

## a. Public Users

### Home page

Basically the page incorporates two functions of interest such as the “login page” area, and the “Webdesa Suggestions” area.

**The login area retains the functionality that allows registered users to acquire personalized access to the portal and enter the site.**

The login section is regarded a function that is generate for the customers group and will be explained in detail in the corresponding section.

**The area retrieves randomly a product from database and displays on the home page for each entry of a user. The name, price and image of the product are displayed on screen.**

A connection is made to the database in request of all the existing products.

Products ids are stored in an array on an incremental order

A random number is generated between zero (0) and the size of the array.

The number is used as an index position for a value in the array.

A new statement retrieves the corresponding data of a product based on it's ID.

The product is echoed on the screen within the use of HTML tags.



### Product page

**The page is the area that the products are demonstrated to the users.**

**The left listing bar provides categorization of the products and a users click will display the products respectively.**

The page provides simple HTML links for the listing of products categories and a centered “div” even though there is nothing within to display.

Nevertheless all categories links will generate a



transition to the same page but will additionally deliver a message next to the URL path. The message contains the category name of the product.

**In this image the user has clicked for one of the categories and the products are displayed.**

What is running in the background is that the empty 'div' is triggered with a 'php' script that search for the message on the URL.

In this case the message has been identified by the script, evaluated and incorporated on an mysql statement that was appended to the database.

Of course a connection was made first but for now own we will take this activity for granted.

The HTML "div" now demonstrates the content results of the query as it is primarily manipulated effectively by the PHP scripting.

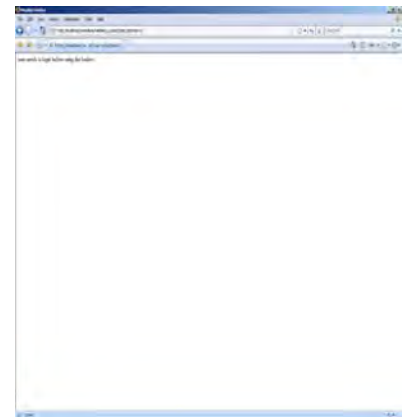
**The user displays a larger image of the product and a full description after clicking on one of the images of the products.**

In fact in the previous example the thumb images where coded to be concluded in HTML links with a reference to a new page passing two arguments to the URL path.

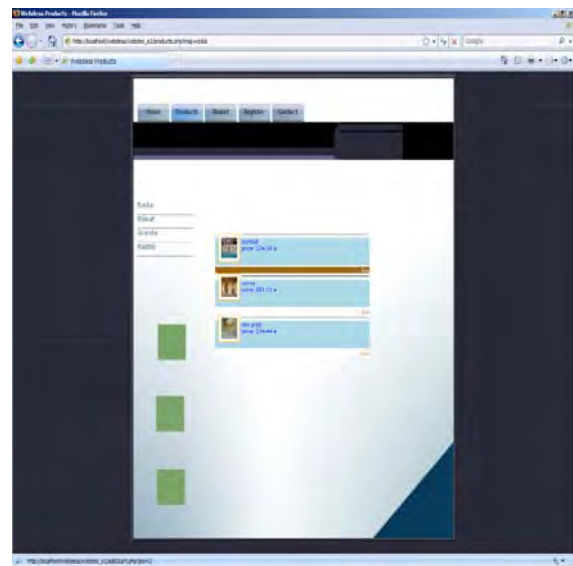
Messages are the ID and th Category of the product. The new page establishes a connection to the database and retrives the essential data to provide a full description of the product.

Additionally a new link is created that prompts the user to buy the product by adding iu to the basket.

**Nevertheless while the user tries to add the product to the basket without login first and a message is displayed the prompts the user to login first in order to continue with the activity.**



The same message will be generated if the user tries to add a product to the basket while on the previous page.



## Basket page

**Basket page provides descriptive information about the products that the user has chosen while navigating to the site and are ready for purchase.**

**For the case of public users the page will generate a message that informs the user that the basket is empty and prompts for login.**

It is obvious that public users cannot exploit of the basket functions.

