MILESTONE TWO

WEB601

Rebekah Rossiter

Contents

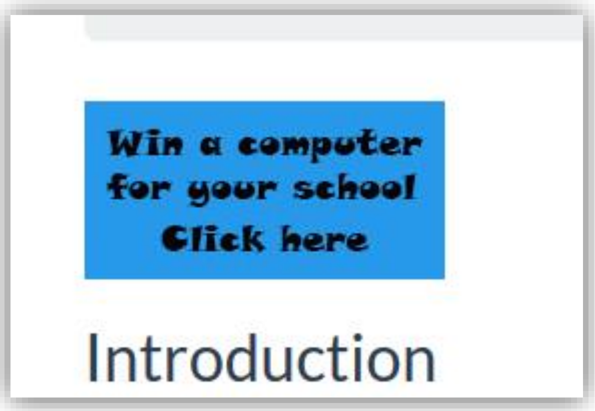
Introduction

Around the world is a darts game that is mainly used for practice and aiming purposes, but can be a fun game if played with another person. This is what my website is about. The purpose of this report is to show my understanding of the processes that I have undertaken in creating this website. As the project is indicative this is the second step of my work. This has been requested by Todd Cochrane and is due on Friday 24th October at 9am.

Computer Competition

Access

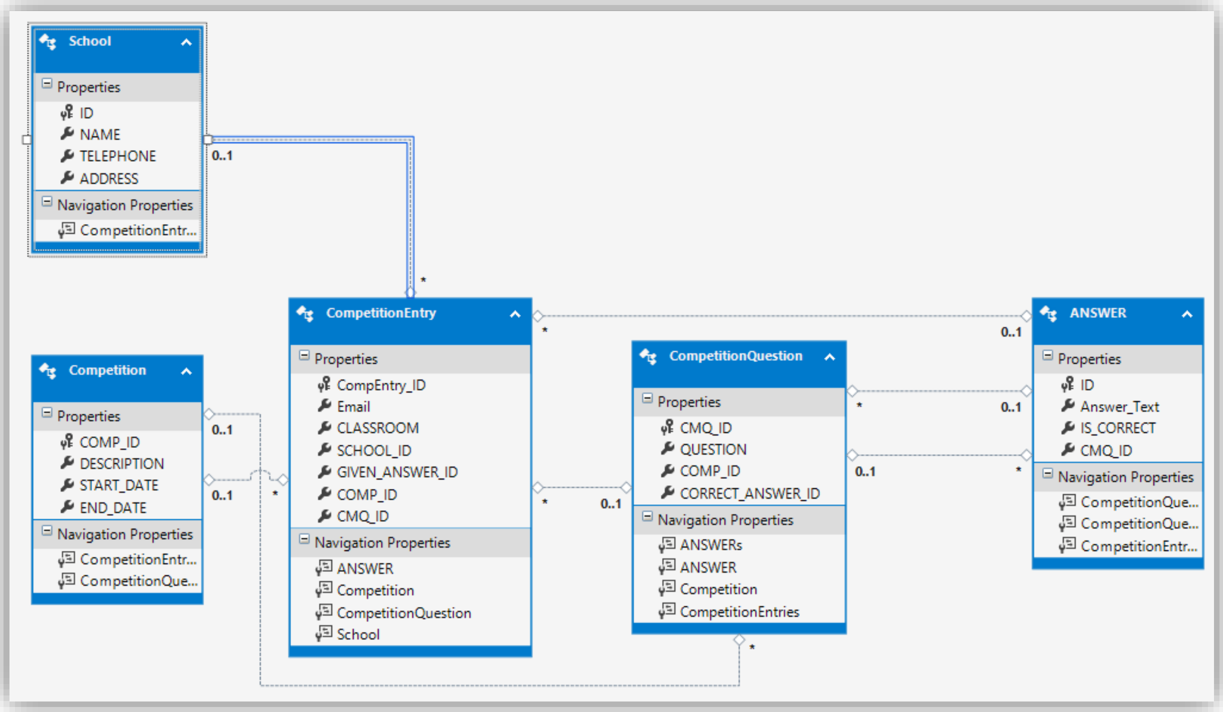
To access the competition page, we were set the task of creating some sort of graphic to act as a link to the page. My first step was to create the graphic. I made it very simple and created it in Paint. Below is a screenshot of the image that I made. As you can see I am not a 'graphics' designer.



