DISSERTATION CALCULATOR TOOL FOR TRACKING AND MANAGING DISSERTATION PROGRESS FOR GRADUATE STUDENTS

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Bat-Od Bat-Otgon

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Program:
Software Engineering

November 2019

Fargo, North Dakota

North Dakota State University
Graduate School

Title

Dissertation Calculator Tool for Tracking and Managing Dissertation Progress for Graduate Students

By

Bat-Od Bat-Otgon

The Supervisory Committee certifies that this *disquisition* complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Gursimran S. Walia
Chair

Dr. Pratap Kotala

Dr. Limin Zhang

Approved:

11/19/2019	Dr. Kendall E. Nygard
Date	Department Chair

ABSTRACT

Dissertation Calculator is a tool for graduate students who are working towards their graduate degree at North Dakota State University to manage and track their dissertation progress. It is also a communication and feedback tool between students and their advisers. It was developed using ASP.NET Core Framework. This project is based on Model-View-Controller and Client-Server model. There are 3 different stakeholders which are student, adviser, and admin so each user has a different view. All the information that is created, modified, and deleted by the users is stored on a database and the host application manages the access to it so it is also incorporates Client-Server model. The client side will be any device with a browser. This tool provides essential help for graduate students who wants to make sure they are on track to complete their dissertation and get useful feedback from their adviser in a timely manner.

ACKNOWLEDGEMENTS

I would like to thank Dr. Gursimran Walia for his guidance and support while I was working on this paper. They were extremely valuable in completing this paper and one of the reasons why this paper exists in the first place.

I would like to thank Dr. Pratap Kotala and Dr. Limin Zhang for agreeing and taking the time to be part of my graduate committee.

I would like to thank Danjel Nygard from Graduate School and Enrico Sassi from Graduate Center for Writers for their valuable input and feedback on completing this project for the paper.

DEDICATION

I would like to dedicate this to my wife and children for their unending love and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF APPENDIX FIGURES

1. INTRODUCTION

For graduate students, dissertation completion is a very crucial part of their program. It can be very daunting process with multiple steps to complete and keep track of. From the beginning of the process such as starting with understanding the expectations to the end of the process where you submit your dissertation to graduate school, the entire process can take very long time. These students can get overwhelmed by the number of steps to complete while taking classes and potentially fall behind with their schedule and miss their expected graduation date. This opens the opportunity to fulfill the need for better method to manage, track, and communicate changes or updates with advisers of the dissertation process with a specialized tool to make process easier and simpler.

With this paper, we are developing a tool called Dissertation Calculator tool to assist graduate students to manage, track, and communicate changes or updates to advisers at North Dakota State University. As it currently stands, there is no such tool available to assist students of NDSU thus the need for this specific tool was made apparent. The tool was requested by Graduate Center for Writers at NDSU for development to be used by graduate students. This paper will cover the entire process of how the tool was developed such as the design and functionalities with the help and input from Graduate School and Center for Writers.

1.1. Dissertation Calculator Tool

Dissertation Calculator is a web application tool to assist graduate students to help them manage and track their dissertation progress. It also provides a way to communicate updates and changes on each step of the progress with their adviser. Advisers can also provide useful feedback to students on each step assisting them to complete their dissertation. The main stakeholders of this tool are graduate students, advisers, and administrators. Each stakeholder will have certain number of features in the tool that they can use to perform specific functions.

Administrator users are most likely going to be employees of Graduate School and they can create, edit, and delete accounts, departments, base template timeline, and academic calendar dates and deadlines. The base templates are for advisers so that they can use it as a starting point for their own template that they can make for students to use. Academic calendars dates and deadlines are very important specifically for graduate students who wants to finish their dissertation on time. Admin user can enter specific dates into the system so that it will show up student's home page to remind them.

Adviser users are faculty who are advising a graduate student in their department and working with them towards completing a dissertation. They can view students who have selected them as their adviser and view their dissertation progress timeline. They can also make comments on each step of the timeline to give students updates, suggestions and provide valuable feedback on time. One of the main functionalities that adviser user can perform is to create a template timeline that has steps already created with description, and recommended duration of days each step should take for the students to use to create their own student timeline based on the template. Adviser users can create these templates from already available templates from the system made by administrators or make them manually.

Student users are any graduate student who are working towards completing their dissertation for their master's or doctoral degree. They will only be able to access the student functionalities such as creating a timeline to keep track and manage their dissertation progress, either manually or based on the template that their adviser has created. The timeline will consist of steps, step description, start date, end date, comments, and completed status. On the timeline, student can add steps, remove steps, and edit steps. Student user can initially set their expected graduation date and their adviser when their first login to the system with their account. Students can change these options later if needed.

There are few things that each stakeholder can do as shared functionality such as registering their account, login into the system, resetting forgotten password, changing their profile information, and login out of the system.

1.2. Paper Organization

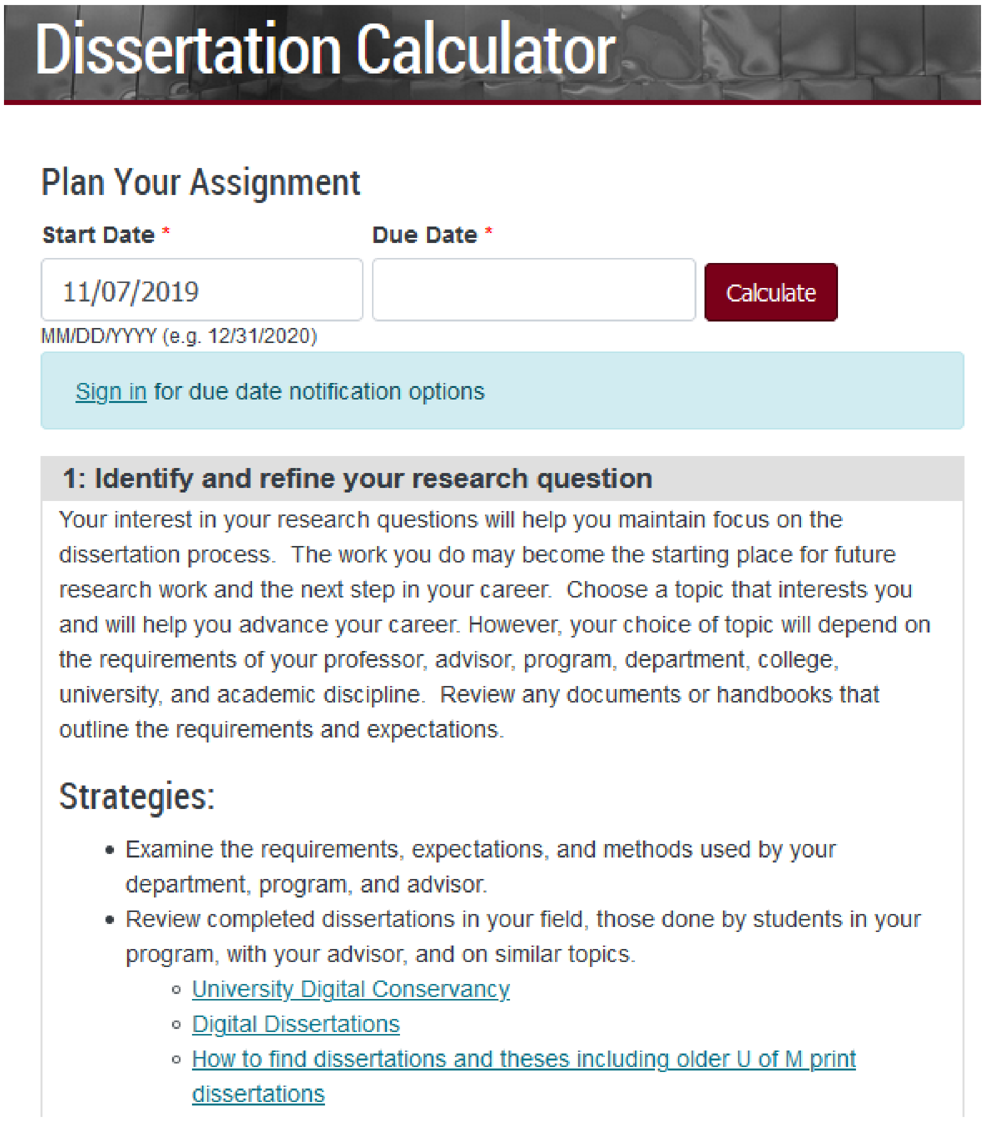
This paper provides background about existing dissertation calculator tools that are used by other education institutions in section 2, details on design in section 3, development process of the tool and its features in section 4, the conclusion and lessons learned from the project will be discussed in section 5, and lastly future work needed for the tool is discussed in section 6.

2. BACKGROUND

There is no easy way to find which educational institutions use what kind of specific tool for dissertation management and tracking progress, as some tools that they use might be only available internally and not publicly available in search engines. Also, there is no specific tool made available to public for students and advisers to use as well. However, there are some tools used by educational institutions such as University of Kentucky, University of Minnesota, Rochester Institute of Technology, Baker University, University of Toronto, University of Missouri that can be looked up by search engines. These tools all work in the same manner by entering an expected due date or date to complete and get number of steps that show a date that the user needs to complete by and description of each step that includes additional information by providing links to other locations. The search terms “Dissertation Calculator”, “Dissertation Tracker”, or “Dissertation Manager” don’t yield useful results apart from what we discovered above. Out of all the tools, one made my University of Minnesota is the most comprehensive one with detailed description, ease of use, and best look and feel.

2.1. Dissertation Calculator by University of Minnesota

Dissertation Calculator tool made my University of Minnesota is referenced in the other tools made by other schools. Most of them have based of their tools on the tool by University of Minnesota. In this tool, you can enter a start date and a due date. Based on these two inputs, the tool gives a timeline of steps or stages of your dissertation progress to complete by a specific date on each steps or stages. Each step includes strategies to complete the step and links to other useful information. It also includes a section called “Tips from the Libraries” where certain specific tips that may apply for the specific step or stage. Each step or stage has a number and how much percentage of time must be spent on it.


Dissertation Calculator

Plan Your Assignment

Start Date * Due Date *

11/07/2019

MM/DD/YYYY (e.g. 12/31/2020)

Calculate

Sign in for due date notification options

1: Identify and refine your research question

Your interest in your research questions will help you maintain focus on the dissertation process. The work you do may become the starting place for future research work and the next step in your career. Choose a topic that interests you and will help you advance your career. However, your choice of topic will depend on the requirements of your professor, advisor, program, department, college, university, and academic discipline. Review any documents or handbooks that outline the requirements and expectations.

Strategies:

- Examine the requirements, expectations, and methods used by your department, program, and advisor.
- Review completed dissertations in your field, those done by students in your program, with your advisor, and on similar topics.
  - University Digital Conservancy
  - Digital Dissertations
  - How to find dissertations and theses including older U of M print dissertations

Figure 2.1. Dissertation Calculator - University of Minnesota

Based off on this brief research, there are no tools like Dissertation Calculator exists, at least from public view or at North Dakota State University. The only tool that may have come close is the Dissertation Calculator tool by University of Minnesota. These tools are all trying to simplify the tracking and managing of dissertation progress easier, but Dissertation Calculator tool discussed in this paper's goal is to make things even more simpler to track and manage and introduce interaction between the student and the adviser by providing more functionality such as commenting, selecting adviser, creating templates, etc.

3. DESIGN

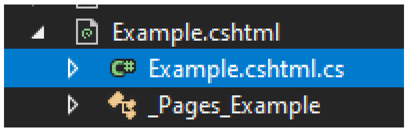
In this section, we will cover the design of Dissertation Calculator tool developed for this paper.

3.1. Framework

The web application is based on ASP.NET Core framework. ASP.NET Core provides two ways to develop a fully featured web application which are MVC web application or Razor Pages web application. Razor Pages is the sub category of this framework which the web application tool of this paper is developed on. It is based on MVC model but much simpler as it provides page focused development and combines the model and controller to make development simpler.

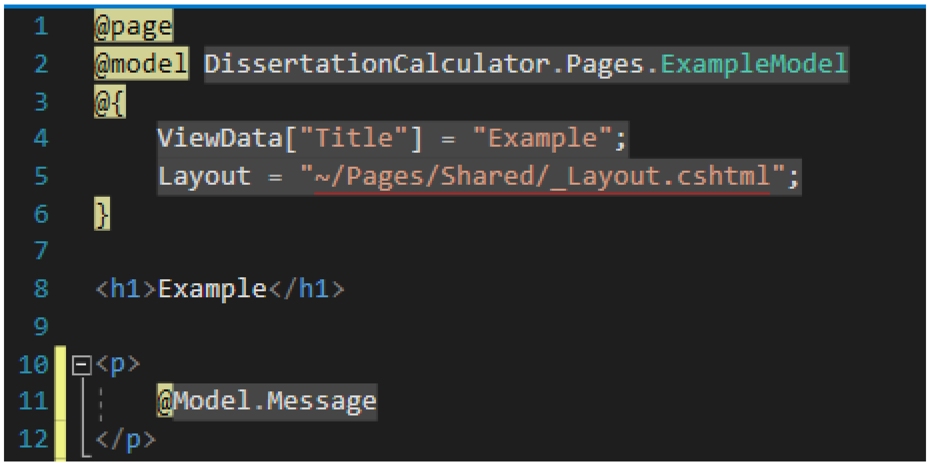
Benefits include of using ASP.NET Core include, a unified story for building web UI and web APIs, architected for testability, razor Pages makes coding page-focused scenarios easier and more productive, Blazor lets you use C# in the browser alongside JavaScript, share server-side and client-side app logic all written with .NET, ability to develop and run on Windows, macOS, and Linux, open-source and community-focused, integration of modern, client-side frameworks and development workflows, a cloud-ready, environment-based configuration system, built-in dependency injection, a lightweight, high-performance, and modular HTTP request pipeline, ability to host on IIS, Nginx, Apache, Docker, or self-host in your own process, side-by-side app versioning when targeting .NET Core, tooling that simplifies modern web development.

Razor Page files consist of two files to make up the page model. First file is ".cshtml" which handles the HTML, CSS, JavaScript codes and second file is ".cshtml.cs" file which handles model and controller part of the code which is written in C#.



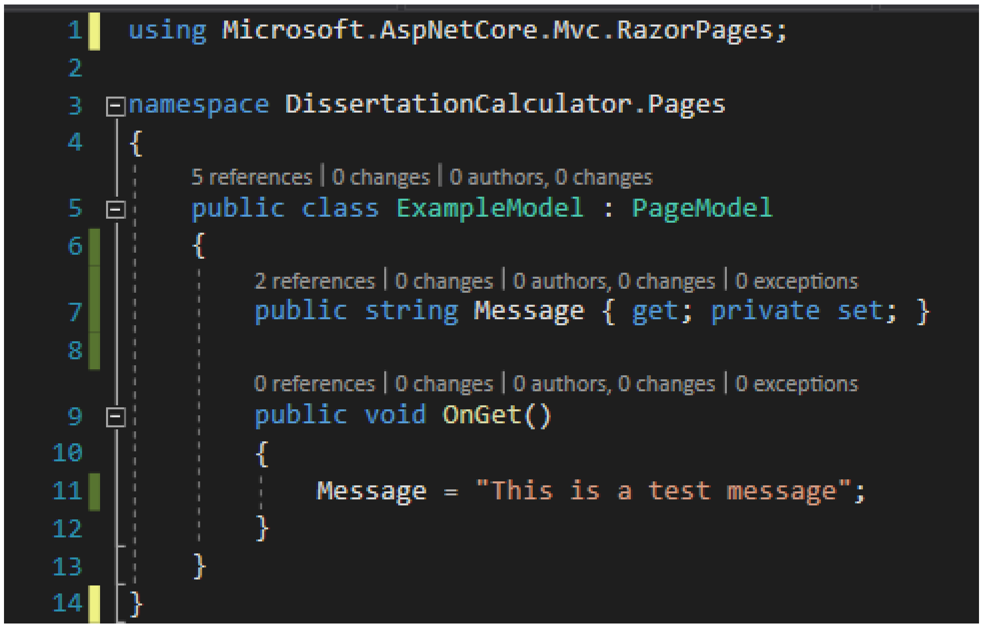
```
◢ Example.cshtml
    ▷ Example.cshtml.cs
    ▷ _Pages_Example
```

Figure 3.1. Example cshtml file organization



```
@page
@model DissertationCalculator.Pages.ExampleModel
@{
    ViewData["Title"] = "Example";
    Layout = "~/Pages/Shared/_Layout.cshtml";
}

<h1>Example</h1>

<p>
    @Model.Message
</p>
```

Figure 3.2. Example.cshtml



```
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace DissertationCalculator.Pages
{
    5 references | 0 changes | 0 authors, 0 changes
    public class ExampleModel : PageModel
    {
        2 references | 0 changes | 0 authors, 0 changes | 0 exceptions
        public string Message { get; private set; }

        0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
        public void OnGet()
        {
            Message = "This is a test message";
        }
    }
}
```

Figure 3.3. Example.cshtml.cs

3.1.1. Entity Framework Core

This project uses a relational database. ASP.NET Core works with a technology called Entity Framework Core to assist with developing database models much simpler. It is a lightweight, extensible, open source, and cross-platform data access technology. It allows to create database objects using .NET objects and takes care of most of the data access code without us needing to write them ourselves. It supports many database engines such as SQL Server, SQLite, in-memory database, Azure Cosmos DB, PostgreSQL, MySQL, MyCAT, Oracle DB and so much more.

With Entity Framework, data access is performed using a model. There are number of ways to generate a model. First, you can generate your model from existing database which you already created in database provider already. Second, manually code your model to match your database. Lastly, you can use a process called Entity Framework Migrations to create database from your model. This project uses the last method to create databases for the system. All the databases are modelled in C# code and then migrated to a database table. EF migrations give the ability to migrate data model to an updated on as models can change during development. Database can get out of sync so dropping it and creating a new database introduces the problem of losing the data. Migrations provide a way to migrate the data from the old database model to the new database model without losing data.

With EF migration, you can create a migration, apply the changes to the database by updating it, customize the migration code so that the database update is exactly what you want, remove migration that is no longer needed, revert back migration changes, generate SQL scripts, and apply migrations at runtime when the application first starts.

3.2. Architecture

Dissertation Calculator tool is based on Model-View-Controller model mixed with Client-Server model. The reason for choosing MVC is that we have 3 different views by 3 different types of users such as students, advisers, and administrators. Each view can be separately developed and maintained easily. This model also provides modifiability by allowing easy changes to user interfaces. As for the client-server model side of things, the web application will use a centralized database to store all the information that will be used by students, advisers, and administrators so that is taken care of on server side. Client-Server model provided interoperability, modifiability, availability, and reusability qualities. Clients are any users with a browser that can use the web application. Modifiability allows centralized change on the server and clients will be able to use the changes right away. Server side will be the database, and the application hosting. This can be multiple nodes to allow for better availability. Database can be on a separate dedicated database server as well as be on the same server as the application server. It will depend on how the environment needs to be scaled and the data or the application could be used for different application easily for better reusability.

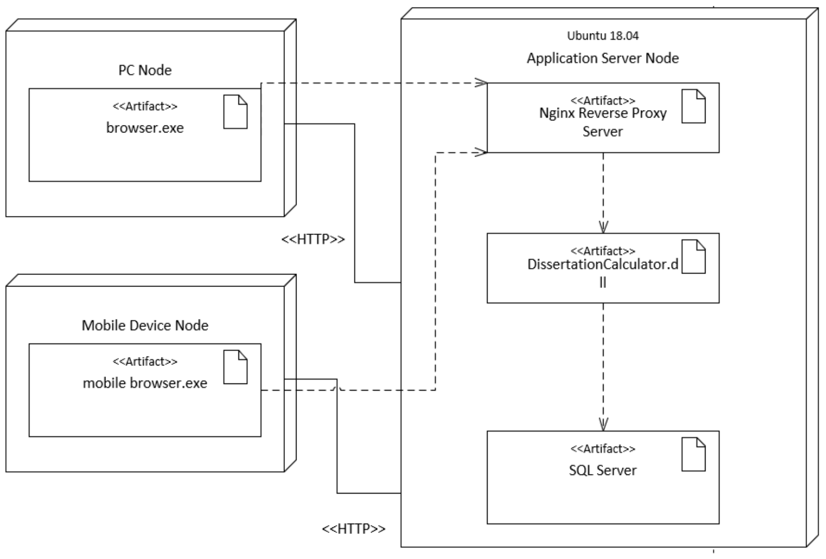


Figure 3.4. Architecture – Deployment Diagram

3.2.1. Database Server

Entity Framework Core supports many database providers but for this project we selected Microsoft SQL Server as the database provider. Microsoft SQL Server 2012 and onward versions are supported for this. Microsoft SQL Server 2017 for Linux running on Ubuntu server was used for this project, but it does not matter to the application where the database is located if a connection can be made as long as the database provider is supported by EF Core.

3.2.2. Application Server

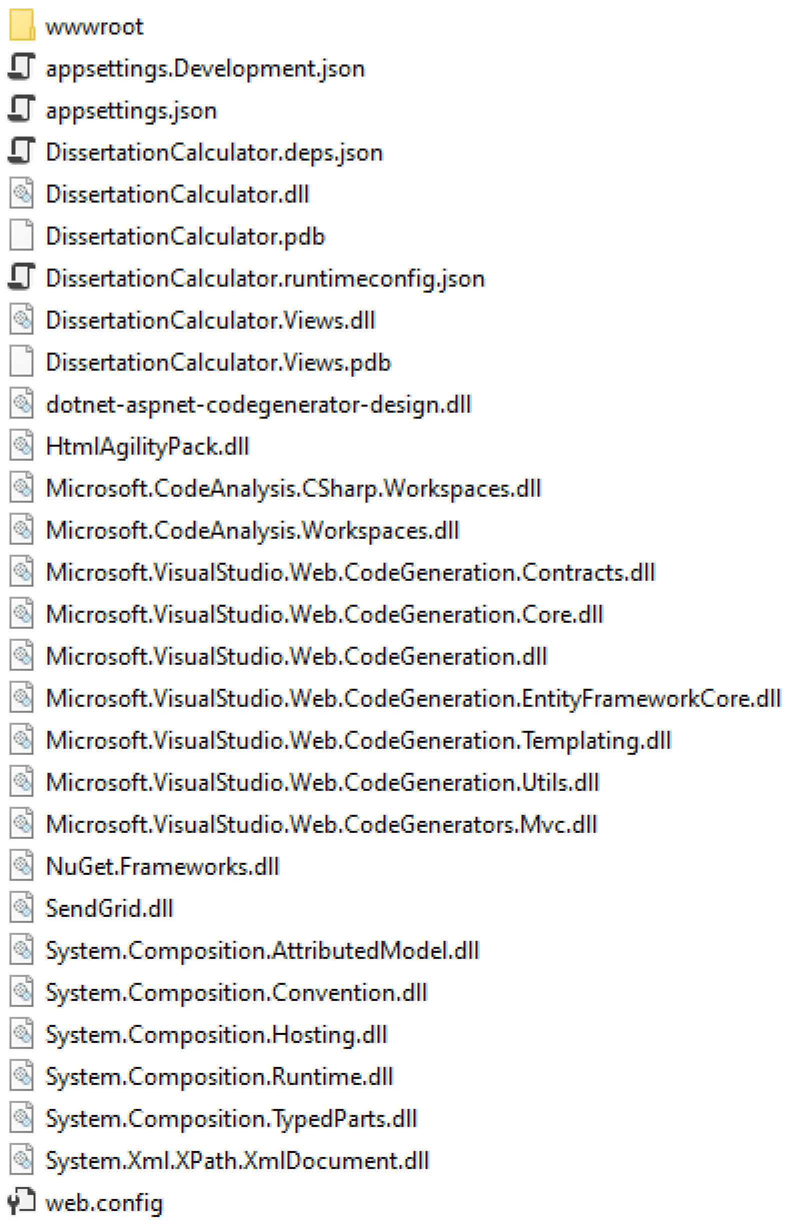
As mentioned before, ASP.NET Core is a cross-platform framework so it can run on Windows, Linux, macOS or in a Docker container. For development such as writing the code, Windows environment was used but for testing, deploying, and hosting the application, a Linux server running Ubuntu 18.04 was used. Nginx, a HTTP, and reverse proxy web server was used to monitor the application service and manage the traffic re-routing incoming traffic to the server to the application. Nginx can be also used for load balancing in case we have multiple nodes for the application server for better availability for the users.

