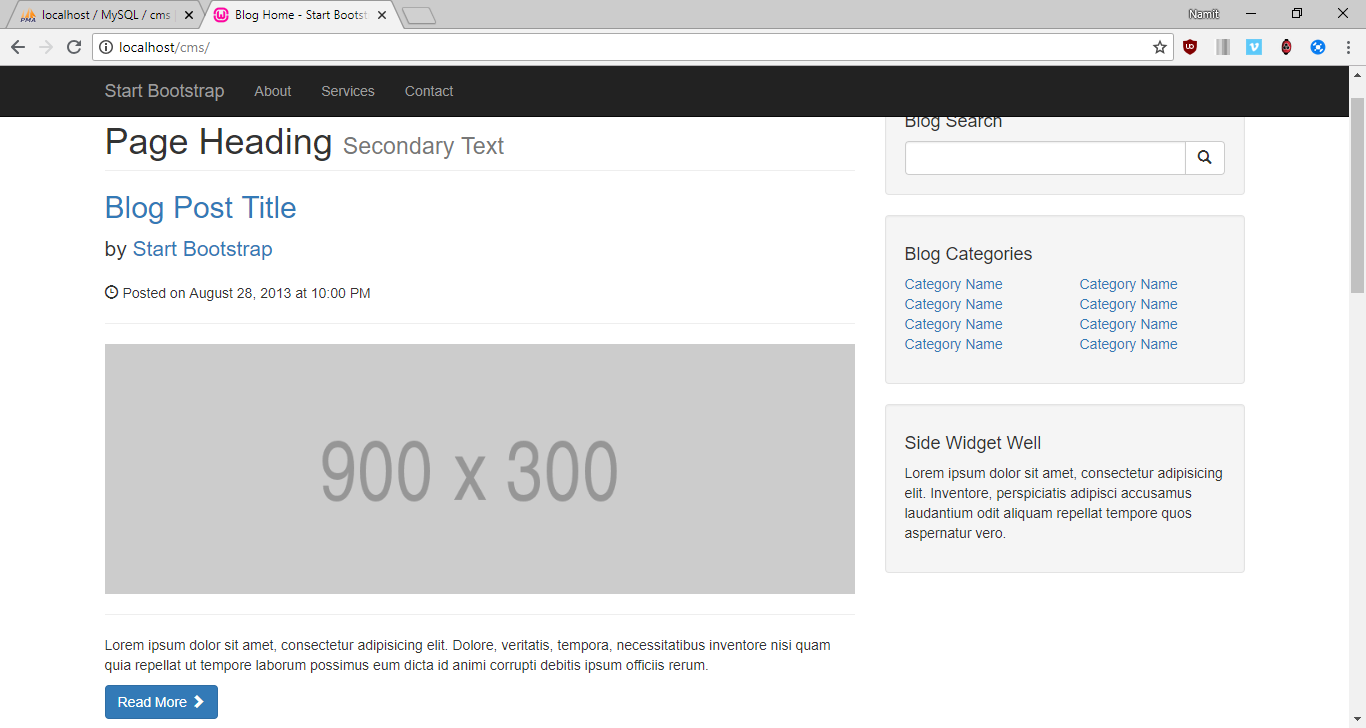
1.) in wamp, go to PHP tools=> phpInpfo-> Set the following:  
a.) display\_errors = NO  
b.) output\_buffer = 4096 Bytes

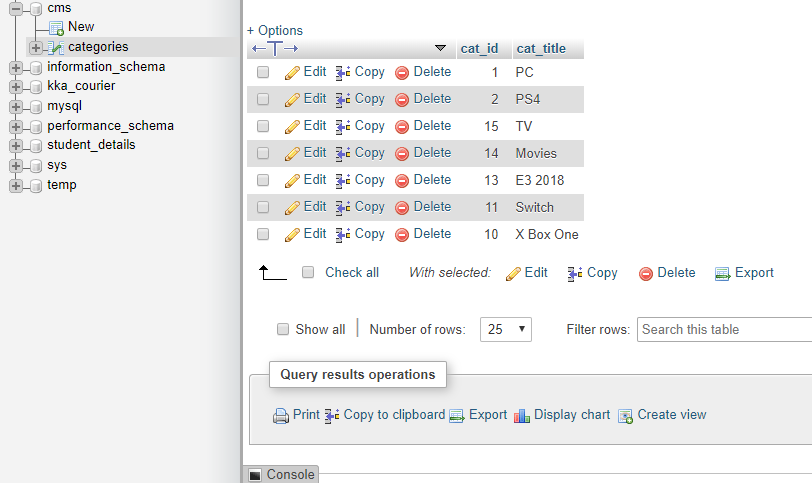
2.) Get a bootstrap template for

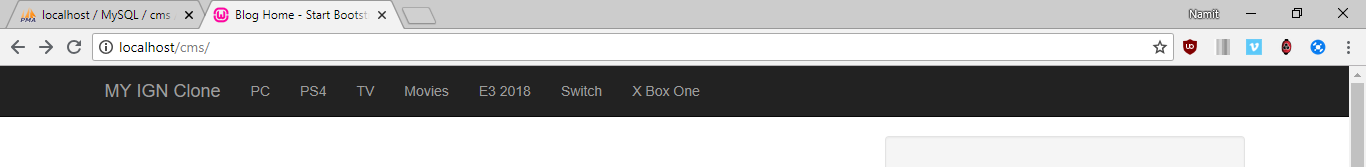
a.)cms:



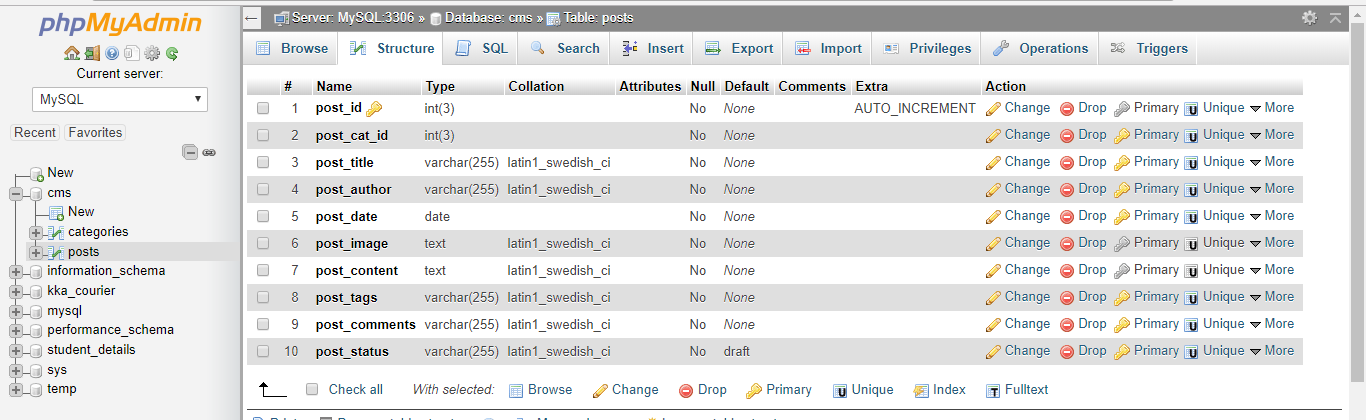
b.)As well as Admin:

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

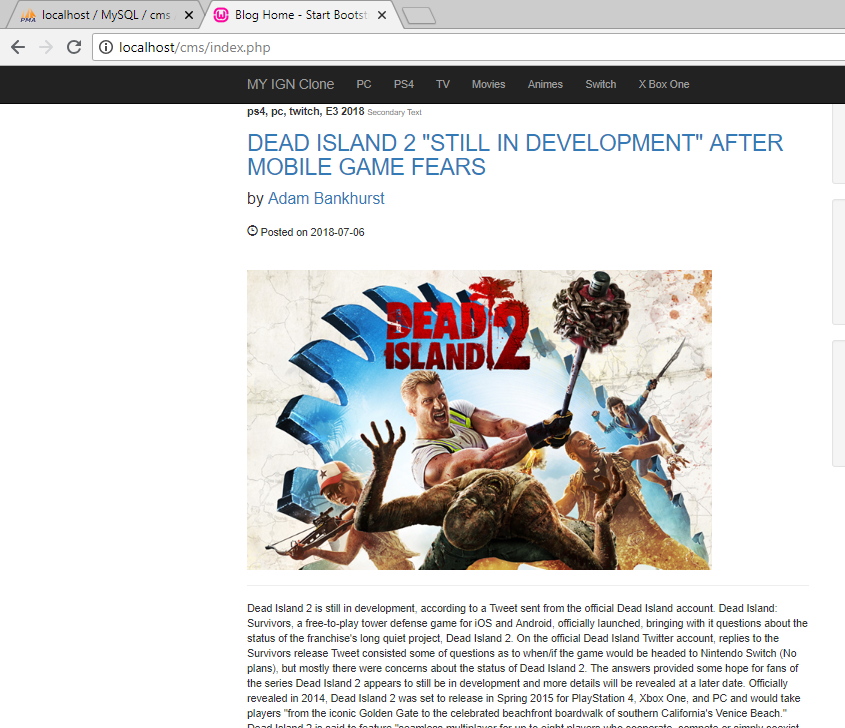
The cat\_title will have the following structure, such that we pull the information from our mySQL database and then we display it withing the TOP NAVBAR with PHP syntax  
  


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:**

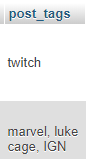
In order to construct a search engine, we need to create an input form with the “POST” type HTTP request method, along with that, we shall name the buttons, input spaces accordingly.

