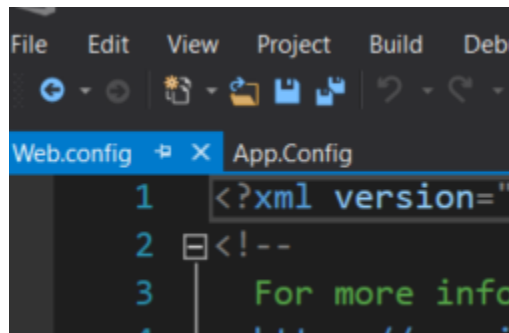


Documentation

The SAT application portfolio project requirements were to work as a development team using the agile and scrum methodologies to build a full stack application. We were to create a Trello board and git hub repo and link to our teammates. One teammate was the driver and the others were the navigators. We implemented a database using entity framework, added login functionality using the identity samples NuGet package. We scaffolded out controllers and converted a template along with additional functionality that will be listed in the steps below.

Struggled with connection strings – we did not originally update our connection string before implementing Identity samples so we were not able to connect to our database. We ended up starting over because it was so early on in the project. We made sure to update the connection string and once we did that we were able to connect to visual studios to our database



Struggled with photo upload functionality - the photo upload functionality was working for a full day then stopped! When taking a closer look we noticed on our student controller our saveas server.mappath was missing the concatenated imageName(which was our variable for the photoUrl.FileName). also on edit we had the image name concatenated but what we needed was the student.PhotoUrl. We were deleting the photo as we were uploading. Once those issues were fixed our photo edit and create worked perfect.

```
Student student, HttpPostedFileBase PhotoUrl)
{
    if (ModelState.IsValid)
    {
        #region File Upload
        if (PhotoUrl != null)
        {
            string imageName = PhotoUrl.FileName;
            string ext = imageName.Substring(imageName.LastIndexOf("."));
            string[] goodExts = new string[] { ".jpg", ".png", ".jpeg", ".gif" };
            if (goodExts.Contains(ext.ToLower()))
            {
                imageName = Guid.NewGuid() + ext;
                PhotoUrl.SaveAs(Server.MapPath("~/Content/assets/img/Student/" + imageName));
                if (student.PhotoUrl != "noImage.png" && student.PhotoUrl != null)
                {
                    System.IO.File.Delete(Server.MapPath("~/Content/assets/img/Student/" + student.PhotoUrl));
                }
            }
            student.PhotoUrl = imageName;
        }
    }
}

#endregion
```

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include = "StudentId,FirstName,LastName,Major,Address,City,State,ZipCode,Phone,Email,PhotoUrl,SSID")]
    Student student, HttpPostedFileBase PhotoUrl)
{
    if (ModelState.IsValid)
    {
        #region File Upload
        string imageName = "noImage.png";
        if (PhotoUrl != null)
        {
            imageName = PhotoUrl.FileName;
            string ext = imageName.Substring(imageName.LastIndexOf("."));
            string[] goodExts = new string[] { ".jpg", ".png", ".jpeg", ".gif" };
            if (goodExts.Contains(ext.ToLower()))
            {
                imageName = Guid.NewGuid() + ext;
                PhotoUrl.SaveAs(Server.MapPath("~/Content/assets/img/Student/" + imageName));
            }
            else
            {
                imageName = "noImage.png";
            }
        }
    }
}
```

25% COMPLETE:

- | |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <input type="checkbox"/> 1. Create a Trello board to track the progress of the project. Create cards for individual tasks and assign the cards to the team member who will complete the work. Ensure cards are detailed and reference necessary resources. |
| <input type="checkbox"/> 2. Assign a team member to take screenshots of the Trello board as the project progresses. This team member should also get screen shots of interesting sections of code. These screen shots will be used for project documentation. |
| <input type="checkbox"/> 3. Build a relational database for the application based on the schema discussed in PMD. |
| <input type="checkbox"/> 4. Add data to the database. All data should be work appropriate. |
