SAT Application

1. Overview

Project Name: SAT Project

Purpose and Scope:

Create an application to track courses that a school might offer using MVC architecture.

Team Members: John Black & Gabriel Ramirez

2. Project Management

Trello Board Overview:

The Trello board consists of four main categories for each task. Backlog, In Progress, Testing, and Done.

Backlog – A list of tasks that need to be completed.

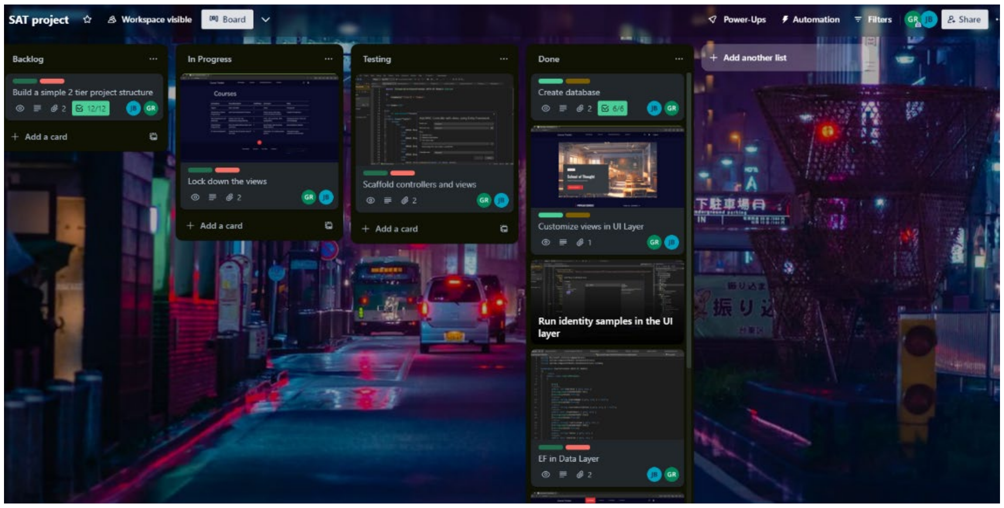
Progress – Task that currently being worked on

Testing – Task that are being tested for functionality.

Done – Task Completed

Task Management:

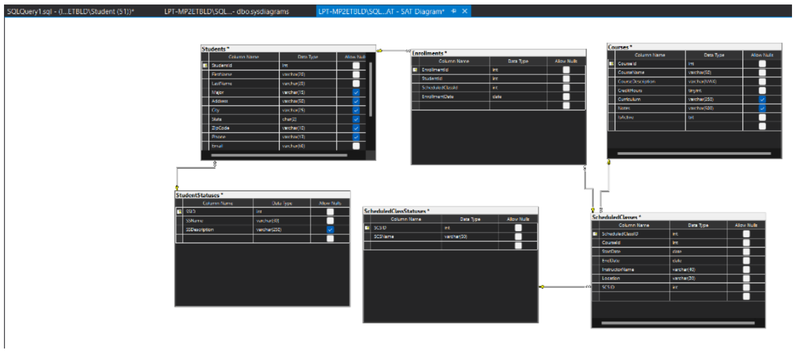
Tasks were completed using the pair programming technique and by alternating the roles of the Driver and Navigator.



3. Development Process

Database Design:

The database design was given and just needed to be implemented using SQL Server Management Studio. This required six tables to be created and mapped out for relationships. The main tables included: Students, StudentStatuses, Enrollments, ScheduledClassStatuses, Courses, and ScheduledClasses.


SQLQuery1.sql - (I...ETBLD\Student (51))*
LPT-MP2ETBLD\SQL...- dbo.sysdiagrams
LPT-MP2ETBLD\SQL...AT - SAT Diagram*

Students *

Column Name	Data Type	Allow Nulls
StudentId	int	
FirstName	varchar(20)	
LastName	varchar(20)	
Major	varchar(15)	✓
Address	varchar(50)	✓
City	varchar(25)	✓
State	char(2)	✓
ZipCode	varchar(10)	✓
Phone	varchar(13)	✓
Email	varchar(60)	

Enrollments *

Column Name	Data Type	Allow Nulls
EnrollmentId	int	
StudentId	int	
ScheduledClassId	int	
EnrollmentDate	date	

Courses *

Column Name	Data Type	Allow Nulls
CourseId	int	
CourseName	varchar(50)	
CourseDescription	varchar(MAX)	
CreditHours	tinyint	
Curriculum	varchar(250)	✓
Notes	varchar(500)	✓
IsActive	bit	

