**CPAN 253**

Project

**Due Date:** 20/04/2018

Create a Content Management System in ASP.NET with either Web Forms or MVC.

Each group should have at least 3 features. A feature can be tier 1, tier 2 or tier 3 (shows the level of complexity of the feature). See below on “What is a feature?” for examples of features.

What is a feature? In summary, you can add specific things to a web site that would increase its value to the client and visitors. These "things" are features. Each feature must have a CRUD (Create, Read, Update, Delete) operation otherwise it is not a feature. See below on “What is a feature?” for more information.

In your Content Management System, you should include an admin side.

In your project also include:

* A login page
* The use of layouts/master pages
* Responsiveness
* A data access component to execute the commands against the database table.

Marking:

* User interface
  + Professional / Clean
  + Easy to navigate / Usability
  + Consistent layout / Cross-browser compatible (think of the audience)
  + Relevant content/images
  + Responsive
* Database design
  + Proper naming conventions for tables and columns
  + Normalized tables
  + Table relationships
* Code + Feature
  + Tier 1
  + Tier 2
  + Tier 3
* Peer review
  + See Peer Evaluation handout

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**What is a feature?**

# Introduction

We are building complex database driven Web sites that are also known as a content management system or CMS. The database stores information that will be presented to the visitor via a browser. The CMS resides on the server and using a scripting language like PHP or ASP.net gives our clients the ability to control the information. A website that has CMS and provides enough features would directly benefit a client from adopting your plan and paying your fees. One of the things that you will think of is what are the specific things I could add to this Web site that would increase its value to the client, their visitors and clients, and that would be interesting to me. These “things” are features and that is what this paper is about.

# The importance of CRUD

CRUD is an acronym for Create, Read, Update, and Delete. It refers to what can be done to records in the database. Each feature must have a CRUD operation otherwise it is not a feature. The Interface for the feature may have many different components and interactions that increase the feature’s complexity but it all boils down to CRUD. In a similar fashion each feature must have a public space where the information is displayed and an administrative ( also known as admin) interface where CRUD operations are managed. Let’s take a look at an example.

## A frequently asked questions feature

Frequently Asked Questions (FAQ) is an enormously useful feature where Web site visitors can go to a section of the Web site and find the answers to questions that other visitors usually ask. It is useful for them to get their answers quickly and it is useful to the Web site owners to have those answers prepared so that the only questions they get are those that are unique to the one visitor or they have never heard before. The Web and the Internet have led this type of information architecture as a means of quickly onboarding new users to systems so that the discussions can move to interesting and new things rather than known or old things. See for example the Wikipedia page <https://en.wikipedia.org/wiki/FAQ> or the Internet FAQ archives at <http://www.faqs.org/faqs>. You can see that a complex business represented by their Web site would benefit from having a FAQ page for their visitors.

At its most basic level a FAQ has a question and an answer. So your database base would have three columns: primary key, question, and answer. The primary key is numeric, the question is probably a restricted length text field, and the answer is an unrestricted length text field. The public display of the FAQ feature would be all the questions and all the answers displayed in a list view: question on the top and answer on the bottom.

The admin side of the feature would represent the CRUD operations so that you would have options to create a new FAQ, edit an existing FAQ, or delete an FAQ. The admin would have a similar listing of the FAQs with each individual item in the list being a link to a form where you could see the current information and change it and a link called “Delete” to remove the record. Clicking on the link to edit the FAQ would take you to a form where the content for that FAQ would reside in the field. Here you could make your changes. A button called “Save” would be nearby where you would issue the command to update the particular record.

If you were creating a new FAQ there would be an “add new” link nearby and clicking that would take you to a form where the question and answer fields would be represented by form fields in html. You would then enter the information into the form fields. Again you would have a nearby button called “Save” that would create the record and add it to the database. On the public side the list view would add the record to the public view.

At this point you would have several “pages” that would show up on the Web site. There would be the public list page, an individual page for each question, a list view on the admin side (with CRUD features available), a similar individual page, and an “add new” page. This would be a tier one feature.

## Adding complexity

Getting this feature to work at this level is a very important step. Now you can begin to add complexity and begin to turn this from a tier one feature to a tier two or tier three feature. In fact, you may even begin to experiment with how FAQs are written and displayed - which would be interesting to say the least. What sort of complexity are we talking about?

It soon becomes apparent that for a complete FAQ we need more than just questions and answers: we would also need a category field in the database and a means of sorting the FAQs based on their category. We could also use a date field for the age of the FAQ as some questions may change over time. We could use validation to prevent someone from inserting empty records into the database. We could also add more visitor interaction by exposing the add new FAQ form to the public. For public facing forms we should have a means of preventing spam, for example CAPTCHA - <http://www.captcha.net> , Mollum, <https://en.wikipedia.org/wiki/Mollom>, or Honeypot, <http://www.dexmedia.com/blog/honeypot-technique/>, and similar validation to prevent empty or improper submissions. Now we would be getting to the tier two and tier three levels of complexity.