Database

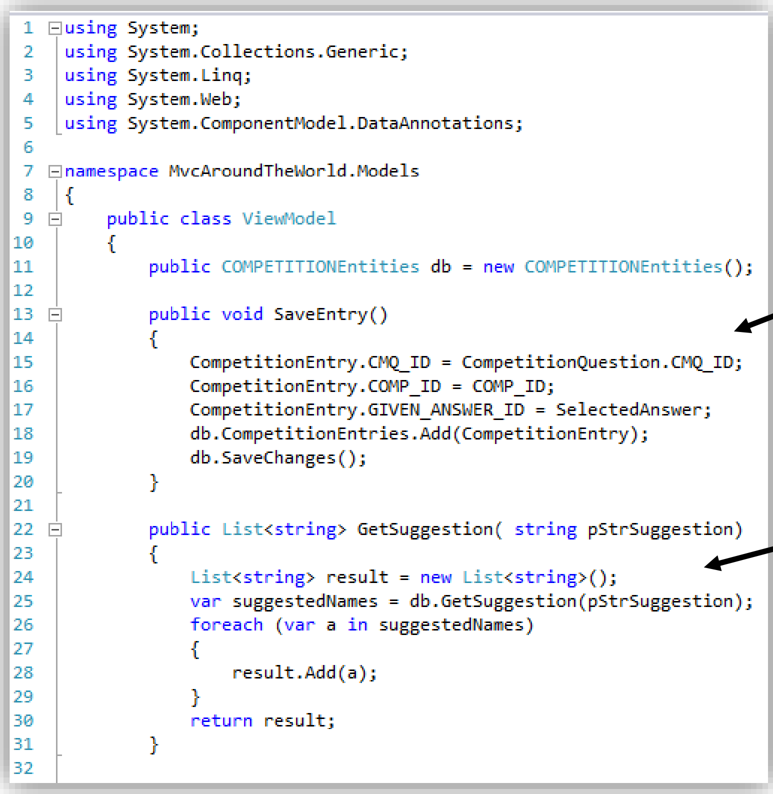
Next I needed to create a database for the competition using SQL Server. The database is called COMPETITION. I ended up using the one that Todd provided the class. I did have one that was working but my setup was a bit more complicated and his one seemed to be a bit easier to use for this project.

In Visual Studio I had to set up a data connection with the database and called it COMPETITIONEntities. I then had to create a model to be able to use the database. I just left it with its default name Model1. This is what was automatically created:



ViewModel

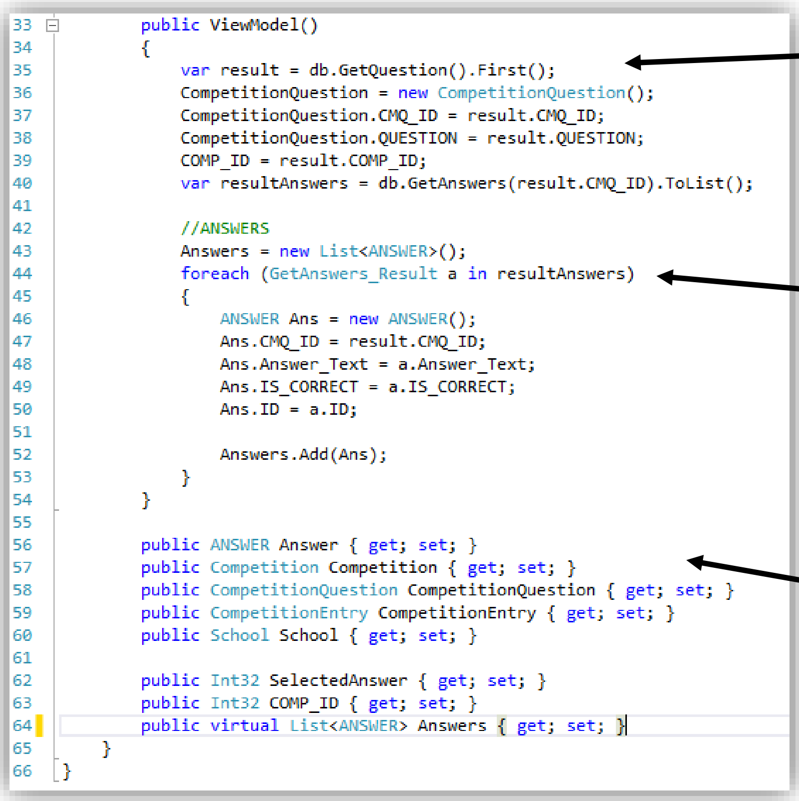
The next step was to create a new model that would use Model1 with any appropriate views. The code in ViewModel was again supplied by Todd. I had my own code but seeing as I decided to use his database it seemed appropriate to use his code and just make suitable changes.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;

