

# MVC ASP.NET Identity customizing for adding profile Parameters to User Model

## Blog Project: Adding LastName, FirstName, Photo to User

In this project we will see in detail about using ASP.NET Identity in MVC Application;

1) Add new Parameters such as LastName, FirstName, and Photo to AspNetUsers sign-in.

2.)To upload and store User Profile Image to AspNetUsers table in SQL Server by using FileInfo, FileStream, and BinaryReader methods to store the image as a byte data type.

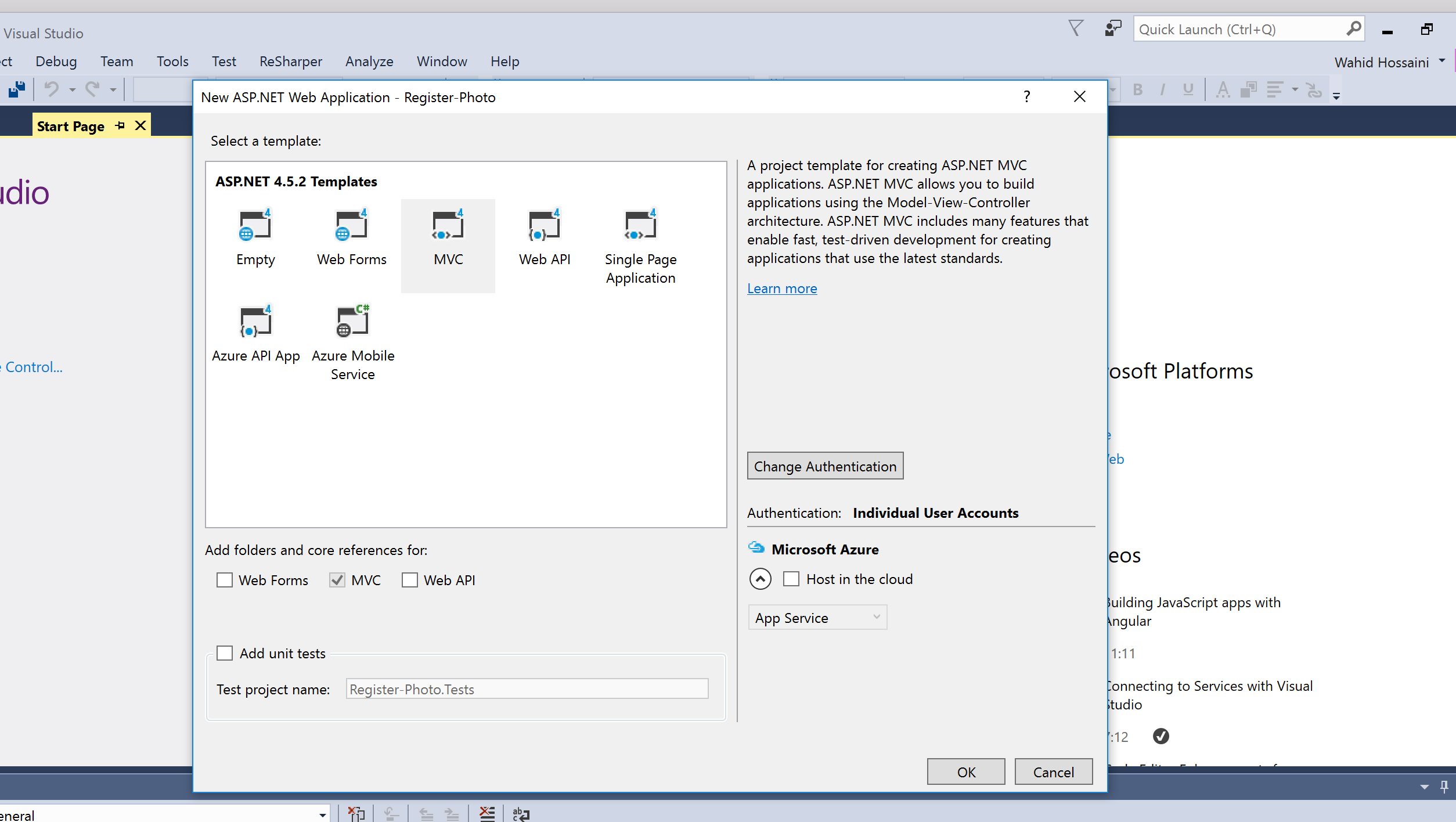
3.) Display the authenticated Logged in users Uploaded profile Image in home page and in Title bar.

