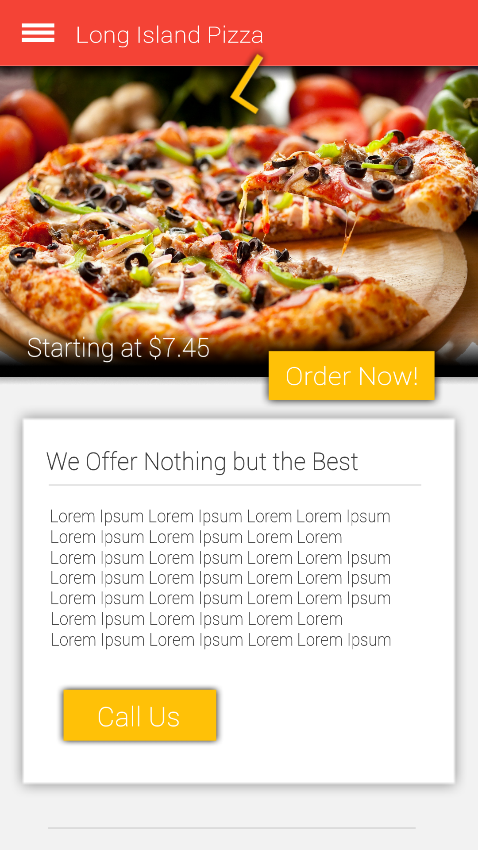
Project Documentation

UI Mockups

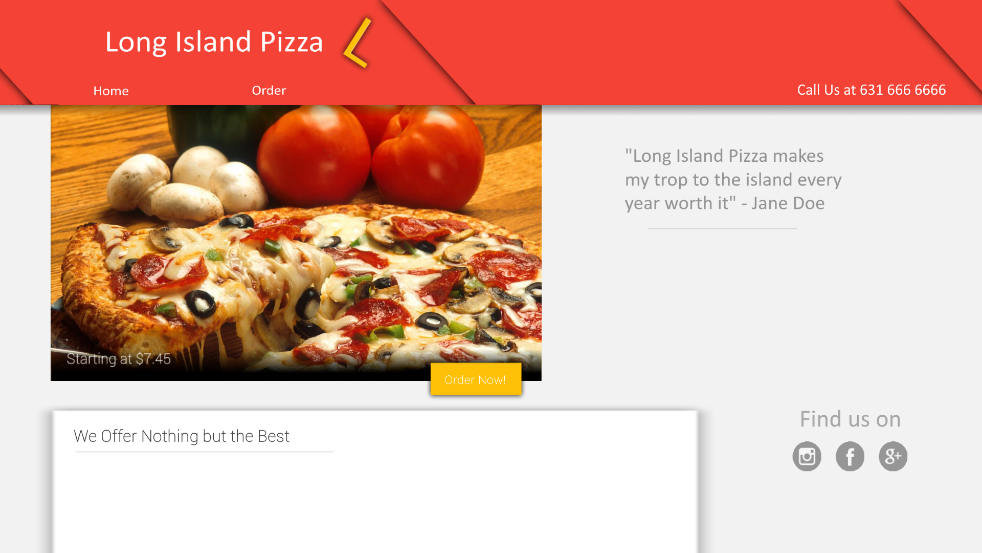
Application Link: http://www.li-pizza.azurewebsites.net

As I first began the project I decided to mockup my interfaces starting from my mobile UI to my desktop UI. I incorporated Material Design into my UI with colors chose from a Material Design color pallet found online.

Mobile



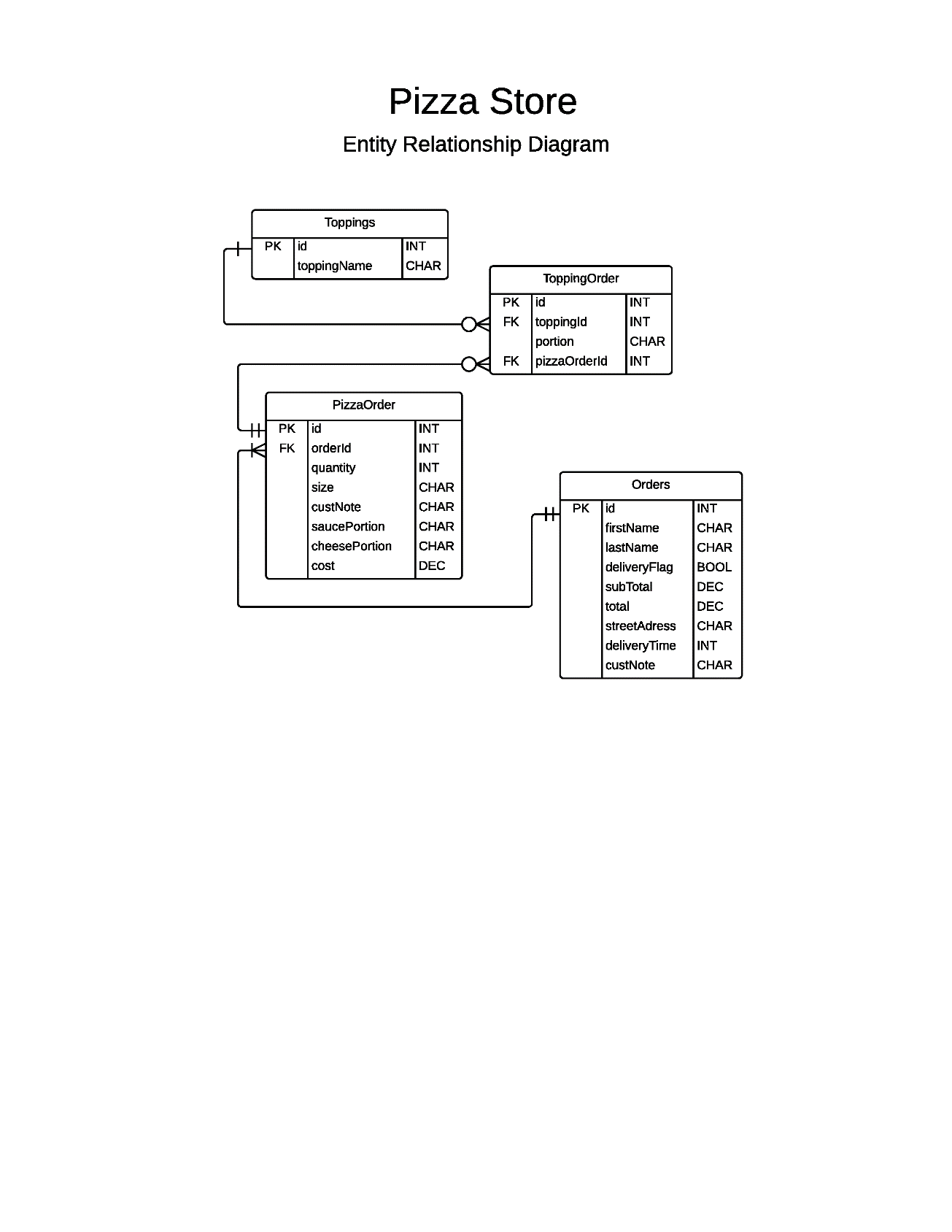
Desktop



Data Structures

I decided that I would first design a SQL based relational database and follow up by designing a noSQL MongoDB based one in case I found connecting to MariaDB too cumbersome. This turned out to be true and I proceeded to educate myself on Asp.Net and its use of the Mongo Driver for basic CRUD operations.

SQL Based Database



NOSQL Based Database

C - String

C - ObjectId

C - Boolean

C - Integer

C - Double

/ - denotes limited options

Orders -

{

\_id: “Objectid”,

name: {

firstName: “Jane”,

lastName: “Doe”

},

delivery: “true/false”,

address: {

streetAddress: “3 William St.”,

city: “Bay Shore”,

state: “NY”,

zipCode: “11706”

},

complTime: “30”,

delivNote: “leave by doorstep”,

pizzas: [

{

\_id: “Objectid”,

order\_id: “Objectid”,

size: “Small/Medium/Large”,

quantity: “4”,

toppings: [

{

\_id: “Objectid”,

pizza\_id: “Objectid”,

topping\_id: “Objectid”,

portion: “Light/Normal/Extra”,

},

{

}…

],

cost: “14.32”,

pizzaNote: “make heart shape out of pepperoni slices”

},

{

}…

],

subTotal: “14.32”,

total: “14.32”

}

Toppings -

{

\_id: “Objectid”,

name: “Pepperoni”,

quantUsed: “4”,

quantCeiling: “20”

}

Application Models

AddPizzaViewModel

Model sent out and manipulated by the AddPizza view. Includes the size of the pizza, the quantity of that pizza, a list for the portions for each topping, a list of each available topping, and a list of toppings that were chosen for said pizza.

public class AddPizzaViewModel

{

public string size { get; set; }

public int quantity { get; set; }

public List<string> toppingPortion { get; set; }

public List<string> toppingName { get; set; }

public List<bool> ifChecked { get; set; }

public AddPizzaViewModel()

{

toppingPortion = new List<string>();

toppingName = new List<string>();

ifChecked = new List<bool>();

}

}

CheckOutViewModel

Model sent out and manipulated by the CheckOut view. Includes the customer’s first name, last name, a bool detailing whether an order is to be delivered or not, the customer’s street address, city, state, and zip code.

public class CheckOutViewModel

{

public string firstName { get; set; }

public string lastName { get; set; }

public bool delivery { get; set; }

public string streetAddress { get; set; }

public string city { get; set; }

public string state { get; set; }

public int zipCode { get; set; }

public CheckOutViewModel()

{

delivery = new bool();

}

}

DBConnection

Model that physically connects to the database and establishes a means to run CRUD operations to the toppings and orders collections. Model is instantiated in the controller as readonly for convenience. Sanitation function for forms is also present.

