Tanner Croom, Katie Hand, Garrett Jones

Team 2; ISTM 420-501

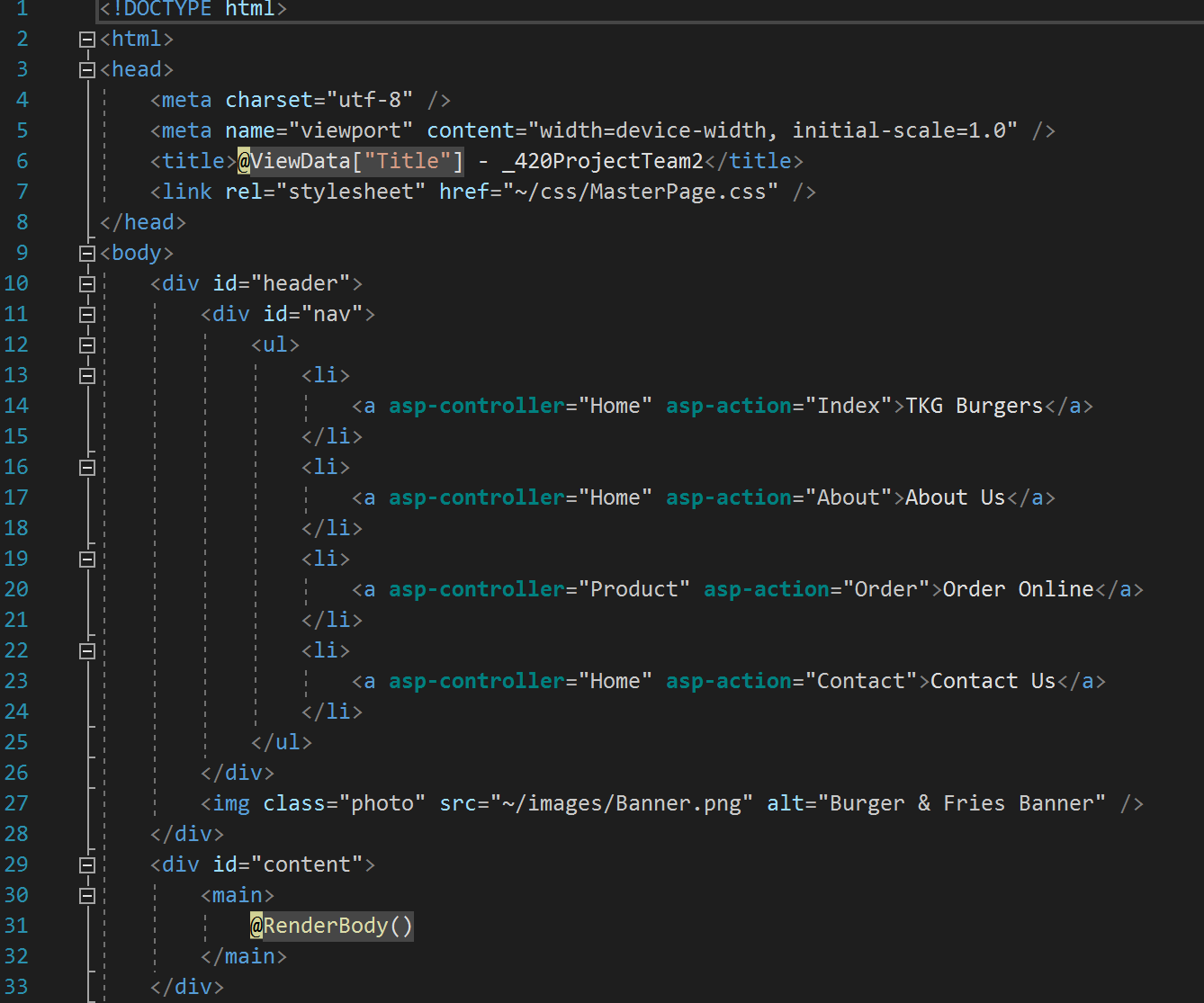
Spring 2022

Project Final Report

***MasterPage***

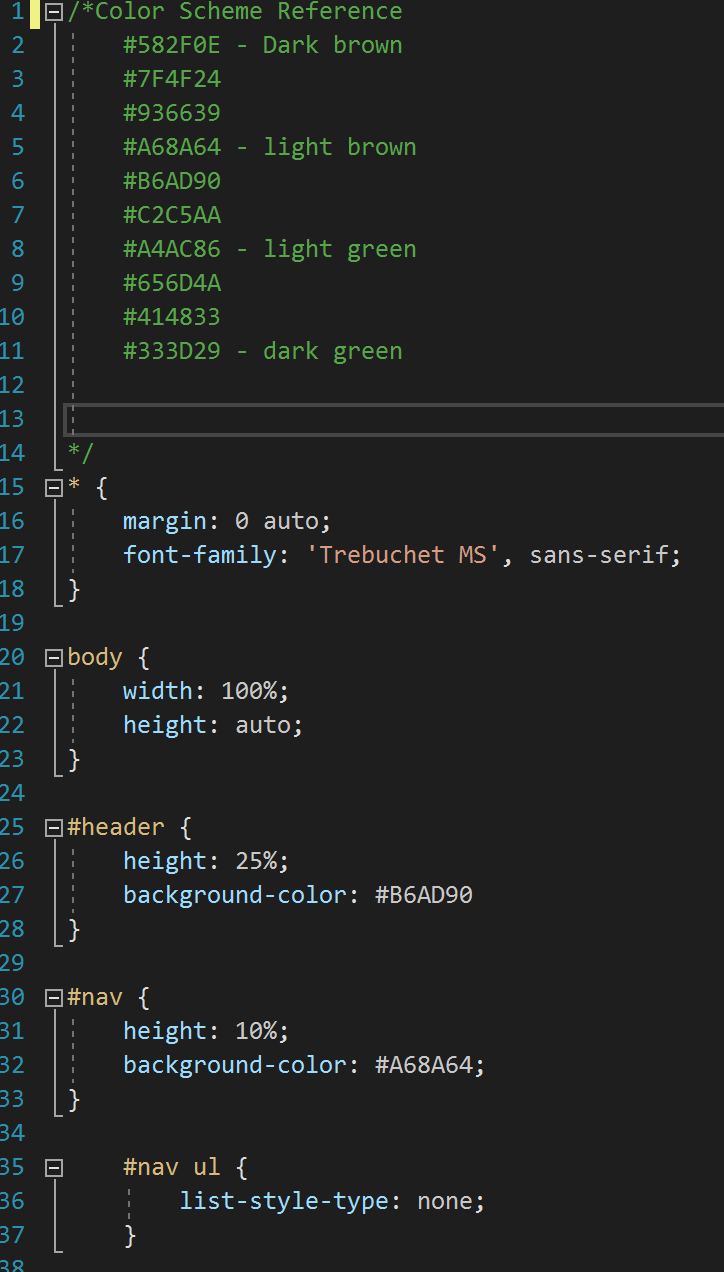
Our MasterPage serves as the Layout of our web application. The body is split into three main divs, a header, content, and footer. The header contains both our navigation div, as well as our banner to properly welcome users to each of our views.The navigation div holds links to each of our main views: Home, About, Order, and Contact. The content div consists of the “RenderBody” Razor expression, which is where each of the following views are displayed. Our footer div contains a color scheme-matched bar with copyright information to pull the whole page together.

