Web Development 2 - Project Marking Guide

Goal

The goal of this assignment is to create a CRUD-based 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.

Before you begin your work please read over How to CMS, a guide to starting your first content management system.

Example CMS Sites

An example of a simple content management system can be seen here. The admin link is hidden in the top right-hand corner of the header (hover your mouse there). The login credentials are ghostface / killa. This example CMS does not implement all the features listed in this document.

A more complex CMS can be seen here at the Bored Game Geek site, a clone of the BoardGameGeek website. This site is also not complete and there is no password on the admin functionality.

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-2008 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 🌟 will result in a 1% deduction.

Reminder! A mark of at least 50% on the project MUST be obtained to pass the course.

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 50% of your final 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 in-class marking sessions will be held the last 2 hour period(s) of the semester (depending on your section). Please check Learn for the final submission due date; *you also must submit your final project in Learn.*

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. It is your responsibility to come prepared to a marking session with a list of the features you wished to have marked. Please ensure your mark tracking spreadsheet is up-to-date. In order for a feature to be considered complete, you must have spent sufficient time and effort on its implementation. When in doubt, check with your instructor.

For features listed with a green number you should read over the extra clarifications listed for that feature at the end of this document before you ask to have it marked.

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.

Please don't wait until the last moment to show your instructor your progress; once our scheduled class time for project marking for the week is finished, that's it -- time's up! If you haven't had your project features checked off by your instructor to achieve the marks for the milestone, then you will have missed the opportunity.

It's best to have your work checked bit-by-bit as you go, a feature or two at a time. Don't pass up the opportunities to meet with your Instructor during the earlier classes in the week.

Tracking Your Progress

A spreadsheet has been built to allow you to track your progress on the project.

The progress tracking spreadsheet can be seen here.

You will make a copy of the spreadsheet in one of two ways:

1) **Recommended** - If you have a Gmail account you can use the “Make a Copy” action in the file menu to save an editable version of the spreadsheet to your Google drive.

2) **Not Recommended, but should still work** - You can use the “Download As” action in the file menu to save an Excel version of the spreadsheet to your computer. The Excel conversion will not be perfect and may be glitchy, which is why it’s not recommended.

Tracking your progress using this spreadsheet requires that you update the following two spreadsheet columns:

- **To Do** - Use this column to track your progress on the tasks associated with each feature. Mark them complete by changing the “To Do” value from “No” to “Yes”.
- **Complete** - When you have completed all the tasks associated with a feature, *mark the feature as as “**Maybe**”* completed. Once your Instructor confirms that you have completed the feature, set the “Complete” value to “Yes”.

The spreadsheet calculates the project features that are marked as “Maybe” and “Yes” as “Verified” and “Unverified” marks at the top (in row 1). The deductions for required features which have not yet been completed, as well as missed milestones, are also calculated and displayed.

The marks listed in this spreadsheet are not official. Your Instructor records the official project grades. It is important to keep your spreadsheet up-to-date, and have it ready for project marking sessions.

Project Milestones

The project milestones are:

1. Completed requirements worth 15+ marks during or before the final marking session in week 11, November 9-13.

2. Completed requirements worth 30+ marks during or before the final marking session in week 12, November 16-20.

3. Completed requirements worth 50+ marks during or before the final marking session in week 13, November 23-27.

4. Completed requirements worth 75+ marks during or before the final marking session in week 14, November 30-December 4.

5. Final marking will take place during the final week of regular classes, December 7-10. (Final exams begin on December 11th.)

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 four milestones your mark will be capped at 60.

Second chances. If you missed a milestone you can recoup the loss of 10 marks by getting 10 marks over 100 (per milestone missed). For example, if you missed Milestone 1 but received 110 marks on the project, you would get 100% on the project, not 90%. If all four milestones are missed, you would need 140 marks 😱 to achieve 100%.

The List of Possible Features

IMPORTANT: Full details of all feature requirements can be found in the progress tracking spreadsheet mentioned above. We've left the written versions of the requirements below, but you should always consider the spreadsheet as the authoritative source for feature requirements.

*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:

- **1.1.** Submit a project proposal and have it approved by your instructor. [5%] 🌟
- **1.2.** 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.)

2. 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. For example, for the Bored Game Geek site each boardgame is a page (i.e. a row in the Game table) and the boardgame attributes (name, description, number of players, etc) are the page data.

You *must* use the PHP Data Objects (PDO) extension -- not MySQLi -- for all database operations.

For features 2.1 through 2.6 you can initially implement logins in the same way as they were done on the blogging assignment. However, more marks are available in section 7 for building your own login system.