| <input type="checkbox"/> 5. The project history is tracked using Git and the code base is stored in a public repository on Github. (Note: ensure that you are not pushing usernames or passwords) |
| <input type="checkbox"/> 6. Create an ASP.NET Web Application for the UI layer of your project. |
| <input type="checkbox"/> 7. Implement authentication using Identity Samples. |
| <input type="checkbox"/> 8. Select a multi-page template, download the files, and bring them into an Archive folder in your UI project. |
| <input type="checkbox"/> 9. Create the data layer and utilize Entity Framework to create domain models for your database objects. |
| <input type="checkbox"/> 10. Update the connection strings in the web.config of your UI layer. |

50% COMPLETE:

- | |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <input type="checkbox"/> 11. Copy all of the template resources (JS, CSS, Fonts, etc) out of the archive folder and paste them into the appropriate folder in the UI layer. |
| <input type="checkbox"/> 12. Rename the _Layout to _OriginalLayout and then utilize the appropriate HTML file from the template to create a new _Layout. |
| <input type="checkbox"/> 13. Update all navigation links and image file paths. |

1. Each of us created a Trello board and Invited one another to see said board
2. Ashley documented and took screen shots of project
3. Ashley built database, created and shared Script and Schema with team
4. Each teammate added their own dummy data
5. Each teammate committed and pushed their changes to github
6. Corey created the ASP.NET web application and added Identity samples
7. Each teammate agreed on one template found on our FED1 bootstrap resources
8. Corey downloaded and added template to _archive folder
9. Bill created the data layer and metadata
10. Bill updated the connection strings
11. Ashley converted the template and moved CSS,JS,Fonts etc. to content and updated links.
12. Ashley updated and renamed _layoutpage
13. Corey updated all navigation links using ActionLinks

<input type="checkbox"/>	14. Update CSS and JS bundles to include all of the resources for the template. Remove any bundled resources that will not be used. Link to the Styles and Scripts bundles on the _Layout page.
<input type="checkbox"/>	15. Build metadata buddy classes in the data layer.
<input type="checkbox"/>	16. Scaffold out all controllers and views in the UI layer.
<input type="checkbox"/>	17. Build a contact form and implement the functionality to send form submissions to your email.

75% COMPLETE:

<input type="checkbox"/>	18. Create and utilize custom properties to show a summary of information about scheduled classes and to display the students full name.
<input type="checkbox"/>	19. Implement image upload functionality for the student create and edit forms.
<input type="checkbox"/>	20. Implement soft deletes for students, courses, and scheduled classes.
<input type="checkbox"/>	21. Utilize a session variable in the course workflow to define where the user should be sent (list of active or retired courses) when they click on the cancel button for edit or back on details.
<input type="checkbox"/>	22. Utilize session variables to define where a user should be sent after a soft delete.
<input type="checkbox"/>	23. Utilize a session variable to define the value of the status of a scheduled class status shown in the delete view.
<input type="checkbox"/>	24. Add functionality to allow the user to view the index of the Student Status controller in a table layout or tiled layout.

100% COMPLETE:

<input type="checkbox"/>	25. Create documentation for the project and link to the documentation from the Home/Index of the project.
<input type="checkbox"/>	26. Create a backup script of the database.
<input type="checkbox"/>	27. Ensure all code is pushed to the Github repo. (Note: ensure that you are not pushing usernames or passwords)
<input type="checkbox"/>	28. Each team member should create a live database and subdomain on SmarterASP. They should execute the backup script on the live database, deploy the project files, and link to the live project from their personal site. The description of this project should include details on that person's individual contributions.