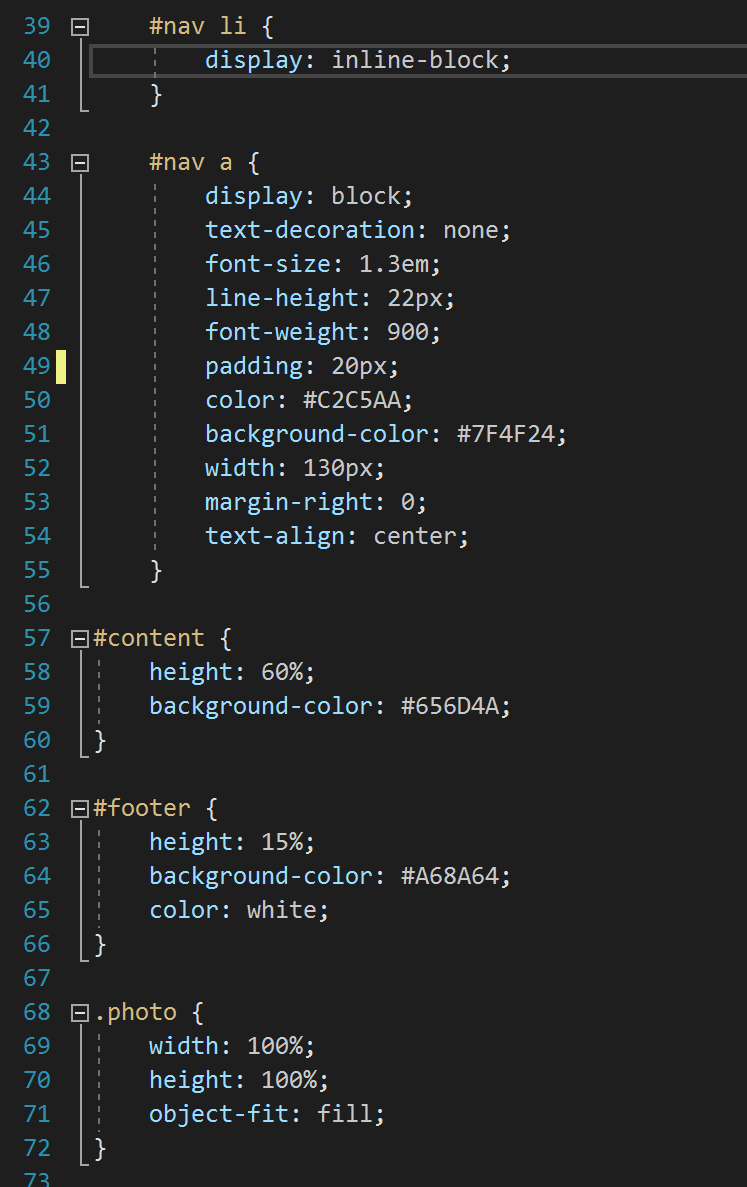
*MasterPage.cshtml*

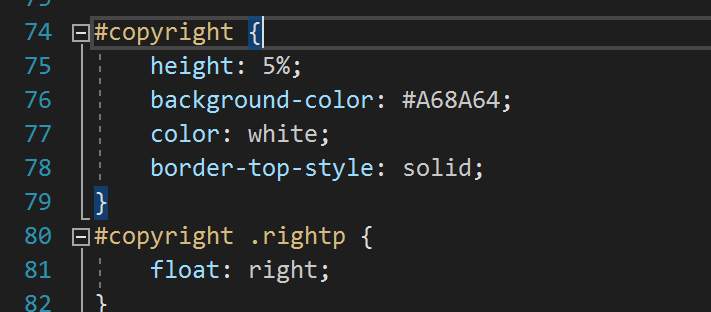




*MasterPage.css*

****

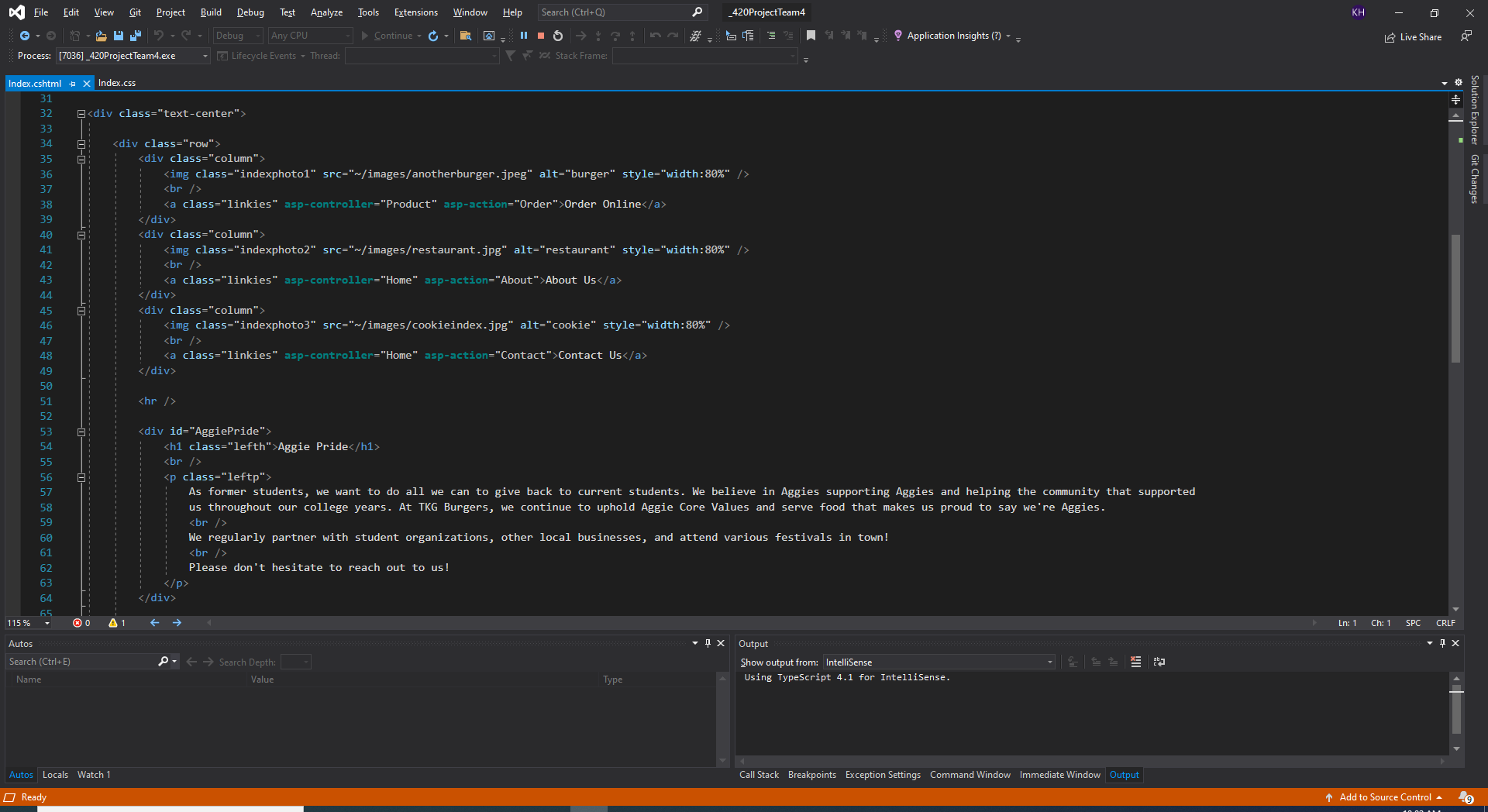
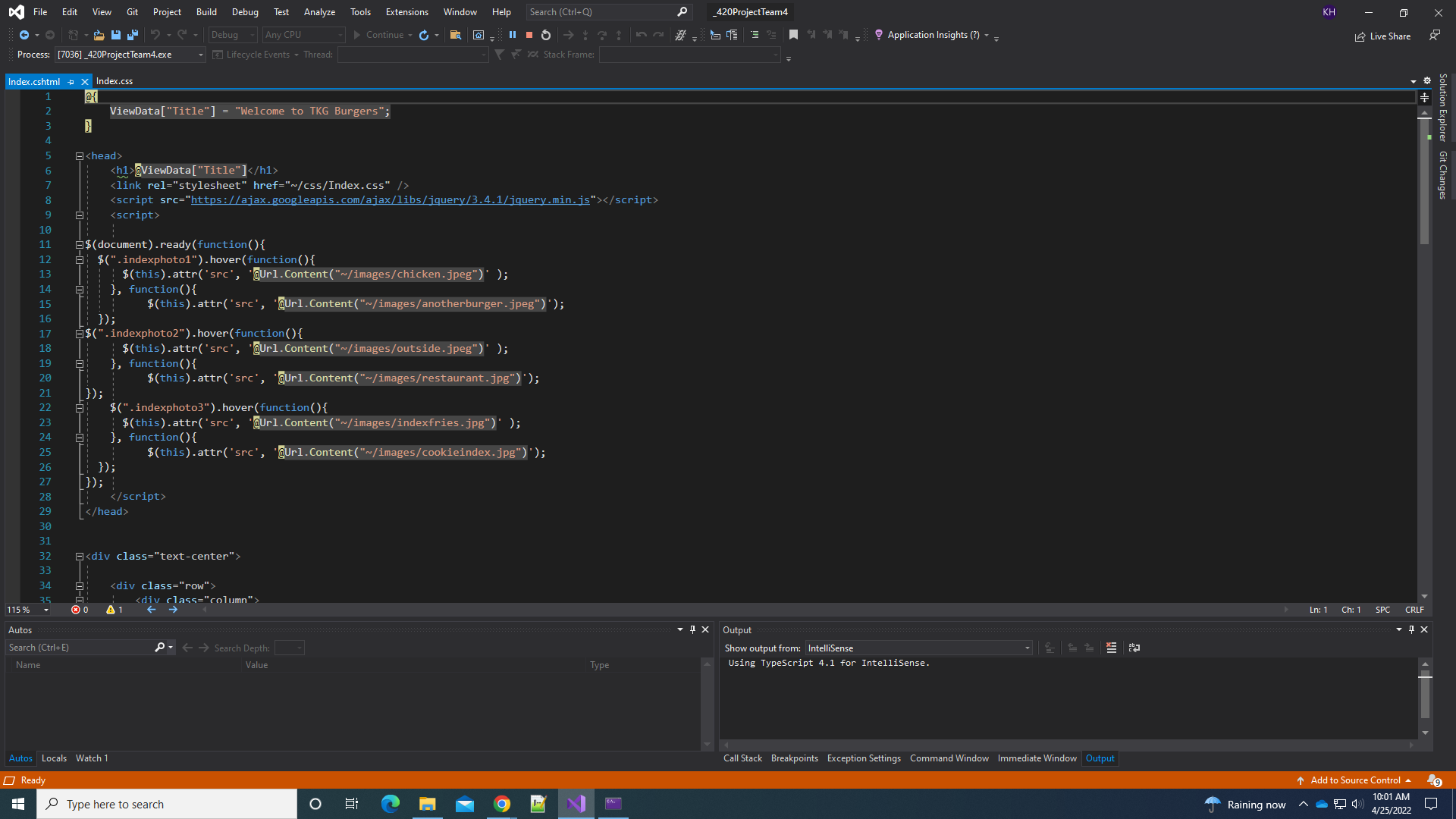
****