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

2.1. Create new page (insert) by entering the required data into an HTML form. 🌟
2.2. Edit (update) and delete the data associated with an existing page. 🌟
2.3. 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, or by updated at date. 🌟
2.4. Create a list of categories that apply to the pages in your system and assign categories to existing pages.
2.5. View and moderate (delete or disemvowel) comments submitted by non-admins. (Assumes you have implemented 2.9.)
2.6. Make use of a WYSIWYG editor when adding or editing page data.

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

2.7. Navigate the pages of the system by way of a menu, list or table. 🌟
2.8. Navigate the pages by way of their associated categories.
2.9. Comment on specific pages and have those comments show up along with the page. (One-to-many relationship to your "main" table is required; a different one-to-many relationship could also work -- please check with your instructor.)
2.10. Cookies or Sessions are used to present commenters with a CAPTCHA to verify that they are human before they are allowed to submit.

3. 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 results to a specific category of pages using a dropdown menu.
3.3. Search results are paginated.

4. Validation and Security (marked on last day)

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. 🌟

5. 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 URLs that include ids and are SEO friendly.

5.5. Enhance permalinks from 5.4 so that they are “super pretty” by way of Apache mod rewriting.

6. 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. (Reminder: A page is defined as a database-backed entity in your CMS.) 🌟

6.2. Remove an associated image from a page (from the database and the uploads/images folder).

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. 🌟

7. 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.
Login to the site using a username & password. (Cookies / Session be used to remember when users have already logged in.)

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

7.5. Register for a login account by providing a username and a password.

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

7.3. Passwords stored in the user table are hashed and salted. Login functionality must also be implemented that supports these hashed/salted passwords.

As an admin or non-admin user I should be able to: (Worth 5% Each)

7.4. *Login to the site using a username & password. (Cookies / Session be used to remember when users have already logged in.)*

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

7.5. *Register for a login account by providing a username and a password.*

8. AJAX, API, Email, Jquery, or Epic Failure

8.1 to 8.5 The following technologies have been used in a non-trivial way: (Worth 5% Each*)

-
- 3rd Party API
- PHP Generated Email (can use GMail)
- jQuery
- Advanced Coding Practices

You must receive approval from your instructor to proceed with any of these technologies before you begin implementing one of these features.

*At most you can implement **two** of these. Meaning, you'll only get either 5% or at most 10% from this section.

8.6. Epic Fail (I tried to build an epic feature and failed... and I can prove it.)

9. 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, byte.host, etc. or a VPS like Digital Ocean. (You need to show your application running on the host to your instructor.) You can also use this link to get a $10 Digital Ocean credit. (Full disclosure: If you use the second link and end up spending $25 on future hosting, the not-for-profit Open Democracy Manitoba will receive a $25 referral credit.)
9.2. Your project makes use of the Composer package management system and you are using at least two Composer packages.
9.3. Your project code is versioned using git source control with at least 20 commits in one or more branches. The repo must also be pushed to a private repo on Github or Bitbucket. 🌟

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.

2.1 Your database must include at least 10 pages involving real data. No key mashing or any variations of lorem ipsum allowed. Only admin / logged-in users can reach this form and create new pages.

2.3 There must be some indication of the type of sorting currently applied to the list.

2.4 Categories must be implemented using a separate categories table with a 1-to-many association with your pages table. Users must be able to CUD categories in this table using an HTML form. Assigning categories to pages should be done using a drop-down list box (HTML select element). 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 WYSIWYG editing must be added to your page create and update forms. The addition of a WYSIWYG editor should not break how pages are displayed. 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, CKEditor, Summernote, WYSIHTML5, Quill. Make sure your formatted text displays as such!

2.10 The student must have implemented the CAPTCHA system using PHP including a dynamically generated image and a verification system using session/cookies. Embedded CAPTCHAs, like reCAPTCHA, will not receive marks -- i.e. you have to build your own CAPTCHA.

5.1 & 5.2 These marks will be awarded in the final week only as they involve your entire website.

5.4 & 5.5 The two types of SEO permalinks are explained in this linked document.

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. You may use a plugin/script for this.

7.2 Just having a users table does not qualify you for this mark. You also need to have implemented within your application the ability for logged in admin users to be able to view, add, edit and delete users. In other words, user CRUD for admins. Being able to perform user CRUD from PHPMyAdmin does **not** count.

7.3 Password salting and hashing must be done using PHP's password_hash() function.

7.5 When registering for an account users should be asked for their username (which can be their email address) and the password. The user have to repeat the password within the same form. If the two passwords do not match they will be presented with an error message and asked to try again. Also, be sure to check that the username is not already in use!

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.

9.3. Your twenty commits need to spread out across the entire project timeframe. Each commit should be accompanied with a commit message detailing the intent of the changes you made. Commit message should be well-written complete sentences using proper grammar and punctuation.