14. Ashley removed any resources that would not be used
15. Bill built metadata buddy classes
16. Corey scaffolded out the controllers and CRUD functionality using MVC 5 Controller with views, using entity framework
17. Bill built out a contact form that is linked to his email. Fully functional
18. Bill customized properties to display full name
19. Ashley implemented photo upload functionality
20. We did not use this as one of our SAT challenges (soft delete)
21. Bill added a session variable to switch from active or retired courses
22. We did not implement a soft delete as one of our SAT challenges
23. We did not implement session variables for scheduled class status
24. Bill added the tile view for students so you are able to switch back and forth between tiled or table view
25. Ashley created documentation
26. Ashley created backup script of the database
27. Each team member has pushed all code to github
28. Each team member has created a subdomain on smarterasp and deployed the project through filezilla to add to their personal site

Identity Matrix used to determine what certain users can see

HomeController	Index	Contact				
Anonymous	x	x				
Scheduling	x	x				
Admin	x	x				
Courses	Active	Retired	Detail	Create	Edit	Delete
Anonymous						
Scheduling						
Admin	x	x	x	x	x	x
Enrollments	Index	Details	Create	Edit	Delete	
Anonymous						
Scheduling	x	x	x	x	x	
Admin	x	x	x	x	x	
Scheduled Classes	Index	Details	Create	Edit	Delete	
Anonymous						
Scheduling	x	x	x	x		
Admin	x	x	x	x	x	
Students	Index	Details	Create	Edit	Delete	
Anonymous						
Scheduling	x	x				
Admin	x	x	x	x	x	
Student Statuses	Index	Details	Create	Edit	Delete	
Anonymous						
Scheduling						
Admin	x	x	x	x	x	

Created new log ins that connected to SQL from VS

Id	Email	EmailConfirmed	PasswordHash	SecurityStamp
a2a56bd3-3795-4d3c-876c-467933f07497	Admin@fake.com	0	ABhsfkFMbdc3sH9GEFDNqPLHencnvsDNgi4ZjNObtdVThihga...	8a688469-1270-45a6-9460-4d7e8a0771ed