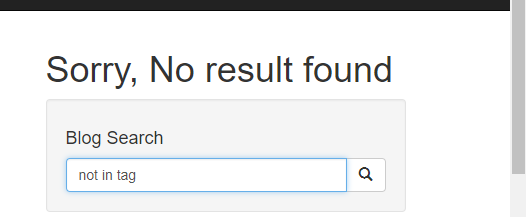
The Search form: 

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.

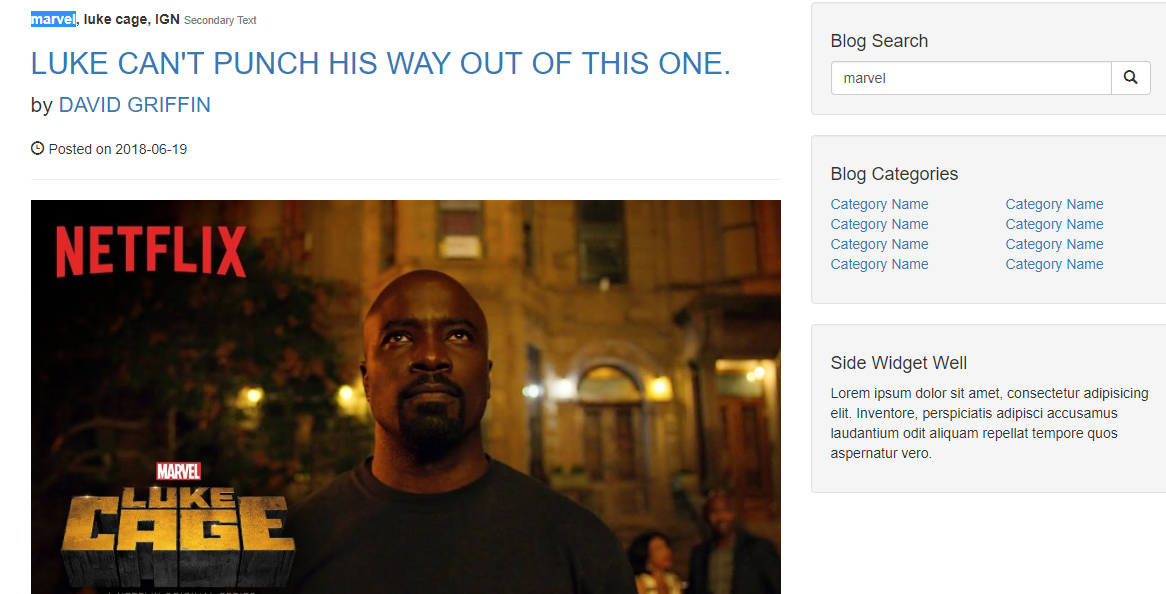
So for that, an SQL Query is used which will be: **“ SELECT \* FROM posts WHERE post\_tags LIKE ‘%$search%’ ”;**

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



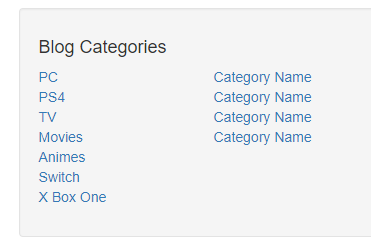
And if there are no such rows obtained, then with the help of an SQL function ie “ **mysqli\_num\_row()** “,   
we can throw a Result not found error message:

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.

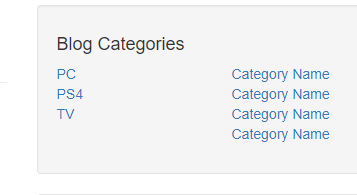


5.) **Sidebar UI:** The Sidebar UI is the component of the webpage, which displays the “BLOG CATEGORIES” section,

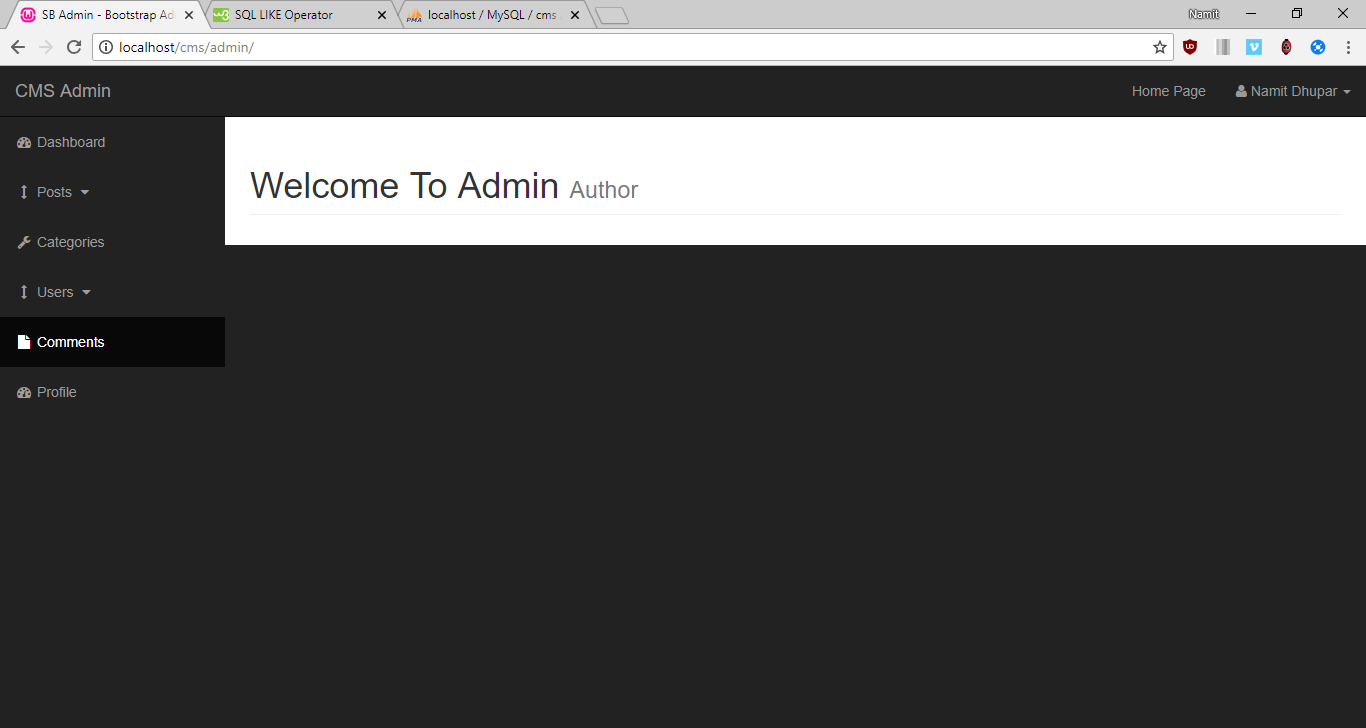
And in order to make this particular section relevant, we fetch the category title from the database and append them onto the Sidebar section, with the help of the following MySQL query= “SELECLT \* FROM categories” where categories is the table name, such that it looks like follows after a successful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



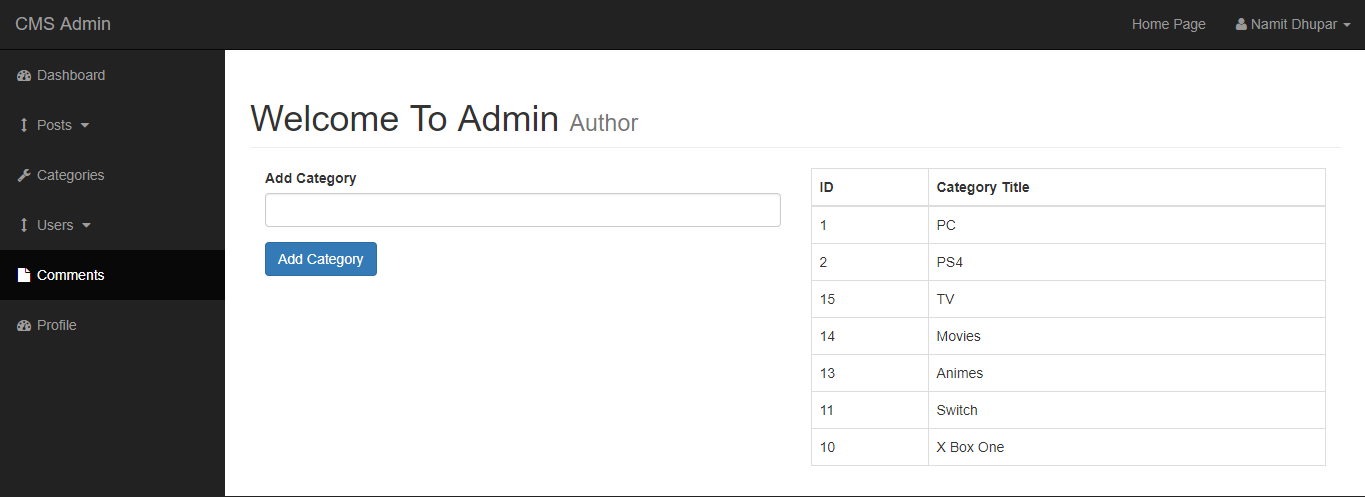
We can even reduce the number of category titles to be displayed in the sidebar with the help of LIMIT keyword within the mySQL query, so now the sidebar looks like:



