# Introduction

Any Web development project should start with a requirements document. What you are creating are the set of instructions a developer would follow in order to build the features you are describing. It is the scope of the project and includes all the information a developer would need to build it.

One of the key purposes of this assignment is to help you develop a mental model of what a content management system is. Mental models are a powerful set of expectations about how we see things in the world. For example: an automobile. Automobiles are ubiquitous in our world – there is no city in the world that has not spent untold billions of time, effort, and wealth, in making their city automobile friendly. Some of us can drive an automobile – all of us have been in one and have a good idea of how it works. Therefore, if I asked you to describe an automobile you would be able to do a decent job of it: even if you don’t drive. The same is true for this assignment and what we are doing in the program in general. However, most of you have never done this kind of work before and so your mental model needs to be developed. That is the purpose of you doing this assignment.

The individual elements for each feature description are as follows:

1. Title and version.
2. Narrative description.
3. Database schema.
4. Dataflow diagrams.
5. Wireframes.

As a set of documents this will be a blueprint that a developer would use to build this feature using the technologies specified in the requirements document itself. You are creating a set of documents for someone else and as a result you have to be clear and consistent in your work. I will go through each of the above elements below.

# Title and version

What is the name of your feature and what is the version of this document? For example, you could be working on an events calendar. The title would be:

Events calendar: version 1.01 (Bernie Monette)

Or a Frequently asked questions tool:

Frequently asked questions (FAQ): version 1.2 (Fred Smith)

Make sure the title is clear and your versions make sense to you and your team. If it helps you and your team to manage the process of creating and keeping this document up to date; you could also add the date. The date should certainly be included in the main document. Make sure to put your name in the title. This is your work after all.

# Narrative description

This is your feature in plain English: something anyone could understand. Here you will describe what the feature is and how it will work. You will write out the user stories that will provide the means of showing what your feature will do. User stories are a key element for knowing what the user will be able to do with your feature. For example, a user story for a frequently asked question feature would be:

“A logged in admin user can add a frequently asked question.”

Or for a job application feature:

“A registered user can apply for a volunteer position.”

If we break these stories down, we can see what needs to be in place for them to work. In the first story we have a “logged in admin user” so we will need a login system and a means of distinguishing different types of users. User permissions will also need to be in place. The second part “can add a frequently asked question” means that we have a form and a mechanism for adding a question. This might mean a link in admin side that says something like “Add new” or “Add new question”. HTML forms have fields of various types, validation, and code that interacts with the server and the database.

In the second story we have a “Registered user” which means that users can register and further confirms a mechanism for providing user rights and permissions. We can also infer a form to submit an application as well as a means of reviewing applicants – this is based on the word “apply”.

Remember too that all features have to have CRUD capability. Each element of CRUD is a user story. This section is also a good place to show what error and success messages you will have for your feature. You do want to have these system messages in place before you start programming. These messages are a key element of the user experience – we all know how impossible it is to use a system with messages that are not easy to understand. We are building features that our audience will want to use.

CRUD is a short form for Create, Read, Update, and Delete. These are the common database functions that are the basis for content management. Every feature for every Web site in the world will have some or all aspects of CRUD and yours will too.

# Database schema

In this section you will determine what information you intend to collect. You will also show the tables you will create to support your feature. In each table you will begin to denote which data fields are required by a setting of “not null”. This is in turn will be seen in the wireframes where you will mark fields that are required.

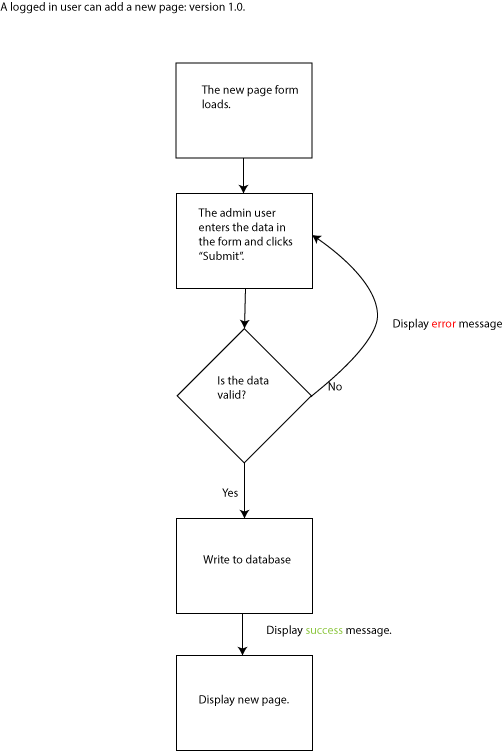
Excel is probably the best means to show your database tables – but any clear means of showing the tables, their connections, and their details is fine. Below is a possible schema for the MVP for a CMS driven Web site.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |  |
|  |  |  |  |  |