StudentStatuses *

Column Name	Data Type	Allow Nulls
SSID	int	
SSName	varchar(50)	
SSDescription	varchar(250)	✓

ScheduledClassStatuses *

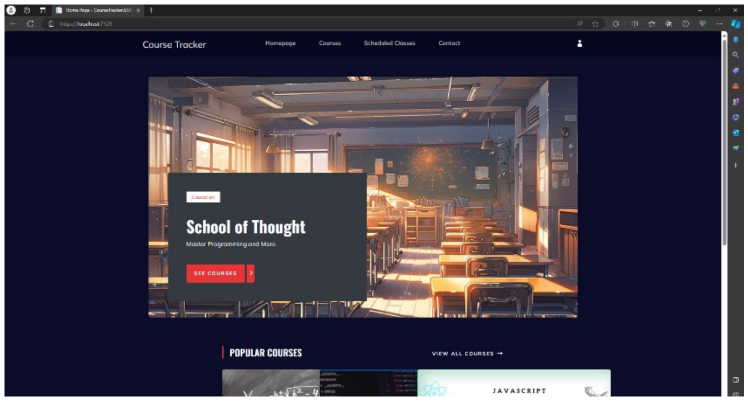
Column Name	Data Type	Allow Nulls
SCSID	int	
SCSName	varchar(50)	

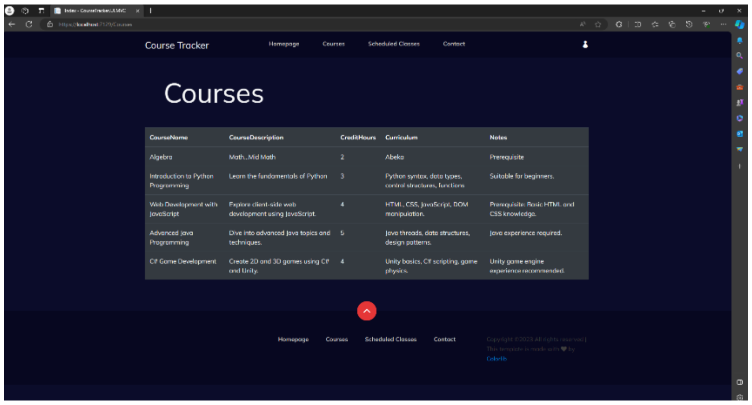
ScheduledClasses *

Column Name	Data Type	Allow Nulls
ScheduledClassID	int	
CourseId	int	
StartDate	date	
EndDate	date	
InstructorName	varchar(40)	
Location	varchar(20)	
SCSID	int	

UI Layer:

The application utilized a bootstrap template for the Views.





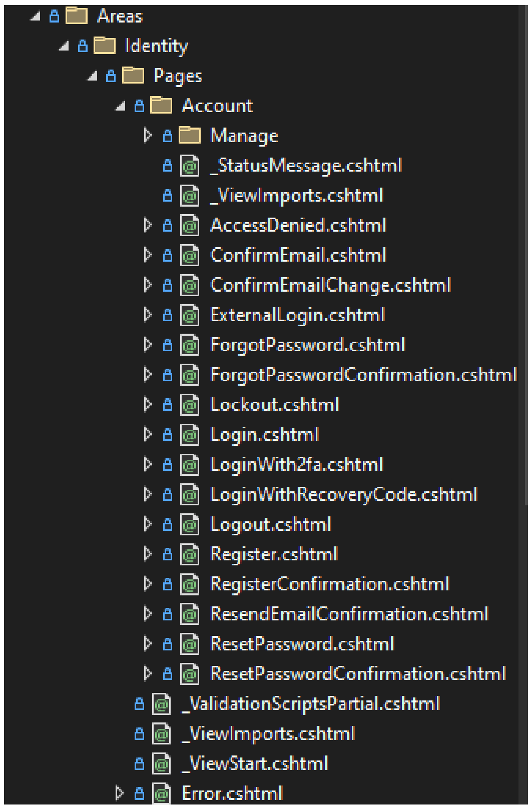
Courses

CourseName	CourseDescription	CreditHours	Curriculum	Notes
Algebra	Math...Mid Math	2	Abeka	Prerequisite
Introduction to Python Programming	Learn the fundamentals of Python	3	Python syntax, data types, control structures, functions	Suitable for beginners.
Web Development with JavaScript	Explore client-side web development using JavaScript.	4	HTML, CSS, JavaScript, DOM manipulation.	Prerequisite: Basic HTML and CSS knowledge.
Advanced Java Programming	Dive into advanced Java topics and techniques.	5	Java threads, data structures, design patterns.	Java experience required.
C# Game Development	Create 2D and 3D games using C# and Unity.	4	Unity basics, C# scripting, game physics.	Unity game engine experience recommended.