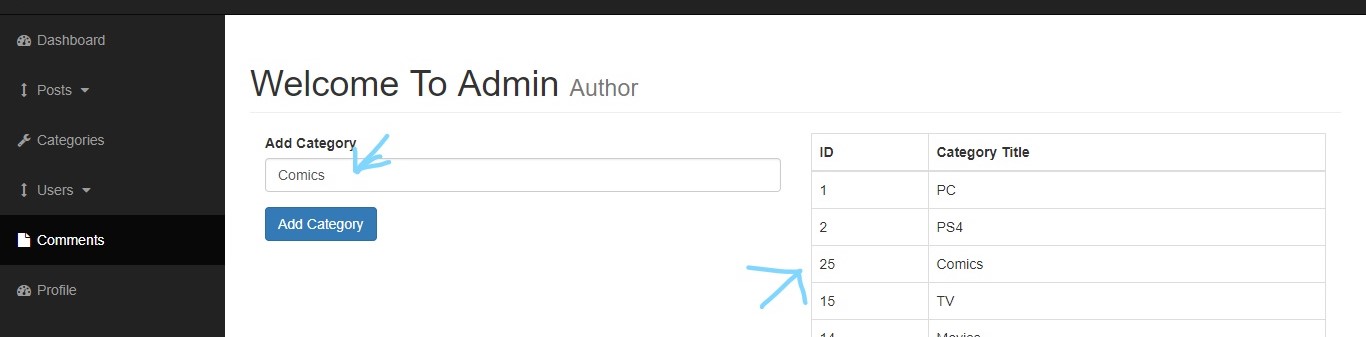
6.) **Admin Page:** Initially, our admin page looks as follows



**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;



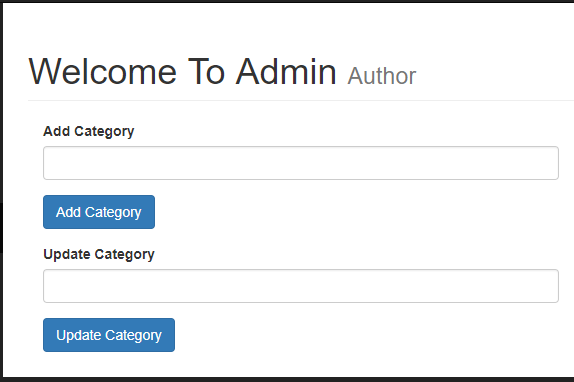
And in order for Delete to come into action, we use the **following MySQL Query “DELETE FROM categories WHERE cat\_id = {$cat\_id} “;**

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”).**

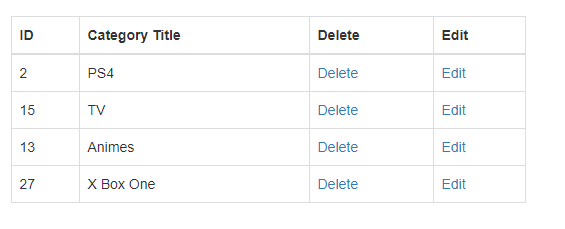
For Example, If we want to Delete Switch from the table, we simply click on the ‘delete’ link in same column, Hence we see that “Switch is now no longer in Database.

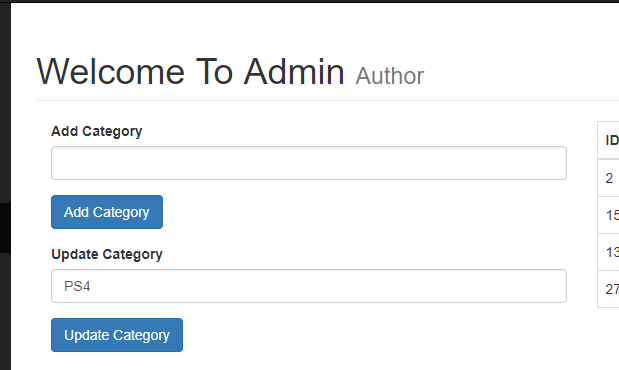


C**.) Updating a category:** For that we make a separate form which will update the category of the entered ID/Title



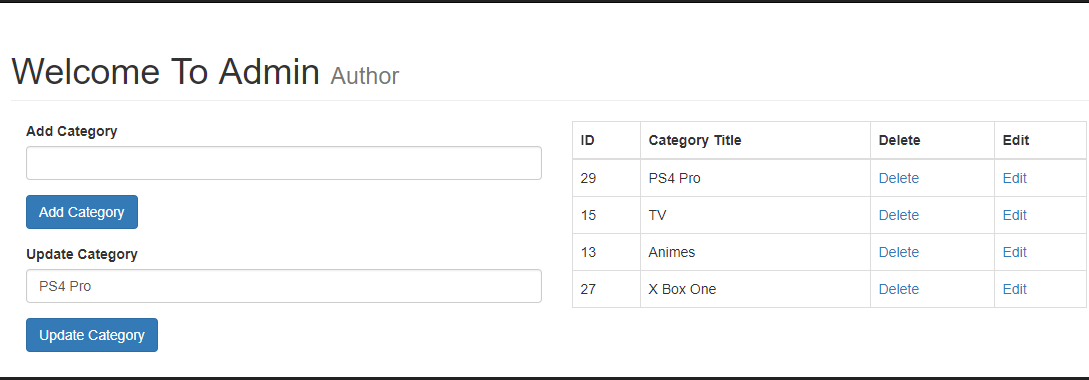
And Along with that we make another column in the table called ‘Edit’, and it’s purpose will be that when the admin user clicks on the ‘Edit’ link, the category title belonging to that category id will get added to the Input Text region and hence we will be able to edit to it further and make Update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

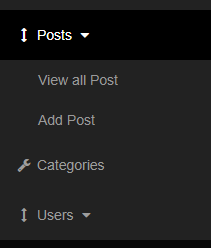


To Update the title(PS4 in this case), we use the following MySQL Query: **“UPDATE categories SET cat\_title = ‘{$updated\_title}’ WHERE cat\_id = $cat\_id “;**

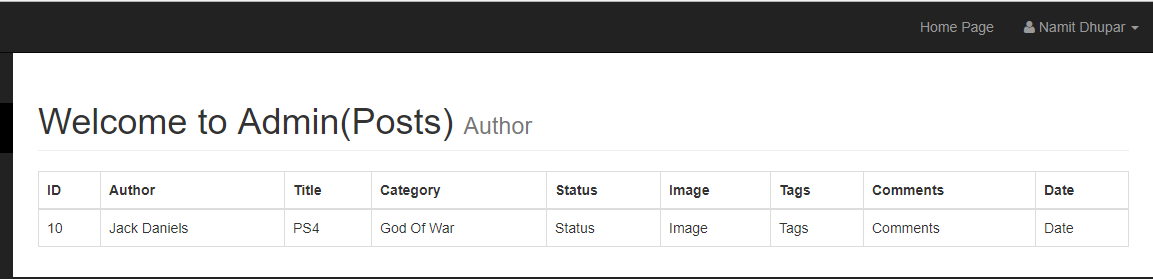
Hence, we obtain the updated result successfully as:



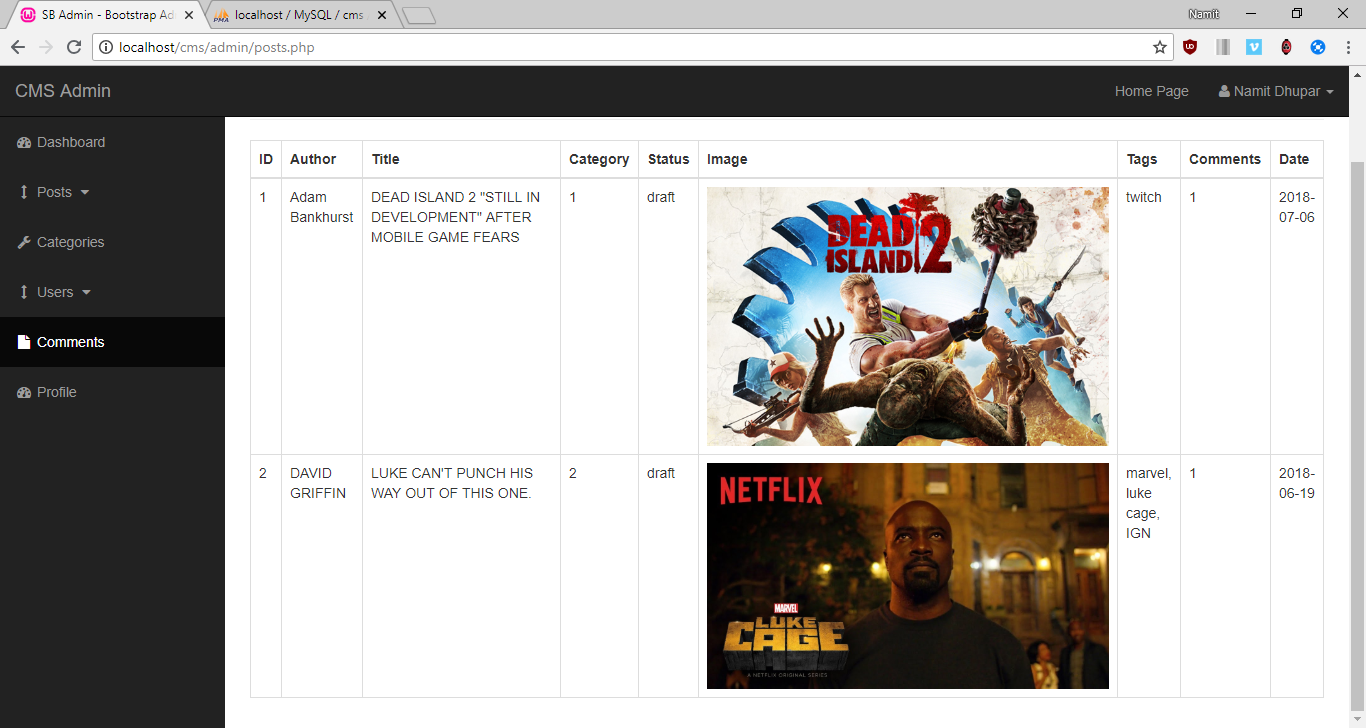
7.) **Adding A Post :** Inside the Admin, We have another Side bar entity called **Post,** which is a dropdown, that gives us an option for Either Adding or Viewing post.



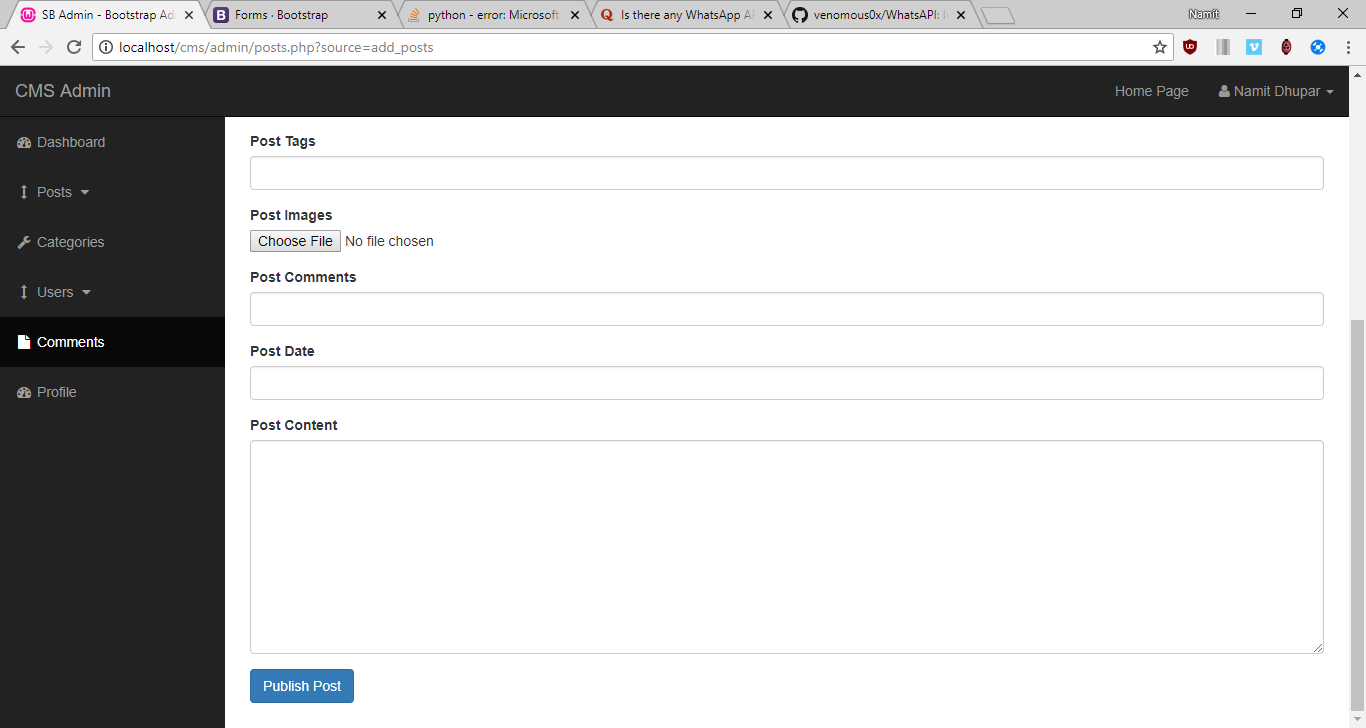
a**.) Add Post:** So, In order to demonstrate that, we simply create a table inside the main page for posts, which will further help in demonstrating it’s functionality(With the Table, meant just to be a placeholder.



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**

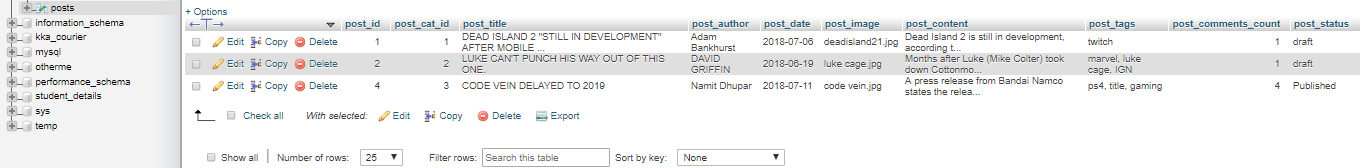
**Ie** from the downloads, we are able to move this image file into the “Images” folder of our CMS project

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

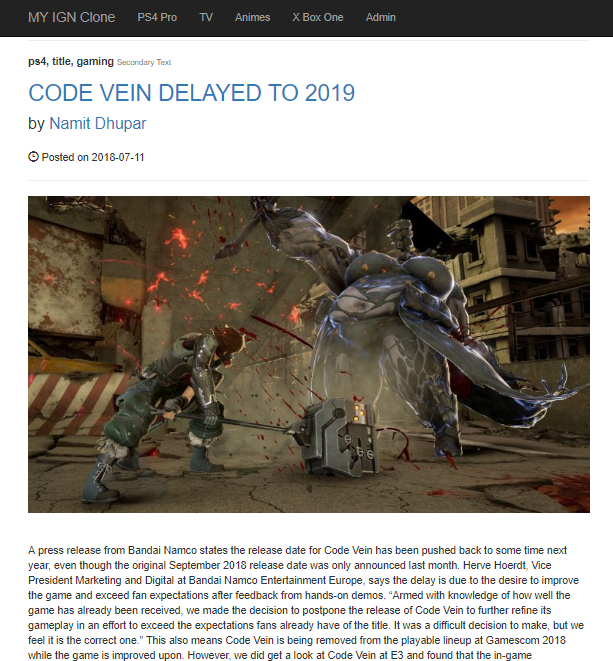
**a.) Post Data**: So, Now we input the information inside the following form, which we will see within the Database this time



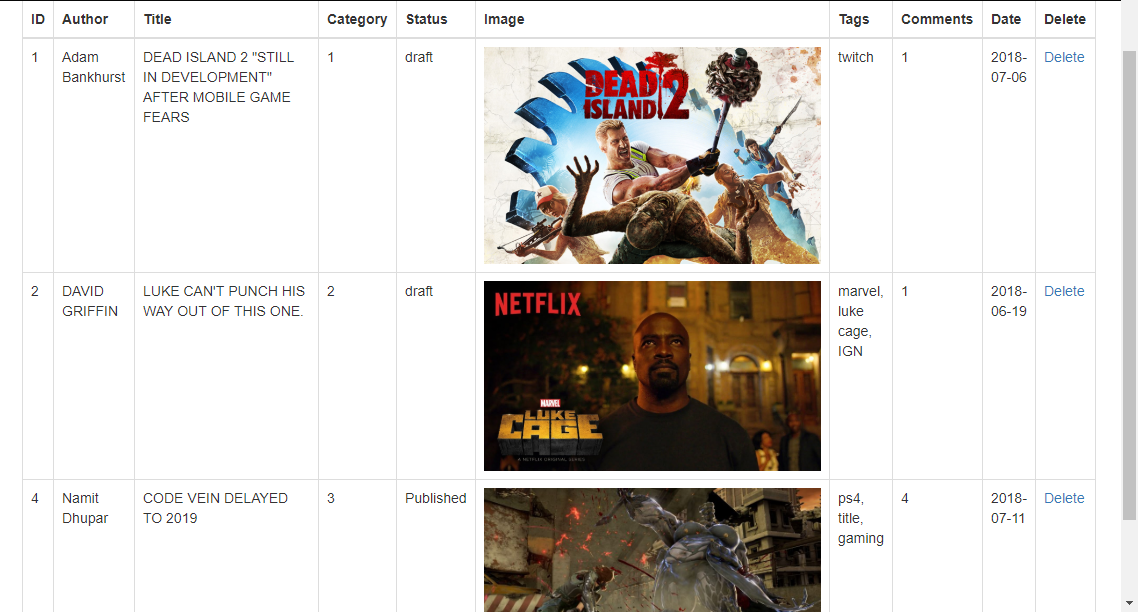
So after Pressing the publish post button, the entire data will go into the post table of the database:



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

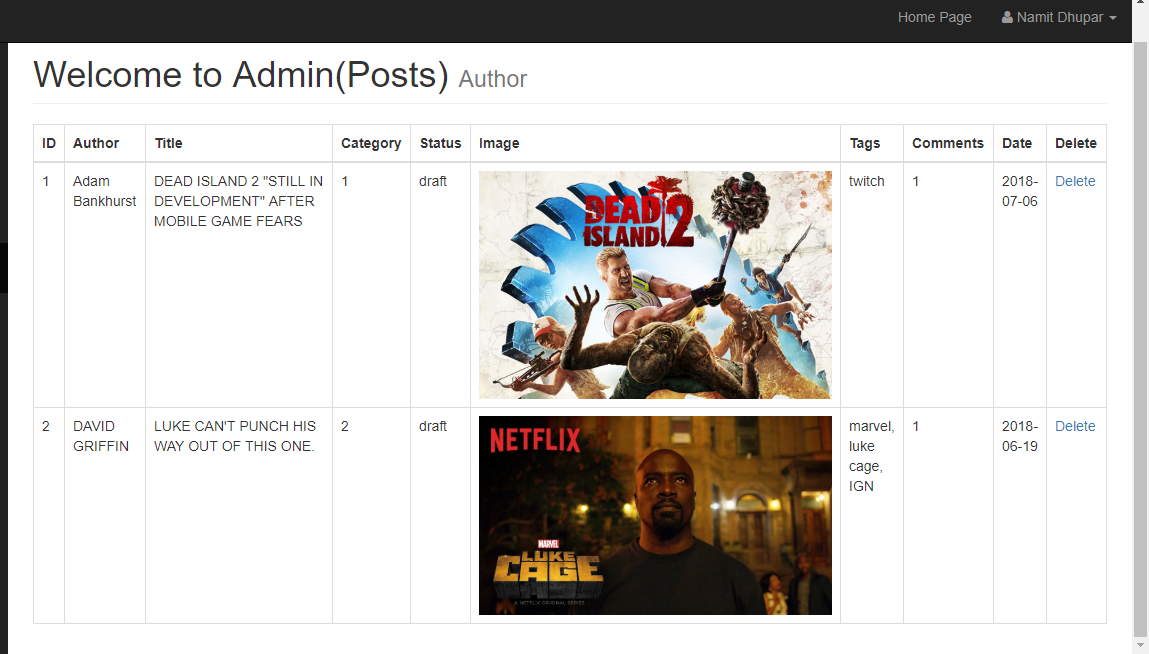


So here the MySQL query used will be: **"INSERT INTO posts (post\_cat\_id, post\_title,post\_author,post\_date,post\_image,post\_content,post\_tags,post\_comments\_count,post\_status) VALUES('{$post\_cat\_id}','{$post\_title}','{$post\_author}',now(),'{$post\_image}','{$post\_content}','{$post\_tags}','{$post\_comments\_count}','{$post\_status}')";** where **now()** displays current date.

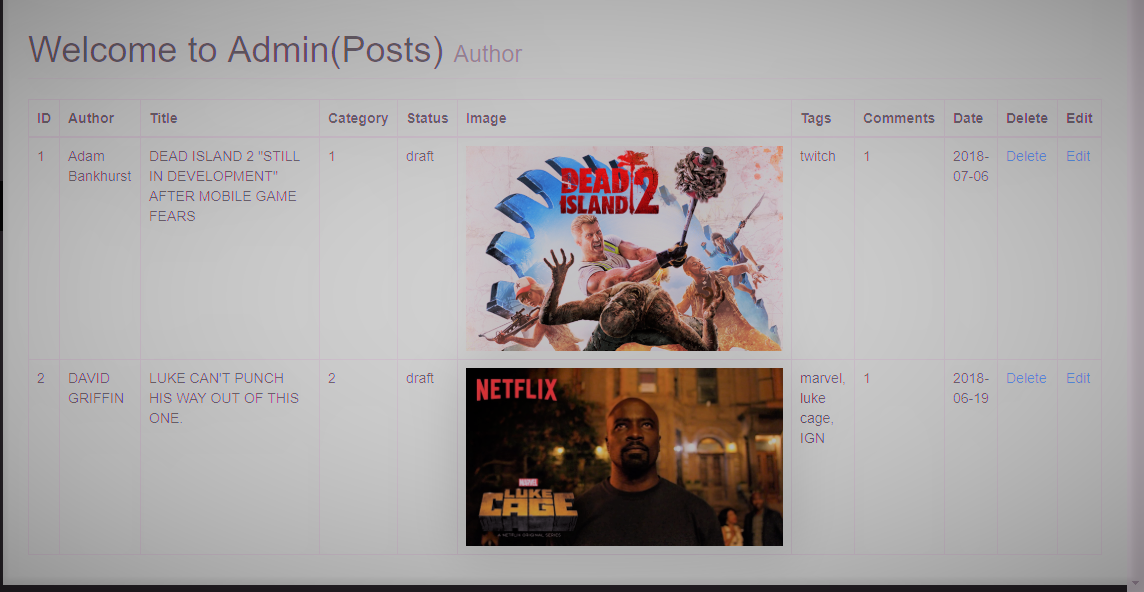
b.) **Delete Post ID:** We make a separate Delete link in the post table, 

So, now if we want to delete the post with ID=4, we use the following MySQL Query: **“DELETE FROM posts WHERE post\_id = $post\_id”**

Hence, we obtain the following result after pressing the Delete Link, our post table will look like this:

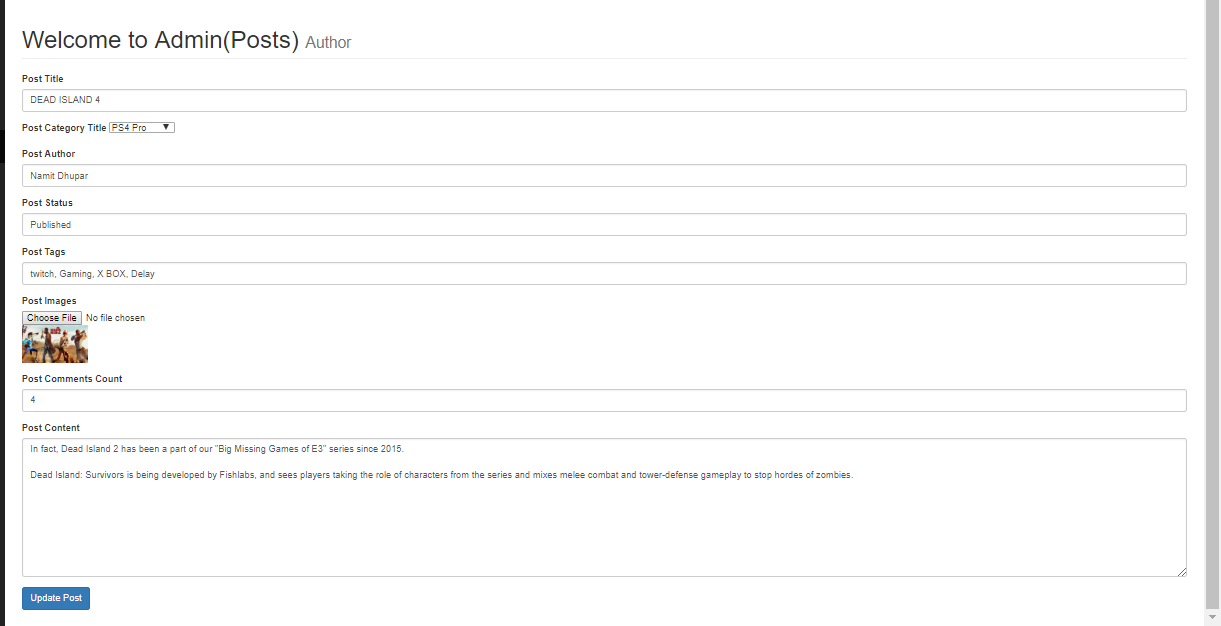


c.) **Edit a post**: Similar to Delete, we create an Edit click link inside the above table, as above:



Now, We can edit the particular post, by just clicking on the edit option, and then we get redirected to a page where the particular post’s data be presented onto the Input area of the form:

Ex: on editing the First post, We see the following; ***(See the changes made in the URL)***



This was possible due to the following MySQL query: **“SELECT \* FROM posts”** just as seen in **update\_cat.**

Hence, We can finally make the changes within our same edit page, with the help of the following SQL query:

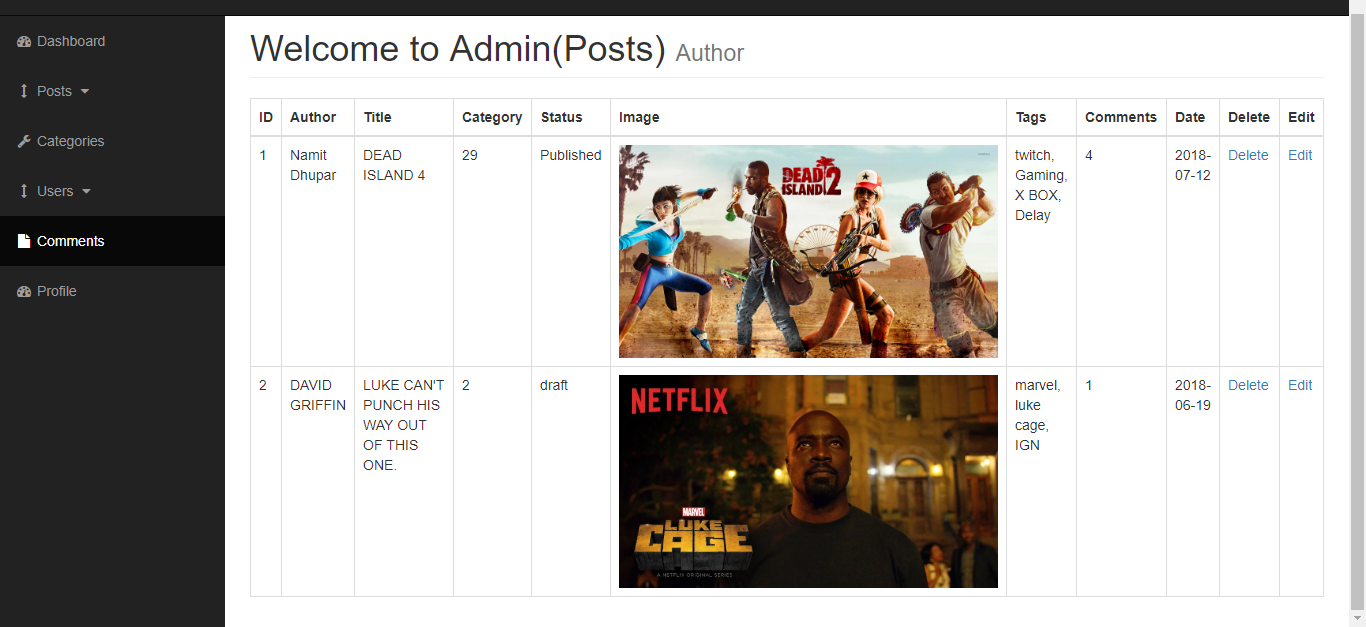
**"UPDATE posts SET post\_title = '{$post\_title}', post\_author = '{$post\_author}', post\_cat\_id = '{$post\_cat\_id}', post\_status = '{$post\_status}', post\_image = '{$post\_image}', post\_tags = '{$post\_tags}', post\_content = '{$post\_content}', post\_date = now(), post\_comments\_count = '{$post\_comments\_count}' WHERE post\_id = $update\_post\_id";**

Hence, on clicking the button called, Update the Post, we finally see changes within our Database, but when it comes to updating images, we need to find a bit different approach, which mean that we have to first check if the Image Choose file input is empty, which can be know by a PHP function called,  
**if (empty($post\_image)),** and later an SQL statement of**: “SELECT \* FROM posts WHERE post\_id = $post\_id”.**

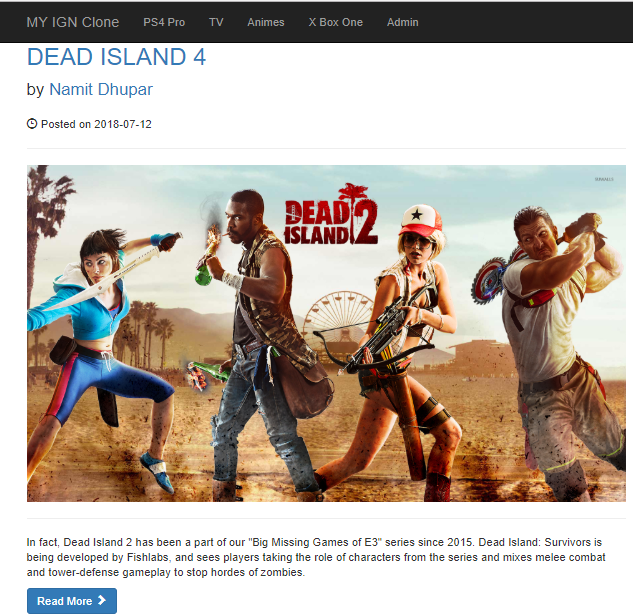
So after all this, we are finally able to see the changes inside our Database after Pressing the **Update Post** button as follows:



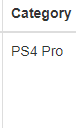
As well as in our **View post table**:



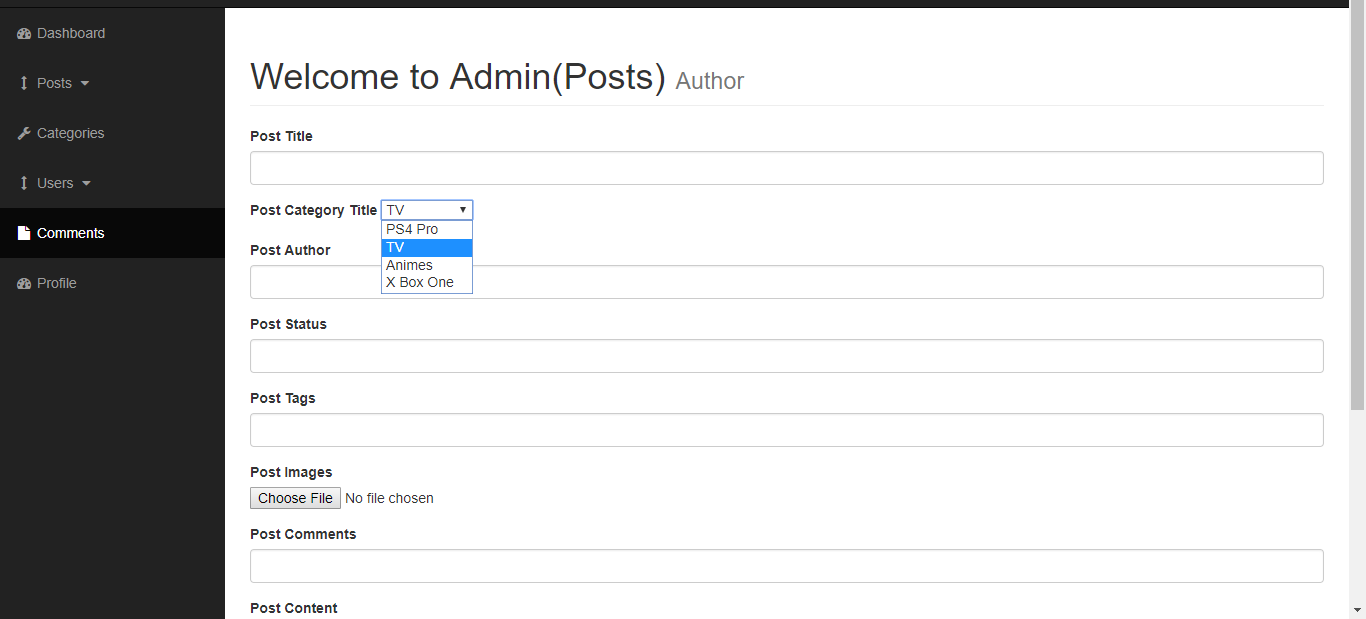
And finally, within our **Home Page also**:



d.) **Displaying Category Title in the posts table:** If We want to display the category title instead of the id, then we make another MySQL query: “**SELECT \* FROM category WHERE cat\_id = ${post\_cat\_id}**”. Which will present us with the following output:



e.) **Displaying Category Title in the dropdown menu of the Add post page:** In order to do this, we use the same query that is: “**SELECT \* FROM categories**”, and hence we see the following result while adding a new post:



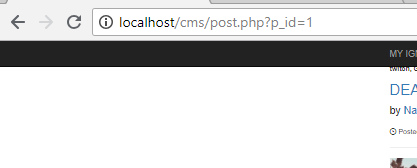
So, when we enter a new entry inside the database’s post table, we find that the finally created post displays the category title as mentioned by the Category title dropdown.