3.2.3. Client

For client, any device with an internet or network connection to the application server will be able to connect using a browser. Google Chrome, Firefox, and Microsoft edge browsers were used to test the application and all three worked without any issues. Mobile device browsers are supported, and the UI of the application will scale accordingly.

3.3. DissertationCalculator.dll

DissertationCalculator.dll is the main application that is running on the server using .NET runtime. Each client request is made to this application via browser then database operations are made from application to the database.



```
wwwroot
appsettings.Development.json
appsettings.json
DissertationCalculator.deps.json
DissertationCalculator.dll
DissertationCalculator.pdb
DissertationCalculator.runtimeconfig.json
DissertationCalculator.Views.dll
DissertationCalculator.Views.pdb
dotnet-aspnet-codegenerator-design.dll
HtmlAgilityPack.dll
Microsoft.CodeAnalysis.CSharp.Workspaces.dll
Microsoft.CodeAnalysis.Workspaces.dll
Microsoft.VisualStudio.Web.CodeGeneration.Contracts.dll
Microsoft.VisualStudio.Web.CodeGeneration.Core.dll
Microsoft.VisualStudio.Web.CodeGeneration.dll
Microsoft.VisualStudio.Web.CodeGeneration.EntityFrameworkCore.dll
Microsoft.VisualStudio.Web.CodeGeneration.Templating.dll
Microsoft.VisualStudio.Web.CodeGeneration.Utils.dll
Microsoft.VisualStudio.Web.CodeGenerators.Mvc.dll
NuGet.Frameworks.dll
SendGrid.dll
System.Composition.AttributedModel.dll
System.Composition.Convention.dll
System.Composition.Hosting.dll
System.Composition.Runtime.dll
System.Composition.TypedParts.dll
System.Xml.XPath.XmlDocument.dll
web.config
```

Figure 3.5. Application Files

Most of the other dll files are required packages and libraries to run the application. “wwwroot” folder contains static files used by the application such as CSS files, JavaScript libraries such as jQuery and bootstrap.

3.4. Database Design

Entity Framework Core manages the data access in the application. It also makes the database modeling and creation much simpler as .NET objects. All the attributes are written as .NET objects in the code in the model classes. Each specific information that are created, modified, and deleted have specific database tables such as Users, Departments, Timelines, Comments, Dates and Deadlines. We used relational database in this project as it is easy to understand the relationship between related information. In this project, we have 17 relational database tables. 7 of which are used and generated automatically for authentication, authorization, and user account information such as usernames, email addresses, passwords, etc. These are provided by the framework and generated automatically. However, it can be customized to work with your application in specific ways you want.

There are 10 tables created using EF based on models from .NET classes. These are Academic Calendar Terms, Academic Calendar Term Date, Base Template Timeline, Base Template Timeline Steps, Comment, School Department, Student Timeline, Student Timeline Step, Template Timeline, and Template Timeline Step.

Academic Calendar Terms is used to model specific terms such as fall, spring, and summer terms in a school year. Academic Calendar Term Date is used to model the specific dates in each term that it may have such as dissertation submission deadline, final submission deadline, and graduation day etc.

Base Template Timeline is used to model a template that is available in the system to adviser users. It has time line name such as “Template for 2019-2020”. Each timeline will have steps so that is modeled by the class Base Template Timeline Step. This will be used to model, step name, step description, created date, created by information, and step duration.

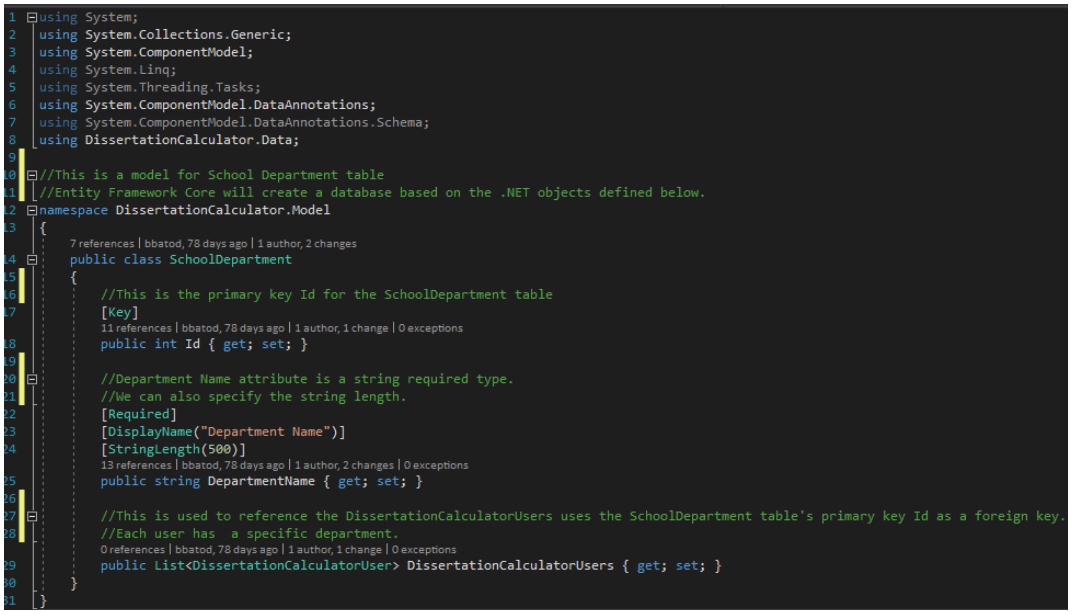
Comment is used to model the commenting system in the project. Each student timeline step has comments. Comments are modeled with comment creator, comment message, comment created time, and parent comment for reply feature.

School Department is used to model each specific department in a school. It will have department id and department name.

Student Timeline is used to model a specific timeline that a student will create. It has name, date created, and steps. The steps are modeled with the class Student Timeline Step class. It is used to model step number, step name, step description, start date, end date, completed status, and comments.

Template Timeline is used to model template timelines created by adviser users. It has template timeline name, date created, created by, and template timeline steps. Each step is modeled by the class Template Timeline Step. It has step number , step name, step description, and step duration.

The figure below is a snippet of modeling of school department table in .NET object for EF Core.



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Threading.Tasks;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using DissertationCalculator.Data;

//This is a model for School Department table
//Entity Framework Core will create a database based on the .NET objects defined below.
namespace DissertationCalculator.Model
{
    7 references | bbatod, 78 days ago | 1 author, 2 changes
    public class SchoolDepartment
    {
        //This is the primary key Id for the SchoolDepartment table
        [Key]
        11 references | bbatod, 78 days ago | 1 author, 1 change | 0 exceptions
        public int Id { get; set; }

        //Department Name attribute is a string required type.
        //We can also specify the string length.
        [Required]
        [DisplayName("Department Name")]
        [StringLength(500)]
        13 references | bbatod, 78 days ago | 1 author, 2 changes | 0 exceptions
        public string DepartmentName { get; set; }

        //This is used to reference the DissertationCalculatorUsers uses the SchoolDepartment table's primary key Id as a foreign key.
        //Each user has  a specific department.
        0 references | bbatod, 78 days ago | 1 author, 1 change | 0 exceptions
        public List<DissertationCalculatorUser> DissertationCalculatorUsers { get; set; }
    }
}
```

Figure 3.6. School Department Model for Database

3.5. Relational Database Diagram

The figure below is the relational database diagram for the tables used in the application. It shows which tables are related to which and what are the primary key and what tables have foreign keys in other tables as well. For all the tables, the attribute "Id" is the primary key.

When a table is referencing another table and has a foreign key, it is in the format of "TableNameId". For example, "StudentTimelineStep" table has a foreign key called "StudentTimelineId" so the table it is referencing is the "StudentTimeline" table. Another example is, "TemplateTimeline" table. It has the foreign key called "DissertationCalculatorUserId". The reason this is not using "AspNetUsersId" is because in the application code, we have inherited the identity user class and created our own called "DissertationCalculatorUserId" so EF Core automatically knows to map this foreign key to the primary key "Id" of "AspNetUsers" table. The figure 8 and 9 show the relational database diagram of the whole system. We have two separate diagrams because everything would not fit in one diagram. The common table for both is "AspNetUsers" which is the table for the containing all the information about the users of the system. Figure 9 is showing everything related to identity meaning tables used for authentication, authorization, and roles.

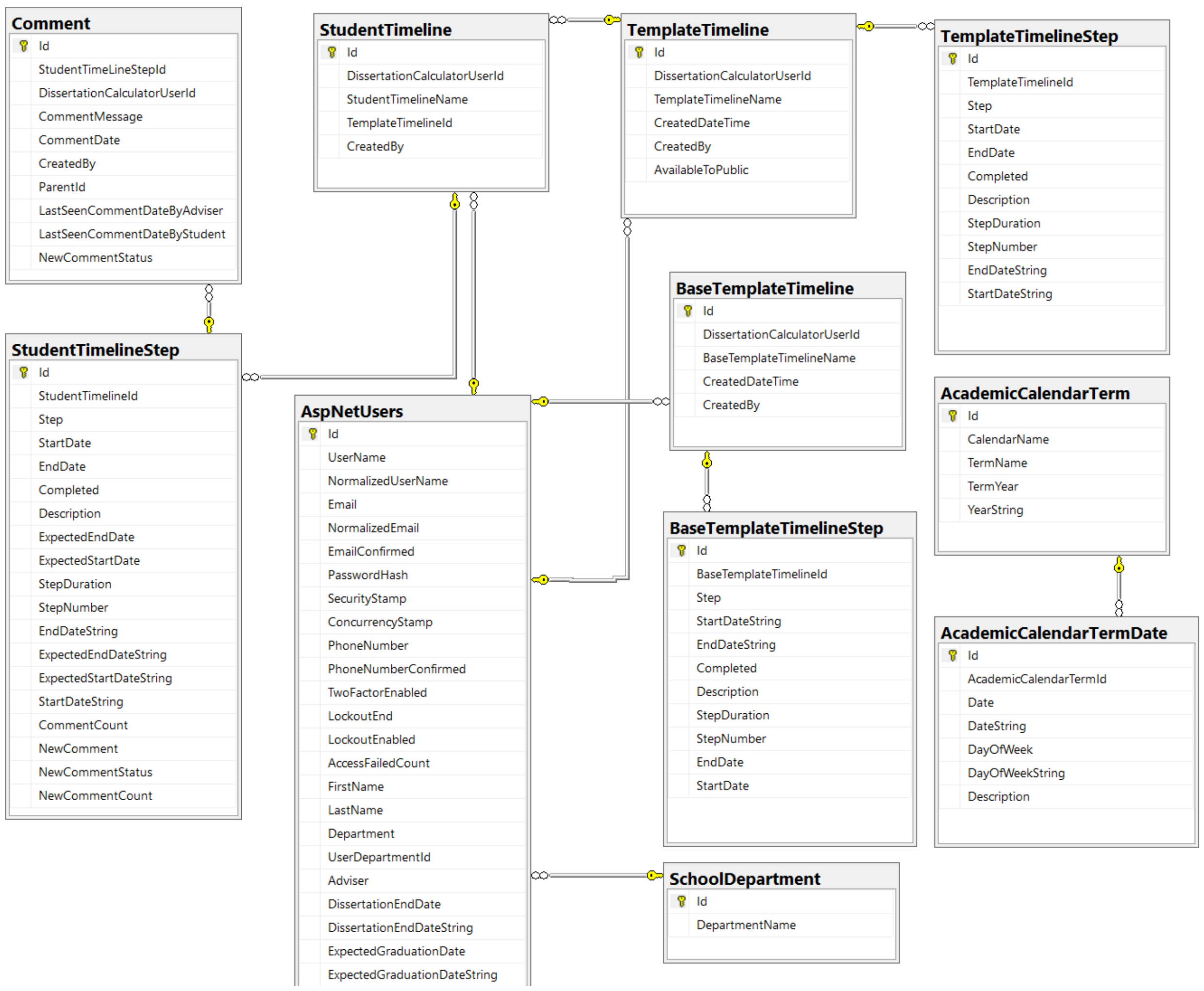


Figure 3.7. Relational Database Diagram Part 1

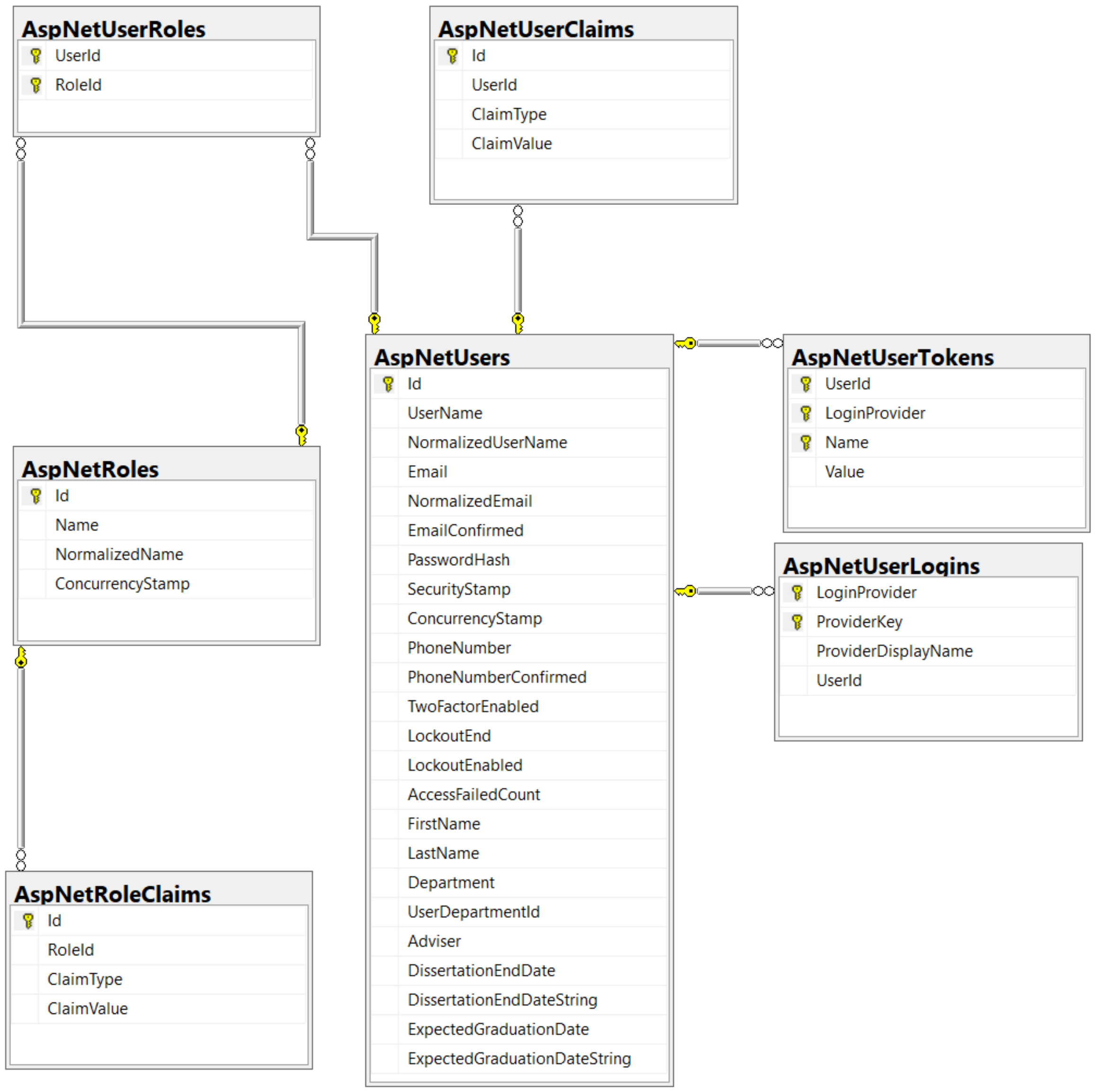


Figure 3.8. Relational Database Diagram Part 2

3.6. Use Case Diagram

Figure 10 below shows the use case diagram of Dissertation Calculator tool. We have three actors which are student, adviser, and administrator. Almost all uses cases include the use case Login because any users will need to be logged into the system before using any functionality.

Student actor can select an adviser, set expected graduation date, create timeline, view their timeline, add, remove, and edit a step, comment on a step, and reset their password. Advisers can view student list, view student timeline, comment on a step, create template timeline, and reset their password. Administrators can create a new account, edit accounts, edit departments, edit academic calendar dates and deadlines, create base template timeline, and reset their password.

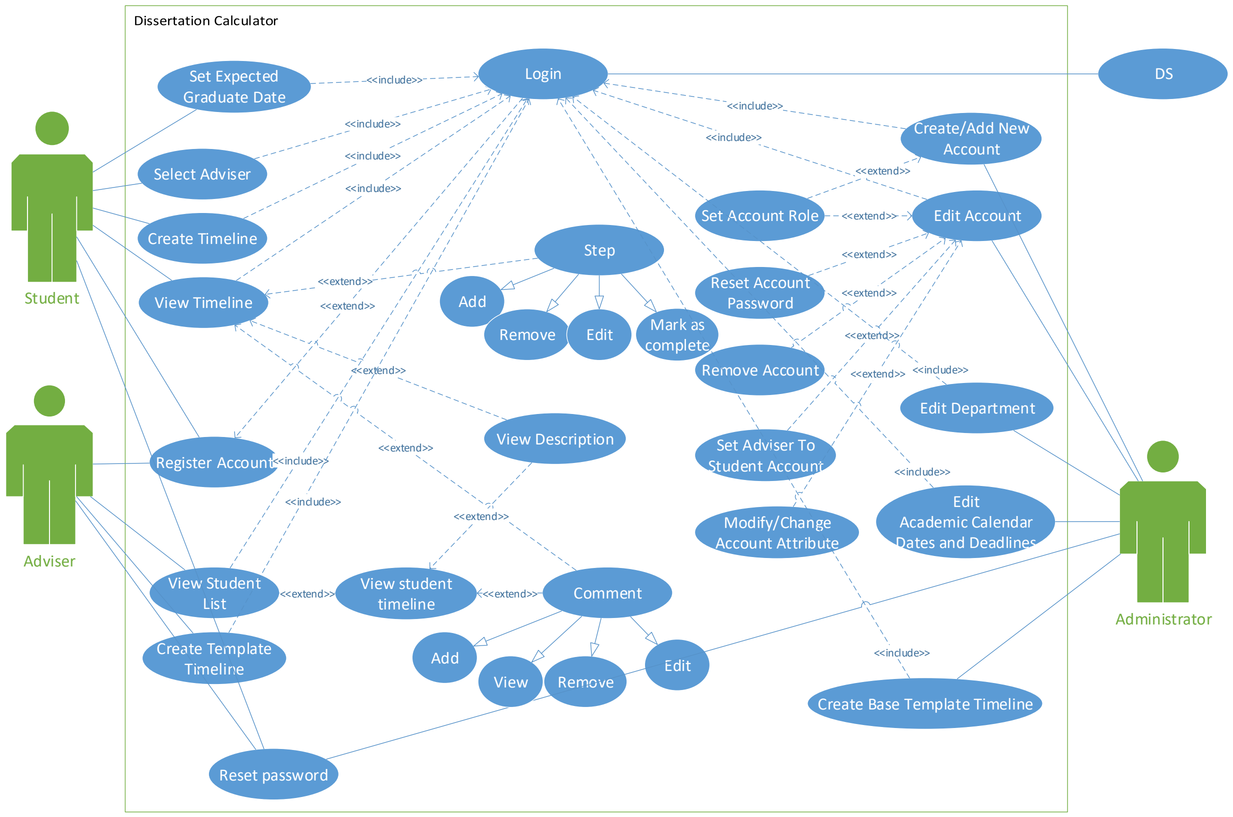


Figure 3.9. Use Case Diagram

4. DEVELOPMENT

In this section, the details about the development process of Dissertation Calculator tool. Initially, there was a previous tool developed as a class project using Maven, Tapestry, and Cayenne development tools by students of CSCI 413. User stories were collected as a requirement document and the project was based off on those user stories. There were lots of limitations to this early system that is why there was a need for an upgraded system that has more features that is on par with the updated user stories.

4.1. Dissertation Calculator

The tool was developed using Microsoft's ASP.NET Core framework on Windows with Visual Studio 2019. This was developed to introduce more features and change the look and feel of the previous tool. This is not a true production environment project yet, but it is more feature full version of the tool. It is also ready to be tested for production environment deployment. One of the major reasons why ASP.NET Core was selected was that it was based on C# which made the development more familiar and the other added benefits mentioned earlier also made good contribution as it was a framework that is easily suited to develop web applications.

4.1.1. Environment

4.1.1.1. Development Tools

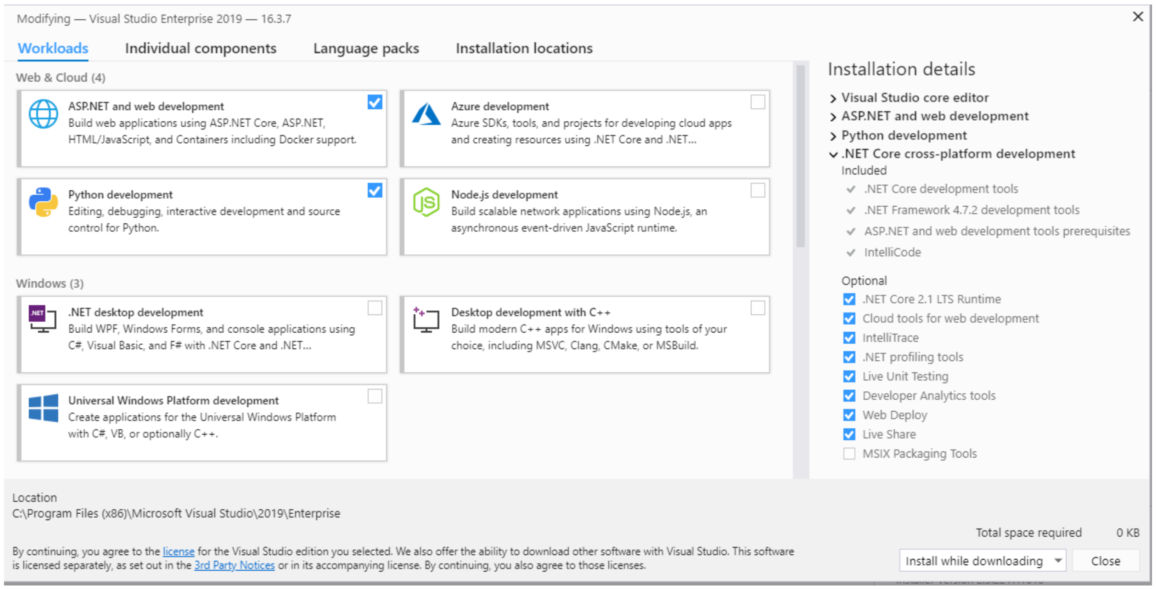
- Development Environment – Windows 10 Pro Version 1903 Build 18362.418
- Development Framework – ASP.NET Core
- Development Languages – C#, HTML, JavaScript, CSS
- Versioning System – Github
- IDE: Microsoft Visual Studio Enterprise 2019
- Local Database Provider: SQL Express

- Local Web Server: IIS Express
- Test Browser: Mozilla Firefox and Google Chrome

The development tools were selected as they are the most comprehensive and works out of the box for ASP.NET Core development. Almost everything gets setup automatically and from the first time you compile your code to running it, it takes one click to have everything up and running.

Windows 10 operating system is a no brainer choice for C# development. Linux can be used but Windows 10 provides the best experience when developing C# applications using Visual Studio. It can run all the required tools with easy and requires little to no configuration.

Visual Studio on Windows is just one of the best IDE to use for development. It is simple to install and configure to make it up and running for development. It also requires little to no configuration. When installing it, the specific development tools just have to be selected and it configures the features and requirements automatically. If any change or extra feature is required, the Visual Studio Installer can be used to add or remove features.


Modifying — Visual Studio Enterprise 2019 — 16.3.7

Web & Cloud (4)

✓ ASP.NET and web development
Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker support.

☐ Azure development
Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET Core and .NET...

✓ Python development
Editing, debugging, interactive development and source control for Python.

☐ Node.js development
Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Windows (3)

☐ .NET desktop development
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET Core and .NET...

☐ Desktop development with C++
Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.

☐ Universal Windows Platform development
Create applications for the Universal Windows Platform with C#, VB, or optionally C++.

Installation details

- Visual Studio core editor
- ASP.NET and web development
- Python development
- .NET Core cross-platform development

Included
- ✓ .NET Core development tools
- ✓ .NET Framework 4.7.2 development tools
- ✓ ASP.NET and web development tools prerequisites
- ✓ IntelliCode

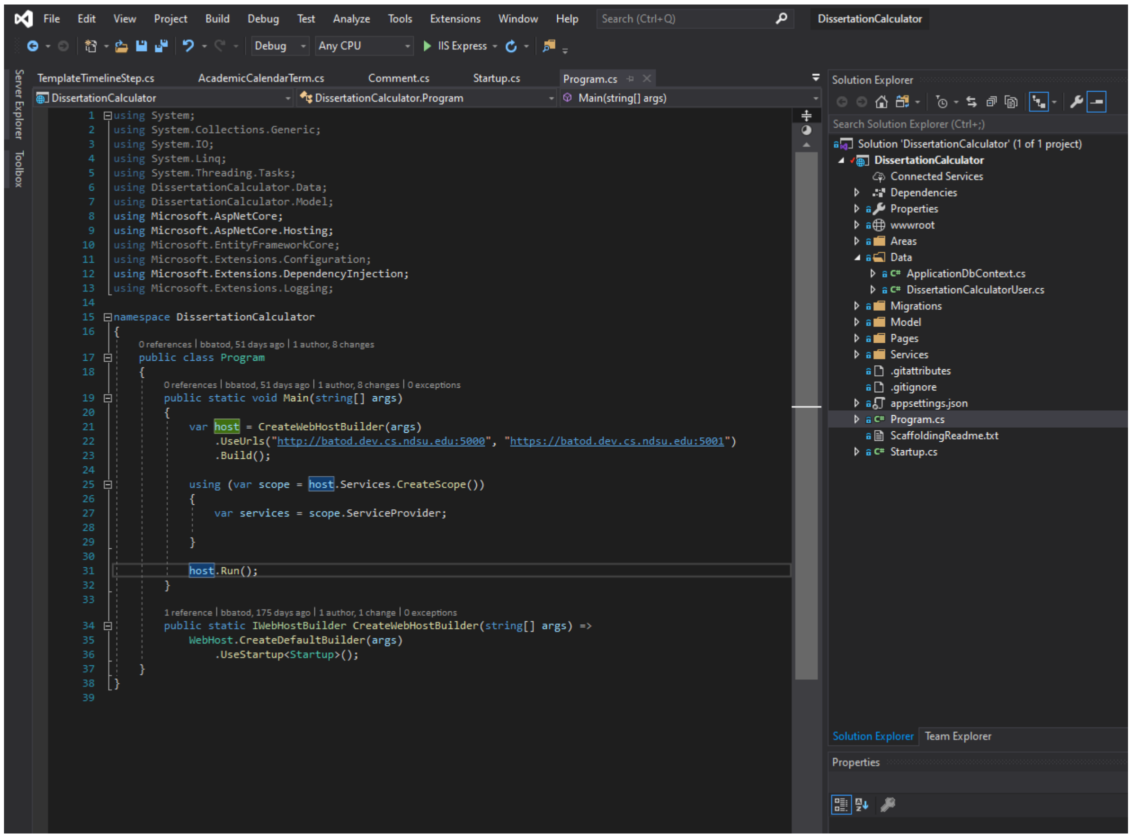
Optional
- ✓ .NET Core 2.1 LTS Runtime
- ✓ Cloud tools for web development
- ✓ IntelliTrace
- ✓ .NET profiling tools
- ✓ Live Unit Testing
- ✓ Developer Analytics tools
- ✓ Web Deploy
- ✓ Live Share
- ☐ MSIX Packaging Tools

Location
C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

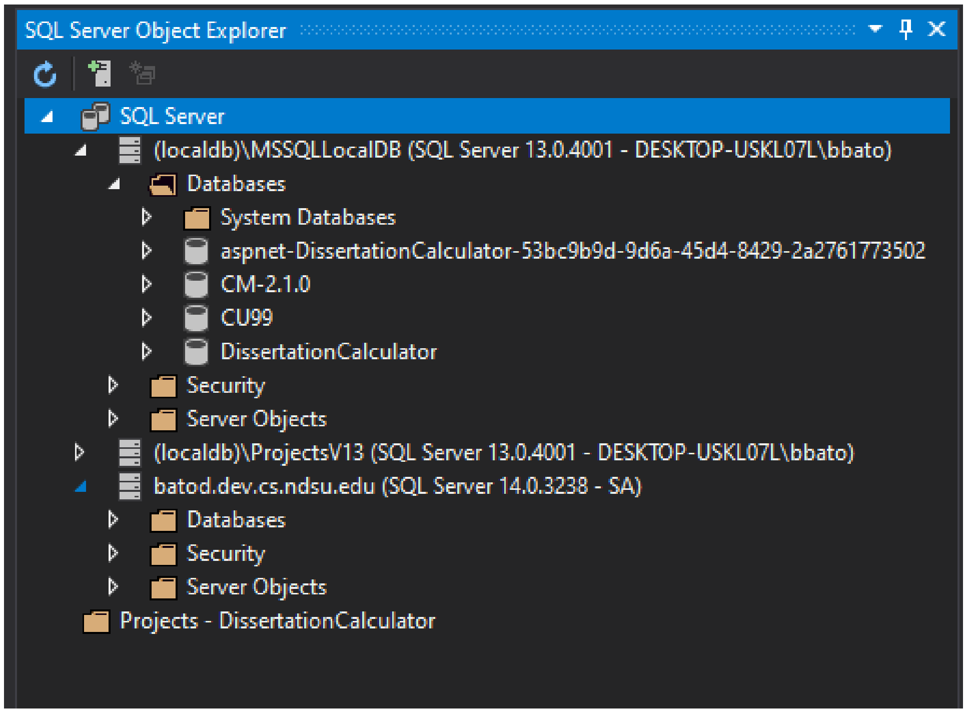
Total space required 0 KB

Figure 4.1. Visual Studio Installer



