

Intro to Content Management Systems



When you first start learning web development. You learn the basics such as HTML, CSS, elements (the building blocks of HTML pages), how to generate HTML pages and so on. This is all well and good to use if you want to keep a simple site and don't mind doing the extra work of maintaining HTML the old school way.

The thing is, most sites on the web do not work this way. Most sites on the web are

using some form of CMS system. If you have a client that wants a new website, chances are they want a CMS system and rightly so. With a CMS system, anyone can update, add, delete or move around content to the site without knowing how to code. CMS systems allow clients to focus on what it is they do (their own business) whilst streamlining the workflow of maintaining a website. It also is cheaper to run in the long run depending on what it is they are doing.



Similar to how you update a profile or account on social media. Social media platforms are essentially content management systems built for the masses to use.

There are different levels of CMS systems.

Small simple and basic, home made content management that can generate content within a HTML template.

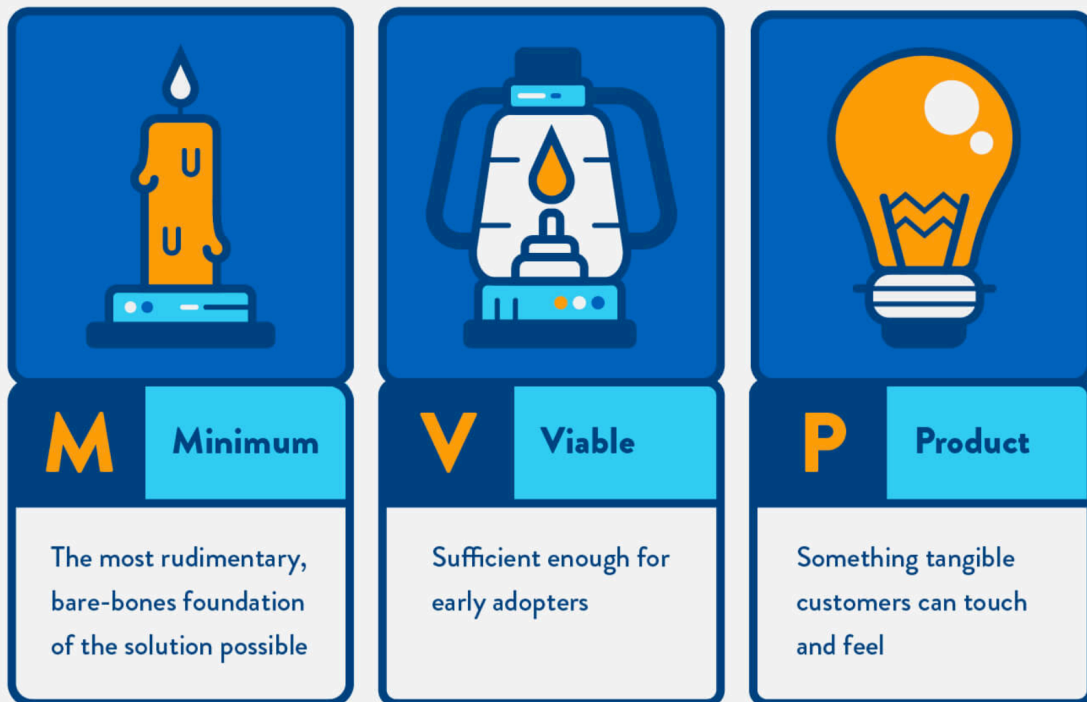
Med to Larger style (and in many cases open source) CMS systems that can accommodate small businesses all the way up to large corporate entities. This can include blogs/articles, e-commerce, archived content, networked content, social media platforms and more.

Enterprise solution (also in many cases built on open source) CMS systems that can be networked or geared to very large scale applications.

And everything in between.

Minimum Viable Product (MVP)

WHAT IS A MINIMUM VIABLE PRODUCT



When working on projects of such scale. We want to create a starting place. Would it make sense to build some vast project, not knowing exactly what we might need? What if we need to get user feed back on a beta before we know? What if the client does not fully know what functionality they will need?

They still want us to ship our code though...
And we still do not have a full picture of
what we need to actually build.