namespace MvcAroundTheWorld.Models
{
    public class ViewModel
    {
        public COMPETITIONEntities db = new COMPETITIONEntities();

        public void SaveEntry()
        {
            CompetitionEntry.CMQ_ID = CompetitionQuestion.CMQ_ID;
            CompetitionEntry.COMP_ID = COMP_ID;
            CompetitionEntry.GIVEN_ANSWER_ID = SelectedAnswer;
            db.CompetitionEntries.Add(CompetitionEntry);
            db.SaveChanges();
        }

        public List<string> GetSuggestion( string pStrSuggestion)
        {
            List<string> result = new List<string>();
            var suggestedNames = db.GetSuggestion(pStrSuggestion);
            foreach (var a in suggestedNames)
            {
                result.Add(a);
            }
            return result;
        }

```

This void matches the contents of the page to database and saves it.

This list contains and adds suggestions for the page.



```
        public ViewModel()
        {
            var result = db.GetQuestion().First();
            CompetitionQuestion = new CompetitionQuestion();
            CompetitionQuestion.CMQ_ID = result.CMQ_ID;
            CompetitionQuestion.QUESTION = result.QUESTION;
            COMP_ID = result.COMP_ID;
            var resultAnswers = db.GetAnswers(result.CMQ_ID).ToList();

            //ANSWERS
            Answers = new List<ANSWER>();
            foreach (GetAnswers_Result a in resultAnswers)
            {
                ANSWER Ans = new ANSWER();
                Ans.CMQ_ID = result.CMQ_ID;
                Ans.Answer_Text = a.Answer_Text;
                Ans.IS_CORRECT = a.IS_CORRECT;
                Ans.ID = a.ID;

                Answers.Add(Ans);
            }
        }

        public ANSWER Answer { get; set; }
        public Competition Competition { get; set; }
        public CompetitionQuestion CompetitionQuestion { get; set; }
        public CompetitionEntry CompetitionEntry { get; set; }
        public School School { get; set; }