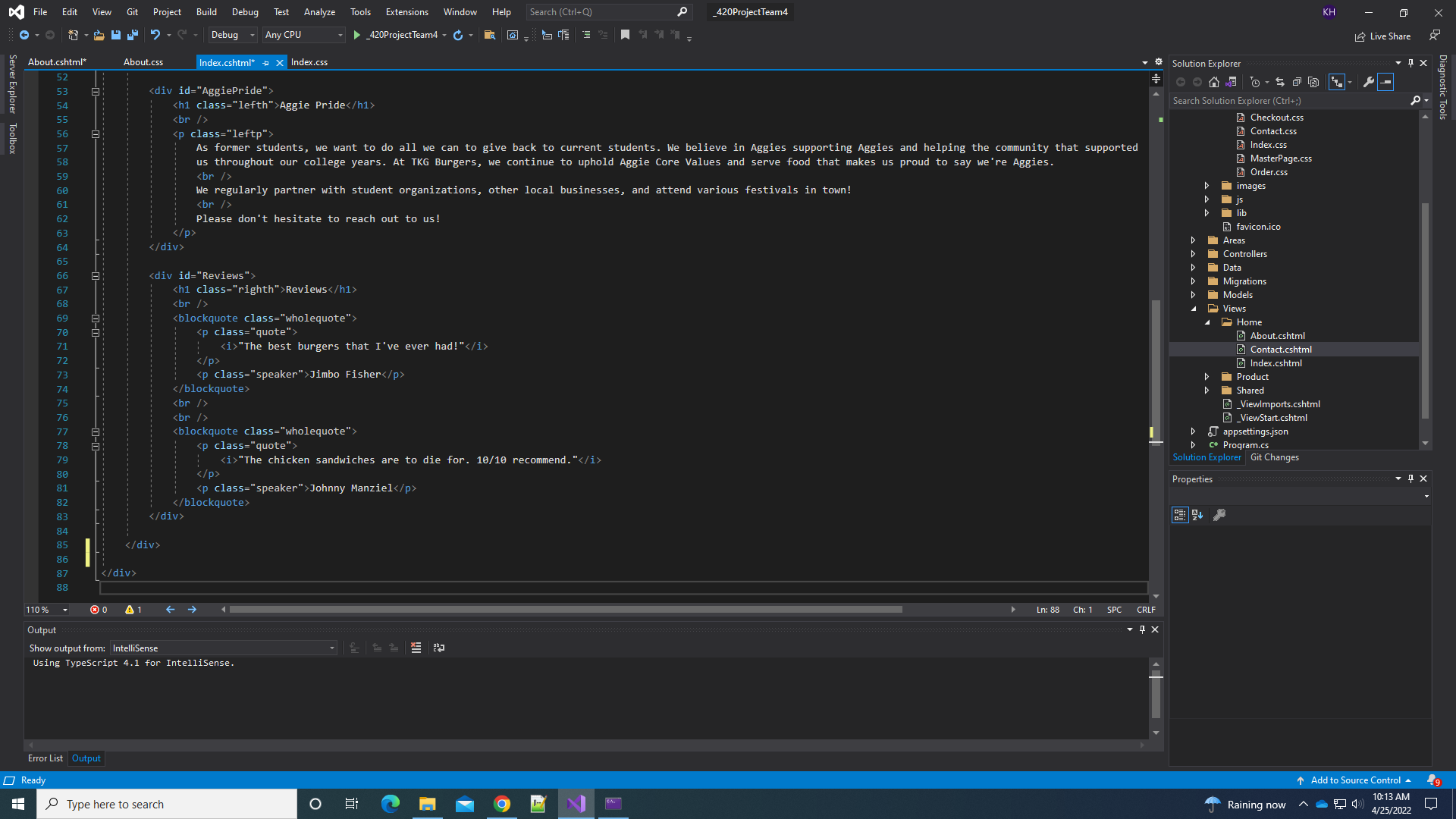
****

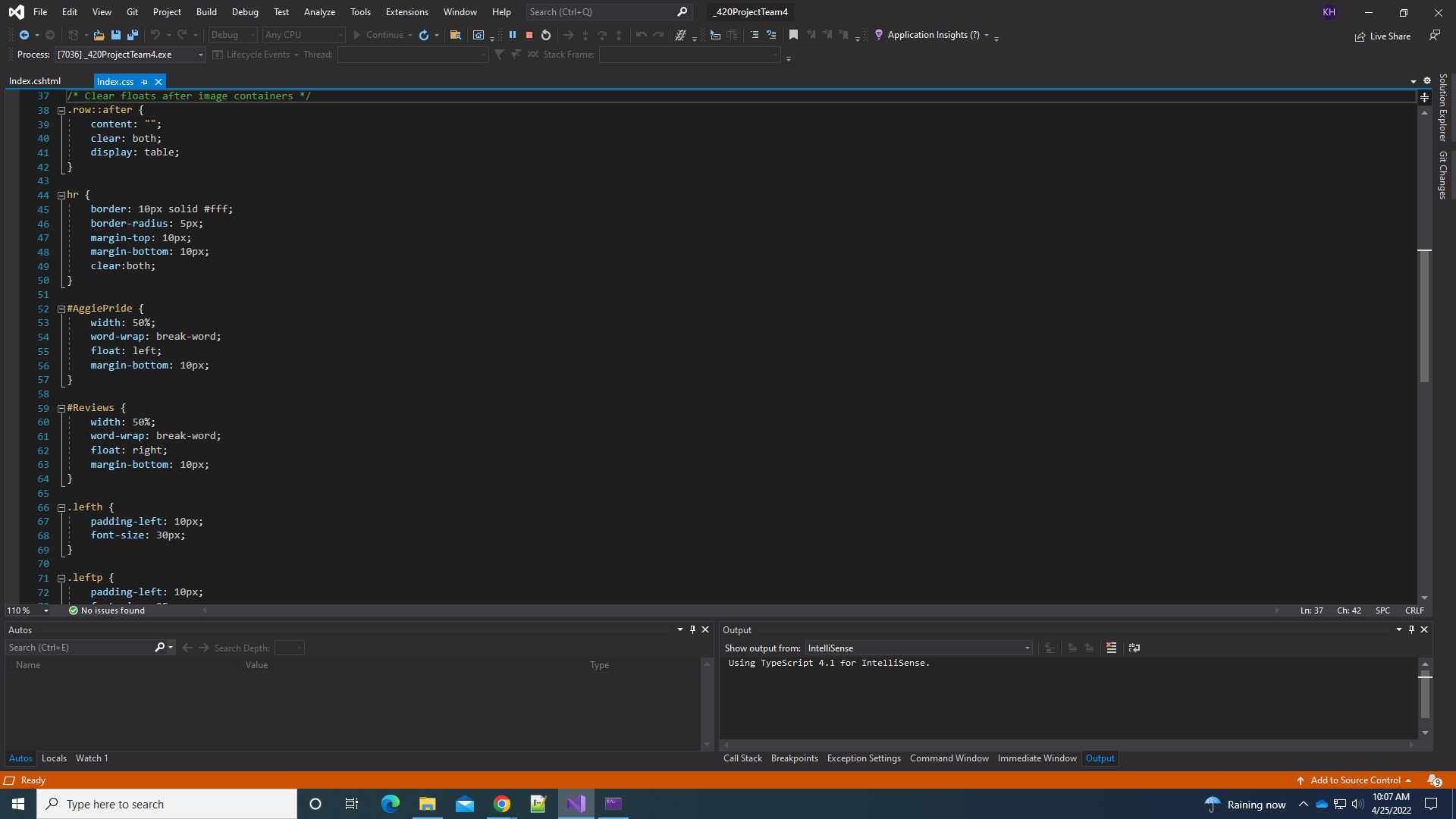
***Index***

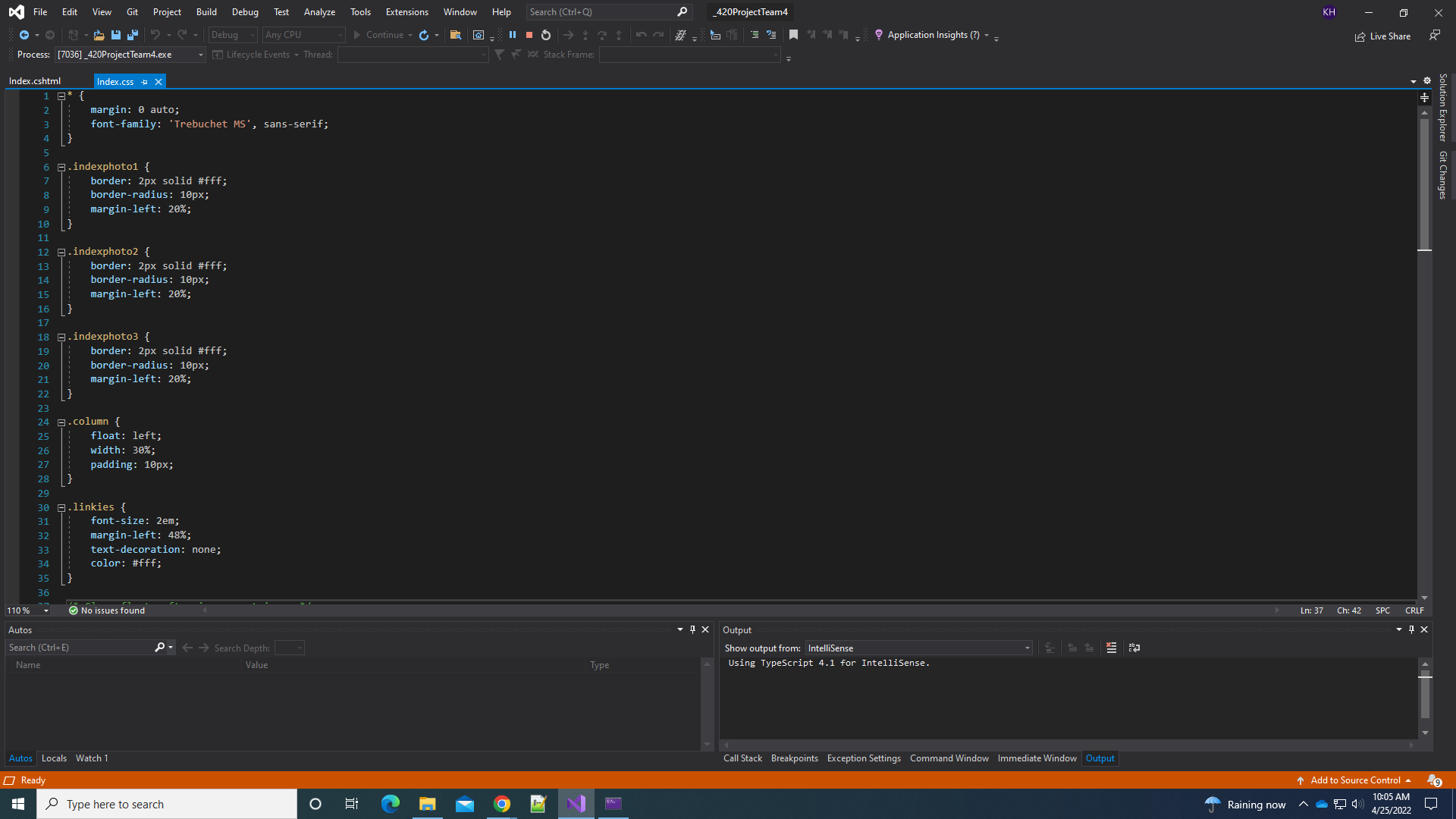
The Index view serves as the Homepage for our web application. It is the page that customers will first see when they launch the website. There are three different divs: the image section with links to other pages, the Aggie Pride section, and the Reviews section. The image section uses jQuery to switch between images when the user hovers over an image. Each image has a border and is spaced using float and padding through CSS. The links on this section go to Order, About Us, and Contact pages. The Aggie Pride and Reviews sections are able to be placed next to each other using float:left and float:right. To prevent the words from running into each other the width and word-wrap had to be changed in CSS. Additionally classes were utilized to change the font-size for both the Aggie Pride and Reviews sections, and to change the color for Reviews section. These allowed the sections to be displayed in a more accessible and aesthetic manner.