Copyright ©2023 All rights reserved | This template is made with ♥ by Colorlib

Authentication and User Management:

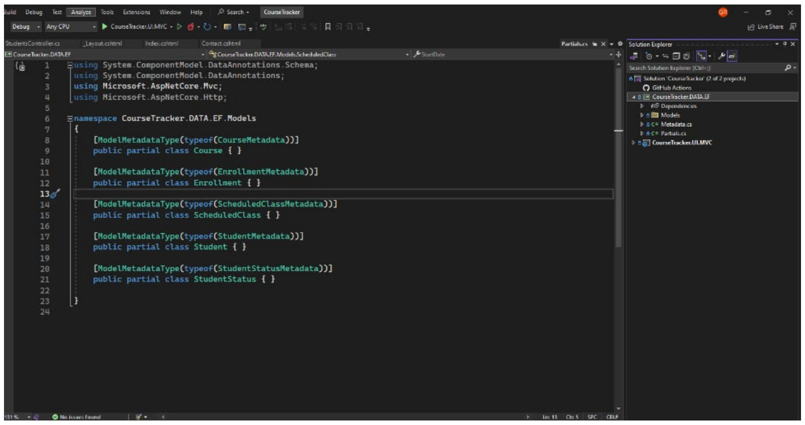
Authentication and user management roles were handled through ASP.NET Core Identity. This added the login functionality for user registration, password, and role management.



```
Areas
  Identity
    Pages
      Account
        Manage
        _StatusMessage.cshtml
        _ViewImports.cshtml
        AccessDenied.cshtml
        ConfirmEmail.cshtml
        ConfirmEmailChange.cshtml
        ExternalLogin.cshtml
        ForgotPassword.cshtml
        ForgotPasswordConfirmation.cshtml
        Lockout.cshtml
        Login.cshtml
        LoginWith2fa.cshtml
        LoginWithRecoveryCode.cshtml
        Logout.cshtml
        Register.cshtml
        RegisterConfirmation.cshtml
        ResendEmailConfirmation.cshtml
        ResetPassword.cshtml
        ResetPasswordConfirmation.cshtml
      _ValidationScriptsPartial.cshtml
      _ViewImports.cshtml
      _ViewStart.cshtml
      Error.cshtml
```

Data Layer and Entity Framework:

The models for the DATA.EF project was brought in by scaffolding the SA database. After scaffolding, there two classes added. Metadata which held properties that would allow data to be passed/viewed from the connected database. A Partial class was created to make the code manageable and easy to read which allowed for their definitions to be split across multiple files.



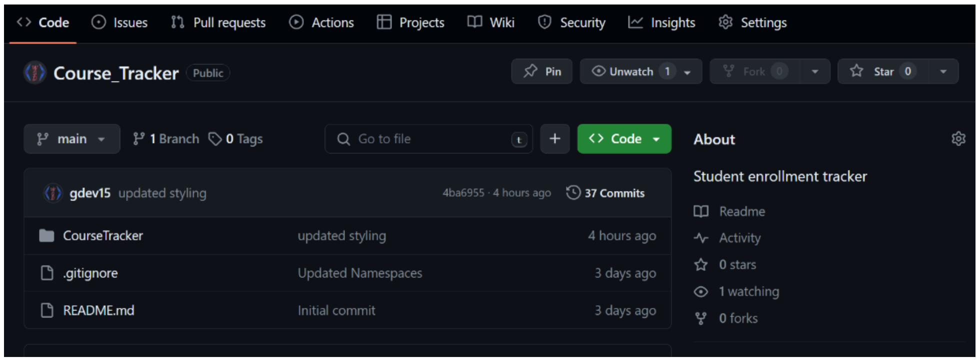
```
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Http;

namespace CourseTracker.DATA.EF.Models
{
    [ModelMetadataType(typeof(CourseMetadata))]
    public partial class Course { }

    [ModelMetadataType(typeof(EnrollmentMetadata))]
    public partial class Enrollment { }

    [ModelMetadataType(typeof(ScheduledClassMetadata))]
    public partial class ScheduledClass { }

    [ModelMetadataType(typeof(StudentMetadata))]
    public partial class Student { }

    [ModelMetadataType(typeof(StudentStatusMetadata))]
    public partial class StudentStatus { }

}
```