        public Int32 SelectedAnswer { get; set; }
        public Int32 COMP_ID { get; set; }
        public virtual List<ANSWER> Answers { get; set; }
    }
}
```

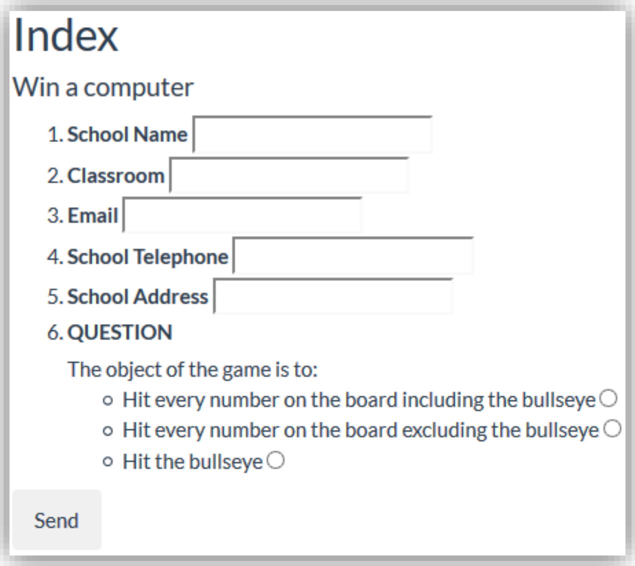
This create a variable called result and gets the first question from the database. It then matches the details to result.

Here it is creating a new list called Answers. It then starts a loop. Inside that loop it goes and does the matching. Then it adds the answer.

Here is the getting and setting of the variables

Competition Entry Page

The next step was to create an index page for the competition. It had to contain four text fields, a question and three answers with radio buttons. To do this I had to create a new view. It is named Index inside a folder called ViewModel.



Index

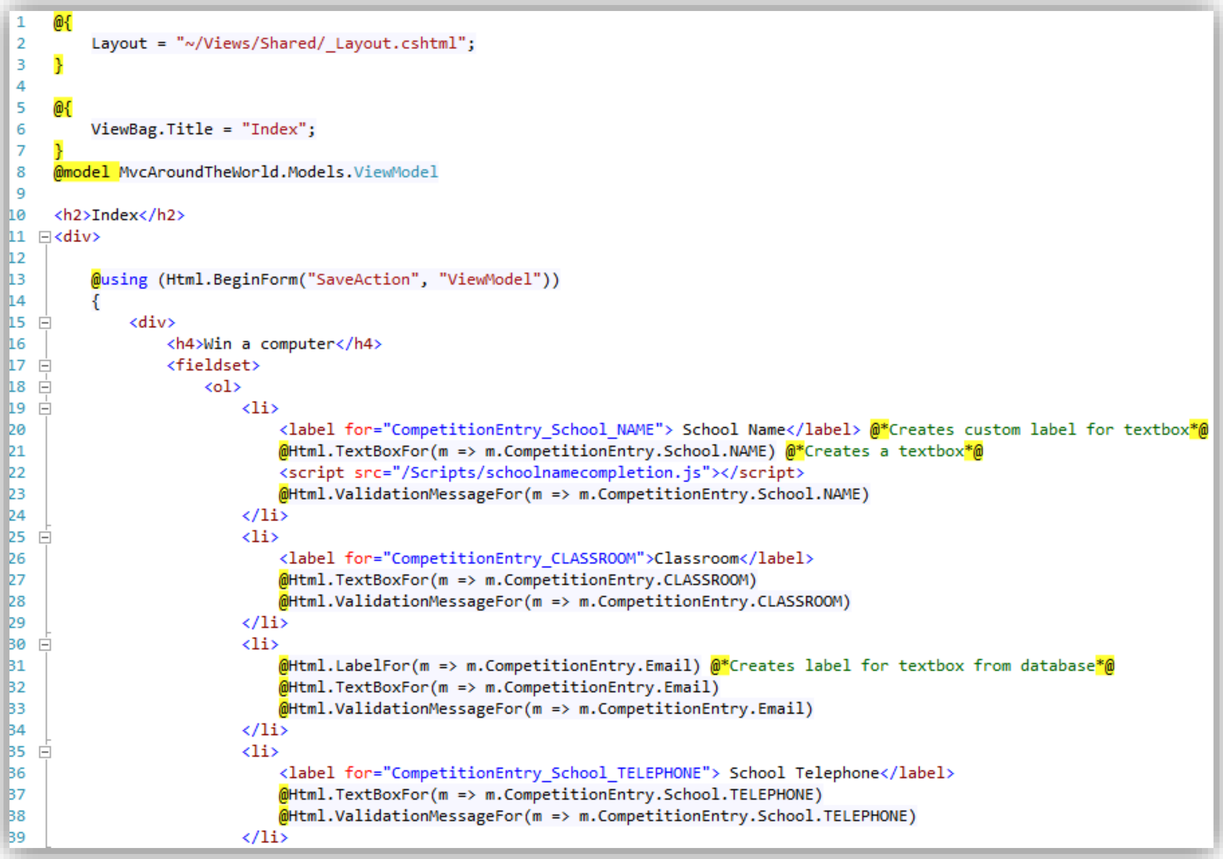
Win a computer

1. School Name
2. Classroom
3. Email
4. School Telephone
5. School Address
6. QUESTION

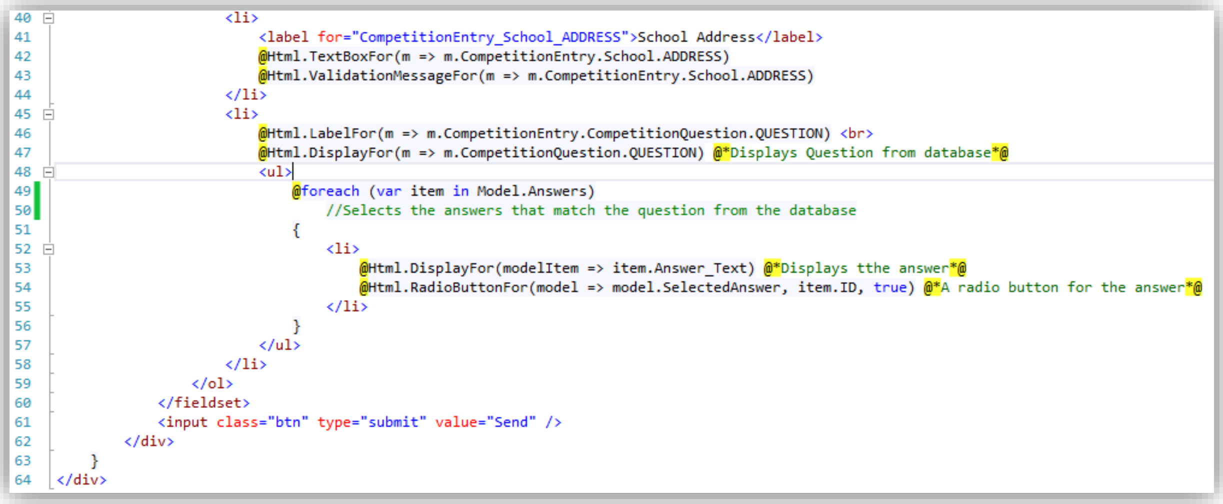
The object of the game is to:

- Hit every number on the board including the bullseye ○
- Hit every number on the board excluding the bullseye ○
- Hit the bullseye ○

Send