public class DBConnection

{

public String connectionString = "mongodb://sl249:1409papi@candidate.53.mongolayer.com:10328,candidate.54.mongolayer.com:10059/pizzaDB?replicaSet=set-5611b747922f00a21800151e";

public String DataBaseName = "pizzaDB";

public MongoDatabase Database;

public DBConnection()

{

var client = new MongoClient(connectionString);

var server = client.GetServer();

Database = server.GetDatabase(DataBaseName);

}

public MongoCollection<Order> Orders

{

get

{

var Orders = Database.GetCollection<Order>("Orders");

return Orders;

}

}

public MongoCollection<Topping> Toppings

{

get

{

var Toppings = Database.GetCollection<Topping>("Toppings");

return Toppings;

}

}

public string StripTagsCharArray(string source)

{//from-http://www.dotnetperls.com/remove-html-tags

char[] array = new char[source.Length];

int arrayIndex = 0;

bool inside = false;

for (int i = 0; i < source.Length; i++)

{

char let = source[i];

if (let == '<')

{

inside = true;

continue;

}

if (let == '>')

{

inside = false;

continue;

}

if (!inside)

{

array[arrayIndex] = let;

arrayIndex++;

}

}

return new string(array, 0, arrayIndex);

}

}

Order

Model depicts the order collection with a list of pizzaOrders. The class includes attributes for an identifier, first name, last name, delivery trigger, street address, city, state, zip code, time for completion, subtotal, sales tax, and total. Functions for setting the order total and time for completion are also present.

public class Order

{

[BsonId]

[Required]

public ObjectId \_id { get; set; }

[Required]

public string firstName { get; set; }

[Required]

public string lastName { get; set; }

[Required]

[BsonRepresentation(BsonType.Boolean)]

public bool delivery { get; set; }

[BsonIgnoreIfNull]

public string streetAddress { get; set; }

[BsonIgnoreIfNull]

public string city { get; set; }

[BsonIgnoreIfNull]

public string state { get; set; }

[BsonRepresentation(BsonType.Int32)]

[BsonIgnoreIfNull]

public int? zipCode { get; set; }

[Required]

[BsonRepresentation(BsonType.Int32)]

public int complTime { get; set; }

[Required]

[BsonElementAttribute("pizzas")]//child document obj

public IList<PizzaOrder> pizzaOrder { get; set; }

[Required]

[BsonRepresentation(BsonType.Double)]

public double subTotal { get; set; }

[Required]

[BsonRepresentation(BsonType.Double)]

public double salesTax { get; set; }

[Required]

[BsonRepresentation(BsonType.Double)]

public double total { get; set; }

public Order()

{

pizzaOrder = new List<PizzaOrder>();

this.\_id = ObjectId.GenerateNewId();

}

public void setOrderTotal()

{//sets order total

double subTotal = 0;

double salesTax = 0;

foreach (var a in pizzaOrder)

{

subTotal = subTotal + a.cost;

}

this.subTotal = Math.Round(subTotal, 2, MidpointRounding.AwayFromZero);

if (this.delivery == true)

{//if delivery + $1.00

subTotal = subTotal + 1;

}

salesTax = Math.Round((subTotal \* .04), 2, MidpointRounding.AwayFromZero);

this.salesTax = salesTax;

this.total = Math.Round((subTotal + salesTax), 2, MidpointRounding.AwayFromZero);

}

public void setOrderTime()

{//sets time for completion

int numPizzas = 0;

int time = 0;

foreach (var a in pizzaOrder)

{

numPizzas = numPizzas + a.quantity;

}

time = numPizzas \* 10;

if(delivery == true)

{//extra 20 minutes for delivery

time = time + 20;

}

this.complTime = time;

}

}

PizzaOrder

Model depicts an inner array in the order collection that contains each individual pizza order. A list of this type of object is present in the order model. The model contains attributes for an identifier, order identifier, size, quantity, list of topping orders, and cost. A function to calculate the price of the individual pizza order is also present.

public class PizzaOrder

{

[BsonId][Required]

public ObjectId \_id { get; set; }

[Required]

public ObjectId order\_id { get; set; }

[Required]

public string size { get; set; }

[Required][BsonRepresentation(BsonType.Int32)]

public int quantity { get; set; }

[BsonElementAttribute("toppings")][BsonIgnoreIfNull]//child document obj

public IList<ToppingOrder> toppingOrder { get; set; }

[Required][BsonRepresentation(BsonType.Double)]

public double cost { get; set; }

public PizzaOrder()

{

toppingOrder = new List<ToppingOrder>();

this.\_id = ObjectId.GenerateNewId();

}

public void setPizzaTotal()

{//sets the total for the pizza order

double cost = 0;

if(this.size == "Small")

{//size conditional

cost = cost + 7.45;

}

else if(this.size == "Medium")

{

cost = cost + 10.45;

}

else if(this.size == "Large")

{

cost = cost + 13.45;

}

foreach(var a in this.toppingOrder)

{

if(a.portion == "Light")

{//topping portion conditional

cost = cost + .5;//50 cents for light topping

}

else if(a.portion == "Normal")

{

cost = cost + 1;//One dollar for normal topping

}

else if(a.portion == "Extra")

{

cost = cost + 1.5;//One dollar and fifty cents for extra topping

}

}

this.cost = Math.Round((cost \* this.quantity), 2, MidpointRounding.AwayFromZero);

}

}

ToppingOrder

Model depicts an inner array in the order collection that contains each individual topping order. A list of this type of object is present in the PizzaOrder model. The model contains attributes for an identifier, pizza order identifier, name, and portion.

public class ToppingOrder

{

[BsonId][Required]

public ObjectId \_id { get; set; }

[Required]

public ObjectId pizza\_id { get; set; }

[Required]

public string name { get; set; }

[Required]

public string portion { get; set; }

public ToppingOrder()

{

this.\_id = ObjectId.GenerateNewId();

}

}

Topping

Model depicts the Topping. A list of this type of object is present in the PizzaOrder model. The model contains attributes for an identifier, name, quantity used, and quantity ceiling.

public class Topping

{

[BsonId]

public ObjectId \_id { get; set; }

[Required]

public string name { get; set; }

[BsonRepresentation(BsonType.Int32)]

public int quantUsed { get; set; }

[Required][BsonRepresentation(BsonType.Int32)]

public int quantCeiling { get; set; }

}

Application Views

\_Layout

Master view which houses each individual view in the application.

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1">

<meta name="theme-color" content="#D32F2F">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<title>@ViewBag.Title - My Long Island Pizza</title>

<link rel="icon" type="image/png" href="~/Images/favIcon.png">

<link rel="stylesheet" href="~/CSS/styles.css">

<link href='https://fonts.googleapis.com/css?family=Roboto:400,300' rel='stylesheet' type='text/css'>

@Scripts.Render("~/bundles/jquery")

@Scripts.Render("~/bundles/modernizr")

<script src="~/Scripts/velocity.min.js"></script>

<script src="~/Scripts/anim.js"></script>

</head>

<body body onload ="init()">

<div class="off-canvas">

<div class="menu-wrapper">

<div class="primary-nav menu-element">

<div class="nav-content">

<div id="drawerHead">

<div id="headerText">Menu</div>

</div>

<ul class="nav">

<li class="nav-item">@Html.ActionLink("Home", "Index", null, new { @class = "fadeButton" })</li>

<li class="nav-item">@Html.ActionLink("Order", "Order", null, new { @class = "fadeButton" })</li>

</ul>

</div>

</div>

</div>

</div>

<div class="container">

<div id="body-header">

<h1 class="mobile-view">@ViewBag.Title</h1>

@Html.Raw(File.ReadAllText(Server.MapPath("~/images/letter.svg")))

<h1 class="desktop-view">Long Island Pizza</h1>

<div class="nav-wrapper">

<label id="nav-trigger-anim"><span></span></label>

</div>

<span class="number desktop-view">Call Us at 555-555-5555</span>

</div>

<div id="content">

@RenderBody()

</div>

</div>

@RenderSection("scripts", required: false)

</body>

</html>

Index

View that displays when the user first arrives on the site. Links to the order page are present in both the navigation and banner.

@{

ViewBag.Title = "Long Island Pizza";

}

<div id="banner">

<img id="bannerImg" src="~/Images/pizza-stock.jpg"/>

<div id="gradient"></div>

<span class="bannerText">Starting at $7.45</span>

@Html.ActionLink("Order Now", "Order", null, new { @class = "btn btn-3" })

</div>

<div id="text">

<h2>Nothing but the Best</h2>

<div id="textContent">

You simply won't believe your taste buds when you chow down on one of our delicious home made pizzas.

There's a reason Long Island Pizza is the number one pizza restaurant in Suffolk County. You'll be back, and we'll be waiting.

<span class="buttonArea mobile-view"><a href="tel:555-555-5555" class="btn btn-2">Call Us</a></span>

</div>

</div>

<div id="right">

<div id="social">

<h2>Find us on</h2>

<a href="https://instagram.com/the\_og\_peruvian\_puff\_pepper/" id="socialLink" target="\_blank">@Html.Raw(File.ReadAllText(Server.MapPath("~/images/instagram19.svg")))</a>

<a href="https://www.facebook.com/ricardoandrebeas" id="socialLink" target="\_blank">@Html.Raw(File.ReadAllText(Server.MapPath("~/images/facebook2.svg")))</a>

<a href="https://plus.google.com/u/0/+RicardoBeas249/posts" id="socialLink" target="\_blank">@Html.Raw(File.ReadAllText(Server.MapPath("~/images/google-plus.svg")))</a>

</div>

<div class="quote">

<h3>

"Long Island Pizza makes my trip to the island every year worth it" - Jane Doe

</h3>

</div>

</div>

<hr class="break">

Order

View that displays when the user arrives to the order page. Page displays an SVG, button, and error message when the user’s cart is empty. Otherwise, current pizza orders and order totals are displayed. A button is also present on each pizza order card to delete that pizza order from the order. Two buttons are present to either add another pizza or go straight to the checkout page.

