Web Development 2 - Project Marking Guide

# Goal

The goal of this assignment is to create a CRUD-based [Content Management System](http://en.wikipedia.org/wiki/Web_content_management_system) (CMS) for a fictional client using a variety of the technologies you have studied this term. The type of web site or web application you create is up to you. You may wish to use your project from Web Development 1 as a starting place.

Once your CMS is deployed as a client website, your client should never have to contact you to make changes to the site’s content. An example of a simple content management system [can be seen here](http://stungeye.com/school/cms/). The admin link is hidden in the top right-hand corner of the header (hover your mouse there). The login credentials are ghostface / killa.

Before you begin your work please read over [How to CMS, a guide to starting your first content management system](https://docs.google.com/document/d/15LLz5WHd0qadikfH6cLnzz6dsdcNMgcjO16Co0CT7Dk/edit?usp=sharing).

# How Your Project Will be Marked

This document lists the possible features that can be included in the web application you are building for your WEBD-2006 final project.

**Each feature that you implement is worth a percentage of your project mark. Features are categorized into three levels of difficulty, worth 1%, 2% and 5% per feature.**

**Failing to implement a feature marked with a star 🌟 will result in a 1% deduction.**

For example, a student would be awarded a final project mark of 79% if they:

* Completed three 1% features (3%)
* Complete nineteen 2% features (38%)
* Complete eight 5% features (40%)
* Fail to implement two starred features. (-2%)
* Met all project milestones.

The project itself is worth 30% of your grade in this course. Time management will be an important factor in your grade for this project. You should strive to complete at least one feature every day so that you are not swamped with work by the end of the term.

# When Projects Will be Marked

You will have at least one opportunity every week for in-class marking of your assignment. The final version of your assignment must also be submitted to Learn by **Wednesday, April 20th** at 11:50pm. The final in-class marking sessions will be held on **Thursday, April 21st**.

**Your project mark will be based *only* on marks you receive during in-class marking.**

During an in-class marking session you will demonstrate your project’s features to your instructor. For each feature demonstrated your instructor will determine if that feature will be marked as completed or not. In order for a feature to be considered complete, you must have spent sufficient time and energy on its implementation. When in doubt, check with your instructor.

With this marking process you are accumulating marks throughout the process, or in video game terms you are levelling up your mark. After any of the in-class marking sessions you will

know your current project mark.

# The List of Possible Features

*Features with a* ***green feature number*** *are subject to extra clarifications found at the end of this document.*

Above each group of features the type of user the feature applies to is listed. It is important that you pay attention to these details. A feature implemented for the incorrect type of user will not be considered complete.

1. **Project Planning & Project Management**

*As the project manager you should be able to:*

* 1. Submit a project proposal and have it approved by your instructor. [5%] 🌟
  2. Meet or exceed the milestone goals listed in this document. (Deduction for missed milestones detailed along with milestone dates in the clarifications at the end of this document.)

## Content Management System CRUD

You are building a content management system that will generate a website consisting of a number of pages. Page data should be stored in one or more database tables. What constitutes a page will depend on the type of CMS you are building and the data you are gathering.

*As an administrative or logged-in user I should be able to: (Worth 5% Each)*

* 1. Create new pages by entering the required data into an HTML form. 🌟
  2. View a list of pages that already exist in the system with the ability to view the list sorted by title, by created at date, and by updated at data. 🌟
  3. Create a list of categories that apply to the pages in your system and assign categories to existing pages.
  4. View and moderate (delete or [disemvowel](https://en.wikipedia.org/wiki/Disemvoweling)) comments submitted by non-admins. (Assumes you have implemented 2.9.)
  5. Edit and delete the data associated with an existing page. 🌟
  6. Make uses of a [WYSIWYG](https://en.wikipedia.org/wiki/WYSIWYG) editor when adding or editing page data.

*As a non-administrative user I should be able to: (Worth 5% Each)*

* 1. Navigate the pages of the system by way of a menu, list or table. 🌟
  2. Navigate the pages by way of their associated categories.
  3. Comment on specific pages and have those comments show up along with the page.
  4. Cookies or Sessions are used to present commenters with a [CAPTCHA](https://en.wikipedia.org/wiki/CAPTCHA) to verify that they are human before they are allowed to submit.

## Content Search

*As a user of the website I should be able to: (Worth 5% Each)*

**3.1.** Search for specific pages by keyword using a search form.

**3.2.** Search for specific pages by keyword while limiting the search to a specific category using a dropdown menu.

**3.3.** Search results are [paginated](http://www.smashingmagazine.com/2007/11/pagination-gallery-examples-and-good-practices/).

## Validation and Security

*As the CMS programmer I should have: (Worth 1% Each)*

**4.1.** Implemented validation rules that are used on the data provided when creating and updating pages. 🌟

**4.2.** Sanitized and validated the numericality of all ids retrieved from GET or POST parameters used in SQL queries. 🌟

