubuntu terminal, use the command, *sudo pip install django.*
To check if it is installed, *python -c 'import django; print django.get_version()'.*
This will print the version of Django installed.

Why Django?

- **A Robust Object-Relational Mapping System :** It supports a large number of database systems and switching from one engine to another is very easy. This provides the developer great flexibility.
- **Readable and cruft-free URLs.**
- **Powerful Template System :** It is very easy to inject programming concepts in the templates.
- **Automatic Admin Interface :** Django comes up with a ready-made admin interface, which allows a developer to control Web applications via a Web portal. This admin interface is generic, and can be extended and customised, depending upon project requirements.
- **Internationalization :** The goal of internationalization and localization is to allow a

single Web application to offer its content in languages and formats tailored to the audience.
- **Super performance :** This is due to a very granular cache system.
- **Complete Development Environment :** It comes with a lightweight Web server for development and testing. When the debugging mode is enabled, Django provides very thorough and detailed error messages, which in turn, makes debugging easier, and allows one to quickly isolate and fix bugs.
- **A rapidly growing community**

Django Project Architecture :

An empty django project folder consists of the following important files :

1. **settings.py :** This contains project settings, such as database connection information, email server configuration, installed apps, media folder location and static folder location with their URLs, template folder location and reference to some Django modules.
2. **urls.py :** Each request to the application goes through urls.py, and entries in the urlpatterns list, to decide which request is to be forwarded to which view of a particular app.
3. **manage.py :** It contains commands for maintaining and managing our Django project. For example: startapp, startproject, runserver, syncdb, etc.
4. **__init__.py :** An empty file that tells Python that this directory should be considered a Python package.
5. **wsgi.py :** An entry-point for WSGI-compatible webservers to serve your project.

Each project consists of small modules called apps. Each app in turn consists of yet another set of files as follows :

1. **models.py :** This is the place for all your application data. These are basically classes extending *models.Model*, which is provided by Django ORM library. It is recommended that business logic should be written in models so that it remains intact with models and can be used by any other app of your Django project.
2. **views.py :** This is the place where you render HTML pages for the HTTP requests that are redirected to particular view function via urls.py. Basically, views contains functions, each of which get the HTTP request as first parameter. They get data from models and render HTML pages using templates, or redirect to another view function.
3. **admin.py :** This is the file to be manipulated with to change the way a project appears to the admin.
4. **urls.py :** It consists of the urlpatterns that are dealt with by the server.
5. **tests.py :** These are functions performing unit test cases. This file should be effectively used for regression testing (to check the side effects of your code changes).

Django Internal Architecture (The flow of an HTTP Request) **:**

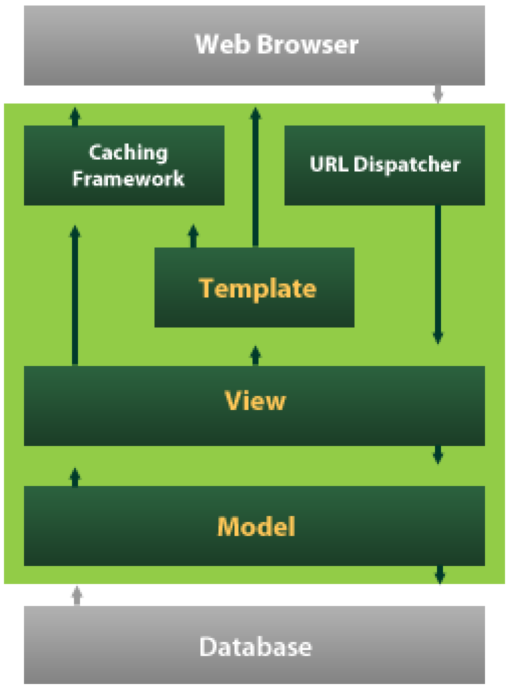


Image taken from www.pythonhosted.org

What happens when we request for a url?

1. The URL dispatcher (urls.py) maps a requested URL to the view function and calls it. If caching is enabled and there exists a cache version of the page, then it is returned. In any other case, the page is loaded.
2. The view function (views.py) performs the requested action which is generally reading or writing from/to a database.
3. The model (modesl.py) defines the Python data and interacts with it.
4. After performing the requested task, the view returns an HTTP Response object to the web browser which can again be saved in cache for a specified length of time.
5. Templates are generally used by the view to return appropriate HTML pages as and when required.

References :

1. http://ii.library.jhu.edu/2013/01/08/the-abcs-of-moocs/
2. http://cit.duke.edu/blog/2012/09/moocs-what-role-do-they-have-in-higher-education/
3. http://publications.cetis.ac.uk/wp-content/uploads/2013/03/MOOCs-and-Open-Education.pdf
4. http://nogoodreason.typepad.co.uk/no_good_reason/2013/05/if-education-were-free-what-would-moocs-be.html
5. http://www.forbes.com/sites/sap/2012/09/06/massive-open-online-course-a-threat-or-opportunity-to-universities/
6. MOOCS and Open Education : Implications for Higher Education – A White Paper By Li Yuan and Stephen Powel.
7. **Pro Git** by Scott Chacon
8. **Think Python –** How to think like a computer scientist
9. **Dive into Python** by Mark Pilgrim
10. www.docs.djangoproject.com
11. http://twitter.github.io/bootstrap
12. https://dev.twitter.com/blog/bootstrap-twitter
13. http://www.linuxforu.com/2013/03/django-a-web-framework-with-immense-potential/