Prerequisites

Visual Studio 2015 and a MVC new project with individual Sign-in Account.

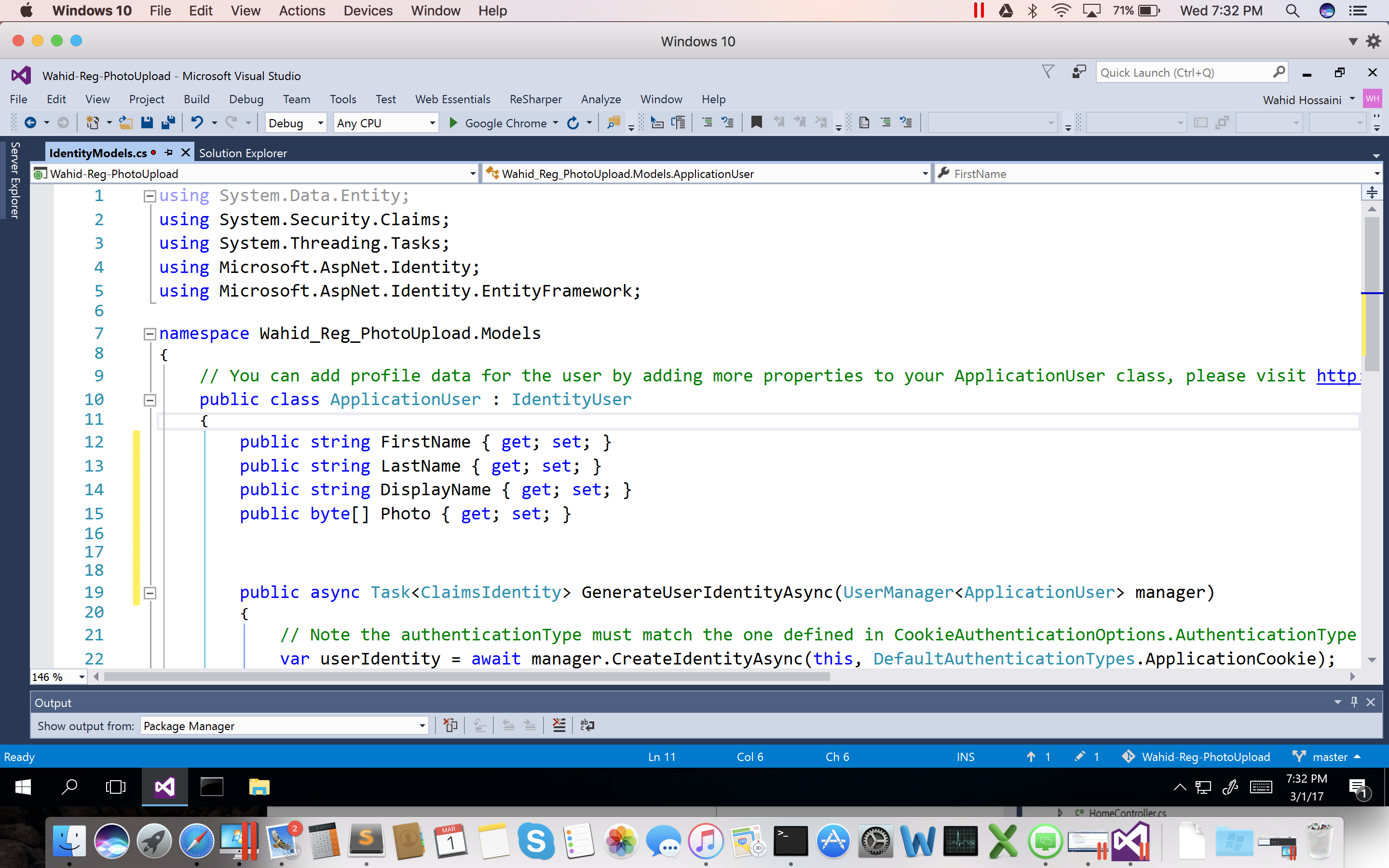
**Step 1: Creating New Project**

After installing our Visual Studio 2015 click Start, then Programs and select Visual Studio 2015 - Click Visual Studio 2015. Click New, then Project, select Web and then select ASP.NET Web Application. Enter your project name and click OK. Select MVC and click OK. Please see figure below.



