

Sulphur

[Web Content Management System]

MAJOR PROJECT REPORT

Project Report submitted in the partial fulfillment of the requirement for
the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY



Mohd. Seraj Alam

01020703114

Varun Taneja

01620703114

Ravinder Kumar

01920703114

Abha Kumari

03620703114

Kishor Das

00320703114

Guided by

Dr. Srinivasa KG

(Assoc. Professor)

CANDIDATE'S DECLARATION

It is hereby certified that the work which is being presented in the B. Tech Major Project Report entitled "**SULPHUR (WEB CONTENT MANAGEMENT SYSTEM)**" in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** and submitted in the **Department of Information Technology of Ch. B.P. Govt. Engineering College, New Delhi (Affiliated to Guru Gobind Singh Indraprastha University, Delhi)** is an authentic record of our own work carried out during a period from **January 2017 to May 2017** under the guidance of **Dr. Srinivasa KG, Assoc. Professor.**

The matter presented in the B. Tech Major Project Report has not been submitted by me for the award of any other degree of this or any other Institute.

Mohd. Seraj Alam

01020703114

Varun Taneja

01620703114

Ravinder Kumar

01920703114

Abha Kumari

03620703114

Kishor Das

00320703114

This is to certify that the above statement made by the candidate is correct to the best of my knowledge. He/She/They are permitted to appear in the External Major Project Examination.

Dr. Srinivasa KG

ABSTRACT

This project investigates different existing open-source **Web Content Management Systems (WCMS)** and their features. The main objective of WCMS is to allow non-technical users to easily update and manage a website's content. But WCMS products available on the market today have integrated many other complex functionalities, which makes the user interface difficult to understand and use. All WCMS require some amount of training on installation and where to start. After integration users need to read the manual or take lessons to use other available advanced features. What we explored here is whether to implement a WCMS from existing options compared with the feasibility of building a new one from scratch. This paper discusses the considerations of developing a new WCMS and develops a **Content Management System** which is quite easy to use and implement, including the advantages and disadvantages relative to an off-the-shelf product.

ACKNOWLEDGEMENT

We express our deep gratitude to **Dr. Srinivasa KG**, Assoc. Professor, Department of Information Technology for his valuable guidance and suggestion throughout our project work.

We would like to extend our sincere thanks to All Teachers of Department for their time to time suggestions to complete our project work. We are also thankful to [Prof.K.C. Tiwari](#), Principal for providing me the facilities to carry out our project work.

Signature of students.

Mohd. Seraj Alam

01020703114

Varun Taneja

01620703114

Ravinder Kumar

01920703114

Abha Kumari

03620703114

Kishor Das

00320703114

TABLE OF CONTENTS

	PAGE
CANDIDATE'S DECLARATION	1
ABSTRACT	2
ACKNOWLEDGEMENT	3
TABLE OF CONTENTS	4
LIST OF TABLES	6
LIST OF FIGURES	6
LIST OF ABBREVIATIONS	7
CHAPTER:	
1 INTRODUCTION	8
1.1 What is Content Management System (CMS)?	9
1.2 Project Definition and Purpose	10
1.3 Content Overview	10
2 BACKGROUND	
2.1 A Short History of CMS Development	12
2.2 Who needs a Content Management System?	12
2.3 Types of Content Management Systems	13
2.3.1 Enterprises CMS	
2.3.2 Web CMS	
2.4 What is Open Source?	16
2.5 Commercial CMS	17
2.6 Comparative study of Open Source CMS	17
2.7 Problems in the existing Open Source CMS	19
2.8 Performance for Deployed Applications	21
2.9 To Build or To Buy?	22
2.9.1 Using an Existing CMS	

2.9.2	Build your own CMS	
2.10	Advantages of this Project	24
2.11	Project Limitations	25
3	TOOLS	
3.1	Django Framework	26
3.1.1	Model	
3.1.2	View	
3.1.3	Template	
3.2	Python	27
3.2.1	Python Features	
3.3	My SQL	28
3.3.1	What is Database?	
3.3.2	MySQL Database	
4	FEATURES, DESIGN AND IMPLEMENTATION	
4.1	Features	31
4.1.1	Default Templates	
4.1.2	Access Control	
4.1.3	Easily Editable Content	
4.1.4	Workflow Management	
4.2	System Requirement and Specification	34
4.3	Design	35
4.3.1	Database Design	
4.3.1.1	Class diagram/database architecture	
4.3.2	UML diagrams	
4.3.2.1	Use – Case diagrams	
4.3.2.2	Sequence Diagrams	
4.4	Implementation	41
4.5	TESTING	52
4.6	OUTPUT SCREENS	56
4.7	CONCLUSION	58
4.8	FUTURE ENHANCEMENT	59

LIST OF TABLES

	PAGE
Table 2.1 Comparison of Different WCMS	18

LIST OF FIGURES

Figure 2.1	19
------------	----

Google Trends:

- Drupal,
- Joomla,
- WordPress.

Figure 2.2	
------------	--

Page load time in Various CMS	21
-------------------------------	----

LIST OF ABBREVIATION

CMS: Content Management System

WCMS: Web Content Management System

CSS: Cascade Style Sheet

HTML: Hyper Text Markup Language

API: Application Programming Interface

GUI: Graphical User Interface

WYSWYG: What You See What You Get

SQL: Structured Query Language

RDBMS: Relational DataBase Management System

WSGI: Web Server Gateway Interface

CHAPTER 1

INTRODUCTION

Content management (CM) is the set of processes and technologies that support the collection, managing, and publishing of information in any form or medium. Content may take the form of text (such as documents), multimedia files (such as audio or video files), or any other file type that follows a content lifecycle requiring management. Content management is an inherently collaborative process. It often consists of the following basic roles and responsibilities:

- Creator: responsible for creating and editing content.
- Editor: responsible for modifying the style of delivery, including translation and localization.
- Publisher: responsible for releasing the content for use.
- Administrator: responsible for managing access permissions to folders and files, usually accomplished by assigning access rights to user groups or roles.
- Consumer, viewer, or guest: the person who reads or otherwise takes in content after it is published or shared.

1.1 What is Content Management System (CMS)?

A content management system (CMS) is a computer system or an application that allows publishing, editing, or modification of content, as well as site maintenance, from a central page. The content on the World Wide Web consists of HTML, XML, and other documents and media files. This content can be published manually by editing and organizing files on a file system exposed to the web through a web server, requiring much technical expertise and tedious work. Content management systems store the actual content which can be text or images in a database. The system then automatically pulls the content out and shows it on appropriate pages based on the rules. Content management systems were created to help people publish documents and media with less technical intervention and in a more consistent and automated fashion. Anything from a document management system, to a media asset management system, to a portal, to a blog system could be considered a Content Management System.

Content management system speeds up the content updating process and makes it easy for a non-technical user. This helps in keeping the content up-to-date and timely. They are frequently used for editing, controlling, versioning and publishing specific documentation. They are used for different types of applications to manage its content. Below are some of the CMS applications.

- Blogs: comments on a particular topic
- News: reading newspapers online instead of reading the hard/physical copy
- Wikis: these are open to public editing.

The core features of Content Management Systems vary widely from system to system; many simpler systems provide only a handful of

features, while others, especially enterprise systems, are much more complex and powerful.

- Allow for a large number of people to share and contribute to stored data
- Control access to data based on user role (i.e., define information users or user groups can view, edit, publish, etc.)
- Facilitate storage and retrieval of data
- Control data validity
- Simplify report writing
- Define data as almost anything: documents, movies, texts, pictures, phone numbers, articles etc.

1.2 Project Definition and Purpose

In this project, we have built a Simple CMS which is a web content management system that can be used to dynamically manage the content of a simple static website. For eg. the news section of the Computer Science Department needs to be updated very often and since it is static HTML page, a technical person having knowledge of HTML and CSS is required to update even a small portion of the section. The primary objective of this thesis is to demonstrate how Simple CMS engine could be integrated with the “News & Events” section page, so that the process of updating the content becomes faster and easier for the department. The main purpose of this project is to have a user-friendly content administration interface that includes most common CMS functions appropriate for small and simple websites, so that a novice user can manage the website content. A user having less coding knowledge can easily add, edit and format the website’s content using the rich text editor integrated in the Simple CMS engine without having to deal with the HTML and CSS code.