**4.3.** Sanitized all strings retrieved from GET or POST parameters to prevent HTML injection attacks. 🌟

## Layout and Design

*As the website designer you should be able to: (Worth 2% Each)*

**5.1.** Create valid markup and CSS for all pages on the website.

**5.2.** Design a consistent look and feel for all pages on the website.

**5.3.** Build your markup and styling around a CSS framework like Bootstrap.

*As the website designer you should be able to: (Worth 5% Each)*

**5.4.** Create page [permalink](https://en.wikipedia.org/wiki/Permalink) URLs that include ids and are SEO friendly.

**5.5.** Create permalinks that are “super pretty” by way of Apache mod rewriting.

## Image Uploads and Processing

*As an administrative or logged-in user I should be able to: (Worth 5% Each)*

**6.1.** Add an optional image to a page by way of a form upload. 🌟

**6.2.** Remove an associated image from a page.

**6.3.** Images are automatically resized when uploaded.

*As a non-administrative user I should be able to: (Worth 2% Each)*

**6.4.** View the image associated with a page when navigating pages. 🌟

## Administrative Logins

*As the website designer you should ensure that: (Worth 2% Each)*

**7.1.** Only admins can perform admin CUD tasks. 🌟

*As the website designer you should ensure that: (Worth 5% Each)*

**7.2.** Usernames & passwords are stored in a users table with CRUD admin access.

**7.3.** Passwords stored in the user table are hashed and salted.

**7.4.** Cookies / Session are used to remember when users have already logged in.

## Javascript, DOM, XML and JSON

**8.1.** One of the following technologies has been used in a non-trivial way:   
 (Worth 5% Each)

* + - DOM
    - AJAX
    - XSD
    - XSLT
    - JSON
    - XML
    - jQuery, or other language/technology acceptable to your instructor

## Deployment and Dependency Management

*As the website programmer you should ensure that: (Worth 5% Each)*

**9.1.** You can deploy your CMS to a shared host like [DreamHost](http://dreamhost.com), a VPS like [Digital Ocean](https://www.digitalocean.com/), or a cloud system like [Heroku](https://www.heroku.com/) or [AWS](https://aws.amazon.com/).

**9.2.** Your project makes use of the [Composer package management system](https://getcomposer.org/) and you are using at least two [Composer packages](https://packagist.org/).

# Requirement Clarifications

*Requirements marked above in* ***green*** *are subject to the following clarifications.*

**1.1.** You ***must*** submit a project proposal and have it approved by your instructor in order to receive ***any*** marks on your projects. If you miss the submission deadline for the proposal you will still need to submit one before you begin your project.

**1.2** The project milestones are:

* Project mark of 15+ on or before the week of March 29 - April 1.
* Project mark of 40+ on or before the week of April 4-8.
* Project mark of 65+ on or before the week of April 11-15.

**Each milestone that you miss will result in a cap on your maximum project mark.** If you miss a single milestone, the maximum amount of project marks you can accumulate will be 90. If you miss two milestones you will not be able to “level-up” your mark beyond 80. If you miss all three milestones your mark will be capped at 70.

**2.1** Your database must include at least 10 pages involving real data. No key mashing or any variations of lorem ipsum allowed.

**2.10** The student must have implement the CAPTCHA system using PHP along with session or cookies. Embedded CAPTCHAs, like [reCAPTCHA](https://www.google.com/recaptcha/intro/index.html), will not receive marks.

**2.3** If you cannot think of categories that fit your data please discuss this with your instructor. Having a similar 1-to-many table association in your project may qualify you for these marks.

**2.6** You need not worry about WYSIWYG editing when you first start the project. Once you’ve got your CRUD working you should research a Javascript WYSIWYG library and enhance both your new and edit forms so that they support WYSIWYG editing. A few JS WYSIWYG libraries: [TinyMCE](http://tinymce.moxiecode.com/), [CKEditor](http://ckeditor.com/), [Summernote](http://summernote.org/), [WYSIHTML5](https://xing.github.io/wysihtml5/), [Quill](http://quilljs.com/).

**5.4** Search Engine Optimized (SEO) permalinks are URLs that not only contain IDs but also contain keywords relevant to the page’s content. Imagine you’ve got a page about kitty cats and that page data resides in your page table at primary key id #5.

Non-SEO-friendly URL: <http://example.com/animals.php?id=5>

SEO-friendly URL: <http://example.com/animals.php?id=5&a=kitty-cat>

[Googlebot](https://support.google.com/webmasters/answer/182072?hl=en) now sees a kitty cat in your URL. A URLcat. Your pages data table will need a new column for this extra keyword data. I usually name this column permalink or slug.

**5.5** S*uper pretty* SEO-friendly URLs can be built using Apache’s URL rewriting. For a real-world example see the URLs of [the six candidate pages listed here](http://winnipegelection.ca/electoral_races/2/mayoral-city-of-winnipeg-2014).

A rewritten version of the above SEO URL: <http://example.com/animals/5/kitty-cat/>

[URLcat](https://i.imgur.com/jwI5yqK.png) approves.

Mod Rewrite [Tutorial 1](http://code.tutsplus.com/tutorials/an-in-depth-guide-to-mod_rewrite-for-apache--net-6708) & [Tutorial 2](http://www.sitepoint.com/apache-mod_rewrite-examples/).

**6.1** When an image is uploaded it must be tested for “image-ness” as shown in the course notes before it is moved to the file upload folder. Uploads that do not pass this test will be gracefully rejected.

**6.3** Image resizing must be done by PHP, not using CSS. The filesize of the image should change as a result of the resizing.

**8.1.** Please go over your idea to use one of these technologies with your instructor before you begin coding it. Your instructor will let you know if what you are proposing is considered to be a “non-trivial” use of the technology. The DOM and Jquery marks are mutually exclusive, meaning you can only get one or the other. For the DOM and JQuery marks a student must have implemented an entire non-trivial feature themselves using Javascript or JQuery.