**CST 350: Activity 2**

Michael Melichar

CST 350: Programming in C# III

Brandon Bass

3/26/23

**Adding Text to a View**

**Graphical user interface, application

Description automatically generated**

*This shows adding text to an HTML page using header tags*

**Different Views**

**Graphical user interface, text, application

Description automatically generated***This shows adding a message to a different view in our application*

**Using Variables in View Output**

**Graphical user interface, text, application

Description automatically generated**

*This shows adding a personalized message to a secondary view based on the user’s input.*

**Key Takeaways from Part 1 of Activity 2**

Key takeaways from this part of the activity included how to make different views show different things, whether they are hard coded using text, html tags, or are implemented by way of variables, there are many different ways to modify the view that gets returned back to the user.

**Part 2**

**Using Input Fields**

**Graphical user interface, text, application, website

Description automatically generated**

*This part of the activity demonstrated how to make username and password fields in the view that would be passed to the controller in order to verify a potential login.*

**Successful Login Attempt**

**Graphical user interface, text, application

Description automatically generated**

*This demonstrates a successful comparison against the username and password hard coded into my program from what was submitted to the form, shown above.*

**Failed Login Attempt**

**Graphical user interface, application

Description automatically generated**

*This demonstrates a failed attempt to login as the username and password did not match the username and password hard coded into my code.*

**Key Takeaways from Part 2 of Activity 2**

This part of the activity showed how to pass information between all 3 major components, the model, view, and controller well. I don’t think it is entirely intentional that the assignment was pretty out of date, and I had to do a lot of my own research to find out what deprecated methods I needed to replace with new things, but I feel like I understand it better than just retyping what was on the activity document. It isn’t necessarily pretty, however it showed at a base level how to pass information between these three critical pieces of web application architecture.

**Part 3**

**Form Data ValidationGraphical user interface, text, website

Description automatically generated***This shows how to add data validation server-side in order to not allow a user to skip by authentication by modifying what they see on their end and sending it back to the server as if it were valid input.*

**Additional Custom ValidationA screenshot of a computer

Description automatically generated***This demonstrates the ability to add custom rules to data validation in order to further limit the data that can be input into a form.*

**Key Takeaways from Part 3 of Activity 2**

This part of the activity has not a lot to show in comparison to the amount of work that needed to be had in order to complete it. HTML tags have changed a lot, and the asp action tags are deprecated. A lot of what I found online was saying to use those, and only finally after finding the new @Html.Methods was I able to complete this task. There’s not much to this – you put up a rule that says you can’t input something unless it falls within certain guidelines on the model and the view checks if what the user input is within the model’s parameters. If it isn’t, you send the user back to the same view they started at through the controller, but now with nice error messages telling them where they went wrong. If they succeeded, you pass that information into a different part of the controller’s methods in order to move them to a successful registration or login screen, etc.

**Part 4**

**Showing Object Variables**

**A screenshot of a computer

Description automatically generated***In this screenshot, I was able to print the button states of various buttons that would later be placed in my form. This is just passing around data through the MVC in a different form, but it’s with an object this time.*

**Displaying Images**

**Graphical user interface, text, application

Description automatically generated**

*This demonstrates inserting images that are located within my project’s directories*

**Displaying Lots of Images!**

**Graphical user interface

Description automatically generated with medium confidence**

*This time, we use a for loop to display a whole lot of different images of buttons*

**Displaying Lots of Images… in a Grid!**

**Chart

Description automatically generated**

*This demonstrated using the mod function to display my buttons in a grid*

**My Grid Has Labels!**

**Graphical user interface

Description automatically generated**

*In this image, we demonstrated adding labels to our grid. I also had it centered somehow, and then I was convinced that it broke some functionality, so I removed the centering entirely for the next parts of this assignment.*

**The Button Game is Born!**

**Graphical user interface

Description automatically generated with low confidence***After a lot of fidgeting, I was able to make it so only one button at a time was edited.*

**Proof That I Didn’t Get Randomly Lucky With An All Green Button Game Last Image**

**A picture containing icon

Description automatically generated**

*This image shows a second state that definitely is not randomized or I’m buying a lottery ticket*

**The Game Speaks!**

**Graphical user interface

Description automatically generated**

*Through a few checks, the game now checks if all of the buttons have the same ButtonState. If they do, it will display a less than fun message telling the user so, because I forgot color was an attribute after working so much on button states.*

**Key Takeaways from Part 4 of Activity 2**

With the MVC architecture there’s a lot that can be done, but this really demonstrates the ability to manipulate objects, variables, buttons, forms, you name it, this incorporates it. This shows off how one can change just about anything on a view from the click of a button. It’s not just about doing something simple like changing a button color, with better designed code you can transform a view in a single click or scroll, and this encapsulates that thought process pretty well.