1.3 Content Overview

The remainder of the thesis is organized as follows. Chapter 2 describes different types of CMS, compares the most popular open source CMS and discusses their advantages and disadvantages. It also discusses the advantages and limitations of this project. Chapter 3 describes different tools and technologies used to build this project- Simple CMS. Chapter 4 describes the common CMS features implemented in this project. In addition to that, it also describes the high level architecture, database architecture and implementation of Simple CMS. Finally Chapter 5, a conclusion and description of future work in this project to help contribute more for websites requiring more advanced functions in addition to the common basic CMS features available in the project.

CHAPTER 2

BACKGROUND

2.1 A Short History of CMS Development

Claims about who wrote the first CMS are varied, and include Roxen (1994) and Blitzen (mid-1990s), Ingeniux (1999), and Vignette. The main features were a very structured development environment and one had to use tags and templates because there was no WYSIWYG (What You See Is What You Get) editor. If one could not use HTML, they probably were not going to edit their site. Most of those websites were written by web design agencies rather than by software companies and every agency had their own web designers who could write HTML code. The second stage of CMS development was led by software houses who took over the functionality and started to build today's CMS. Key features which were slowly built in included WYSIWYG text editing, search, and the addition of features like survey tools and podcasts. An early stage leader RedDot, and others, led the growth of specialists such as DotNetNuke and Mambo (which later changed to Joomla and DotNetNuke), which are still used today. CMS became feature-rich and web agencies needed both the technical skills, as well as designers, to adapt each client site into the CMS frameworks.

2.2 Who needs a Content Management System?

Organizations need a content management system if they:

- Use the same content over and over again in one publication (e.g., a warning in an instruction manual may be used several times within that manual).
- Publish the content to more than one media channel (e.g., the content in a printed instruction manual that ships with the product may also be used in the online help information).
- Publish the content in multiple languages.

The types of publications that these organizations produce might include:

- Technical documentation (parts catalogs, software documentation, user's manuals)
- Reference materials (encyclopedias, dictionaries, standards guides)
- Testing and training materials (e-learning programs, testing booklets)
- Marketing and educational materials (packaging, promotional flyers and ads, brochures).

2.3 Type of Content Management Systems

CMS come in all shapes and sizes, and can manage anything a team of individuals is working on. From managing simple static website content, to allowing collaborative documentation across the Internet (wiki), CMSs perform many functions. CMS packages can generally be classified into two categories:

- i. Enterprise CMS
- ii. Web CMS.

2.3.1 Enterprises CMS

Enterprise Content Management (ECM) is a formalized means of organizing and storing documents, and other content, that relate to the organization's processes.

These high-powered software packages are usually comprehensive solutions, delivering effective content management for use on an enterprise, or corporate level.

They are designed to help a corporation become more efficient and cost-effective, and increase accuracy and functionality. It also helps decrease human error and customer response times. They can integrate corporate functions such as shipping and delivery systems, invoicing, employee and human resource issues, customer relations, and document management and sales systems. Enterprise CMS brings data management down to the user's level; so many users can add their individual piece of information to a very large integrated system data. Software companies deploying these complex systems offer highly customized company-wide solutions, which means the software usually comes with a relatively high price tag.

This type of CMS is generally used by large e-Commerce organizations. E-commerce websites are those that have a shopping cart designed to sell items online. The examples of such CMS used for e-Commerce websites are Magneto etc.

2.3.2 WEB CMS

Today, we are all used to the idea that we can create our own documents and publish it on the web. But if we go back fifteen or more years, the only way one could create a website was by understanding 'HTML'. Products like Dreamweaver and FrontPage were not around; this meant that if one wanted to build a website then he/she needed someone with technical skills to write it. Also once they had written it, one still needed someone with technical skills to change it, as changes involved reading HTML code to determine where to add content.

A Web Content Management System (WCMS) is a software system that provides website authoring, collaboration, and

administration tools designed to allow users to create and manage website content with ease. A presentation layer displays the content to the website visitors based on the set of templates. WCMS were developed to resolve the issue of having highly experienced technical staff adding low-level content to a website. WCMS is a web application used to create, manage, store and deploy content on web pages. It is mainly a website maintenance tool that allows non-technical users to make changes to the website, and therefore the website's authoring and administrating tools. A user with little knowledge of programming languages can easily create and maintain the site's content. Web content types can include text, graphics and photos, video or audio, and an application code that renders other content or interacts with the visitor. Web content may be created, organized, and managed in an unlimited number of ways. Therefore a wide variety of WCMS systems have been built to handle many different situations. Some of these applications are general-purpose, providing a consistent general structure to any type of content. Others are more specific to types of content, intended audiences, or workflows.

In order to characterize a WCMS we could say the following:

- It manages small units of information (web pages); each unit of information is interconnected via a navigation structure or path.
- It is focused primarily on page creation and editing; it facilitates content creation, content control, editing, and many essential web maintenance functions by presenting the non-technical user with an interface that requires no knowledge of programming languages or markup languages to create and manage content.
- It provides a publishing engine that allows created or amended content to be made available to a website visitor.
- It often provides an approval process or workflow that ensures that content is validated before it is released or published to a website.

One of the first steps to managing web content is to decide what kind of content one wants to have on their site. The type of content to be published directly influences what type of authoring tools will be needed. For example, if one wants to post blog updates and link to other sites, the architecture of the site will differ from a company who wants to post videos or interactive tutorials. Keeping content organized will not only make it easier for visitors to navigate the site, but will help avoid content overlap or repetition in the future. There are a number of WCM tools on the market that can handle most of these issues. Choosing the right WCMS can be tricky, considering the dozens of available options. When looking for a WCMS, a company should first determine whether their website will require a basic CMS, mid-range CMS, or complex CMS.

2.4 What is Open Source?

The term “open source” is a key distinction. It means the software’s source code is freely available for everyone to see and change, but it also has many wider implications. While proprietary software is created, distributed and maintained by a business, with open source software these tasks are handled by a community of developers and users. Just how effective that community is at its job is an important consideration when choosing an open source CMS. Open source content management systems are free in many ways. A user can do what he/she wishes with the product and the code behind it, extending and integrating it as they see fit. There’s no license cost for the software, and anyone can download and install them on a web server without cost, though it is likely they will have to pay for the server, or pay someone to install the system. An open source CMS takes work. A user either needs to put a lot of time into implementing and maintaining his/her system, or hire someone to do it for them.

2.5 Commercial CMS

Commercial CMS systems overcome some of the disadvantages of open source CMS systems. A distinct advantage of a commercial CMS system is that the vendor has control over the entire environment and most modules that are created for it, so it creates a more user friendly and secure end product. Since commercial CMS systems are not open source, the community does not have access to the Source Code and so possible hacker exploits are much less likely. There are usually fewer core CMS security updates from commercial vendors than open source vendors. Some commercial CMS systems are “cleanly coded” and well-documented, with an extensible architecture which allows a third party developer to read and follow the source code, understand the Application Programming Interface (API) and create custom enhancements to the CMS, without extensive issues.

There are also some commercial CMS systems that are good tools for specific purposes. For e.g. Microsoft SharePoint was designed to quickly create an internal Intranet, especially for clients who are using Microsoft tools. However SharePoint is not appropriate for implementing public Internet sites, as the tools are geared toward Intranets. Moreover customizing SharePoint for Internet applications can be a very expensive and inflexible process. In addition, expensive license fees for SharePoint, is another disadvantage. Other commercial CMS systems are highly complex and priced extremely high. An example of this would be Open Text, who formerly marketed “RedDot CMS” and now market a new Web Content Management system as part of their Web Solutions group. Licensing alone for a typical Open Text project begins at approximately \$125,000.

2.6 Comparative study of Open Source CMS

Table 2.1 compares different existing Web CMS. It tells about Programming languages, databases and web servers used to build these CMS. It also provides information on FTP support and UTF-8 support for these CMS.

Table2.1. Comparison of Different Web Content Management Systems

Product	Programming Language	Database	Web Server	FTP Support	UTF-8 Support
Joomla	PHP	MySQL	Apache	Provided as a free add-on	Limited support available
Drupal	PHP	MySQL, Postgre SQL	Apache, IIS	Limited FTP support	Available
WordPress	PHP	MySQL	Apache, Mod rewrite	Available as a free add-on	Available
Plone	Python	Zope	Apache, IIS, Zope	Available	Available
TYPO3	PHP	MySQL, Postgre SQL, Oracle, MSSQL	Apache, IIS	Available	Available
Open CMS	Java 1.4	MySQL, Postgre SQL, Oracle, MSSQL	Tomcat, Apache	Not Available	Available

Source: Internet Tips. Comparison with other Web Content Management Systems, 2012.
<http://www.internettips.com/departments/joomla-web-content-management-system/5-of-6-comparison-withother-web-content-management-systems>

Joomla, Drupal and WordPress are the most popular open source CMS being used today. Drupal has a steep learning curve and so it's difficult for a first time developer to understand the modules. It requires more advanced coding knowledge when it comes to customizing. Joomla has a better user interface than Drupal for newcomers. It is quite simple to add and edit pages. The design layer is almost the same as Drupal as they both do not have a lot of design templates. So the issue is that most of the Joomla sites will look the same. WordPress is the perfect solution for small blogging websites as it has simple and user friendly interface. There are a lot of great design templates available for the websites. The issue with WordPress

is when one wants to do something outside the blogging world. Figure 2.1 shows the recent Google trends for these CMS. The blue scale stands for Drupal, red for Joomla and yellow for WordPress. As it's seen from the figure WordPress leads the way due to its user friendly interface and large number of design templates.