# Dataflow diagrams

We use these diagrams to explain how the feature will work. We model the interaction using boxes, diamonds, and arrows. In one sense each box is a page, the diamonds describe decision points, and the arrows show us the flow of the interaction. Each element of CRUD requires a dataflow diagram – these will be for primarily administrative functions. Although if your feature has public functions – then these will also require a dataflow. Within the dataflow you will also add messages, either success, failure, or status, to show what the message says and how it is triggered. Each dataflow diagram will start with a user story to show what the user will be able to do and to act as the test for that part of the feature. If the user can do what the story says in your application – then you know you are finished.

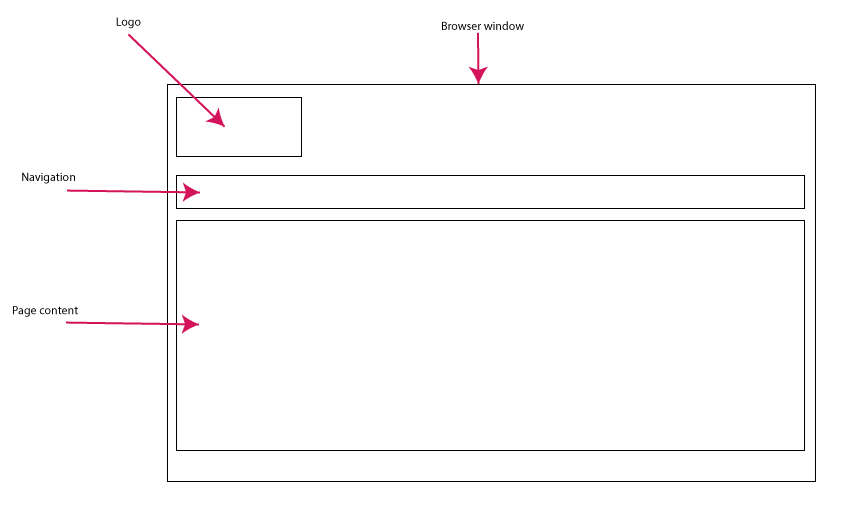
Take a look at the example below. This is a dataflow for the “create” element of CRUD for the MVP. The user story tells you that this is for an admin user who has logged in. There may be several routes to the “add new” functionality – this will depend on how you and your team have managed the admin dashboard for the CMS. The first step is “The new page form loads.” And from there the admin user can enter the data they have prepared. At this point they click “Submit” (which is something to be careful about – make sure you and your team have determined what buttons are to be called – so that everyone uses the same name.). Once the user clicks submit – then the data is validated – this is represented by the diamond. A “yes” response means to keep moving while a “No” requires the feature to take the user back – with an informative error message. Error messages are in red while success messages are in green. The interaction finishes with the new page displayed – or you can go back to the admin dashboard – whatever you and your team have decided.



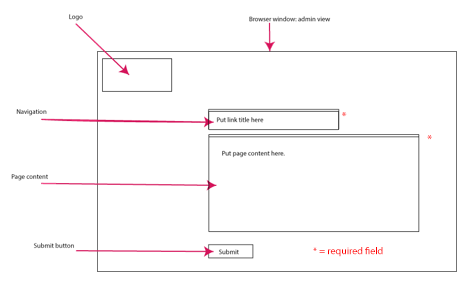
This is the dataflow for Create – read, update, and delete would be next. Read is simple – and doing a Read dataflow is not always needed – it depends on the interaction. Update would look identical – except this time the admin user would be pulling in the form filled in with the data to be edited. Delete is different – rather than a validation diamond you would have a confirmation diamond – this would be a status message (in yellow) asking the user if they were sure they wanted to delete the record. You can expect to have between 4 and 10 dataflow diagrams depending on the feature you are describing. You may also consider that a responsive or an accessible version might require a change in the interaction – if that is the case – dataflow diagrams reflecting responsive (or accessible) interactions may be needed.