ec072676-0ba0-4432-b3a3-c5dce33e4076	Scheduling@fake.com	0	ADzsawOIykCrK+Dv/1Dnyvk3GrbB5pKOeHVAdnCLk6eC2onn...	751ca217-4e8a-4553-8432-7794263ed2eb

DAY
Home
Contact
Enrollments
ScheduledClasses
Students
Course
StudentStatus
RoleAdmin
UserAdmin
Log off
Hello Admin@fake.com

Contact Us

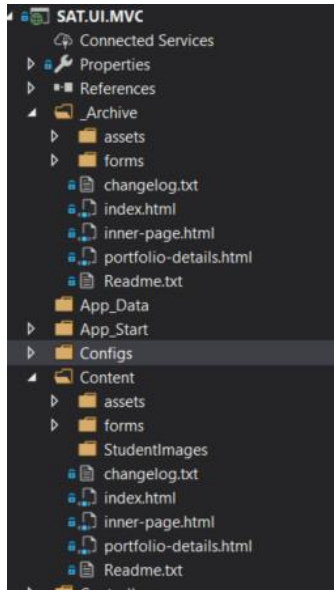
Name:

Email:

Message:

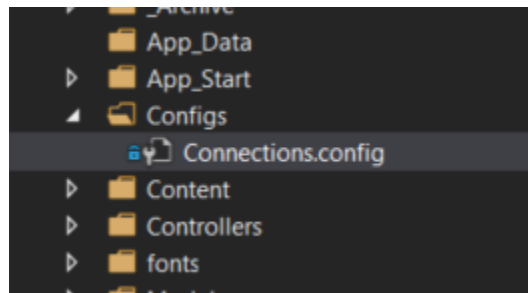
Submit

Added fully functional contact form. This will go directly to Bill's email

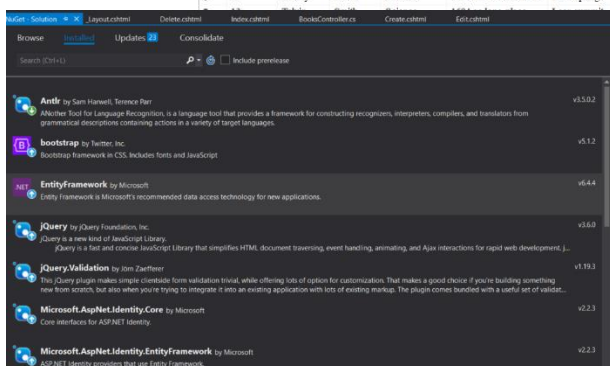
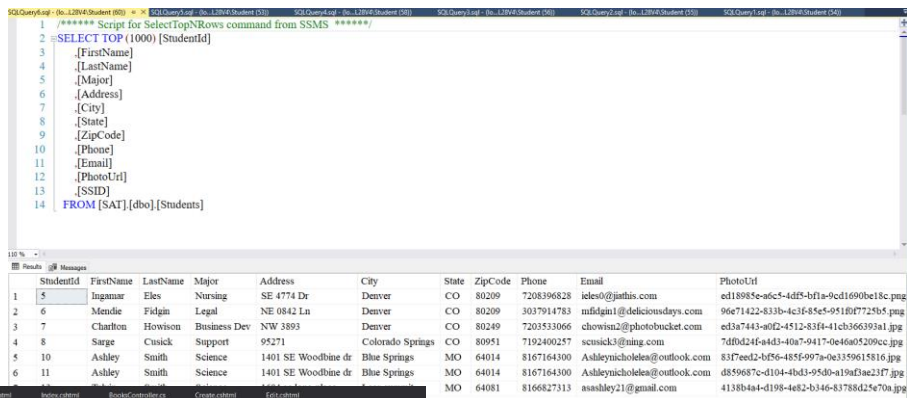


Moved CSS, JS, Images etc. from _Archive to Content folder to use in template conversion

Added Connections.config to hide connection strings



Added Dummy Data



Added Nuget Package Identity Samples

- MetaData
- App.Config
- packages.config
- SATModel.edmx
 - SATModel.Context.tt
 - SATModel.Designer.cs
 - SATModel.edmx.diagram
 - SATModel.tt
- SAT.UI.MVC**
 - Connected Services
 - Properties
 - References
 - _Archive
 - App_Data
 - App_Start
 - Configs
 - Content
 - Controllers
 - AccountController.cs
 - CoursController.cs
 - EnrollmentsController.cs
 - HomeController.cs
 - ManageController.cs
 - RolesAdminController.cs
 - ScheduledClassesController.cs
 - StudentsController.cs
 - StudentStatusController.cs
 - UserAdminController.cs

```
[MetadataType(typeof(StudentMetadata))]  
public partial class Student  
{  
    public string fullName  
    {  
        get { return FirstName + " " + LastName; }  
    }  
}  
#endregion
```


Phone

7208396828

Email

ieles0@jiathis.com

Student Photo



Choose File No file chosen

SSID

Current Student

Save Cancel

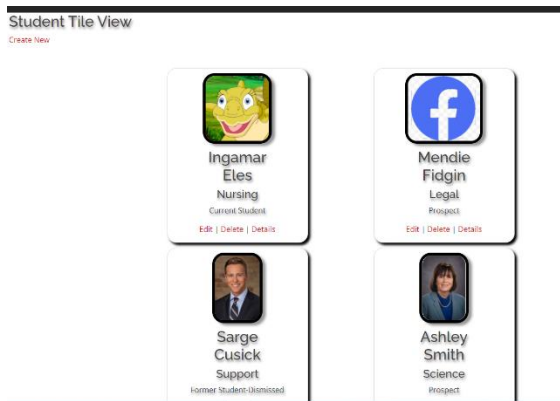
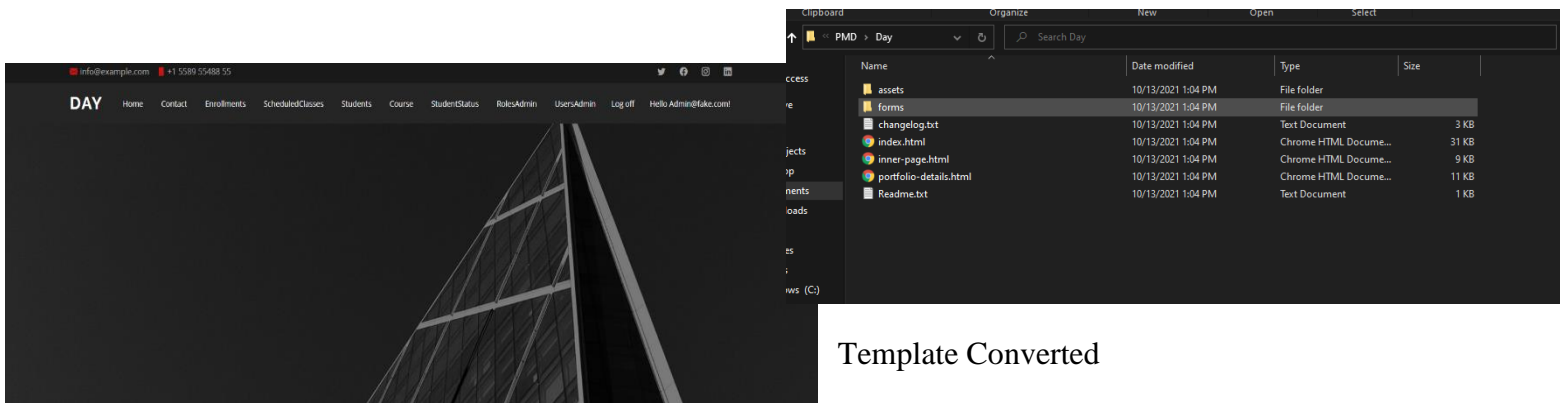
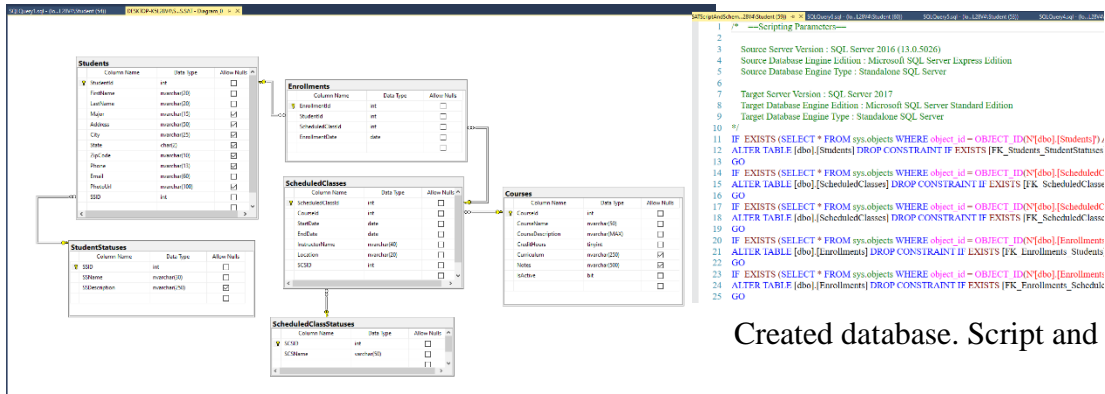
[illegible]

```
Student student = Student.findById(studentId);
if (ModelState.IsValid)
{
    Region File Upload
    if (PhotoUrl != null)
    {
        string imageName = PhotoUrl.FileName;
        string ext = imageName.Substring(imageName.LastIndexOf(".", 1));