@model PizzaStore.Models.Order

@{

ViewBag.Title = "Order";

}

@if (ViewBag.Order != null)

{

<h2 class="heading2">Order</h2>

for (var c = 0; c < Model.pizzaOrder.Count; c++)

{

<div id="text3">

<h2>@Html.Raw("(" + Model.pizzaOrder[c].quantity + ") ")

@Html.Raw(Model.pizzaOrder[c].size + " Pizza(s)")</h2>

<a href="@Url.Action("Delete", new { id= c })">@Html.Raw(File.ReadAllText(Server.MapPath("~/images/cross.svg")))</a>

@if(Model.pizzaOrder[c].toppingOrder.Any())

{

<h3>Toppings:</h3>

}

@foreach (var d in Model.pizzaOrder[c].toppingOrder)

{

<span class="topping">

@Html.Raw(d.portion + " ")

@Html.Raw(d.name)

</span>

}

<h3>@Html.Raw("$" + Model.pizzaOrder[c].cost.ToString("0.00"))</h3>

</div>

}

<div class="totalDetails">

<h3 class="totals">SubTotal: @Html.Raw(" $" + Model.subTotal.ToString("0.00"))</h3>

<h3 class="totals">Sales Tax: @Html.Raw(" $" + Model.salesTax.ToString("0.00"))</h3>

<h3 class="totals">Total: @Html.Raw(" $" + Model.total.ToString("0.00"))</h3>

</div>

@Html.ActionLink("Add Pizza", "AddPizza", null, new { @class = "btn btn-2" })

@Html.ActionLink("Checkout", "CheckOut", null, new { @class = "btn btn-2" })

}

else

{

<h2 class="message">@Html.Raw(ViewBag.Message)</h2>

@Html.Raw(File.ReadAllText(Server.MapPath("~/images/pizzaslice.svg")))

@Html.Raw(File.ReadAllText(Server.MapPath("~/images/pizzaslice-sm.svg")))

@Html.ActionLink("Add Pizza", "AddPizza", null, new { @class = "btn btn-2 btn-center" })

}

<hr class="break">

AddPizza

View that displays when the user arrives to the add pizza page. Page displays a form with inputs for pizza size, quantity, and checkbox for each topping. Once a checkbox for a topping is checked off, a dropdown for that topping’s portion is made available to the user.

@{

ViewBag.Title = "Add Pizza";

var a = 0;

var d = 0;

}

@{

IEnumerable<SelectListItem> ItemsInDropDown = new List<SelectListItem>(){

new SelectListItem{ Text="Small - $7.45", Value="Small"},

new SelectListItem{ Text="Medium - $10.45", Selected = true, Value="Medium"},

new SelectListItem{ Text="Large - $13.45", Value="Large"}

};

IEnumerable<SelectListItem> ItemsInDropDown2 = new List<SelectListItem>(){

new SelectListItem{ Text="Light - 50¢", Value="Light"},

new SelectListItem{ Text="Normal - $1.00", Selected = true, Value="Normal"},

new SelectListItem{ Text="Extra - $1.50", Value="Extra"}

};

}

@using (Html.BeginForm())

{

<div id="text2">

<h2>Create Pizza</h2>

@if (ViewBag.error != null)

{

<span class="error">@Html.Raw("-" + ViewBag.error)</span>

}

<div class="form-group">

@foreach(var name in Model.toppingName)

{

@Html.HiddenFor(model => model.toppingName[d], new { Value = @name });

d++;

}

<label class="formLabel">Size</label>

@Html.DropDownListFor(model => model.size, ItemsInDropDown , new { @class = "field noCheck" })

</div>

<div class="form-group">

<label class="formLabel">Quantity <span class="sm">(Maximum of 10)</span></label>

@Html.TextBoxFor(model => model.quantity, new { type = "number", min = "1", max = "10", @class = "field2 noCheck" })

</div>

<div class="form-group">

@foreach (var topping in Model.toppingName)

{

var label = "toppingOrder[" + a + "].name";

<label for = "@label" class="">@topping</label>

@Html.CheckBoxFor(model => model.ifChecked[a], new {name = topping, @class = "checkbox toppingClick " + topping })

@Html.DropDownListFor(model => model.toppingPortion[a], ItemsInDropDown2, new {@class = "field " + topping + " fieldList"})

a++;

}

</div>

<div class="form-group">

<input type="submit" value="Add to Order" class="btn btn-4" />

</div>

</div>

<hr class="break">

}

CheckOut

View that displays when the user arrives to the checkout page. Page displays a form with inputs for first name, last name, and checkbox to indicate a delivery. If the checkbox is checked off, the form extends to reveal additional inputs for street address, city, state, and zip code.

@model PizzaStore.Models.CheckOutViewModel

@{

ViewBag.Title = "Check Out";

}

@{

IEnumerable<SelectListItem> ItemsInDropDown = new List<SelectListItem>(){

new SelectListItem{ Text="Alaska", Value="Alaska"},

new SelectListItem{ Text="Alabama", Value="Alabama"},

new SelectListItem{ Text="Arkansas", Value="Arkansas"},

new SelectListItem{ Text="Arizona", Value="Arizona"},

new SelectListItem{ Text="California", Value="California"},

new SelectListItem{ Text="Colorado", Value="Colorado"},

new SelectListItem{ Text="Connecticut", Value="Connecticut"},

new SelectListItem{ Text="Delaware", Value="Delaware"},

new SelectListItem{ Text="Florida", Value="Florida"},

new SelectListItem{ Text="Georgia", Value="Georgia"},

new SelectListItem{ Text="Hawaii", Value="Hawaii"},

new SelectListItem{ Text="Iowa", Value="Iowa"},

new SelectListItem{ Text="Idaho", Value="Idaho"},

new SelectListItem{ Text="Illinois", Value="Illinois"},

new SelectListItem{ Text="Indiana", Value="Indiana"},

new SelectListItem{ Text="Kansas", Value="Kansas"},

new SelectListItem{ Text="Kentucky", Value="Kentucky"},

new SelectListItem{ Text="Louisiana", Value="Louisiana"},

new SelectListItem{ Text="Massachusetts", Value="Massachusetts"},

new SelectListItem{ Text="Maryland", Value="Maryland"},

new SelectListItem{ Text="Maine", Value="Maine"},

new SelectListItem{ Text="Michigan", Value="Michigan"},

new SelectListItem{ Text="Minnesota", Value="Minnesota"},

new SelectListItem{ Text="Missouri", Value="Missouri"},

new SelectListItem{ Text="Mississippi", Value="Mississippi"},

new SelectListItem{ Text="Montana", Value="Montana"},

new SelectListItem{ Text="North Carolina", Value="North Carolina"},

new SelectListItem{ Text="North Dakota", Value="North Dakota"},

new SelectListItem{ Text="Nebraska", Value="Nebraska"},

new SelectListItem{ Text="New Hampshire", Value="New Hampshire"},

new SelectListItem{ Text="New Jersey", Value="New Jersey"},

new SelectListItem{ Text="New Mexico", Value="New Mexico"},

new SelectListItem{ Text="Nevada", Value="Nevada"},

new SelectListItem{ Text="New York", Selected = true, Value="New York"},

new SelectListItem{ Text="Ohio", Value="Ohio"},

new SelectListItem{ Text="Oklahoma", Value="Oklahoma"},

new SelectListItem{ Text="Oregon", Value="Oregon"},

new SelectListItem{ Text="Pennsylvania", Value="Pennsylvania"},

new SelectListItem{ Text="Rhode Island", Value="Rhode Island"},

new SelectListItem{ Text="South Carolina", Value="South Carolina"},

new SelectListItem{ Text="South Dakota", Value="South Dakota"},

new SelectListItem{ Text="Tennessee", Value="Tennessee"},

new SelectListItem{ Text="Texas", Value="Texas"},

new SelectListItem{ Text="Utah", Value="Utah"},

new SelectListItem{ Text="Virginia", Value="Virginia"},

new SelectListItem{ Text="Vermont", Value="Vermont"},

new SelectListItem{ Text="Washington", Value="Washington"},

new SelectListItem{ Text="Wisconsin", Value="Wisconsin"},

new SelectListItem{ Text="West Virginia", Value="West Virginia"},

new SelectListItem{ Text="Wyoming", Value="Wyoming"}

};

}

@using (Html.BeginForm())

{

<div id="text2">

<h2>CheckOut</h2>

@if(ViewBag.ErrorList != null)

{

foreach (var a in ViewBag.ErrorList)

{

<span class="error">@Html.Raw("-" + a)</span>

}

}

<div class="form-group">

<label class="formLabel">First Name</label>

@Html.TextBoxFor(model => model.firstName, new { @class = "field", required = "" })

</div>

<div class="form-group">

<label class="formLabel">Last Name</label>

@Html.TextBoxFor(model => model.lastName, new { @class = "field", required = "" })

</div>

<div class="form-group">

<label class="formLabel">Delivery? <span class="sm">($1.00 Charge)</span></label>

@Html.CheckBoxFor(model => model.delivery, new { @class = "checkbox delivery"})

</div>

<div class="form-group adressForm">

<label class="formLabel">Street Adress</label>

@Html.TextBoxFor(model => model.streetAddress, new { @class = "field adressInput" })

</div>

<div class="form-group adressForm">

<label class="formLabel">City</label>

@Html.TextBoxFor(model => model.city, new { @class = "field adressInput", })

</div>

<div class="form-group adressForm">

<label class="formLabel">State</label>

@Html.DropDownListFor(model => model.state, ItemsInDropDown, new { @class = "field adressInput" })

</div>

<div class="form-group adressForm">

<label class="formLabel">Zip Code</label>

@Html.TextBoxFor(model => model.zipCode, new { @class = "field adressInput", maxlength = "5", type = "number" })

</div>

<div class="form-group">

<input type="submit" value="Review Order" class="btn btn-4" />

</div>

</div>

<hr class="break">

}

Review

View that displays when the user arrives to the review page. Page displays the entire user’s order with each pizza order, contact/delivery information, and order totals being displayed.