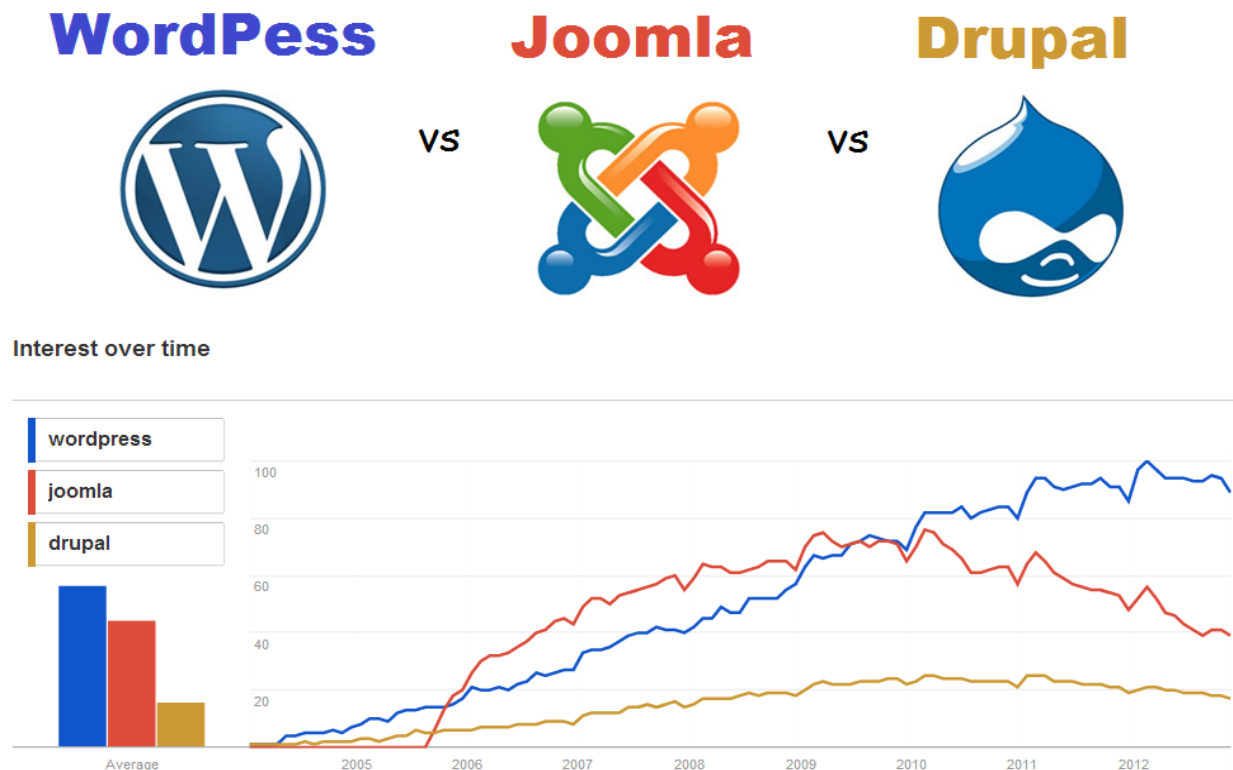


Figure 2.1 Google Trends: Drupal, Joomla, WordPress.

2.7 Problems in the existing Open Source CMS

The main purpose of a Content Management System is to make it easy for even a novice computer user to maintain and manage a site. There are many CMSs available in the market, but deciding which is the most suitable will largely depend on the website's requirements and the budget.

Both Drupal and Joomla have a very bloated and confusing administration. This is because both these packages have many advanced features in addition to the normal basic content management features. For example, polls providing the capabilities to capture votes on different topics in the form of multiple-choice questions, or news feeds, which provide syndicated content (RSS, RDF, and Atom feeds). This makes the CMS complex and difficult to use. Generally there will be some trade-offs between the complexity of the requirement and the simplicity in choosing a CMS. For example, if the website only requires a series of text based pages with a spattering of images, then a simple, basic CMS will be easy to use. However, if the requirements for the website include having multiple blogs, video and audio uploads, a forum, an events calendar, and an e-commerce facility, then it is good to choose a CMS capable of providing those facilities. Inevitably though, that CMS is going to be a little more complex to maintain. If the website requires complex features, the best thing to do is to hide these complex features. A good user interface should make most common tasks the most prominent and hide rare tasks so that they do not get in the way. There was research done by University of Minnesota Office for Information Technology's usability lab which identified many usability problems with Drupal's administration. As of this writing, Drupal's administration interface is confusing and not user friendly. Joomla's administration usability and learning curve is better than Drupal's, but not enough to provide a noticeable advantage to the end-user over Drupal.

WordPress has a much better and very intuitive administration design, which makes it easier to learn. It includes features such as drag and drop, resulting in the generation of code without technical intervention. It would be more correct to describe such products as 'website builders' than Web Content Management Systems. The main feature that is not seen in most of today's complex CMSs is intuitive and user-friendly website administration. Although there are no license fees with open source CMS, one still needs to pay a vendor to create an interface design or a theme, install the CMS, configure it for use, implement the "theme", and also pay for ongoing support.

2.8 Performance for Deployed Application

Figure 2.2 shows a graph that compares the page load time for textual information for different open source CMSs with Simple CMS. It shows that Joomla and Drupal have higher page load times than WordPress. Simple CMS has nearly similar response time to WordPress. When any page get loaded from the server to a client machine, the client machine stores some data on its site so when that page gets loaded again it takes less time to load compared to the first time. In case of textual information, Joomla and Drupal take 1.10 seconds less time than without the caching, while WordPress only reduces 0.32 seconds in PLT. It seems that it caches less data in memory.

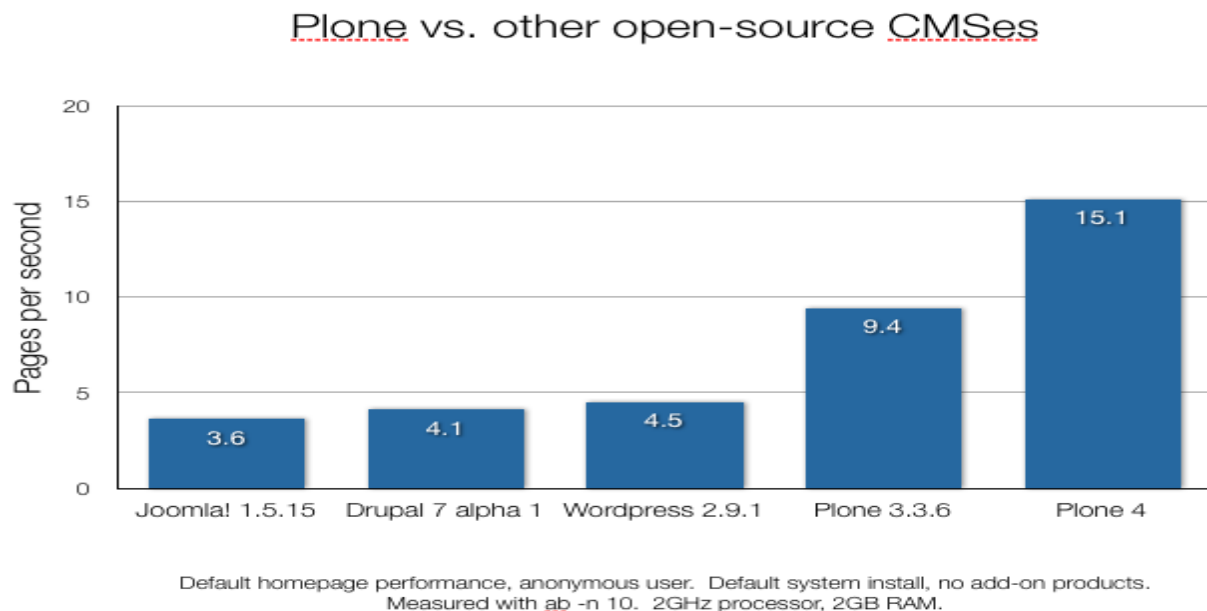


Fig 2.2 Page load Time in various CMS

Source: <http://jstahl.org/wp-content/uploads/2010/08/plone-4-vs-others.png>

2.9 TO BUILD OR TO BUY?

There is no single best way to implement a CMS. When choosing between writing/creating a new CMS for yourself and using existing software there are three key pieces of information need to be taken into account.

