Personal Website

For this assignment you will create a Personal website using .NET MVC.

# User StorIES:

As a programmer I would like to post my resume online so that prospective employers and customers can see it.

As a programmer I would like to provide prospective employers and customers the ability to contact me.

As a programmer I would like to provide portfolio pages that show samples of my work.

As a programmer I would like to provide prospective customers the ability to request a bid.

As a programmer I would like to provide prospective customers the ability to pay me for completed work.

As a programmer I would like to provide a real time chat application on my site that customers can use to communicate with me.

# DETAILED REQUIREMENTS:

## Style

* Your site should have at least a unique color scheme and employ a background image and/or logo to make it your own. It must not look like the default project.
* Make sure you change all default template items like the copyright footer and the Application name.

## authorization authentication

* Your site must provide the capability to log in users
* Your site must provide two roles: customer and admin
* Use configuration.cs seed method to initialize roles, create a starting admin account and provide a starting password that will be changed once the site is started.

## Resume

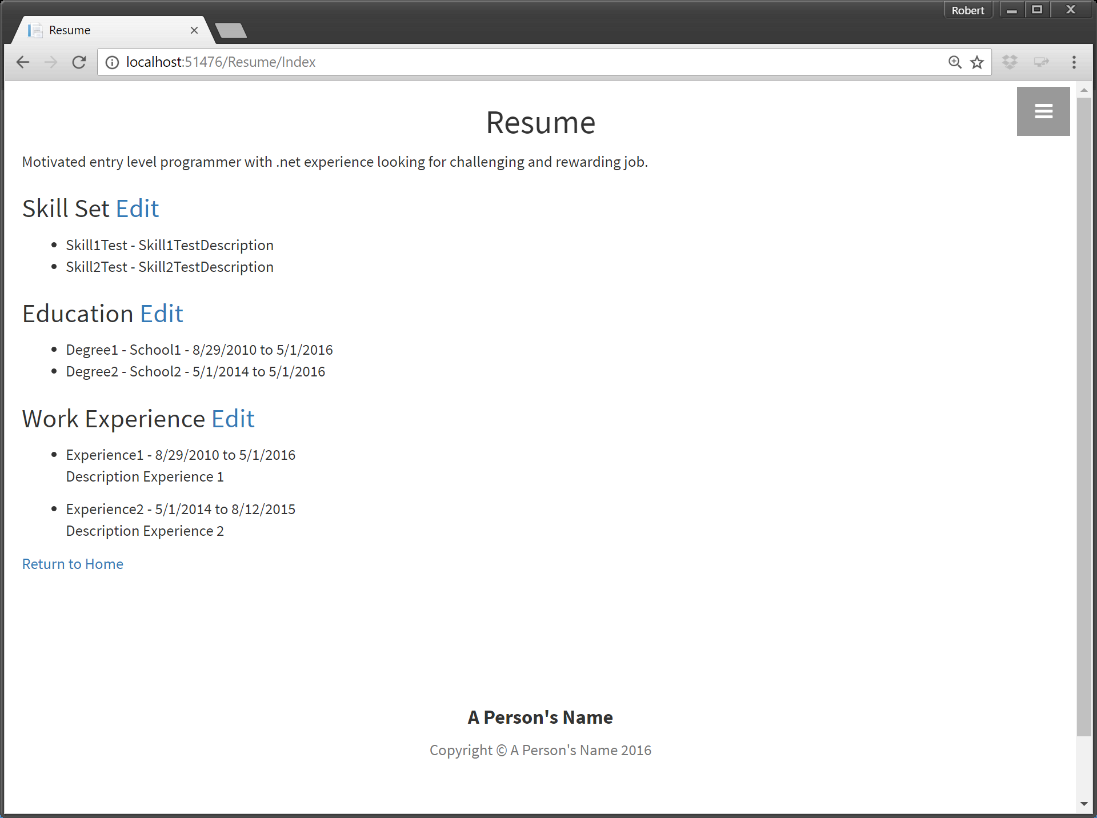
* You website must present your resume. Instead of just showing a static HTML resume page, you must store information for your resume in at least three domain models. You can pick the categories but should be something like Education, Work Experience and Technical Skills.