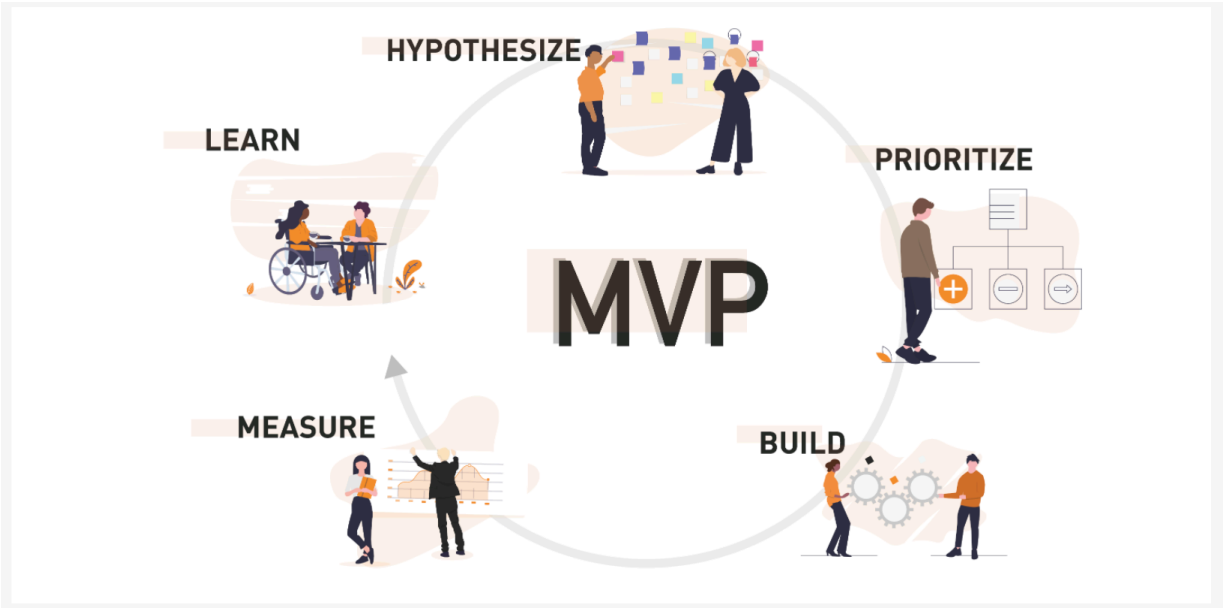
In this case we can build an MVP. The
concept of this is to build the most for the
least effort. This way we can build a base.

A number of core features that we know we
do need

whilst allowing us to allowing us to let
people actually use the platform

So that those users/client can give us
feedback

So that we can build more features



Determining the base

There are a number of ways to determine that the base or core features are. We could work with the client to figure that out. We could also have focus groups and so on depending on what it is we were attempting to build.

We know for certain that the client needs to be able to update content on a website. So based on what you understand about websites, what would be the least amount

work that can be done here? What are the most bare bones basic functions we need to have to make a working, testable product?

It should:

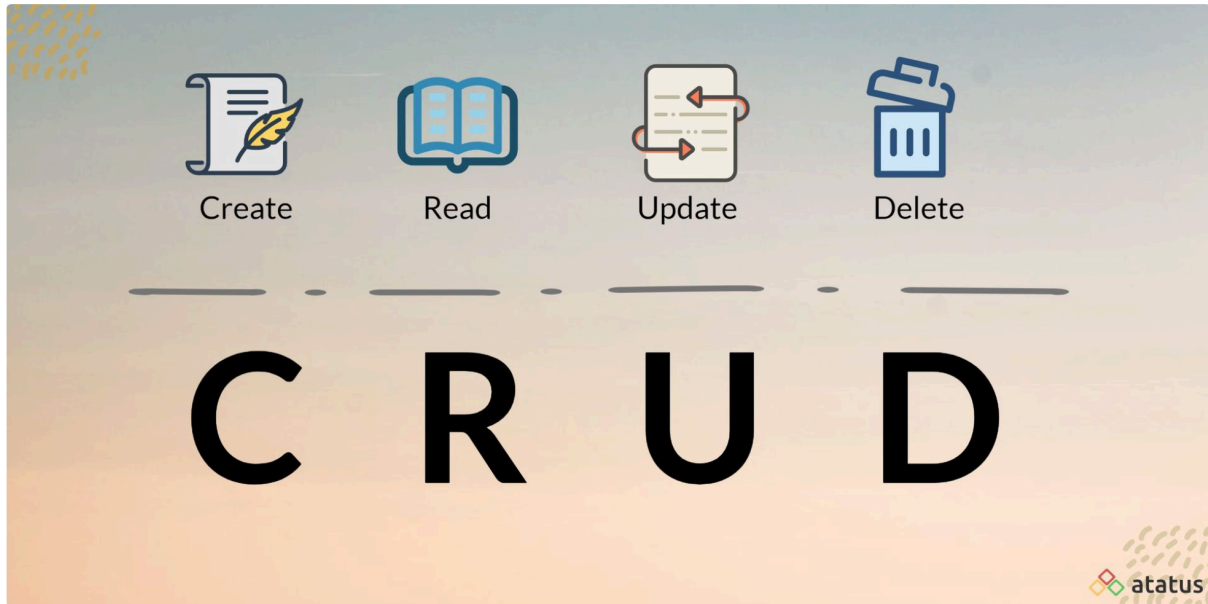
Have a page(s)