@model PizzaStore.Models.Order

@{

ViewBag.Title = "Review";

}

<div id="text2">

<h2>Order</h2>

@for (var c = 0; c < Model.pizzaOrder.Count; c++)

{

<span class="reviewHead">@Html.Raw("(" + Model.pizzaOrder[c].quantity + ") ")

@Html.Raw(Model.pizzaOrder[c].size + " Pizza(s)")</span>

if (Model.pizzaOrder[c].toppingOrder.Any())

{

<h3>Toppings:</h3>

}

foreach (var d in Model.pizzaOrder[c].toppingOrder)

{

<span class="topping">

@Html.Raw(d.portion + " ")

@Html.Raw(d.name)

</span>

}

<h3>@Html.Raw("$" + Model.pizzaOrder[c].cost.ToString("0.00"))</h3>

<hr class="break2">

}

<h3>To:</h3>

<span class="topping">

@Html.Raw(Model.firstName + " ")

@Html.Raw(Model.lastName)

</span>

@if (Model.delivery == true)

{

<h3>Address:</h3>

<span class="topping">@Html.Raw(Model.streetAddress)</span>

<span class="topping">@Html.Raw(Model.city + ", ")@Html.Raw(Model.state + " ")@Html.Raw(Model.zipCode)</span>

}

<br/><!--#lazzy-->

</div>

<div class="totalDetails2">

<h3 class="totals">SubTotal: @Html.Raw(" $" + Model.subTotal.ToString("0.00"))</h3>

<h3 class="totals">Sales Tax: @Html.Raw(" $" + Model.salesTax.ToString("0.00"))</h3>

@if(Model.delivery == true)

{

<h3 class="totals">Delivery Charge: @Html.Raw(" $1.00")</h3>

}

<h3 class="totals">Total: @Html.Raw(" $" + Model.total.ToString("0.00"))</h3>

</div>

@Html.ActionLink("Place Order", "OrderPlaced", new { reviewFlag = "true" }, new { @class = "btn btn-2 btn-review" })

<hr class="break">

OrderPlaced

View that displays when the user arrives to the order placed page. Page displays a conformation message, the order‘s time to completion, and the order id to the user.

@model PizzaStore.Models.Order

@{

ViewBag.Title = "Order Placed";

TimeSpan span = TimeSpan.FromMinutes(Model.complTime);

}

<div id="text2">

<h2>Order Placed!</h2>

<span class="reviewHead">Thank you for choosing Long Island Pizza!</span>

<h3>Your Order ID is: </h3> <span class="red">@Html.Raw(Model.\_id)</span>

@if (Model.complTime > 60 && Model.complTime % 60 != 0)

{

if (Model.delivery == true)

{

<h3>Your order should arrive in: </h3> <h2>@span.ToString("hh").TrimStart('0') Hours and @span.ToString("mm") Minutes</h2>

}

else

{

<h3>Your order should be ready in: </h3> <h2>@span.ToString("hh").TrimStart('0') Hours and @span.ToString("mm") Minutes</h2>

}

}

else if (Model.complTime > 60 && Model.complTime < 120)

{

if (Model.delivery == true)

{

<h3>Your order should arrive in: </h3> <h2>@span.ToString("hh").TrimStart('0') Hour and @span.ToString("mm") Minutes</h2>

}

else

{

<h3>Your order should be ready in: </h3> <h2>@span.ToString("hh").TrimStart('0') Hour and @span.ToString("mm") Minutes</h2>

}

}

else if (Model.complTime < 60)

{

if (Model.delivery == true)

{

<h3>Your order should arrive in: </h3> <h2>@span.ToString("mm").TrimStart('0') Minutes</h2>

}

else

{

<h3>Your order should be ready in: </h3> <h2>@span.ToString("mm").TrimStart('0') Minutes</h2>

}

}

else if (Model.complTime % 60 == 0 && Model.complTime != 60)

{

if (Model.delivery == true)

{

<h3>Your order should arrive in: </h3> <h2>@span.ToString("hh").TrimStart('0') Hours</h2>

}

else

{

<h3>Your order should be ready in: </h3> <h2>@span.ToString("hh").TrimStart('0') Hours</h2>

}

}

else if (Model.complTime == 60)

{

if (Model.delivery == true)

{

<h3>Your order should arrive in: </h3> <h2>@span.ToString("hh").TrimStart('0') Hour</h2>

}

else

{

<h3>Your order should be ready in: </h3> <h2>@span.ToString("hh").TrimStart('0') Hour</h2>

}

}

<br /><!--#lazzy-->

</div>

@Html.ActionLink("Start New Order", "Order", null, new { @class = "btn btn-2 btn-review" })

<hr class="break">

Application Controllers

HomeController

Controller used for entire application. Contains routes for each route as well as validations for forms, validation for session variables, form handling, and a database interactions through the DBConnection object.

public class HomeController : Controller

{

private readonly DBConnection Context = new DBConnection();//database connection

public ActionResult index()

{//home

return View();

}

[HttpPost]

public ActionResult addPizza(AddPizzaViewModel pizzaView)

{//post to add pizza

if(pizzaView.quantity <= 0)

{//if quantity is less than one

TempData["Error"] = "Must Order at Least 1 Pizza";

return RedirectToAction("AddPizza");

}

else if(pizzaView.quantity > 10)

{//if quantity is greater than 10

TempData["Error"] = "No More Than 10 Pizza Per Individual Order";

return RedirectToAction("AddPizza");

}

else

{

}

PizzaOrder pizza = new PizzaOrder();

pizza.size = pizzaView.size;

pizza.quantity = pizzaView.quantity;

for (var b = 0; b < pizzaView.ifChecked.Count; b++)

{

if (Convert.ToBoolean(pizzaView.ifChecked[b]) == true)

{

ToppingOrder topping = new ToppingOrder();

var toppings = Context.Toppings.FindAll().SetSortOrder(SortBy<Topping>.Ascending(r => r.name));

foreach (var a in toppings)

{

if(a.name == pizzaView.toppingName[b])

{

topping.\_id = a.\_id;//assigns each topping order its topping ID from the toppings collection

}

}

topping.name = Convert.ToString(pizzaView.toppingName[b]);

topping.portion = Convert.ToString(pizzaView.toppingPortion[b]);

topping.pizza\_id = pizza.\_id;//assigns topping order its pizza ID

pizza.toppingOrder.Add(topping);

}

else

{

}

}

pizza.setPizzaTotal();//sets the pizzas total

Order anOrder = (Order)Session["Order"];

pizza.order\_id = anOrder.\_id;//assisgns pizza its order ID

anOrder.pizzaOrder.Add(pizza);//adds Pizza to order of pizzas

anOrder.setOrderTotal();//sets the order total

Session["Order"] = anOrder;//save order to Session object

return RedirectToAction("Order");

}

public ActionResult addPizza()

{//add pizza route

if (Session["Order"] != null)

{//if session exists

AddPizzaViewModel aView = new AddPizzaViewModel();

var toppings = Context.Toppings.FindAll().SetSortOrder(SortBy<Topping>.Ascending(r => r.name));//gets toppings from DataBase

foreach (var a in toppings)

{

aView.toppingName.Add(a.name);

aView.ifChecked.Add(false);

aView.toppingPortion.Add("Normal");

}

if (TempData["Error"] != null)

{//gets error message

ViewBag.error = (string)TempData["Error"];

}

else

{

}

return View(aView);

}

else

{

return RedirectToAction("Order");

}

}

public ActionResult Order()

{//order summary route

if (Session["Order"] != null)

{//if session exists

Order anOrder = (Order)Session["Order"];//get temporary order Object

if (anOrder.pizzaOrder.Any())

{

ViewBag.Order = anOrder;

return View(anOrder);

}

else

{//no pizzas in order

ViewBag.Message = "No Pizzas Purchased";

return View();

}

}

else

{//no session

Order anOrder = new Order();

Session["Order"] = anOrder;//new session

ViewBag.Message = "No Pizzas Purchased";//default message

return View();

}

}

public ActionResult Delete(int id)

{//delete pizza order route

if (Session["Order"] != null)

{//if session exists

Order anOrder = (Order)Session["Order"];//get session object

anOrder.pizzaOrder.RemoveAt(id);//removes pizza from order

anOrder.setOrderTotal();//update total

Session["Order"] = anOrder;//save order to Session object

return RedirectToAction("Order");

}

else

{

return RedirectToAction("Order");

}

}

public ActionResult CheckOut()

{//final checkout form

if (Session["Order"] != null)

{//if session exists

if(TempData["ErrorList"] != null)

{//list of error messages for validation

ViewBag.errorList = (List<string>)TempData["ErrorList"];

}

else

{

}

Order anOrder = (Order)Session["Order"];//get temporary order Object

if (anOrder.pizzaOrder.Any())

{

return View();

}

else

{//no pizzas in order

return RedirectToAction("Order");

}

}

else

{//if there is no order session redirect back to order page and create session

return RedirectToAction("Order");

}

}

[HttpPost]

public ActionResult CheckOut(CheckOutViewModel aView)

{//final checkout form and submissiom

if(aView.delivery == false)

{//no delivery

if(String.IsNullOrEmpty(aView.firstName) == true || String.IsNullOrEmpty(aView.lastName) == true)

{//check for empty fields

List<string> errorList = new List<string>();

if(String.IsNullOrEmpty(aView.firstName) == true)

{

errorList.Add("Please Provide a First Name");

}

else

{

}

if (String.IsNullOrEmpty(aView.lastName) == true)

{

errorList.Add("Please Provide a Last Name");

}

else

{

}

TempData["ErrorList"] = errorList;

return RedirectToAction("Checkout");

}

else

{//if form is complete

Order anOrder = (Order)Session["Order"];

anOrder.firstName = Context.StripTagsCharArray(aView.firstName);

anOrder.lastName = Context.StripTagsCharArray(aView.lastName);

anOrder.delivery = aView.delivery;

Session["Order"] = anOrder;

return RedirectToAction("Review");