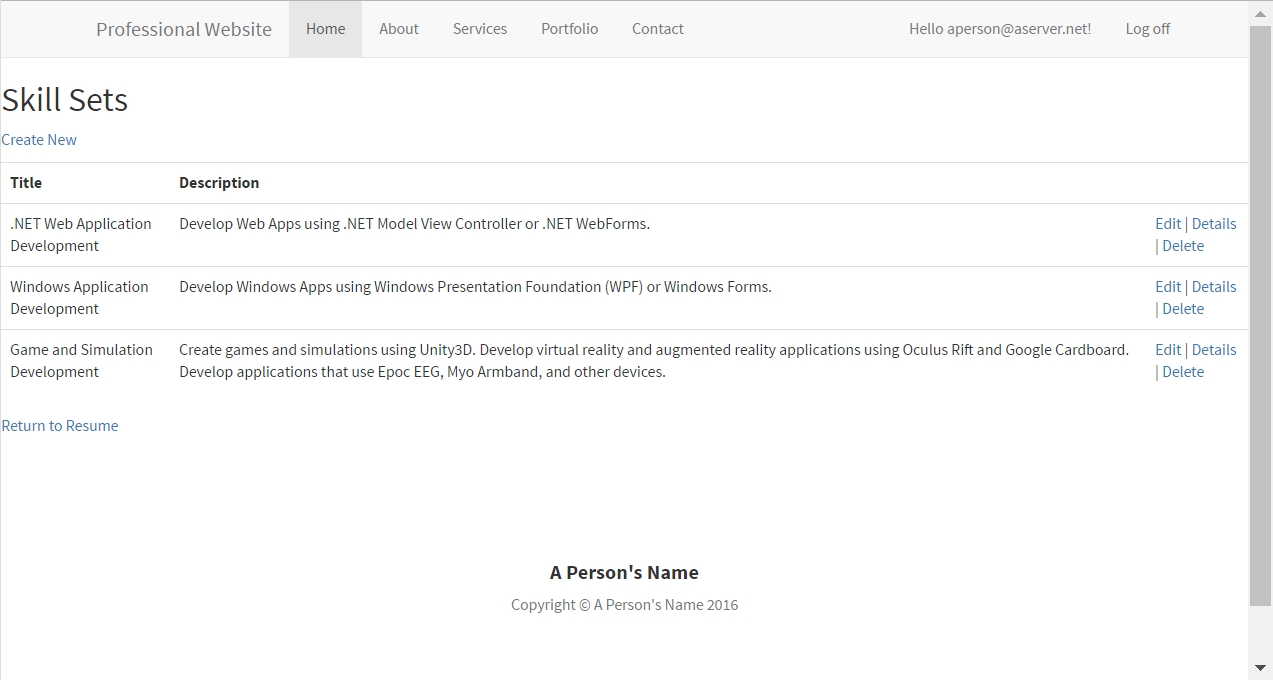
* You must use a View Model for the information your resume action controller method will pass to the resume view. The View Model will contain Lists to hold your three categories of information. In addition, provide string properties, as needed to pass in any preamble and other information.