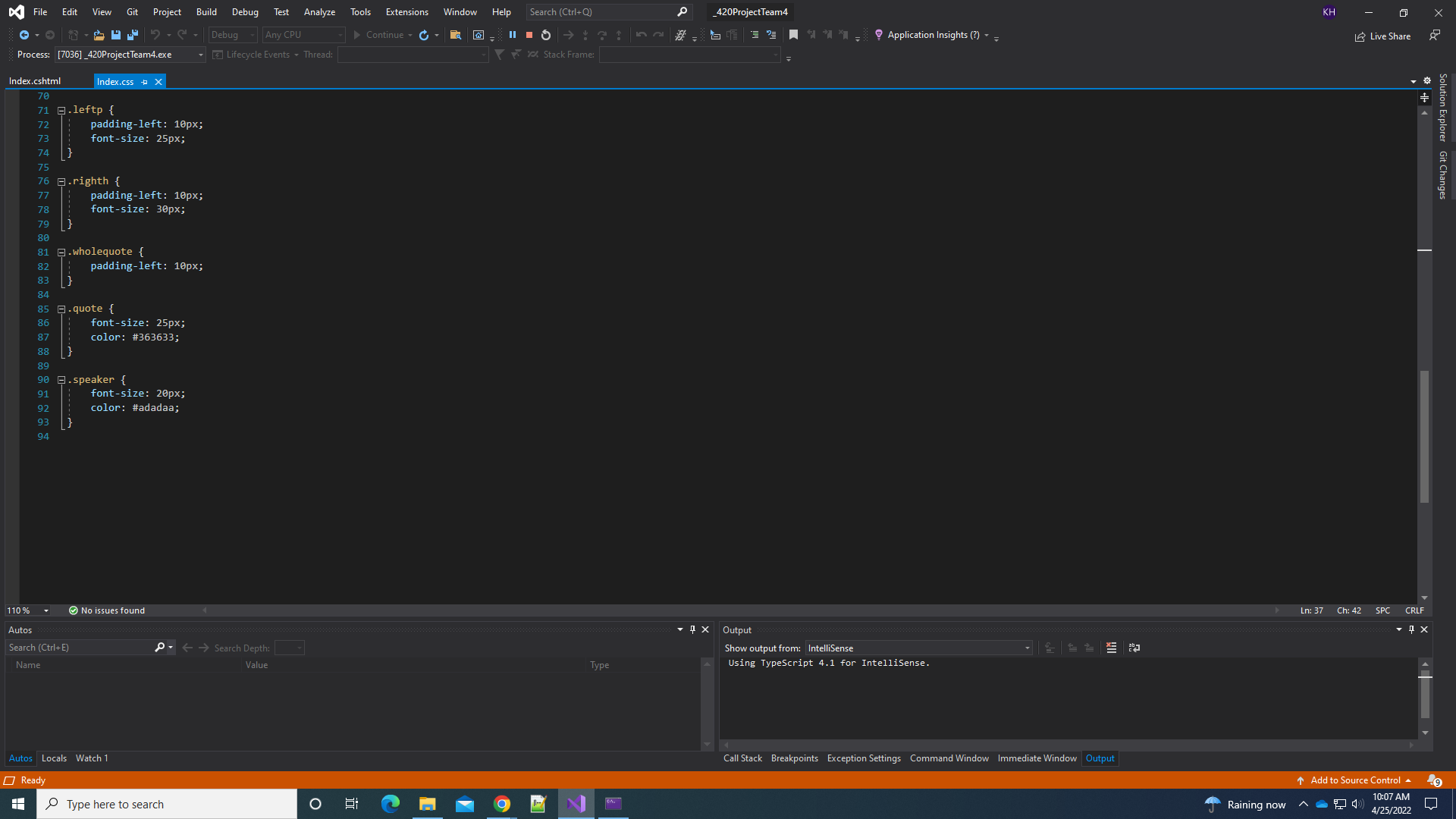
*Index.cshtml*



**

*Index.css*

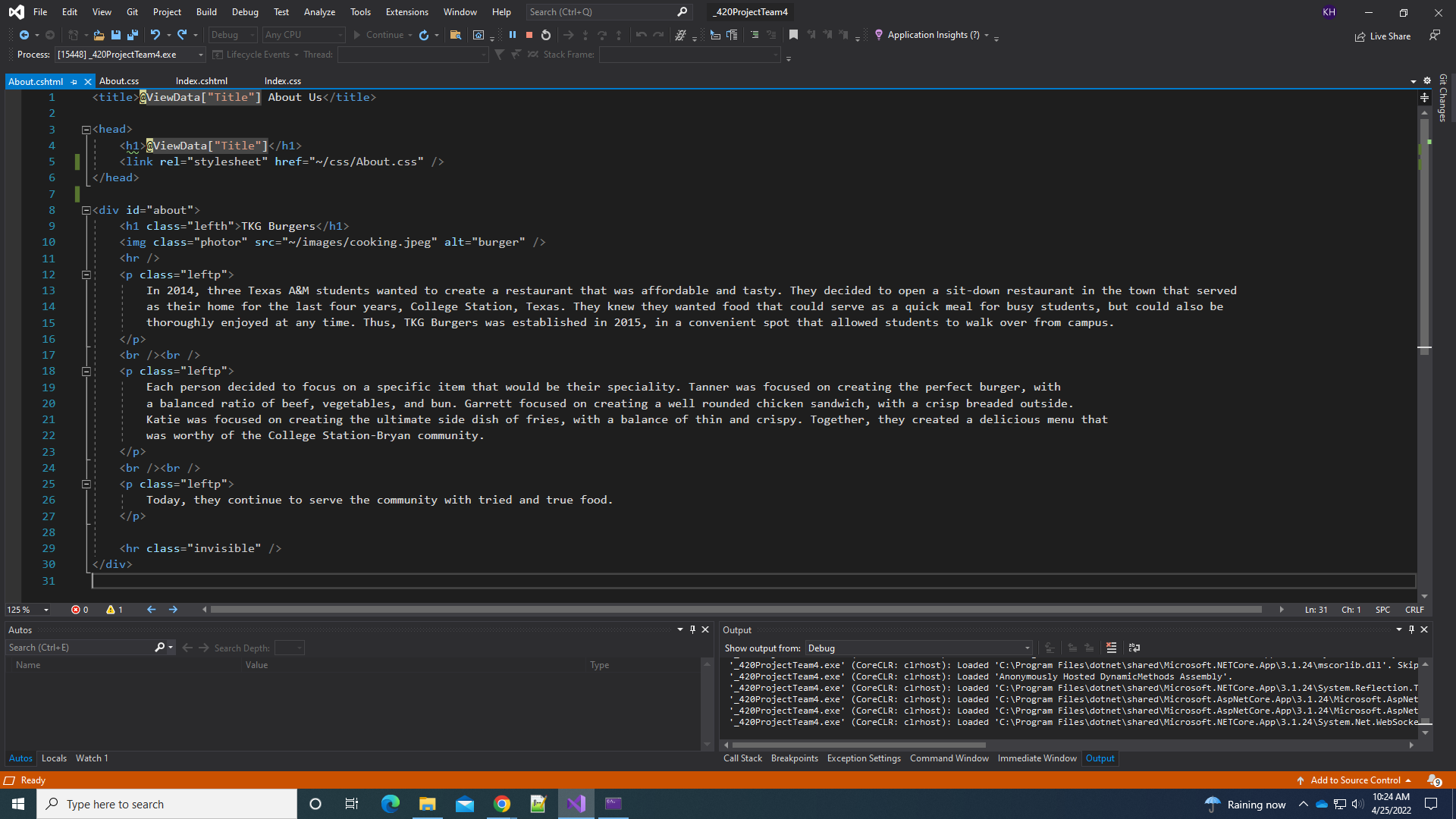




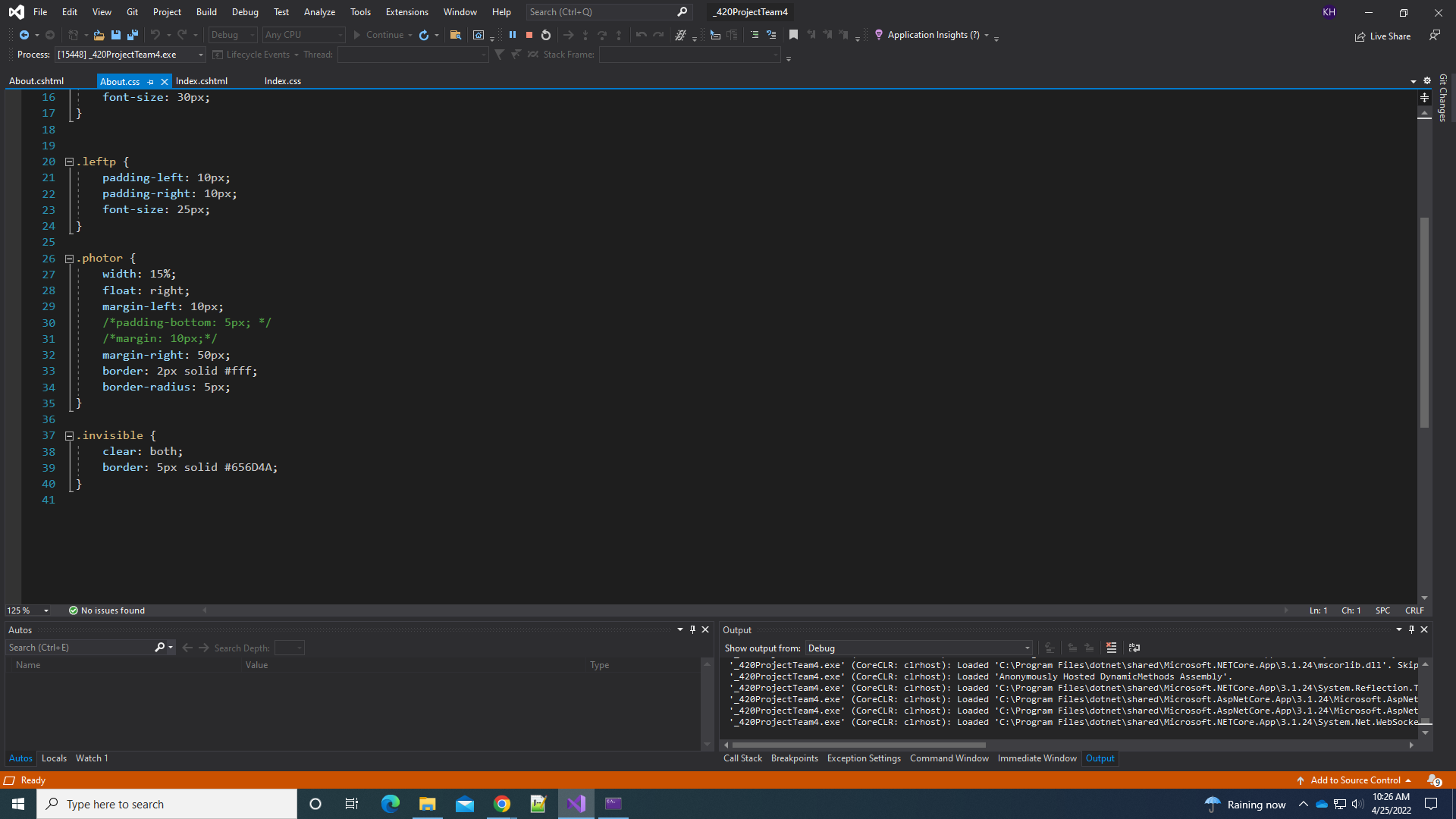
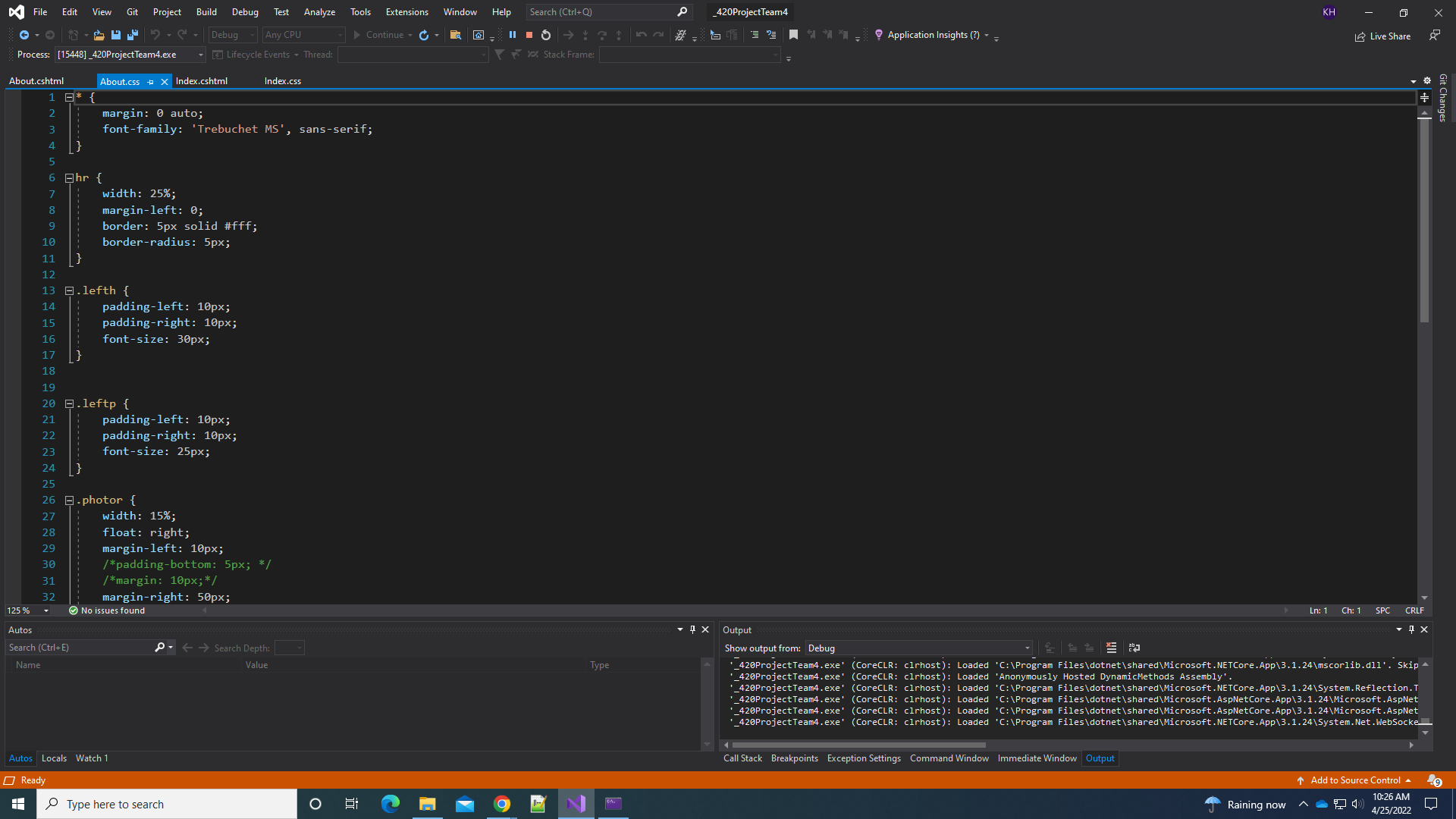
***About***

The About view serves as an About Us page. It allows the customers to better understand the history behind TKG Burgers and who they are now. To create an easy to read section, the html has a heading <h1> and multiple paragraphs <p>. Using CSS, the font-size and padding were adjusted for spacing. An image was added for a visual that gives the customer something to look at besides a block of text. Using CSS, the image was resized, had a border, and was floated to the right. An <hr> tag was also added under the “TKG Burgers” heading to create a line break, allowing for better readability.