Have a link(s) to the page(s)

Have a means to add pages, delete pages and update pages

Have a database to store the content

Enter:



CRUD - In computer programming these are the 4 basic operations for persistent storage. Basically these are the 4 things we need to be able to do to have a viable content management system.

CRUD	HTTP
Create	POST, PUT if we have `id` or `uuid`
Read	GET
Update	PUT
Delete	DELETE

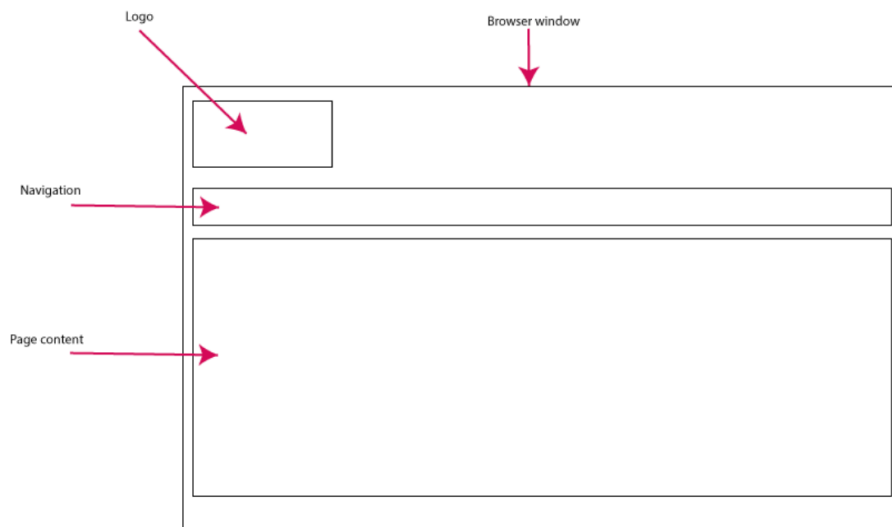
CRUD	SQL
Create	INSERT
Read	SELECT
Update	UPDATE
Delete	DELETE

Adding pages and adding navigation

Since we are making a content management system - anything we can add to the database is content managed by our system. However, there is core content that we need to be able to add and that is paged content. A page in this case would be something like the “about us” page or the “copyright page” or anything that is a single topic: content. Content has to be managed and we do that with CRUD.

What follows are wireframes showing different aspects of the main operational

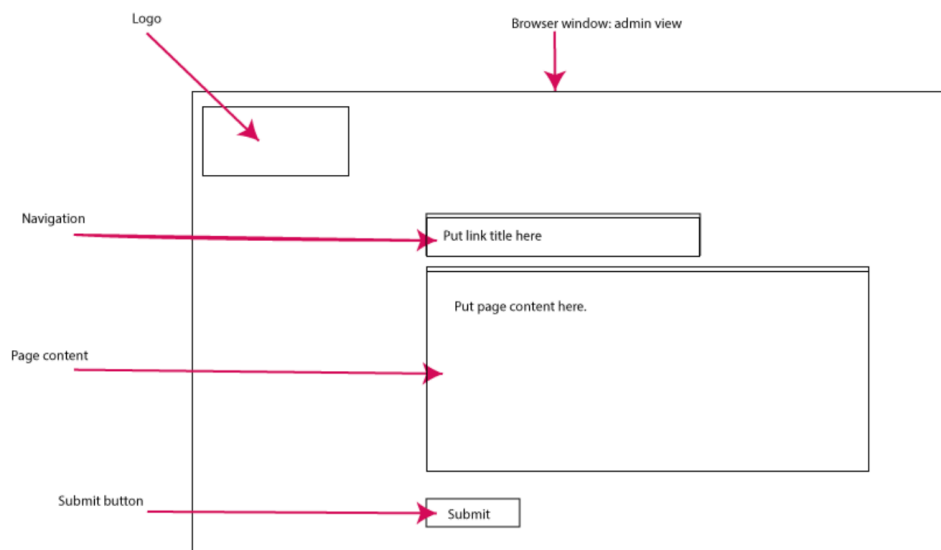
features of CRUD – Create, Read, Update, and Delete.