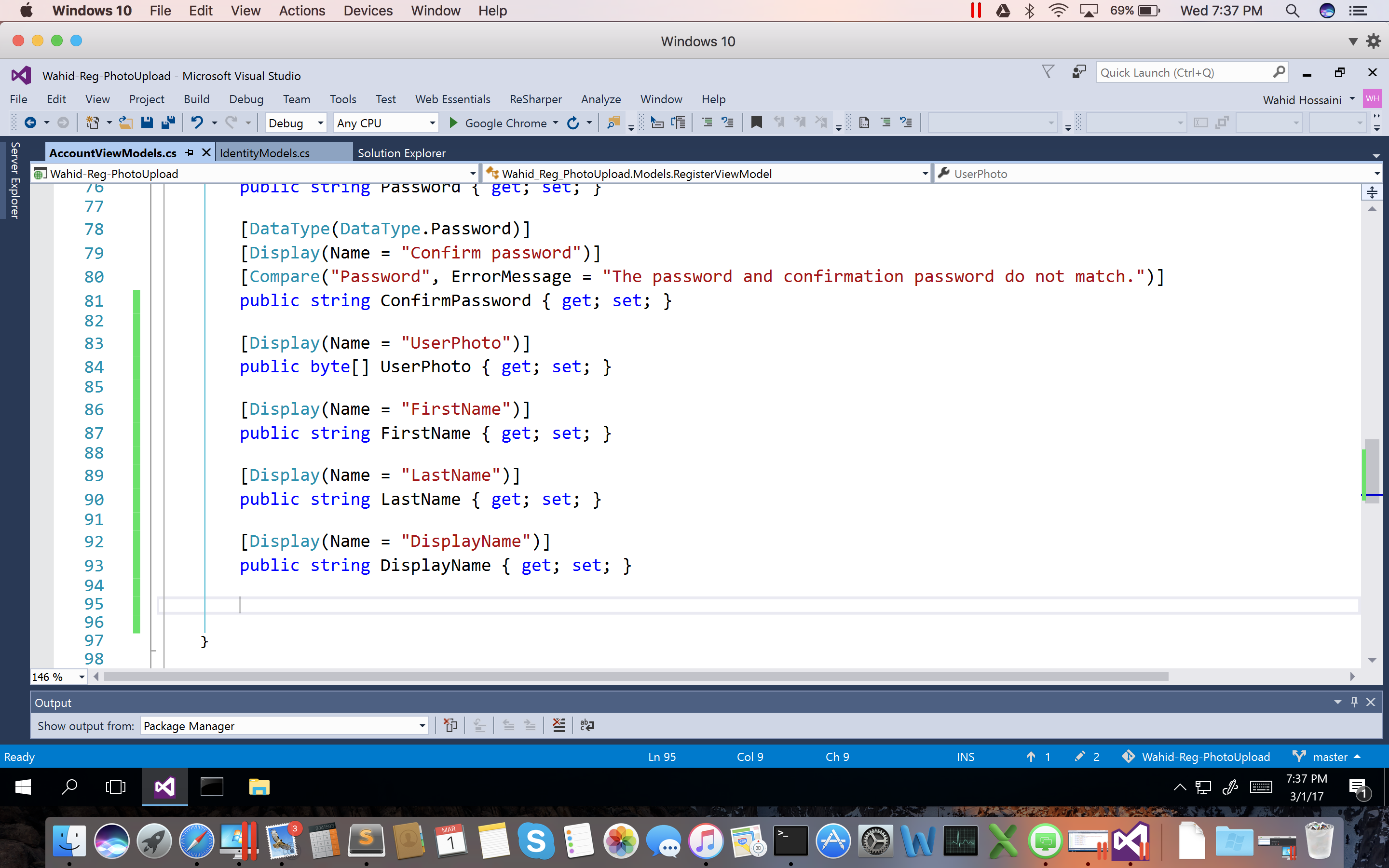
**Step 2: Go to Model folder and find the IdentityModels.cs**

In IdentityModels.cs we need to add the LastName, FirstName and image properties to be used for storing our information to database. In ApplicationUser class we will be adding constructors for the new properties.



