Coursework 2: Dynamic Web Applications

SOFT20181: Internet applications programming

Benjamin Doe (N0561281)

2016

# User Interface Design

I based my user interface (UI) off of current restaurant sites such as Hungry House, Just Eat and Wetherspoons. I also considered design heuristics, more specifically the ten general principles defined by Jakob Neilsen. Other design and usability considerations are mouse movement distances keeping in mind Fitts’ law and also visually pleasing and easy colours. The Heuristics are more “rules of thumb” rather than strict guidelines that systems with user interfaces should consider and use. However, they do enable a more pleasing and user friendly experience.

## Error prevention

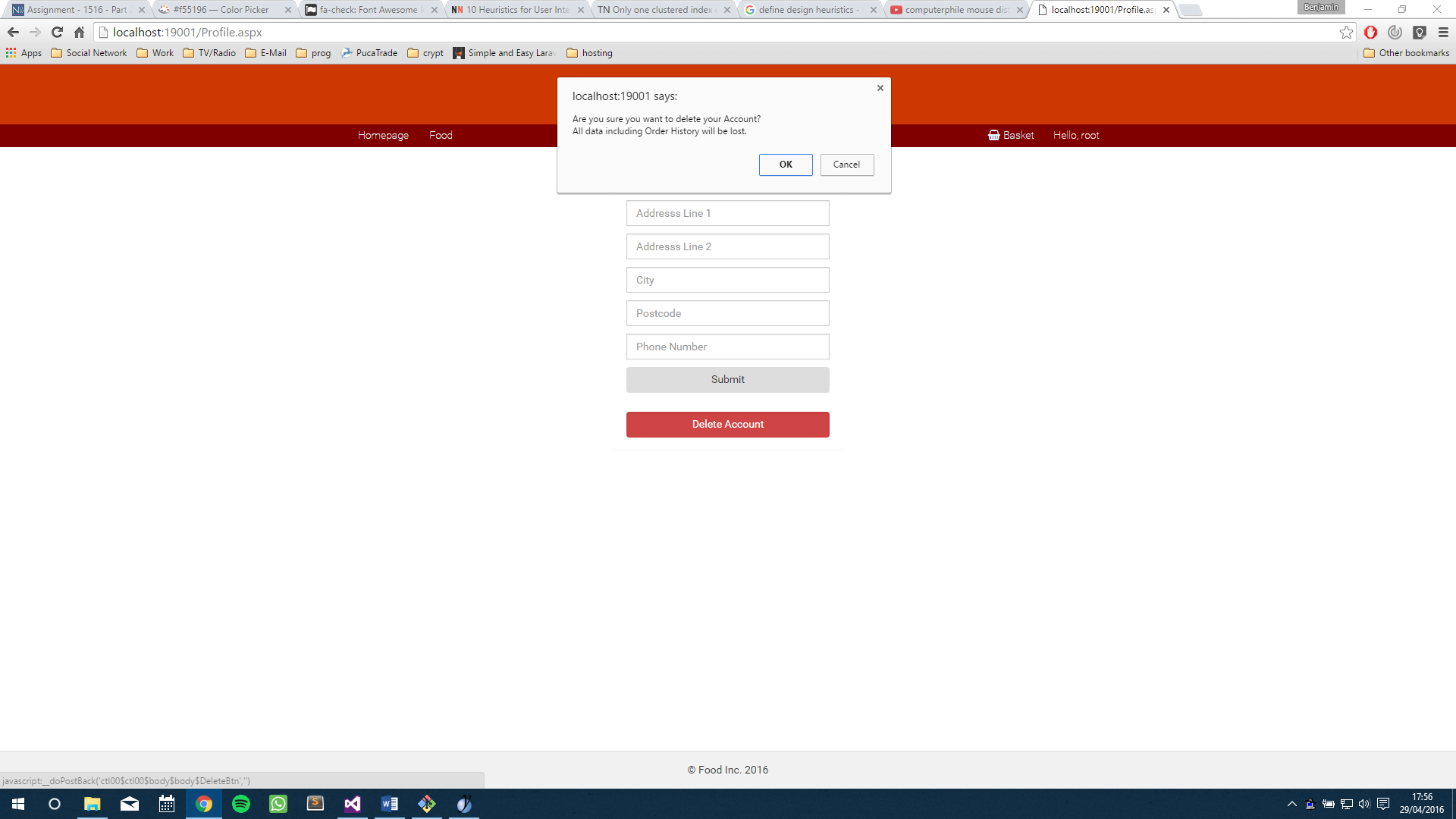
ASP has a very easy way of dealing with input errors using their validation controller which can deal with an array of things from empty inputs to patterns. However, because HTML5 can deal itself with regex patterns, I went with the design decision to implement any regex and error checking to the HTML attribute. This should, theoretically, reduce the time it takes the IIS server to return a page because the IIS server would need to compile the ASP files and return them into HTML, something that the browser can read. I have also implemented some JavaScript (JS) popups and confirm boxes so that if the user were to accidentally click a delete button, it would ask the user whether or not the user wanted to really go ahead with the decision.