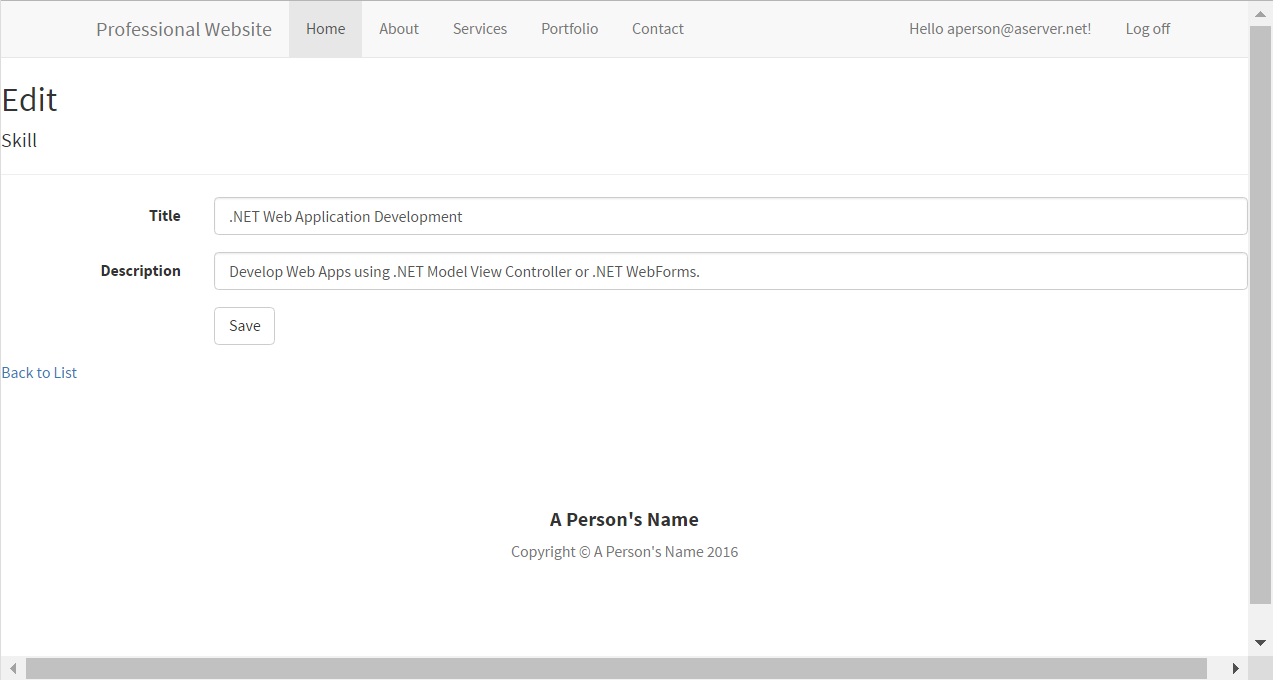


* Use this view model to create a view to show your resume:



* Provide the capability to edit the three tables of data. Make sure you set authorization attributes so only a person with admin privileges can access the edit forms.



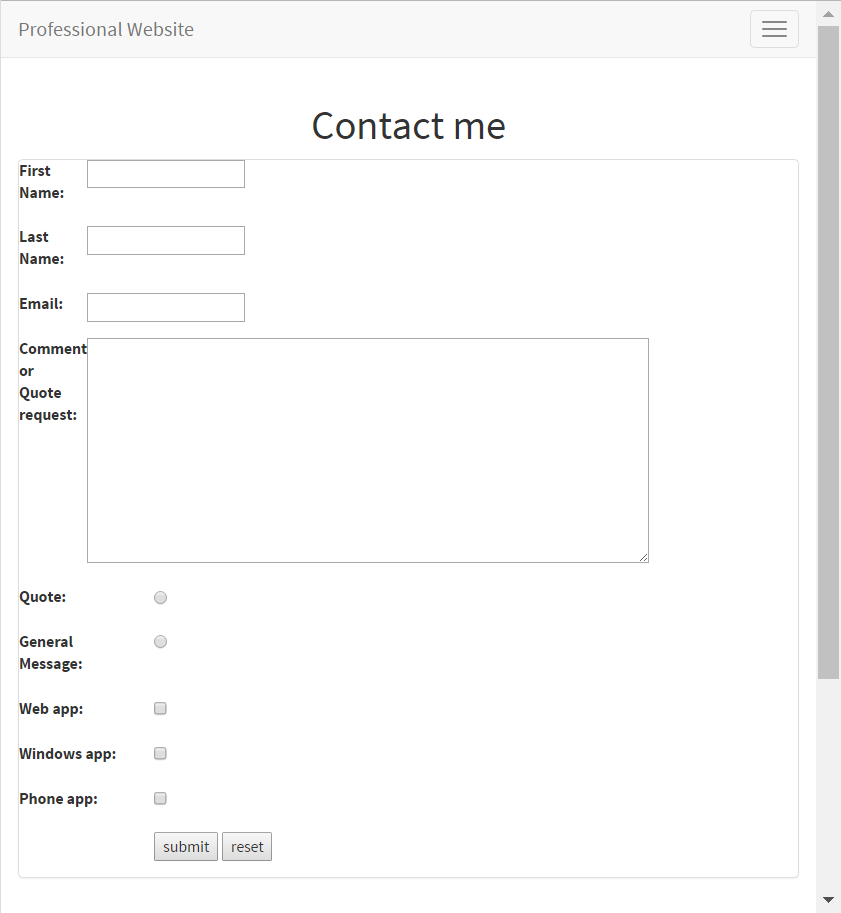


## Portfolio Pages

* Your site should have at least three portfolio pages.
* Page is a relatively static page with at least one image tag showing a picture.
* View will just have text explaining your portfolio item and an image that can be a screen capture of the application you are describing.
* You don’t have to write a book. In fact you can just place some Lorem Ipsum text as a place holder. See http://www.lipsum.com/.