1. The functionality and features required which will decide how complex it will be.
2. How closely does an existing system match the requirements and what would it cost to implement it versus building a new CMS?
3. How soon is the CMS system needed?

If the requirements are not very complex, and if it is not needed immediately, it would be faster and less expensive to create a new Simple CMS. Otherwise, researching the different existing CMSs is required before starting to use one. In addition, one can also contribute to one of the existing open CMSs and can get feedback from experienced programmers who can help to configure it according to one's needs.

2.9.1 Using an Existing CMS

Pros:

- Lots of hosted solutions and online help are available for the off-the-shelf products. These products also provide live help and a dedicated resource to help the website builder. There are blogs which discusses the issues faced while installing or using such a product and can help a new user to overcome similar problems.

- Others can serve as administrator if the user who integrated it is unavailable. Once the website is integrated into the CMS, any user can read the manual and use it to manage the content.

Cons:

- Research time in selecting the one that is closest to user requirements.
There are lots of CMSs available in the market, but each of these products has different features. So one needs to research each of them and their features and select the closest one.
- An open source CMS is likely to have several pieces of functionality that are not required by a simple website, slowing the overall speed of the site.
- Steeper learning curve. These off-the-shelf products have a lot of advanced features and so it's difficult for a first time user to understand all the modules.
- Bug fixes and security updates are released by the vendor. These CMS systems issue new versions of their own core CMS system quite often, normally to close security hole, fix bugs and add new features. And installing these new versions of CMS systems is required or the website will potentially be easy to hack if security holes are not fixed. Installing a new version of these CMS systems is complex and convoluted and requires disabling all existing modules prior to an upgrade.
- More limited in terms of customization

2.9.2 Building Your Own CMS

Pros:

- Total control over the features: The developers are familiar with the system and are able to build new features quicker without having to navigate through a certain design pattern set out by an open source solution.
- Low cost. It will be cheap to build one meeting your requirements than to get the off the-shelf product to do what is needed.

- Easy to understand. Considering the requirements first, it will be easier for everyone in the firm to understand it.
- No bug fixes from other developers unlike the existing open source CMS.

Cons:

- Implementation time: it is difficult for companies to make their own CMS because they might not have the appropriate in-house professionals to do so
- Only you understand it and can fix it. There are problems in retaining the CMS knowledge when the individuals who created it leave the organization.

2.10 Advantages of this Project

- Simple CMS is easy to install and administer.
- The user interface is more intuitive.
- It is able to import and use static WebPages in the CMS.
- Easily create editable regions for clients within the front-end of the website.
- It features easy menu page creation and a nice WYSIWYG editor, which allows easy content formatting, image uploading and image resizing.
- It also gives the ability to add a new style sheet or modify the existing ones.
- The workflow will only allow the newly created page to be visible to the users when the administrator approves and publishes it.
- The audit trail report displays the information regarding all changes made to the website pages and thus helps in deterring fraud by keeping a record of which pages were modified or deleted.

2.11 PROJECT LIMITATIONS

You cannot create/build a new template using Simple CMS. There is only one editable region on a page. One can edit a simple website that has a homepage with all the other pages as the tabbed menu/navigation bar. There can be only one navigation bar whose pages can be modified.

CHAPTER 3

TOOLS

Several technological tools were used in this project; each is summarized in the sections that follow.

3.1 Django framework

Django is an open source web application framework written in Python. The primary goal of Django is to make the development of complex, data-based websites easier. Thus Django emphasizes the reusability and pluggability of components to ensure rapid developments. Django consists of three major parts: model, view and template.

3.1.1 Model

Model is a single, definitive data source which contains the essential field and behavior of the data. Usually one model is one table in the database. Each attribute in the model represents a field of a table in the database. Django provides a set of automatically-generated database application programming interfaces (APIs) for the convenience of users.

3.1.2 View

View is short form of view file. It is a file containing Python function which takes web requests and returns web responses. A response can be HTML content or XML documents or a “404 error” and so on. The logic inside the view function can be arbitrary as long as it returns the desired response. To link the view function

with a particular URL we need to use a structure called URLconf which maps URLs to view functions.

3.1.3 Template

Django's template is a simple text file which can generate a text-based format like HTML and XML. The template contains variables and tags. Variables will be replaced by the result when the template is evaluated. Tags control the logic of the template. We also can modify the variables by using filters. For example, a lowercase filter can convert the variable from uppercase into lowercase.

3.2 Python

Python is the language used to build the Django framework. It is a dynamic scripting language similar to Perl and Ruby. The principal author of Python is Guido van Rossum. Python supports dynamic typing and has a garbage collector for automatic memory management. Another important feature of Python is dynamic name resolution which binds the names of functions and variables during execution. Like Perl, Python source code is also available under the GNU General Public License (GPL).

3.2.1 Python Features

Python's features include —

- **Easy-to-learn** — Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** — Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** — Python's source code is fairly easy-to-maintain.
- **A broad standard library** — Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

- **Interactive Mode** – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** – Python provides interfaces to all major commercial databases.
- **GUI Programming** – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** – Python provides a better structure and support for large programs than shell scripting.

3.3 MySQL

MySQL is the most popular Open Source Relational SQL Database Management System. MySQL is one of the best RDBMS being used for developing various web-based software applications. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company.

3.3.1 What is a Database?

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.

Other kinds of data stores can also be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those types of systems.

Nowadays, we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as **Foreign Keys**.

A Relational DataBase Management System (RDBMS) is a software that —

- Enables you to implement a database with tables, columns and indexes.
- Guarantees the Referential Integrity between rows of various tables.
- Updates the indexes automatically.
- Interprets an SQL query and combines information from various tables.

3.3.2 MySQL Database

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons —

- MySQL is released under an open-source license. So you have nothing to pay to use it.
- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.

- MySQL works very quickly and works well even with large data sets.
- MySQL is very friendly to PHP, the most appreciated language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
- MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

CHAPTER 4

FEATURES, DESIGN AND IMPLEMENTATION

This chapter discusses the various features, architecture and design components of the project. Screen images of the graphical user interface for select functions are in the Appendix.

4.1 FEATURES

The following are some of the key features of web based CMS and its implementation in this project.

4.1.1 Default Templates

Standard output templates built using HTML and CSS that can be automatically applied to new and existing content, allowing the appearance of all content to be changed from one central place. This separates the content from the presentation. This project has 3 default templates that can be used according to the requirement. The user can also integrate their own template (HTML and CSS) files into the Simple CMS engine and can set it as the default theme.

4.1.2 Access Control

Session management and user accounts are typically included in a CMS.

Roles can then be created and assigned any number of permissions. Users are then assigned roles, giving them the permissions included in those roles. For example, an administrator might create an "Editor" role, assigning editing permissions of all content to that role, rather than granting that role to the appropriate user accounts. When users with that role log into the CMS, they automatically have access to edit content . This project has three user defined roles: admin, editor and designer. Different roles can be assigned to different user accounts and thus each has access to different sections of the CMS. The admin user has access to all the sections, whereas the editor can create and modify pages and the designer can manage stylesheets. All the users can modify their own user profile.

4.1.3 Easily Editable Content

Content is separated from the visual presentation of a site, which makes it much easier and quicker to edit and manage. The standard web element for entering large blocks of text is the HTML textarea. To format text using a textarea, a user needs to include HTML elements or some other application-specific markup. To overcome this difficulty, most WCMS software includes a rich text editor for editing rich text within web browsers, which presents the user with a WYSIWYG (what-you-see-is-what-you-get) editing area. WYSIWYG editors allow non-technical individuals to create and edit content. These editors allow the users to see what their content will look like while they are in the process of editing it. The aim is to reduce the effort for users trying to express their formatting directly as valid HTML markup. Rich text editors manage the content formatting.

