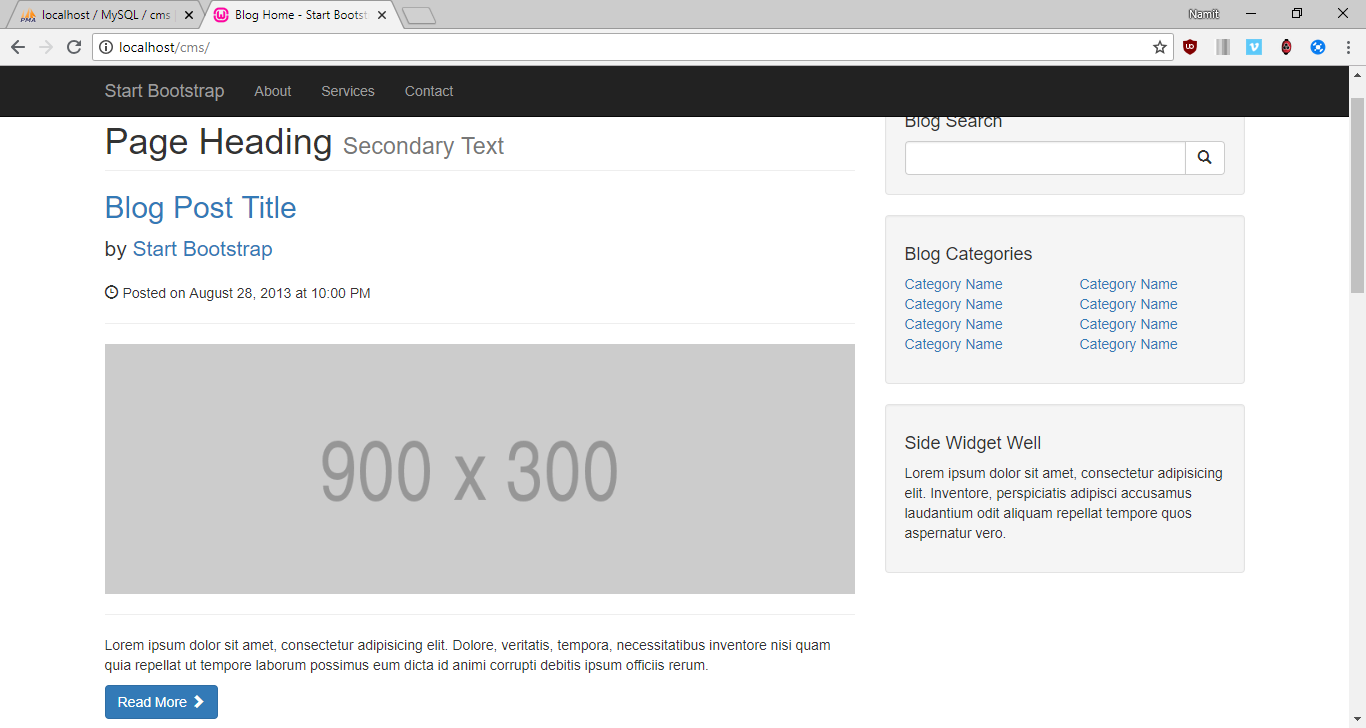
1.) in wamp, go to PHP tools=> phpInpfo-> Set the following:  
a.) display\_errors = NO  
b.) output\_buffer = 4096 Bytes