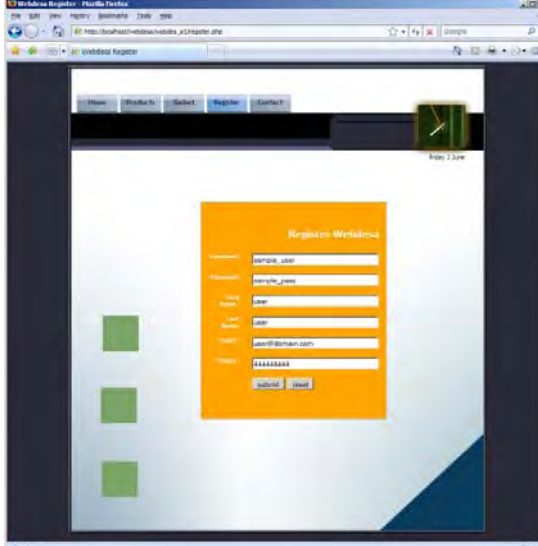
The procedures that are deployed for this page will be discussed in detail in a later section.



The page is the area where the users can provide their individual information to the system and obtain a registration. The user has to fill in for all inputs since all are credited essential for the registration.

The page incorporates an HTML form that is using a 'post' method to send data on a PHP script for evaluation and further processing.

In the side image the user fills with information the corresponding inputs and submits for registration.



A screenshot of a web browser displaying a registration form titled "Register Webstore". The form is orange and contains several input fields: "Username" (with "sample\_user" entered), "Email" (with "sample\_email@gmail.com" entered), "Password" (with "sample\_password" entered), "Confirm Password" (with "sample\_password" entered), "First Name" (with "John" entered), "Last Name" (with "Doe" entered), "Phone Number" (with "1234567890" entered), and "Address" (with "123 Main St" entered). There are "Submit" and "Cancel" buttons at the bottom of the form. The browser's address bar shows a local URL.

The next step is that the system passes the data to the PHP script that handles effectively in order to send them for recording in the database. The script makes the connection, enlists the data to a "insert" mysql statement and dispatch the instruction.

**A message informs the user about the successful registration!**

In the background, the HTML script incorporates an extra PHP tag that searches for an argument in the URL area. The message will echo nothing unless the URL message is retrieved.

When the external PHP script that handles the users input receives the data, it is programmed to dispatch two distinct activities with the database.



A screenshot of the same web browser displaying the "Register Webstore" form. The form now shows a success message at the top: "sample\_user, you have successfully registered the account!". The input fields are now empty, and the "Submit" and "Cancel" buttons are still present at the bottom. The browser's address bar shows the same local URL.

The first step is to check if the data already exist in the database. Therefore a statement with a unique A value is enlisted in a select statement and dispatched to the database. If the argument is found stored in

the database then the activity stops and an appropriate message is concluded in the “redirecting” URL.

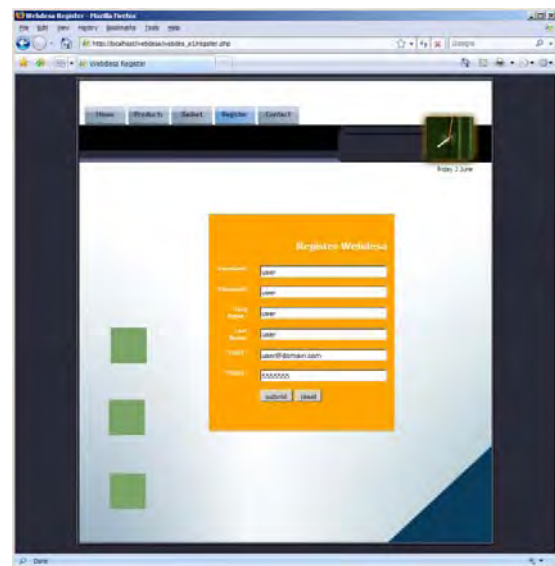
For our case the email of the user is checked.

**In the right side image we can see that a user with the same email tries to register the Webdesa database.**

In contrast if the argument was not found in the database, then a “insert” statement is developed with all of the form data and is instructed to the database. For this case “redirection” to the basket page will feedback with an appropriate message regarded to the successful impact of the data storage, like we saw in the first screenshot.

**The image demonstrates the behavior of the system after submitting with an already registered email.**

The site prevents user from registering while after searching the database found the email is already in use.