4.1.4 Workflow Management

Workflow features are implemented to assist teams with the process of editing and publishing content. The content publication is generally immediate or scheduled. For example, the simplest type of workflow is used in this project: add new content (page) to a publishing queue, where it awaits final approval from a site administrator before the content appears on the web site. More robust workflow might delegate work to separate types of users, often based on security roles. An editor, for example, might automatically be notified when new content is posted, letting him/her edit and approve the content before publishing.

This project uses a simple workflow where the newly added page is first entered into the publishing queue and, once the administrator approves and publishes it, the new page appears as of the menu items on the homepage.

4.1.5 Audit Trial Report

The Audit Trail Report shows the date and time that every page was created or modified. In addition to the date and time, the name of the user who made the entry or modification is also listed, along with the type of action performed: Created, Modified, or Published/Unpublished. This report can help deter fraud by keeping a record of which pages were modified or deleted; and it can let you know what data needs to be reentered in the case of data corruption. Only the admin user can view this report.

4.2 System Requirement and Specification

Hardware requirements

Processor	PENTIUM IV 3.0 GHZ
RAM	500 MB
Hard Disk	80 GB
Mouse	Logitech Serial Mouse
Keyboard	Standard 104 Enhanced Keyboard

Software requirements

Framework	Django 2.0.2
Web Server	Apache
Browser	Internet Explorer
Server side scripting	Python 3.x
Database	mysql
Language	python 3.6.5
Client side scripting	HTML, CSS
Code Editor	Microsoft Visual Code

4.3 Design

Design is the first step in the development phase of any principle or technique, for the purpose of defining a device, a process or system in sufficient detail to permit its physical realization. Once the software requirements have been analyzed and specified, the software design involves three technical activities-Design, Coding, Generation and Testing that are required to build and verify the software

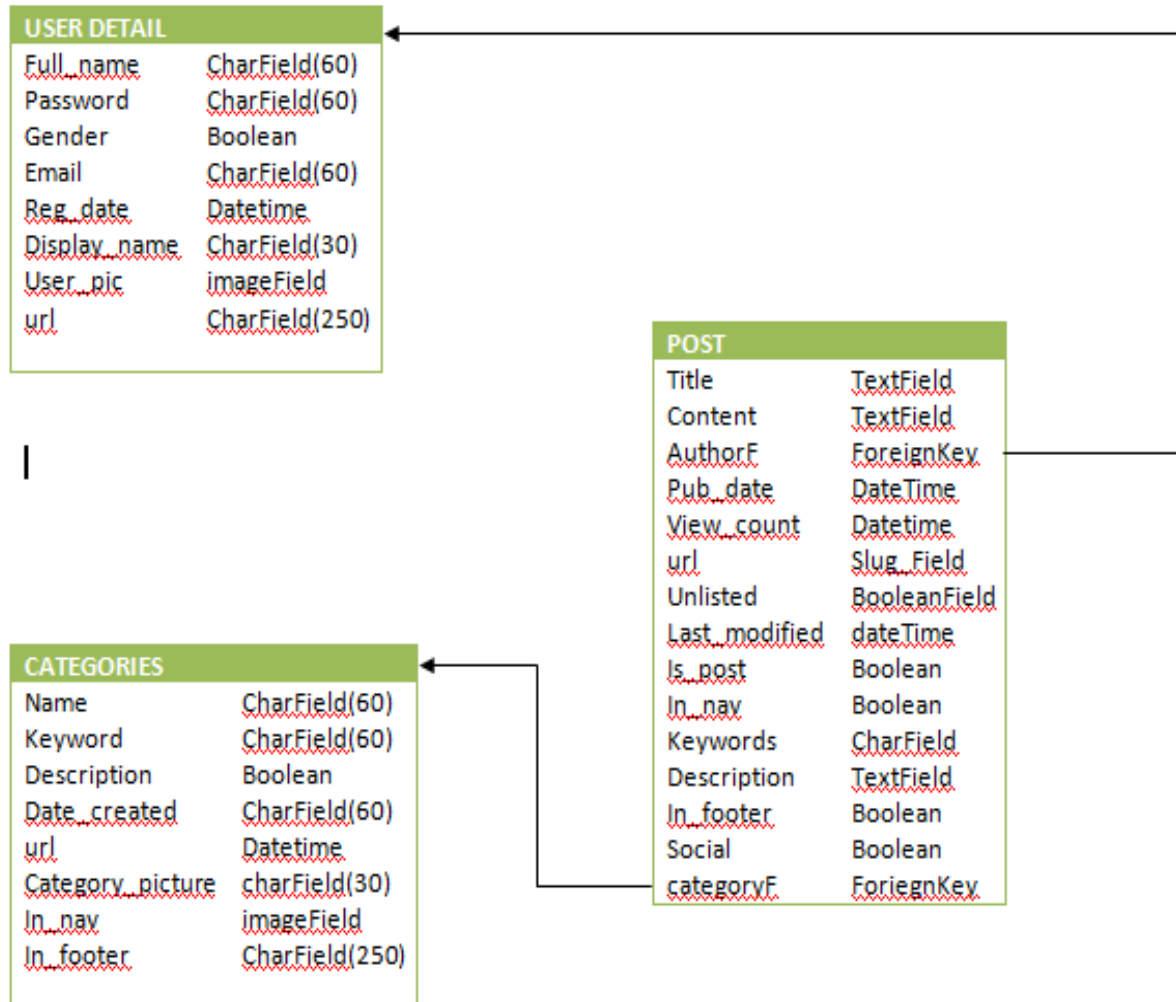
4.3.1 DATABASE ARCHITECTURE

The admin and the user interface use the same database engine. In order to manage the content of the site the user does not need to edit the HTML content or manually go to write the query, but he/she can login to the CMS and make the required changes. There are five tables required to be created in the database on the server where the application is hosted. Table 4.1 describes each of the tables used in this project.

Table 4.1. Simple CMS Tables and Descriptions

TABLE	Description
Users Detail	Stores user information: first name, last name, email, username, password and roleID
User Id	Stores userID and username
Category	Stores description of post and pages
Post	Stores the configurations of website like the template and the stylesheet used

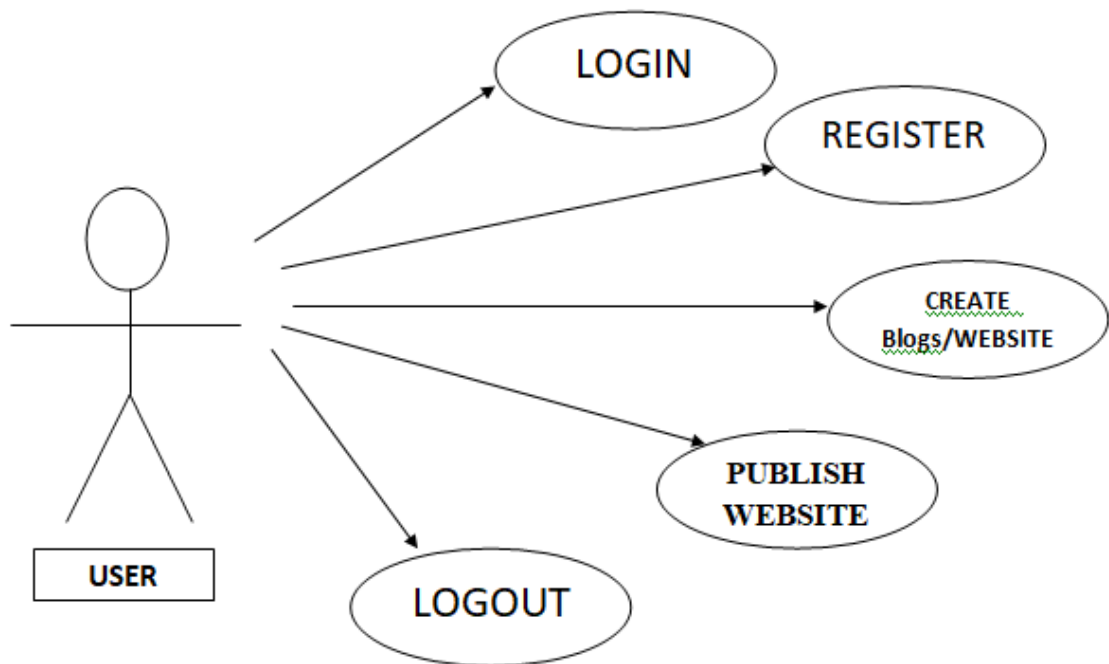
4.3.1.1 Simple class Diagram/ Database Architecture of sulphur:-