```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using DissertationCalculator.Data;
using DissertationCalculator.Model;
using Microsoft.AspNetCore;
using Microsoft.AspNetCore.Hosting;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;

namespace DissertationCalculator
{
    0 references | bbatod, 51 days ago | 1 author, 8 changes
    public class Program
    {
        0 references | bbatod, 51 days ago | 1 author, 8 changes | 0 exceptions
        public static void Main(string[] args)
        {
            var host = CreateWebHostBuilder(args)
                .UseUrls("http://batod.dev.cs.ndsu.edu:5000", "https://batod.dev.cs.ndsu.edu:5001")
                .Build();

            using (var scope = host.Services.CreateScope())
            {
                var services = scope.ServiceProvider;

            }

            host.Run();
        }

        1 reference | bbatod, 175 days ago | 1 author, 1 change | 0 exceptions
        public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
            WebHost.CreateDefaultBuilder(args)
                .UseStartup<Startup>();
    }
}
```

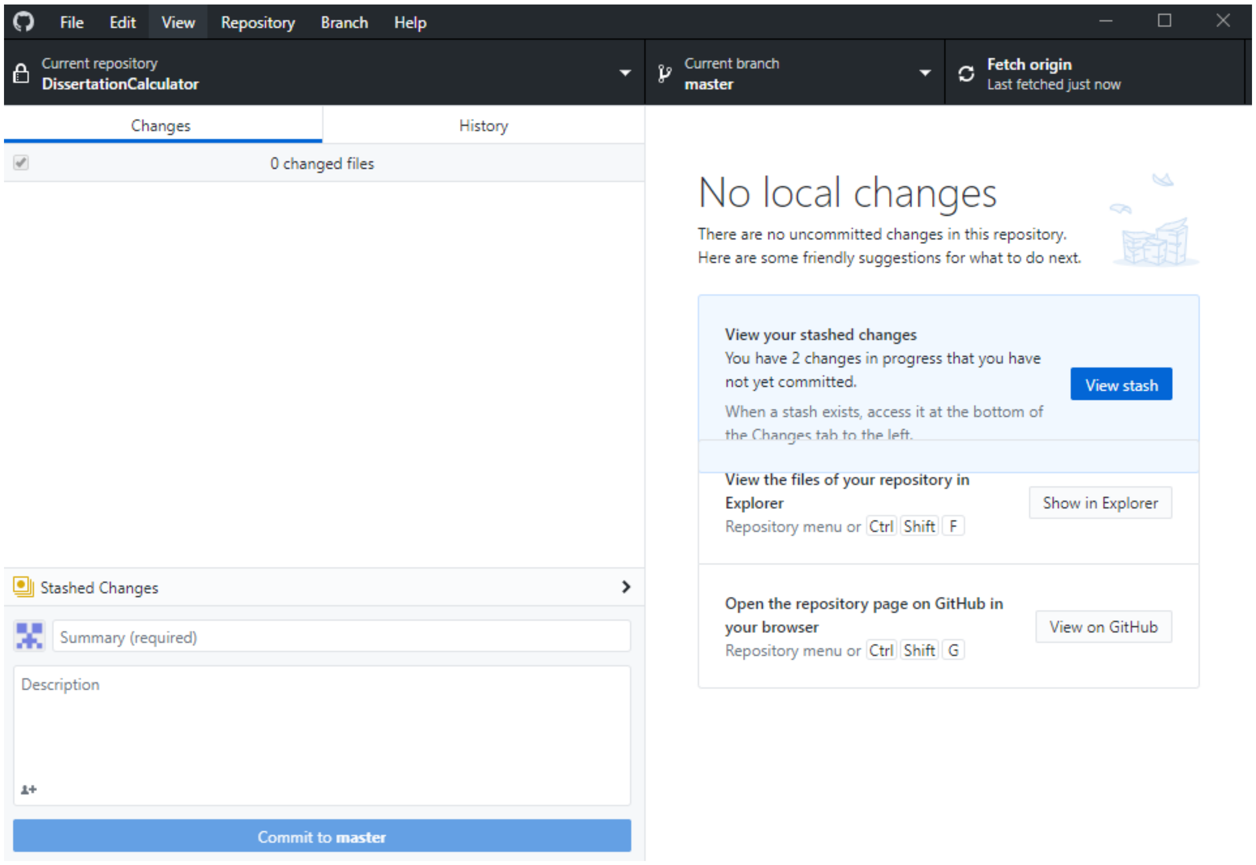
Figure 4.2. Visual Studio 2019



```
SQL Server
  (localdb)\MSSQLLocalDB (SQL Server 13.0.4001 - DESKTOP-USKL07L\bbato)
    Databases
      System Databases
      aspnet-DissertationCalculator-53bc9b9d-9d6a-45d4-8429-2a2761773502
      CM-2.1.0
      CU99
      DissertationCalculator
    Security
    Server Objects
  (localdb)\ProjectsV13 (SQL Server 13.0.4001 - DESKTOP-USKL07L\bbato)
  batod.dev.cs.ndsu.edu (SQL Server 14.0.3238 - SA)
    Databases
    Security
    Server Objects
  Projects - DissertationCalculator
```

Figure 4.3. SQL Server Object Explorer in Visual Studio

Github was used as the versioning tool for the project. Github Desktop application made things very simple as it allowed GUI controls to create, merge, and delete branches. It also made it easier to push the branch to remote so that multiple devices could be used to develop the project.



No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

View your stashed changes
You have 2 changes in progress that you have not yet committed.
When a stash exists, access it at the bottom of the Changes tab to the left.

View the files of your repository in Explorer
Repository menu or Ctrl Shift F

Open the repository page on GitHub in your browser
Repository menu or Ctrl Shift G

Figure 4.4. Github Desktop

SQL Express and IIS Express gets installed and configured when Web Development tools are installed when installing Visual Studio. They are the default tools for database provider and web server when developing with Visual Studio. It is very easy to move over to the production environment tools like SQL Server and IIS as the Express versions cover most of the functionality in a smaller scale.

4.1.1.2. Test Environment

Local:

- Test Environment: Windows 10
- Database Provider: SQL Express
- Web Server: IIS Express
- Client: Mozilla Firefox

Hosted/ Deployed Remote Server:

- Test Environment: Ubuntu 18.04
- Database Provider: SQL Server 2017 Developer
- Web Server: Kestrel and Nginx
- Client: Mozilla Firefox and Google Chrome

The local test environment is the same environment as the development environment since it can run the application locally using SQL Express for data provider and IIS Express for the web server. As for testing on an actual server, a server running Ubuntu 18.04 was provided by the Computer Science department for the project to test deployment and remote hosting. The server is a VM hosting Ubuntu 18.04 with 4 CPU cores, 8GB of RAM, and 54GB storage. Microsoft recently started to support SQL Server on Linux which made the transition from SQL Express to SQL Server much easier. For this project, we used SQL Server 2017 Developer version as provided for free by Microsoft.

ASP.NET Core applications are cross-platform which means it can be run on Windows, Linux, macOS, or Docker environments. On Linux, the application uses a self-hosted Kestrel web server meaning it is integrated into the application so it only requires the executable to be run on the server given that the requirements such as .NET Core Framework is installed on the server. Nginx is used as a HTTP and reverse proxy server. The server accepts requests on "http://batod.dev.cs.ndsu.edu" address so Nginx is listening on this address on the server and re-routes the traffic to the Kestrel web server which the application is running on. This can be used to configure security, load balancing, and SSL certificates.

As for client to test the application, Mozilla Firefox and Google Chrome were used but almost any browser will work with the application as it is using industry standard code thanks to ASP.NET Core framework.

4.2. Configuring Host Server

The host server was setup using Ubuntu 18.04 running on a virtual machine provided by the Computer Science department. The general specifications of the server are 4 CPU cores, 8GB of RAM, and 54GB storage. These specifications are more than enough for testing the application and possibly as a production environment but that needs to be tested with live users to determine. The application run on any environment that ASP.NET Core is supported so hosting it can be easily configured on physical server or cloud server on Microsoft Azure or Amazon Web Services

On the machine, SQL Server 2017 installed using the installation guide provided by Microsoft. They recommend using Ubuntu 16.04 but with the latest update to SQL Server 2017, 18.04 works great on Ubuntu 18.04. The installation requires command line commands to be run on the server, but it does not take long time to have it installed, and up and running.

As for hosting and deploying the ASP.NET Core application on a Linux server, we will need to make sure it meets all the prerequisites such as running the supported version of Ubuntu, installing .NET Core runtime on the server, configuring Nginx to reroute traffic, and make sure the server is monitoring the running application to keep it running if it crashes or the server is restarted.

4.3. Running the Application

In order to run the application, we will need to publish the application. It is a term used for Visual Studio projects for compiling and getting the actual executable to run. There are two ways to publish an ASP.NET Core application.

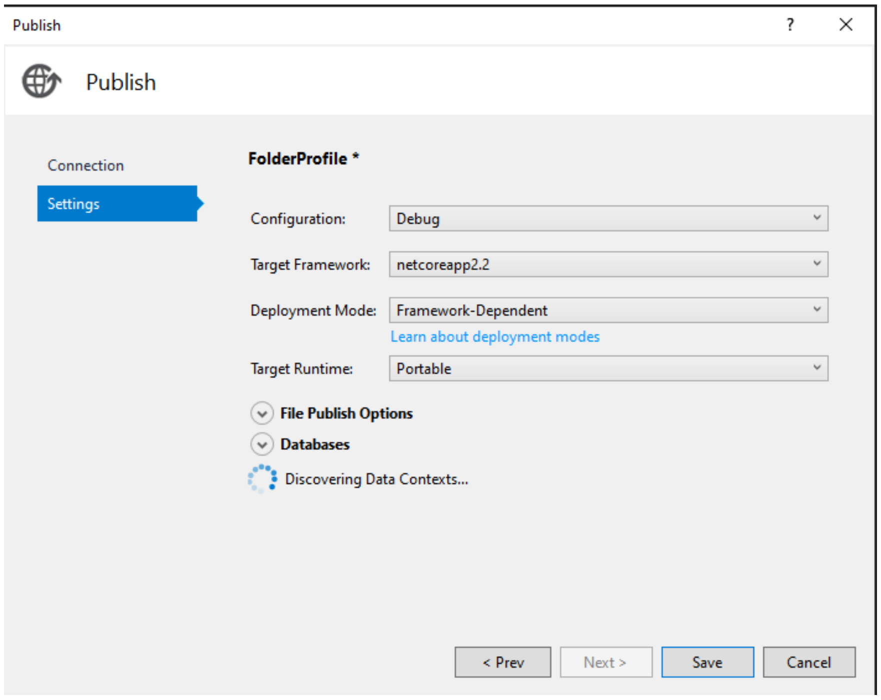
First method is a framework-dependent deployment. This method has few advantages and disadvantages. The advantages are that we don’t have to define a target operating system that the .NET Core app will run on in advance as long as a .NET Core runtime is installed and configured on the server, the size of the deployment is relatively small, it also allows to use the latest version of the runtime that is running on the server which means the app can run on the latest version without any trouble, and multiple different apps can use the same runtime on the same server to run. The disadvantages are that the application you are trying to run on can only run on a version that your app targets or later version that is already installed on the host system, and .NET Runtime and libraries may change in the future without your knowledge and your app my stop working or change its behavior.

Second method is self-contained deployments which allows us to deploy the app with any other required third-party dependencies including the version of .NET Core runtime. This also has its advantages and disadvantages.

The advantages are that you will have the control of the version of .NET Core that is deployed with the .NET core app and can only be serviced by you, and this way, you can make sure the target system can run the application since you are providing the version of .NET Core to the system.

The disadvantages are that you must select the target platform in advance before deploying the packages that come with the application. Size of the deployment can get big due to bundling all the other third party dependencies and .NET Core runtime version that you specified. The other disadvantage is that running multiple self-contained .NET Core app on the server can consume lots of resources such as disk space as .NET Core files will have to be duplicated on multiple applications.

For this project and testing, framework-dependent deployment was selected as the advantages such as not needing to define a target platform, having small deployment size, running latest version of the runtime, are all suitable for this project. In Visual Studio, when you publish your app, it gives us the options on which type of deployment we want to do and where we want to deploy it. For my testing purposes, I selected to deploy the executable files in a folder on the local computer first and then copy the files to the server using FTP protocol with WinSCP tool.

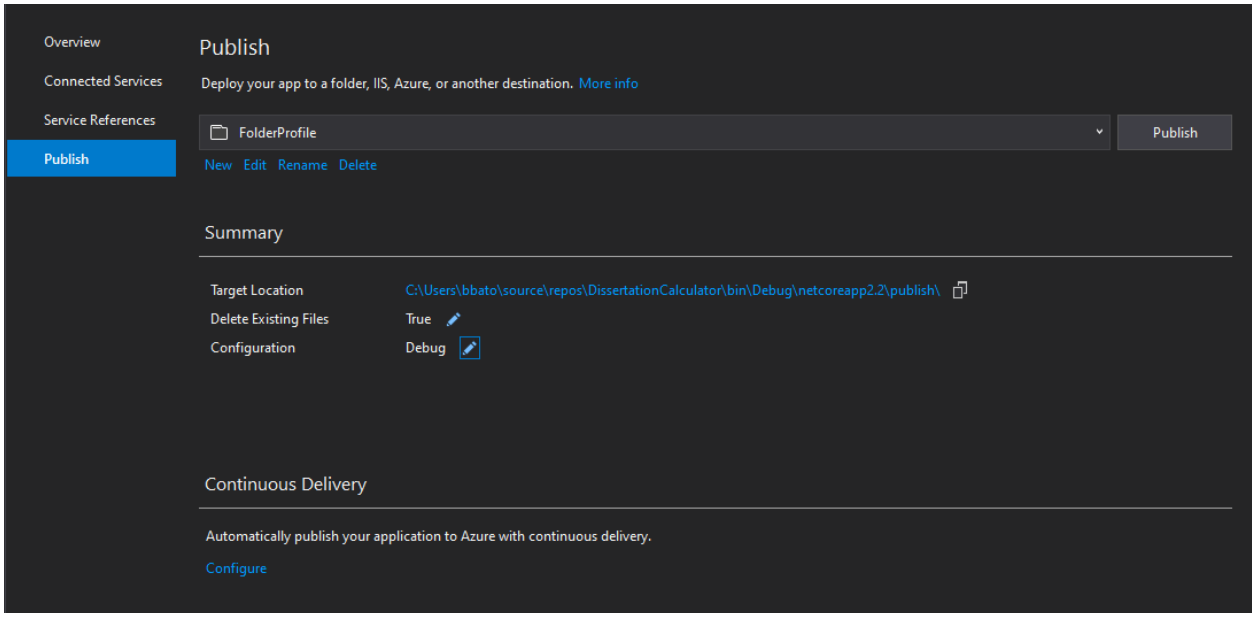


FolderProfile *

Configuration: Debug
Target Framework: netcoreapp2.2
Deployment Mode: Framework-Dependent
Learn about deployment modes
Target Runtime: Portable

File Publish Options
Databases
Discovering Data Contexts...

Figure 4.5. Visual Studio Publish Application Menu



Publish

Deploy your app to a folder, IIS, Azure, or another destination. More info

FolderProfile

Publish

New Edit Rename Delete

Summary

Target Location C:\Users\bbato\source\repos\DissertationCalculator\bin\Debug\netcoreapp2.2\publish\

Delete Existing Files True

Configuration Debug

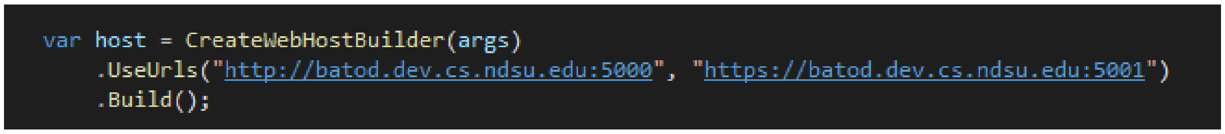
Continuous Delivery

Automatically publish your application to Azure with continuous delivery.

Configure

Figure 4.6. Visual Studio Publish Application Options

Once the files are copied to the server and .NET Core runtime is configured to run on the server, we needed to make sure that the application web server is listening on specific URLs so that outside requests can be routed. The application was coded to listen on two URLS, “http://batod.dev.cs.ndsu.edu:5000” for HTTP requests and “https://batod.dev.cs.ndsu.edu:5001” for HTTPS requests.



```
var host = CreateWebHostBuilder(args)
    .UseUrls("http://batod.dev.cs.ndsu.edu:5000", "https://batod.dev.cs.ndsu.edu:5001")
    .Build();
```

Figure 4.7. Application URL Configuration

The Nginx server then reroutes “http://batod.dev.cs.ndsu.edu:80” traffic to “https://batod.dev.cs.ndsu.edu:5001” automatically.

Nginx configuration:

```
server {

   listen        80;

   server_name   batod.dev.cs.ndsu.edu *.batod.dev.cs.ndsu.edu;

   location / {

       proxy_pass         http://localhost:5000;

       proxy_http_version 1.1;

       proxy_set_header   Upgrade $http_upgrade;

       proxy_set_header   Connection keep-alive;

       proxy_set_header   Host $host;

       proxy_cache_bypass $http_upgrade;

       proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;

       proxy_set_header   X-Forwarded-Proto $scheme;

   }

}
```

After this, we make sure the server is monitoring the app and making sure it is running. For this, we create a service file and enable to run at system startup.

For this, a service file is created with the command below:

```
sudo nano /etc/systemd/system/kestrel-dissertationcalculator.service
```

Then the following configuration is added and saved:

```
[Unit]
Description=Dissertation Calculator App running on Ubuntu

[Service]
WorkingDirectory=/home/batotgon/www
ExecStart=/usr/bin/dotnet /home/batotgon/www/DissertationCalculator.dll
Restart=always
# Restart service after 10 seconds if the dotnet service crashes:
RestartSec=10
KillSignal=SIGINT
SyslogIdentifier=dotnet-example
User=batotgon
Environment=ASPNETCORE_ENVIRONMENT=Development
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false

[Install]
WantedBy=multi-user.target
```

Once the file is created and saved, we need to enable the service to make sure it starts when the system starts.