**Step 3: Find in Model Folder the AccountViewModel Class:**

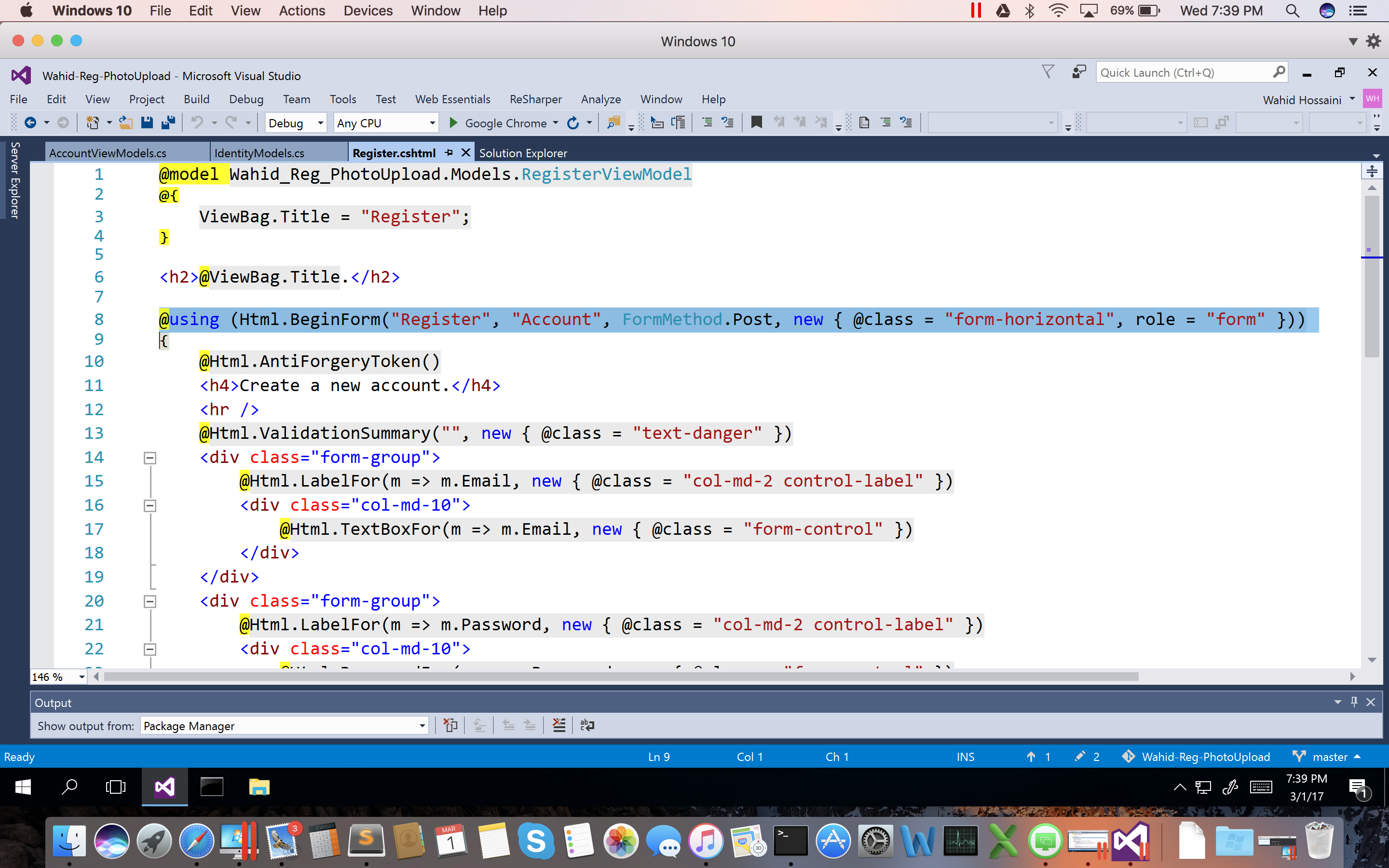
Next in **AccountViewModel.cs** check for the RegisterViewModel and add the properties like below.



**Step 4: Edit Register view to add our New Properties**

Go to Views Folder and in the Accounts find the file named: Register.cshtml.