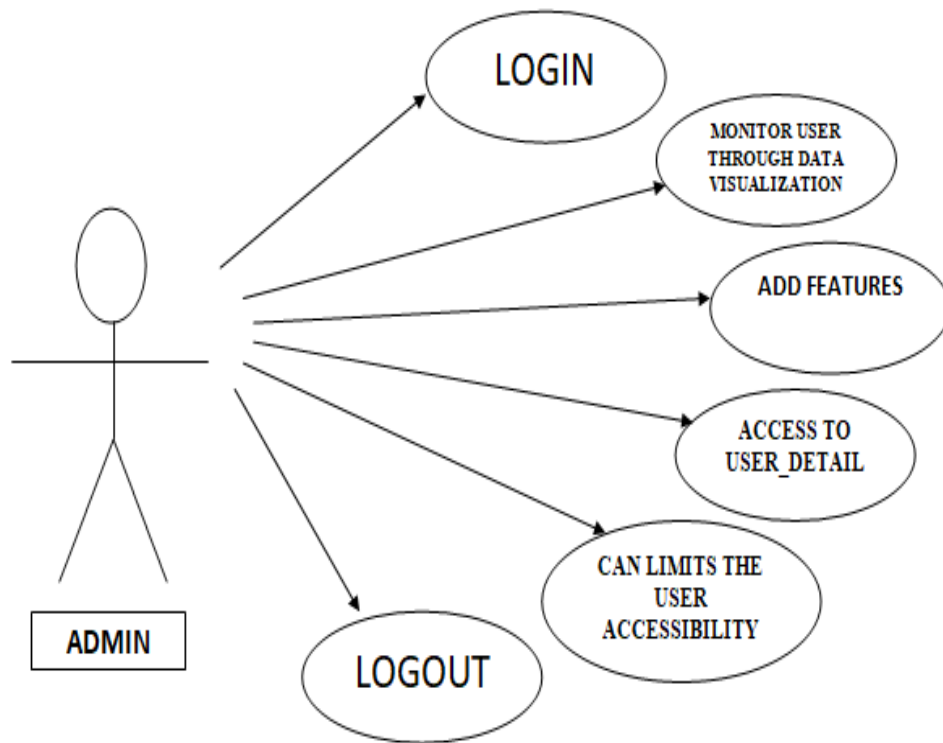
4.3.2 UML Diagrams

Unified Modeling Language (UML) is a standardized general-purpose modeling language .in the field of software engineering. UML includes a set of graphical notation techniques to create abstract models of specific systems.