```
sudo systemctl enable kestrel-dissertationcalculator.service
```

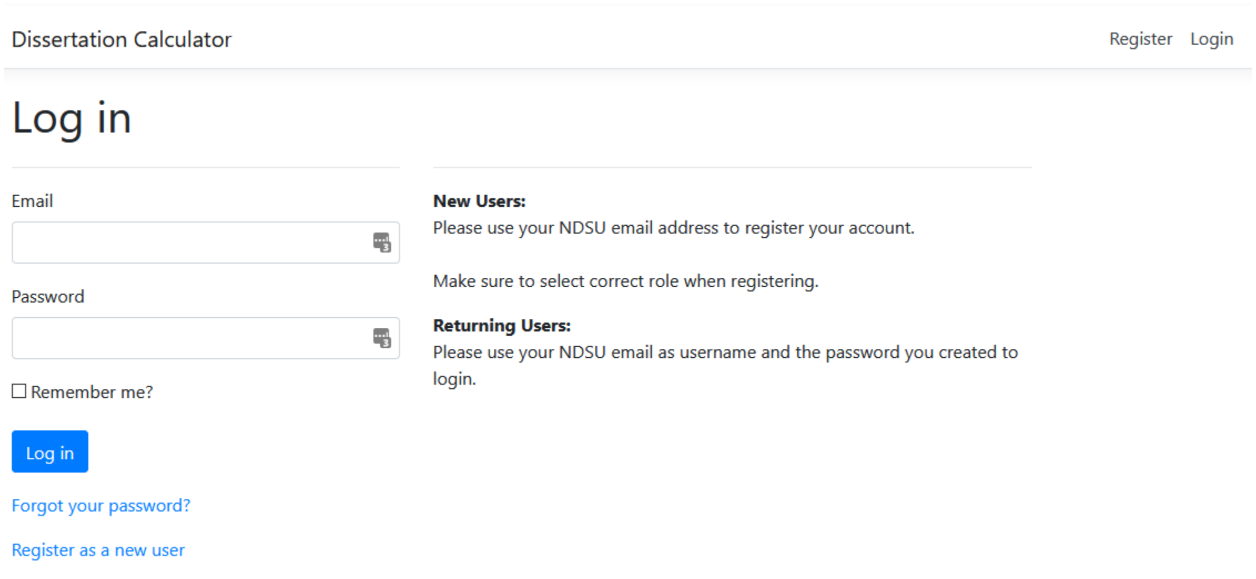
The service is started, and the status is checked with the following commands.

```
sudo systemctl start kestrel-dissertationcalculator.service
sudo systemctl status kestrel-dissertationcalculator.service
```

Output:

```
● kestrel-dissertationcalculator.service - Dissertation Calculator App running on Ubuntu
  Loaded: loaded (/etc/systemd/system/kestrel-dissertationcalculator.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2019-10-29 11:38:39 CDT; 1 day 2h ago
Main PID: 12454 (dotnet)
   Tasks: 21 (limit: 4915)
  CGroup: /system.slice/kestrel-dissertationcalculator.service
          └─12454 /usr/bin/dotnet /home/batotgon/www/DissertationCalculator.dll
```

After all the configuration and running the services, the app will be accessible.



New Users:
Please use your NDSU email address to register your account.

Make sure to select correct role when registering.

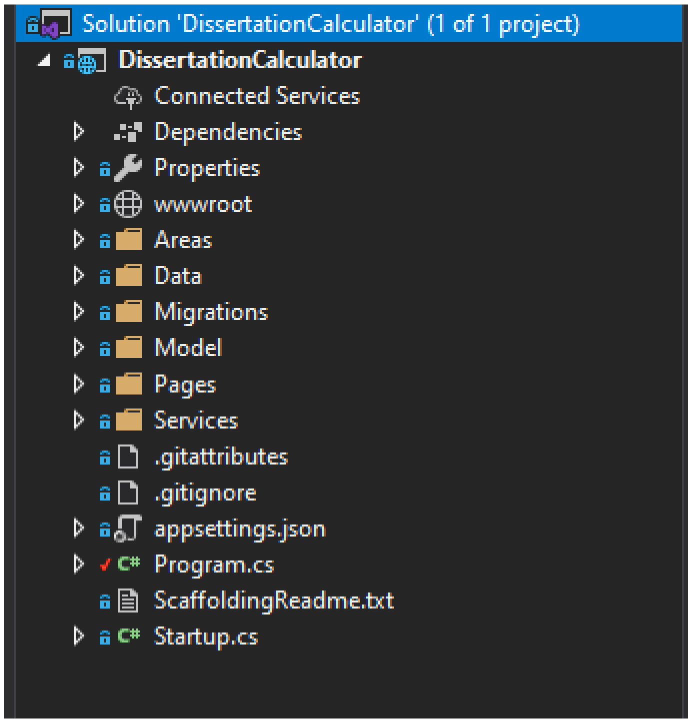
Returning Users:
Please use your NDSU email as username and the password you created to login.

Figure 4.8. Dissertation Calculator Tool Login

4.4. Development Progress

The project was developed in three main phases. The first phase was to get a ASP.NET Core application up and running with basic authentication. Second phase was to get different roles in the system and give them different views. There are three main views such as student, adviser, and administrator views. Each stakeholder has different functionalities, so each view was developed separately. This also gives the advantage to maintain each view separately from each other without affecting the other. The third phase was to code on each different view to give each specific user their functionalities based on the user stories that was developed before the start of the project.

4.4.1. Project File Organization

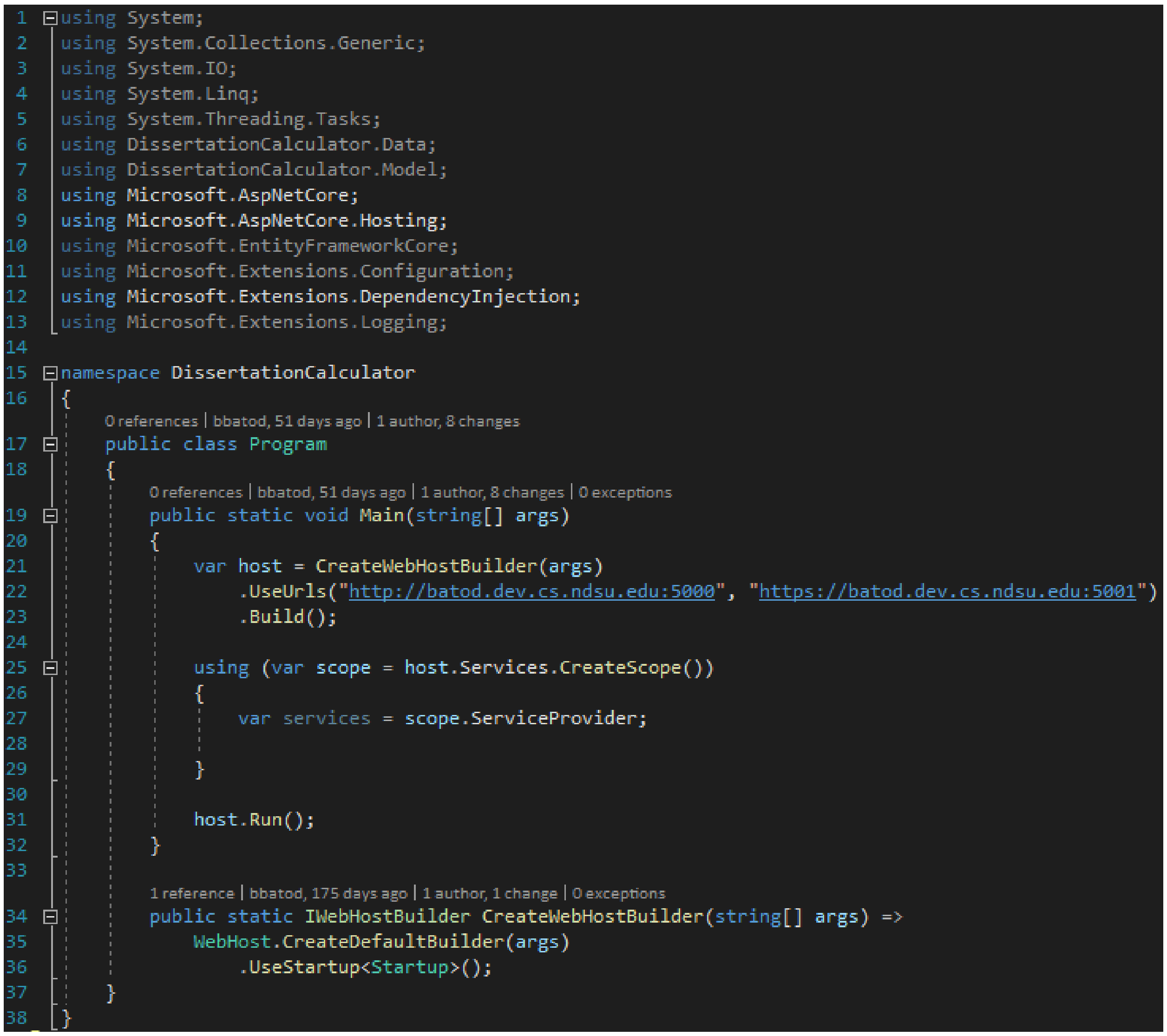


```
Solution 'DissertationCalculator' (1 of 1 project)
  DissertationCalculator
      Connected Services
      Dependencies
      Properties
      wwwroot
      Areas
      Data
      Migrations
      Model
      Pages
      Services
      .gitattributes
      .gitignore
      appsettings.json
      Program.cs
      ScaffoldingReadme.txt
      Startup.cs
```

Figure 4.9. File Organization in Visual Studio

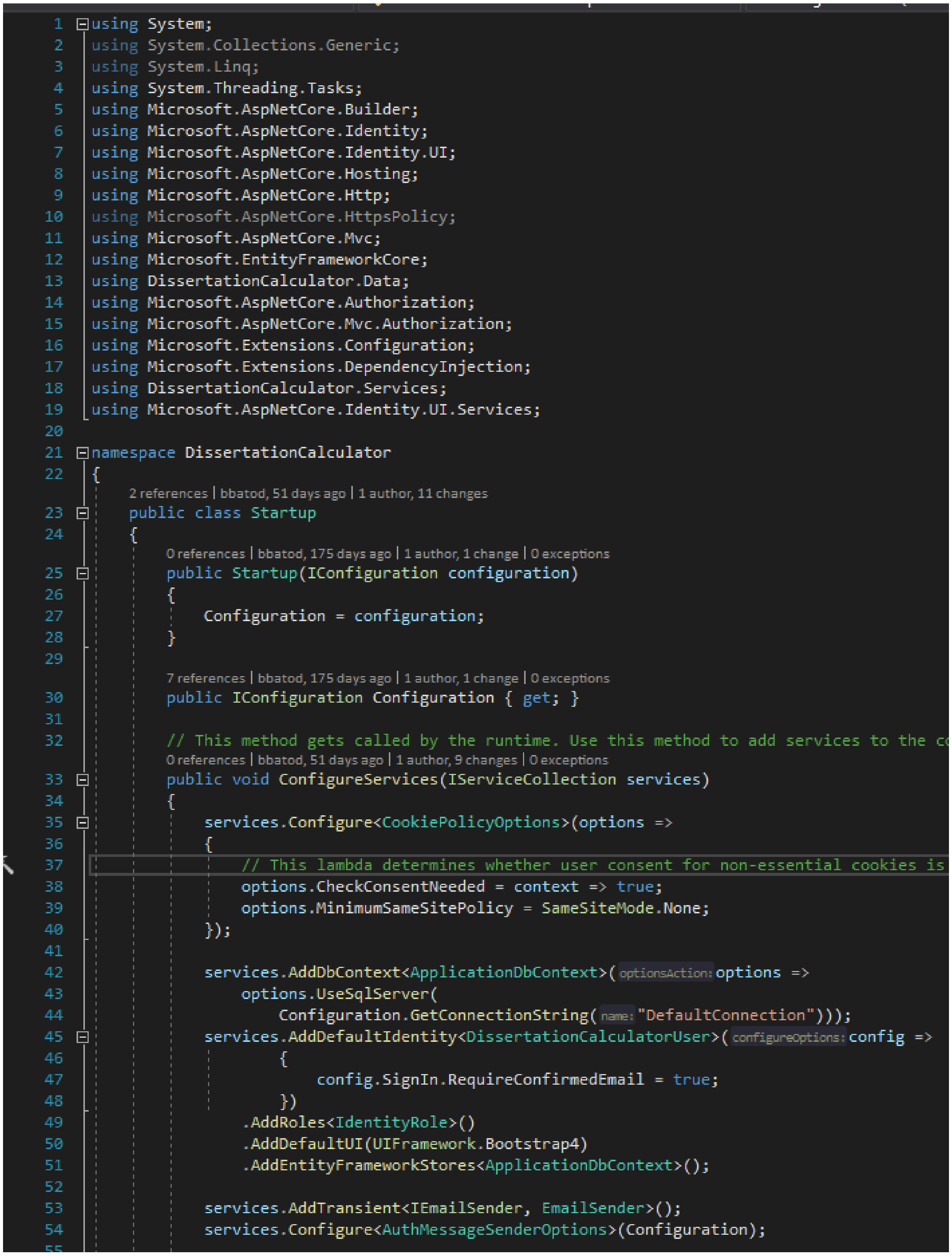
Areas folder contain the code for identity, authentication, and profile related code. This is created by ASP.NET Core automatically when authentication is selected to be included in the project, but we must scaffold it to have all the code generated so that they can be modified to our needs. ASP.NET Core Identity is a library provided so scaffolding it will provide the source code so that we can customize it for our needs in the project. This way, we can customize the login page, log out page, user profile page etc. Data folder contains the database context and customized User class to add more custom user properties. Migrations folder is used by ASP.NET Core Entity Framework Core to migrate .NET Core objects to relational database objects. Model folder contains the .NET Core objects for the database model. Pages folder contains all the pages used by the users of the system. Admin folder contains all the pages used by Administrator user. Adviser folder contains all the pages used by Adviser. Student folder contains all the pages used by Student users. Shared folder contains pages that are used throughout the application such as "_layout.cshtml" page where it contains the code for the style, script, layout of pages used by all the pages in the system. This way, we don't have to keep repeating the same code on all pages for style, script, and layout. Services folder contains few service codes that are utilized by the email service which is using a third-party package called SendGrid.

The other two important files in the project are "Program.cs" and "Startup.cs". "Program.cs" is the main program of the project where we create the host for the web application. This process created the web server that is built-in within ASP.NET Core projects, "Startup.cs" is a class where services required by the app are configured.



```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using DissertationCalculator.Data;
using DissertationCalculator.Model;
using Microsoft.AspNetCore;
using Microsoft.AspNetCore.Hosting;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;

namespace DissertationCalculator
{
    0 references | bbatod, 51 days ago | 1 author, 8 changes
    public class Program
    {
        0 references | bbatod, 51 days ago | 1 author, 8 changes | 0 exceptions
        public static void Main(string[] args)
        {
            var host = CreateWebHostBuilder(args)
                .UseUrls("http://batod.dev.cs.ndsu.edu:5000", "https://batod.dev.cs.ndsu.edu:5001")
                .Build();

            using (var scope = host.Services.CreateScope())
            {
                var services = scope.ServiceProvider;

            }

            host.Run();
        }

        1 reference | bbatod, 175 days ago | 1 author, 1 change | 0 exceptions
        public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
            WebHost.CreateDefaultBuilder(args)
                .UseStartup<Startup>();
    }
}
```

Figure 4.10. Program.cs



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.UI;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using DissertationCalculator.Data;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc.Authorization;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using DissertationCalculator.Services;
using Microsoft.AspNetCore.Identity.UI.Services;

namespace DissertationCalculator
{
    2 references | bbatod, 51 days ago | 1 author, 11 changes
    public class Startup
    {
        0 references | bbatod, 175 days ago | 1 author, 1 change | 0 exceptions
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        7 references | bbatod, 175 days ago | 1 author, 1 change | 0 exceptions
        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the c
        0 references | bbatod, 51 days ago | 1 author, 9 changes | 0 exceptions
        public void ConfigureServices(IServiceCollection services)
        {
            services.Configure<CookiePolicyOptions>(options =>
            {
                // This lambda determines whether user consent for non-essential cookies is
                options.CheckConsentNeeded = context => true;
                options.MinimumSameSitePolicy = SameSiteMode.None;
            });

            services.AddDbContext<ApplicationDbContext>(optionsAction: options =>
                options.UseSqlServer(
                    Configuration.GetConnectionString(name: "DefaultConnection")));
            services.AddDefaultIdentity<DissertationCalculatorUser>(configureOptions: config =>
                    {
                        config.SignIn.RequireConfirmedEmail = true;
                    })
                .AddRoles<IdentityRole>()
                .AddDefaultUI(UIFramework.Bootstrap4)
                .AddEntityFrameworkStores<ApplicationDbContext>();

            services.AddTransient<IEmailSender, EmailSender>();
            services.Configure<AuthMessageSenderOptions>(Configuration);
```

Figure 4.11. Startup.cs

4.5. Functionalities

In this section, the functionalities of the application will be discussed, and details of each functionality will be highlighted to explain how things are developed to work.

4.5.1. Shared Functionalities

There are number of functionalities that each user such as student, adviser, and admin can all perform in this application. We will cover them below.

4.5.1.1. Register

If you don't have an account in the system, you will not be able to use it so first thing each user will have to do is to register an account. When you are not sign in, there will be a Register button at the top right corner of the application as seen on the figure below. Once you click on it, it will ask for the user to input specific information such as first name, last name, department, user role like student or adviser, email, and password.

User must input their own NDSU email address as email confirmation is required to finish setting up the account. This way, we can prevent other users from using other people's email addresses. Once you enter all the required information and click on "Register", system creates the account but "EmailConfirmed" attribute is set to false in the database and confirmation email is sent to the user's email address. If "EmailConfirmed" is set to false, the system will not allow the user to login even though the account exists. When the user clicks on the link in the email as shown in figure below, system confirms the email address as confirmed as shown in figure below and allows the user to login to the system. Password are hashed in the database with PBKDF2 with HMAC-SHA256, 128-bit salt, 256-bit subkey, 10000 iterations algorithm as it is the default hashing method provided by ASP.NET Core Identity library. Because the system uses ASP.NET Core Identity for authentication, authorization, and security, it is very easy to customize its options such as lockout mechanism if users try wrong password too many times, password requirements, sign in options such as requiring confirmed email address or even confirmed phone number, allowed characters in usernames, in the context of the our application this applies to email address field, and cookie settings such as cookie name, expiration time, and password hashing options like number of iterations.

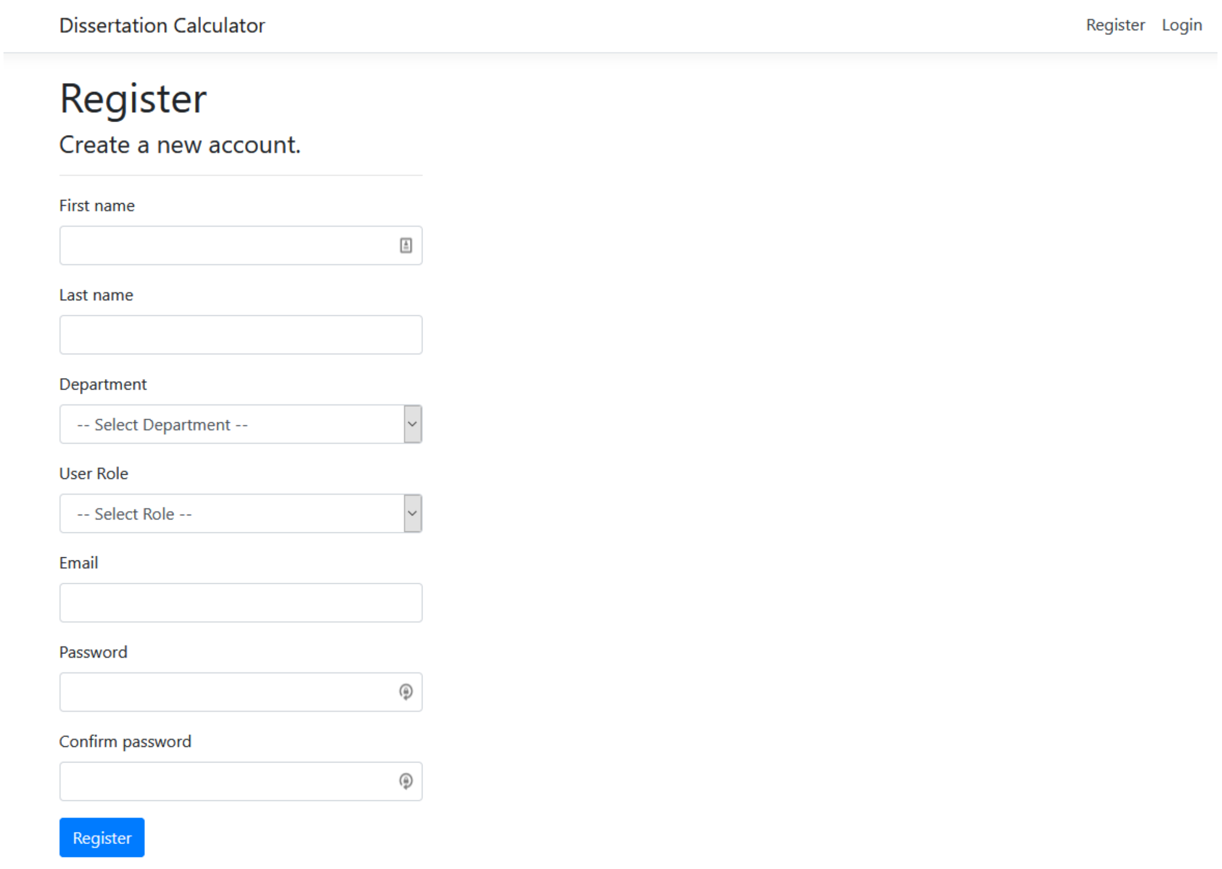
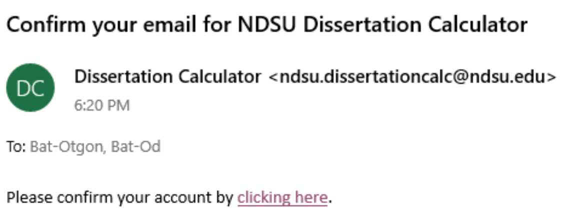


Figure 4.12. Register Page



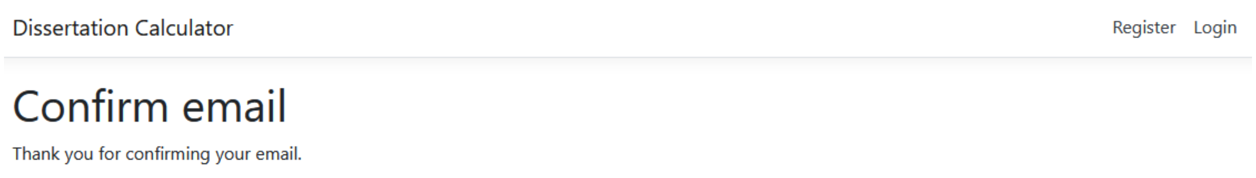
Confirm your email for NDSU Dissertation Calculator

DC Dissertation Calculator <ndsu.dissertationcalc@ndsu.edu>
6:20 PM

To: Bat-Otgon, Bat-Od

Please confirm your account by clicking here.