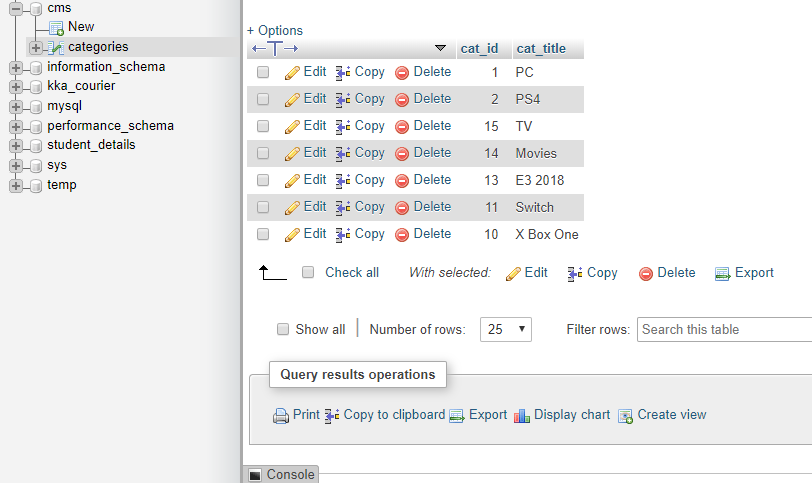
2.) Get a bootstrap template for

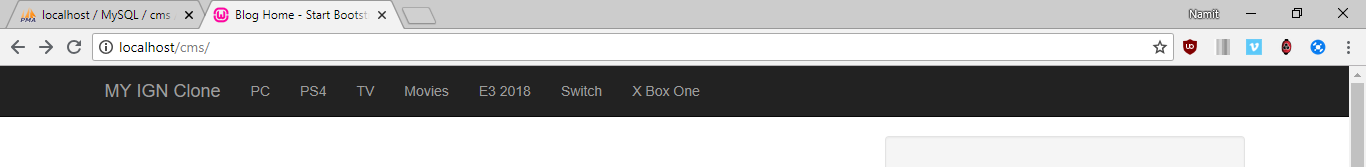
a.)cms:



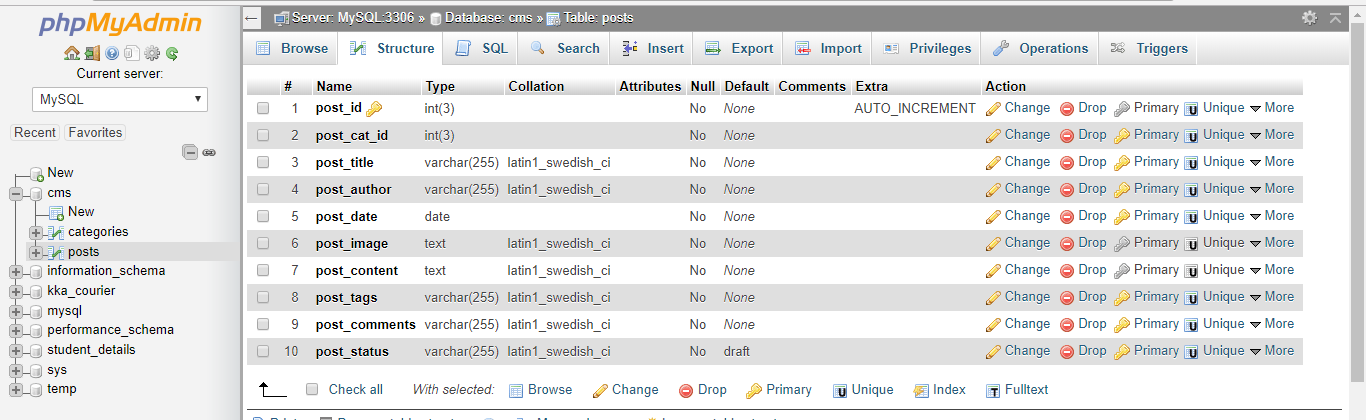
b.)As well as Admin:

3.) Build a database in localhost/phpmyadmin called CMS, and tables such as:  
 a.) Courier : having 2 columns   
 i.) cat\_id(Int) = Primary Key, autoIncrement  
 ii.) cat\_title(Varchar) = length : 255

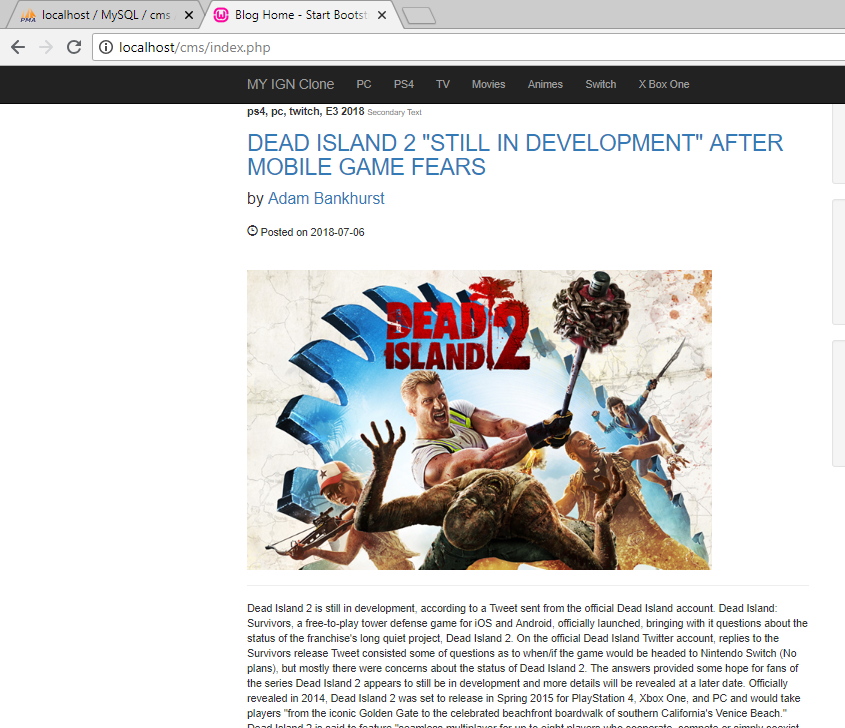
The cat\_title will have the following structure, such that we pull the information from our mySQL database and then we display it withing the TOP NAVBAR with PHP syntax  
  