4.3.2.1 Use-Case Diagrams

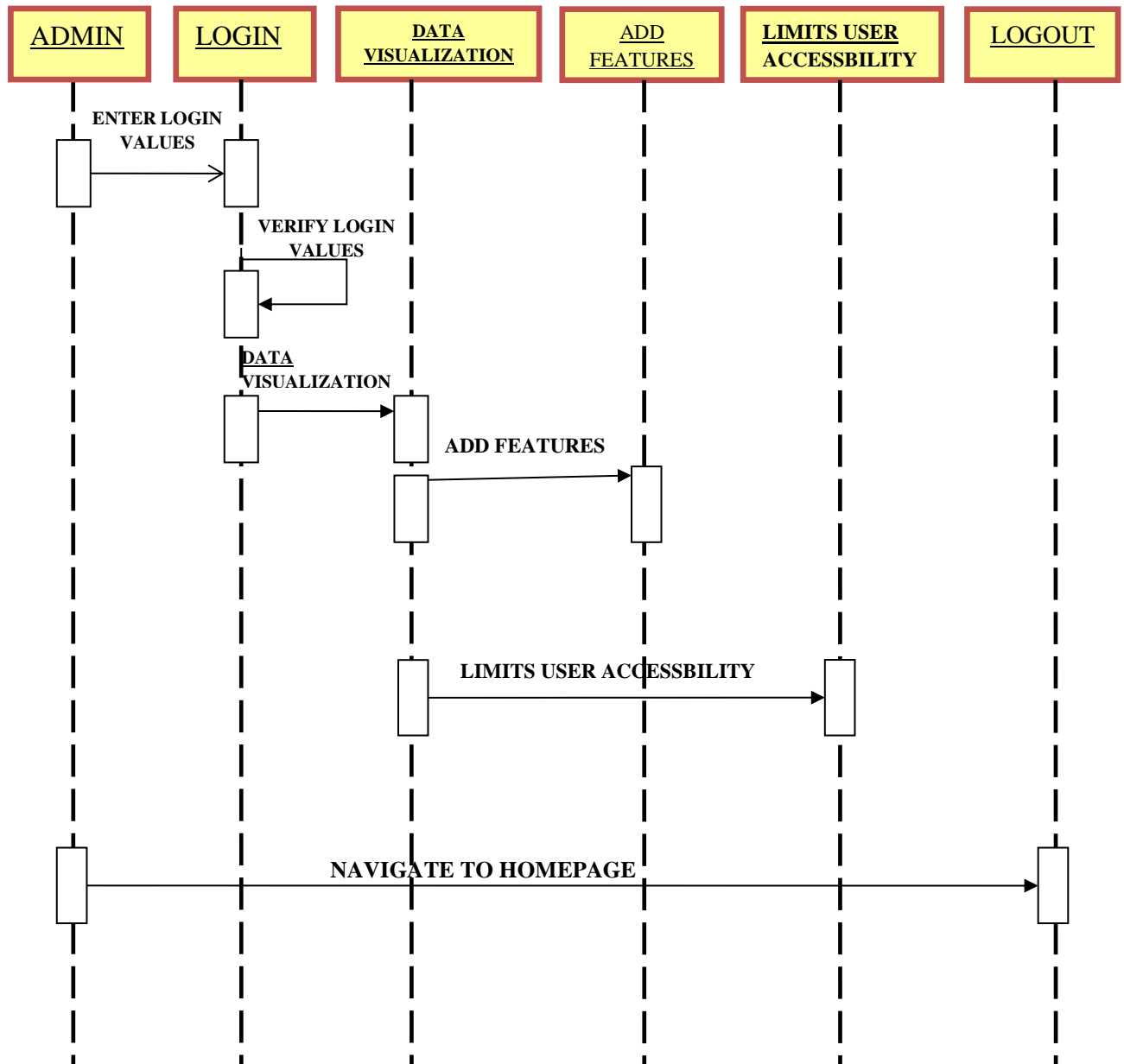


USE CASE DIAGRAM FOR USER

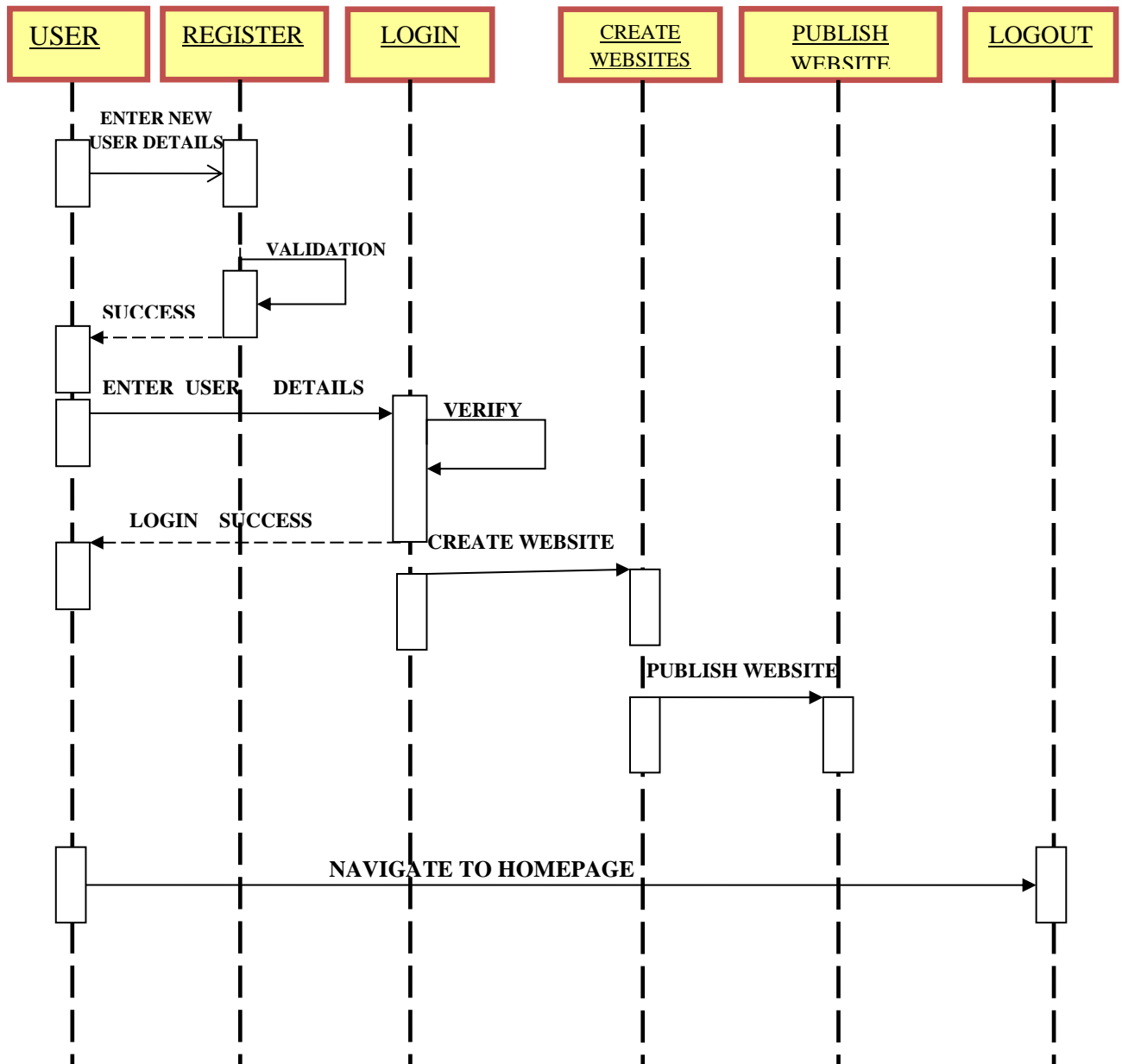


USE-CASE DIAGRAM FOR ADMIN

4.3.2.2 SequenceDiagram



Sequence Diagram for Admin



Sequence Diagram for User






4.4 Implementation












Set up development environment












1. Install and Set up Django Framework
2. Download Python 3.x and install then give the path of python in Enviroment Variables.
3. Install PIP---Pip is used to install python packages for command Prompt
4. Install and Setup MySQL









Start Project Development

1. start project using cmd.
2. Configure the setting in setting.py of your root project
3. create webapps in the project--- sadimn, blog
4. sadmin app is used to handling the registration, log in and logout process of admin and blog is used to handle all the processes relaed to blog like add post, edit post, preview, delete etc.

Name	Date modified	Type	Size
 blog	4/21/2018 3:08 PM	File folder	
 sadmin	4/21/2018 3:08 PM	File folder	
 sulphur	4/21/2018 3:21 PM	File folder	
 db.sqlite3	4/21/2018 3:08 PM	SQLITE3 File	0 KB
 manage.py	4/21/2018 3:08 PM	Python File	1 KB

 __pycache__	4/21/2018 3:35 PM	File folder	
 lib	4/21/2018 3:08 PM	File folder	
 templates	4/21/2018 3:08 PM	File folder	
 __init__.py	4/21/2018 3:08 PM	Python File	0 KB
 apps.py	4/21/2018 3:08 PM	Python File	1 KB
 forms.py	4/21/2018 3:08 PM	Python File	1 KB
 my.cnf	4/21/2018 3:08 PM	CNF File	1 KB
 settings.py	4/21/2018 3:13 PM	Python File	4 KB
 urls.py	4/21/2018 3:08 PM	Python File	1 KB
 views.py	4/21/2018 3:08 PM	Python File	4 KB
 wsgi.py	4/21/2018 3:08 PM	Python File	1 KB

 __pycache__	4/21/2018 3:35 PM	File folder	
 migrations	4/21/2018 3:08 PM	File folder	
 templates	4/21/2018 3:08 PM	File folder	
 __init__.py	4/21/2018 3:08 PM	Python File	0 KB
 admin.py	4/21/2018 3:08 PM	Python File	1 KB
 apps.py	4/21/2018 3:08 PM	Python File	1 KB
 forms.py	4/21/2018 3:08 PM	Python File	1 KB
 models.py	4/21/2018 3:08 PM	Python File	1 KB
 tests.py	4/21/2018 3:08 PM	Python File	1 KB
 urls.py	4/21/2018 3:08 PM	Python File	1 KB
 views.py	4/21/2018 3:08 PM	Python File	3 KB

 __pycache__	4/21/2018 3:35 PM	File folder	
 migrations	4/21/2018 3:08 PM	File folder	
 __init__.py	4/21/2018 3:08 PM	Python File	0 KB
 admin.py	4/21/2018 3:08 PM	Python File	1 KB
 apps.py	4/21/2018 3:08 PM	Python File	1 KB
 models.py	4/21/2018 3:08 PM	Python File	3 KB
 tests.py	4/21/2018 3:08 PM	Python File	1 KB
 views.py	4/21/2018 3:08 PM	Python File	1 KB

Coding Snippets

Below are some code samples of Project:--

Sadmin—views.py

```
from django.shortcuts import render, redirect
from .forms import loginform
from sulphur.lib.authentication import is_loggedin
from .models import User
# Create your views here.

def sadmin(request):
    if is_loggedin(request):
        return redirect('dashboard/')
    else:
        return render(request, 'sadmin/login.html', {'loginform' : loginform})

def login(request):
    # Checking if the user is already loggedin
    if is_loggedin(request):
        return redirect('../dashboard/')
    else:
        # Validating the login form came from login.html
        logindetails = loginform(request.POST)

        if logindetails.is_valid():
            # Extracting the data from login form
            data = {
                'username': logindetails.cleaned_data['username'],
                'password': logindetails.cleaned_data['password'],
            }
```

```

        data['password'] = sha256(data['password'].encode('utf-
8')).hexdigest()

    # Getting the record by usernam
    try:
        u = User.objects.get(username = data['username'])
    # sending to login if it is wrong
    except User.DoesNotExist:
        return render(request, 'sadmin/login.html', {'loginform':
loginform, 'message': 'Username is incorrect',})

    if m.password == data['password']:
        # logging in
        request.session['active'] = True

        # Adding the current logged in data into the session
        request.session['username'] = u.login
        request.session['full_name'] = u.full_name
        request.session['gender'] = u.gender
        request.session['email'] = u.email
        request.session['reg_date'] = u.reg_date
        request.session['display_name'] = u.display_name
        request.session['last_modified'] = u.last_modified
        request.session['user_pic'] = u.user_pic

        return redirect('/sadmin/')
    else:
        return render(request, 'sadmin/login.html', {'loginform':
loginform, 'message': 'Password is incorrect'})

def dashboard(request):
    return render(request, 'sadmin/dashboard.html')

```

Sulphur---setting.py

```
"""
Django settings for sulphur project.

Generated by 'django-admin startproject' using Django 2.0.3.

For more information on this file, see
https://docs.djangoproject.com/en/2.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/2.0/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'kbnw0_ia3&8wku%db0vm)oi1xo3+_j*z3+=nl0(cfup21ohdf('

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'sulphur.apps.SulphurConfig',
    'blog.apps.BlogConfig',
    'sadmin.apps.SadminConfig',
```

```

'django.contrib.admin',
'django.contrib.auth',
'django.contrib.contenttypes',
'django.contrib.sessions',
'django.contrib.messages',
'django.contrib.staticfiles',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'sulphur.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'sulphur.wsgi.application'

```



```

# Database
# https://docs.djangoproject.com/en/2.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'sulphur',
        'USER': 'varun',
        'PASSWORD': 'password',
        'HOST': 'localhost',    # Or an IP Address that your DB is hosted on
        'PORT': '3306',
    }
}

# Password validation
# https://docs.djangoproject.com/en/2.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/2.0/topics/i18n/

```

```

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'Asia/Kolkata'

USE_I18N = True

USE_L10N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.0/howto/static-files/

STATIC_URL = '/static/'

```

Blog---models.py

```

from django.db import models
from ckeditor.fields import RichTextField
from ckeditor_uploader.fields import RichTextUploadingField
# -----
class Category(models.Model):

    # Category Name, like games, tech, movies etc.
    name = models.CharField(max_length=200)

    # Date Created
    date_created = models.DateTimeField()

    # URL to a particular category page
    url = models.SlugField()

    # A profile picture for the caetegory