# Wireframes

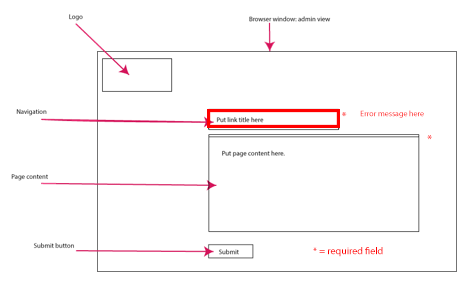
From a broad perspective the point of these documents is to go from very loose “descriptions” to very tight “requirements”: programming the feature itself is the most specific of all. Our narrative description is in English and is not very specific. Then our database is logical but not very useful in telling us how the feature will work: it is simply what data will be collected and how. The dataflow diagrams require us to be more exact in how the feature will work, what goes first, what goes next, and how error, success, and status will be handled. The wireframes are our first attempt to show how the feature will look. The wireframes will show forms, outputs, and views. The arrangement of the data from the database on the screen. They will also show how errors will be displayed and what fields are required.



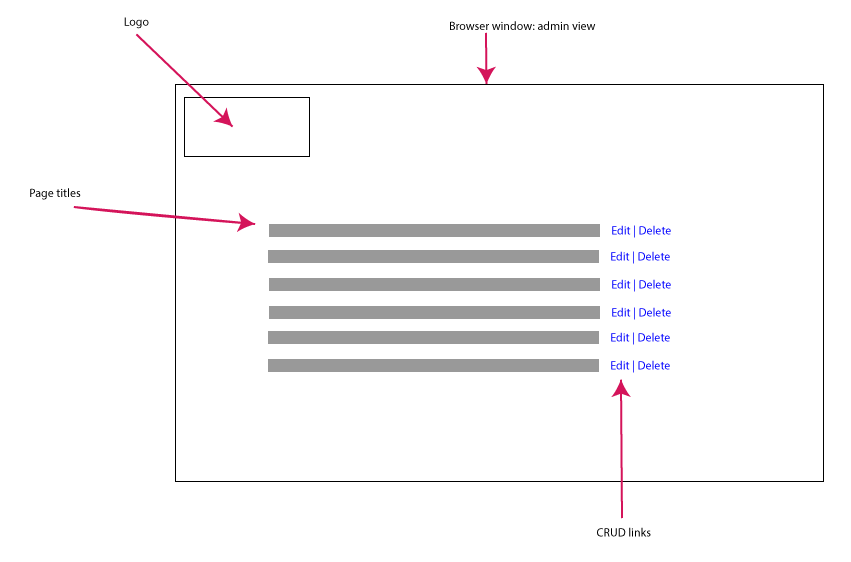
This is the Read view of a page of content.



This is the view of the html form that produces the Read view above. If the form is blank – then it is a Create wireframe and if the form is filled – it is an update wireframe.



Here is the same form with an error in place. You would include the error message in this wireframe.



This wireframe is a view of all the records available for the admin users. The gray boxes refer to the page titles and the links on the right are “Edit” and “Delete”. Clicking on “Edit” retrieves the data from the database and places it into the form. Clicking on “Delete” will trigger the delete functionality. You could either load the page and then confirm deleting it or just show the confirmation message.

Your wireframes will be more complex than these. With your team you will build a home page wireframe. This wireframe will hold most of the navigation and the layout for the Web site. Your feature will fit into the main wireframe.

You need to remember that you are not only writing this document for yourself but also for the client and any future developers who might be tasked with creating or maintaining it. Which leads to the main question I ask myself when reading your work: is this ready to go to the client right now? I don’t mean theoretically either. I am asking whether, as a professional developer, I can put your work in front of client and confidently use it as a means of saying: this the feature we have agreed to build. If I can’t do that – then your mark will be lower and you will have to make changes.

Today, you and your team will do the dataflow diagram to map out the main user activity for a Web site. Using user stories you will outline the steps a user would take to satisfy that action. You can use any software to create this diagram – but I must be able to open and read the file. Word is your safest bet. It is due at the end of class today or tomorrow before 5 pm.