# Lab: Creating PizzaMore Web Site using C#, Bootstrap and CGI

This **tutorial** provides step-by-step guidelines to build a **“PizzaMore” app** in C#, HTML, CSS, Bootstrap and CGI. The app should implement **sign in** / **sing up** / **main** / **menu** pages.

## Password Hasher

We need to make some sort of security. But since we haven’t talked much about security, we are going to make it the easiest way we can think of. We are simply going to append the word SECRET in front of the given password. Let’s create the static class that is going to do so. Call it **Password Hasher** and put it in the **PizzaMore.Utilities.** In it, create a method, that is static string Hash(string password) in it, as we said, simply return the password, with the string “**SECRET**” in front of it.

## Sign Up

It is time to give ability to the user to register in our page. First of all, create new project in our solution called SignUp add **references** to the **Utilities**, **Data** **projects** and **Entity** **Framework**. The logic behind that page is pretty simple. **If** the request is **GET** **display** the page for sign up (the **static signup.html** page you already created earlier) otherwise **if** the request is **POST** **retrieve** the **data** from the POST request, **process** the **information** to create new user and add it to the database.

## Sign In

Again, add new project this time called **SignIn** and add the same references as in the **SignUp** project. The logic for that page is simple and very similar to the sign-up page. If the request is GET display the page for sign in (singin.html). If it is POST - try to find user with given email from the database, using the **PasswordHasher** class check if password is valid. If all validations pass, then create new session for that user and add that session to the database.

## Menu

Imagine our pizza restaurant wants their users to suggest their own pizza recipes and each user can vote for the pizzas they like. So, at the end of each week the suggestion with most votes would be the included in the menu for the next week as special offer. Only when user is logged in he should have the ability to add new suggestion, see the list of only his/her suggestions and delete them if he/she decide and vote for every suggestion that has been made from all users.

That page would be accessible only when the user is signed (has an existing session) in otherwise he/she must be displayed the redirect page provided in the resources (the dinosaur game).

The page would contain a navigation bar with several options:

* Add Suggestion – redirects to a AddSuggestion.exe
* Your Suggestions – redirects to YourSuggestions.exe
* Text in the format “Signed in as, {email of the logged in user}
* Sign Out – button that sign out the user and redirect him/her to the Home.exe

Let’s first implement the get of the Menu page:

In the method in which you are going to do this, first print the header, next call a method that you need to create, which is void GenerateNavbar(). Then print the content of the file called **menu-top.html**. After that call another method that you need to create again, called GenerateAllSuggestions(). Finally print the content of menu-bottom.html. Here is what you need to output on the console in GenerateNavbar():

Console.WriteLine("<nav class=\"navbar navbar-default\">" +

"<div class=\"container-fluid\">" +

"<div class=\"navbar-header\">" +

"<a class=\"navbar-brand\" href=\"Home.exe\">PizzaMore</a>" +

"</div>" +

"<div class=\"collapse navbar-collapse\" id=\"bs-example-navbar-collapse-1\">" +

"<ul class=\"nav navbar-nav\">" +

"<li ><a href=\"AddPizza.exe\">Suggest Pizza</a></li>" +

"<li><a href=\"YourSuggestions.exe\">Your Suggestions</a></li>" +

"</ul>" +

"<ul class=\"nav navbar-nav navbar-right\">" +

"<p class=\"navbar-text navbar-right\"></p>" +

"<p class=\"navbar-text navbar-right\"><a href=\"Home.exe?logout=true\" class=\"navbar-link\">Sign Out</a></p>" +

$"<p class=\"navbar-text navbar-right\">Signed in as {Session.User.Email}</p>" +

"</ul> </div></div></nav>");

In the GenerateAllSuggestions() you need to take all the pizzas from the database and after that do the following:

Console.WriteLine("<div class=\"card-deck\">");  
foreach (var pizza in pizzas)  
{  
 Console.WriteLine("<div class=\"card\">");  
 Console.WriteLine($"<img class=\"card-img-top\" src=\"{pizza.ImageUrl}\" width=\"200px\"alt=\"Card image cap\">");  
 Console.WriteLine("<div class=\"card-block\">"); Console.WriteLine($"<h4 class=\"card-title\">{pizza.Title}</h4>");  
 Console.WriteLine($"<p class=\"card-text\"><a href=\"DetailsPizza.exe?pizzaid={pizza.Id}\">Recipe</a></p>");  
 Console.WriteLine("<form method=\"POST\">");  
 Console.WriteLine($"<div class=\"radio\"><label><input type = \"radio\" name=\"pizzaVote\" value=\"up\">Up</label></div>"); Console.WriteLine($"<div class=\"radio\"><label><input type = \"radio\" name=\"pizzaVote\" value=\"down\">Down</label></div>"); Console.WriteLine($"<input type=\"hidden\" name=\"pizzaid\" value=\"{pizza.Id}\" />");  
 Console.WriteLine("<input type=\"submit\" class=\"btn btn-primary\" value=\"Vote\" />");

Console.WriteLine("</form>");  
Console.WriteLine("</div>");   
Console.WriteLine("</div>");

}  
Console.WriteLine("</div>");

Since the upper method is going to display all the suggested pizzas, lets now make the pizza suggestion possible.