*About.cshtml*



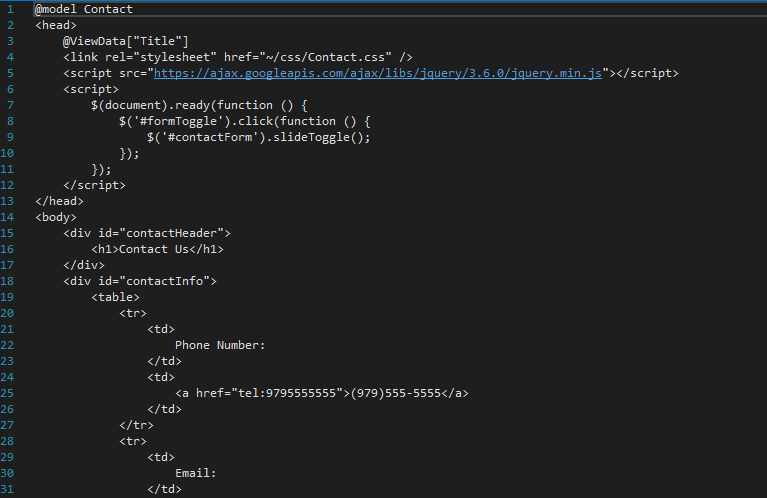
*About.css*

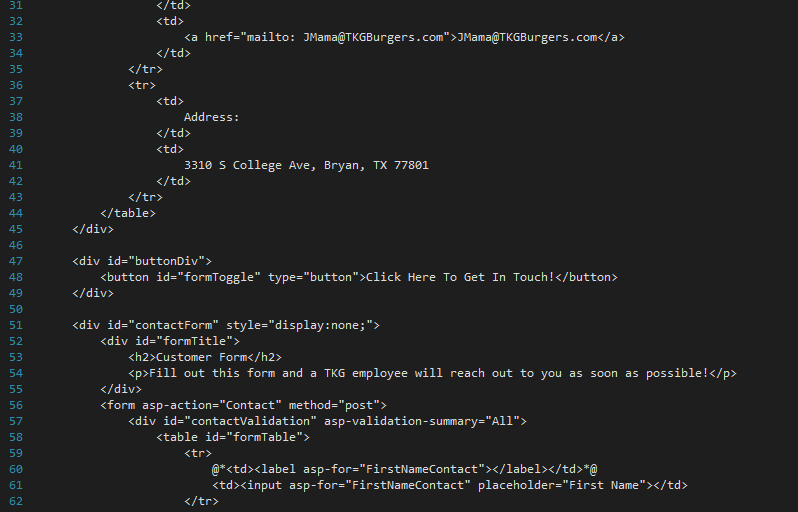


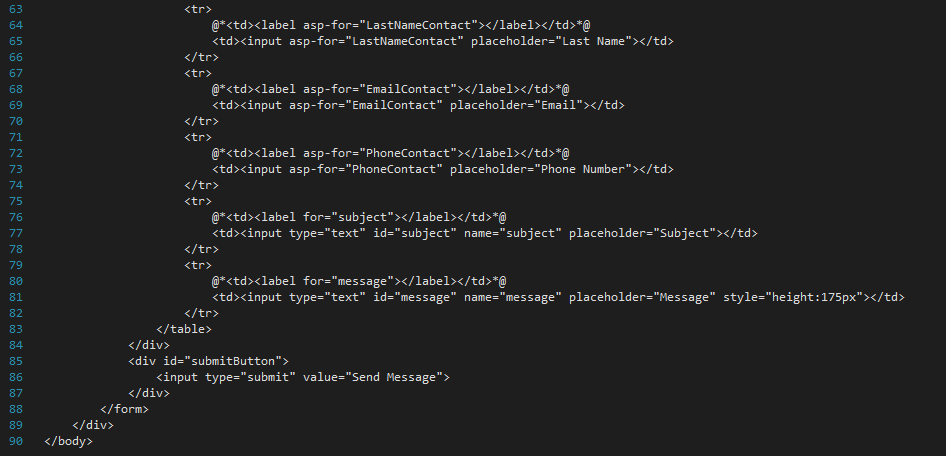
***Contact***

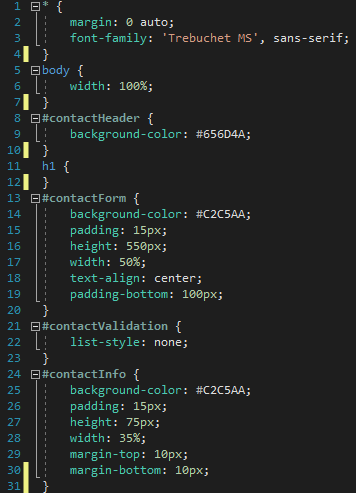
The Contact page gives its contact information (phone, email, and address) and allows for users to contact TKG Burgers through a provided form. The contact information is provided in the div “contactInfo” inside of a table. The phone number and email address are hyperlinked so clicking on them will produce a connection. Then there is the “formToggle” button in the “buttonDiv” which is used to toggle the form between visible and not. In the script div there is code that produces a slide Toggle jquery and in line CSS is used to have the div start in the hidden mode (style=”display:none;”). The jQuery can be toggled back and forth to hide and show the div. The “contactForm” div contains the form for customers to input their data in that the jQuery acts on. This contains first name, last name, phone number, email address, subject, and message fields that customers can populate to give information to TKG. Lastly there is a submit button that submits the form information and resets the page. The form is also validated using the same code as the checkout cshtml page. For CSS the body is split into two main divs, the “contactInfo” and the “contactForm”. The contactInfo table is centered with a 35% width and 10px margin on the top and bottom to allow for spacing from other divs. The contactFrom div has a 50% width and height of 550px because it is the main content of the page for customers to fill out. It also has a 100px padding on the bottom to allow for the validation prompt errors to show up when the user incorrectly inputs the form. The input fields have a border shadow and curved corners (border-radius:4px;) to give a more professional field. They also have inline CSS that gives the placeholder label. All of the buttons have similar CSS with the green color that give a “submit” feel and spacing to separate it from divs. They also have the hover feature to change colors when hovered over.