Such that, the TOP NAVBAR looks like the following:

Now To Enable the Main Posts to be made Dynamically in the website, we create another table called posts



After entering information in the Posts inside the database, we fetch the posts columns from the Database into our website, which now looks like this:



4.) **Making Search Engine:**

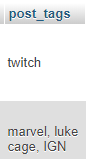
In order to construct a search engine, we need to create an input form with the “POST” type HTTP request method, along with that, we shall name the buttons, input spaces accordingly.

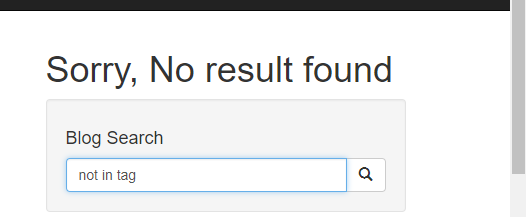
The Search form: 

The Search Engine works as following: Such that whenever we pass an input inside the above text field, we try to match or compare the entered keys with the information already inside our database, which would in turn find the required string and present that in the webpage.

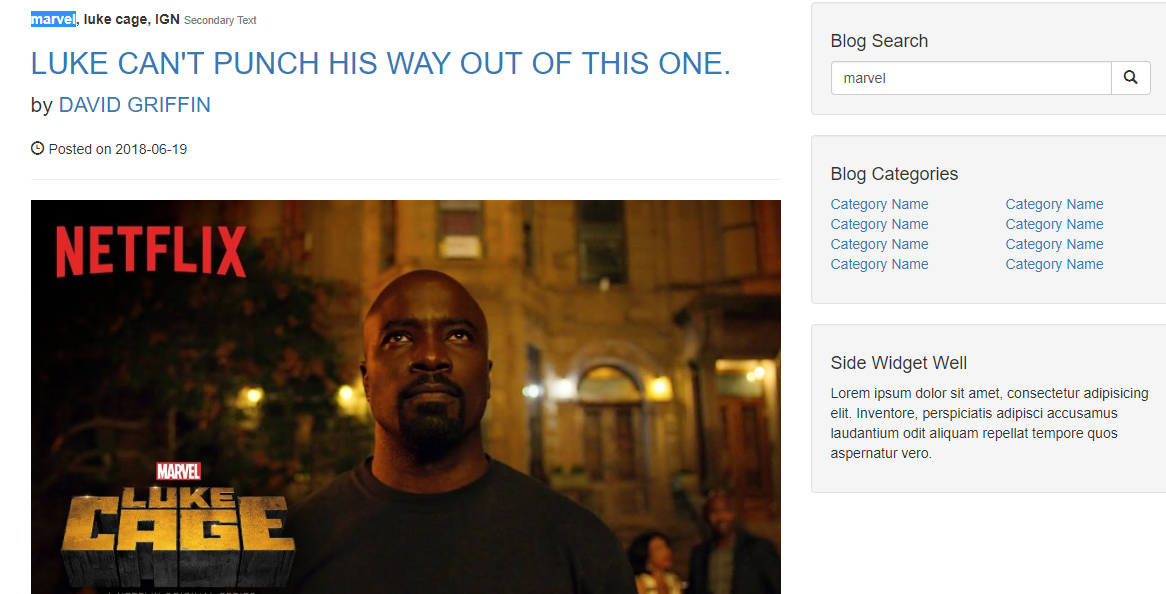
So for that, an SQL Query is used which will be: **“ SELECT \* FROM posts WHERE post\_tags LIKE ‘%$search%’ ”;**

The contents of post\_tags column inside the database looks as follows:



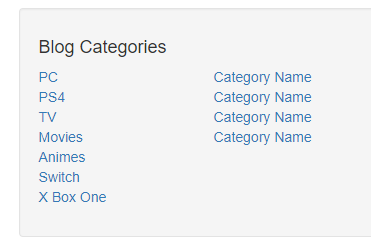
And if there are no such rows obtained, then with the help of an SQL function ie “ **mysqli\_num\_row()** “,   
we can throw a Result not found error message:

So, the functionality of the search engine should be that it must display just one individual Post, which contains the ‘Searched’ keyword in it’s table’s column, such that: “Observe the contents of the Search box and the highlighted text, which thereby concludes that our Search Engine works.