Front End/Public view – **R**ead of **C**RUD

When we add this content, we need to be able to add a link to this content in the navigation system. The image above shows you a wireframe version of the MVP we are talking about. The logo anchors and identifies the page, the navigation gives us all the links to the available pages, and the page content is for this page and link item.

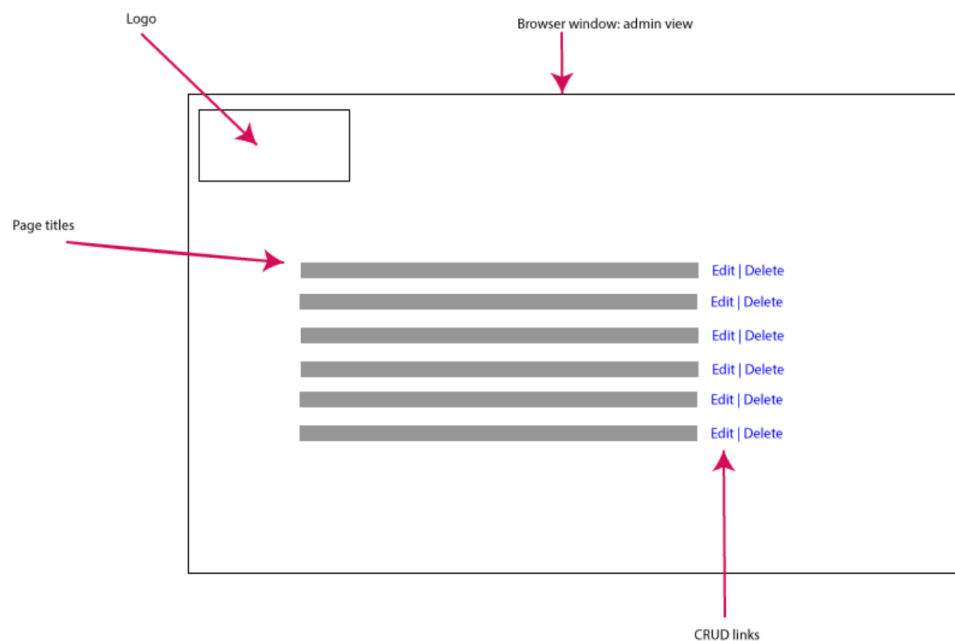
The links to the pages would go into the navigation region and the individual content would go in the larger page area. Input is managed using an HTML form – in fact this is where most of the programming will occur. It is with this form that CRUD operations are done.



Back End / HTML form view – **C**reate and **U**date of **C**RUD

There are two components to this MVP:
add a page, and add a link to that page.

This information would be managed by CRUD (Create, Read, Update, and Delete) operations. We would also need an output page that generates a list all of the pages entered into the database. With each page reference (a link to that page) would be options to edit or delete. At the bottom of this page would be a button or link to create a new page.



A listing of all pages – select either Edit or

Delete – Update or Delete in CRUD.

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. Since we are building the MVP we will only work with text for now. But as you can see – adding elements to the MVP is easy to do. We have to be careful that we do not add too many of these new elements as we would no longer be controlling what the MVP is.

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. For the MVP we would simply create the

primary navigation. For the purposes of this MVP only the primary navigation is

necessary.

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

Primary_key

Foreign_key

Primary_parent

Secondary_Page

Tertiary_Nav

Primary_key

Foreign_key

Secondary_parent

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 ensure that it is optimized and functional for our needs.

Overall, this is the least of what is needed for a content management system: pages and links. As well as a means to add new pages and add new links. Or to edit those pages and links.

Your assignment is: in 250 words or less describe either ebay, Facebook, or Youtube, as an MVP.