**COMM644  
Final Project**

**Pizza Order Form**

In this project you will focus on creating a web form that allows the user to order a pizza online. Begin by creating an attractive online form complete with a background graphic and a logo in the upper left hand corner. Use Bootstrap to create the responsive design. Specifically, I would encourage you to use the Narrow Jumbotron example outlined here: <http://getbootstrap.com/examples/jumbotron-narrow/>. Ultimately, it’s up to you to decide the design. Aside from the background and logo, don’t spend too much time on the overall design…maybe 1 week of the 4 weeks that you have to complete this. The goal here is to focus on the front-end functionality of the form by utilizing HTML web forms, web form controls, and JavaScript. Once the design of the web page has been created, perform the following steps to complete your assignment:

**Part 1 – Delivery Location**

1. Create a multi-part form that allows the user to enter their personal information and to build a pizza for delivery online. Start with the personal information that the delivery man will use to deliver your pizza. Include fields for name, address type, street address, optional apartment number, suite, or room number, city, state, zip code, phone number, and email.
   1. The address type should be a drop down list that contains options for house, apartment, business, campus, hotel, dorm, or other.
   2. If the user selects other, an additional text box should programmatically appear that allows the user to freely enter the “other” address type. If anything other than “other” is selected, the text field should not be visible.
   3. Write a function to validate and ensure that the user enters a value (or makes a selection) for each of the form items here. Use regular expressions to ensure that their name doesn’t contain any numbers and that their zip code, phone number, and email address are formatted correctly. Make sure that the user only enters two alpha characters in the state field. Do not use the HTML5 “maxlength” or “required” attributes on any of the fields. I want you to learn how to require, limit, and validate inputs using pure JavaScript in this assignment.

**Part 2 – Build Your Order**

1. STEP 1: Add a radio button group that includes the 4 dough options listed here:
   1. Hand Tossed
   2. Thin Crust
   3. New York Style
   4. Gluten Free
2. STEP 2: Depending on which dough option the user selects, a drop down menu that contains the size and cost of the pizza will be programmatically populated based on the options below:
   1. If Hand Tossed Selected: Small ($9.99), Medium ($12.99), Large ($14.99)
   2. If Thin Crust Selected: Medium ($11.99), Large ($13.99)
   3. If New York Style Selected: Large ($16.99), Extra Large ($19.99)
   4. If Gluten Free Selected: Small ($10.99)

I want you to work with objects using literal notation in this step. Specifically the pizza sizes/prices should be stored in an object and then bound to the drop down list programmatically but only when the user selects a dough option first. Although the entire form (in this part) can be built this way, I want you to only focus on this technique for this step.

1. STEP 3: Add a drop down list for cheese options. Populate the list items with these options:
   1. Light: no charge
   2. Normal (default): no charge
   3. Extra: +$2.99
   4. Double: +$3.99
2. STEP 4: Add a drop down list for sauce options. Populate the list items with these options:
   1. Regular Tomato: no charge
   2. Hearty Tomato: +$.99
   3. BBQ Sauce: +$1.99
3. STEP 5: Next add a checkbox group of options for toppings. Each topping is $.99 extra. You offer the following toppings: pepperoni, sausage, ham, bacon, salami, peppers, olives, jalapenos, mushrooms, pineapple, and onion. Use a checkbox list here so the user can select more than one option.

**General Things to Consider**

1. Aside from validating the user input in Part 1, you will need to perform some validation tasks in the Build Your Order process. Specifically, you will need to write a function and associated event handler to validate and make sure that the user selects a dough style and a size first before proceeding. Do not let the user proceed to/select a cheese option, sauce, or topping option unless they’ve selected a dough style first. Since cheese and sauce options are drop down lists, you can simply select a default value here to guarantee selection. Since toppings are optional, a default selection here is also not necessary.
2. Keep a running total of the user’s order in the upper right hand corner somewhere. Make the font big enough so that it’s pronounced. Update this total as the user proceeds through the order building process. For every option that adds an additional charge, the total should change to reflect that addition (or subtraction) dynamically.
3. At the bottom of the form, add a “Finished Building Pizza” button. When this button is clicked, a confirmation box should appear asking the user if they’re sure that they’re done. If they select Cancel, the user should continue to be able to edit the forms in Parts 1 and 2. If the user selects OK, the Billing Information should appear (see below).

**Part 3 – Billing Information**

1. Add a series of fields similar to the fields you added in Part 1 so that the user can enter their full name, street address, optional apartment number, suite, or room number, city, state, and zip code. Add a checkbox with a label that reads “Same as delivery information”. When this checkbox is selected, the information from the Part 1 fields should be copied into these fields. You will need to add validation to these fields as well.
2. Add a series of fields for credit card information including credit card number, CVC code (three digit code), and expiration date (month (drop down list) and year (drop down list)). You’ll need to add validation to these fields as well including required validation for card number, CVC code, and expiration dates that are further out than the current date. Additionally, you’ll need to validate the credit card number (explained next).
3. This next step will be the most challenging part of the assignment. You will need to validate the credit card number based on a couple of different criteria. Those criteria are outlined below:

**Credit Card Number Validation Rules**

|  |  |  |  |
| --- | --- | --- | --- |
| **Card Type** | **Card Number Prefix** | **Number of Digits (Length)** | **Checksum Digit \*** |
| Visa | 4 | 13 or 16 | 10 |
| MasterCard | 51, 52, 53, 54, or 55 | 16 | 10 |
| American Express | 37 | 15 | 10 |

For almost all credit cards, the Luhn Formula is used to determine whether or not the credit card number is even valid. This algorithm generates a single digit which is then used as the last digit of the card number. By performing the same calculations, you can determine if the number matches the checksum digit and is therefore valid.

**\* The Luhn Formula Explained**

1. Double the value of alternate digits beginning with the second to last digit from the right. Or, reverse the string and begin with second digit from the left.
2. Each doubled value becomes individual digits (16 becomes a 1 and a 6)
3. Add the individual digits in step 1 to each of the values of the other digits
4. Divide the total by the checksum digit (10).
5. If the checksum digit divides evenly into the total, the card number is valid

Examples (Visa):

**4**5**1**2**1**1**3**0**1**4**8** 4**3**2**5** 2  
852221602416462102  
8+5+2+2+2+1+6+0+2+4+1+6+4+6+2+1+0+2

Sum = 54

54/10 leaves a remainder. Card number is invalid.

**4**5**1**2**1**1**3**0**1**4**6** 4**3**2**5** 2

852221602412462102  
8+5+2+2+2+1+6+0+2+4+1+2+4+6+2+1+0+2

Sum=50

50/10 leaves no remainder. Card number is valid.

Notes:

1. If the card number is invalid, display that error next to the card number field. Allow the user to reenter.
2. Validate the card number so that an error appears if the user enters anything other than numeric.
3. Validate based on the number of digits first before you perform checksum validation.
4. Display the card type to the user based on the entered card number prefix.