Figure 4.13. Email Confirmation


Dissertation Calculator Register Login

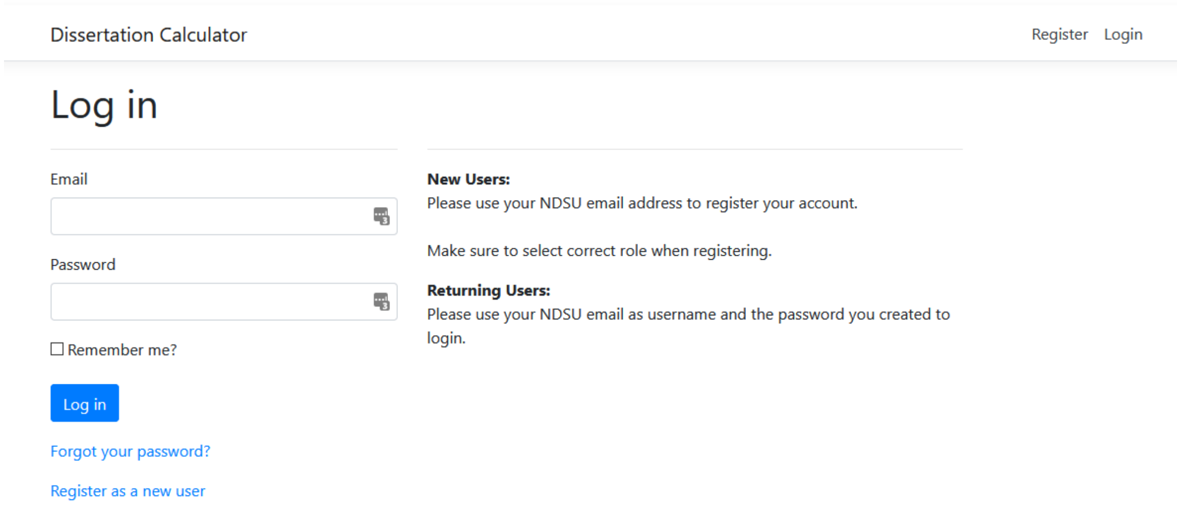
Confirm email

Thank you for confirming your email.

Figure 4.14. Page Confirming Email Was Confirmed

4.5.1.2. Login

Login feature is functionality that everyone can perform if they have an account in the system. If the username and password does not match and not found in the system. The application generates an error message and requests the user to try again. As it stands right now, there is no auto locking feature, but it can be added down the line to the system with minimal coding. Username entered in Email field must have @ndsu.edu. If it does not, the system will be thrown an error explaining it must have it. "Remember me?" checkbox will allow the user to have the system remember their credentials. What this means if user closes their browser and comes back to the application in the same browser, it will let the user access the application without prompting for login. If you log out, manually, the cookies are expired and the "Remember me?" option will need to be checked again for this to work.


Dissertation Calculator Register Login

Log in

Email

Password

☐ Remember me?

Log in

Forgot your password?

Register as a new user

New Users:
Please use your NDSU email address to register your account.

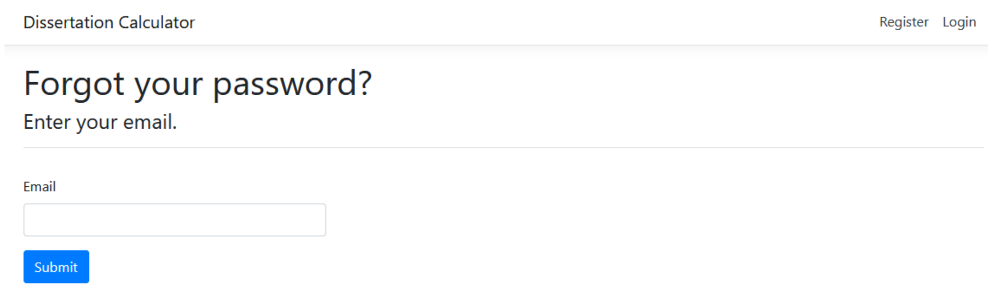
Make sure to select correct role when registering.

Returning Users:
Please use your NDSU email as username and the password you created to login.

Figure 4.15. Log in Page

4.5.1.3. Forgot Password or Reset Password

Forgot password and reset passwords are two separate functionalities related to password resets. Forgot password is in case the user has no idea what their password is and reset password is for users who are already logged into the system and wants to change their password.


Dissertation Calculator Register Login

Forgot your password?

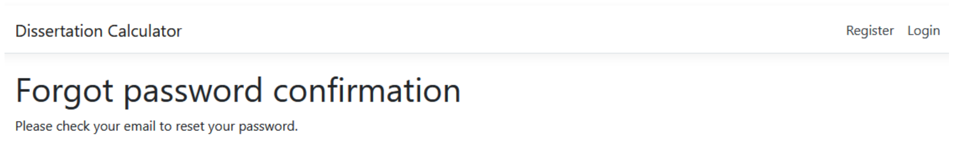
Enter your email.

Email

Submit

Figure 4.16. Forgot Password Page

Forgot password functionality uses your email address to send you an email with a link to reset your password.

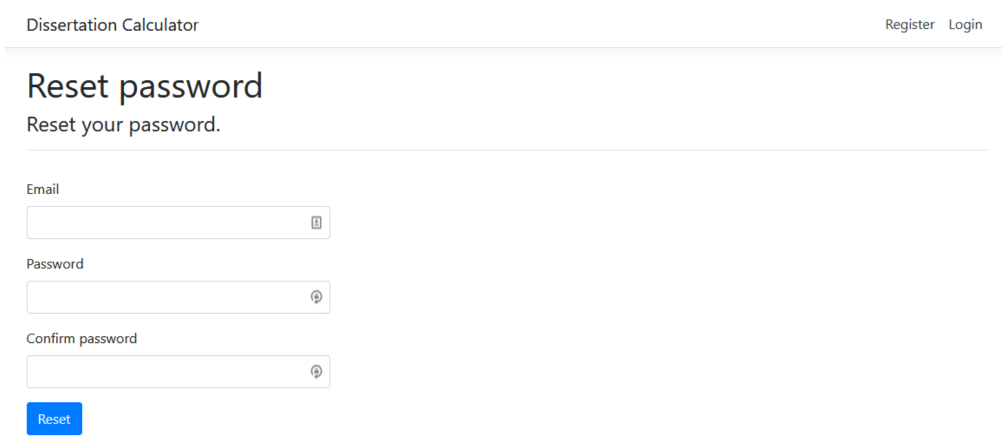

Dissertation Calculator Register Login

Forgot password confirmation

Please check your email to reset your password.

Figure 4.17. Forgot Password Email Confirmation Request

Once you click on the link, it will give you the option to set a new password without asking your old.


Dissertation Calculator Register Login

Reset password

Reset your password.

Email

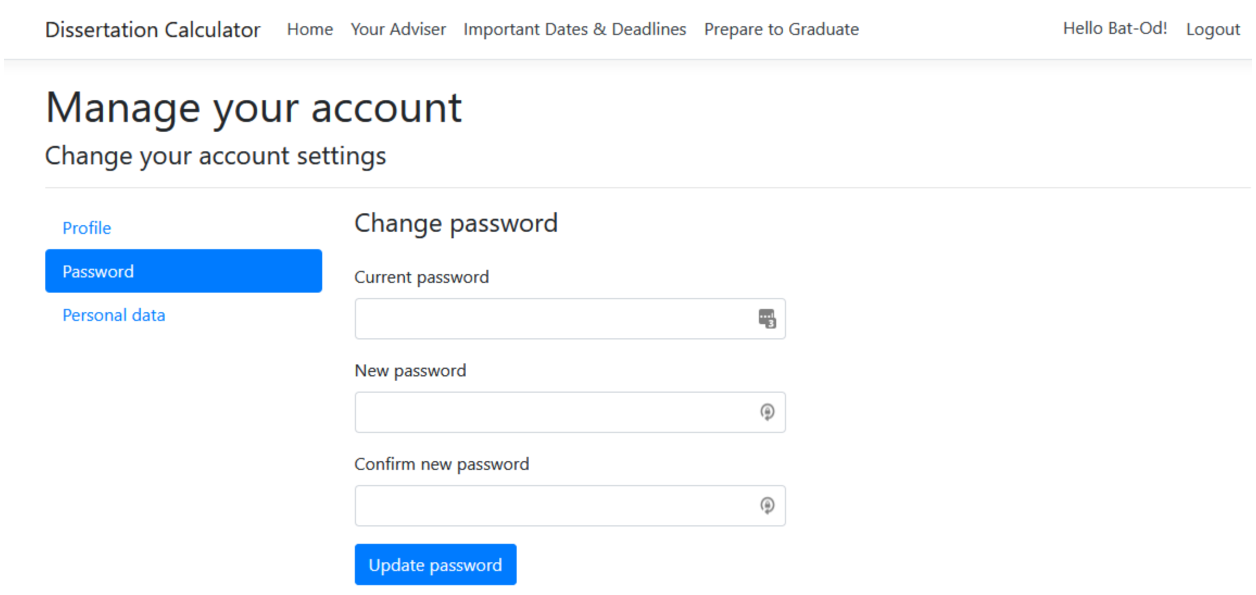
Password

Confirm password

Reset

Figure 4.18. Change Password Page After Clicking on Email Link

Reset password functionality requires the user to be logged in already and is accessible through the manage your account menu when you click on your name at the top right corner of the page. The user will have to enter their current password and the new password that they would like to use and click on update. Once password is changed, the application gives a status message saying it has been changed.



Manage your account

Change your account settings

Change password

Current password

New password

Confirm new password

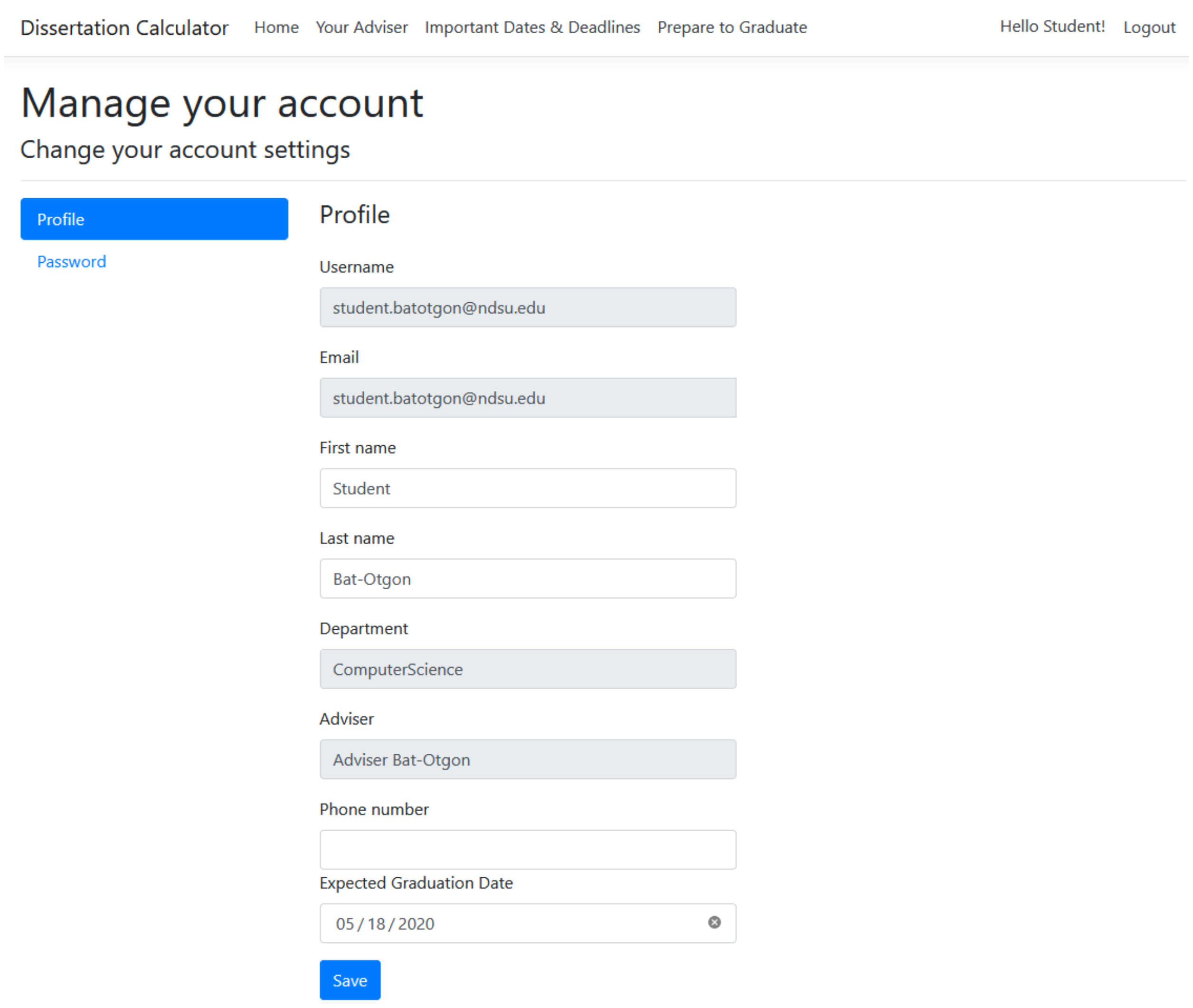
Figure 4.19. Reset Password Page

4.5.1.4. Logout

When user clicks on Logout button at the top right corner of the page, it expires the session of the user and redirects the user to the login page. ASP.NET Core Identity library has a class called “SignInManager” which is used in this case and the method called “SignOut” is called to log out the user by clearing the cookies of the sign in session.

4.5.1.5. Manage Account

Each user have the ability to manage their profiles. There are two options available which are “Profile” and “Password”. In “Profile” section user can change their name, set a phone number, and change their expected graduation date.



Manage your account

Change your account settings

Profile

Username: student.batotgon@ndsu.edu

Email: student.batotgon@ndsu.edu

First name: Student

Last name: Bat-Otgon

Department: ComputerScience

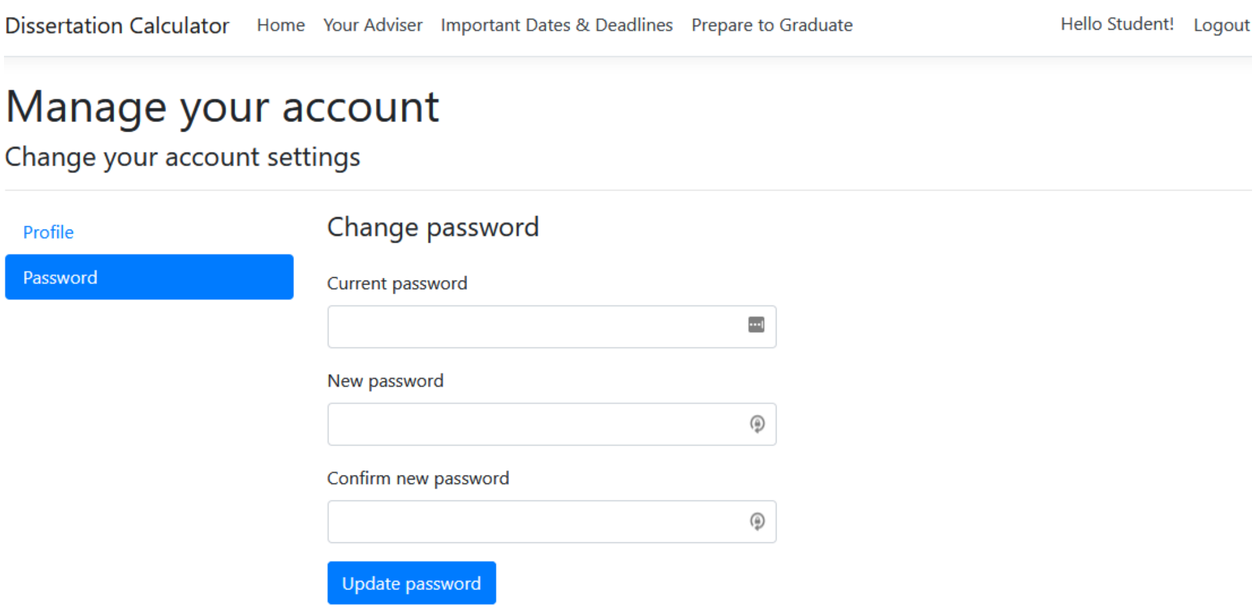
Adviser: Adviser Bat-Otgon

Phone number:

Expected Graduation Date: 05 / 18 / 2020

Figure 4.20. Profile Page

In “Password” section, user can change their password by entering the current password, entering a new password, and confirming the new password.


Dissertation Calculator Home Your Adviser Important Dates & Deadlines Prepare to Graduate Hello Student! Logout

Manage your account

Change your account settings

Profile

Password

Change password

Current password

New password

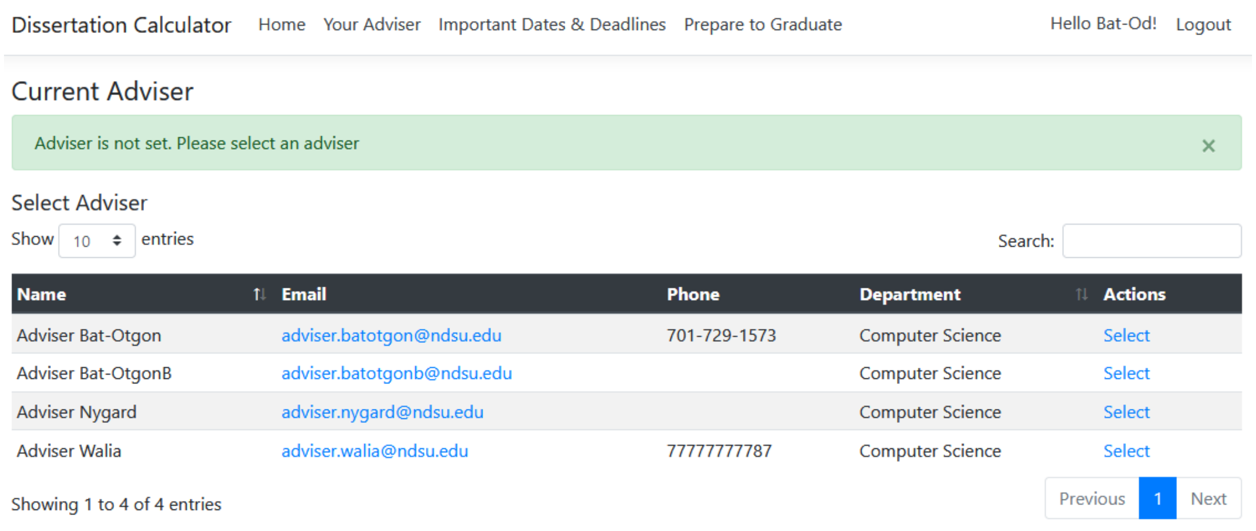
Confirm new password

Update password

Figure 4.21. Change Password Page

4.5.2. Student Functionalities

Student users can perform several functions while using the system. The first action that they must perform after login in is to select an adviser if they do not have an adviser set so adviser user must have an account already created and set as adviser role. If the user’s adviser does not have an account, they will have to request them to create one and then they can select their adviser from the list. User can also search and sort the adviser list.



Current Adviser

Adviser is not set. Please select an adviser

Select Adviser

Show 10 entries

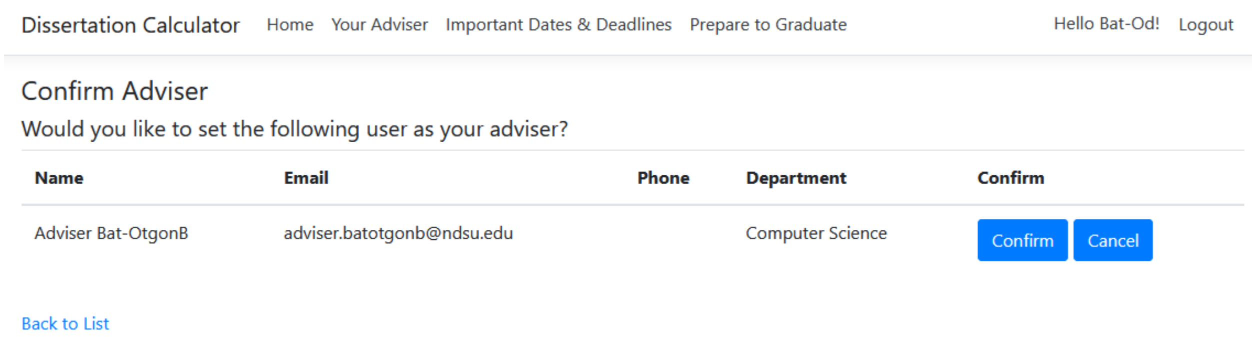
Search:

Name	Email	Phone	Department	Actions
Adviser Bat-Otgon	adviser.batotgon@ndsu.edu	701-729-1573	Computer Science	Select
Adviser Bat-OtgonB	adviser.batotgonb@ndsu.edu		Computer Science	Select
Adviser Nygard	adviser.nygard@ndsu.edu		Computer Science	Select
Adviser Walia	adviser.walia@ndsu.edu	77777777787	Computer Science	Select

Showing 1 to 4 of 4 entries

Figure 4.22. Select Adviser Page

After selecting an adviser, they will confirm the selection and click on "Confirm" button to finalize the selection and set the adviser as their adviser.



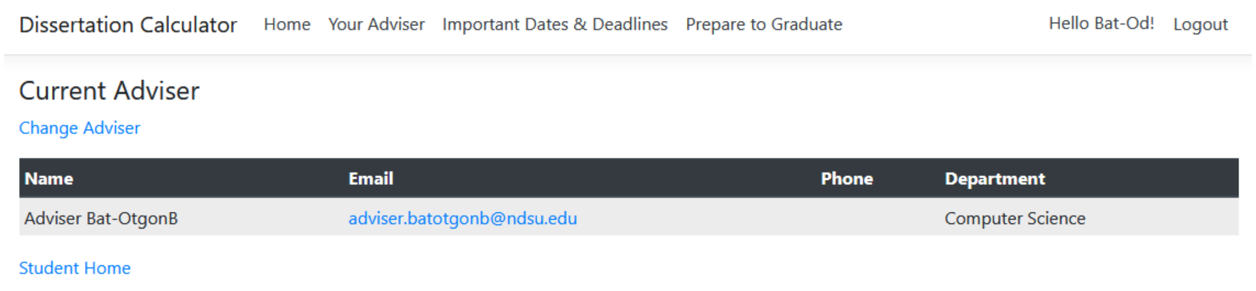
Confirm Adviser

Would you like to set the following user as your adviser?

Name	Email	Phone	Department	Confirm
Adviser Bat-OtgonB	adviser.batotgonb@ndsu.edu		Computer Science	Confirm Cancel

Figure 4.23. Confirm Adviser Page

Current adviser will be then displayed, and student can go to their home page.

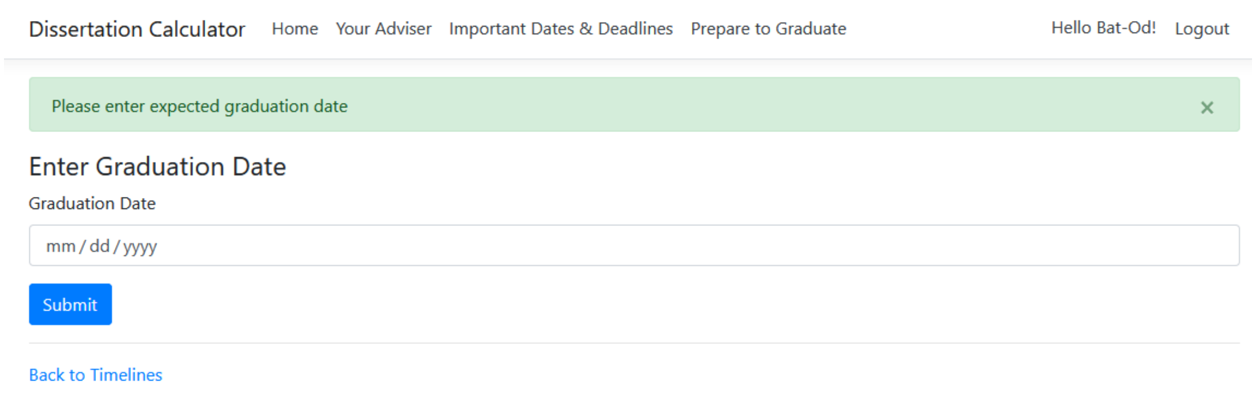


Current Adviser

Name	Email	Phone	Department
Adviser Bat-OtgonB	adviser.batotgonb@ndsu.edu		Computer Science

Figure 4.24. Current Adviser Page

After this, when student tries to go to the home page and if they have not entered an expected graduation date, the system will ask the student to enter a date and it will be saved to their profile. Student can view this information later in their profile and adviser can see the student's planned graduation date.