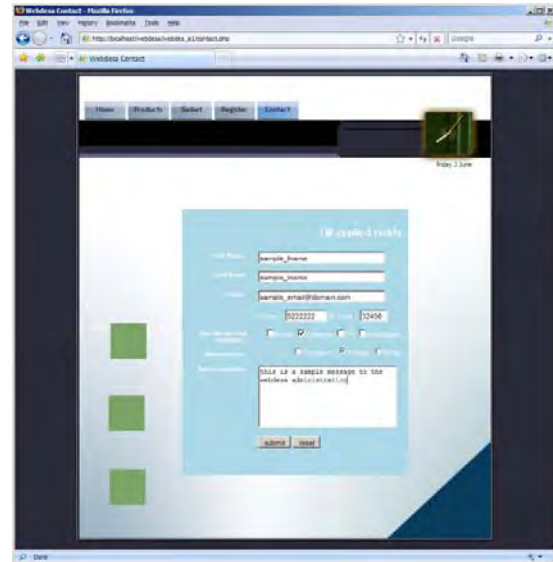


The page will maintain a solution while a tries to contact with authorized personnel of the Webdesa Company. Some additional information are in request for the users but those are not essential for sending the message.

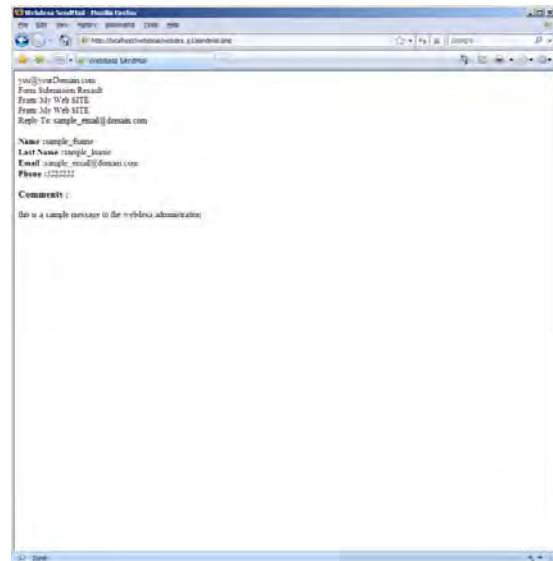
Again the form is handled by an external PHP script file that stores and manipulates data. Since there is no mail functionality the output is concluded and echoed to a HTML page only to demonstrate of the result of the data that were stored.

For the case of an existing e-mail the “mail()” function would be used passing as argument the content stored in the “msg” variable.

In the image user enters the contact area, provides with personal info and sends a message to the webdesa administration.



And the output that is echoed on an HTML describes all data stored by the system and will be sent while a valid email address has been inserted and a mail() function where implemented.





## b. Customers

### Login - Home page (registered users)

The page in addition to what has been described to Public Users pages maintain additional functionality for registered users for logging in the system. Of course the user has to be registered in the webdesa database before moving to login.

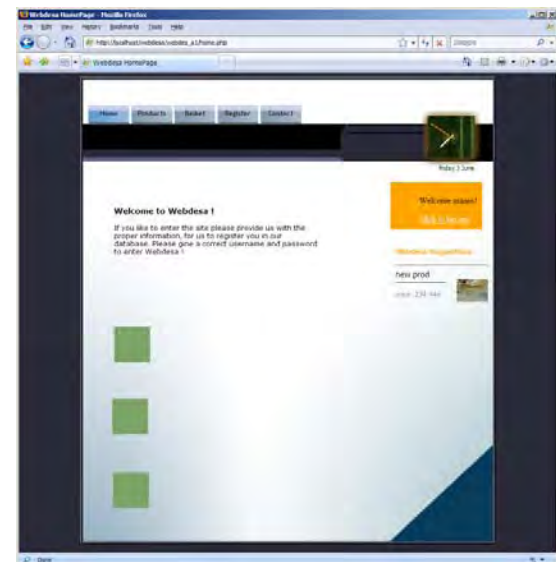
In the example as it is described in the side image a registered user attempts to login the system.

The website recognizes the user and let him login. In addition now the name of the user is displayed in with a 'welcome' message and the form that existed in the page has disappeared.

In this scenario there are more than a single procedures running at the same time where that of a maximum importance is the login one.

This is achieved as the form - while submitted - is handled by a PHP script. The script settles a connection to the database and inquires the username and password of the user. For the case that the user's inputs were found the system initializes a "session" for the user and exploiting it's functional integration with "cookies" functionality, names the "session" with the same name as the username. Therefore from now own it is easy to name the user calling at the session's stored name. Nevertheless session stores and additional PHPSESSID like `$_COOKIE['PHPSESSID']` but it makes no effective use for viewing since it is a complicated alphanumeric.

The form is hidden since the HTML form tag is triggered with a PHP script that indexing the page for an open session. If this is true the php script alters the name of the form's ID and hides it making use of CSS powerful features.



