COM580 Assignment 2

ASP.NET MVC Entity Framework

Solution

Code available online at: https://github.com/mark-ruddy/university_webapp

a) Code for the "*Student*" C# class model and screen snapshots of the associated database table structure and content

StudentModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ComponentModel.DataAnnotations;

public class StudentModel
{
    public int ID { get; set; }

    [DataType(DataType.Text)]
    [Required(ErrorMessage = "First Name Required")]
    public string FirstName { get; set; }

    [DataType(DataType.Text)]
    [Required(ErrorMessage = "Surname Required")]
    public string Surname { get; set; }

    [DataType(DataType.Text)]
    // international mobile phone number regex
    [RegularExpression(@"^\+[1-9]{1}[0-9]{3,14}$", ErrorMessage = "Not a valid Telephone
Number")]
    [Required(ErrorMessage = "Telephone No. Required")]
    public string TelephoneNo { get; set; }

    [DataType(DataType.Text)]
    // default email regex built into HTML5
    [RegularExpression(@".+\@.+\..+", ErrorMessage = "Not a valid Email")]
    [Required(ErrorMessage = "Email Required")]
    public string email { get; set; }

    [DataType(DataType.Text)]
    [Required(ErrorMessage = "Country of Origin Required")]
    public string countryOfOrigin { get; set; }
}
```

b) Code for the "*IStudentRepository*" interface C# class and "*StudentRepository*" C# class.

IstudentRepository.cs

```
using static StudentModel;

interface IStudentRepository{
  IEnumerable<StudentModel> SelectAll();
  StudentModel SelectByID(int id);
  void Insert(StudentModel student);
  void Update(StudentModel student);
  void Delete(int id, int? replacementID);
  void Save();
}
```

StudentRepository.cs

```
using Microsoft.EntityFrameworkCore;
using static StudentModel;
using static MvcStudentContext;


public class StudentRepository : IStudentRepository {
  private MvcStudentContext db = null;

  public StudentRepository() {
    this.db = new MvcStudentContext(null);
  }

  public StudentRepository(MvcStudentContext db) {
    this.db = db;
  }

  public IEnumerable<StudentModel> SelectAll() {
    return db.StudentModel.OrderBy(s => s.Surname).ToList();
  }

  public StudentModel SelectByID(int ID) {
    return db.StudentModel.Find(ID);
  }

  public void Insert(StudentModel student) {
    db.StudentModel.Add(student);
  }

  public void Update(StudentModel student) {
    db.Entry(student).State = EntityState.Modified;
  }

  public void Delete(int ID, int? replacementID) {
    StudentModel existing = db.StudentModel.Find(ID);
    db.StudentModel.Remove(existing);
  }