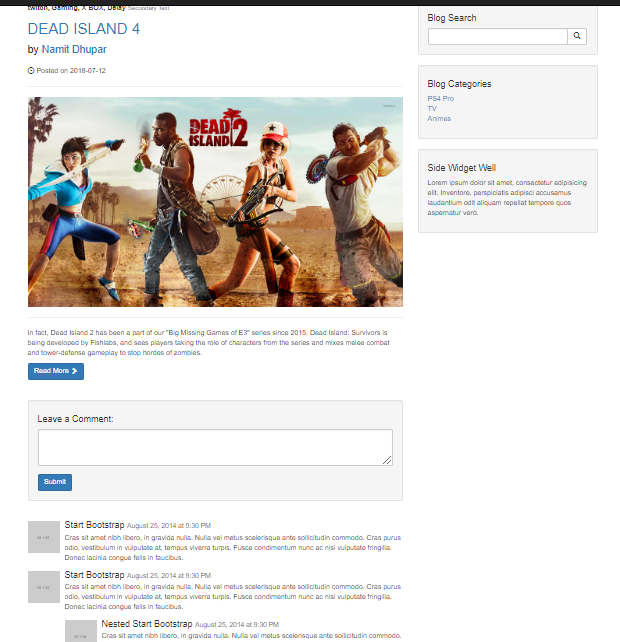
f**.) Displaying Indivisual posts by clicking on the title:** According to this logic, we need to click on a title and this takes us to a page with the contents associated with that particular id only,

Such that, on the index.php file, we give the following link to title such that: **post.php?p\_id=<?php echo $post\_id; ?>** as the link to that page, and the $post\_id is used to get the post of that particular ID where we clicked, by the help of the following MySQL function: “**SELECT \* FROM posts WHERE post\_id = $post\_id”.**

Observing the URL, we pretty much get a clear Idea that how the particular ID is displayed inside the address bar:

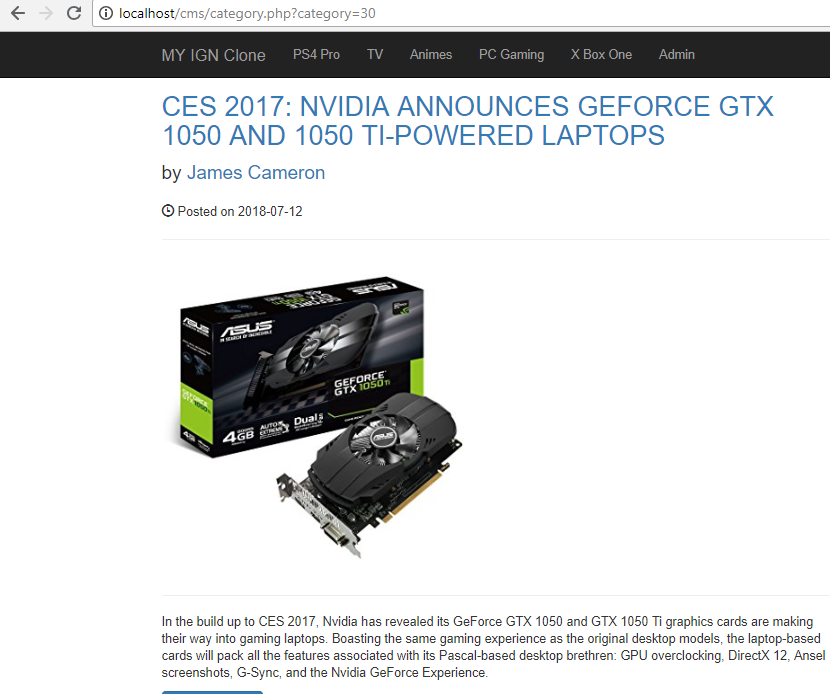


And hence, the entire page with p\_id=1 will look something like this:



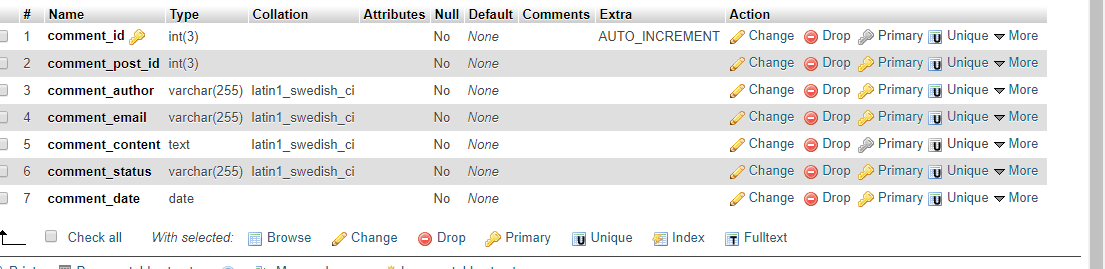
g.) **Adding Category to the Sidebar**: The Sidebar of the homepage consists of the categories of all my posts present within my CMS application.Such that, the webpage has the following posts belonging to it’s respective category, 

Now On clicking on the Sidebar of the home page, we want to display the post, belonging to that very category only, so this was done by passing a URL in the sidebar’s file ie “ **category.php?category=$cat\_id”** and this was made possible by the MySQL function: “**SELECT \* FROM posts WHERE post\_cat\_id = $post\_cat\_id”.** Like if we click on the PC gaming category on the above sidebar link, we are sent to the above URL and Page:

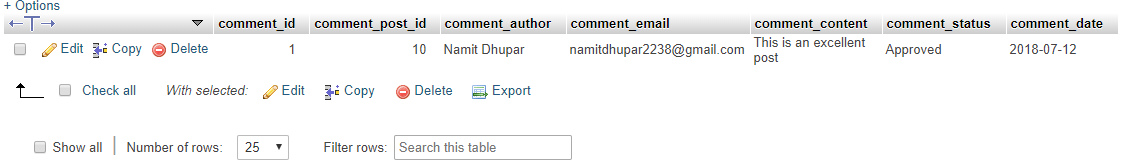


8.) **Adding Comments:**

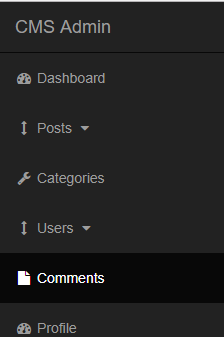
a**.) Create Comment table:** inside the CMS database, we create another table, called the comment table, and this table’s structure is as follows:



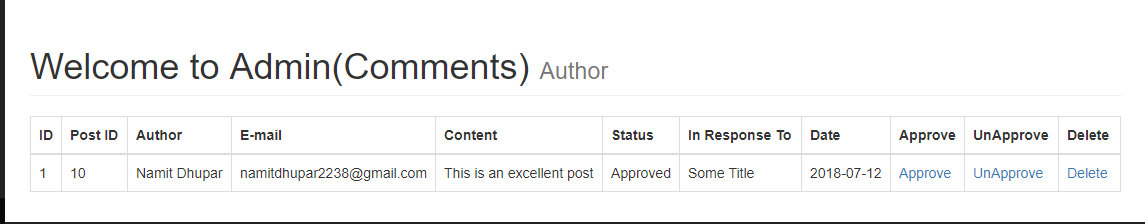
And after inserting some data, our table looks something like this:



b**) Displaying the comments in the Webpage:** We have the following data, which we want to present within our webpage, for this we got to our Admin Page and on it’s side Navbar we see an option called, Comment:



c.)**Displaying Comments in the Table:** For this, we simply make a form related to the database’s comments table column and with the help of the MySQL function**: “SELECT \* FROM comments”** we fetch the data from the column of the database and add it to our table:

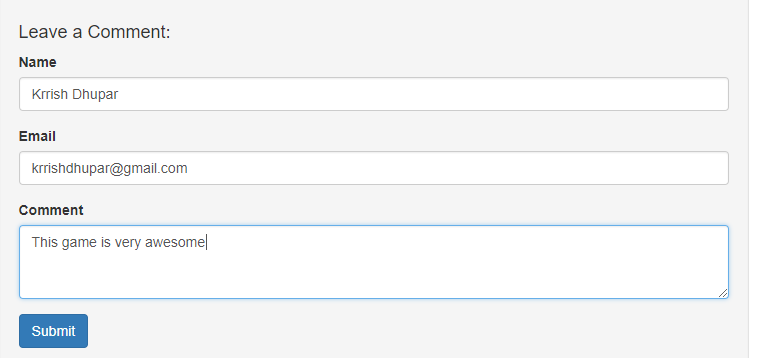


d.) **Storing comments from HomePage to the Database:** Construct a Form for the user to enter their comments inside the website:

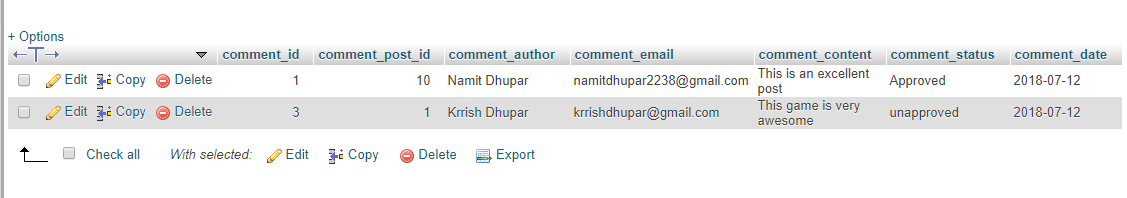


Now, we use the superglobal method **for $\_GET[‘p\_id’];** as well as the MySQL Query**:” INSERT INTO comments (comment\_post\_id, comment\_author ,comment\_email ,comment\_content, comment\_status, comment\_date) VALUES($the\_post\_id, '{$comment\_author}', '{$comment\_email}', '{$comment\_content}', 'unapproved', now())";**

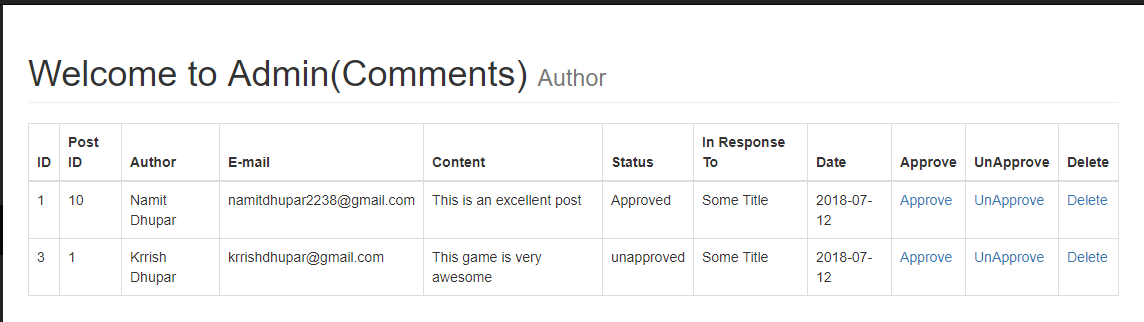
Which will insert all the data from the Form into our database, We fill the details and enter the submit Button.



After Entering the submit button, We can see the table is created inside our database:



And the Comment Table in the Admin page as well:



e.) **Dynamic Responder:** Even though our table has all the dynamic values, but there still is a column called “In Response To” which carries on with a static value, even though we need to display the information that comment has been made about which title of the post in our website.

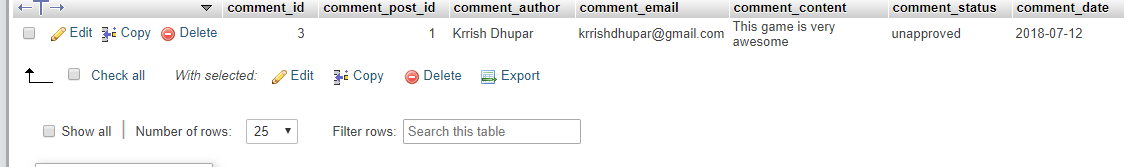
So with the help of the MySQL Query: **“SELECT \* FROM posts WHERE post\_id = $comment\_post\_id”,** we need to interact with the posts table in the database and by passing the URL referencing the post table, ie: **posts.php?p\_id=$post\_id**, We can determine the Title related to the Comment:



e.) **Deleting the comment:** We need to delete undesirable comments, So in order to do that, we use the following MySQL query: **“ DELETE FROM comments WHERE comment\_id = $del\_comment\_id**”.



So, for example, If we want to remove the comment of the Bad Guy, then we simple click on the delete button and this would simply remove the person’s from the database.

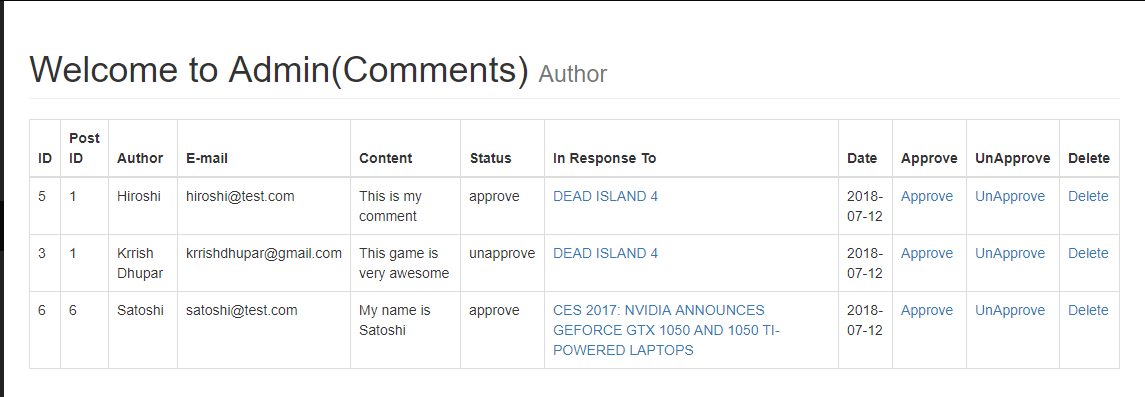


Hence we see, that the comment is no longer available inside the database.

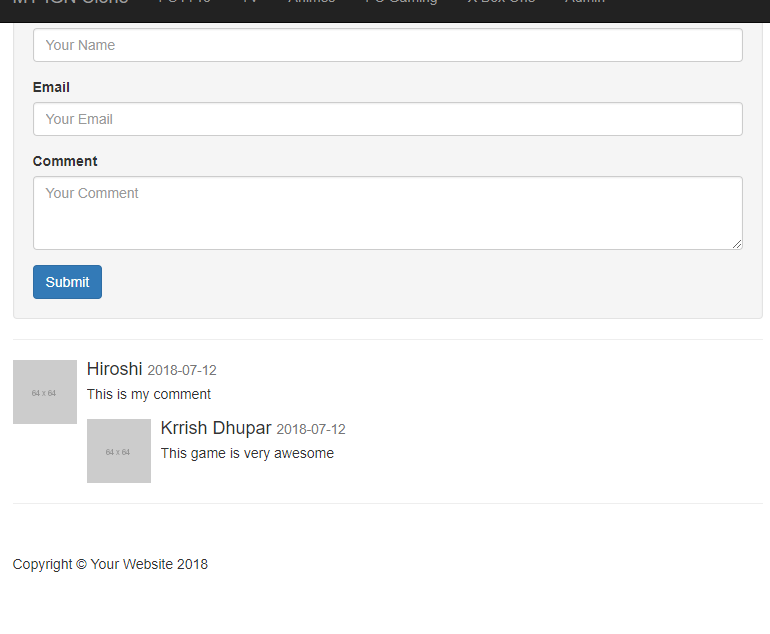
f**.) Approving and Unapproving comments:** This one is very much similar to Deleting a comment from the database table, MySQL function:

**i.) *Unapprove* = “UPDATE comments SET comment\_status = ‘unapprove’ WHERE $comment\_id = $unapprove\_comment\_id”.**

**ii.) *Approve* = “UPDATE comments SET comment\_status = ‘approve’ WHERE $comment\_id = $approve\_comment\_id”.**

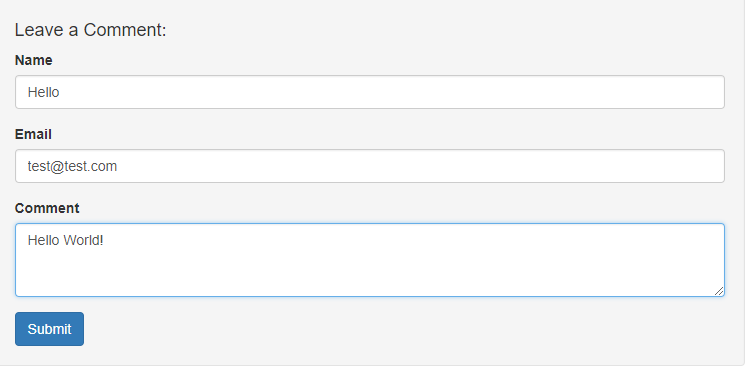
****

**g.) Displaying the approved comments only in the front page:** For this we use a MySQL function called **"SELECT \* FROM comments WHERE comment\_post\_id = $my\_post\_id AND comment\_status = 'approved' ORDER BY comment\_id DESC";**   
Here the last four query words means that here we show by the Newest comment first. Hence we see this following output:



h.) **Making the Comment Count in the posts.php Dynamic:** The MySQL Query used here is **“UPDATE posts SET post\_comments\_count = post\_comments\_count + 1 WHERE post\_id = $the\_post\_id"**

Such that the when we make a comment inside any given post, the post\_comments\_count column within the posts table will increment by 1 each time a comment gets submitted.



So after clicking on the submit button, we have to note that post\_comments\_count was earlier static and was set to 4 by default, hence by now making it dynamic and incrementing this by 1, we must see that the post\_comments\_count should be 5 now, which is true.