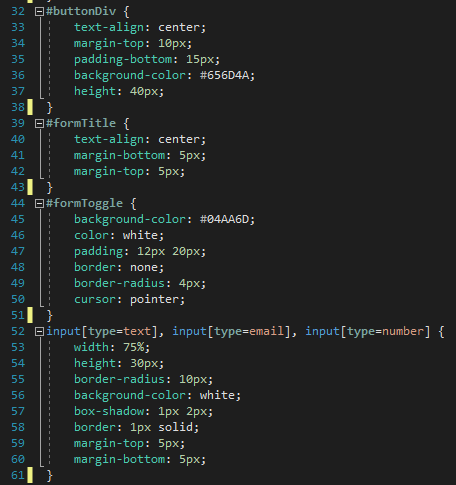
*Contact.cshtml*

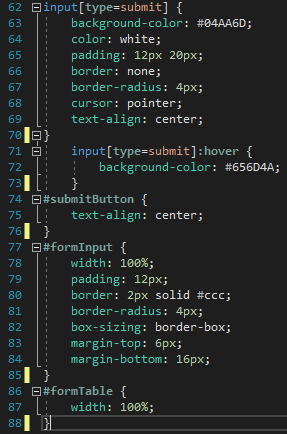
******





*Contact.css*



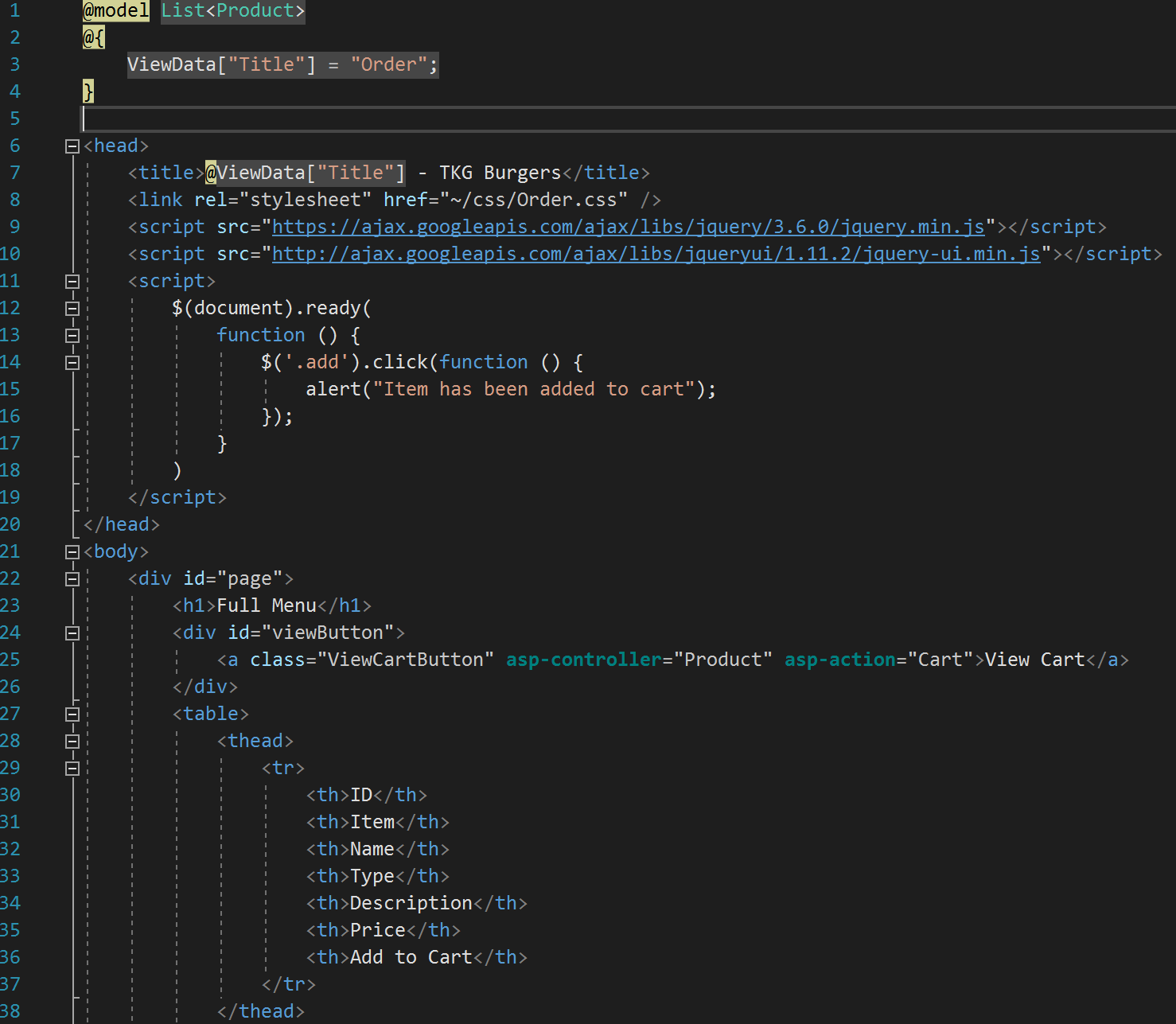


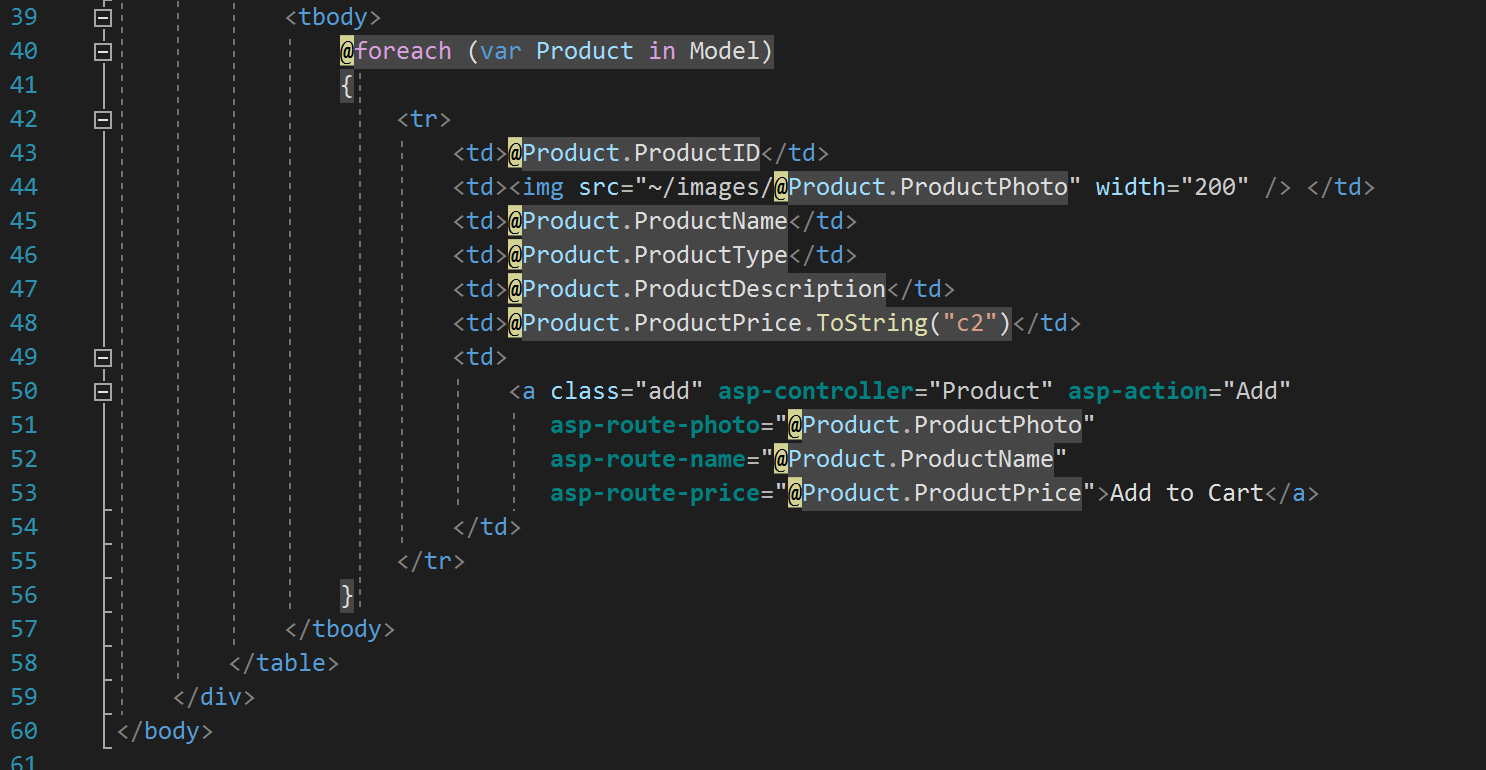
***Order***

The Order view serves as a menu, allowing users to select items to Add to Cart and Order Online. Each item in the Products table of our Product database is added to a table through a Razor “foreach” loop code block, allowing users to see every item available for purchase. Each item contains details such as which of our three types of items the product is (Entree, Side, or Drink), as well as the price and an image, among other details. The entire page is generally contained in the singular Page div, as the item table is self-contained inside its own table tags, and just one additional div for the “View Cart” button is included. When the “Add To Cart” button is clicked for an item, the specified item is passed to the ProductController, where details from said item is used to create a new item in the CartItem table of the Product database. As this is completed, a jQuery statement is executed, throwing an alert onto the page to confirm that the item has been added to the user’s cart.

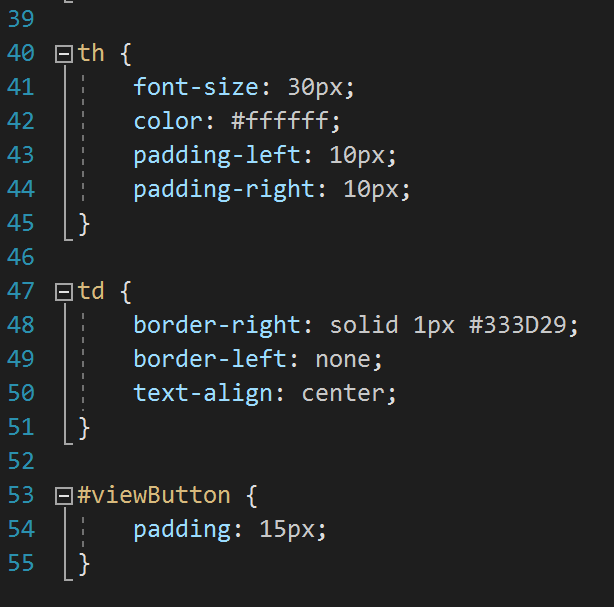
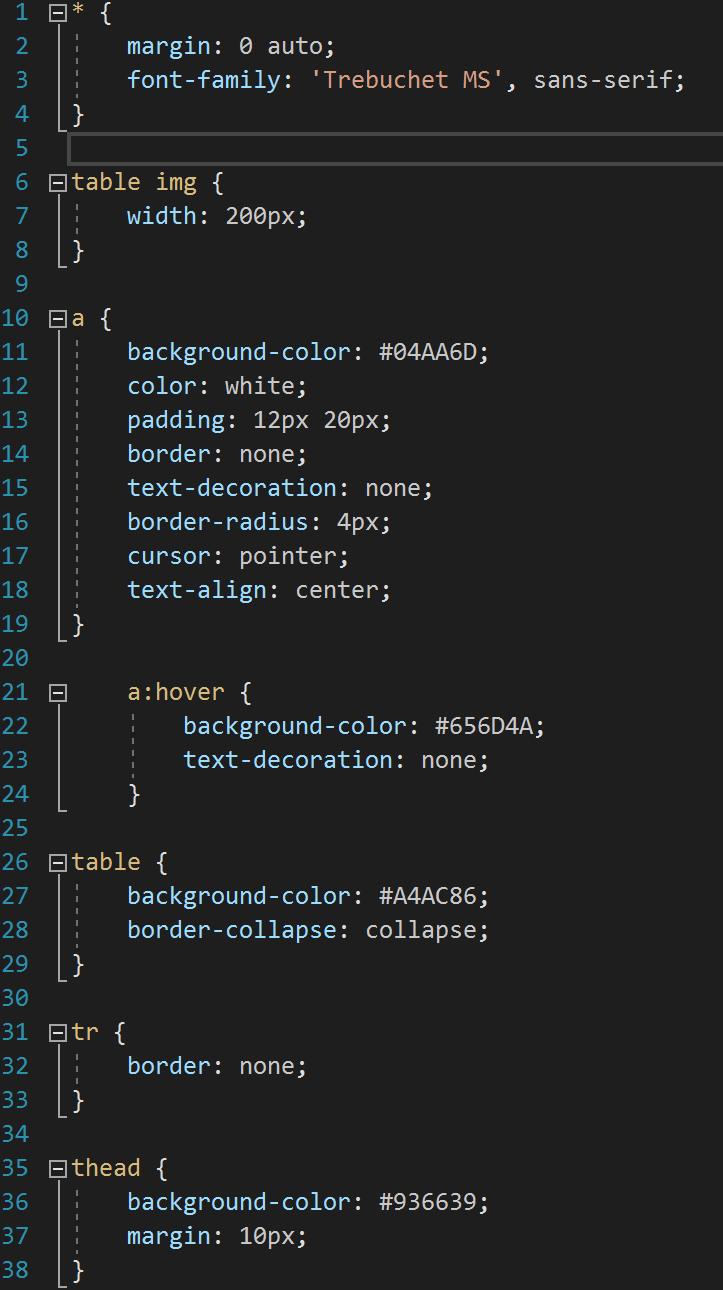
With CSS, the buttons used to interact with the page are designed to match those seen elsewhere on the site, including in Contact, Cart, and Checkout. These buttons change background color when hovered over, and compliment the color scheme of our web app. The Order table is also formatted in CSS to match other tables in the web app, and provide text alignment for each property in the table and image resizing to better fit the table dimensions. In addition, padding was added to each of the buttons in the view, to allow for a more aesthetically-pleasing user experience.