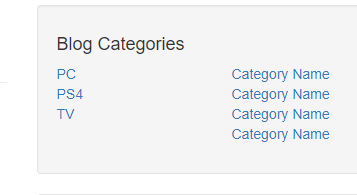


5.) **Sidebar UI:** The Sidebar UI is the component of the webpage, which displays the “BLOG CATEGORIES” section,

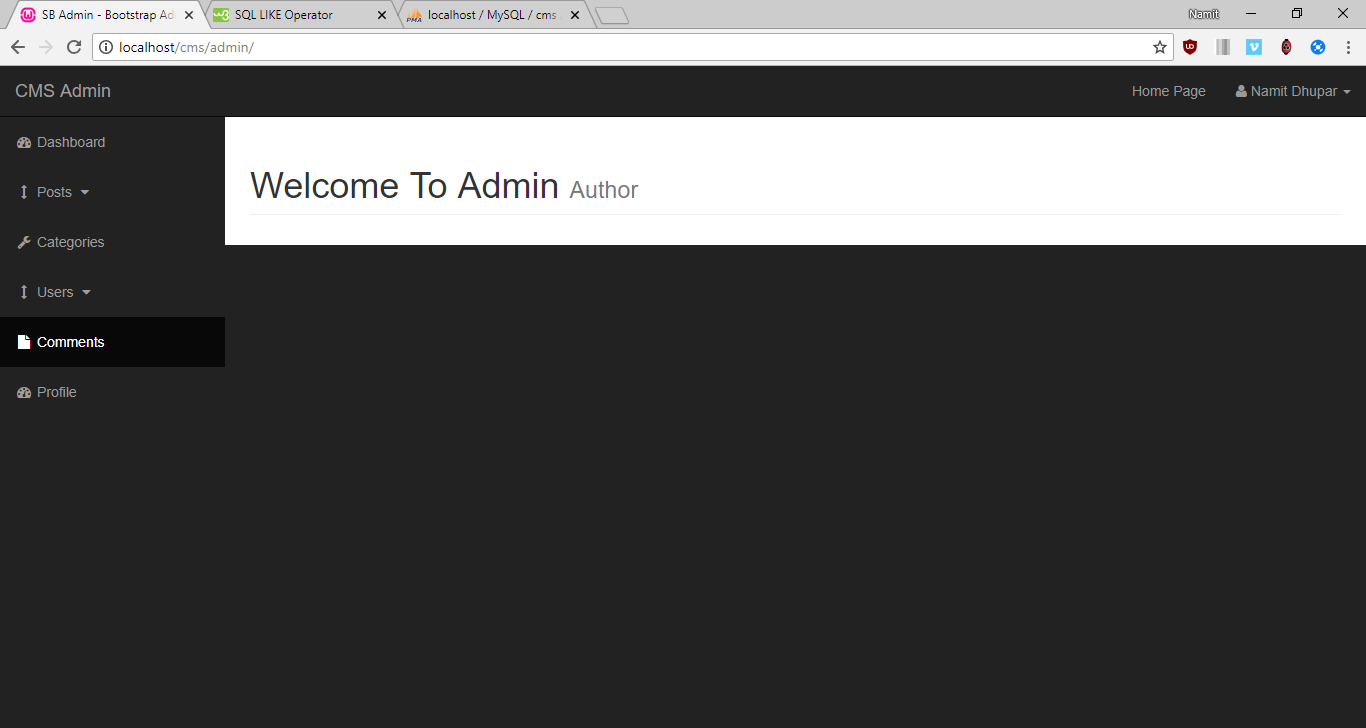
And in order to make this particular section relevant, we fetch the category title from the database and append them onto the Sidebar section, with the help of the following MySQL query= “SELECLT \* FROM categories” where categories is the table name, such that it looks like follows after a successful query execution:



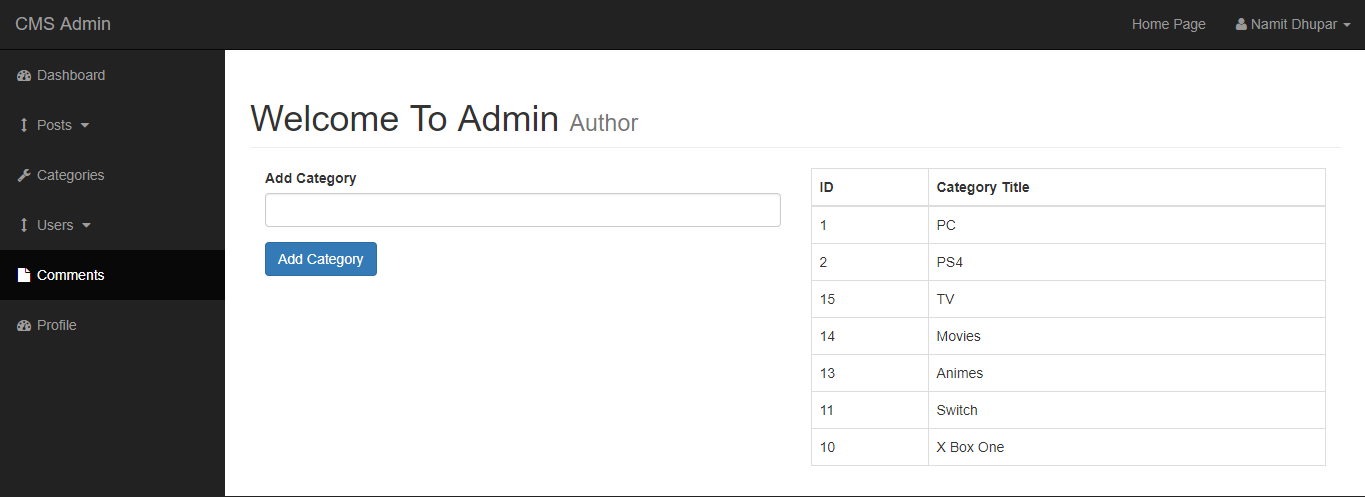
We can even reduce the number of category titles to be displayed in the sidebar with the help of LIMIT keyword within the mySQL query, so now the sidebar looks like:



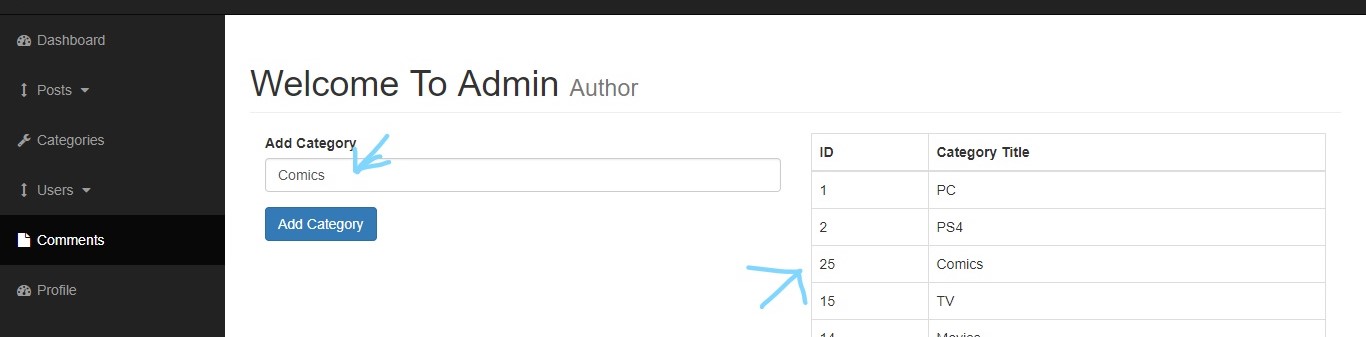
6.) **Admin Page:** Initially, our admin page looks as follows



**a.) Adding Category**: Within The admin page, we have an ADD Category, where we have a text box input in which we enter some characters and send them into the catagories table of the Database: The Page looks as Follows, where I have shown the contents of the database as Bootstrap table, and My aim is to append more data within the table just be entering it inside the text box.

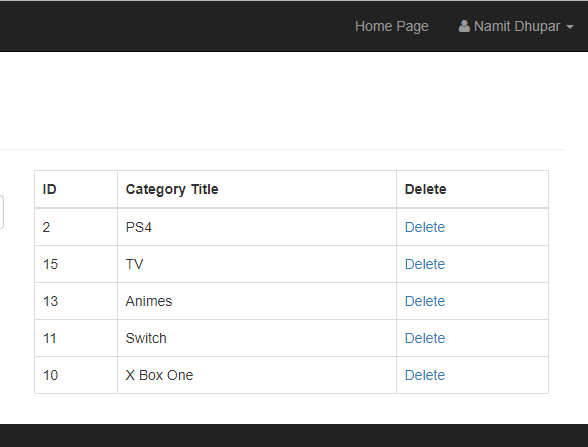


And this is achieved by the following **MySQL query : “INSERT INTO categories(cat\_title) VALUE (‘{$cat\_title}’)”;**



So now we add an additional column called ‘Comics’.

b.) **Deleting a Category**: We make another column in the following Table, called Delete;