}

}

else

{//delivery

if(String.IsNullOrEmpty(aView.firstName) == true || String.IsNullOrEmpty(aView.lastName) == true || aView.zipCode <= 0 || String.IsNullOrEmpty(aView.streetAddress) == true || String.IsNullOrEmpty(aView.city) == true || String.IsNullOrEmpty(aView.state) == true)

{//check for empty fields

List<string> errorList = new List<string>();

if(String.IsNullOrEmpty(aView.firstName) == true)

{

errorList.Add("Please Provide a First Name");

}

else

{

}

if (String.IsNullOrEmpty(aView.lastName) == true)

{

errorList.Add("Please Provide a Last Name");

}

else

{

}

if (String.IsNullOrEmpty(aView.streetAddress) == true)

{

errorList.Add("Please Provide a Street Address");

}

else

{

}

if (String.IsNullOrEmpty(aView.city) == true)

{

errorList.Add("Please Provide a City");

}

else

{

}

if (String.IsNullOrEmpty(aView.state) == true)

{

errorList.Add("Please Provide a State");

}

else

{

}

if (Convert.ToString(aView.zipCode).Length == 5 || Convert.ToString(aView.zipCode).Length == 4)

{//no error

}

else

{

errorList.Add("Zip Code Is Invalid");

}

TempData["ErrorList"] = errorList;

return RedirectToAction("Checkout");

}

else

{//if form is complete

if (Convert.ToString(aView.zipCode).Length == 5 || Convert.ToString(aView.zipCode).Length == 4)

{//no error

}

else

{//innapropriate zip code length

List<string> errorList = new List<string>();

errorList.Add("Zip Code Is Invalid");

TempData["ErrorList"] = errorList;

return RedirectToAction("Checkout");

}

Order anOrder = (Order)Session["Order"];

anOrder.firstName = Context.StripTagsCharArray(aView.firstName);

anOrder.lastName = Context.StripTagsCharArray(aView.lastName);

anOrder.delivery = aView.delivery;

anOrder.streetAddress = Context.StripTagsCharArray(aView.streetAddress);

anOrder.city = Context.StripTagsCharArray(aView.city);

anOrder.state = Context.StripTagsCharArray(aView.state);

anOrder.zipCode = aView.zipCode;

anOrder.setOrderTotal();

Session["Order"] = anOrder;

return RedirectToAction("Review");

}

}

}

public ActionResult Review()

{//order review route

if (Session["Order"] != null)

{//if session exists

Order anOrder = (Order)Session["Order"];

if (String.IsNullOrEmpty(anOrder.firstName) == true || String.IsNullOrEmpty(anOrder.lastName) == true)

{//in case user did not go to checkout and typed in review address

return RedirectToAction("CheckOut");

}

else

{

}

return View(anOrder);

}else

{//no session

return RedirectToAction("Order");

}

}

public ActionResult OrderPlaced(string reviewFlag)

{//order submission summary route

if (Session["Order"] != null && reviewFlag == "true")

{//coming from review

Order anOrder = (Order)Session["Order"];//get session order Object

anOrder.setOrderTime();//sets order completion time

foreach (var a in anOrder.pizzaOrder)

{//updates DB on what toppings were used

foreach (var b in a.toppingOrder)

{

int num = 0;

if(b.portion == "Light")

{

num = 1;

num = num \* a.quantity;

var topping = Context.Toppings.FindOneById(b.\_id);

topping.quantUsed = topping.quantUsed + num;

Context.Toppings.Save(topping);

}

else if(b.portion == "Normal")

{

num = 2;

num = num \* a.quantity;

var topping = Context.Toppings.FindOneById(b.\_id);

topping.quantUsed = topping.quantUsed + num;

Context.Toppings.Save(topping);

}

else if(b.portion == "Extra")

{

num = 3;

num = num \* a.quantity;

var topping = Context.Toppings.FindOneById(b.\_id);

topping.quantUsed = topping.quantUsed + num;

Context.Toppings.Save(topping);

}

else

{

}

}

}

Context.Orders.Insert(anOrder);//inserts Order to DataBase

Session.Abandon();//abandon session

return View(anOrder);

}

else

{//if there is no order session redirect back to order page and create session

return RedirectToAction("Order");

}

}

}

JavaScript

Anim.js

Handles page events and interactions such as triggering a dropdown in the add pizza page or expanding the checkout form when the delivery checkbox is checked.

function init()

{

animations();

}

function animations()

{

var menuBoolean = false;

var menuWidthThird = 240 / $('.menu-element').length;

$('.nav-wrapper').click(function(ev) {

$('#nav-trigger-anim').toggleClass('active');

if (menuBoolean == false)

{//menu open

menuAnimIn($('.container'), menuWidthThird);

menuBoolean = true;

$('body').css("overflow", "hidden");

}

else

{//menu close

menuAnimIn($('.container'), 0);

menuBoolean = false;

$('body').css("overflow", "auto");

}

});

$(window).resize(function () {

//closes menu if screen size is changed to that of a desktop experience

if ($(window).width() > 768)

{

if (menuBoolean == true)

{

menuAnimIn($('.container'), 0);

menuBoolean = false;

$('#nav-trigger-anim').toggleClass('active');

$('body').css("overflow", "auto");

}

else

{

}

}

else {

}

});

$(window).resize(function () {

//repositions button according to screen size

var bannerHeight = ($("#bannerImg").height()) / 1.1;

$('.btn-3').css("margin-top", bannerHeight);

});

$('.delivery').click(function () {

//expands form when delivery checkbox is checked

if ($(this).is(":checked")) {

$(".adressForm").css("visibility", "visible");

$(".adressForm").css("position", "static");

}

else

{

$(".adressForm").css("visibility", "hidden");

$(".adressForm").css("position", "absolute");

}

if ($('.adressInput').attr('required'))

{

$('.adressInput').removeAttr('required');

}

else

{

$('.adressInput').attr('required', '');

}

});

$('.toppingClick').click(function () {

//activates portion dropdown when topping checkbox is checked

var className = $(this).attr('class').split(' ')[2];

if ($(this).is(":checked"))

{

$(".field." + className).css( "visibility", "visible" );

$(".field." + className).css( "position", "static" );

}

else

{

$(".field." + className).css("visibility", "hidden");

$(".field." + className).css("position", "absolute");

}

});

$(document).ready(function () {

//fade in for pages

$('#content').fadeIn(400);

});

$(document).ready(function () {

//repositions button when page is first loaded

var bannerHeight = ($("#bannerImg").height()) / 1.1;

$('.btn-3').css("margin-top", bannerHeight);

});

}

function menuAnimIn(obj, fromLeft) {

$(obj).velocity({

left: fromLeft

}, {

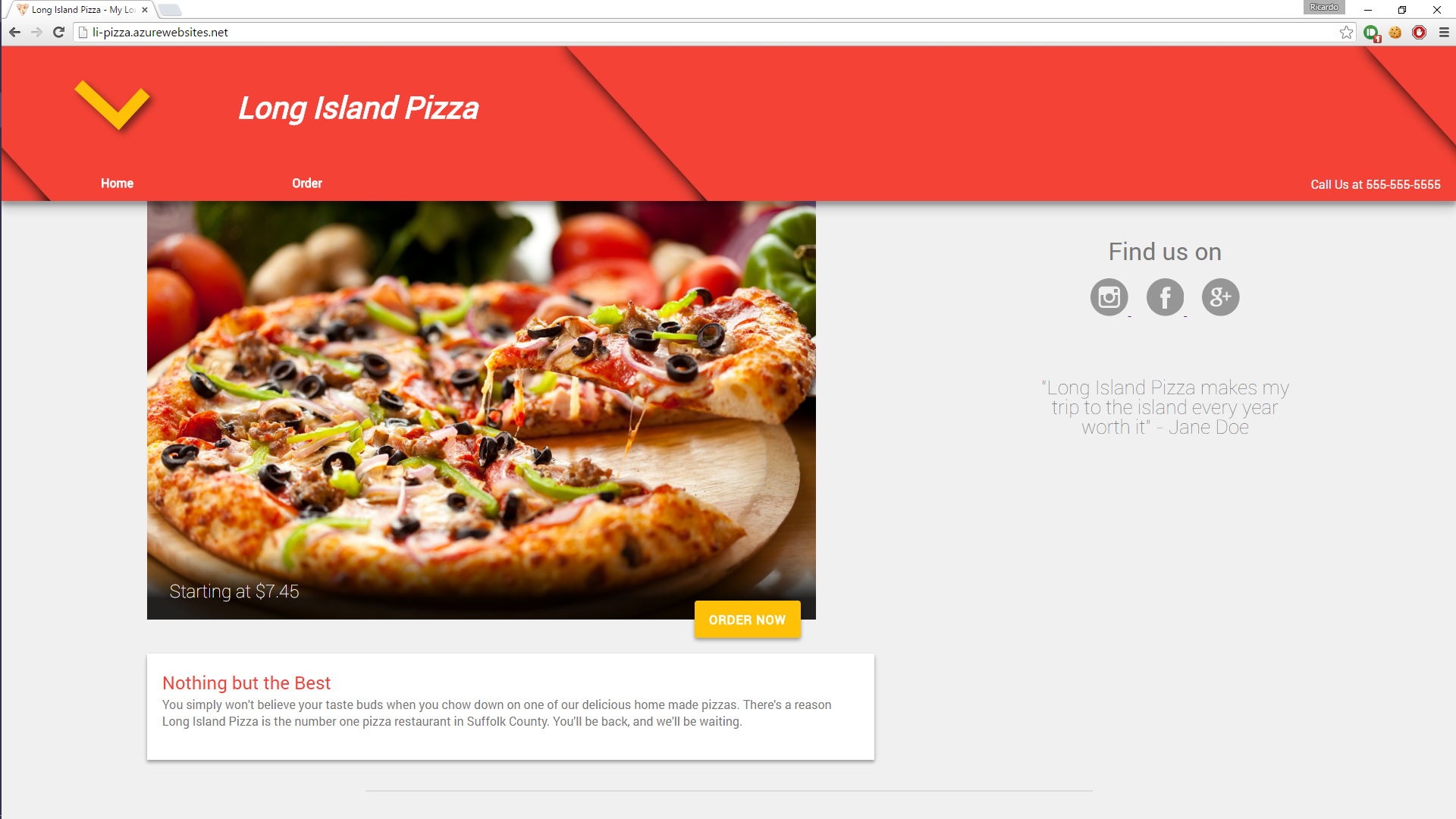
easing: 'easeInSine'