Figure 1 - Example of a confirm box shown to the user on the profile page when they click "DELETE ACCOUNT"

## Visibility of System Status

The idea of this is to tell the user what is going on or at least make it clear to the user what is happening with the system. In terms of web design, this could be a small message to tell the user that their request has been handled successfully or a simple redirect to a page to show that the request responded with success or failure.

## Minimalist Design

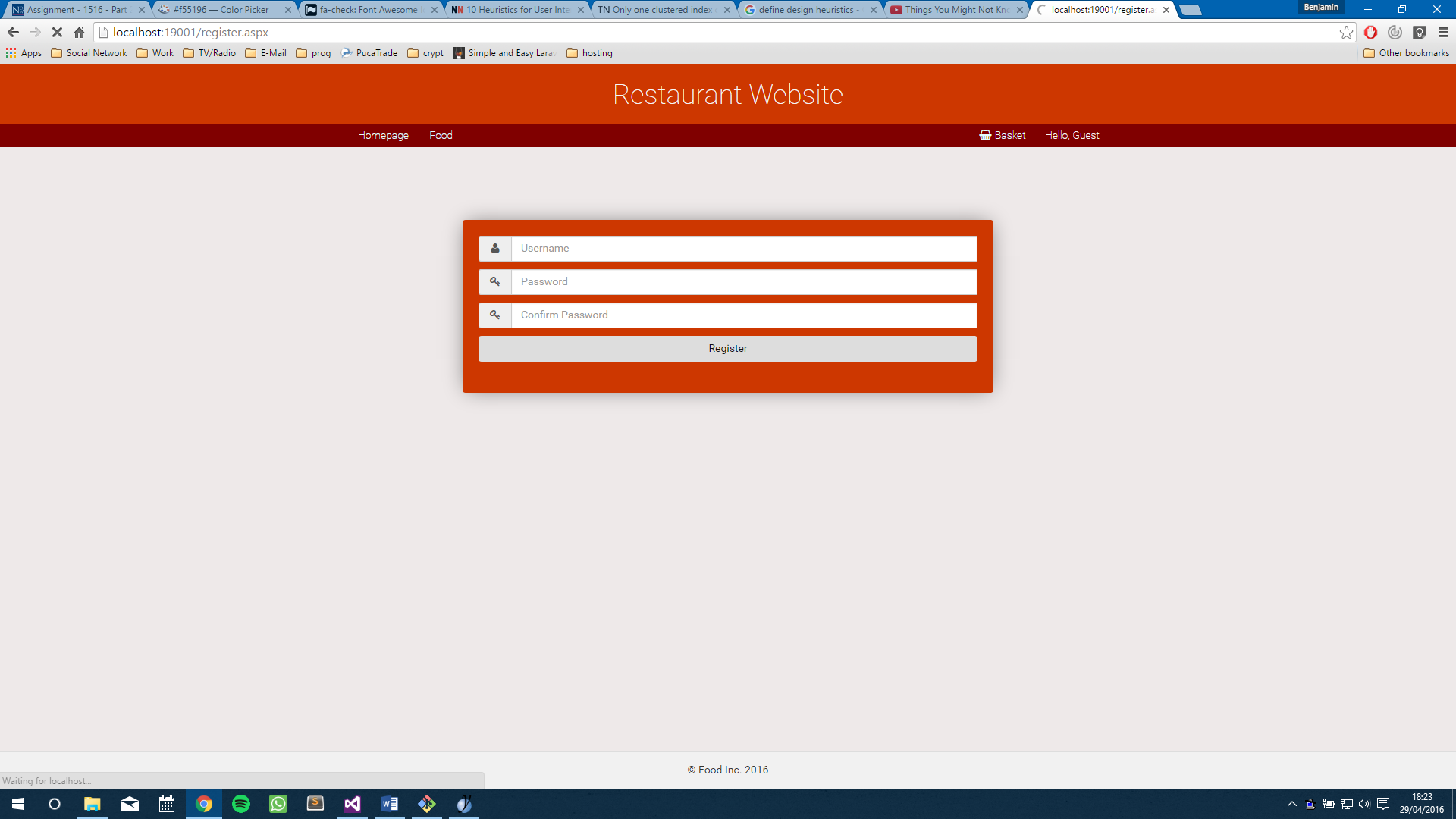


Figure 2 - Website header displaying minimalist design

A minimalist design means that the website is aesthetically pleasing but then also means that the HTML document should be loaded quickly because of the things that need to be loaded. In a minimalist design, there should be as few things happening on the page as possible and the things that are happening are bare essentials or things that would enhance the design of the User interface. This should lead to the browser having to render less things and therefore give a much faster load time.

## **Efficiency of use**

The browser needs to load the document as fast as possible so that the user feels that the website has been built to a high standard and also so they are also happy with the overall user experience. Google Webmasters recommends a load time of about 0.5 seconds. This is a recommendation after all so it isn’t set in stone however it is a good guideline. Things that can hold this back is mainly the server and the speed on the server machine which the site is being hosted on. For testing, I had to use my laptop, which is quite powerful but nowhere near an enterprise or server grade machine. For this reason, I’d say that my average page load time was 3 seconds which is very slow and sometimes it has been as slow as 10 seconds. The way that I could have reduced this is I could go through the code and refactor it in a more efficient way that would take up less processing power which would lead to a more professional and quick website.

# Database Design

The database used in this is a Transact-SQL database that can be accessed either through Visual Studio using their Server Connections tab where you can write DDL commands and insert data. Within the ASP code, you need to have a connection string, which is a string that points to a database file or a database server, and set up a SQL connection with the connection string. Once a connection has been set up and opened, SQL code can be executed and the results can be used in the code. Another method of getting the data into the ASP pages is by having an SQL Data Source that is added to the mark-up file and contains queries to fetch and alter data.