*Order.cshtml*





*Order.css*

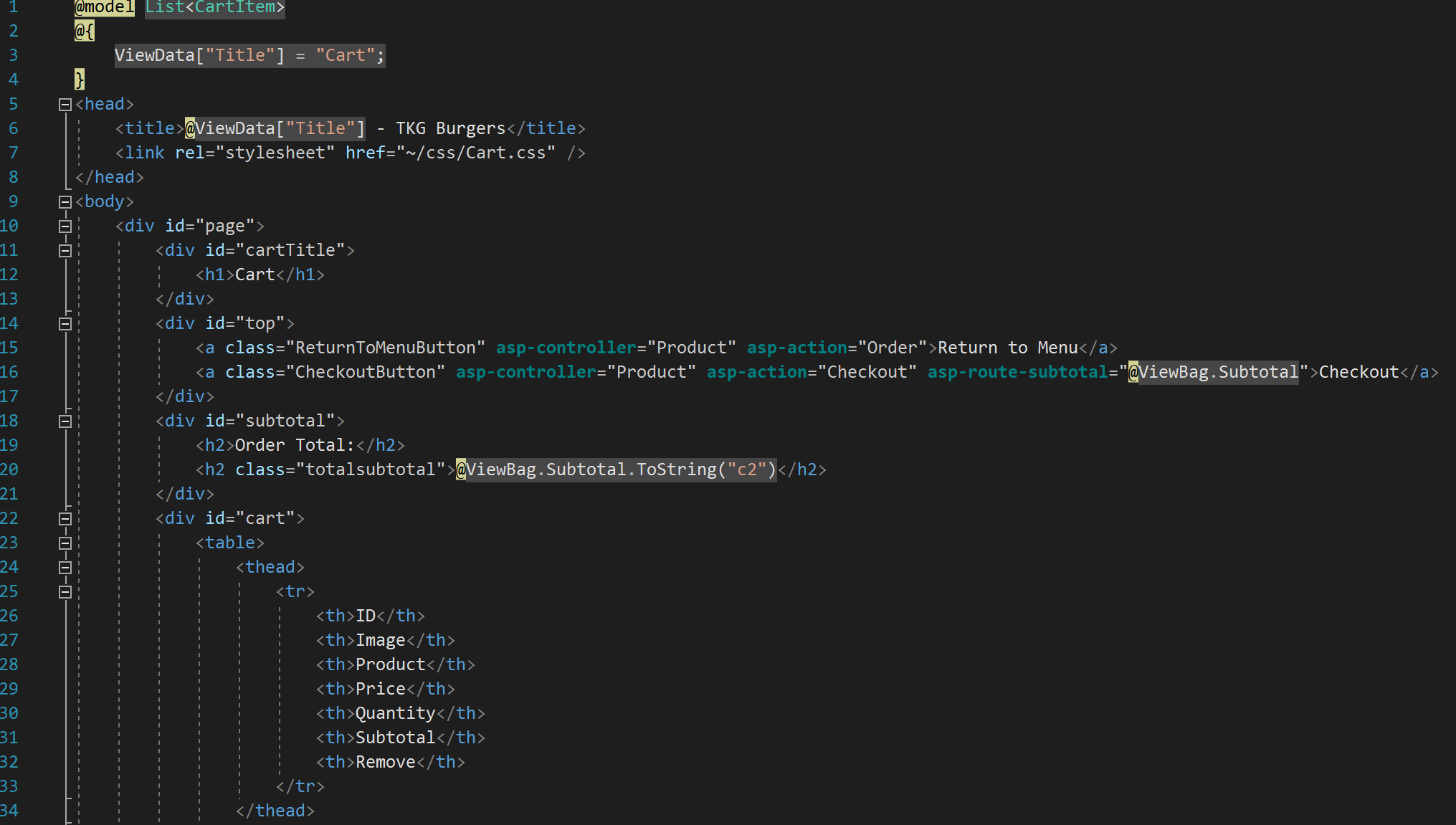


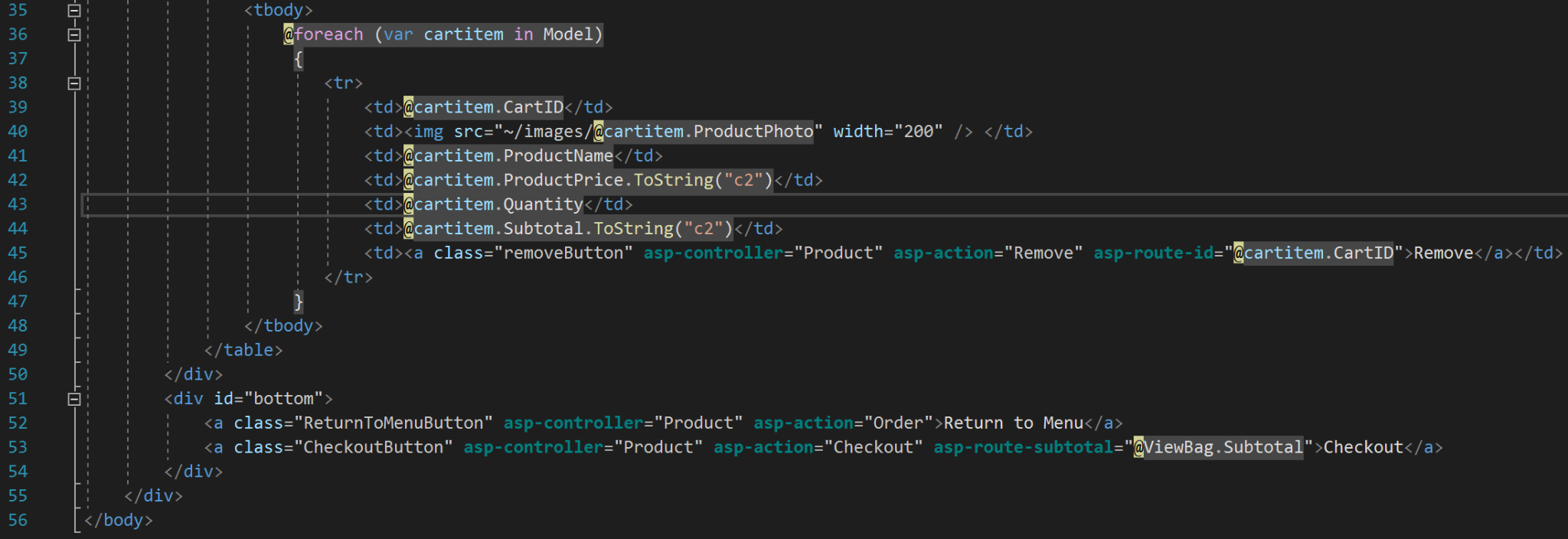
***Cart***

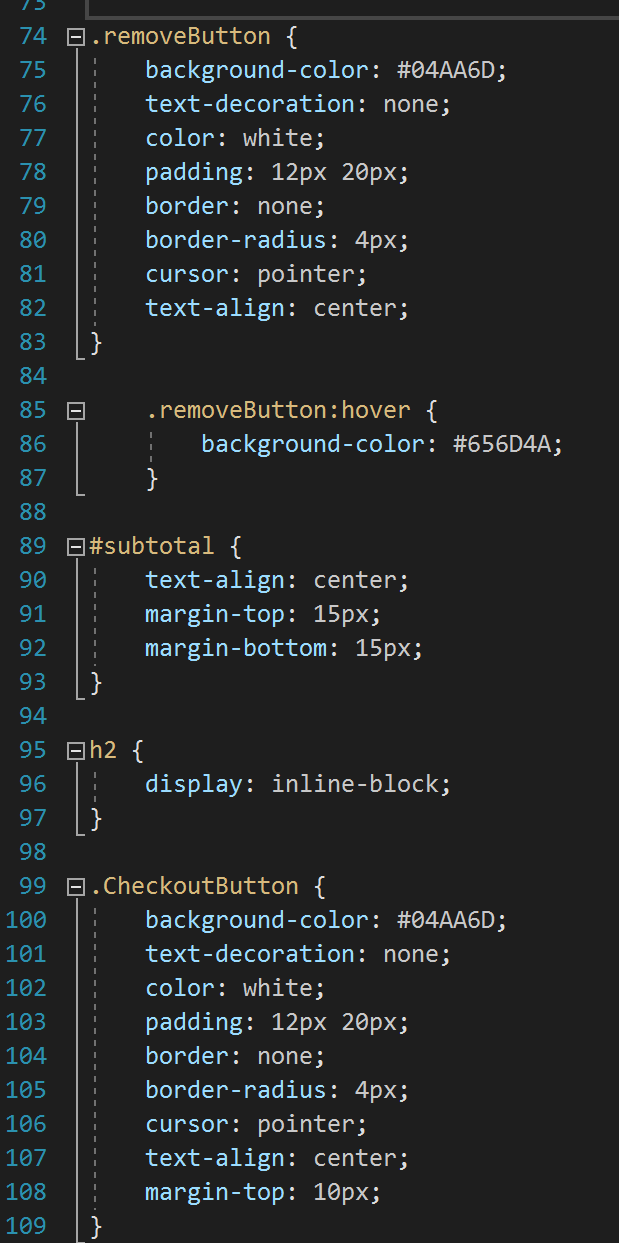
The Cart view of our web app works similarly to our Order view, in that it displays a table of items, in addition to buttons allowing the user to either return to the Order view or proceed to the Checkout view. However, the items displayed in the Cart view (again through a Razor “foreach” loop code block) are only those which the user has added to the cart in the Order view. Items in this table can be removed by clicking the “Remove” button next to the table item. This removes the full quantity of the item. If the user wishes to add to the quantity of an already existing item, they can simply click “Add to Cart” again on the Order view, and the app will add to the quantity of the existing item in the cart.

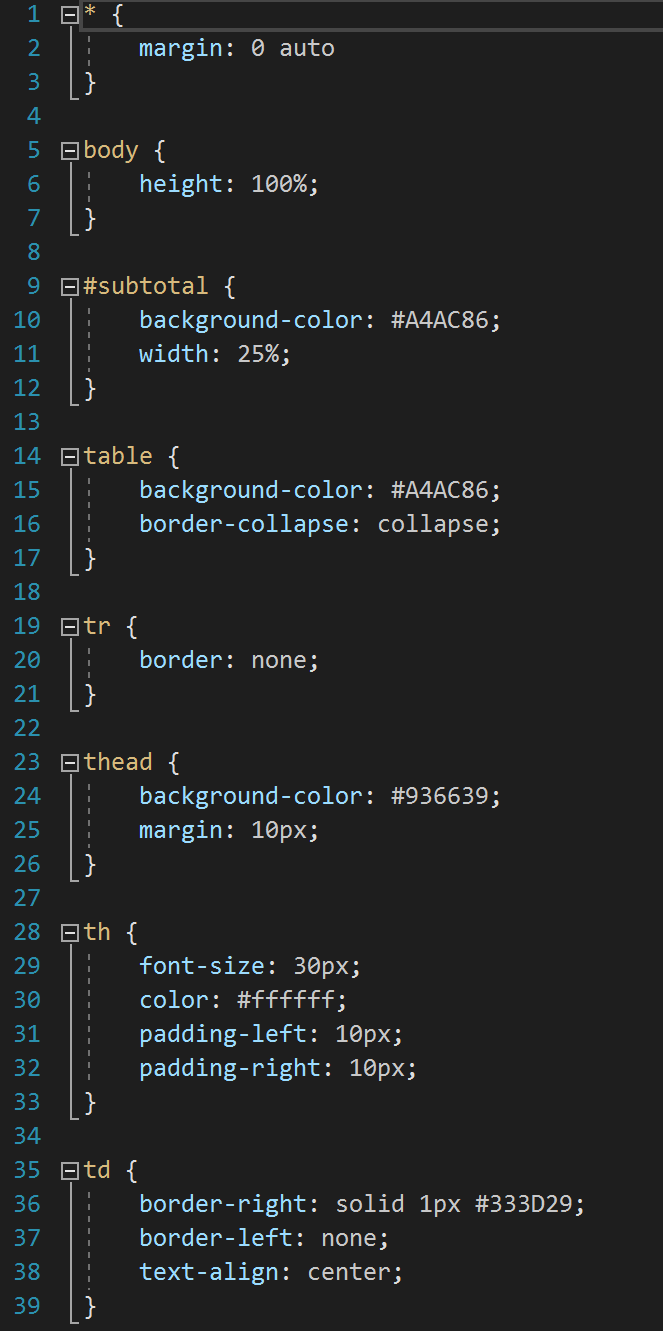
The Cart view contains a Page div that encompassses the entirety of the body, as well as a div for the title of the page, a “top” div that holds the “Return to Menu” and “Checkout” buttons, a “subtotal” div that holds the combined subtotal of items in the cart, a div containing the Cart table, and a “bottom” div that contains the same buttons as the “top” div, solely for user convenience. If the number of items in the cart becomes large, it may be more convenient for users to have these navigation buttons also shown at the bottom of the cart, so they do not have to scroll all the way back up to the top after reviewing their cart. In addition, there are classes for each button, and an additional class to format the text displayed in the subtotal div. The table has been formatted to match the table seen in the Order view, to further provide a clean, consistent look across each view in our web app, complete with button margins & padding, text alignment, and image resizing.

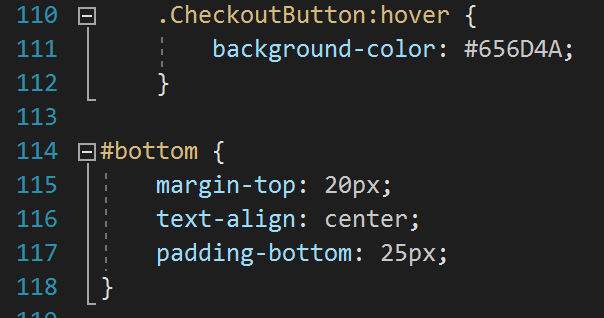
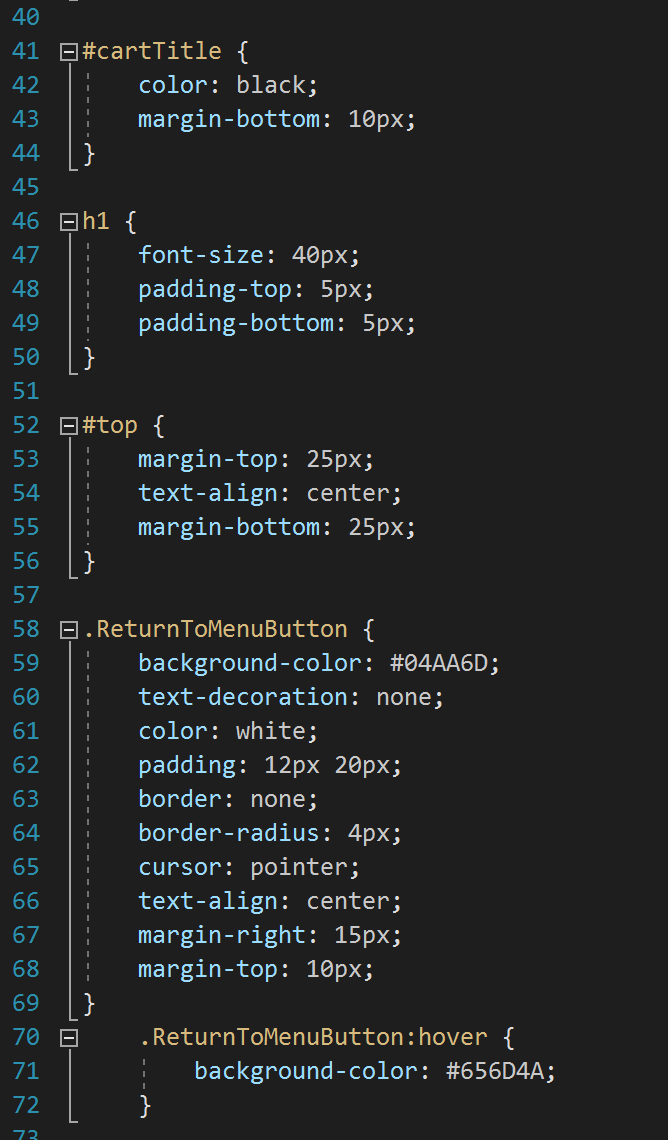
*Cart.cshtml*