```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}

@{
    ViewBag.Title = "Index";
}
@model MvcAroundTheWorld.Models.ViewModel

<h2>Index</h2>
<div>

    @using (Html.BeginForm("SaveAction", "ViewModel"))
    {
        <div>
            <h4>Win a computer</h4>
            <fieldset>
                <ol>
                    <li>
                        <label for="CompetitionEntry_School_NAME"> School Name</label> @*Creates custom label for textbox*@
                        @Html.TextBoxFor(m => m.CompetitionEntry.School.NAME) @*Creates a textbox*@
                        <script src="/Scripts/schoolnamecompletion.js"></script>
                        @Html.ValidationMessageFor(m => m.CompetitionEntry.School.NAME)
                    </li>
                    <li>
                        <label for="CompetitionEntry_CLASSROOM">Classroom</label>
                        @Html.TextBoxFor(m => m.CompetitionEntry.CLASSROOM)
                        @Html.ValidationMessageFor(m => m.CompetitionEntry.CLASSROOM)
                    </li>
                    <li>
                        @Html.LabelFor(m => m.CompetitionEntry.Email) @*Creates label for textbox from database*@
                        @Html.TextBoxFor(m => m.CompetitionEntry.Email)
                        @Html.ValidationMessageFor(m => m.CompetitionEntry.Email)
                    </li>
                    <li>
                        <label for="CompetitionEntry_School_TELEPHONE"> School Telephone</label>
                        @Html.TextBoxFor(m => m.CompetitionEntry.School.TELEPHONE)
                        @Html.ValidationMessageFor(m => m.CompetitionEntry.School.TELEPHONE)
                    </li>
```

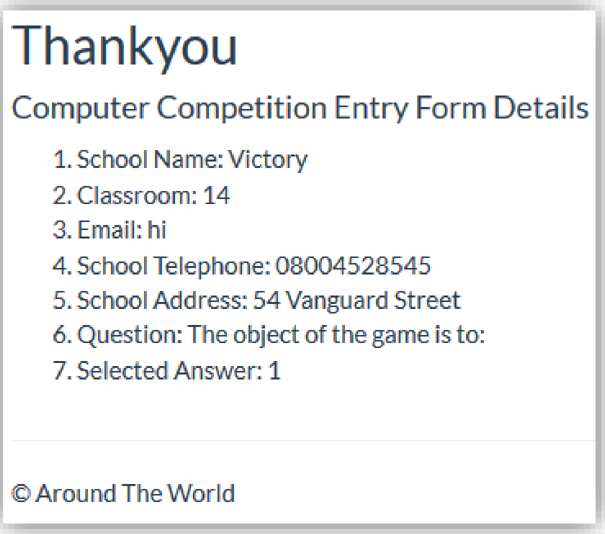