```

```

category_picture = models.ImageField()

# Keywords to be in HTML meta tag
keywords = models.CharField(max_length=200)

# Description to be in HTML meta tag
description = models.CharField(max_length=200)

# Set if it is in navbar
in_nav = models.BooleanField(default=0)

# Set if this link inside the footer
in_footer = models.BooleanField(default=0)

# Set if the category is not suppose to shown in the website
unlisted = models.BooleanField(default=0)

def __str__(self):
    return self.name

# -----
class Post(models.Model):

    # This is the title of the post
    title = models.CharField(max_length=200)

    # This is the main contentof the post
    content = RichTextUploadingField(blank=True)

    # This is the ID of the author of the post which is related to the model User
    # in the app admin
    author = models.ForeignKey('admin.User', on_delete=models.CASCADE)

    # DateTime on which the post is published
    pub_date = models.DateTimeField()

    # It is the number of views
    view_count = models.PositiveIntegerField()

```

```

# URL to a post
url = models.SlugField()

# Set if the post is not supposed to be shown
unlisted = models.BooleanField(default=0)

# Date of post last modified
last_modified = models.DateTimeField()

# SET if this is a static page. If this is SET so
is_page = models.BooleanField(default=0)

# SET if this is in navbar
in_nav = models.BooleanField(default=0)

# SET if this is in footer
in_footer = models.BooleanField(default=0)

# keywords in HTML meta tag
keywords = models.CharField(max_length=200)

# description in HTML meta tag
description = models.CharField(max_length=200)

# SET if it is a social link
social = models.BooleanField(default=0)

# This is the ID of the category of the post which is related to the model
Category in app blog
category = models.ForeignKey('Category', on_delete=models.CASCADE)

def __str__(self):
    return self.title

```

4.5 Testing

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. The user tests the developed system and changes are made according to their needs. Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform.

When a system is developed, it is hoped that it performs properly. In practice however some errors always occur. The main purpose of testing an information system is to find the errors and correct them. A successful test is one that found an error. The main objectives of the system testing are,

- To ensure during operation the system will perform as per specification.
- To make sure that the system meets user requirements during operation.
- To verify that the controls incorporated in the system function as intended.
- To see that when correct inputs are fed to the system and the outputs are correct.
- To make sure that during operation, incorrect input processing and output will be deleted.

Test Cases

Test case 1: Unit testing in login form.

- Test Objective: Validating the login form.
- Description: In this form the user enters the username and password. If the username and password are correct then the login is successful and user enter into his corresponding page else error message appears.
- Test Environment: django/python test framework
- Action: User enters the correct username and password.
- Result: Login Successful
- Action 2: User enter invalid username and password.
- Result 2: Error message: User doesn't exist.
- Action 3: Username –Correct Password : Incorrect
- Result 3: Error Message: Password is incorrect.

Test Case 2: Unit testing in Registration form.

- Test Objective: To validate the user details.
- Description: In this form user enters his Details. All fields that are compulsory must be filled. If these compulsory fields are not entered then the dialog box “All fields are mandatory” appears or in case of any incorrect/invalid details an error message appears .
- Test Environment: django /python test framework.
- Action: User doesn't enter all the fields.
- Result: Registration Unsuccessful.

Test case 3: Integration Testing

- Test Objective: Checking the connectivity of all the forms.

- Description: Connecting all the forms and verifying its functionality.
- Test Environment: django /python test framework.
- Action: All forms are connected properly
- Result: Project Executed successfully.

Test case 4: Black box testing for admin's change password form.

- Test Objective: To validate the constraints imposed.
- Description: In this form when admin tries to change the password without providing the confirm password, the password cannot be changed with an error "Invalid password".
- Test Environment: django /python test framework.
- Action: User doesn't enter the confirm password.
- Result: Password is not changed.

Test case 5: White box testing for User's details.

*What is White Box Testing?

White box testing refers to a scenario where, the tester deeply understands the inner workings of the system or system component being tested. This method is named so because the software program, in the eyes of the tester, is like a white/transparent box; inside which one clearly sees.

The key principles that assist you in executing white box tests successfully are:

- Statement Coverage – ensure every single line of code is tested.
- Branch Coverage – ensure every branch (e.g. true or false) is tested.
- Path Coverage – ensure all possible paths are tested.

Advantages

- Testing can be commenced at an earlier stage. One need not wait for the GUI to be available.
- Testing is more thorough, with the possibility of covering most paths.

Disadvantages

- Since tests can be very complex, highly skilled resources are required, with a thorough knowledge of programming and implementation.
- Test script maintenance can be a burden if the implementation changes too frequently.
- Since this method of testing is closely tied to the application being tested, tools to cater to every kind of implementation/platform may not be readily available.

4.6 Output Screens

View of All Posts

Hello Siraj

Posts

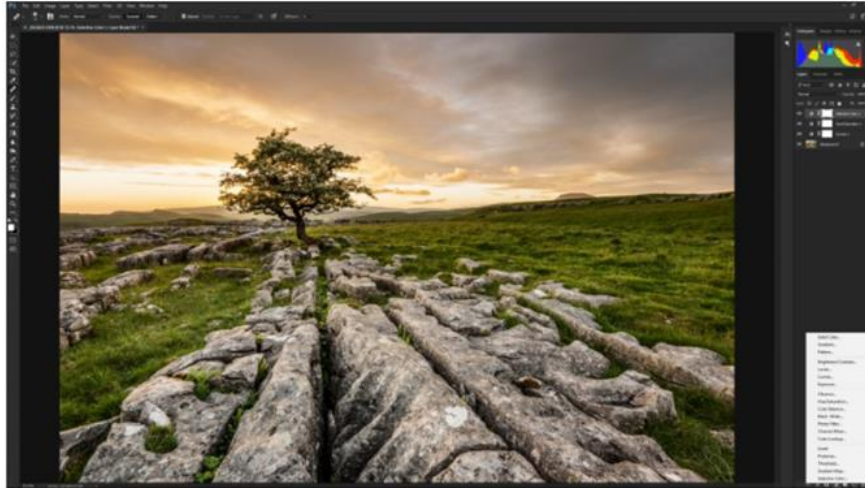
Add New Post

Title	Description	Category	Author	Date
SQL Injection delete edit unpublish	SQL injection is a code injection technique that might destroy your database.	Network Security	Siraj	April 22, 2018, 11:10 a.m.
Cross-Site Request Forgery delete edit unpublish	What is CSRF and how to avoid CSRF attacks	Network Security	Siraj	April 22, 2018, 12:33 p.m.
10 Photoshop editing skills every photographer should know delete edit unpublish	Adobe Photoshop has long been the industry standard image-editing software for photographers	Photography	Siraj	April 22, 2018, 12:46 p.m.

View of a Single Post

April 22, 2018, 2:16 p.m. By Siraj

How Photographers edit their photos



Adobe Photoshop has long been the industry standard image-editing software for photographers. It's a comprehensive and powerful program, and despite its reputation for being complicated, it can actually be used with relative ease by photographers of all ability levels.

The possibilities it opens up are almost endless; from basic adjustments to high-end retouching to creating surreal composite images, Photoshop can do it all. So, whether you intend to make major adjustments to your shots or just minor tweaks, these 10 tips will help you do it faster and more effectively.

1. Using adjustment layers

Adjustment layers are the professional way to apply edits to your images. Adjustment layers sit above the Background layer (your original image), and allow you to make multiple adjustments without altering the original image or degrading its quality. Adjustment layers are accessed by clicking on the half-white, half-black circle icon at the foot of the Layers panel. If you want to save an image with adjustment layers intact you'll need to save it as a TIFF or PSD – a JPEG is a flattened and compressed file type that doesn't support layers.

2. Converting to black and white



4.7 Conclusion

Main objective of Sulphur is, providing a tool which is Easy to use and efficient.

Which provides all hands-on features to the user to build their Website/ Blog without bother for programming and scripting to manage backend and frontend.

The development of Sulphur is divided into four phases

- Research
- Analyzing
- Implementation
 - ➔ Backend
 - ➔ Frontend
 - ➔ Deployment
- Rendering

In conclusion we feel that overall we have been very successful in project development. This is mainly due to the fact that the final project looks aesthetically pleasing and has met all the criteria as desired. As well as this, everything used in my project was entirely created by us. By creating everything ourselves, we have furthered my skills as software developer and have created an overall successful design, which in our opinion makes us feel that this unit has been a success.

4.8 Future Enhancement

- Adding different themes and thus user can choose theme according to his requirement
- User can set a previous publish date for a post, if he want
- Listing of post in different format like latest to old, old to latest, most viewed to less viewed, less viewed to most viewed and author wise and category wise.
- Registration and log in using Google account and Facebook Account
- Like, Dislike Options for a post
- Visitor account registration feature
- Comment on a post