At last we can additionally see that a logout link is dynamically created when a user enters the system. The link will stop close the session while it is dispatched by a user.

This is accomplished because the dynamically created link references a PHP script that sets both of the cookies that are related to the session to contain blank values and that delegates to an effective way of closing sessions.

**The side image depicts the behavior of the system while the user emerges the log out function. Form is back on the viewing and the user's greeting is no longer visible.**



### Basket page (registered users)

Now let's find out how the basket functions for registered users that have added some products to it.

Since some of the procedures have already been previewed on the previous section we will escape referring to the background activity in detail.

**A user fills in with the inputs provided in the login area.**





And the system successfully recognizes the user.



The user moves to the basket page from the navigation menu.

First of all we can see that the system generates a different viewing for the registered than the one we preview before (in public). Now the basket displays contents and doesn't output any messages to the user.

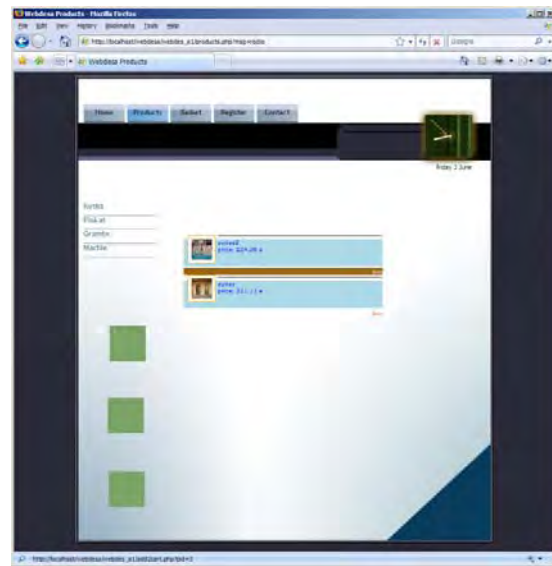
It is obvious that the user's basket is just empty.



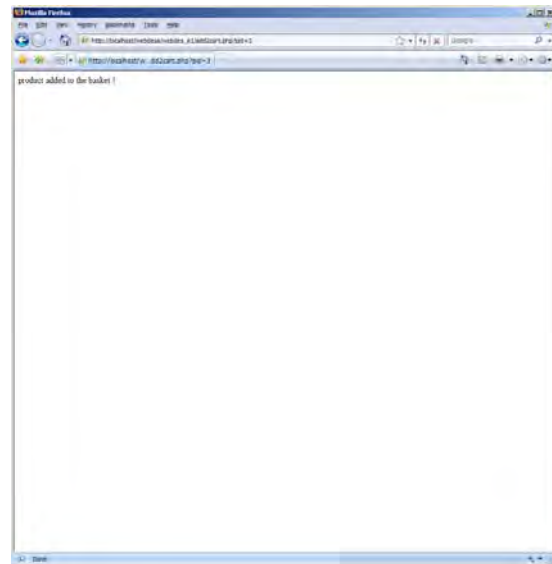
**The user moves to the products display area and puts a product to the basket.**

“Adding to the basket” activity is generated with an HTML link that makes reference to the “add2basket” PHP script. The links along with the reference to the script sends an inline argument to the URL with that stores the ID of the product. This function is generated in a previous script where the category products are retrieved from the database and presented on screen (products page).

So “add2basket” makes nothing more than connecting to the database and dispatching an “insert” statement to the “usercart” table (introduced in the MySQL section) enlisting the product ID and the username of the active session retrieves from the cookie superglobal.



**The system informs the user about adding a product to the basket successfully.**



The system as it developed for the assignment does not include functionality to remove products from the basket, or to clear the basket in total. This omitting was made only for time constraint reasons thus not exceeding deadline. While to conclude a removing function, since we have Product IDs and Username, using HTML links to trigger mysql “delete” statements would handle the objective adequately

Therefore we visualize that users with a discount of 25% will retain the proportion in the “discount-price” and the same is true for users with a 0% since the final price remain intact.



## c. Administration

### Login - Home Page (administration)

**An authorized person, like the super-administrator (sadmin) logs in the system.**

What is critical to point out, even though it will be obvious on the followings, is that the administrative personnel will obtain extra pages of contents. Moreover while pages are more than a single, those can only be generated through a root page that is labeled as “Content”. The “Content” page maintains a distinct button in the navigation menu that is visible only when an authorized person logs the system. This is achieved with an inline PHP scripting that exists in all pages of the system and is set to investigate the case where a PHP session is active and that the name of the established session matches the name of the administrator.