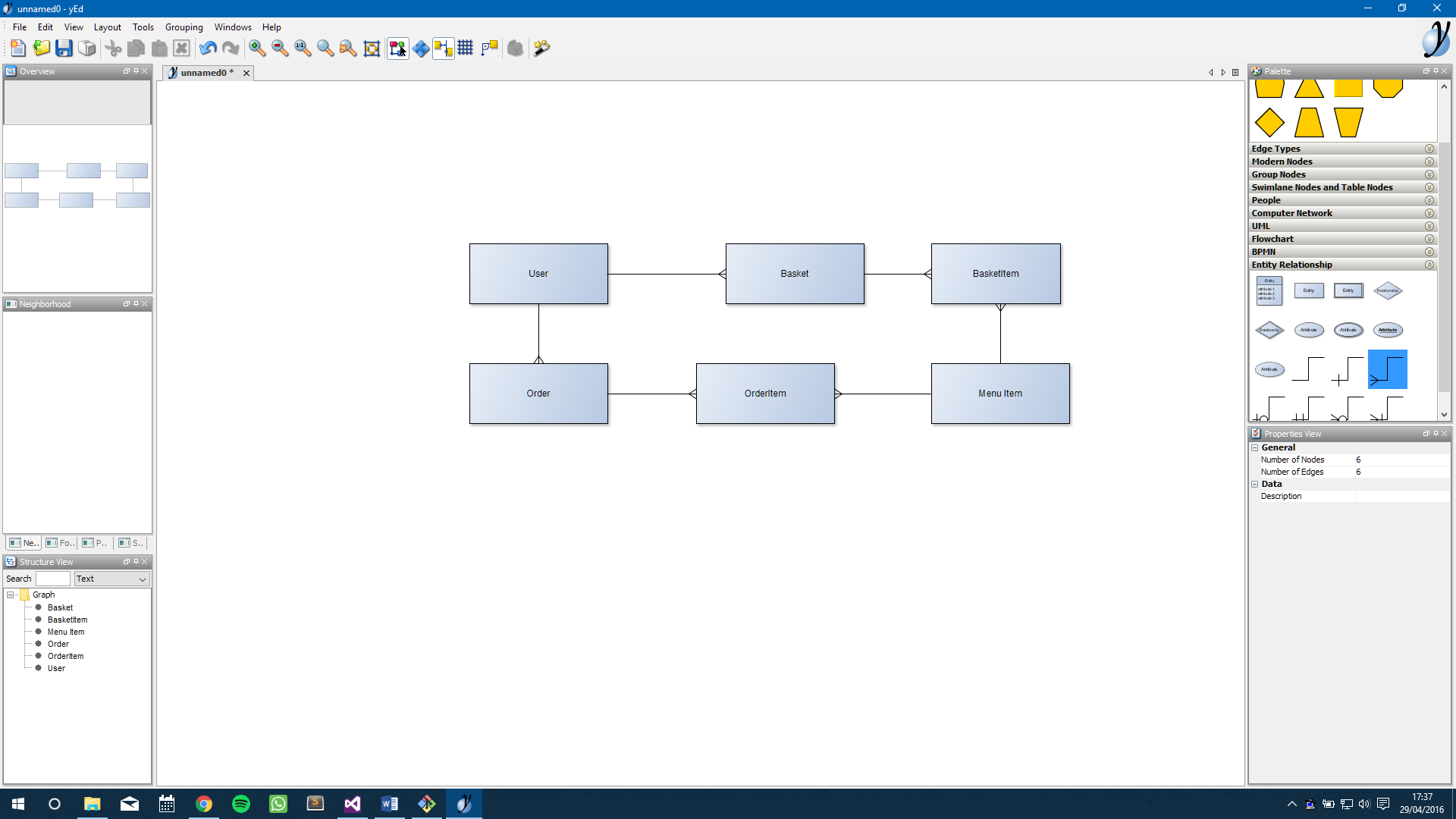


Figure 3 - Entity Relationship Diagram of initial database design

When I had designed the system, I intended on having a database-driven basket for registered users. This would have done exactly as the basket does that is implemented right now except more because the basket would have been accessible from any device that the user is logged into. However, I did manage to complete an order history system for the website that stores the past orders of a registered user. Later I saw that favourites would have been a good feature to have on the website and that this could just be a junction table with the item and the username as a composite key but this was too late to implement. The way I would have implemented it would have been by giving the user an option to click a hollow star that could have been placed next to the title and when clicked would have been filled in.