```
                    <li>
                        <label for="CompetitionEntry_School_ADDRESS">School Address</label>
                        @Html.TextBoxFor(m => m.CompetitionEntry.School.ADDRESS)
                        @Html.ValidationMessageFor(m => m.CompetitionEntry.School.ADDRESS)
                    </li>
                    <li>
                        @Html.LabelFor(m => m.CompetitionEntry.CompetitionQuestion.QUESTION) <br>
                        @Html.DisplayFor(m => m.CompetitionQuestion.QUESTION) @*Displays Question from database*@
                        <ul>
                            @foreach (var item in Model.Answers)
                                //Selects the answers that match the question from the database
                            {
                                <li>
                                    @Html.DisplayFor(modelItem => item.Answer_Text) @*Displays tthe answer*@
                                    @Html.RadioButtonFor(model => model.SelectedAnswer, item.ID, true) @*A radio button for the answer*@
                                </li>
                            }
                        </ul>
                    </li>
                </ol>
            </fieldset>
            <input class="btn" type="submit" value="Send" />
        </div>
    }
</div>
```

The begin form is creating the form on the page. Each list item is a ‘field’ on the form.

Thank You Page

Next to be created was a thank you page. This page was required to display what was entered on the previous page.

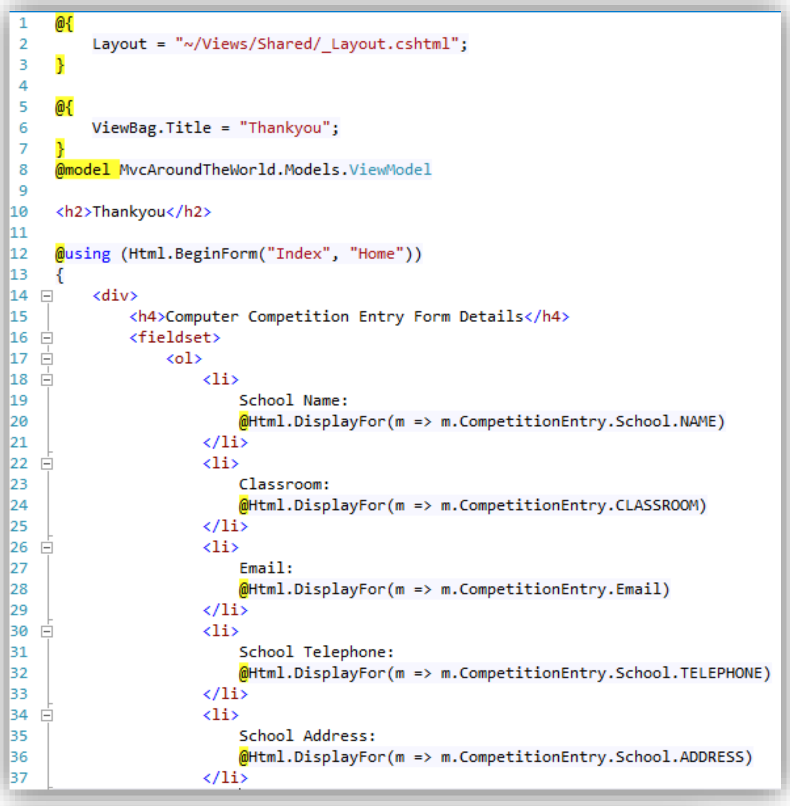


Thankyou

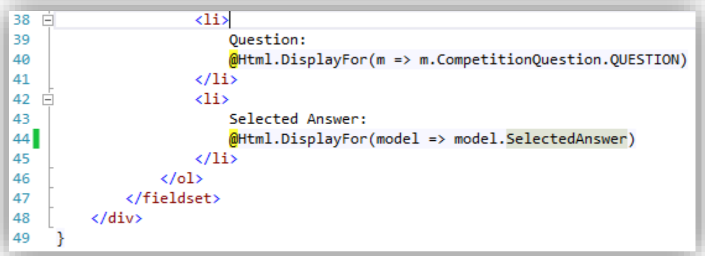
Computer Competition Entry Form Details

1. School Name: Victory
2. Classroom: 14
3. Email: hi
4. School Telephone: 08004528545
5. School Address: 54 Vanguard Street
6. Question: The object of the game is to:
7. Selected Answer: 1

© Around The World



```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}

@{
    ViewBag.Title = "Thankyou";
}
@model MvcAroundTheWorld.Models.ViewModel

<h2>Thankyou</h2>

@using (Html.BeginForm("Index", "Home"))
{
    <div>
        <h4>Computer Competition Entry Form Details</h4>
        <fieldset>
            <ol>
                <li>
                    School Name:
                    @Html.DisplayFor(m => m.CompetitionEntry.School.NAME)
                </li>
                <li>
                    Classroom:
                    @Html.DisplayFor(m => m.CompetitionEntry.CLASSROOM)
                </li>
                <li>
                    Email:
                    @Html.DisplayFor(m => m.CompetitionEntry.Email)
                </li>
                <li>
                    School Telephone:
                    @Html.DisplayFor(m => m.CompetitionEntry.School.TELEPHONE)
                </li>
                <li>
                    School Address:
                    @Html.DisplayFor(m => m.CompetitionEntry.School.ADDRESS)
                </li>
```

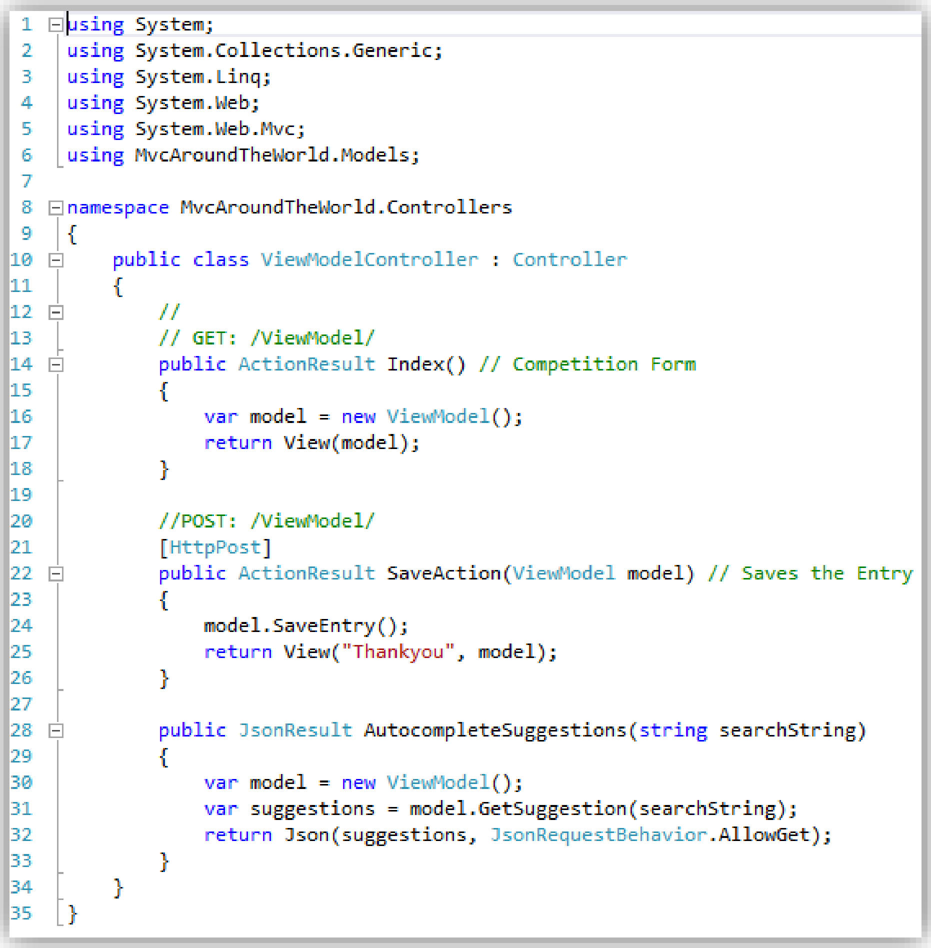