})

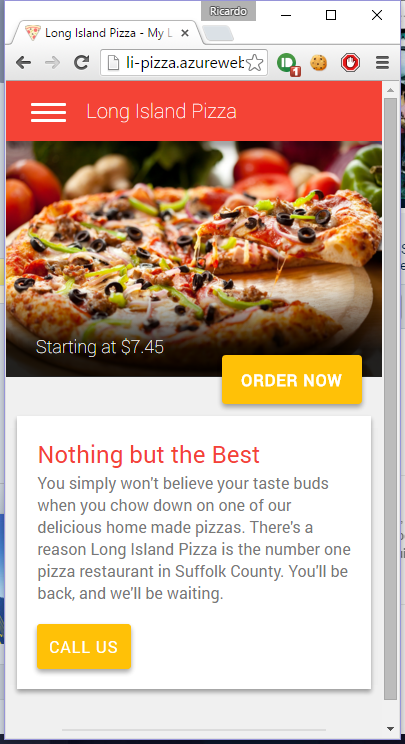
}

Application Screenshots

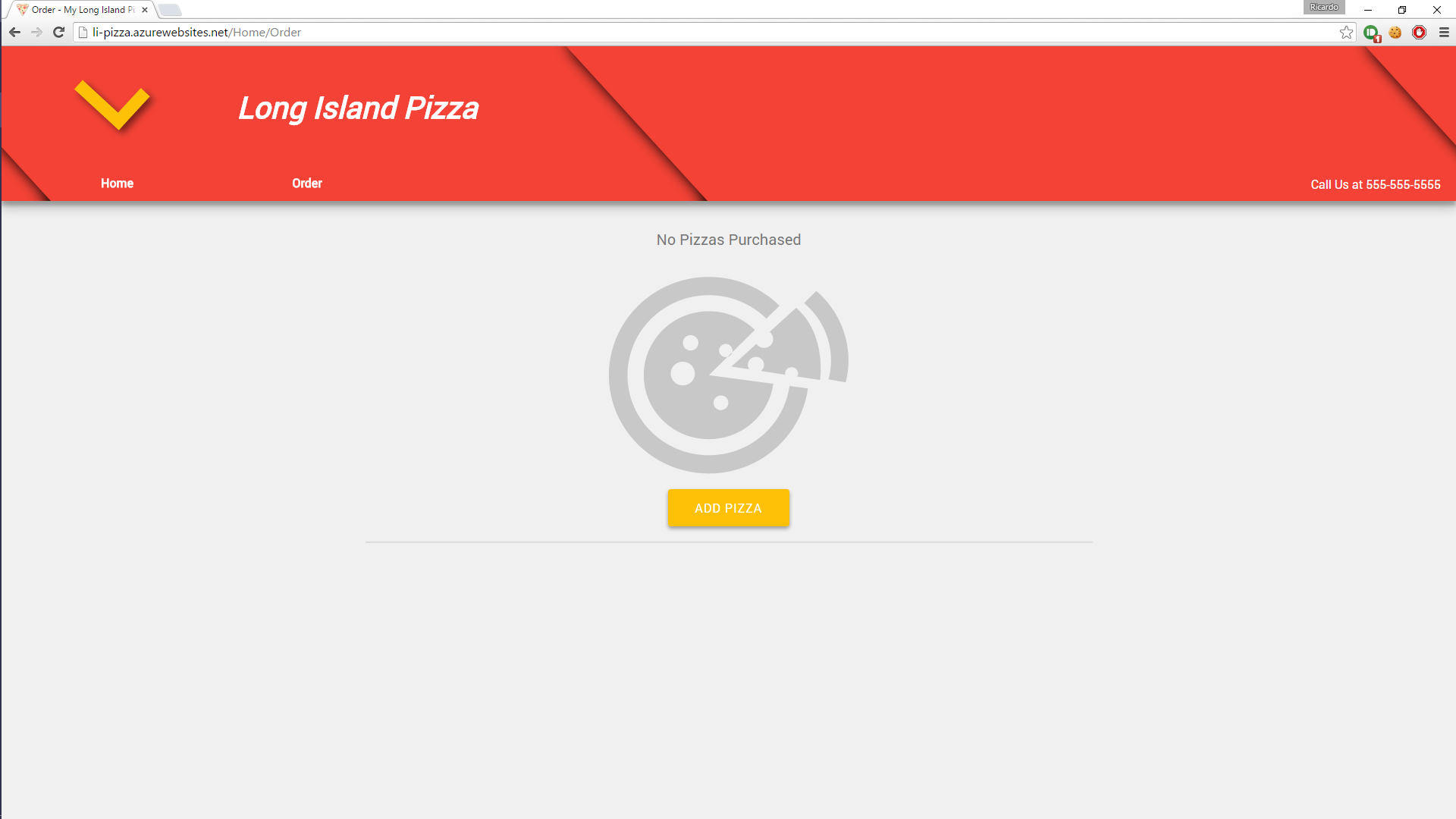
Desktop View - Home



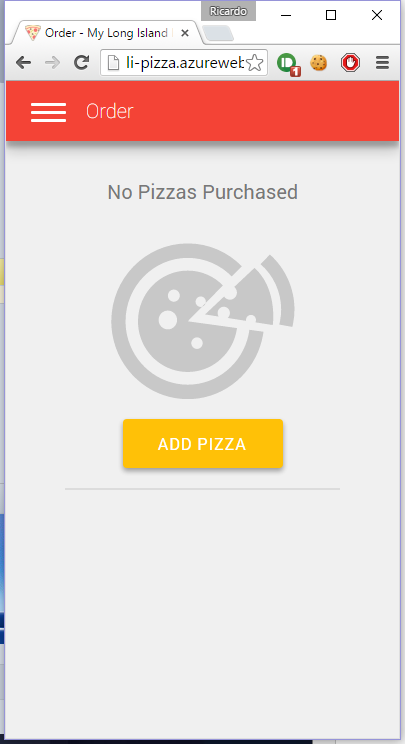
Mobile View - Home



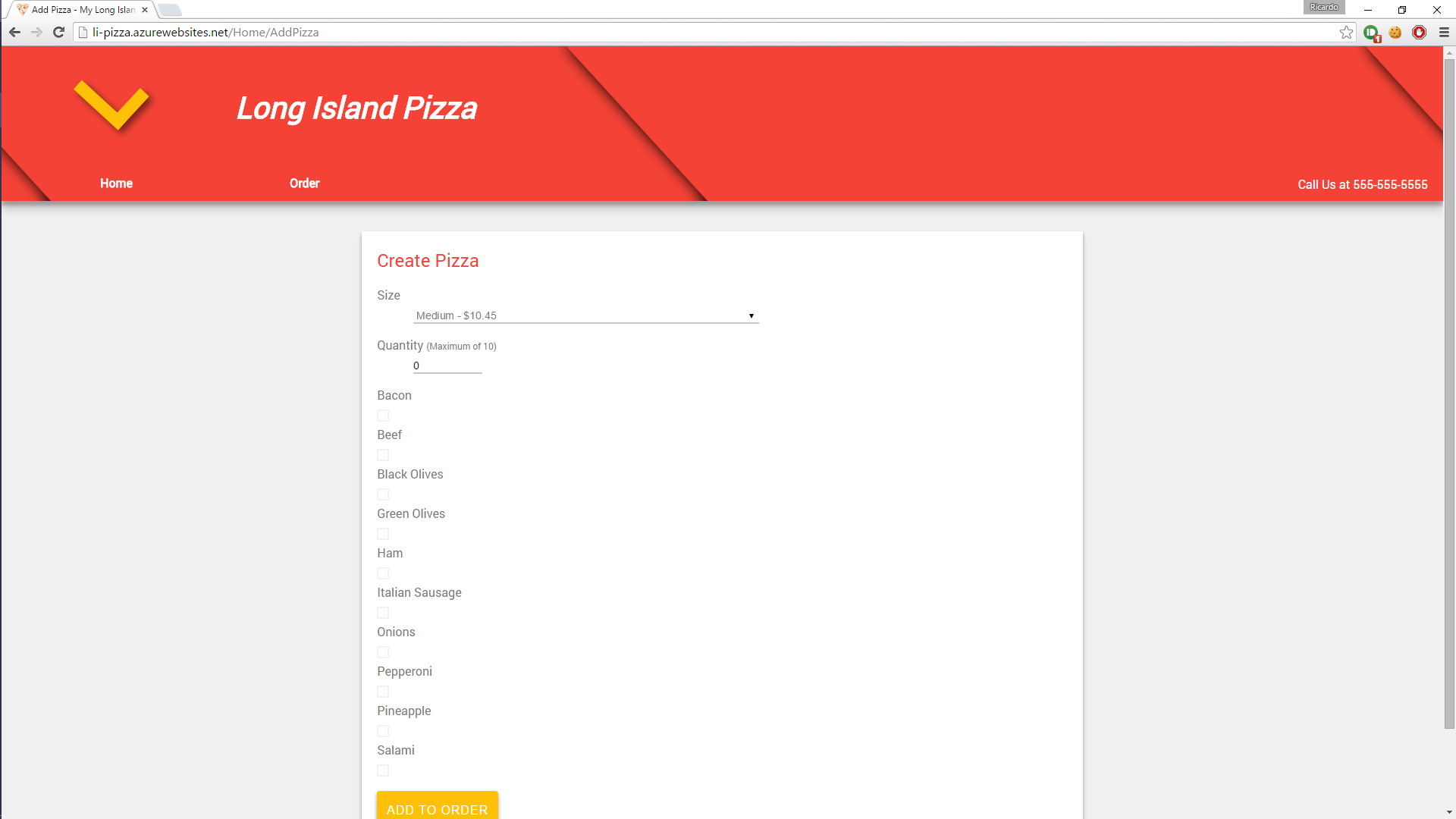
Desktop View - Order