Please enter expected graduation date

Enter Graduation Date

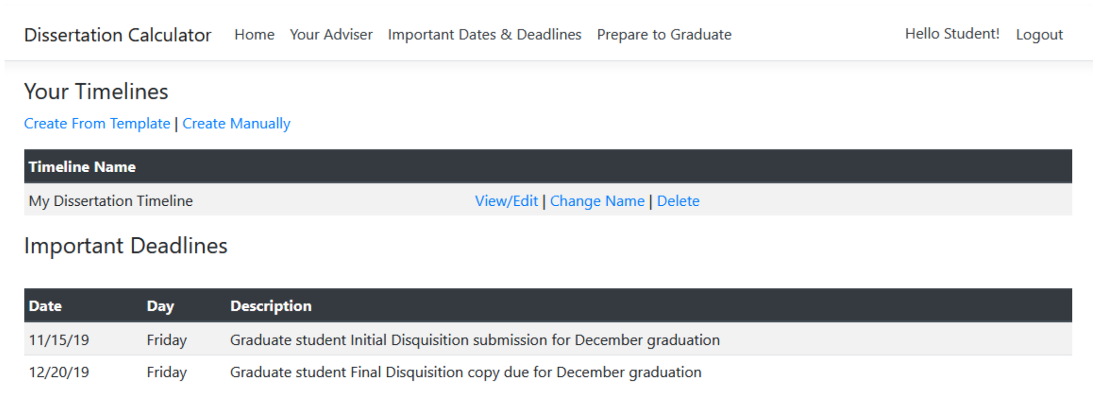
Graduation Date

mm / dd / yyyy

Figure 4.25. Enter Expected Graduation Page

On the home page, student has some options to navigate and perform actions. In the navigation student has the options "Home", "Your Adviser", "Important Dates & Deadlines", and "Prepare to Graduate". "Home" takes the student user to the landing page which is "Your Timelines" page. "Your Adviser" page shows the current adviser of the student and gives options to change the adviser if needed. "Important Dates & Deadlines" link opens a new tab in the browser and takes the user to NDSU's Records and Registration Office's Dates and Deadlines page. "Prepare to Graduate" link takes the user to NDSU's Graduate School page with information on preparing to graduate for graduate students.

From the home page, student can view their timeline if they have one created, otherwise they will have to create one using one of two options. First option is "Create From Template", which allows the student to use a template created by their adviser or publicly available that is made by other advisers and set to public status. Student's can change the name of their timeline or even delete it. If they delete the timeline, all the related information such as steps in the timeline will be deleted though the cascade delete in the database as these steps are no longer need to be on the database since the timeline is deleted.


Dissertation Calculator Home Your Adviser Important Dates & Deadlines Prepare to Graduate Hello Student! Logout

Your Timelines

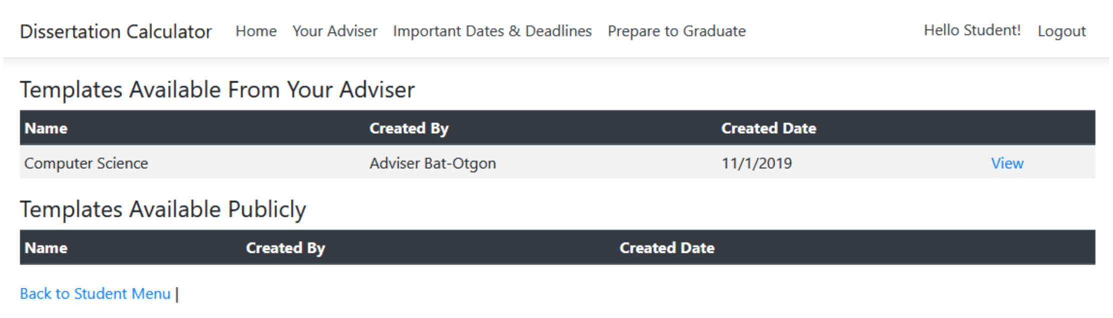
Create From Template | Create Manually

Timeline Name	
My Dissertation Timeline	View/Edit | Change Name | Delete

Important Deadlines

Date	Day	Description
11/15/19	Friday	Graduate student Initial Disquisition submission for December graduation
12/20/19	Friday	Graduate student Final Disquisition copy due for December graduation

Figure 4.26. Student Home Page


Dissertation Calculator Home Your Adviser Important Dates & Deadlines Prepare to Graduate Hello Student! Logout

Templates Available From Your Adviser

Name	Created By	Created Date	
Computer Science	Adviser Bat-Otgon	11/1/2019	View

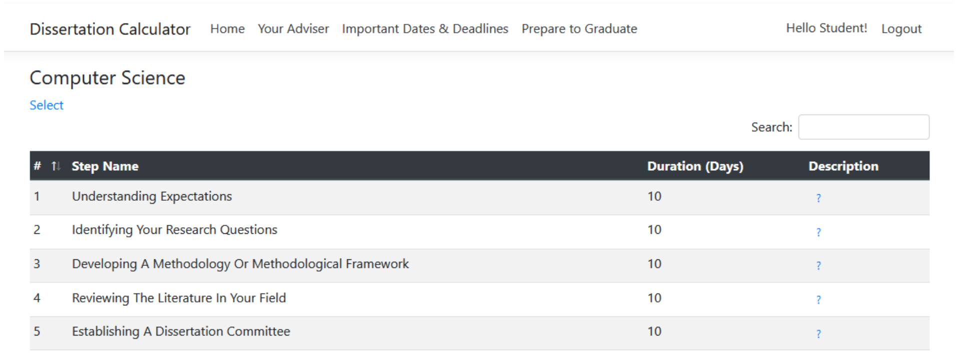
Templates Available Publicly

Name	Created By	Created Date

Back to Student Menu |

Figure 4.27. Create From Template Page

The user can click on "View" and see the template created by their adviser and select it to create their timeline. The user will have to enter a new name and confirm the selection to finalize the creation of their timeline based on the template.


Dissertation Calculator Home Your Adviser Important Dates & Deadlines Prepare to Graduate Hello Student! Logout

Computer Science

Select

Search:

#	Step Name	Duration (Days)	Description
1	Understanding Expectations	10	?
2	Identifying Your Research Questions	10	?
3	Developing A Methodology Or Methodological Framework	10	?
4	Reviewing The Literature In Your Field	10	?
5	Establishing A Dissertation Committee	10	?

Figure 4.28. Template Timeline View

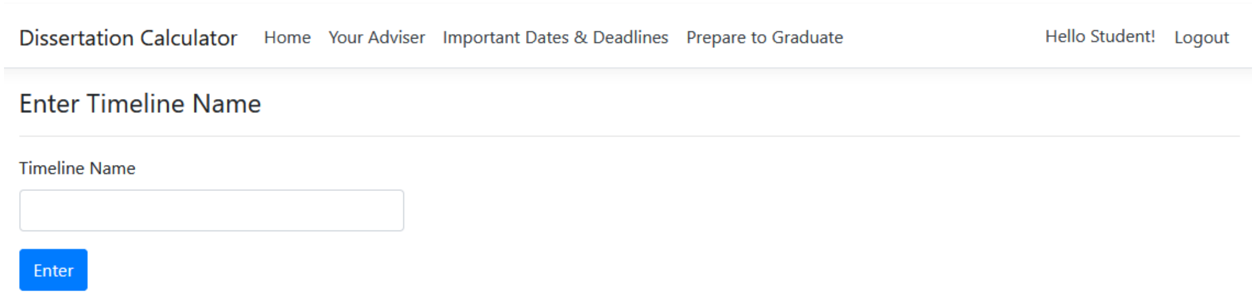
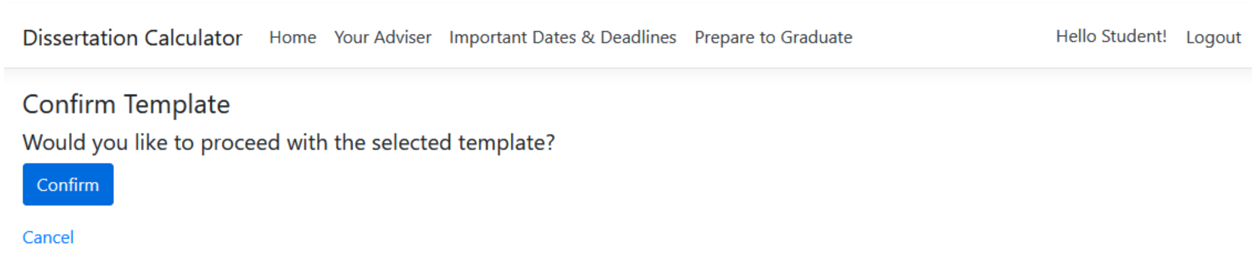


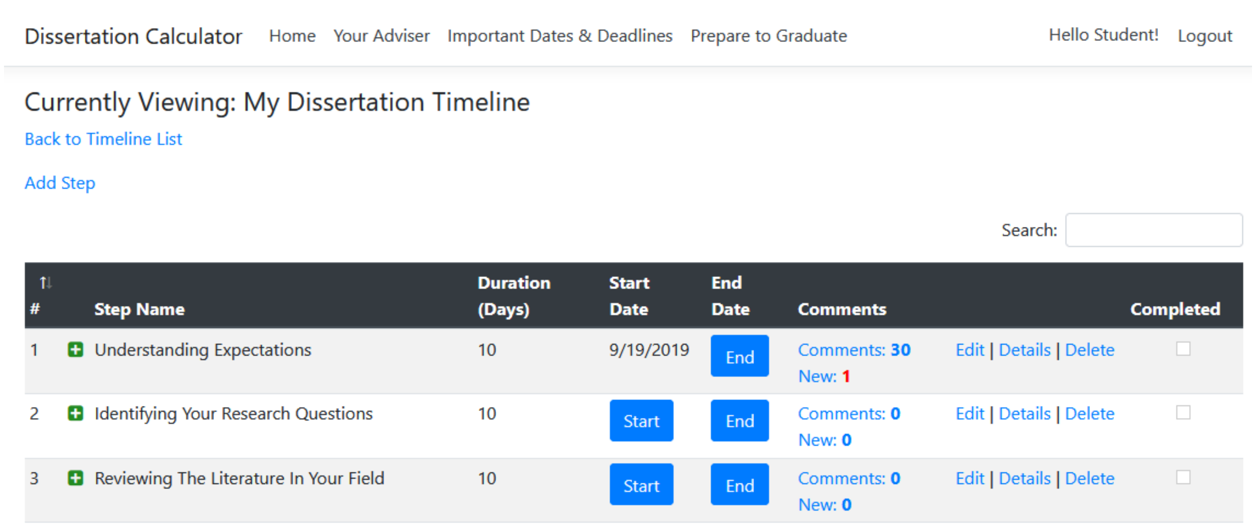
Figure 4.29. Enter New Name for New Timeline



Would you like to proceed with the selected template?

Figure 4.30. Confirm Creation of Timeline with Selected Template

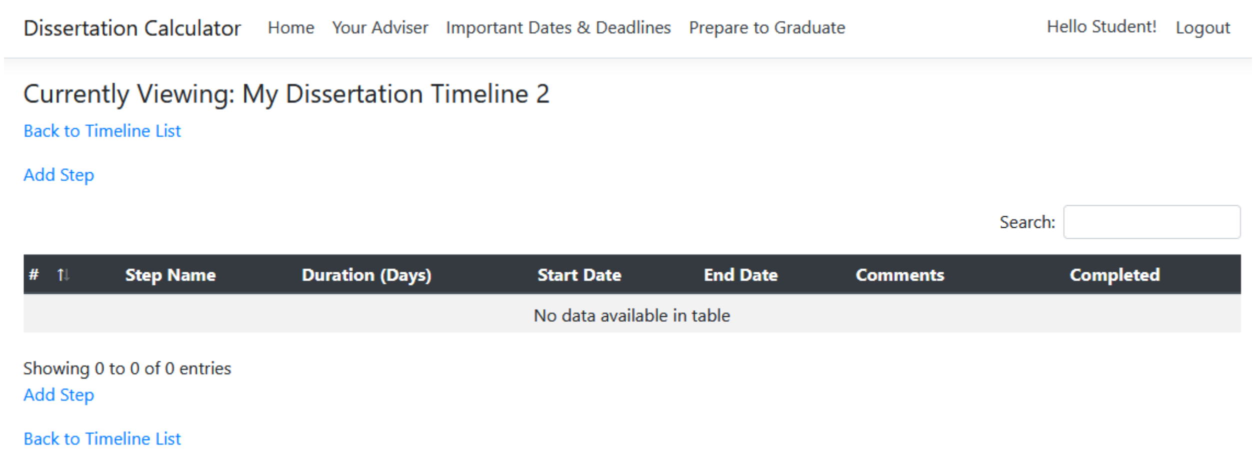
The timeline will be created, and the user will be redirected to the page with the timeline prepopulated with the steps from the template.



#	Step Name	Duration (Days)	Start Date	End Date	Comments		Completed
1	Understanding Expectations	10	9/19/2019	End	Comments: 30 New: 1	Edit | Details | Delete	☐
2	Identifying Your Research Questions	10	Start	End	Comments: 0 New: 0	Edit | Details | Delete	☐
3	Reviewing The Literature In Your Field	10	Start	End	Comments: 0 New: 0	Edit | Details | Delete	☐

Figure 4.31. Student Timeline View

Second option is “Create Manually” which allows the student to create a timeline on their own with customizing each step for their own. Student will need to enter the timeline name and allow the student to customize with their own steps.


Dissertation Calculator Home Your Adviser Important Dates & Deadlines Prepare to Graduate Hello Student! Logout

Currently Viewing: My Dissertation Timeline 2

Back to Timeline List

Add Step

Search:

#	Step Name	Duration (Days)	Start Date	End Date	Comments	Completed
No data available in table						

Showing 0 to 0 of 0 entries

Add Step

Back to Timeline List

Figure 4.32. Blank Student Timeline

Important Dates and Deadlines will show the dates for the current year all the time. This information is entered by the administrator in the system and will display on the home page of student's as these dates are very important.

4.5.2.1. Student Timeline

The student timeline is the main feature of this tool. Without this functionality, this tool would not be as useful. There are number of actions a student user can perform on a timeline. First, they can add a step of their own if needed even if the timeline is created using a template. New steps will always get added at the bottom of the timeline.

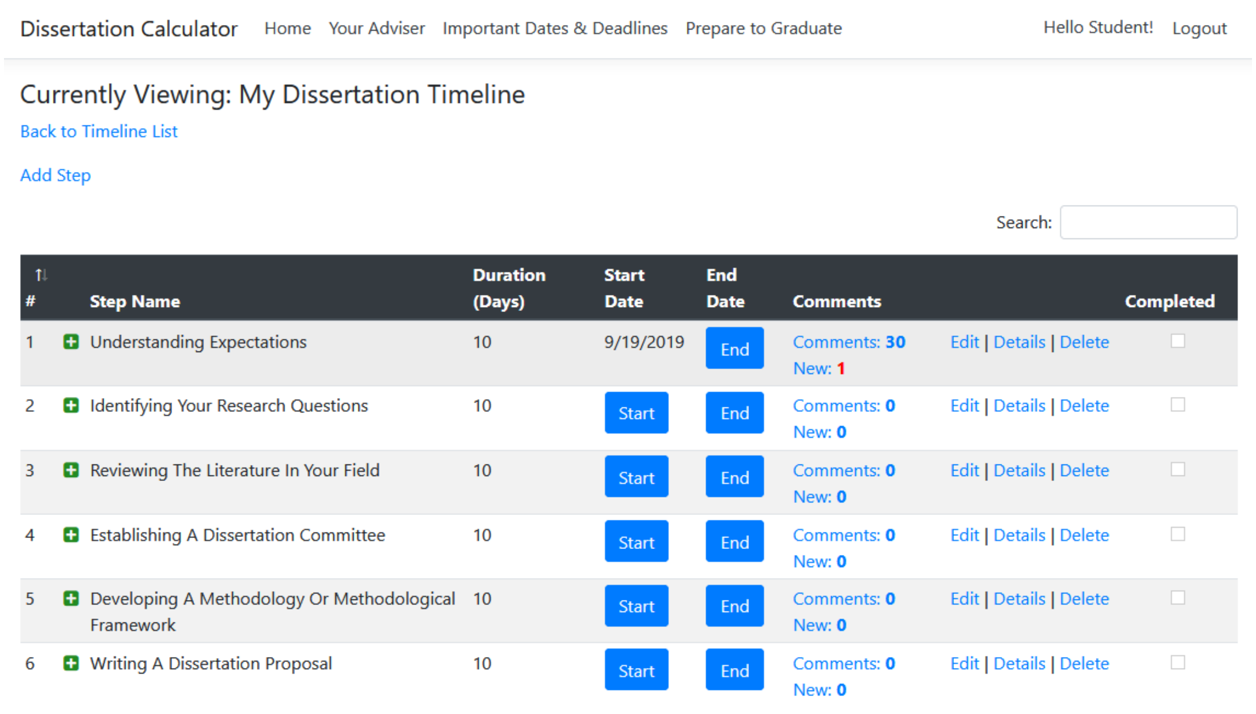
If the student wants to reorganize the steps, they can do so by dragging the steps up and down. The ordering of steps will get automatically saved and step number will be updated accordingly in the database.

Step descriptions are hidden in a child row and can be viewed by click on the green plus sign in front of the step name.

Start and End Date columns are clickable buttons and if the student click them, it will enter the date of the current date and time when the student clicks on it.

Comments link will take the student user to the comments page for that specific step where adviser might have left a feedback on it or the student can leave a feedback to their adviser as well. It will also show the total number of comments and if there is a new comment made by their adviser, it will indicate it in the “New” field with a red number.

Edit, details, delete links perform exactly what the names describe. Edit link will allow the student to edit the step name, duration, start date, end date, and completed status. Complete status column on the timeline will show up as marked if End Date is entered or Complete check box is marked. It works both ways. Details link will take the user to a page and show all the details of the step. Delete will allow the user to delete a step and once a step is deleted, all the other steps are updated with new step number automatically in case a step is deleted from top or middle.



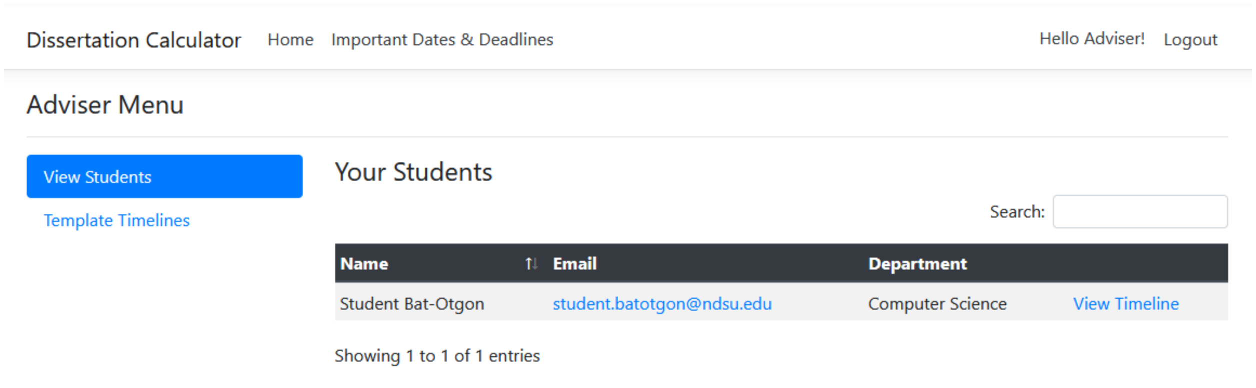
#	Step Name	Duration (Days)	Start Date	End Date	Comments		Completed
1	Understanding Expectations	10	9/19/2019	End	Comments: 30 New: 1	Edit | Details | Delete	☐
2	Identifying Your Research Questions	10	Start	End	Comments: 0 New: 0	Edit | Details | Delete	☐
3	Reviewing The Literature In Your Field	10	Start	End	Comments: 0 New: 0	Edit | Details | Delete	☐
4	Establishing A Dissertation Committee	10	Start	End	Comments: 0 New: 0	Edit | Details | Delete	☐
5	Developing A Methodology Or Methodological Framework	10	Start	End	Comments: 0 New: 0	Edit | Details | Delete	☐
6	Writing A Dissertation Proposal	10	Start	End	Comments: 0 New: 0	Edit | Details | Delete	☐

Figure 4.33. Student Timeline

4.5.3. Adviser Functionalities

Adviser users have two main functionalities that are viewing students and their timelines, and creating, modifying, and deleting template timelines that can be used by students. There is a left pane menu called "Adviser Menu" with two options "View Students" and "Template Timelines".

"View Students" will allow to see students who have selected the adviser user as their adviser. It will show their name, email address, department, and link to their timeline.



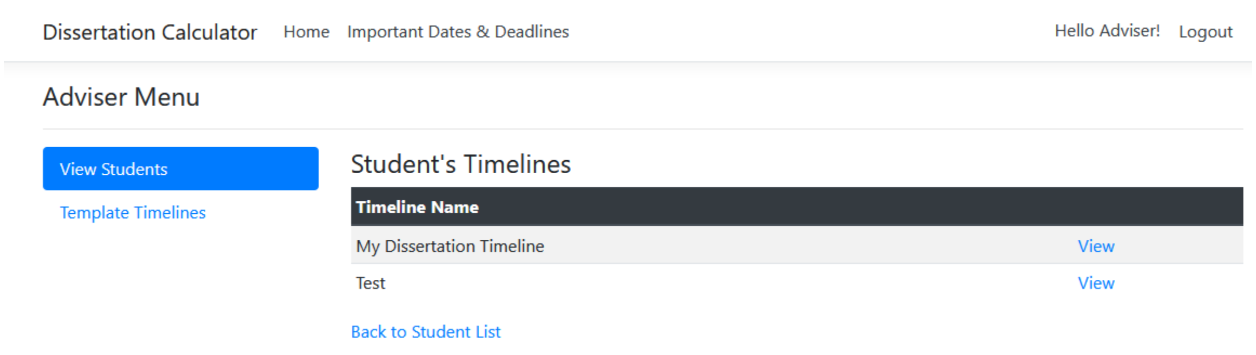
Name	Email	Department	
Student Bat-Otgon	student.batotgon@ndsu.edu	Computer Science	View Timeline

Showing 1 to 1 of 1 entries

Figure 4.34. Adviser Home Page - View Students

When you click on the email address, it will automatically launch the default email client on the user's computer and allow them to send an email right away. Otherwise, the adviser user can sort the table with student name and do search with any keyword that might be found in the three columns.

"View Timelines" link next to the student's row will allow the adviser to see their student's timeline and check on their progress. If the student has multiple timelines, it will display the names of their timelines.



Dissertation Calculator Home Important Dates & Deadlines Hello Adviser! Logout

Adviser Menu

View Students
Template Timelines

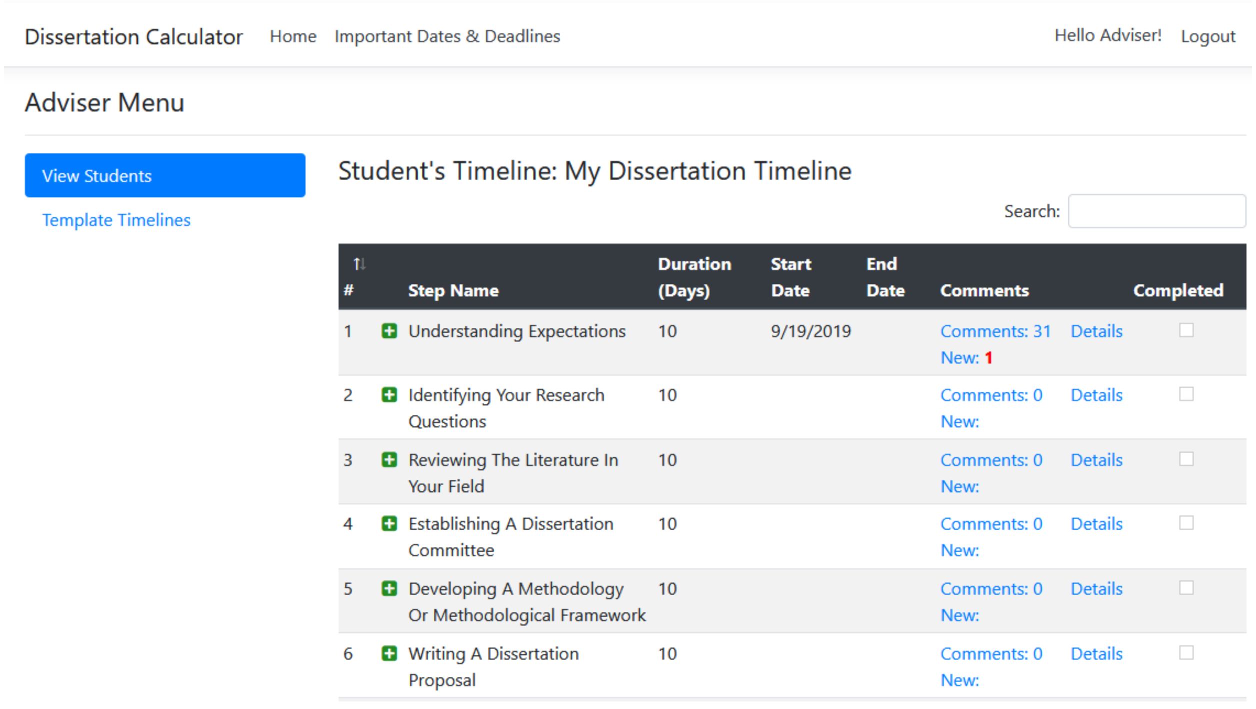
Student's Timelines

Timeline Name	
My Dissertation Timeline	View
Test	View

Back to Student List

Figure 4.35. Adviser Viewing Student Timelines

Adviser user then will have to click on "View" link to view the actual timeline and see the progress.