```
                <li>
                    Question:
                    @Html.DisplayFor(m => m.CompetitionQuestion.QUESTION)
                </li>
                <li>
                    Selected Answer:
                    @Html.DisplayFor(model => model.SelectedAnswer)
                </li>
            </ol>
        </fieldset>
    </div>
}
```

Each list item has text as a 'heading'. It is then displaying the details that were entered on the previous page by calling it from the database.

ViewModel Controller

A controller had to be created to handle the distribution of the Index view for the competition.

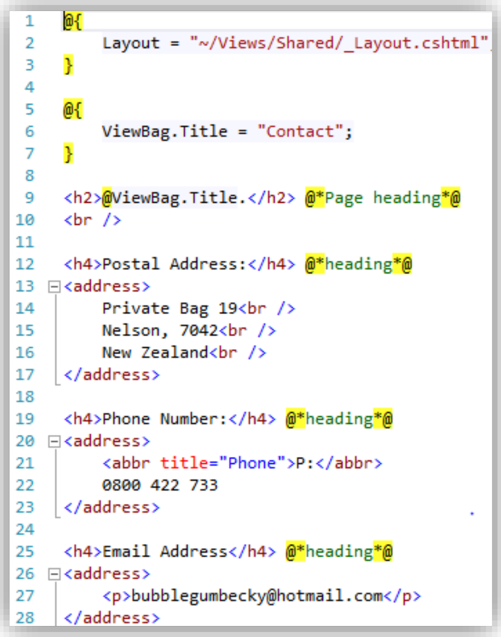


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using MvcAroundTheWorld.Models;

namespace MvcAroundTheWorld.Controllers
{
    public class ViewModelController : Controller
    {
        //
        // GET: /ViewModel/
        public ActionResult Index() // Competition Form
        {
            var model = new ViewModel();
            return View(model);
        }

        //POST: /ViewModel/
        [HttpPost]
        public ActionResult SaveAction(ViewModel model) // Saves the Entry
        {
            model.SaveEntry();
            return View("Thankyou", model);
        }

        public JsonResult AutocompleteSuggestions(string searchString)
        {
            var model = new ViewModel();
            var suggestions = model.GetSuggestion(searchString);
            return Json(suggestions, JsonRequestBehavior.AllowGet);
        }
    }
}
```

It also contains the SaveAction that returns the thank you view. Then there is the AutoCompleteSuggestions that uses Json to return The GetSuggestion list that was created in ViewModel.

Contact

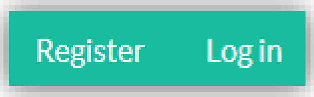
For the contact page I just added details for contacting the 'owner' of the website. There are three headings and three address tags that contain the appropriate details.

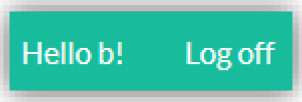


```
@{
    Layout = "~/Views/Shared/_Layout.cshtml"
}

@{
    ViewBag.Title = "Contact";
}

<h2>@ViewBag.Title.</h2> @*Page heading*@
<br />

<h4>Postal Address:</h4> @*heading*@
<address>
    Private Bag 19<br />
    Nelson, 7042<br />
    New Zealand<br />
</address>

<h4>Phone Number:</h4> @*heading*@
<address>
    <abbr title="Phone">P:</abbr>
    0800 422 733
</address>

<h4>Email Address</h4> @*heading*@
<address>
    <p>bubblegumbecky@hotmail.com</p>
</address>
```

Register and Login

I edited register and login so that they work again.





Content

At this point in time all of my content is completed. The only thing that currently needs adjusting is the scoreboard. If I have time I will try to make it interactive. My thoughts on doing that are by possibly adding click event listeners that will just draw a line on a number when it is clicked.

Conclusion

Due to being ill for most of this section I am very pleased with how much progress that I have made with my website.