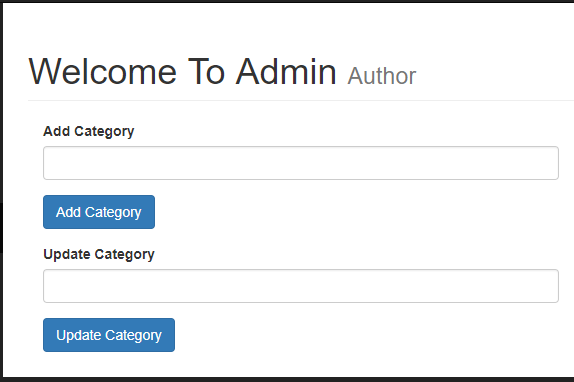
And in order for Delete to come into action, we use the **following MySQL Query “DELETE FROM categories WHERE cat\_id = {$cat\_id} “;**

And for the deletion to take effect in the php page, we redirect to that same page via a **PHP function of header(“Location: categories.php”).**

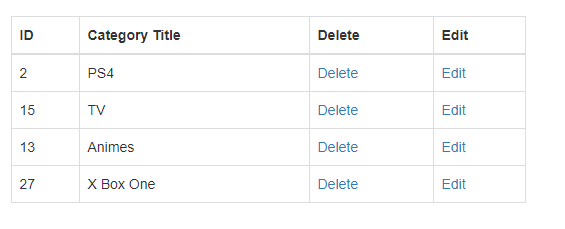
For Example, If we want to Delete Switch from the table, we simply click on the ‘delete’ link in same column, Hence we see that “Switch is now no longer in Database.

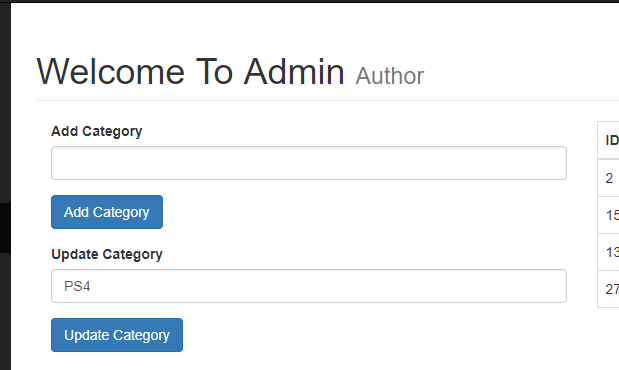


C**.) Updating a category:** For that we make a separate form which will update the category of the entered ID/Title



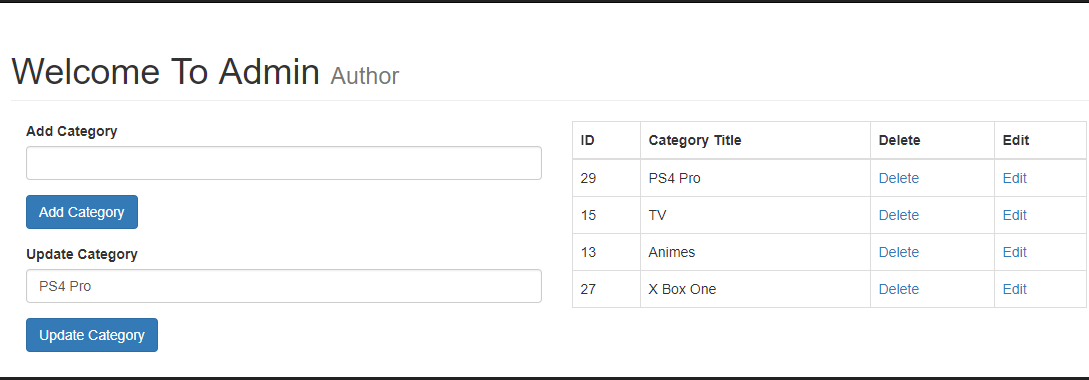
And Along with that we make another column in the table called ‘Edit’, and it’s purpose will be that when the admin user clicks on the ‘Edit’ link, the category title belonging to that category id will get added to the Input Text region and hence we will be able to edit to it further and make Update it’s value within the database.