## Contact Page

* Provide a contact page.



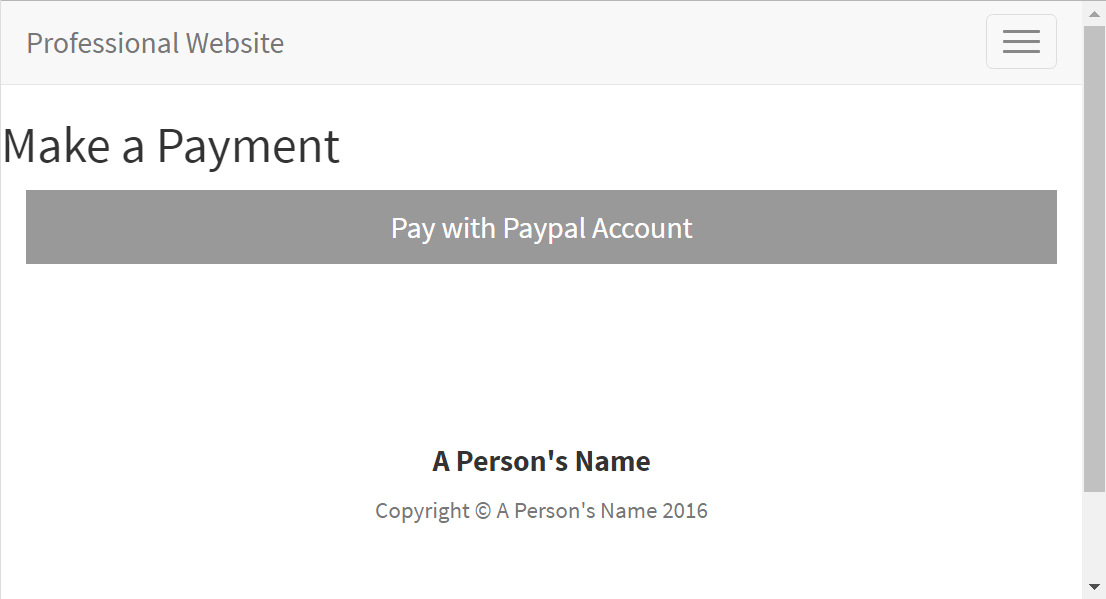
* Contact page must not display your email address directly.
* Contact model must have fields for:
  + ContactInfoID
  + FirstName
  + LastName
  + Email
  + Comments
  + Booleans for is requesting quote, is job offer, is general contact, that will be selected using a radio buttons
  + Booleans for different types of development (examples: Web App, Phone App, Windows App) that will be selected using checkboxes pick options you are capable of doing now.
* Example of UML:

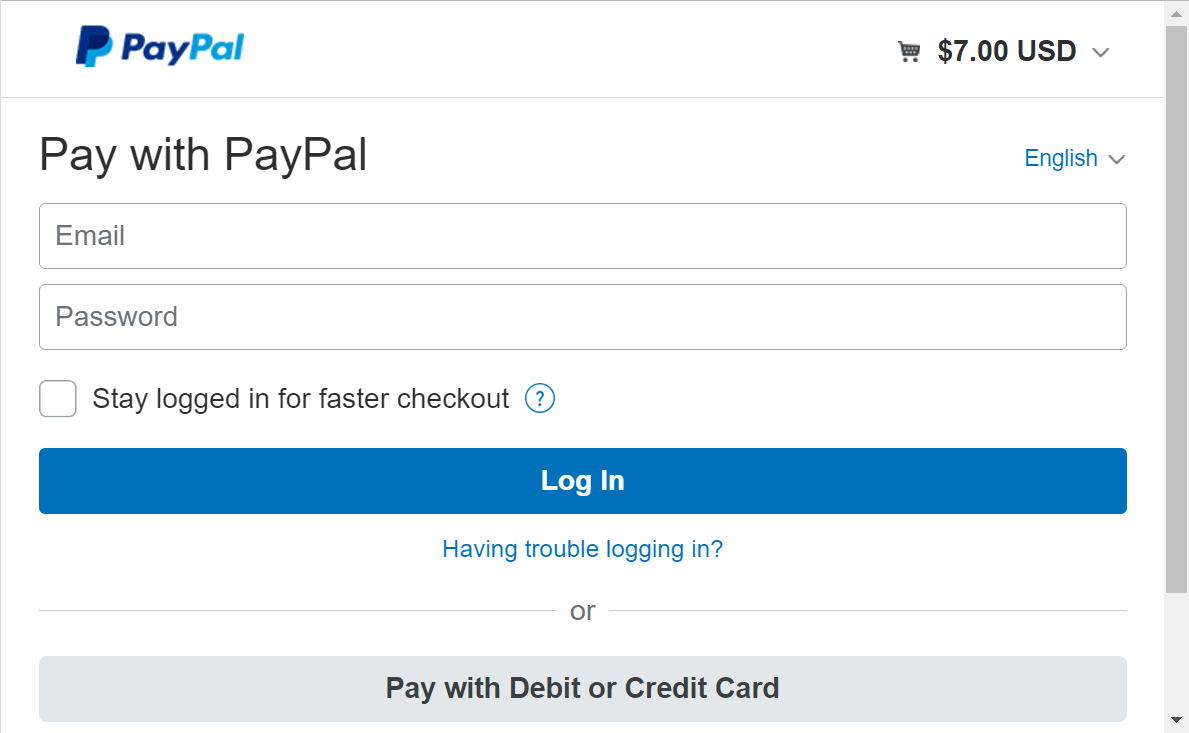


* When user clicks submit button and email should be sent to your email address and the contact should be saved to the database.
* See this blog page for information on how to do this: <https://wordpress.com/post/robgarnerblog.wordpress.com/373>

## Payment Page

* Your site must provide the capability to receive payment in some way.





* See the article at: <https://robgarnerblog.wordpress.com/2016/09/18/adding-paypal-payment-page-to-a-website/>
* The tutorial at <http://www.codeproject.com/Articles/870870/Using-Paypal-Rest-API-with-ASP-NET-MVC> describes a way to set up both direct credit card and paypal payments. You may set up just the paypal functionality.
* The tutorial works using the paypal sandbox. The sandbox does not actually process payments and so is the best way to test the functionality.
* If you do not feel comfortable setting up a paypal account just insert the code into your application and let me know.
* I am just going to look for basic functionality. Don’t get too complex with this. Example: You could create a complex invoicing system. For first iteration just provide ability to pay a static amount is ok.

## Controls

* Your site must incorporate Checkboxes. Options would be in your contact page to have the user specify whether they are requesting a quote or just making contact or in your resume pages to set whether an item should be displayed or not.
* Your site must incorporate radio buttons. An option could include placing radio buttons in your contact page to let user specify type of development wanted.
* See <http://www.w3schools.com/bootstrap/bootstrap_forms_inputs.asp> for how to make checkboxes and radio buttons.
* Your site must incorporate a date picker somewhere on the site. A good place to do this would be in your resume pages where you set the start and end dates for education and/or experience. See <https://www.asp.net/mvc/overview/older-versions/using-the-html5-and-jquery-ui-datepicker-popup-calendar-with-aspnet-mvc/using-the-html5-and-jquery-ui-datepicker-popup-calendar-with-aspnet-mvc-part-4> for how to do this.

## Blog

* Add a blog to your site. There are a couple options for this.
  + Create your own from scratch. It should at least do the following: Provide capability to store articles in your database, edit and modify them online from the admin account. Provide a way for the user to view and navigate through them. An article on doing this is at <http://prideparrot.com/blog/archive/2012/12/how_to_create_a_simple_blog_part1>. Note: this uses alternate technologies to Entity. I recommend just using it as a guide.
  + Another option is to find and incorporate a blog framework into your site. These can be somewhat complex to incorporate. Examples include:
    - <https://github.com/ChrisFulstow/NBlog>
    - <http://www.dotnetblogengine.net/>

## Chat page

* Add a chat page to your site.
* Chat page should allow you to speak real time with someone using Signal-R

# Test Criteria (Not all inclusive):

* Properly comment code (see paper on commenting code)
* Concise efficient code
* All controls properly labeled
* Site clearly explains to user what values to input, what format and what units to input
* No spelling/grammatical errors
* All errors handled
* Correct answers returned for all relevant test cases