This is the last project for the course! It is worth 20% of your grade.

Section A: December 3th, 2019
Section B: December 6th, 2019

About Pages

There are many web engines which allow users to create and modify the webpages on their websites without needing to write any code. One common example is Wordpress. You will be creating your own version of a content management system which manages pages. A page is defined as having:

- A Page Title
- Some Page Text

However, one of the complications is that if we want to create new pages, that would require us to write new code. **Our objective is to build a data driven web application that allows users to create, read, update, and delete Pages which are stored in a database.**

Database

This is achieved by storing each page not as a file in the server side, but as a record in a database. Here is a look at a “Bare Bones” table built in MySQL

	Field	Type	Charset	Attributes	Null	Default	Extra	Action
☐	pageid	int(20)		UNSIGNED	Yes	*NULL*	auto_increment	
☐	pagetitle	varchar(255)			Yes	*NULL*		
☐	pagebody	mediumtext			Yes	*NULL*		

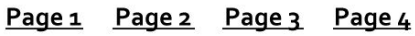
NOTE: The first step is to create a database system that your visual studio environment can connect to. Your teacher cannot provide a database which has EDIT/ADD/DELETE access.

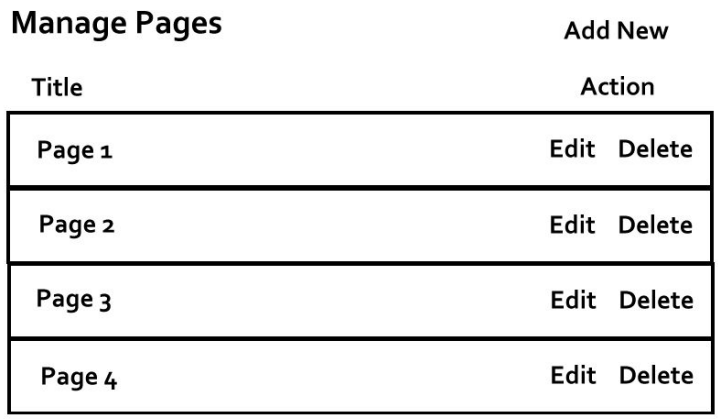
The following tools allow you to create a database on your local computer

- MAMP
- XAMMP
- MySQL Server (does not include PHP, Apache/Nginx, etc.)

Interface

Here is a look at a wireframe for an interface which manages static pages “List Pages”. It will be up to you to design the “New Page” wireframe, “Edit Page” wireframe, as well as the “View Page” wireframe.





The next part will be to use your page data to build a menu. We want to be able to write a reusable piece of code that generates an HTML menu (see the top of the wireframe.) In the rendered HTML (what the browser sees), our menu should look like this.

```
<ul>
    <li><a href="Page.aspx?pageid=1">Page 1</a></li>
    <li><a href="Page.aspx?pageid=2">Page 2</a></li>
    <li><a href="Page.aspx?pageid=3">Page 3</a></li>
    <li><a href="Page.aspx?pageid=4">Page 4</a></li>
</ul>
```

You will have to write a user control which programmatically creates the list item HTML and display it to the web page. See the user control example in “crud_essentials” repository (FeaturedClasses.ascx, FeaturedClasses.ascx.cs).

Algorithm

We've seen how algorithm is the "glue" layer between the Interface and the Query. An algorithm for this webpage should be able to

- Pull information from the interface
- Call the correct Queries
- Show the results (effect) of the Query

Your algorithms should support associating the interfaces of List, Show, Update, New, Delete with the Methods defined in the "Database" class.

Query

You will create a "Page" class which does the following:

- Contains Page Title as a field
- Contains Page Content as a field

You will create a "Database" class which does the following:

- Contains connection string properties
- Has a method which allows the programmer to query page data (returns List<Page>)
- Has a method which allows the programmer to select individual page data (returns Page)
- Has a method which allows the programmer to update a page (returns nothing)
- Has a method which allows the programmer to delete a page (returns nothing)
- Has a method which allows the programmer to add a page (returns nothing)

See SCHOOLDB.cs for an example at a class which does this. Note that SCHOOLDB serves a return type of Dictionary<String, String> for an individual record and List<Dictionary<String,String>> for a result set of records. This is a generic return type. You will be required to make your "Database" class methods return a specific "Page" object (individual data), or a list of "Page" objects (query result set data).

Minimum viable product

The minimum viable product is a strategy which focuses on the bare minimum of an application before building it further. Here is the list of the expected features.

- Can see a list of existing pages in the database (Under “Manage Pages” in the wireframe)
- We can can add pages to the database through the web app
- We can remove pages from the database through the web app
- We can edit pages in the database through the web app
- We can interact with a menu that is dynamically generated from the pages in the database
- When we click a menu item we can see the read the page content.
- Website is given a consistent template structure with the user of Master Pages and Content Pages.
- Contains basic validation on inputs for creating and editing pages

Extra Features

Building the application as it’s described above would be considered the minimum viable product. But how can we extend this further? Here are a list of features that I can think of. If you can think of any, ask me and I can help you think up an implementation. Beware -- Some features are easier than others.

- Feature which shows the date that the page was published
- Feature which allows users to “publish” and “unpublish” pages (unpublished pages cannot be viewed and don’t show up in the menu).
- Feature which changes the menu to a different design on mobile view
- Feature which allows users to select a page “Look”, which changes the CSS on that page
- Feature which allows users to specify an author name
- Feature which creates a different CRUD for “Authors”, and when creating/editing a page, we can specify from a list of “Authors”
- Feature which allows users to specify “Main column 1” and “Main column 2” content text, so that the page view shows each piece of content on a separate column
- The ability to search for pages based off page title

Grading Rubric

	Level 1 (0-25%)	Level 2 (25-50%)	Level 3 (50%-75%)	Level 4 (75%+)
Interface	- No wireframe designs - Interface is shaky and not linked correctly - No use of styling/layout	- Wireframe designs are unclear - Interface is not linked in one area - Attempt at layout	- Wireframe designs are concise and include a concept for up to 3 features - Utilizes data driven interfaces - Decent layout, needs improvement in some areas, inconsistent	- Wireframe designs are very accurate and include an attempt up to 5 features - Executes data driven interfaces - Layout is consistent and clean
Algorithm	- Algorithm does not extract information from the Interface. - Algorithm does not call the correct queries from the database	- Algorithm extracts partial information from the interface. - Algorithm calls some queries from the database.	- Algorithm extracts the correct information from the interface. - Algorithm calls most of the correct queries from the database.	- Algorithm extracts correct queries from the interface. - Algorithm calls the correct queries from the database.

Query	No database class used to connect	Database class is mostly the same as the in class example, there was no effort to work with it	Database class is similar to the example. Attempts to utilize the page class. Comments describe the thinking process	Database class is referenced from the example but is unique for the solution. Comments describe the entire thinking process.
MVP	MVP is missing major components. Fewer than 4 MVP elements completed.	MVP is mostly working, but is not very robust. Fewer than 6 MVP elements completed.	MVP is working well. One extra feature implemented.	MVP is working. Robust design prevents improper user inputs. Two extra features are implemented.