# Login Design

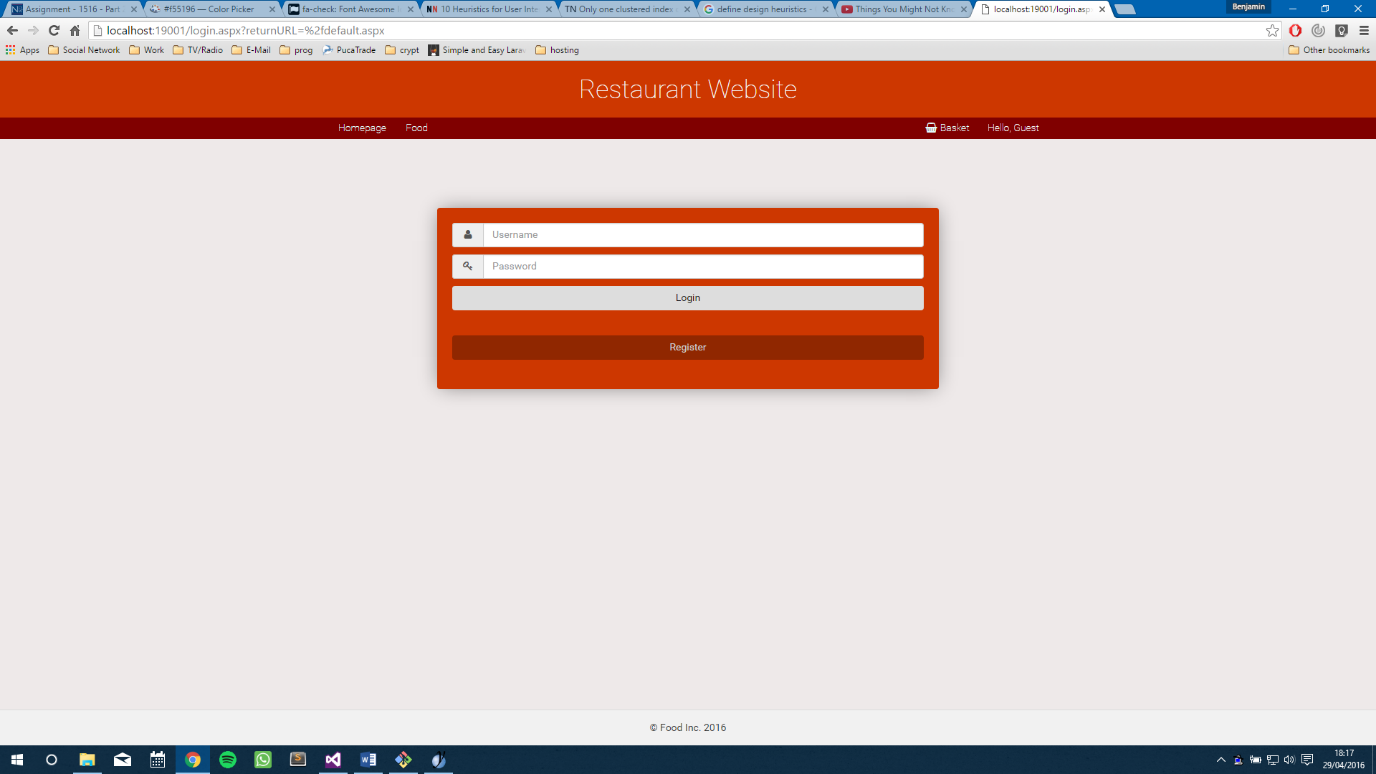
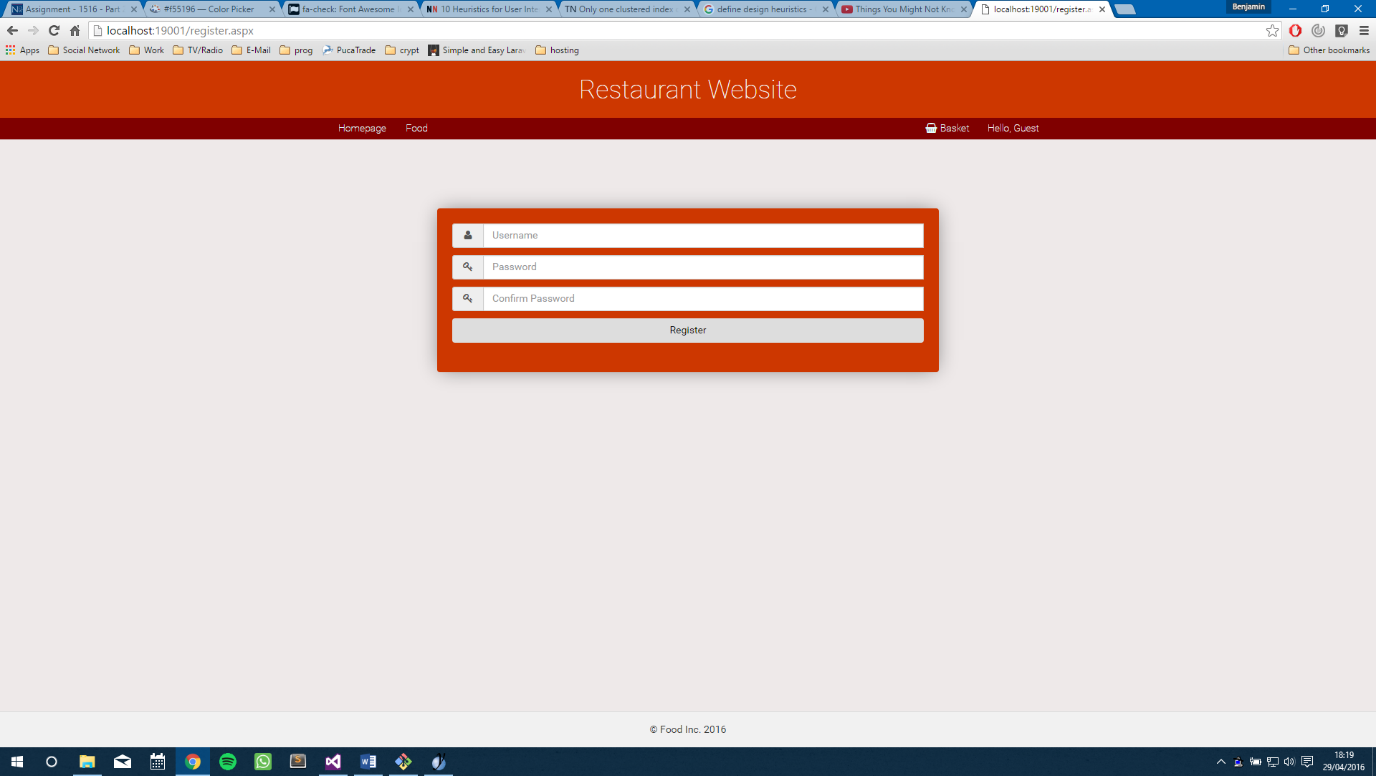
The login and registration pages are done using sessions. The reason I took this approach is because this is the approach that I understood because I have had past experience with it. Indeed, later on I did find out that ASP has an inbuilt function that deals with authentication in a secure manner. The function is called FormAuthentication and given more time, I would have loved to implement this because it also is able to manage roles for a website and therefore, could have managed members from admins. The way that I handled admins is that I had a bit field in the database that stored a value of 1 or 0 and defined whether the user was an admin or not. The session stores data about the user such as their username and also whether or not they are an admin for the site. The website checks these things on page load on the admin pages and depending on whether there is someone logged in, the client is redirected to either the homepage or the login page. If the client is an admin determined by the cookie, they will be allowed access to the admin page. This means that the website employs a single-sign-on system so that the client would only have to log on once to access all the permissions they have been given.