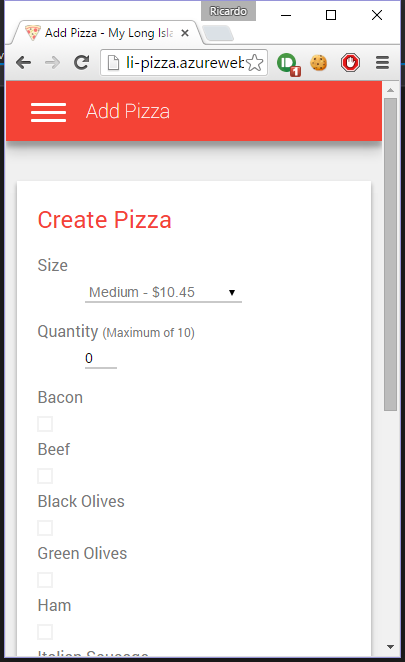
Mobile View - Order



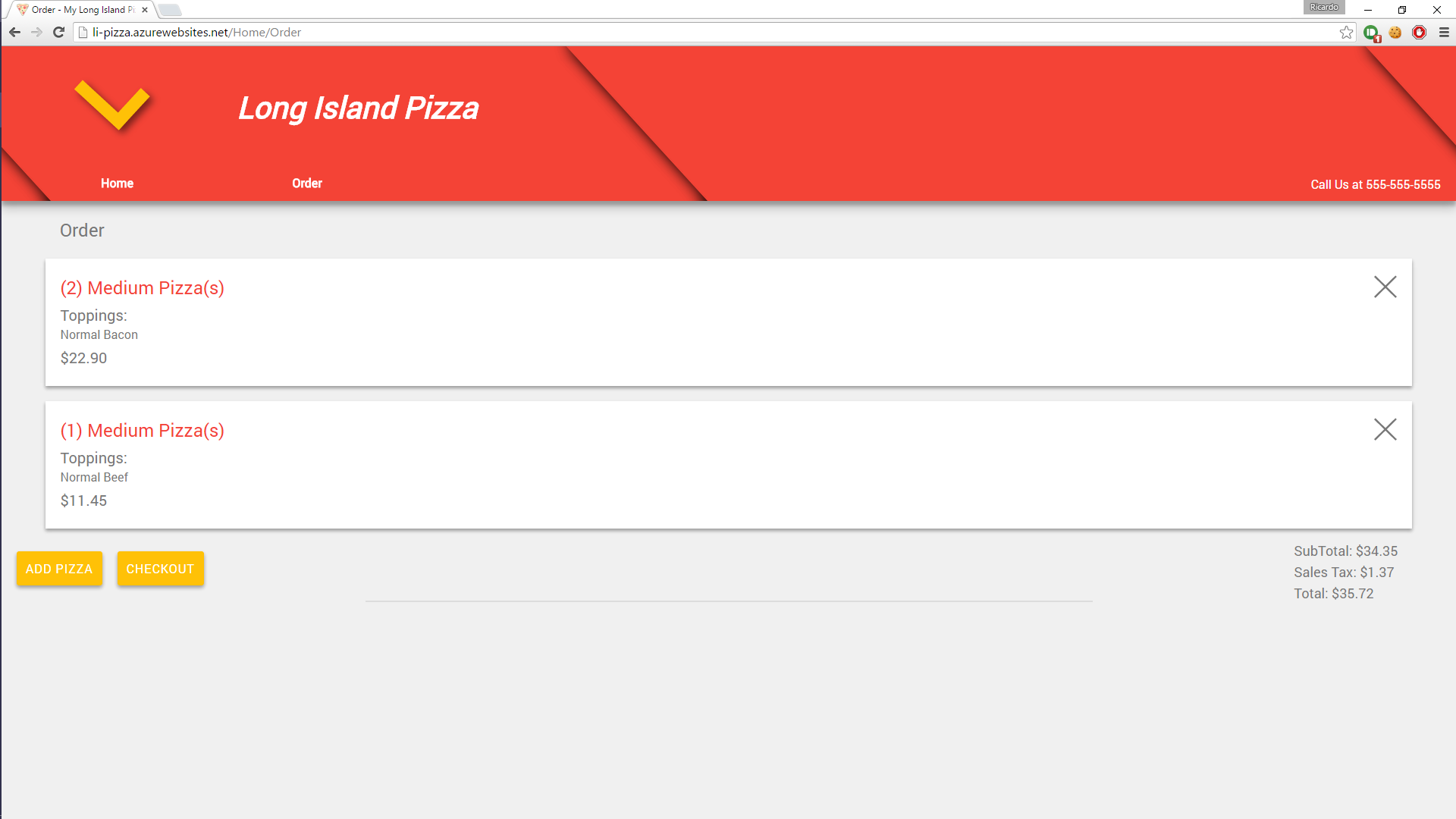
Desktop View – Add Pizza



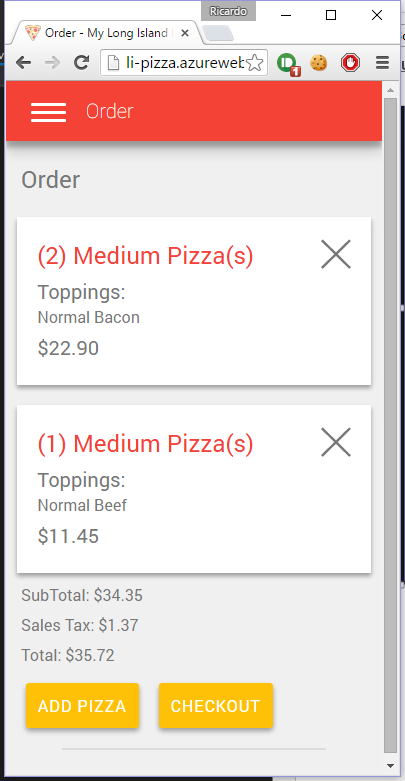
Mobile View – Add Pizza



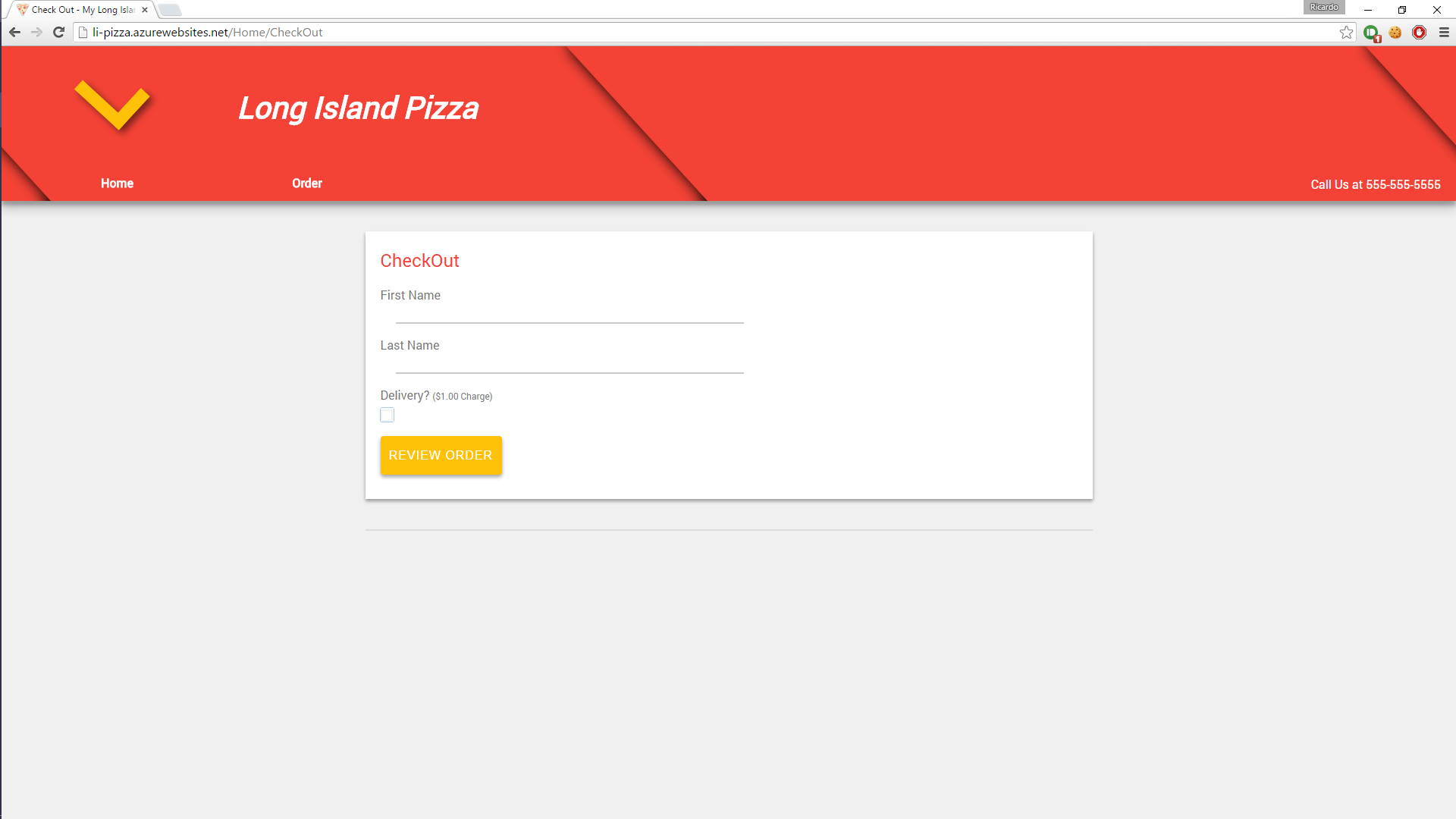
Desktop View – Order (With pizzas in cart)