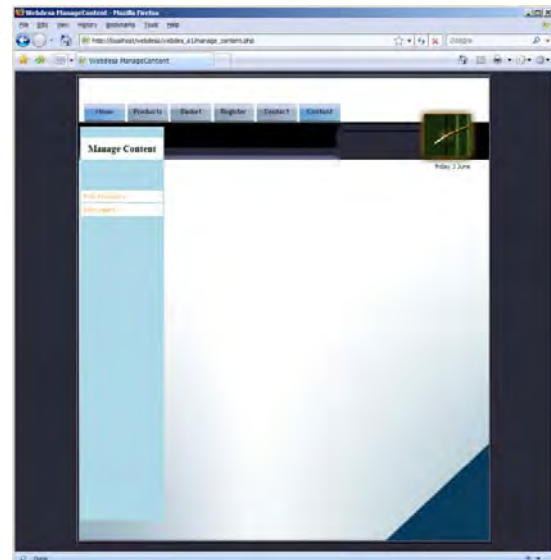


### Content (Manage Content)

**After the “sadmin’s” logins, the system recognizes the authorized user and instantly redirects to the “manage\_content” page (labeled Content in the navigation menu) where the user is empowered to invoke advanced operations on users and products**

What the admin displays in this page are two HTML links on a left side bar that prompts user to “edit users” and “edit products”. Both links reloads the same page though distinct messages as arguments on the URL will trigger different procedures and eventually distinguish the output content of the page.

This is fortified by the use of a PHP script that is set to seek for the URL argument on the page load and process the content accordingly.



In our scenario the “sadmin” follows the link to the “edit products” category and all products records from the database are retrieved and displayed.

In this case the argument of the URL has evaluated and the products scripting was dispatched.

Making a connection to the database the script retrieves and “echoes” back to the page all of the contents of the products table in an appropriately formatted and labeled HTML table that makes sense to the user.

**In addition, each row of the table is set to generate two extra links at the end of it. The links delivers function for each product to be edited or deleted while they are invoked respectably.**

In the background what PHP does for this two links, beyond the fact that they are created dynamically, is to store the ID of each product as an end line argument to the URL that they refer to. So when the new page loads, it will simultaneously “understand” what product the user wants to interact with. Nevertheless the arguments that are passed on the URL are two but we will clarify the case in a minute.

Additionally a “add product” link button is generated at the end of the products listing. The “add product” function triggers a much more undemanding process since it’s scope is only to load a new page, provide the essential inputs to the user and while completed and implicated sent a “insert” statement to the database.



## Edit product

**The “sadmin” selects a product to edit.**

The ID of the product “travels” along with the URL and while the new page loads a connection is made to the database and the corresponding values are retrieved.

The second argument that is passed along with the URL is either a “edit” or “delete” message that will assist the script to evaluate the user’s intention upon the record. That is because both “edit” & “delete” links, described previously, refer to the same HTML



page “edit\_products.php”. Therefore the message will resolve the case and the loading display content of the new page will be in regard to this message.

Another thing that is crucial for explaining the background activity in this certain case is the fact that the HTML form user displays, while on the “edit\_products” page, is loaded dynamically and the whole HTML structure is content in a PHP variable “\$product\_data” and echoed on page load.

This is essential as it provides a handy way for screening data on the form’s inputs making use of HTML “value” property for setting default values on the form inputs.

### Following the scenario, the “sadmin” alters the price and the description of the product.

Here we have to point out that even if the ID of the product is displayed on screen, the truth is that it is there only for administrative providing information to the user. In fact is not enabled to change value and any changes to this input will not be stored in the database.

It is just not participating the “update” statement that is sent to the database while the “update\_product” button is clicked.



### On the scenario “sadmin” submits changes and the system moves back to the listing of products.

The connection to the database and the “update” statement where invoked.

Changes made on the form have been applied in the database as well.





## Delete product

Now the sadmin decides to delete a product from the database because it didn't succeed in selling a lot.. The product's name is "new prod". The sadmin clicks the delete link next to the products description.

In contrast to the image above the product is no longer part of the Webdesa products table.

The case is very close to the previous "edit product" case. As it is mentioned above the link sends the product ID and the message "delete" to the mentioned page and embedded script handles the case.

Actually a connection is made to the database and a "delete" statement that evaluates the product ID is dispatched.