  public void Save() {
```

```
    db.SaveChanges();
  }

  public IEnumerable<String> CountryList() {
    var listOfCountries = File.ReadLines("countries.list").Select(line => new String(line)).ToList();
    return listOfCountries;
  }
}
```

c) Code for the "*StudentController*" C# class.

StudentController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;

namespace university.Controllers
{
    public class StudentsController : Controller
    {
        private readonly MvcStudentContext _context;

        public StudentsController(MvcStudentContext context)
        {
            _context = context;
        }

        // GET: Students
        public async Task<IActionResult> Index()
        {
            return View(await _context.StudentModel.ToListAsync());
        }

        // GET: Students/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var studentModel = await _context.StudentModel
                .FirstOrDefaultAsync(m => m.ID == id);
            if (studentModel == null)
            {
                return NotFound();
            }

            return View(studentModel);
        }
```

```
        // GET: Students/Create
        public IActionResult Create()
        {
            return View();
        }

        // POST: Students/Create
        // To protect from overposting attacks, enable the specific properties you want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult>
Create([Bind("ID,FirstName,Surname,TelephoneNo,email,countryOfOrigin")] StudentModel
studentModel)
        {
            if (ModelState.IsValid)
            {
                _context.Add(studentModel);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(studentModel);
        }

        // GET: Students/Edit/5
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var studentModel = await _context.StudentModel.FindAsync(id);