## Implement AddPizza

Create a new console application called AddPizza. Again, check if the user is logged and if he is not, display the PageNotAllowed(). Now that we’ve made some security concerns it’s time to respond to the different request types. If the request is get, we must show the addpizza.html and that’s it. However if the request is POST, we must retrieve the parameters. Get every single parameter that is passed, by checking what they are from the form in the html file. Then fill all the data for the newly created pizza. There is some data that does not come from the form, so for this data of the pizza, put the default value of the type. Finally add the pizza, to the pizzas of the current user and give the user the same html that he got from the get request.

## Finish Menu

Since we can now add pizzas, they are going to be displayed in the Menu page. Therefore, we can vote for them from there, so the menu script now has to be able to handle not only get, but also post requests. In the block that is responsible for the post method, you must call the void VoteForPizza method that we are going to create right now and after that print the same content you did in the get method. You are passed two parameters. Check what’s their names in the html and you should be able to figure out what to do from now on.

## Your Suggestions

As you might guess. If there is on session, you should give the user the appropriate page. If the request method is get, call the ShowPage method. If the request is method is post, call the DeletePizza method and then the ShowPage. The ShowPage is pretty similar to the implementation of the Menu. First print the header, then print the file content of **yoursuggestions-top.html.** After that PrintListOfSuggestedItems and finally print the file content of **yoursuggestions-bottom.html.** In the PrintListOfSuggestedItems you should take the pizzas that belong to the current user and after that put the following code:

Console.WriteLine("<ul>");

foreach (var suggestion in suggestions)  
{  
 Console.WriteLine("<form method=\"POST\">");  
 Console.WriteLine($"<li><a href=\"DetailsPizza.exe?pizzaid={suggestion.Id}\">{suggestion.Title}</a> <input type=\"hidden\" name=\"pizzaId\" value=\"{suggestion.Id}\"/> <input type=\"submit\" class=\"btn btn-sm btn-danger\" value=\"X\"/></li>");  
Console.WriteLine("</form>");  
}  
Console.WriteLine("</ul>");

You have to figure out on your own, how to delete a pizza, since it’s quite similar to things we’ve done till now.

## Implement DetailsPizza

Finally, since we would like to see some details for a selected pizza from the Menu, there should be a separate page that displays this. Make a new console project called DetailsPizza and it’s only going to handle get methods. What you need to do is get the parameters from the query and then find the pizza with the passed pizzid. Then you can paste the following:

Header.Print();  
Console.WriteLine("<!doctype html><html lang=\"en\"><head><meta charset=\"UTF-8\" /><title>PizzaMore - Details</title><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\" /><link rel=\"stylesheet\" href=\"/pm/bootstrap/css/bootstrap.min.css\" /><link rel=\"stylesheet\" href=\"/pm/css/signin.css\" /></head><body><div class=\"container\">");  
Console.WriteLine("<div class=\"jumbotron\">");  
Console.WriteLine("<a class=\"btn btn-danger\" href=\"Menu.exe\">All Suggestions</a>");  
Console.WriteLine($"<h3>{pizza.Title}</h3>");  
Console.WriteLine($"<img src=\"{pizza.ImageUrl}\" width=\"300px\"/>");  
Console.WriteLine($"<p>{pizza.Recipe}</p>");  
Console.WriteLine($"<p>Up: {pizza.UpVotes}</p>");  
Console.WriteLine($"<p>Down: {pizza.DownVotes}</p>");  
Console.WriteLine("</div>");  
Console.WriteLine("</div><script src=\"/pm/jquery/jquery-3.1.1.js\"></script><script src=\"/pm/bootstrap/js/bootstrap.min.js\"></script></body></html>");

## Logout

Finally, if you want to log out from the menu, it will send get request with some parameters to the home.exe. In order to handle the request, make a new method called TryLogOut() and put it in the beginning of the GET if statement in the Home.cs. The implementation is in your hands.

## Test the program and start improving it

Now that you are done, you should copy all the files from all the bin directories and paste them in the cgi-bin.