        string[] goodExt = new string[] { ".jpg", ".png", ".jpeg", ".gif" };
        if (goodExt.Contains(ext.ToLower()))
        {
            imageName = Guid.NewGuid() + ext;
            PhotoUrl.SaveAs(Server.MapPath("~/Content/assets/img/Student/" + imageName));
            if (student.PhotoUrl != "noImage.png" && student.PhotoUrl != null)
            {
                System.IO.File.Delete(Server.MapPath("~/Content/assets/img/Student/" + student.PhotoUrl));
            }
            student.PhotoUrl = imageName;
        }
    }
}
```

Course Name	Course Description
<div><div><div></div></div><div>Algebra is a branch of mathematics dealing with symbols and the rules for manipulating those symbols.</div></div>	Algebra is a branch of symbols and the rules symbols.
<div>Action</div> <div><div>Edit</div><div>Details</div><div>Delete</div><div>Retired</div><div>Active</div></div>	

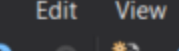
```
</td>
<td>
    @Html.ActionLink("Edit", "Edit", new { id = item.CourseId }) |
    @Html.ActionLink("Details", "Details", new { id = item.CourseId }) |
    @Html.ActionLink("Delete", "Delete", new { id = item.CourseId }) |
    @Html.ActionLink("Retired", "Index", new { id = item.CourseId }) |
    @Html.ActionLink("Active", "Index", new { id = 0 })
</td>
</tr>
```

Added a session variable to switch from active or retired courses



Updated Link

```
<nav id="navbar" class="navbar navbar-dark">
    <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">About</a></li>
        <li><a href="#">Contact</a></li>
        <li><a href="#">Account</a></li>
        <li><a href="#">Enrollments</a></li>
        <li><a href="#">ScheduledClasses</a></li>
        <li><a href="#">Students</a></li>
        <li><a href="#">Courses</a></li>
        <li><a href="#">StudentStatus</a></li>
        <li><a href="#">RolesAdmin</a></li>
        <li><a href="#">UsersAdmin</a></li>
        <li><a href="#">Logout</a></li>
    </ul>
</nav>
```



Trello board for Ashley, Bill & Corey