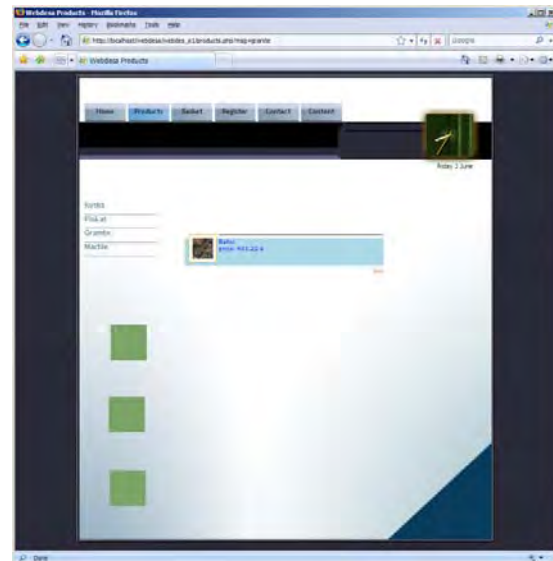


## Add product

Now the sadmin decides to register a new product to the database.

If we move on to the "products" page and follow the "Granite" link listed on the right side menu, before adding a new product, we would display the side image contents.

A single product is displayed.



Because the “sadmin” wants to populate the category with more than one product, he navigates to the “manage\_content” page (labeled Content on the navigation menu) and clicks on to the “add\_product” button.

A form is generated only for this case the inputs are empty.

From the functional perspective there is nothing new to describe on this page.

What it is happening here is that the page “echoes” a defined variable containing the HTML form; though for this case no default values accompany the inputs. There for the inputs are blank. The sadmin fills all the essential inputs and clicks the “add product” button.



The “add\_product” complied to build-in HTML form “action” interoperability is set to load the same page.

The PHP script that dominates the page is searching for a “POST” super-global variable with a product name indication while it is loaded.

Likewise the first time that the page is loaded there is no such variable retrieved therefore the form is screened.

Now that the variable exists the script will handle a connection and a “insert” statement to the database.



The content of the side image has changed.

The new product is listed at the bottom of the product table.



Now if we navigate to the products pages and move on to “granite” area we can see that the new product is displayed in the public listing area!



And additionally works efficiently as it generates a full description viewing while the image is clicked!



Since the “edit users” module potentials are in close distance of what has been described in detail in previous division (edit products), this section will be contended to a screenshots description and a textually labeled - user - activity, as to provide the evidence that it is functional and has successfully been implemented.

The “sadmin” enters the “Manage Content” area and clicks on the edit user link button that resides on the left sidebar. The system responds with displaying all of the registered users of the Webdesa portal.

The “sadmin” decides to update of a user’s discount-group membership and starts by retrieving his data from the database. The “sadmin” clicks on the “edit” button that is aligned next to each of the registered users.



The system retrieves and posts the user’s data properly on an HTML form.



The sadmin decides to also change some data of the user along to his discount group that now sets to value "2". Now the user should increase the discount proportion for purchasing products, since moving to a higher level category. This is something that we will have to find out later.



Up to now we can be confident that the user's data have been updated accordingly, as we can preview in the next image.



Now we will find out whether the “sadmins” change makes any difference and provides impact for the system.

Initially the user logs out the system.



And the new user fills in the login inputs to enter the system.



Webdesa recognizes the user and welcomes with a personalized message.





The user moves to add multiple products to his basket navigating to “Products” page and clicking the “add to basket” for different products. After finishing the user moves to the “Basket” page.

We can see that multiple product have been placed to the user’s basket and his discount benefit seems to provide the expected impact on the total cost of his purchase.



Delete user

Now we continue with the “sadmin’s” potentials under the “edit\_users” district.. The user retrieves the registered users from the Webdesa database.

The “sadmin” was informed that user label with a “user4” username has insulted the policy of the website and was recommend to remove him from the database of the system.



The sadmin dispatches the link next to the user and we can see the “user4 “is no longer part of the webdesa registered users.



Add user

Now the “sadmin” decides to register a new user by clicking on the add\_”user” link button, and the system response with providing a new form to continue with the activity.



The sadmin fills with all of the essential inputs provided by the system.

For the case it is considerably meaningful to point out that the “usergroup” while registering a new user to the system while in the background the process is set to assign a default value of “1” for each new entry that corresponds to a zero discount (0%) benefit for purchasing products.

Therefore no discount input is provided to the “sadmin” in the first place; nevertheless this is something that can be altered by the “sadmin” while invoking the update (edit\_user) module.



In the next image we can see that the new user has successful been registered in the Webdesa database.