First we add enctype = "multipart/form-data" to the Html.BeginForm like below then we add the below code to upload FirstName,LastName and images to AspNetUsers table in our DB.



For the Photo Upload:

**<div** class="form-group"**>**

        @Html.LabelFor(m => m.UserPhoto, new { @class = "col-md-2 control-label" })

**<div** class="col-md-10"**>**

**<input** type="file" name="UserPhoto" id="fileUpload" accept=".png,.jpg,.jpeg,.gif,.tif" **/>**

**</div>**

**</div>**

For LastName,FirstName,DisplayName:

<div class="form-group">

@Html.LabelFor(m => m.FirstName, new { @class = "col-md-2 control-label" })

<div class="col-md-10">

@Html.TextBoxFor(m => m.FirstName, new { @class = "form-control" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(m => m.LastName, new { @class = "col-md-2 control-label" })

<div class="col-md-10">

@Html.TextBoxFor(m => m.LastName, new { @class = "form-control" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(m => m.DisplayName, new { @class = "col-md-2 control-label" })

<div class="col-md-10">

@Html.TextBoxFor(m => m.DisplayName, new { @class = "form-control" })

</div>

</div>

**Step 5: MVC Controller Part**

Next in**AccountController.cs we will update the code in** Register **post method to customize and store the uploaded user image** in ASP.NET identity database.

In the Register post method we will save the uploaded image to the byte array and use this byte array result to be saved in our users table.

[HttpPost]

        [AllowAnonymous]

        [ValidateAntiForgeryToken]

**public** async Task<ActionResult> Register([Bind(Exclude = "UserPhoto")]RegisterViewModel

model)

        {

**if** (ModelState.IsValid)

            {

                // To convert the user uploaded Photo as Byte Array before save to DB

**byte**[] imageData = **null**;

**if** (Request.Files.Count > 0)

                {

                    HttpPostedFileBase poImgFile = Request.Files["UserPhoto"];

**using** (var binary = **new** BinaryReader(poImgFile.InputStream))

                    {

                        imageData = binary.ReadBytes(poImgFile.ContentLength);

                    }

                }

var user = **new** ApplicationUser { UserName = model.Email, Email = model.Email };

                //Here we pass the byte array to user context to store in db

                user.UserPhoto = imageData;

                var result = await UserManager.CreateAsync(user, model.Password);

**if** (result.Succeeded)

                {

                    await SignInManager.SignInAsync(user, isPersistent:**false**, rememberBrowser:**false**);

                    // For more information on how to enable account confirmation and password reset please visit http://go.microsoft.com/fwlink/?LinkID=320771