Figure 4 - Registration Page

Figure 5 - Login Page

# Food Menu, Order History and Basket Design

The food menu was based off of the Hungry House and the Just Eat menu. They had a clean, plain look that was clear to the user immediately what was what and what costed what. The menu items are added to the menu through the admin area and are then displayed in the main site menu page. Each food item has a button associated with it which calls a function in the code behind that triggers an action whereby the relative item is added to the cart. I designed this with mobile in mind so the menu is as wide as possible to maximise screen width and vertical space is saved so as much data can be put on one screen at a time. The model of the basket is a series of classes that maintain the basket structure and the items within it. The basket object can then be stored in a session or a cookie to be retrieved later.

When the user clicks the “Add to basket” button, the code will as the basket class to go and retrieve a basket object from the session cookie. The class will then give the menu the basket and the menu will add the object to the menu. After this, the basket class then goes and saves the basket back into the session. There is an object stored within the basket object called which is the basket item and this basket item holds the quantity and an object of the product. This level of separation means that it is easier to alter the product class without affecting the functionality if the website. In the basket page, the page asks again for the basket and instead of manipulating the data, it displays it in a list view which is controlled by a cookie in the code behind. On this page the user can then “checkout” their basket by pressing the checkout button. This then manipulates the database and adds the basket data into the database structure ready to be processed by an admin manually.

The order history page shows the information about each order including the ID number or “Ref Number” as I have displayed it as and also the quantity and item and subtotal. The order history also includes a widget that displays the process status of the order. An order can be either Processed or Pending. The admins in the admin area have to manually accept each order. This is so that the admins can delete the spam orders if someone decides to be a funny man. When the order is pending, a small spinner shows that tells the user that their order is complete. If the order is processed the page shows a green tick. This isn’t controlled by JavaScript so it won’t update as soon as the order has been processed however this is a good point for improvement.

# Admin Area

The admin area consists of three pages; user control, menu control and order control. Each control does exactly as it says. The user control page can view registered users, remove users and also make admins. The admin column is the only column that the admin is allowed to edit for security and integrity reasons.