*Cart.css*





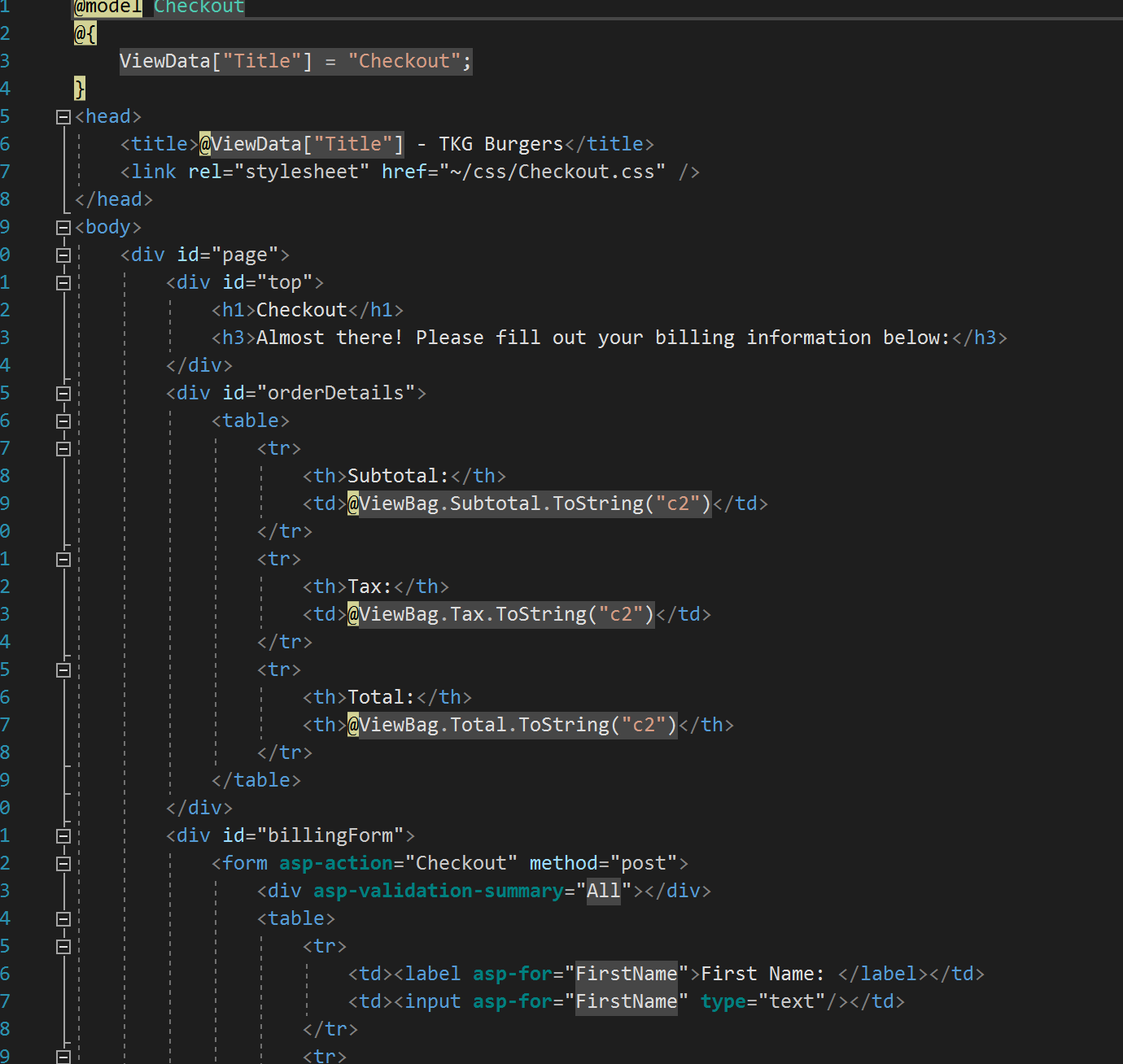
***Checkout***

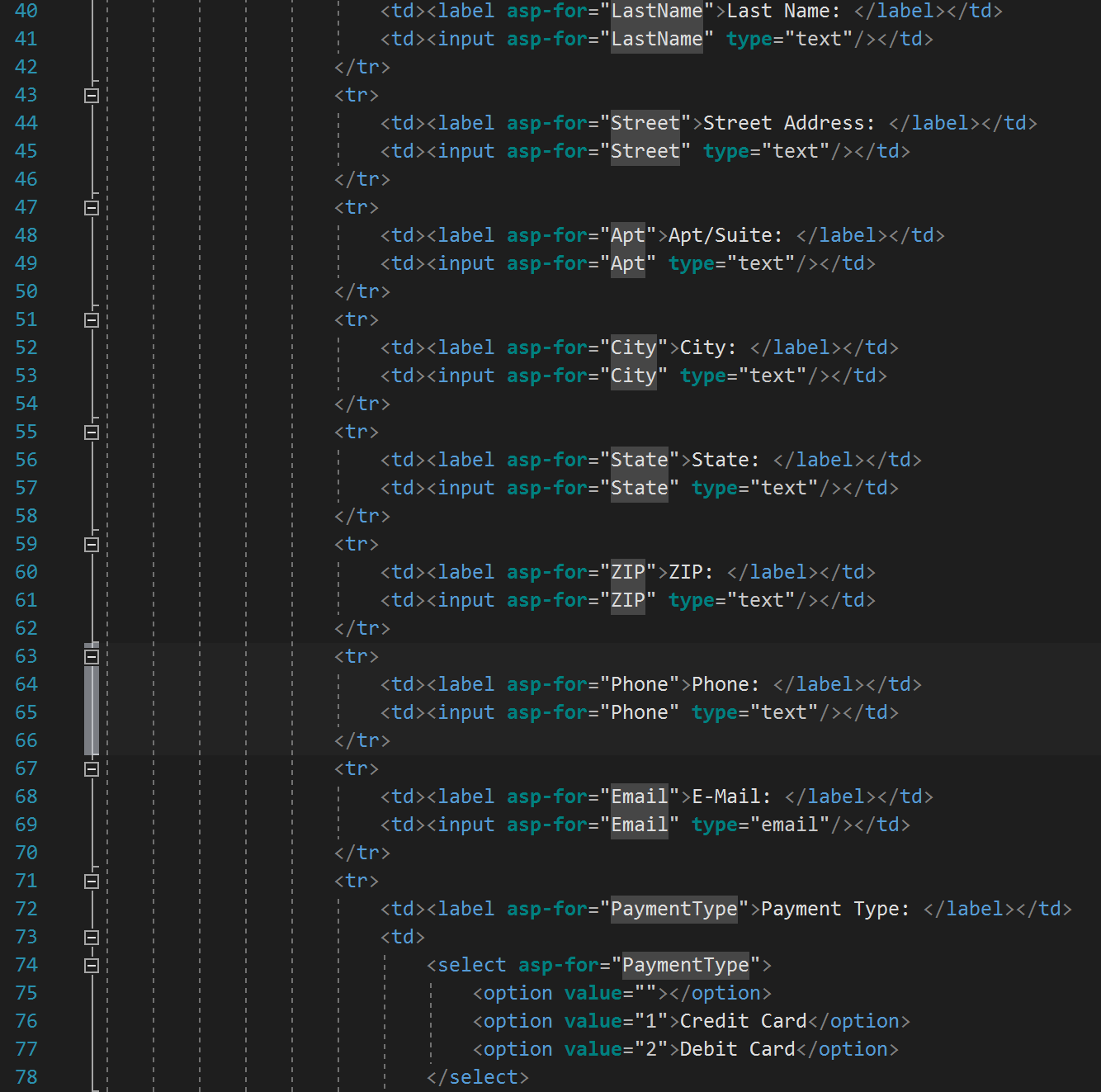
*The Checkout view is our substitute for a customer account on the web app.* Instead of providing a place for the user to create a “customer account,” we decided to make a Checkout view, where a user would enter essentially the same information. These input fields are all validated and will display a helpful message for the user if information is not entered correctly.

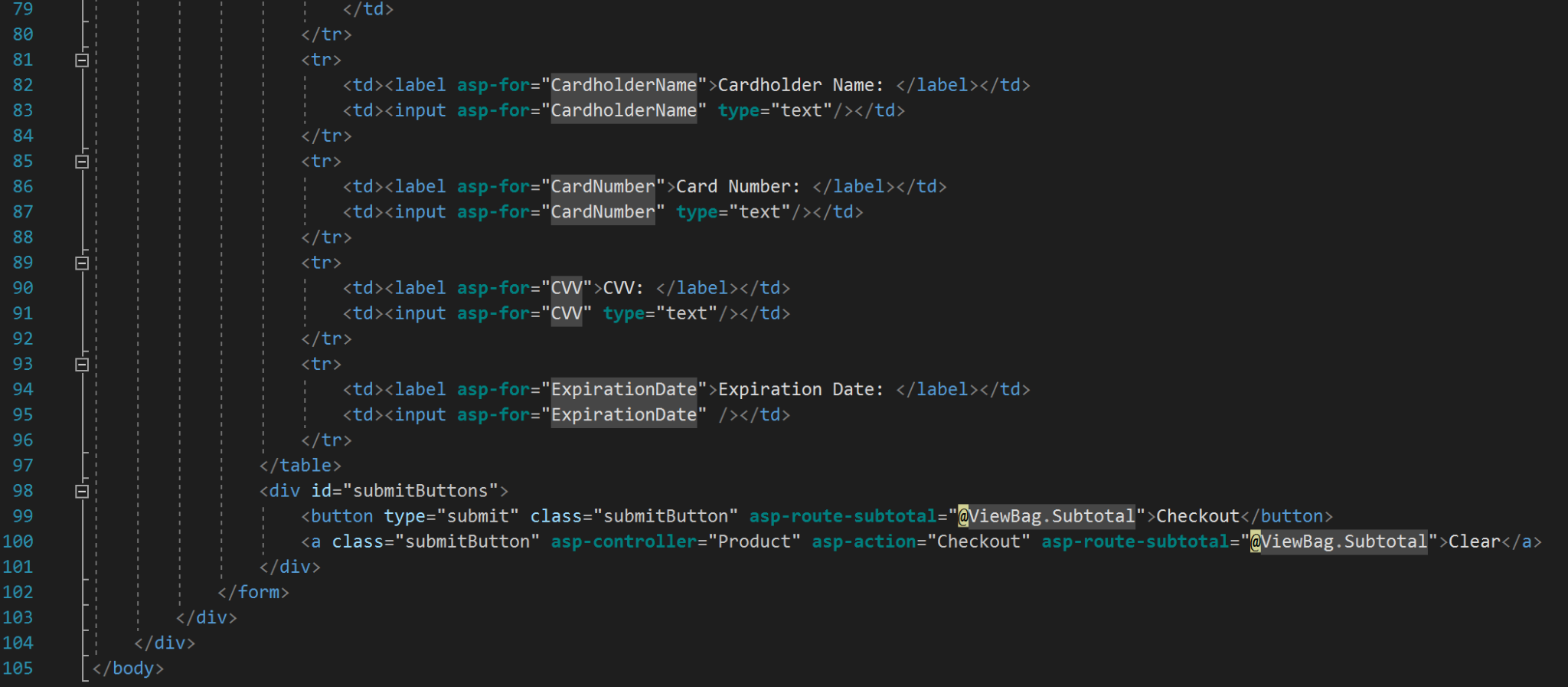
The Checkout view of our app consists of an input form for the user to fill out to complete the purchase of the items currently in their cart. The “totalSubtotal” value calculated in the Cart view is passed to the Checkout view, where the user can see the subtotal for the entire order, the tax added, and the grand total for their order. This information is all displayed in a table above the form. Below the form, there are two buttons: one to complete the checkout process, and one to clear the form of all input to start over. If the input fields are all properly validated, the user is returned to the Index view of the web app. Otherwise, the app returns the checkout view with helpful messages to inform the user what fields need to be edited.

The Checkout view has a “page” div, which encompasses the whole body, a “top” div, which contains the page title and an encouraging message, an “orderDetails” div containing the subtotal, tax, and total, a “billingForm” div that holds the form, and a “submitButtons” div containing the “Checkout” and “Clear” buttons. Each div is formatted in CSS to provide appropriate margin and padding, as well as the same color scheme used across each view in our app. In addition, the buttons are all formatted to match their appearance on other views in the app. For the form, we used borders and box shadows to give each input field a rounded, more playful look that compliments our general theme. These also match the input fields seen in the Contact view of our web app.

*Checkout.cshtml*







*Checkout.css*