            if (studentModel == null)
            {
                return NotFound();
            }
            return View(studentModel);
        }

        // POST: Students/Edit/5
        // To protect from overposting attacks, enable the specific properties you want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id,
[Bind("ID,FirstName,Surname,TelephoneNo,email,countryOfOrigin")] StudentModel studentModel)
        {
            if (id != studentModel.ID)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
```

```
            {
                _context.Update(studentModel);
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!StudentModelExists(studentModel.ID))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }
            return RedirectToAction(nameof(Index));
        }
        return View(studentModel);
    }

    // GET: Students/Delete/5
    public async Task<IActionResult> Delete(int? id)
    {
        if (id == null)
        {
            return NotFound();
        }

        var studentModel = await _context.StudentModel
            .FirstOrDefaultAsync(m => m.ID == id);
        if (studentModel == null)
        {
            return NotFound();
        }

        return View(studentModel);
    }

    // POST: Students/Delete/5
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> DeleteConfirmed(int id)
    {
        var studentModel = await _context.StudentModel.FindAsync(id);
        _context.StudentModel.Remove(studentModel);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }

    private bool StudentModelExists(int id)
    {
        return _context.StudentModel.Any(e => e.ID == id);
    }
  }
}
```

d) Screenshots for web enabled Creating/Editing/Details/Deleting of a student.

List Students

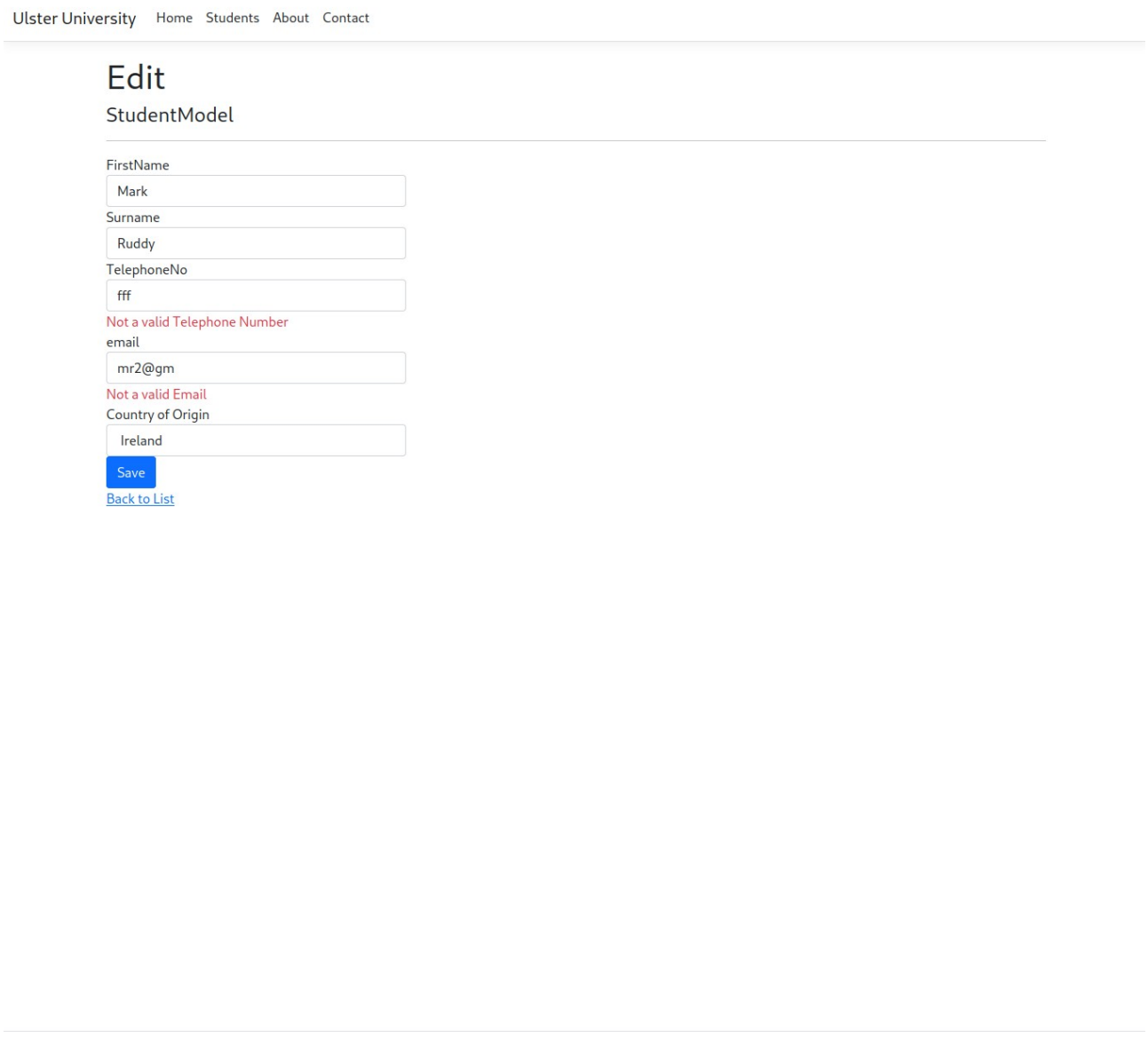
Ulster University Home Students About Contact

Student List

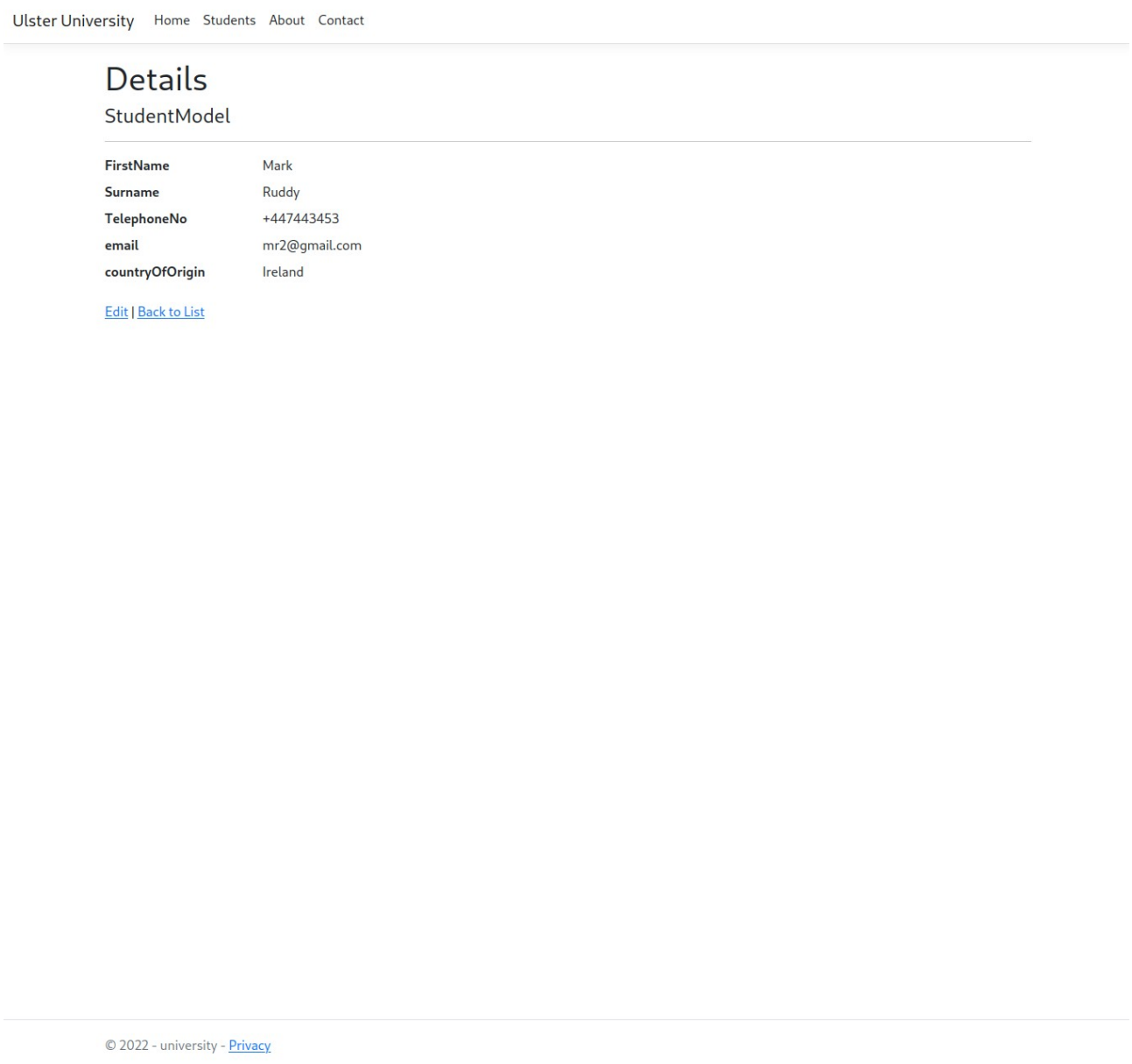
Create New

FirstName	Surname	TelephoneNo	email	countryOfOrigin	
Mark	Ruddy	+447443453	mr2@gmail.com	Ireland	Edit | Details | Delete
Dan	Burns	+443334322	danburns@gmail.com	Albania	Edit | Details | Delete
Robert	McLaughlin	+35323343243	robmcla22@hotmail.com	Ireland	Edit | Details | Delete
Amy	White	+44334338282	amyw99@gmail.com	United Kingdom	Edit | Details | Delete