## Adding pages and adding navigation features

Since we are making a content management system - everything we can add to the database is content managed by our system. However, there is content that we need to be able to add and that is paged content. A page in this case would something like the about us page or the copyright page or anything that is a single amount of content. At the same time if we are adding this kind of content then we need to be able to add a link to that content in the navigation system. This would be two features - but integrated with each other and probably with other features in the system since they too might have to add links to the overall menu on the Web site.

### Add a new page feature

In figuring out this content item you need to determine what are the components of a page that you need to give your client access to. A page has a title, body content, navigation term, the ability to add images, and perhaps other documents like PDF files. Think of these as text fields in an HTML form that issues commands to the database. Your database now has several columns for each of these elements including a link to the menu system. This would be a tier one feature. Adding interaction and validation would probably start to put it into the tier two range.

### Add a link to the menu feature

The menu system is the Web site navigation that we see on every page. This is the essence of creating a scalable Web site. When we add content we have to have a means of adding navigation. For pages this would be the main navigation and if we were adding products we would add navigation to the products pages. We are creating a series of links that are the main navigation, which are the parent links and the secondary navigation that would be the child links. Your database would have to account for this parent and child relationship.

We can see a database of primary navigation, the parent, and subsequent tables with secondary navigation linked to the primary navigation and then linked to the individual page. These tables would be linked to the each of the pages on the Web site.

The database structure could look something like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Primary\_nav | Type |  |  |  |
| Primary\_Key | numeric |  |  |  |
| Foreign\_key | numeric |  |  |  |
| Nav\_word | text |  |  |  |
| Nav\_page | text |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  | Secondary Nav |  | Tertiary\_Nav |
|  |  | Primary\_key |  | Primary\_key |
|  |  | Foreign\_key |  | Foreign\_key |
|  |  | Primary\_parent |  | Secondary\_parent |
|  |  | Secondary\_Page |  | Tertiary\_Page |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  | Page |  |  |
|  |  | Primary\_key |  |  |
|  |  | Page\_title |  |  |
|  |  | Page\_content |  |  |
|  |  | Parent |  |  |
|  |  | Primary\_nav | true/false |  |
|  |  |  |  |  |

Please note: this is not a final database design. We would have to insure that it is optimized and functional for our needs.

Since this is a complex feature and is used by all pages it would be a tier three feature.

## Other forms of complexity

Validation increases the feature’s complexity because it involves more programming and increases the level of interaction between the user and the system. For example the user enters the data, the system checks the data, and if correct, writes it to the database. This involves several steps before it is completed. Now if there is an error then the systems stops the process and then has to send an error message to the user explaining the problem and how to fix it. Writing effective error messages is a requirement and is more work for the developer.

Moreover, in creating the wireframe for the feature - all required fields have to be marked. Usually, we use a red asterisk, \***,** and a note: **“**\* = required information”. Depending on the information we would also put in an example of correctly formatted information. In the case of the date - we might show YYYY/DD/MM or a telephone number (999) 555 5555. We are trying to make sure that fewer errors are entered into the database.

If the feature requires the use of math in the backend - then that too increases the complexity and therefore pushes the feature into tier two or tier three. An example of this might be the emergency room wait time feature. It would be a calculation based on the number of patients, the severity of their cases (represented by a numeric value), and a time estimate. The time is calculated and the results displayed on the page. The database information could also be evaluated over time to gauge how many people were seen in the ER and how fast they were seen.

Tier 1, 2, and 3 features:

Tier 1 feature:

* Database-driven (using LINQ)
* OOP
* It works as designed
* Error handling
* Usability
* Commenting code (all files)
* Labeling controls / Following proper naming conventions
* Professionalism and functionality
* Example: Product List / Details Page, Contact Form, F.A.Q.’s, etc.

Tier 2 feature:

* Database-driven (using LINQ)
* OOP
* It works as designed
* Error handling
* Usability
* Commenting code (all files)
* Labeling controls / Following proper naming conventions
* Professionalism and functionality
* Robustness and efficiency of controls used (not utilising the same controls constantly)
* Dynamic / Interactive
* External/advanced libraries
* Example: Image Gallery (pagination, by category, etc.), Events Calendar, Review / Polling System, Search, etc.

Tier 3 feature:

* Database-driven (using LINQ)
* OOP
* It works as designed
* Error handling
* Usability
* Commenting code (all files)
* Labeling controls / Following proper naming conventions
* Professionalism and functionality
* Robustness and efficiency of controls used (not utilising the same controls constantly)
* Dynamic / Interactive
* External/advanced libraries
* Independent initiative (going beyond what has been covered in the course)
* Example: Events Calendar with Bookings, Advanced Search, Transactions, Chat system