  
The MySQL Query that enables us to do so is: “**SELECT \* FROM categories WHERE cat\_id = {$cat\_id}”;**For example, If we want to edit the PS4 title, then simply, click on edit and we see the following:

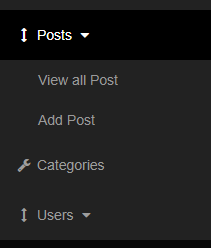


To Update the title(PS4 in this case), we use the following MySQL Query: **“UPDATE categories SET cat\_title = ‘{$updated\_title}’ WHERE cat\_id = $cat\_id “;**

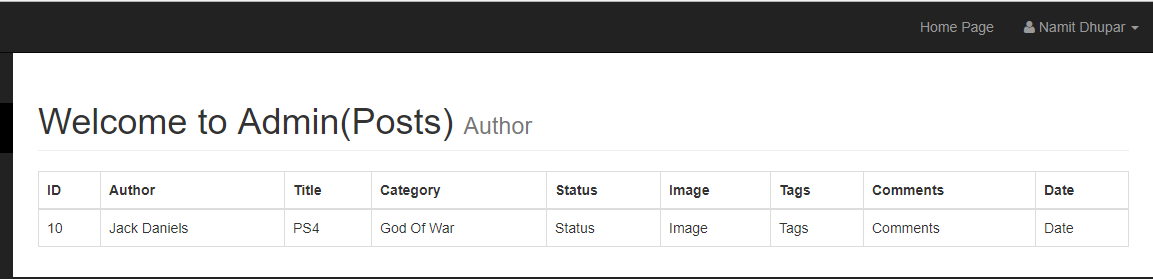
Hence, we obtain the updated result successfully as:



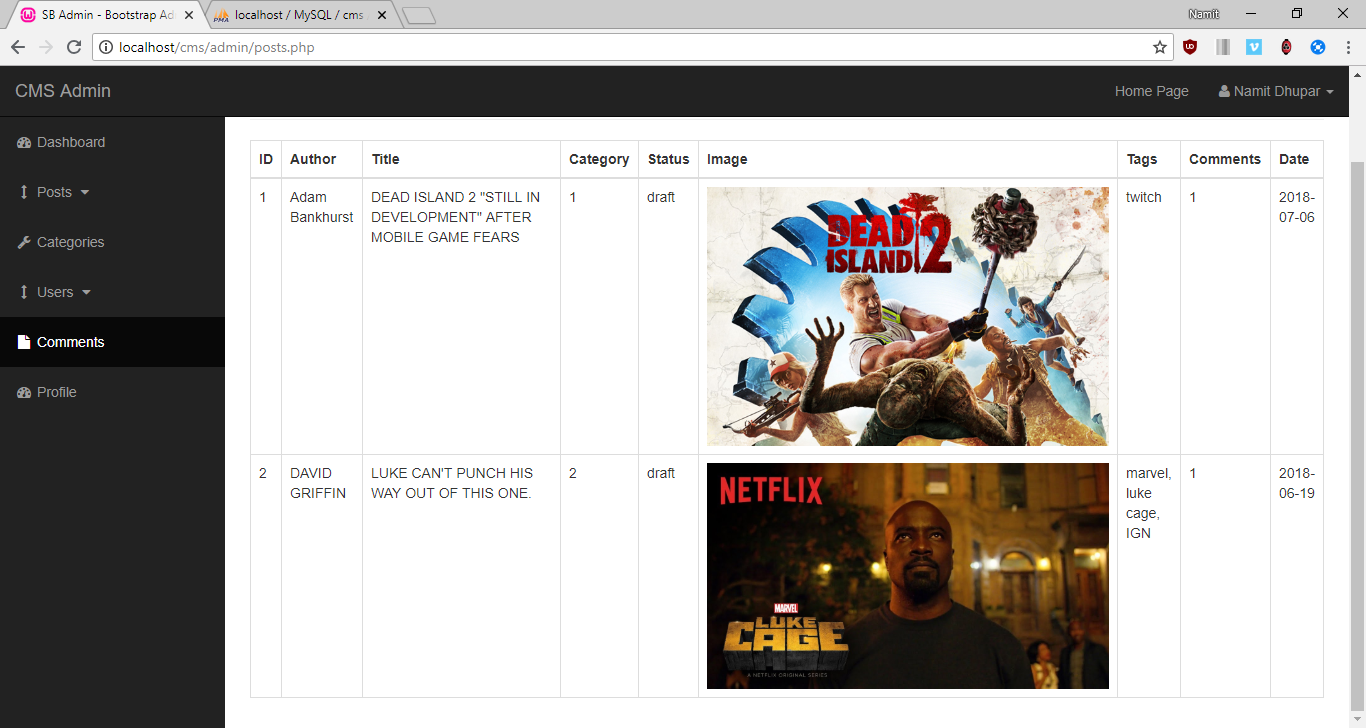
7.) **Adding A Post :** Inside the Admin, We have another Side bar entity called **Post,** which is a dropdown, that gives us an option for Either Adding or Viewing post.