Dissertation Calculator Home Important Dates & Deadlines Hello Adviser! Logout

Adviser Menu

View Students
Template Timelines

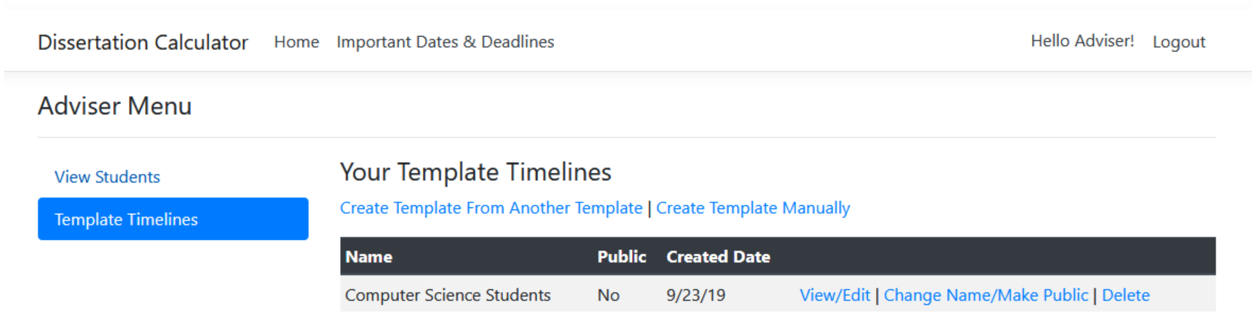
Student's Timeline: My Dissertation Timeline

Search:

#	Step Name	Duration (Days)	Start Date	End Date	Comments		Completed
1	Understanding Expectations	10	9/19/2019		Comments: 31 New: 1	Details	☐
2	Identifying Your Research Questions	10			Comments: 0 New:	Details	☐
3	Reviewing The Literature In Your Field	10			Comments: 0 New:	Details	☐
4	Establishing A Dissertation Committee	10			Comments: 0 New:	Details	☐
5	Developing A Methodology Or Methodological Framework	10			Comments: 0 New:	Details	☐
6	Writing A Dissertation Proposal	10			Comments: 0 New:	Details	☐

Figure 4.36. Adviser Viewing Student Timeline

Adviser's view of student's timeline is very similar to how a student view's their timeline but advisers can only see the information and will not be able to make any modifications to the timeline. Only other action they can perform on the timeline is to comment on a specific step. If a student user left a new comment on a step and the adviser has not read it yet. It will show as new comment in the comment column. Adviser users can see the details of each step as well and the completed status, start date, and end date.



Your Template Timelines

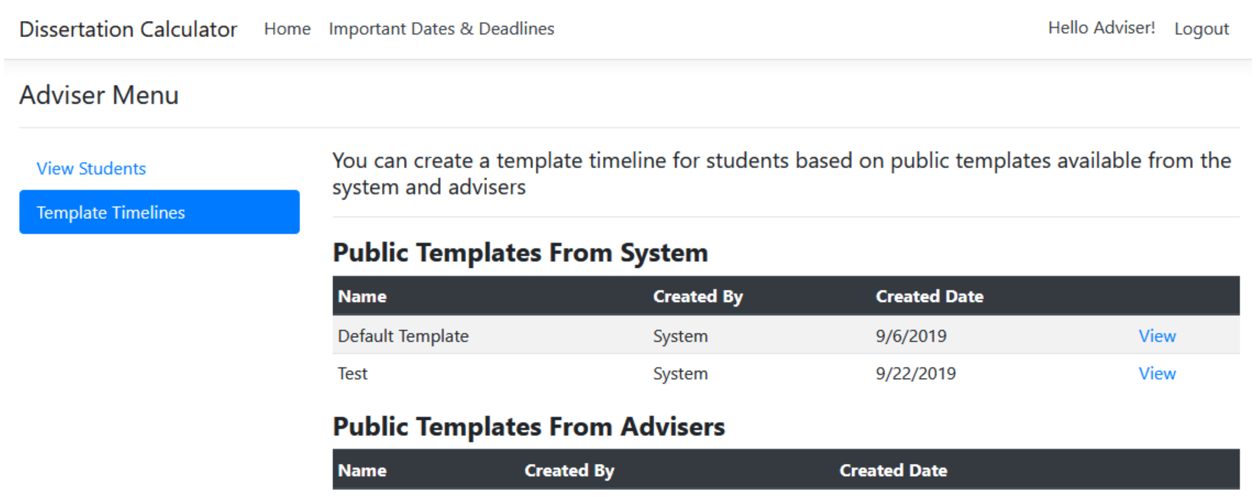
Create Template From Another Template | Create Template Manually

Name	Public	Created Date	
Computer Science Students	No	9/23/19	View/Edit | Change Name/Make Public | Delete

Figure 4.37. Adviser's Template Timelines

The next major functionality that advisers can use is creating templates for students to use. If adviser user has template created, the table will show the name, public status, created date, and actions. There are 3 actions available which are “View/Edit”, “Change Name/Make Public”, and “Delete”. “View/Edit” will allow the adviser to view the template timeline and make edits. “Change Name/Make Public” will allow the adviser user to change the name and mark or unmark the template as public. “Delete” will allow the adviser user to delete the template timeline. This will not delete any student timelines created by using the template.

Adviser users can create new template with two options which are “Create Template From Another Template” and “Create Template Manually”. “Create Template From Another Template” option allows the adviser user to create their template from a base template made available by the system administrator or templates made available public by other advisers.


Dissertation Calculator Home Important Dates & Deadlines Hello Adviser! Logout

Adviser Menu

View Students
Template Timelines

You can create a template timeline for students based on public templates available from the system and advisers

Public Templates From System

Name	Created By	Created Date	
Default Template	System	9/6/2019	View
Test	System	9/22/2019	View

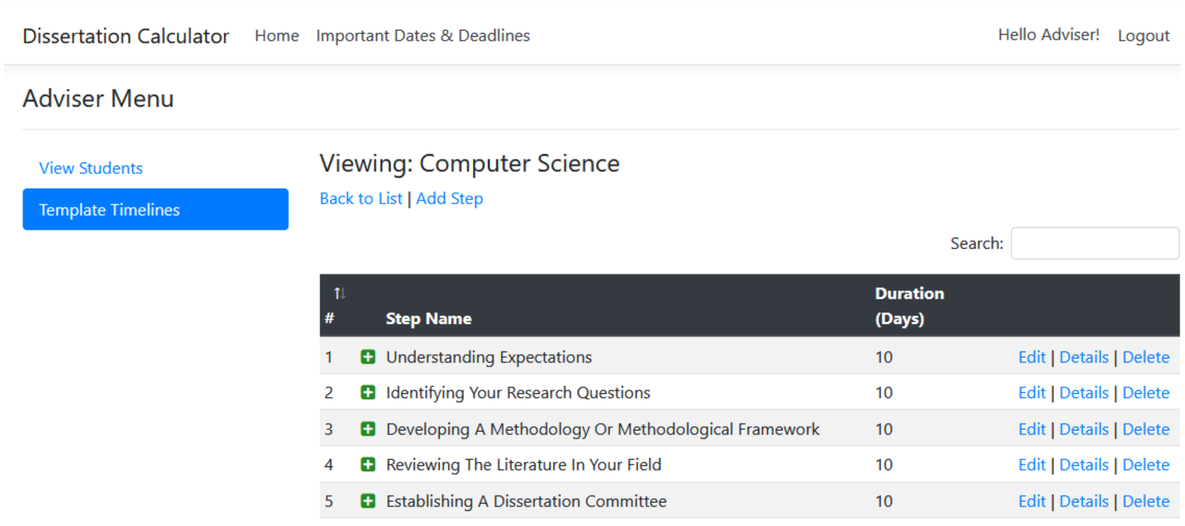
Public Templates From Advisers

Name	Created By	Created Date

Figure 4.38. Public Templates Available to Advisers

When the adviser user clicks on View to see the system template, it will allow them to select it, give it a name of their own, confirm the selection, and view the newly created template from base template.

When viewing a templated that was created, it will allow the adviser user to add new steps, reorganize the steps by dragging and dropping on the timeline, edit step name, step description, and step duration, vie details of each step, and finally delete a step. So all the customization is available for the user.


Dissertation Calculator Home Important Dates & Deadlines Hello Adviser! Logout

Adviser Menu

View Students
Template Timelines

Viewing: Computer Science

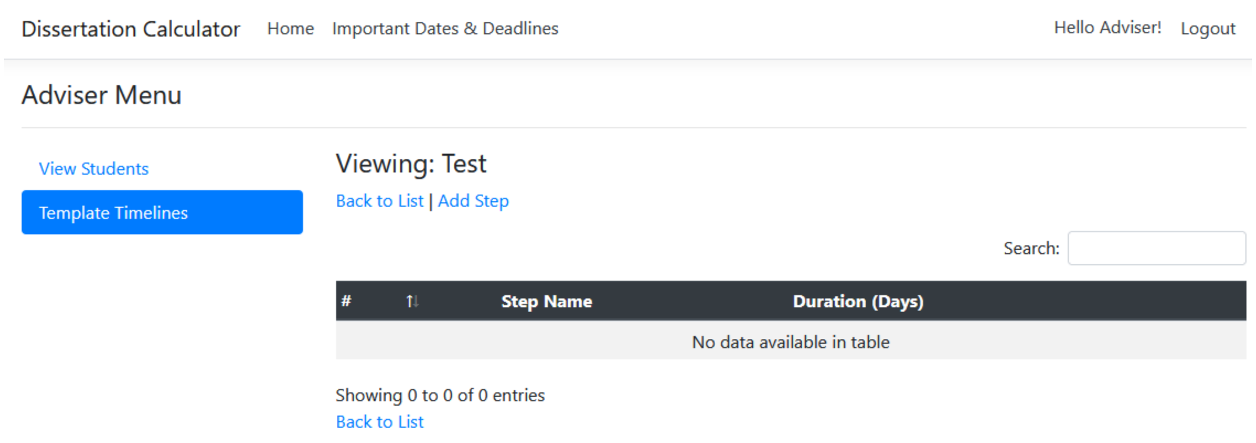
Back to List | Add Step

Search:

#	Step Name	Duration (Days)	
1	Understanding Expectations	10	Edit | Details | Delete
2	Identifying Your Research Questions	10	Edit | Details | Delete
3	Developing A Methodology Or Methodological Framework	10	Edit | Details | Delete
4	Reviewing The Literature In Your Field	10	Edit | Details | Delete
5	Establishing A Dissertation Committee	10	Edit | Details | Delete

Figure 4.39. Adviser Template Timeline View

"Create Template Manually" will allow the adviser user to start with a blank template and customize the steps themselves.


Dissertation Calculator Home Important Dates & Deadlines Hello Adviser! Logout

Adviser Menu

View Students
Template Timelines

Viewing: Test

Back to List | Add Step

Search:

#	Step Name	Duration (Days)
No data available in table		

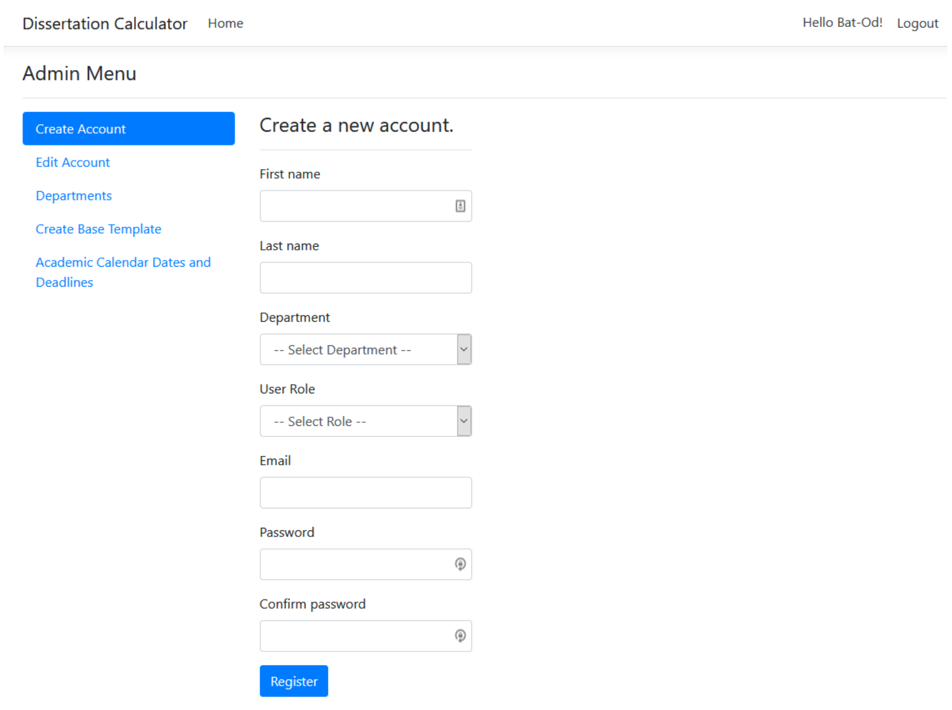
Showing 0 to 0 of 0 entries

Back to List

Figure 4.40. Adviser Blank Template Timeline

4.5.4. Admin Functionalities

Admin users have 5 main functionalities available to them which are "Create Account", "Edit Account", "Departments", "Create Base Template", and "Academic Calendar Dates and Deadlines"


Dissertation Calculator Home Hello Bat-Od! Logout

Admin Menu

Create Account
Edit Account
Departments
Create Base Template
Academic Calendar Dates and Deadlines

Create a new account.

First name

Last name

Department

-- Select Department --

User Role

-- Select Role --

Email

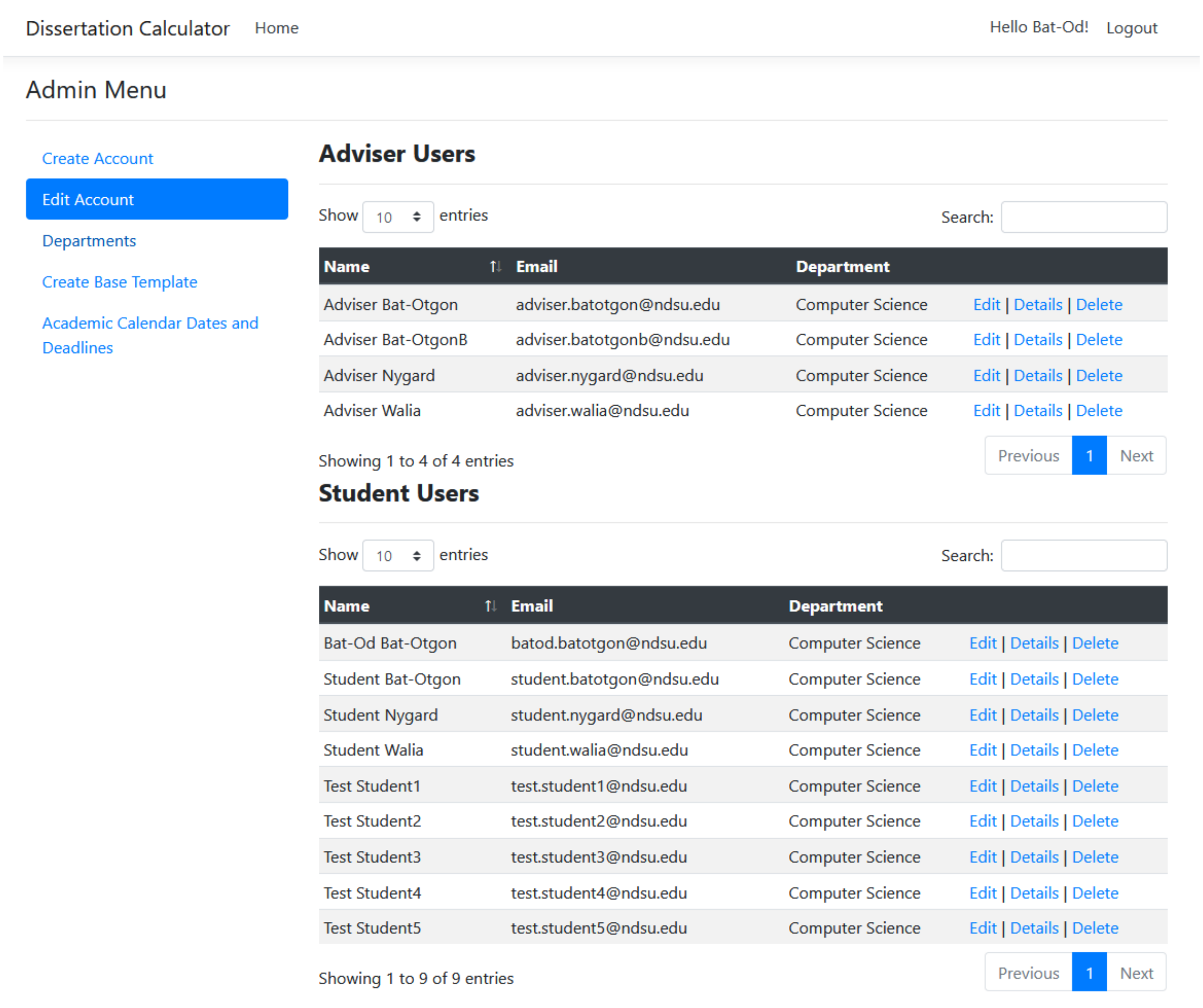
Password

Confirm password

Register

Figure 4.41. Admin Create Account Page

“Create Account” will allow the admin users to create an account in the system. Unlike new users who are required to use their NDSU email address to register, admin can create a user with any email address. Admin users can also create other admin accounts.



Adviser Users

Show 10 entries   Search:

Name	Email	Department	
Adviser Bat-Otgon	adviser.batotgon@ndsu.edu	Computer Science	Edit | Details | Delete
Adviser Bat-OtgonB	adviser.batotgonb@ndsu.edu	Computer Science	Edit | Details | Delete
Adviser Nygard	adviser.nygard@ndsu.edu	Computer Science	Edit | Details | Delete
Adviser Walia	adviser.walia@ndsu.edu	Computer Science	Edit | Details | Delete

Showing 1 to 4 of 4 entries   Previous 1 Next

Student Users

Show 10 entries   Search:

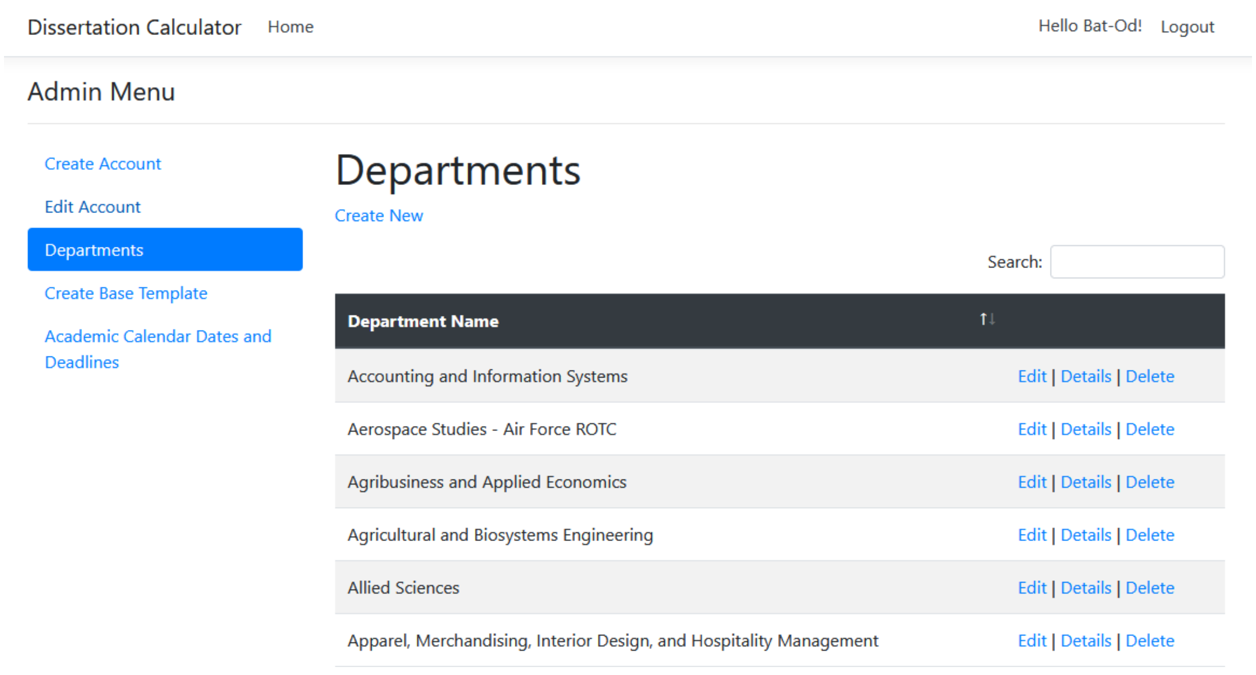
Name	Email	Department	
Bat-Od Bat-Otgon	batod.batotgon@ndsu.edu	Computer Science	Edit | Details | Delete
Student Bat-Otgon	student.batotgon@ndsu.edu	Computer Science	Edit | Details | Delete
Student Nygard	student.nygard@ndsu.edu	Computer Science	Edit | Details | Delete
Student Walia	student.walia@ndsu.edu	Computer Science	Edit | Details | Delete
Test Student1	test.student1@ndsu.edu	Computer Science	Edit | Details | Delete
Test Student2	test.student2@ndsu.edu	Computer Science	Edit | Details | Delete
Test Student3	test.student3@ndsu.edu	Computer Science	Edit | Details | Delete
Test Student4	test.student4@ndsu.edu	Computer Science	Edit | Details | Delete
Test Student5	test.student5@ndsu.edu	Computer Science	Edit | Details | Delete

Showing 1 to 9 of 9 entries   Previous 1 Next

Figure 4.42. Admin Edit Account Page

“Edit Account” option allows the admin users to edit, view details, and delete adviser and student user accounts. “Edit” will allow changing names, department, role, phone number, adviser and expected graduation date if the account is a student account. “Details” link will show the detailed information of the user account. “Delete” will delete the account from the system along with its personal data like timelines and such from the database.

"Departments" option allows the admin users to create, edit, view details, and delete departments. Because this is needed when users are registering their account, changing departments, and finding their advisers in the same department.