**return** RedirectToAction("Index", "Home");

                }

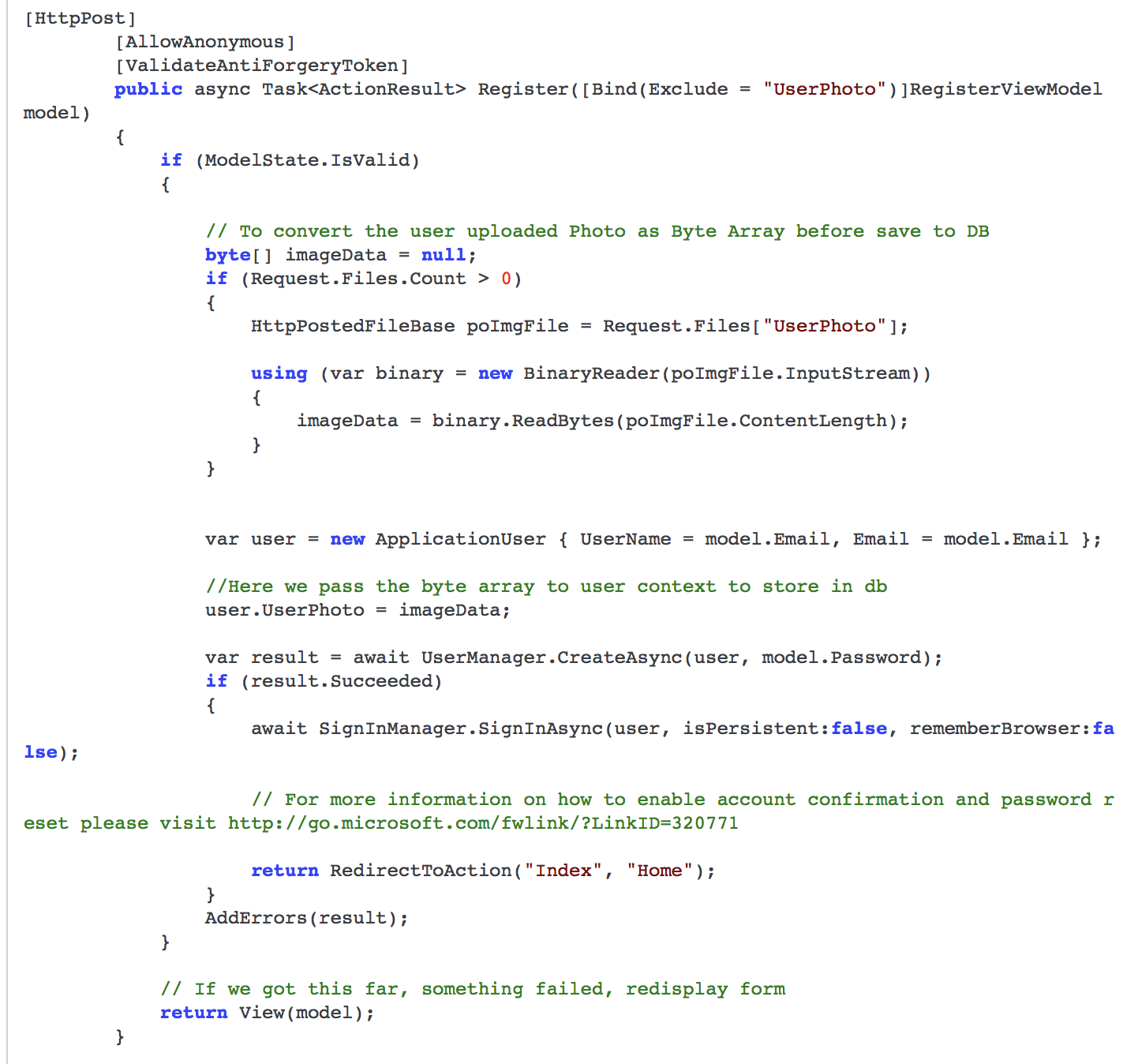
                AddErrors(result);

            }

            // If we got this far, something failed, redisplay form

**return** View(model);

        }



**Step 6: Display User Image in home page**

So now we have successfully completed the user properties and image uploaded part to AspNetUsers Table in our local SQL Server Database.

Next we will see how to display the logged in user Image on home page and in menu bar.

For displaying this we create a FileContentResult Method  to display the image on user home and in menu bar.

Go to the HomeController and Create FileContentResult method as UserPhotos to display the image in home page and in Menu bar.



In home controller we create a method named as UserPhotos and return the image to View page for user profile display.

In this method we check for Authenticated (Logged in) users. If the user is not logged in to our web application then I will display he default image as “coderfoundry.png”. Here we display the image in both top menu and in home page.

If the user is authenticated and successfully logged in to our system then we display the logged in user profile picture in home page.

**Step 7: MVC View Part**

In Home Index View we write the below code to display our logged in users profile picture.

**<h1>**Welcome to Coder Foundry..

**<img** src="@Url.Action("UserPhotos", "Home" )"  style="width:160px;height:160px; background: #FFFFFF;

    margin: auto;

    -moz-border-radius: 60px;

    border-radius: 100px;

    padding: 6px;

    box-shadow: 0px 0px 20px #888;" **/>**

**</h1>**

**\_Layout.cshtml or \_LoginPartial.cshtml**

you could add the image to the Navigation Bar either to the right of Controller buttons or to the Sign-in name by placing it in either of the two files.

**<ul** class="nav navbar-nav"**>**

**<li>**

**<img** src="@Url.Action("UserPhotos", "Home" )" height="48" width="48" **/>**

**Step 8: Run the Application**

So now we have completed both upload and display part. Let’s start run our application and register new user with FirstName, LastName, and Image and check for output.

Source Code and full Project can be found on Nuget Package Library download by the name:

**Wahid-Reg-Upload**