4. Code Management

Git and GitHub:

The project utilized Github for source control.

https://github.com/gdev15/Course_Tracker



gdev15 updated styling 4ba6955 · 4 hours ago 37 Commits

CourseTracker	updated styling	4 hours ago
.gitignore	Updated Namespaces	3 days ago
README.md	Initial commit	3 days ago

Student enrollment tracker

Readme
Activity
0 stars
1 watching
0 forks

5. Implementation Details

Connection Strings:

Updated the connection strings in appsettings.json.

Resource Management:

Managed JS, CSS, Fonts, etc., from the template.

Layout and Navigation:

Extracted containers that were necessary and modified navigation links.

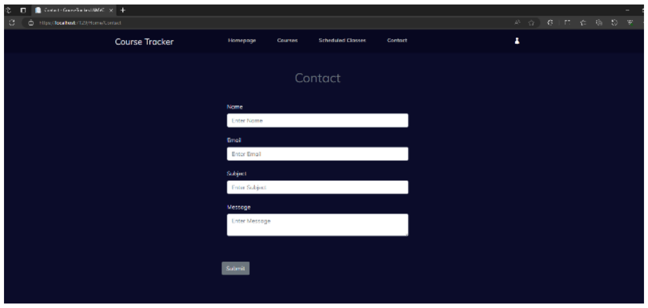
Metadata and Scaffolding:

Metadata was extensively used for purposes like data validation, setting display formats, and providing descriptions or titles for properties in web forms.

Scaffolding was used to automatically produce boilerplate code for basic CRUD (Create, Read, Update, Delete) operations in applications.

6. Features and Functionality

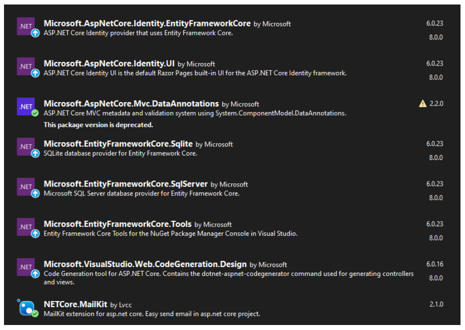
Contact Form: Email functionality was implemented by leveraging the NETCore.MailKit package.



7. Setup and Installation

Package Management:

- Microsoft.AspNetCore.Identity.EntityFrameworkCore
- Microsoft.AspNetCore.Identity.UI
- Microsoft.AspNetCore.MVC.DataAnnotations
- Microsoft.EntityFrameworkCore.Sqlite
- Microsoft.EntityFrameworkCore.SqlServer
- Microsoft.EntityFrameworkCore.Tools
- Microsoft.VisualStudio.Web.CodeGeneration.Design
- NETCore.MailKit



Microsoft.AspNetCore.Identity.EntityFrameworkCore by Microsoft — 6.0.23 / 8.0.0
ASP.NET Core Identity provider that uses Entity Framework Core.

Microsoft.AspNetCore.Identity.UI by Microsoft — 6.0.23 / 8.0.0
ASP.NET Core Identity UI is the default Razor Pages built-in UI for the ASP.NET Core Identity framework.

Microsoft.AspNetCore.Mvc.DataAnnotations by Microsoft — ⚠ 2.2.0
ASP.NET Core MVC metadata and validation system using System.ComponentModel.DataAnnotations.
This package version is deprecated.

Microsoft.EntityFrameworkCore.Sqlite by Microsoft — 6.0.23 / 8.0.0
SQLite database provider for Entity Framework Core.

Microsoft.EntityFrameworkCore.SqlServer by Microsoft — 6.0.23 / 8.0.0
Microsoft SQL Server database provider for Entity Framework Core.

Microsoft.EntityFrameworkCore.Tools by Microsoft — 6.0.23 / 8.0.0
Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.

Microsoft.VisualStudio.Web.CodeGeneration.Design by Microsoft — 6.0.16 / 8.0.0
Code Generation tool for ASP.NET Core. Contains the dotnet-aspnet-codegenerator command used for generating controllers and views.

NETCore.MailKit by Lvcc — 2.1.0
MailKit extension for asp.net core. Easy send email in asp.net core project.

Installation Guide:

- Download the source code through github
- Setup create database
- Update Appsettings.json connection strings to point to the local db