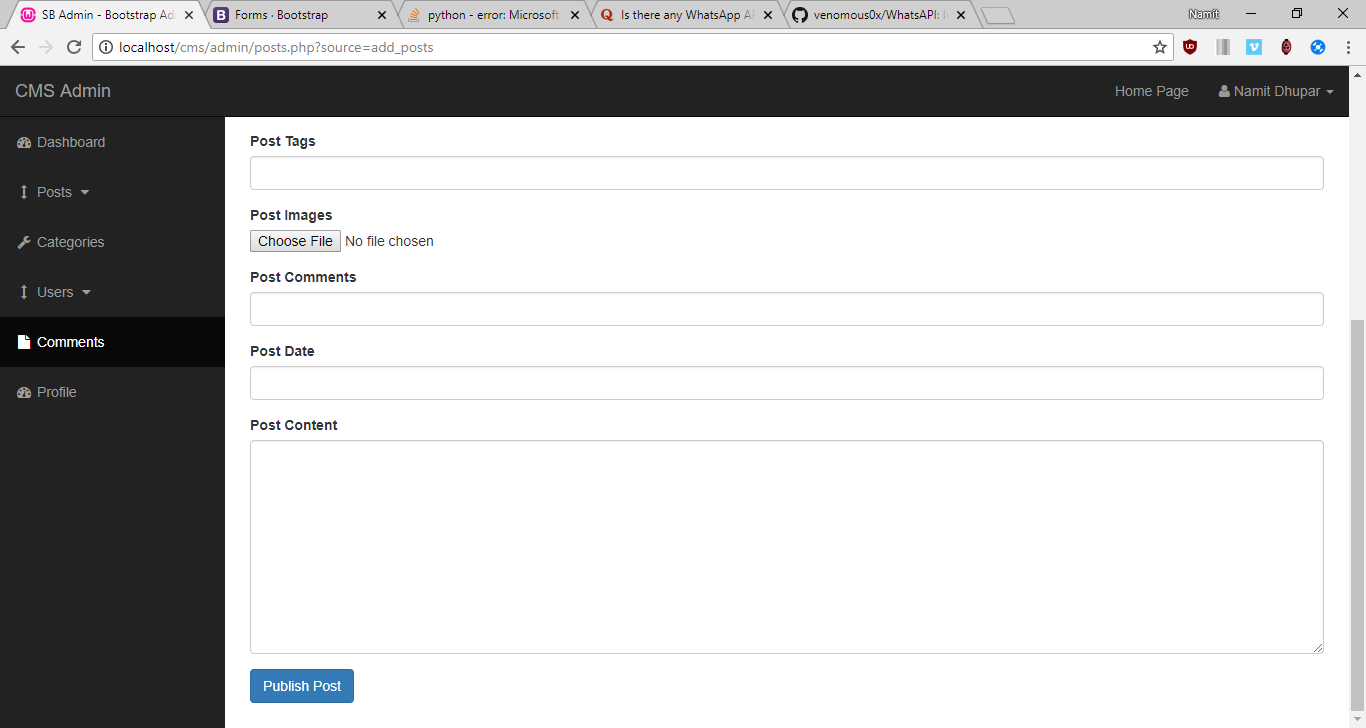
a**.) Add Post:** So, In order to demonstrate that, we simply create a table inside the main page for posts, which will further help in demonstrating it’s functionality(With the Table, meant just to be a placeholder.



Now, in order to add the posts’s database content inside the table, the MySQL query used for the same will be: **“SELECT \* FROM posts”** and hence, with the help of this, we find the following result:



So in order to add post, We create a simple Form related to the fields we created within the database, which looks as follows:



Such that an author can write his/her post and publish it within our webpage, thereby making there own entry within the database without directly interacting with it.

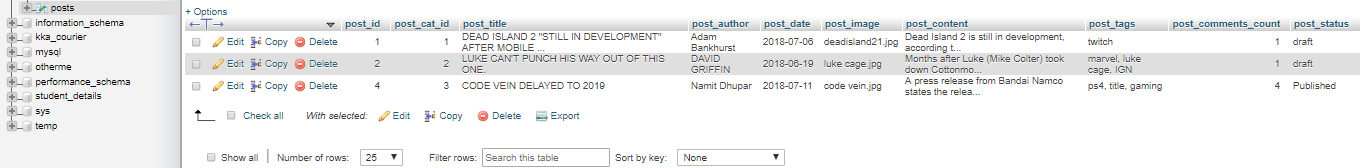
I would like to note another additional functionality for the choose image, where, whenever we chose an image, we must send that first to some temp location before sending that into the server, and to do that we use the super global function called, **$\_FILE**

And the PHP function responsible for this displacement of file is called: **move\_uploaded\_file()**

**a.) Post Data**: So, Now we input the information inside the following form, which we will see within the Database this time



So after Pressing the publish post button, the entire data will go into the post table of the database:



So Finally, we can see the output in the front page of our CMS Application:

