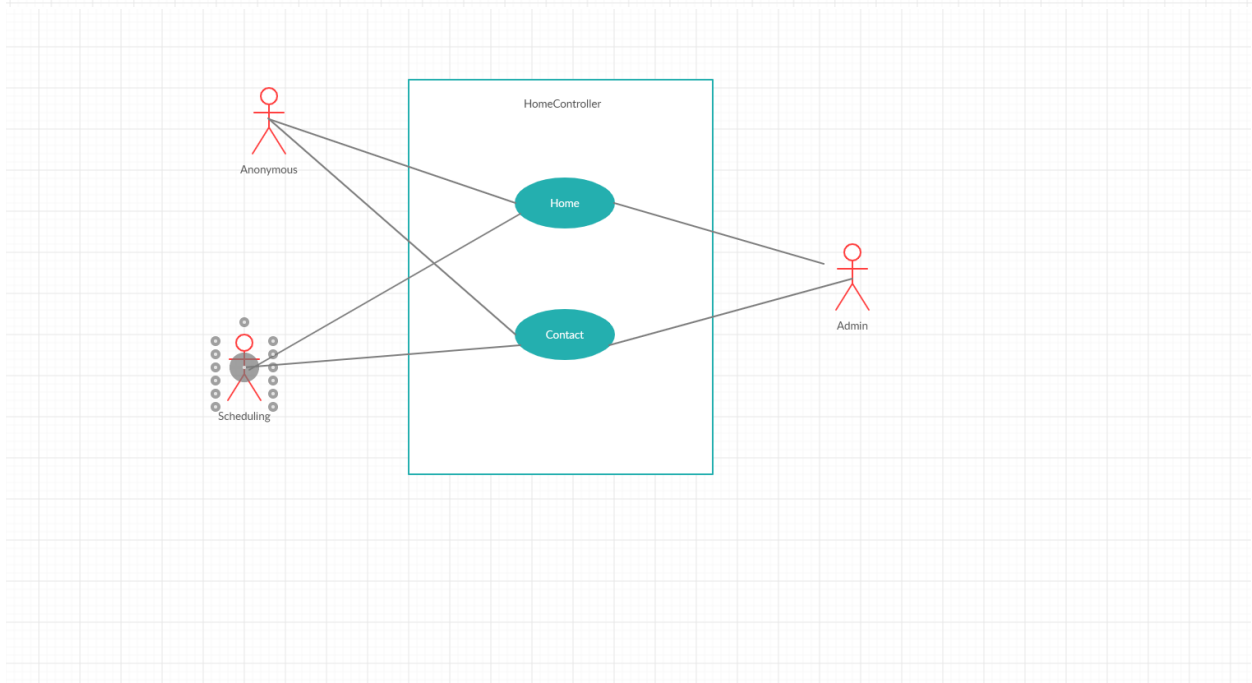
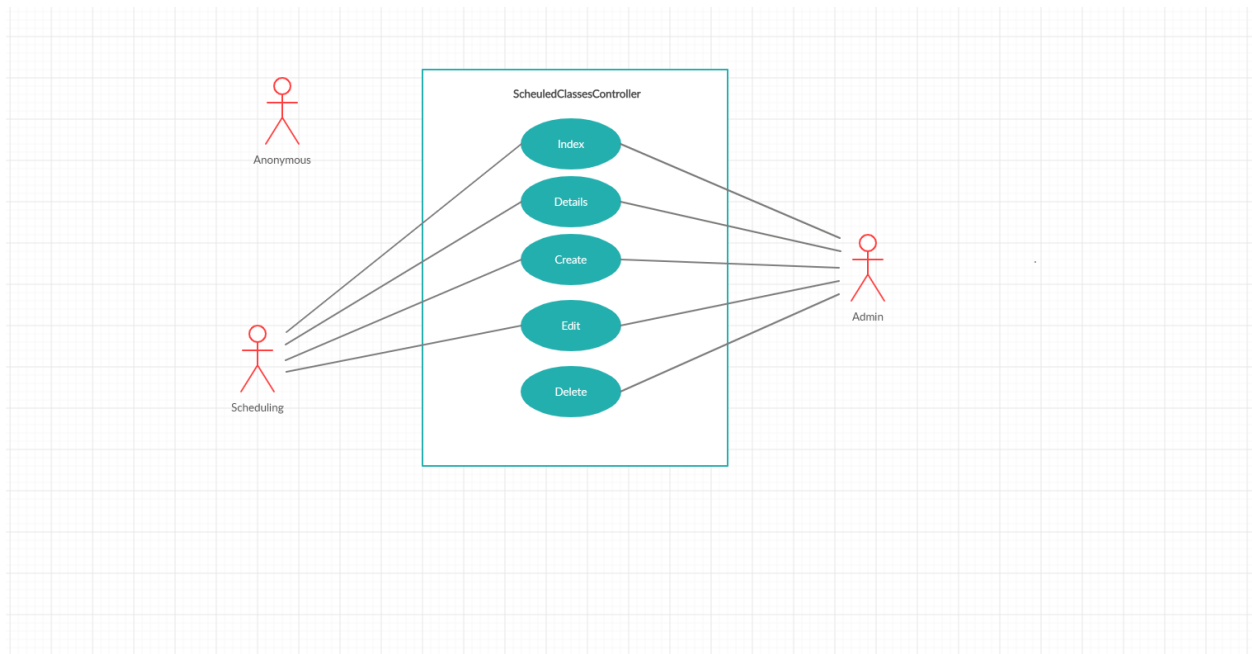
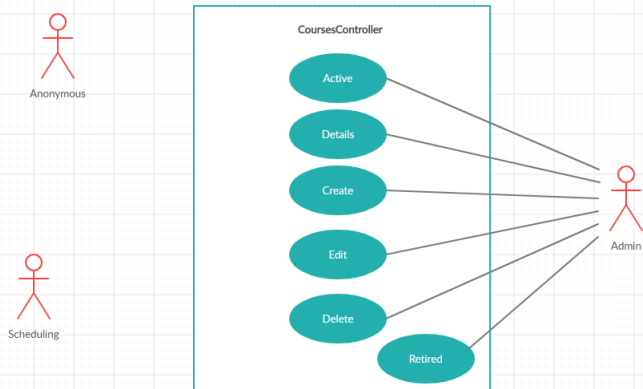
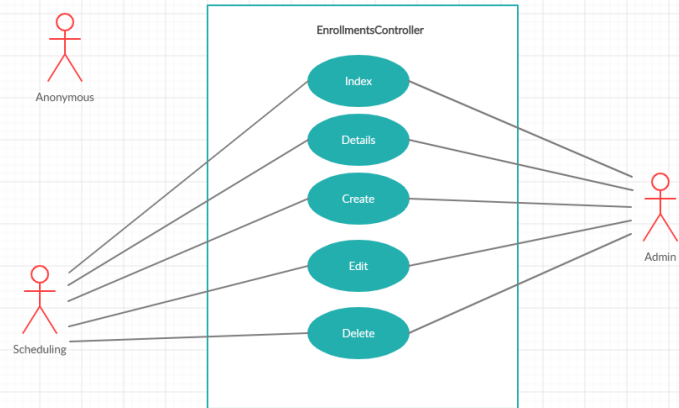
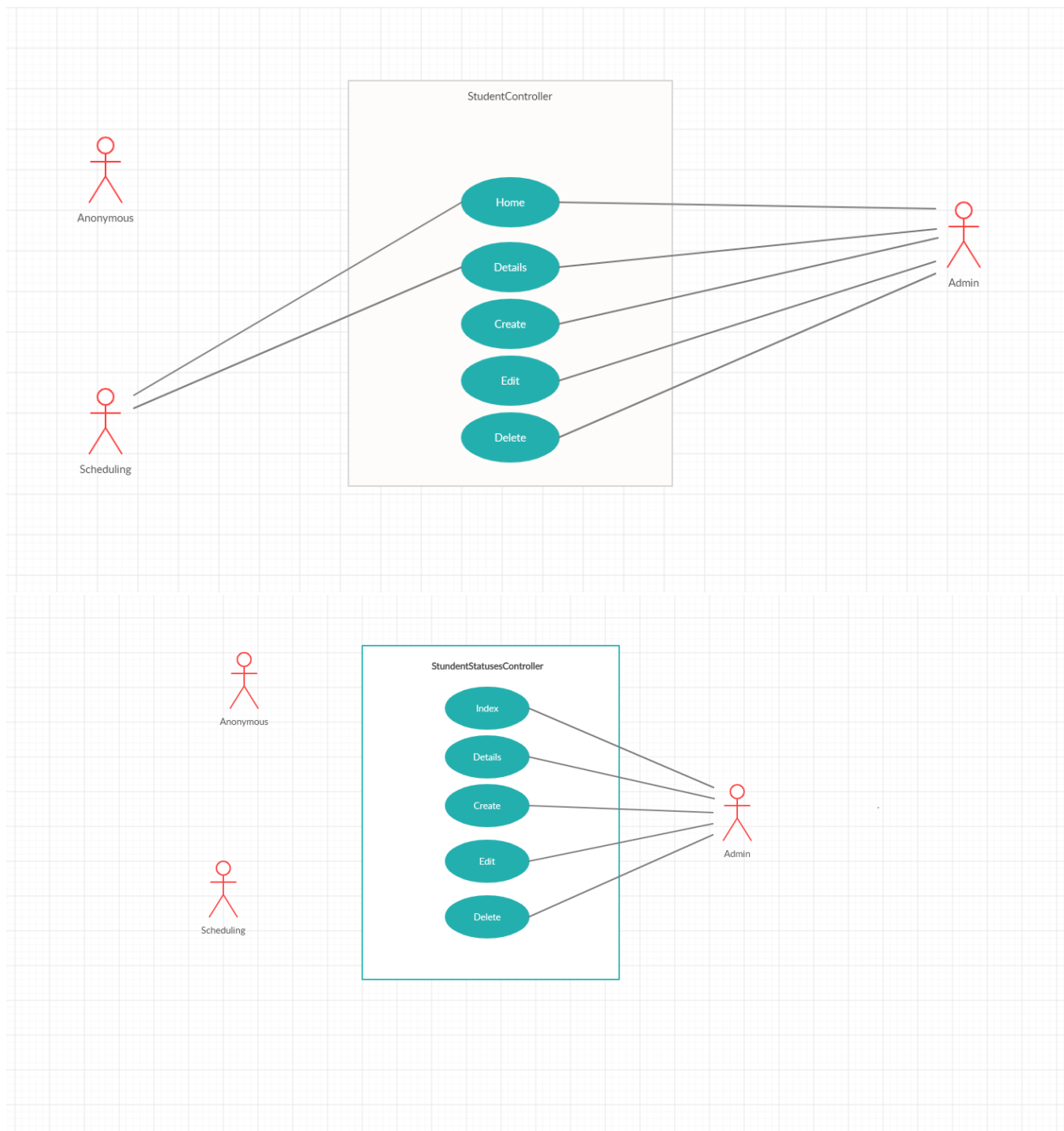


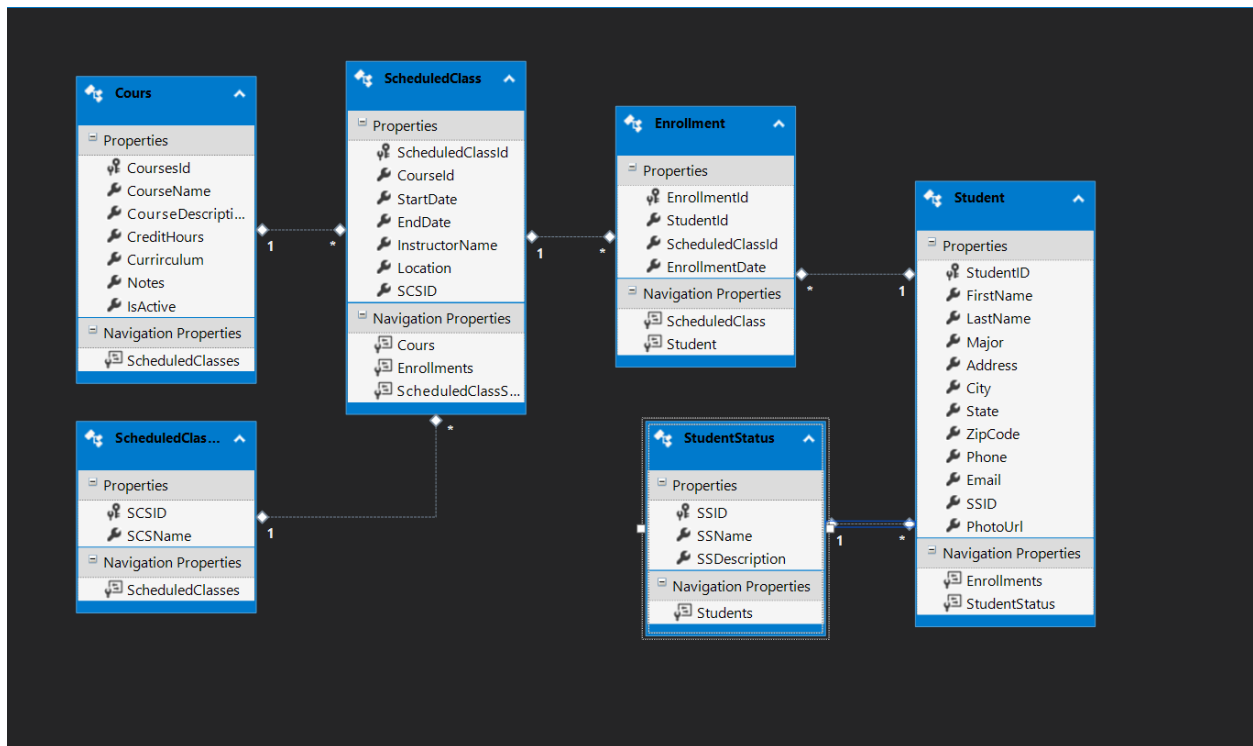
Identity Matrix Diagrams



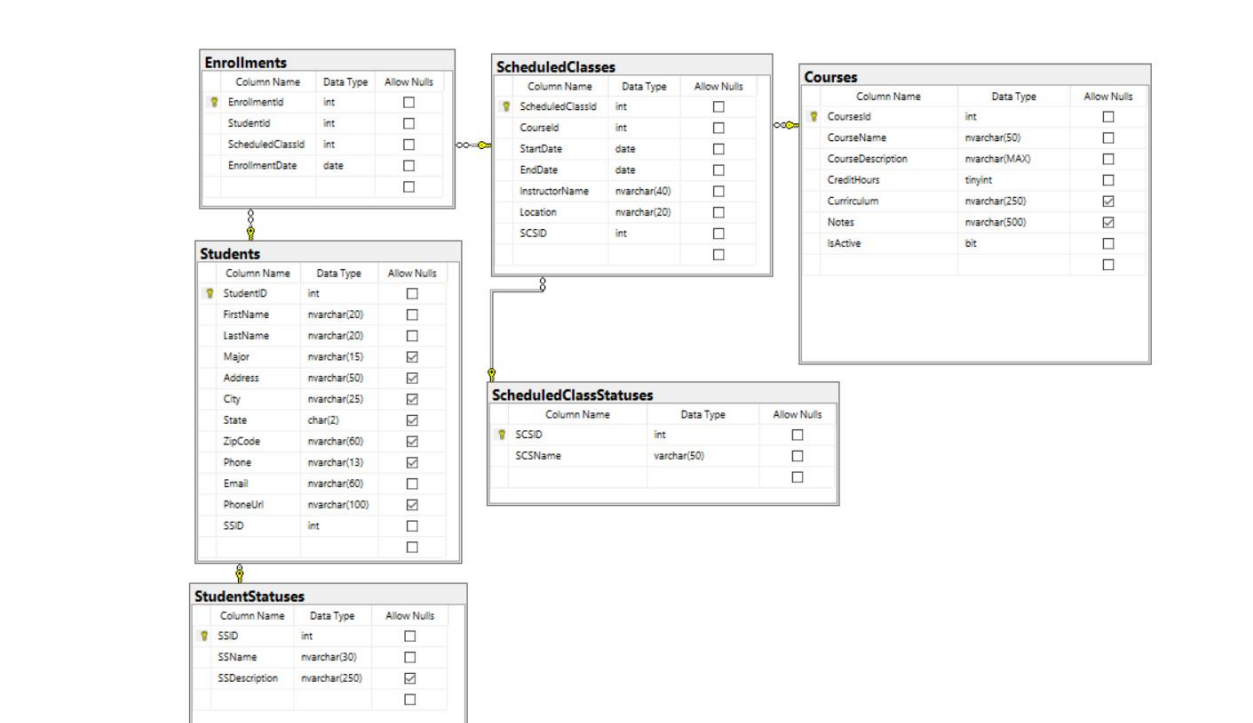




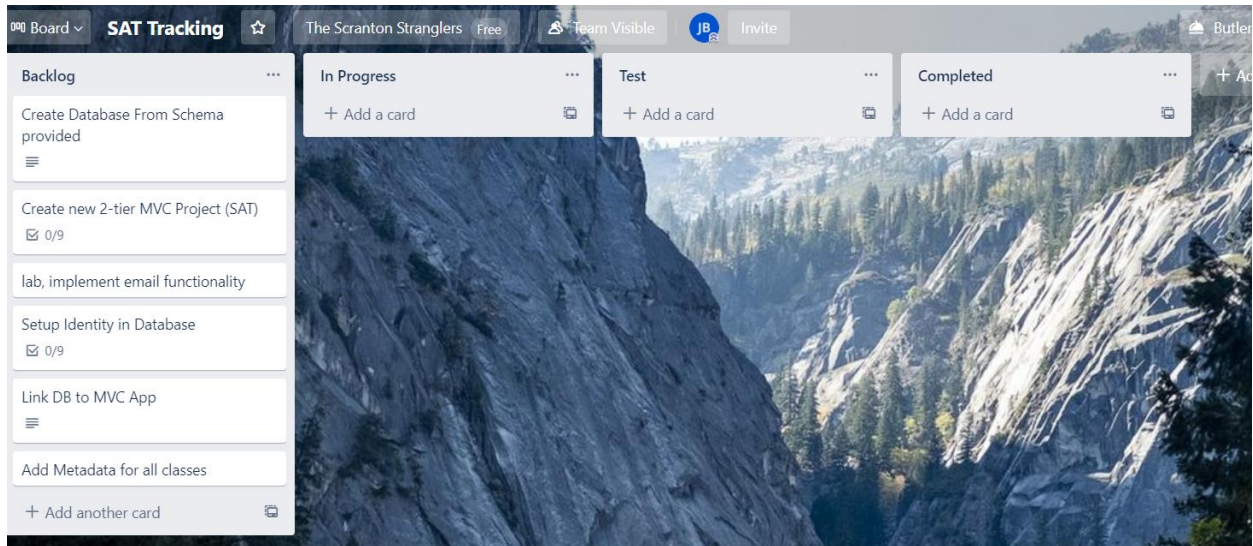
Data Diagram edmx



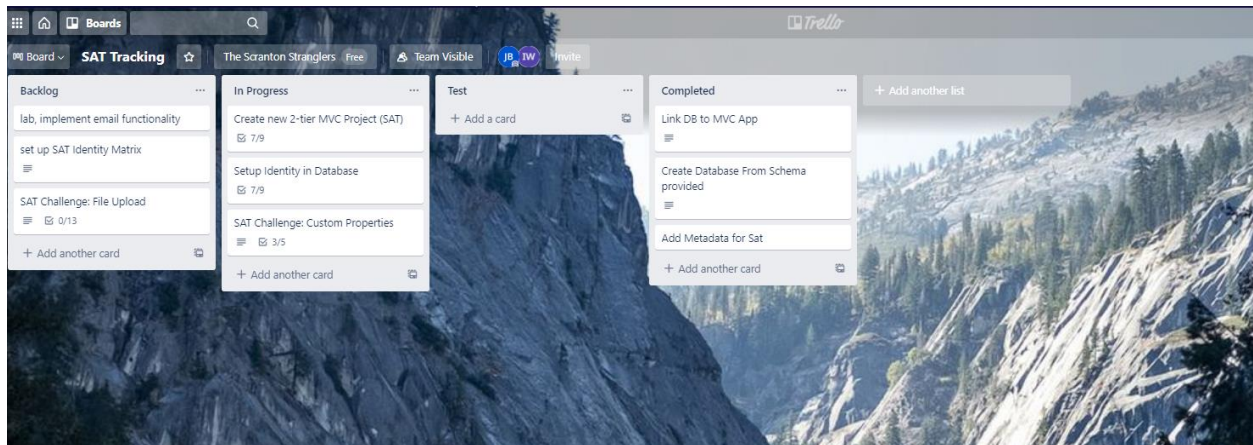
Data Diagram SQL



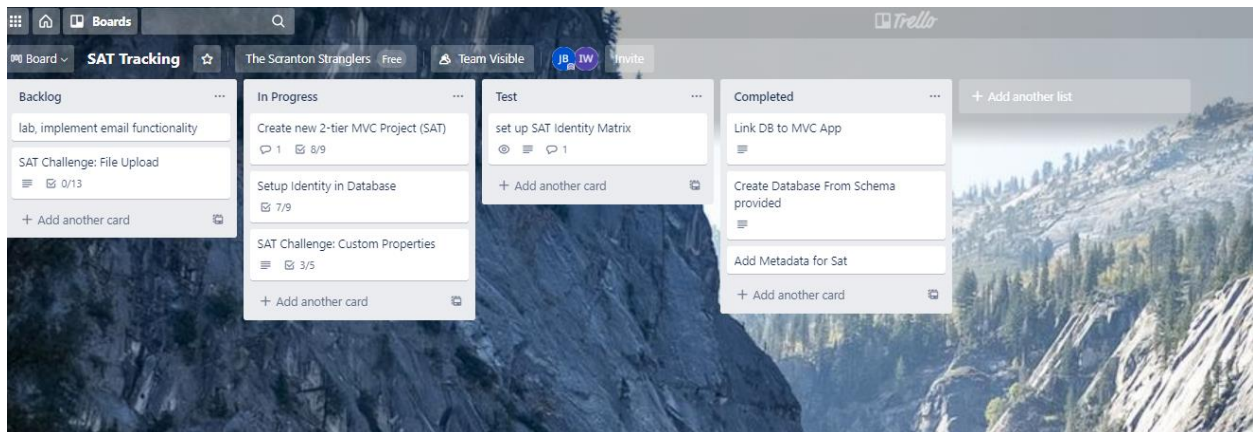
Timeline Trello Boards



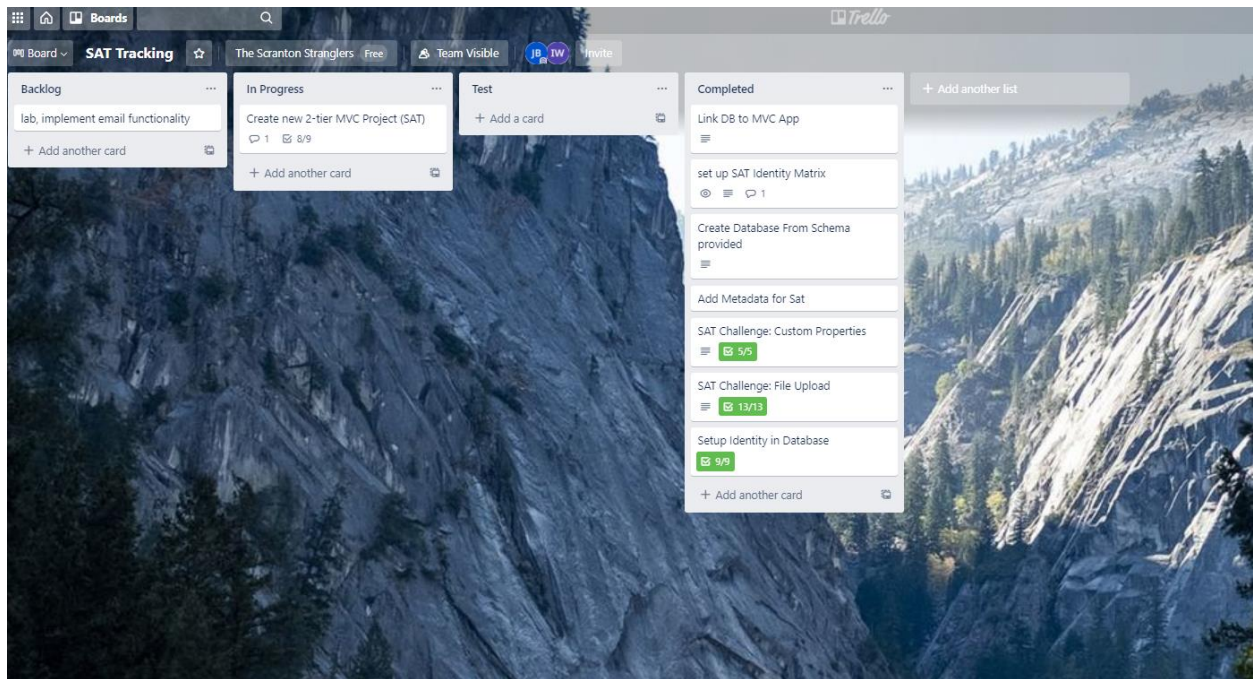
10-28-2020



11-2-2020



11-3-2020



11-4-2020

FROM ADMIN

Index

[Create New](#)

| Enrollment Date | Instructor Name | First Name | |
|-----------------|-----------------|------------|---|
| 5/16/2020 | Stan | John | Edit Details Delete |
| 5/17/2020 | Nuckolls | John | Edit Details Delete |
| 5/18/2020 | Joelle | John | Edit Details Delete |

[Roles](#)
[Users](#)

Index

[Create New](#)

| UserName | |
|---------------|---|
| sat@sat.com | Edit Details Delete |
| test@sat.com | Edit Details Delete |
| admin@SAT.com | Edit Details Delete |

[Roles](#)
[Users](#)

Home Page

[Roles](#)

[Users](#)

© 2020 All rights reserved. Apptonic.co. Design by Bootstrapious

Index

[Create New](#)

| Name | |
|------------|---|
| Admin | Edit Details Delete |
| Scheduling | Edit Details Delete |

[RolesAdmin](#)

[UsersAdmin](#)

© 2020 All rights reserved. Apptonic.co. Design by Bootstrapious

Scheduling Account

Index

Create New

| Start Date | End Date | Instructor Name | Location | Course Name | Status | |
|------------|-----------|-----------------|----------|-------------|-----------|--|
| 12/15/2020 | 5/15/2021 | Nuckolls | Centriq | test1 | Pending | Edit Details |
| 12/15/2020 | 5/15/2021 | Joelle | Centriq | test2 | Completed | Edit Details |
| 12/15/2020 | 5/15/2021 | Stan | Centriq | test3 | Cancelled | Edit Details |

- Enrollment
- Class Schedule
- Students

Home Page

Code from Custom Props and File Upload Challenge

```
// POST: Students/Create
// To protect from overposting attacks, please enable the specific properties you want to bind to, for
// more details see https://go.microsoft.com/fwlink/?linkid=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include = "StudentID,FirstName,LastName,Major,Address,City,State,ZipCode,Phone,Email,PhotoUrl,SSID")] Student student,
    HttpPostedFileBase studentPic)
{
    if (ModelState.IsValid)
    {
        #region File Upload - Using the Image Service

        //use a default image if one is not provided when the record is created - noImage.png
        string imgName = "noImage.jpg";

        //branch - to make sure the input type file (HttpPostedFileBase) is NOT null (it has a file)
        if (studentPic != null)
        {
            //retrieve the image from HPFB and assign to our image variable
            imgName = studentPic.FileName;

            //declare and assign the extension
            string ext = imgName.Substring(imgName.LastIndexOf('.'));

            //create a list of valid extensions
            string[] goodExts = { ".jpeg", ".jpg", ".gif", ".png" };

            //check the extension against the list of valid extensions and make sure the file size is 4MB or LESS (ASP.net limit)
            //if all requirements are met - do the following
            if (goodExts.Contains(ext.ToLower()) && (studentPic.ContentLength <= 4194304))//4mb in bytes
            {
                //rename the file using a GUID (Global Unique Identifier) and concatenate with the extension.
                imgName = Guid.NewGuid() + ext.ToLower();//toLowerCase() is not required, does ensure all ext's are lower case

                string savePath = Server.MapPath("~/Content/img/");

                //image value for the converted image
                Image convertedImage = Image.FromStream(studentPic.InputStream);

                //max image size
                int maxImageSize = 500;
            }
        }
    }
}
```

Email

Image

Existing Image:



Change Image: No file chosen

Status

SAVE

[Back to List](#)

Shows the custom prop of Class Schedule down below

```
namespace SAT.DATA.EF
{
    public class ScheduledClassMetaData
    {
        [Required(ErrorMessage = "** Course is REQUIRED")]
        [Display(Name = "Course")]
        public int CourseId { get; set; }
        [Display(Name = "Start Date")]
        [DisplayFormat(DataFormatString = "{0:d}", ApplyFormatInEditMode = true)]
        public System.DateTime StartDate { get; set; }
        [Display(Name = "End Date")]
        [DisplayFormat(DataFormatString = "{0:d}", ApplyFormatInEditMode = true)]
        public System.DateTime EndDate { get; set; }
        [StringLength(40, ErrorMessage = "** Instructor can only be a max of 40 characters.")]
        [Required(ErrorMessage = "** Instructor is REQUIRED")]
        [Display(Name = "Instructor Name")]
        public string InstructorName { get; set; }
        [StringLength(20, ErrorMessage = "** Location can only be a max of 20 characters.")]
        [Required(ErrorMessage = "** Location is REQUIRED")]
        public string Location { get; set; }
        [Display(Name = "Scheduled Class Status")]
        [Required(ErrorMessage = "** Scheduled Class Status is REQUIRED")]
        public int SCSID { get; set; }
    }


    [MetadataType(typeof(ScheduledClassMetaData))]
    public partial class ScheduledClass
    {
        [Display(Name = "Scheduled Class: ")]
        public string ScheduledClassString
        {
            get { return ($"Start Date: {StartDate:d}, Course Name: {Cours.CourseName}, Location: {Location}") ; }
        }
    }
}
```

ScheduledClassId

Start Date: 12/15/2020, Course Name: test3, Location: Centriq ▼

Shows the custom prop of Full Name down below

Index

| studentFullname | Major | Address | City | State | Zip code | Phone | Email | Image | Student Status |
|-----------------|----------|-------------|-------------|-------|----------|-------------|-----------------|---|---|
| John Doel | Business | 351 Penn dr | Kansas City | MO | 64178 | 18161234567 | email@email.com |  | Current Student Details |

```
namespace SAT.DATA.EF
{
    public class StudentMetaData
    {
        [Required(ErrorMessage = "* First Name is REQUIRED")]
        [Display(Name = "First Name")]
        [StringLength(20, ErrorMessage = "* First Name can only be a max of 20 characters.")]
        public string FirstName { get; set; }
        [Required(ErrorMessage = "* Last Name is REQUIRED")]
        [Display(Name = "Last Name")]
        [StringLength(20, ErrorMessage = "* Last Name can only be a max of 20 characters.")]
        public string LastName { get; set; }
        [StringLength(15, ErrorMessage = "* Majors can only be a max of 15 characters.")]
        public string Major { get; set; }
        [StringLength(50, ErrorMessage = "* Address can only be a max of 50 characters.")]
        public string Address { get; set; }
        [StringLength(25, ErrorMessage = "* City can only be a max of 25 characters.")]
        public string City { get; set; }
        [StringLength(2, ErrorMessage = "* State can only be a max of 2 characters.")]
        public string State { get; set; }
        [StringLength(60, ErrorMessage = "* Zipcode can only be a max of 60 characters.")]
        [Display(Name = "Zip code")]
        public string ZipCode { get; set; }
        [StringLength(13, ErrorMessage = "* Phone can only be a max of 13 characters.")]
        public string Phone { get; set; }
        [StringLength(60, ErrorMessage = "* Email can only be a max of 60 characters.")]
        [Required(ErrorMessage = "* Email is REQUIRED")]
        public string Email { get; set; }
        [Display(Name = "Image")]
        [StringLength(100, ErrorMessage = "* Photo URL can only be a max of 100 characters.")]
        public string PhotoUrl { get; set; }
        [Required(ErrorMessage = "* Student Status is REQUIRED")]
        [Display(Name = "Student Status")]
        public int SSID { get; set; }
    }

    [MetadataType(typeof(StudentMetaData))]
    public partial class Student
    {
        public string studentFullname
        {
            get { return $"{FirstName} {LastName}"; }
        }
    }
}
```

Default new registered user to Scheduling Role

```
// POST: /Account/Register
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = new ApplicationUser { UserName = model.Email, Email = model.Email };
        var result = await UserManager.CreateAsync(user, model.Password);
        if (result.Succeeded)
        {
            var code = await UserManager.GenerateEmailConfirmationTokenAsync(user.Id);
            var callbackUrl = Url.Action("ConfirmEmail", "Account", new { userId = user.Id, code = code }, protocol:
                Request.Url.Scheme);
            //await UserManager.SendEmailAsync(user.Id, "Confirm your account", "Please confirm your account by clicking this
            link: <a href=\"" + callbackUrl + "\">link</a>");
            await UserManager.AddToRoleAsync(user.Id, "Scheduling");
            ViewBag.Link = callbackUrl;
            return View("DisplayEmail");
        }
        AddErrors(result);
    }

    // If we got this far, something failed, redisplay form
    return View(model);
}
```

BundlesConfig Shows Custom UI files taken from Template

```
public class BundleConfig
{
    // For more information on bundling, visit http://go.microsoft.com/fwlink/?LinkId=301862
    public static void RegisterBundles(BundleCollection bundles)
    {
        bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
            "~/Scripts/jquery.min.js",
            "~/Scripts/jquery.cookie.js"));

        bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
            "~/Scripts/jquery.validate*"));

        // Use the development version of Modernizr to develop with and learn from. Then, when you're
        // ready for production, use the build tool at https://modernizr.com to pick only the tests you need.
        bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(
            "~/Scripts/modernizr-*"));

        bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include(
            "~/Scripts/bootstrap.bundle.min.js",
            "~/Scripts/lightbox.js",
            "~/Scripts/front.js"));

        bundles.Add(new StyleBundle("~/Content/css").Include(
            "~/Content/bootstrap.min.css",
            "~/Content/font-awesome.css",
            "~/Content/lightbox.css",
            "~/Content/fontastic.css",
            "~/Content/style.default.css",
            "~/Content/custom.css"));
    }
}
```