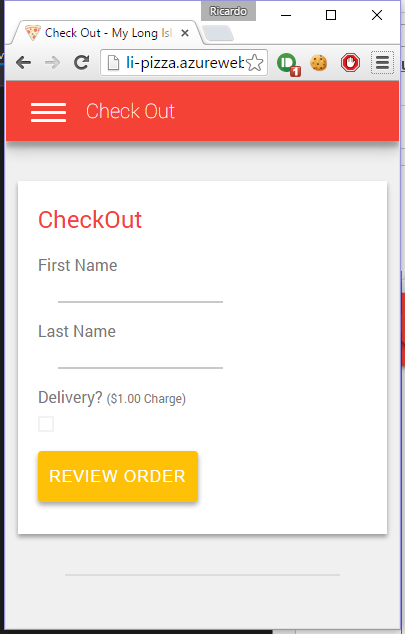
Mopbile View – Order (With pizzas in cart)



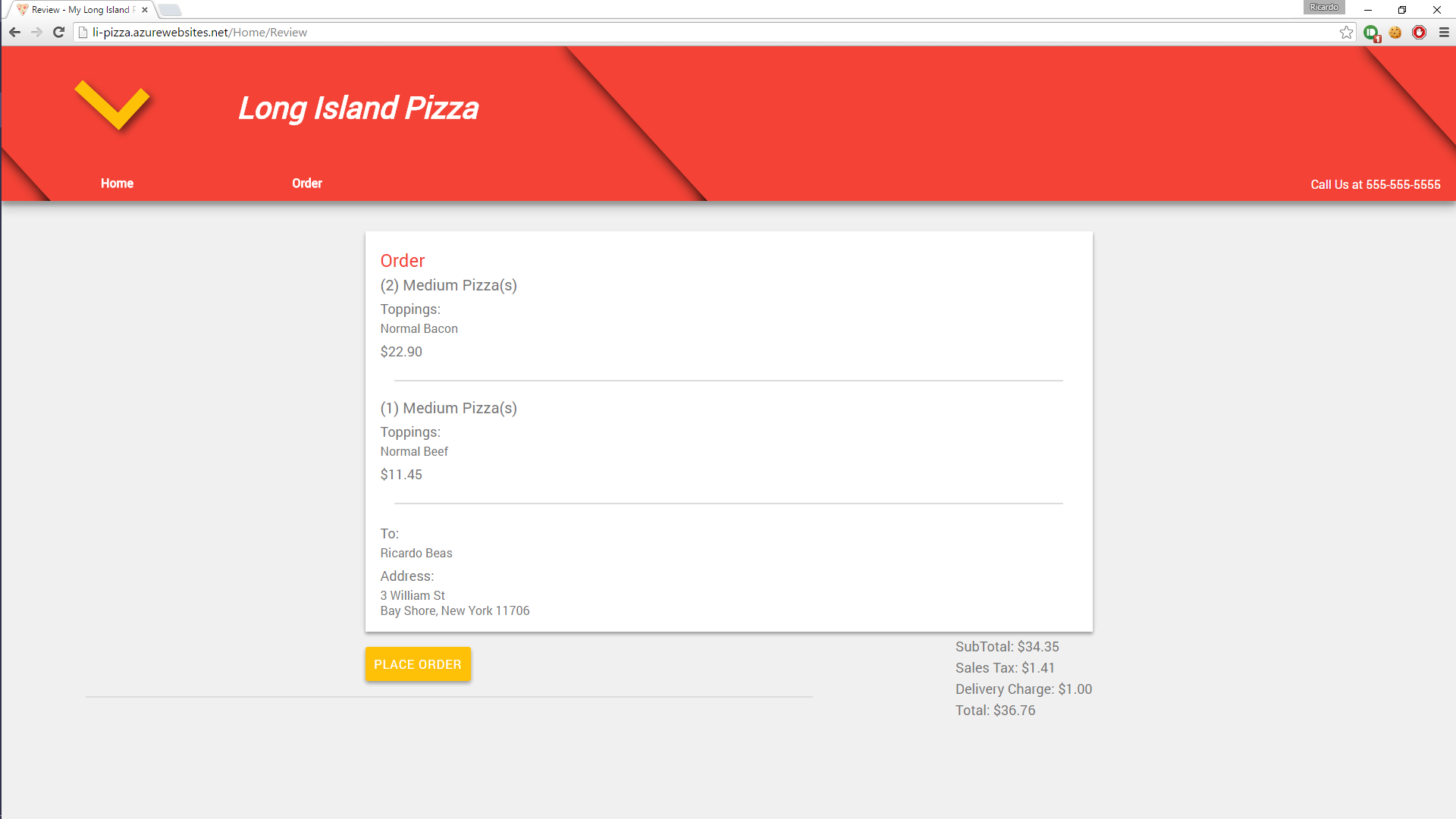
Desktop View – Checkout



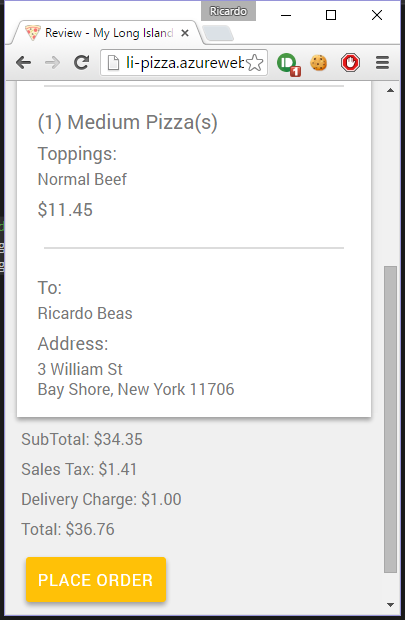
Mobile View – Checkout



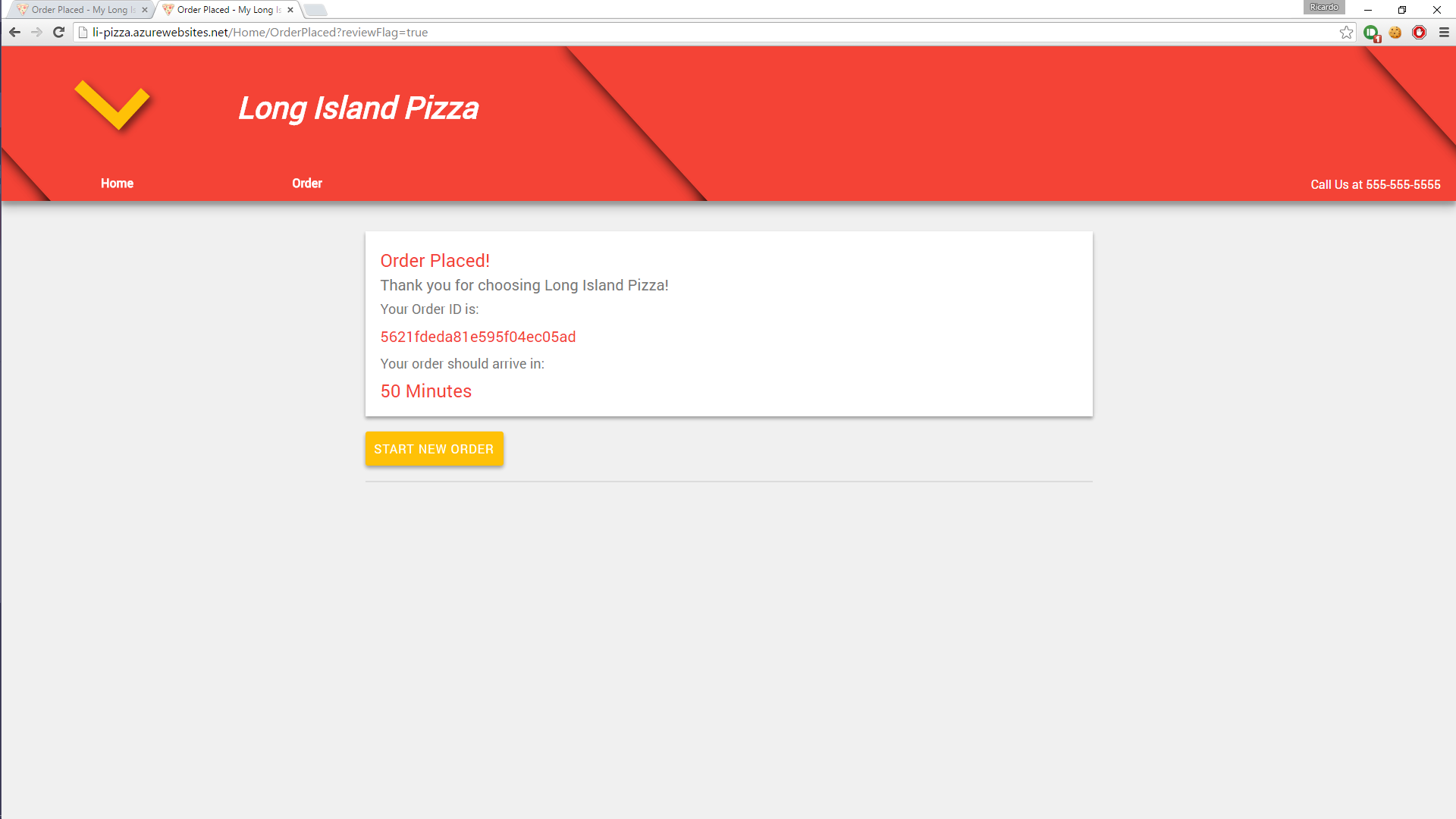
Desktop View – Review



Mobile View – Review



Desktop View – Order Placed



Mobile View – Order Placed