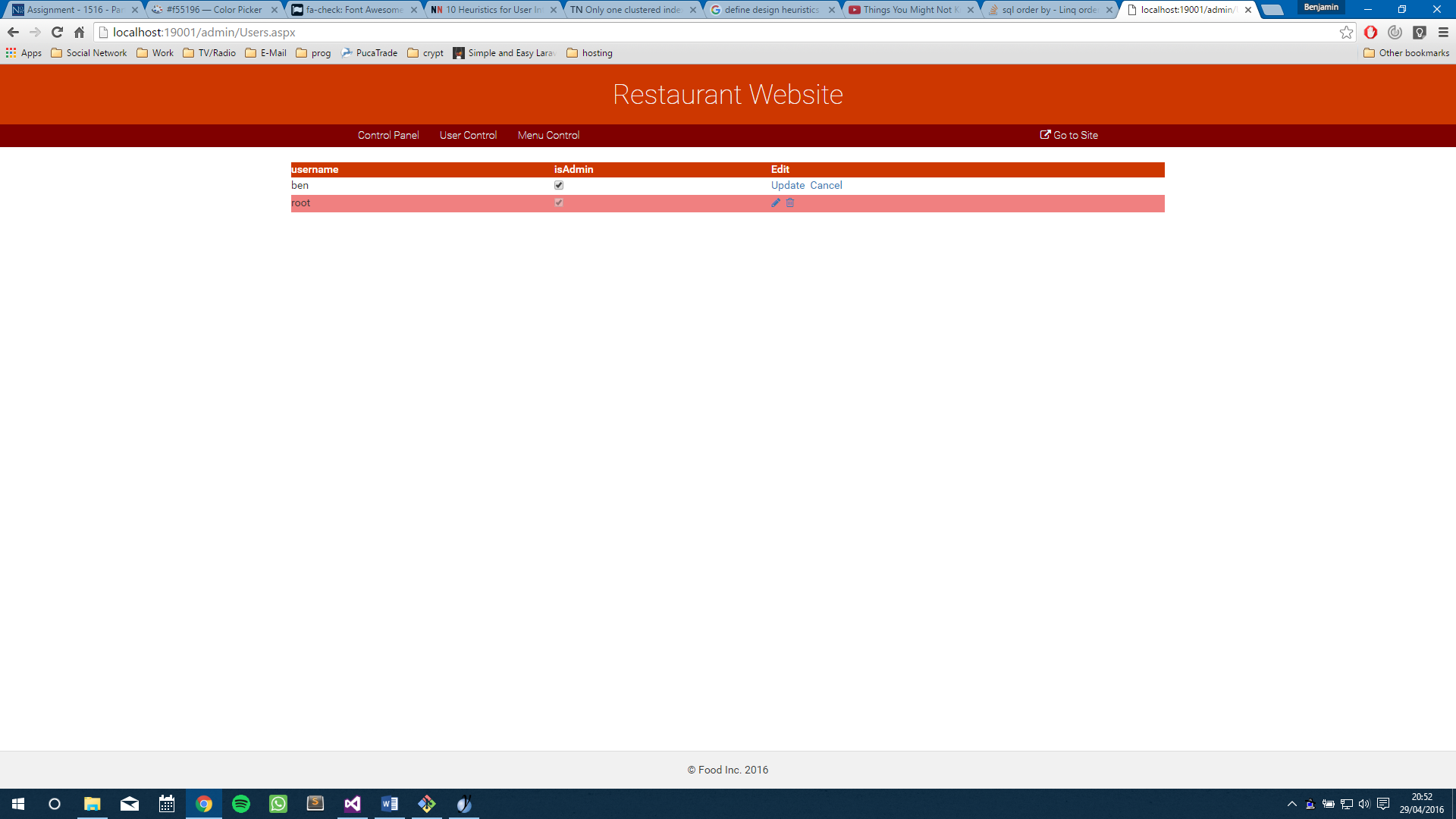


Figure 6 - Admin editing username "ben"