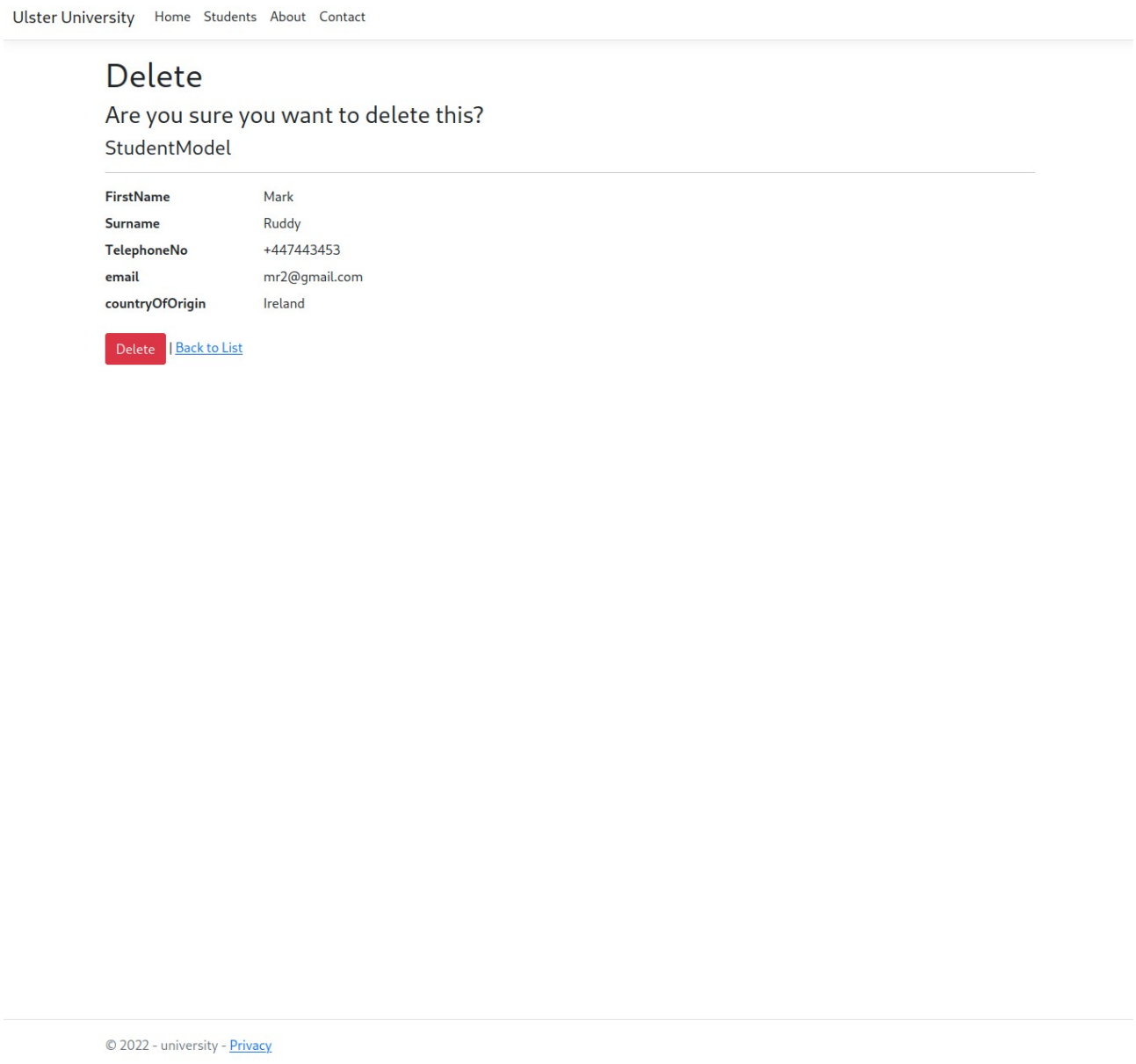
Edit Student



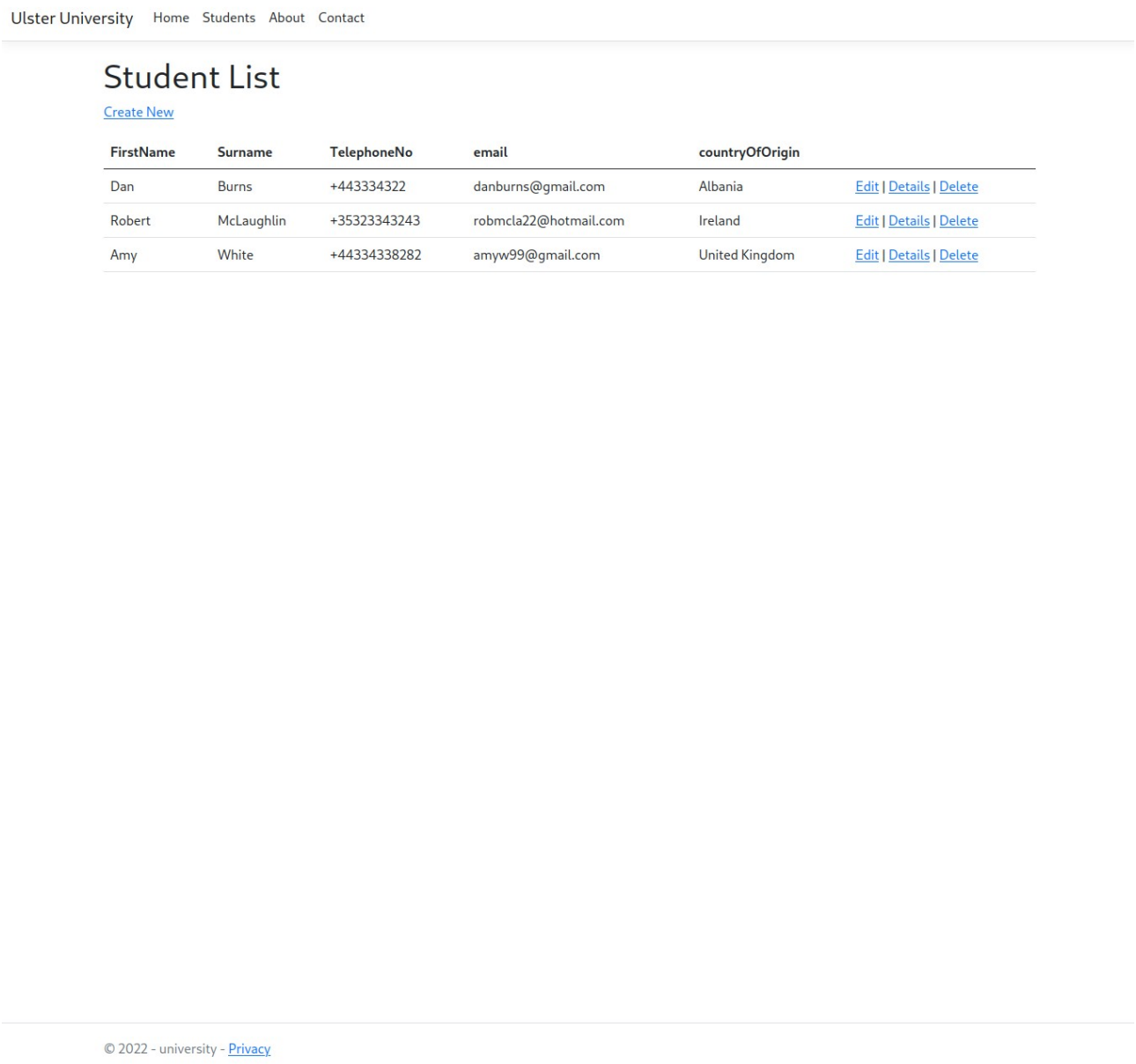
Student Details



Delete Student Confirmation



Post-Delete Student List Updated



e) Provide a rationale covering your design decisions associated with the code development.

General

The webapp was designed around the Model-View-Controller architecture – which involves first the Controller action being requested through a HTTP router, then the action in most cases will query the Model for database info, then the action will call the corresponding View function passing in any required database info. Finally the action will return the HTML, CSS and JS bundle to the users browser.

The webapp uses the SQLite database, which is not recommended for high load production environments but is suitable for small environments or demonstrations. This

allows the DB to be saved as a file in the git repo, and uploaded to github, gitlab, etc. This choice was made so that the realistic data entered could easily be replicated across different computers when cloning the repo down.

This webapp also happens to be developed with .NET Core on a Linux machine, while also being compatible with Windows. This choice was made due to programmer preference, and is seen in the code by use of *using Microsoft.EntityFrameworkCore;* instead of *using System.Data.Entity;*

Controllers

For the University Student Manager webapp there are multiple controllers – Home, About, Contact and Students. Apart from Students, the controllers do not query the SQLite database, as they return a View without database info. The Students controller contains the logic for CRUD control over the Student Model – with generated code using this command:

dotnet-aspnet-codegenerator controller -name StudentsController -m StudentModel -dc MvcStudentContext --relativeFolderPath Controllers --useDefaultLayout --referenceScriptLibraries -sqlite

This command also generates the CRUD frontend **.cshtml** pages, which explains why the site looks similar to other C# ASP.NET MVC apps using this approach. By generating the CRUD interaction the developer can save time instead of implementing each Students action manually. This approach may even be essential if there are multiple models, manually writing code for 10+ CRUD models would be tedious and unnecessary due to the overlap in content.

Model

The model has data verification through RegularExpressions and Required directives. For example the email regex directive is:

[RegularExpression(@".+\@.+\..+", ErrorMessage = "Not a valid Email")]

This email regex is a basic one to prevent students being created with malformed emails. It will not cover every email validation case, which can become very complex.

Repository

The Repository for Students provides a C# interface for CRUD operations on the Student Model. This provides a clear interaction with the model which is implemented by the programmer.

This may allow the programmer to place custom middleware in the methods, such as *Update(StudentModel student)* – any extra code could be placed in this method to log or have some side-effect. When using the usual *context.Update(studentModel)*, it is not possible to change this methods behavior as it is defined by *Microsoft.EntityFrameworkCore*, without overriding the method which may not be desired.