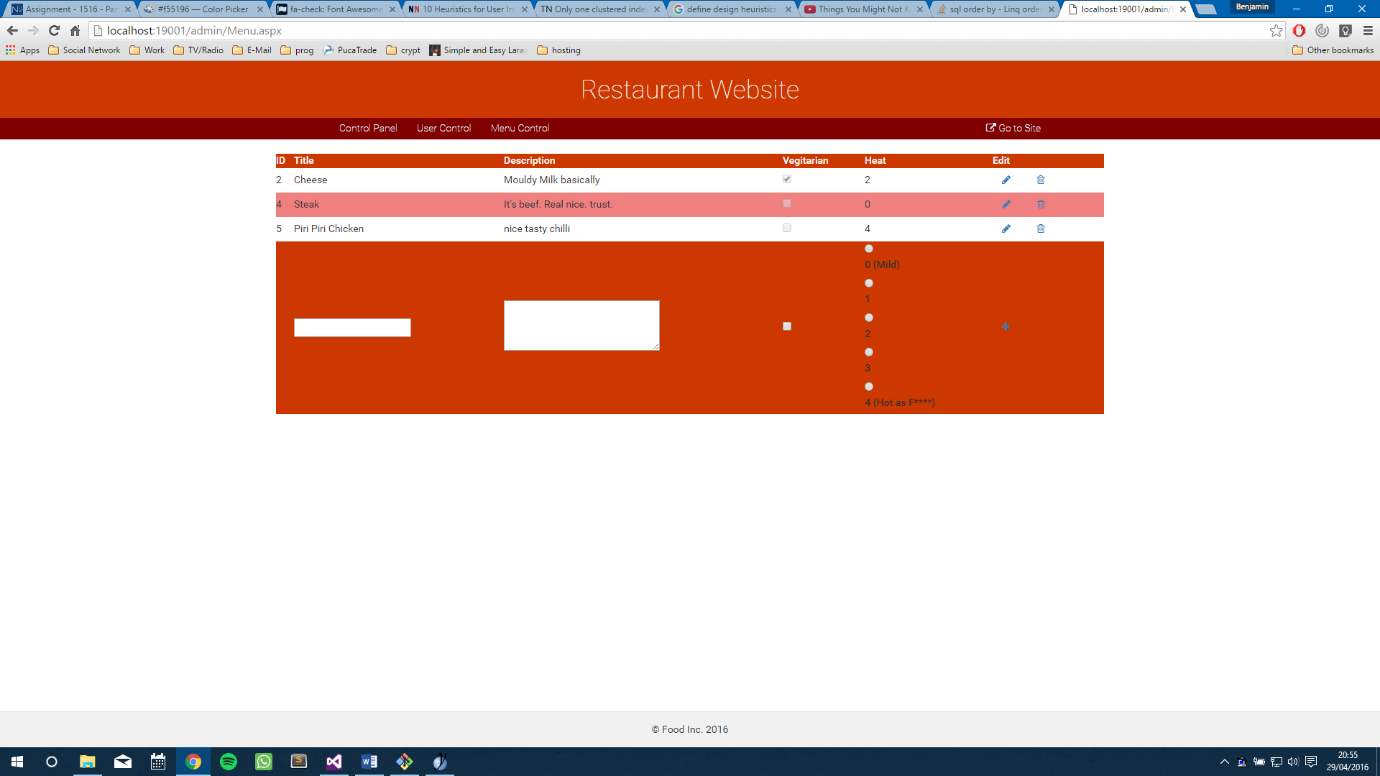
The menu control page controls the menu items in the database. Just like the user control, it can edit and delete only the menu control can edit all of the columns except for the ID field for, again, security and integrity reasons. Here, the admin can add more menu items trivially by adding the data into the textboxes then submitting it. The menu control also features a radio button list which is controlled by the spiciness of the food and scales from 0 to 4, 0 being no heat whereas 4 is extremely hot. Another feature here that I would have liked to implement is the upload of images to the database.

Figure 7 - Menu Item control

The order control page controls the current orders of the site and allows the admin to accept or process orders. The grid view is ordered by the processed Boolean field with the pending orders first. The admin has to ability to remove orders from this list too.

# Problems faced and overcame

One of the initial problems that I faced when starting to code in ASP was the transition between PHP to ASP. The big difference for me was that the ASP code and mark up were interwoven because it is all compiled backend before it is presented to the user. The mark up interacts with the code and the code can select elements from the mark up using their ID given that they are given the attribute “runat=server”.

Another major issue I had was with junction tables and the SQL data reader. Because the data reader stored things in rows, I found it difficult to try to access many to many relationship data. This was used for displaying the order history page. What I ended up doing was checking whether an order of the ID that the reader was on had been created and if it had then I would append that to the order control otherwise I would have to go and make a new control to hold the new order. Doing it this method meant that it was slightly quicker since I would only be using one loop as opposed to a nested loop however because this method was a bit obscure and I didn’t spot this solution immediately, I struggled for a while with this and I feel I wasted time on this where I could have been implementing other things.

# Planned functionality not implemented

There are a few features that I would have liked to have implemented had I had more time on the website. One major one is the ability to add images to good items and work with uploading images. I feel that with my timeframe given, I wouldn’t have been able to find a good solution to the problem. The reason I feel like this is because I had difficulty with the grid views throughout the project and I feel like trying to get a file upload form into the grid view would have been a feat. No doubt doable but a feat nonetheless.

Another feature, this is quite a small one, but another feature would have been a favourites system within the website. The favourites would have been implemented for the food menu page so that the clients could select their favourites and then a shortlist page would have been added under the user navigation controls to the right of the navigation bar. This would have been a nice feature that would have given the website a more interactive and also a more usable website making mouse clicks less frequent.

In the admin control for the orders, the admin can’t actually see the menu items associated with the order. I knew the method that I had to take to make this approach however I couldn’t grasp it in time and was forced to let this functionality go. I would have gone for a bulleted list inside of the grid view and then populated it with the menu items using the order ID of the row to select the menu items from the junction table.

# Disclaimer of ownership

I, Benjamin Doe, hereby state that the code and mark up in the coursework for this module is my own work. I also declare that I have received no outside help from anyone who isn’t a member of staff at the university. The report that is also contained in this document is all my own work. All code, mark up, and descriptive text of the coursework was created and tested by me.