Dissertation Calculator Home Hello Bat-Od! Logout

Admin Menu

Create Account
Edit Account
Departments
Create Base Template
Academic Calendar Dates and Deadlines

Departments

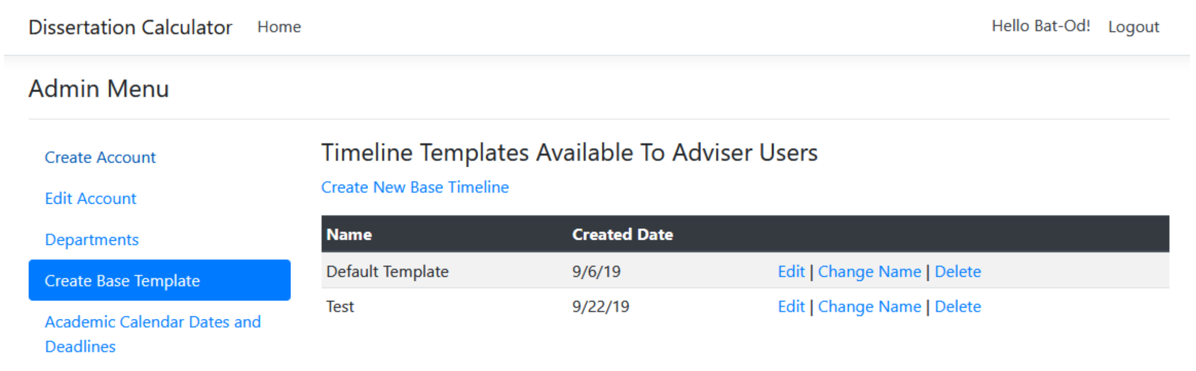
Create New

Search:

Department Name	
Accounting and Information Systems	Edit | Details | Delete
Aerospace Studies - Air Force ROTC	Edit | Details | Delete
Agribusiness and Applied Economics	Edit | Details | Delete
Agricultural and Biosystems Engineering	Edit | Details | Delete
Allied Sciences	Edit | Details | Delete
Apparel, Merchandising, Interior Design, and Hospitality Management	Edit | Details | Delete

Figure 4.43. Admin Departments Page

"Create Base Template" option allows the admin user to create a base template that is available to all adviser users who can use it as a foundation for a template that they are creating for their students. Admin users can also edit, change the name, and delete a base template that is already available.


Dissertation Calculator Home Hello Bat-Od! Logout

Admin Menu

Create Account
Edit Account
Departments
Create Base Template
Academic Calendar Dates and Deadlines

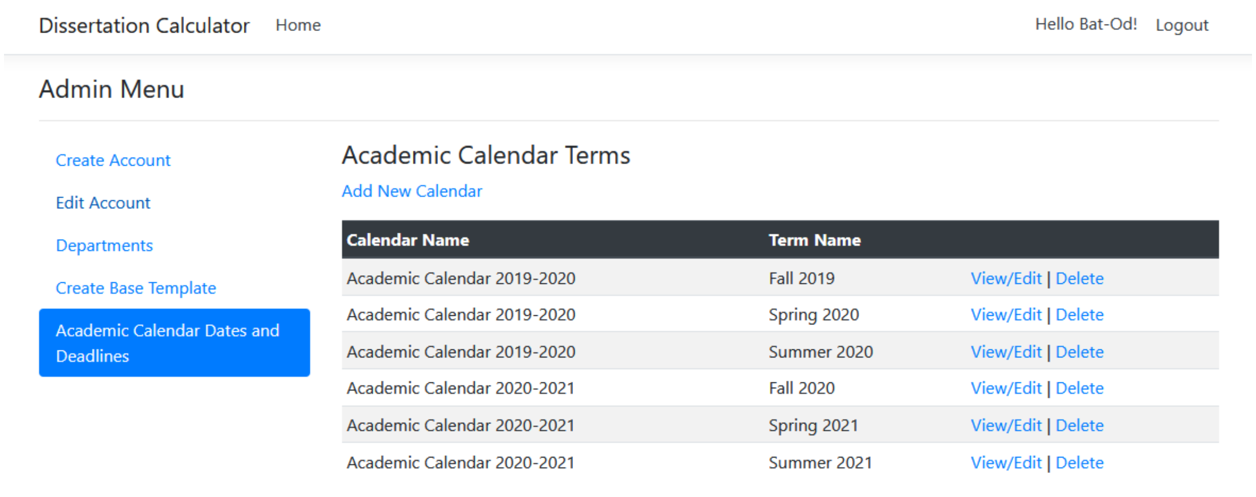
Timeline Templates Available To Adviser Users

Create New Base Timeline

Name	Created Date	
Default Template	9/6/19	Edit | Change Name | Delete
Test	9/22/19	Edit | Change Name | Delete

Figure 4.44. Admin Create Base Template Page

The last function that admin users can use is “Academic Calendar Dates and Deadlines”. With this option, they can create terms for each semester like Fall, Spring, and Summer. Each term can have specific dates in them, and admin users can add them to a term by clicking on “View/Edit”.

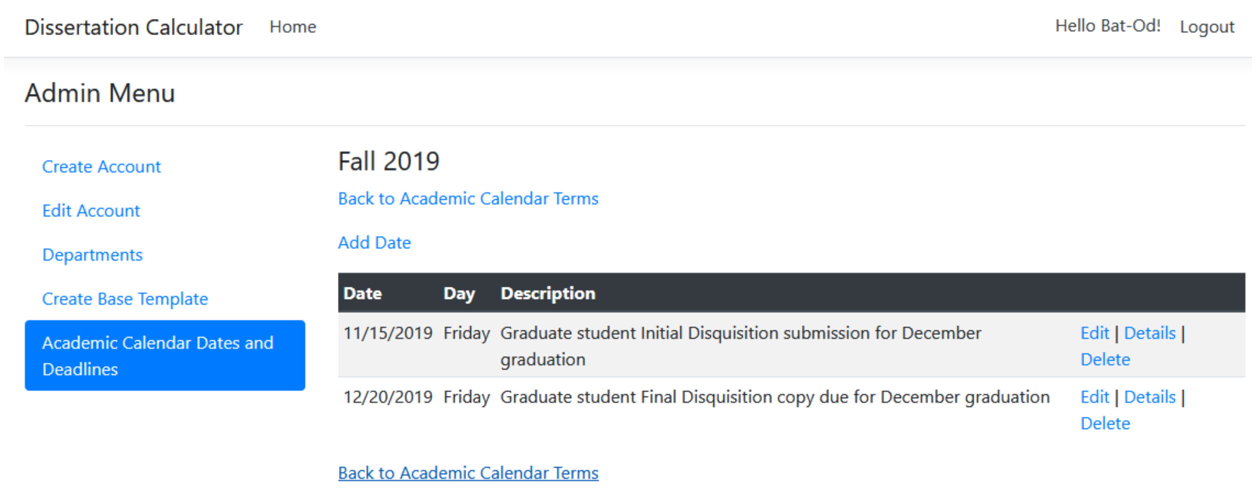


Academic Calendar Terms

Add New Calendar

Calendar Name	Term Name	
Academic Calendar 2019-2020	Fall 2019	View/Edit | Delete
Academic Calendar 2019-2020	Spring 2020	View/Edit | Delete
Academic Calendar 2019-2020	Summer 2020	View/Edit | Delete
Academic Calendar 2020-2021	Fall 2020	View/Edit | Delete
Academic Calendar 2020-2021	Spring 2021	View/Edit | Delete
Academic Calendar 2020-2021	Summer 2021	View/Edit | Delete

Figure 4.45. Admin Academic Calendar Dates and Deadlines Page



Fall 2019

Back to Academic Calendar Terms

Add Date

Date	Day	Description	
11/15/2019	Friday	Graduate student Initial Disquisition submission for December graduation	Edit | Details | Delete
12/20/2019	Friday	Graduate student Final Disquisition copy due for December graduation	Edit | Details | Delete

Back to Academic Calendar Terms

Figure 4.46. Admin Academic Calendar Dates and Deadlines Term Edit Page

4.6. Improvements

When compared to the initial project developed by students of CSCI 431, the new version of Dissertation Calculator has more features. The authentication and authorization is all handled by ASP.NET Core Identity library so it is lot more secure and feature ready. Passwords have requirements, users can only register using their NDSU email, when registering they need to verify their email address by click on a link sent by email, etc.

Student users create timelines based on a template created by their adviser, select and change advisers, comment on a step for feedback exchange with adviser, etc. On a student timeline, they can add, reorganize, delete, edit steps. Each step has descriptions and it is shown from a hidden row on the table with intuitive expanding button function.

Adviser users can see student timelines and see their progress and provide feedback by commenting. They can also create a template that can be used by student users.

Admin users can create, modify, and remove accounts. They can also create, modify, and remove departments. Another functionality that they can do is create, modify, and remove base template that can be used by any adviser so that they can create their templates for students. The last thing they can do is create, modify, and remove academic calendar items such as important dates and deadlines.

5. CONCLUSION

Dissertation Calculator tool is a unique tool and can be very useful for any graduate students who wants to complete their dissertation on time based on their expected end or graduation date. This tool has the potential to be tool that is used by graduate students and maybe even licensed to other schools as a tool. The development of this tool was a great learning experience and a way to improve and educate myself in planning, designing, coding, and testing a whole project. ASP.NET Core is an easy to learn framework with lots of features that make it easier to develop specific functionalities. It also has tons of documentation provided by Microsoft for new developers to get started with it and develop a web application fast and easy. As for the usability, feedback, and importance of the tool, it needs to be tested with real world users who will be using the system which would require careful planning and organization with other users.

6. FUTURE WORK

Although, the tool has many features, there are still things that could be added to make the experience much better.

- Find a way to incorporate the important dates and deadlines into student timeline.
- Reply system in the commenting feature as currently, there is no direct reply to a specific comment.
- Comments could be displayed under each step by expanding the child row in the timeline instead of going to a specific page to be viewed.
- Guide or Tutorial page where users can go to see how to use the system for each stakeholder.
- Chatting feature for advisers and students.
- Progress report for both students and advisers in email or PDF format
- Email notification for student users.
- Possible integration with NDSU's Central Authentication Service for authentication and account importing.
- Purge system to remove users who are no longer part of NDSU.
- Have a super admin who can manage other admin users.
- File attachment feature for each step where specific document might be needed
- Notification inbox feature where new comment or message notification is displayed.

REFERENCES

[1] Anderson, R. (2018, October 23). Scaffold Identity in ASP.NET Core projects. Retrieved October 31, 2019, from https://docs.microsoft.com/en-us/aspnet/core/security/authentication/scaffold-identity?view=aspnetcore-3.0&tabs=visual-studio.

[2] Anderson, R., & Nowak, R. (2019, October 6). Introduction to Razor Pages in ASP.NET Core. Retrieved October 31, 2019, from https://docs.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-2.2&tabs=visual-studio.

[3] ASP.NET Core fundamentals. (2019, October 6). Retrieved October 31, 2019, from https://docs.microsoft.com/en-us/aspnet/core/fundamentals/?view=aspnetcore-3.0&tabs=linux.

[4] Baker University's Dissertation Calculator. (n.d.). Retrieved from https://www.bakeru.edu/library/Dissertation calculator/assignmentcalc/.

[5] Child rows (show extra / detailed information). (n.d.). Retrieved from https://datatables.net/examples/api/row_details.html.

[6] Dissertation and Thesis Calculator: RIT Libraries. (n.d.). Retrieved from https://library.rit.edu/researchguides/disscalc/.

[7] Dissertation Calculator. (n.d.). Retrieved from https://www.lib.umn.edu/ac/dissertation-calculator.

[8] Dykstra, T., & Anderson, R. (2019, September 25). Razor Pages with Entity Framework Core in ASP.NET Core - Tutorial 1 of 8. Retrieved October 31, 2019, from https://docs.microsoft.com/en-us/aspnet/core/data/ef-rp/intro?view=aspnetcore-2.2&tabs=visual-studio.

[9] FixedHeader. (n.d.). Retrieved from https://datatables.net/extensions/fixedheader/.

[10] Get started with SQL Server on Ubuntu - SQL Server. (2019, May 27). Retrieved October 31, 2019, from https://docs.microsoft.com/en-us/sql/linux/quickstart-install-connect-ubuntu?view=sql-server-ver15.

[11] Prerequisites for .NET Core on Linux. (2019, October 10). Retrieved October 31, 2019, from https://docs.microsoft.com/en-us/dotnet/core/linux-prerequisites?tabs=netcore30.

[12] Research Paper Assistance Calculators // Libraries // Mizzou // University of Missouri. (n.d.). Retrieved from https://library.missouri.edu/guides/assigncalc/.

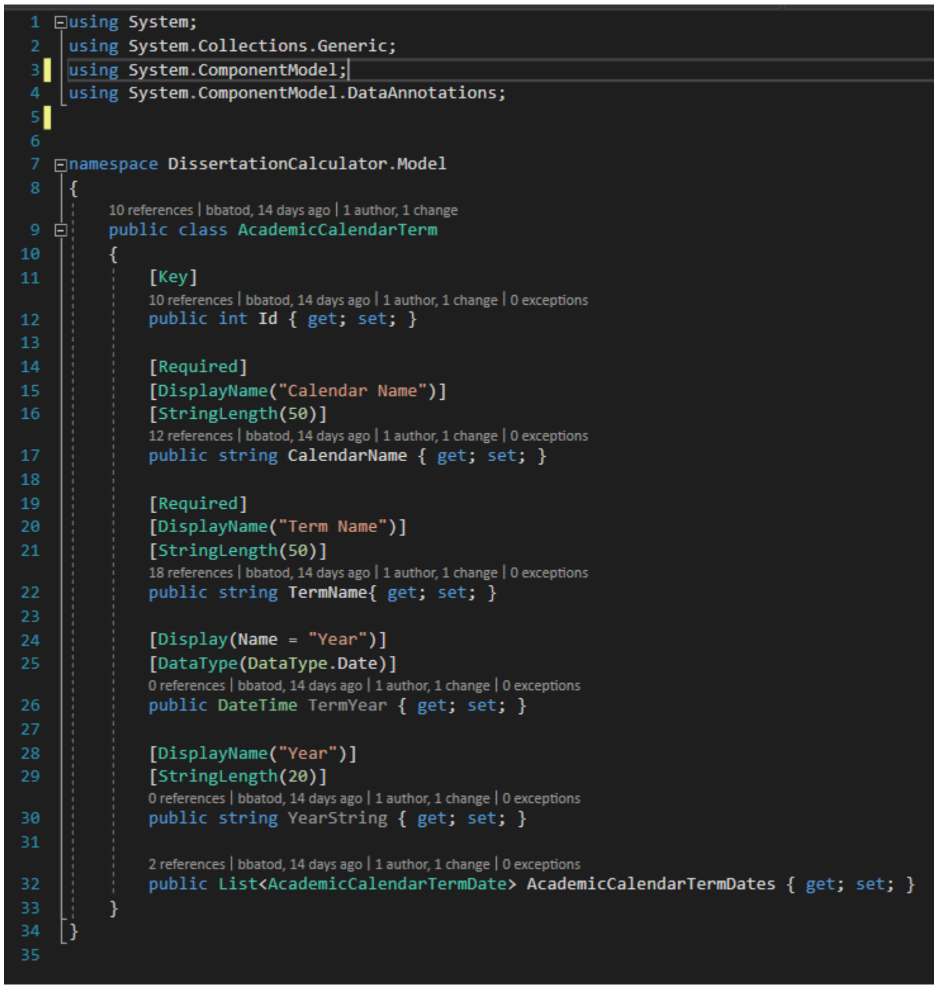
[13] Roth, D., Anderson, R., & Luttin, S. (2019, October 9). Introduction to ASP.NET Core. Retrieved October 31, 2019, from https://docs.microsoft.com/en-us/aspnet/core/index?view=aspnetcore-2.2.

[14] RowReorder. (n.d.). Retrieved from https://datatables.net/extensions/rowreorder/.

[15] Shirhatti, S. (2019, March 30). Host ASP.NET Core on Linux with Nginx. Retrieved October 31, 2019, from https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/linux-nginx?view=aspnetcore-2.2.

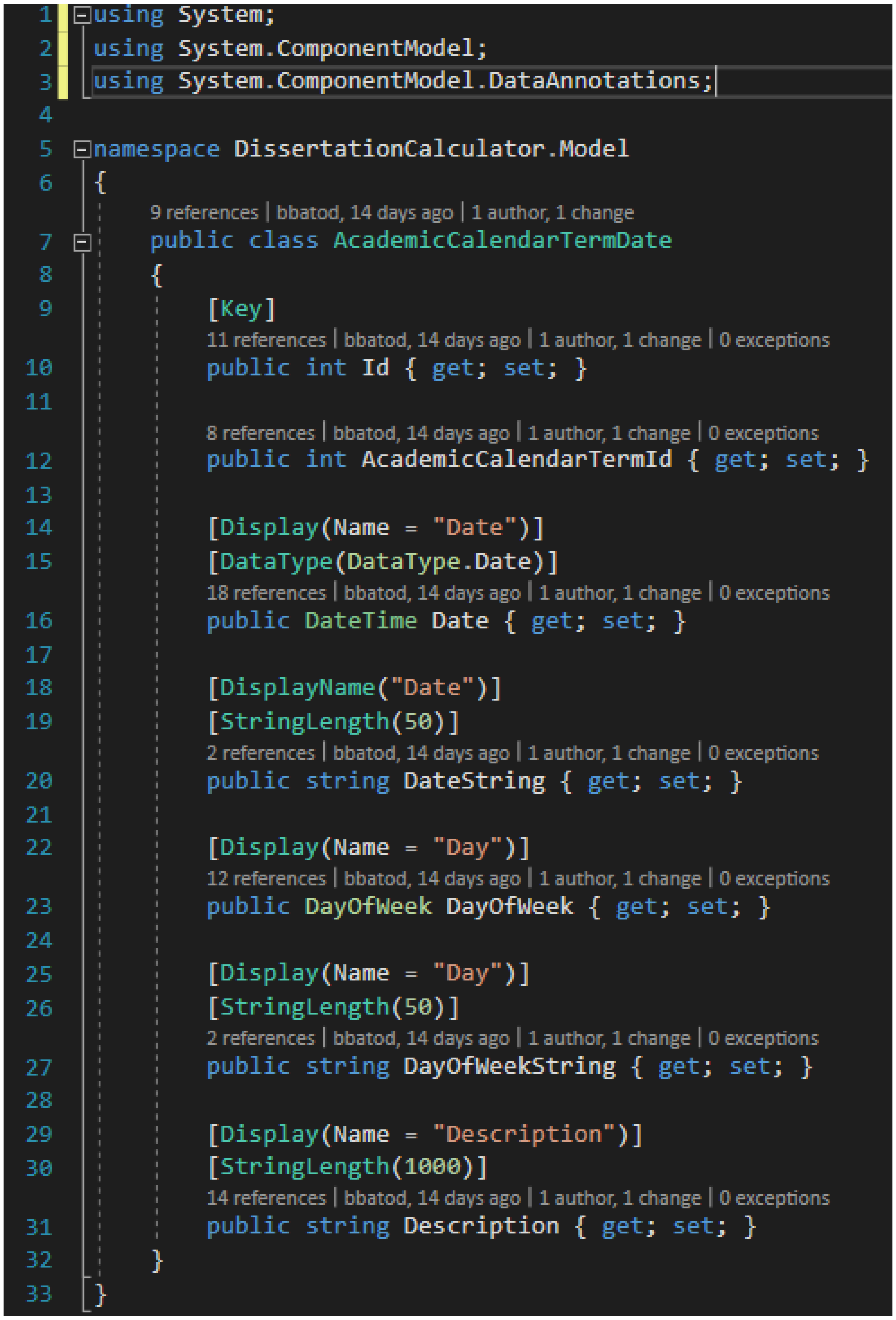
[16] UTSC Homepage. (n.d.). Retrieved from https://ctl.utsc.utoronto.ca/dissertationcal/.

APPENDIX



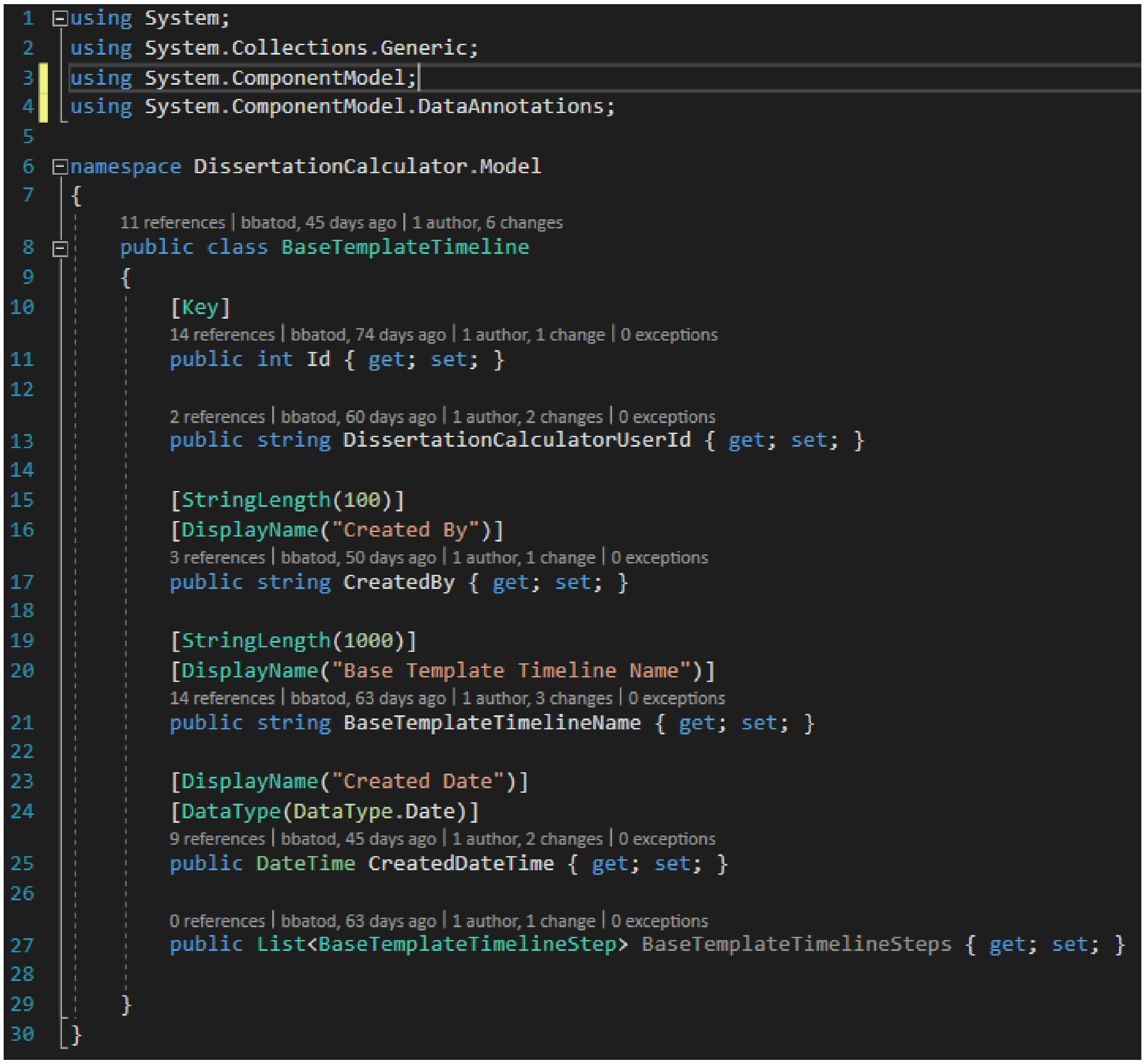
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;


namespace DissertationCalculator.Model
{
    10 references | bbatod, 14 days ago | 1 author, 1 change
    public class AcademicCalendarTerm
    {
        [Key]
        10 references | bbatod, 14 days ago | 1 author, 1 change | 0 exceptions
        public int Id { get; set; }

        [Required]
        [DisplayName("Calendar Name")]
        [StringLength(50)]
        12 references | bbatod, 14 days ago | 1 author, 1 change | 0 exceptions
        public string CalendarName { get; set; }

        [Required]
        [DisplayName("Term Name")]
        [StringLength(50)]
        18 references | bbatod, 14 days ago | 1 author, 1 change | 0 exceptions
        public string TermName{ get; set; }

        [Display(Name = "Year")]
        [DataType(DataType.Date)]
        0 references | bbatod, 14 days ago | 1 author, 1 change | 0 exceptions
        public DateTime TermYear { get; set; }

        [DisplayName("Year")]
        [StringLength(20)]
        0 references | bbatod, 14 days ago | 1 author, 1 change | 0 exceptions
        public string YearString { get; set; }

        2 references | bbatod, 14 days ago | 1 author, 1 change | 0 exceptions
        public List<AcademicCalendarTermDate> AcademicCalendarTermDates { get; set; }
    }
}

```

Figure A.1. Academic CalendarTerm.cs



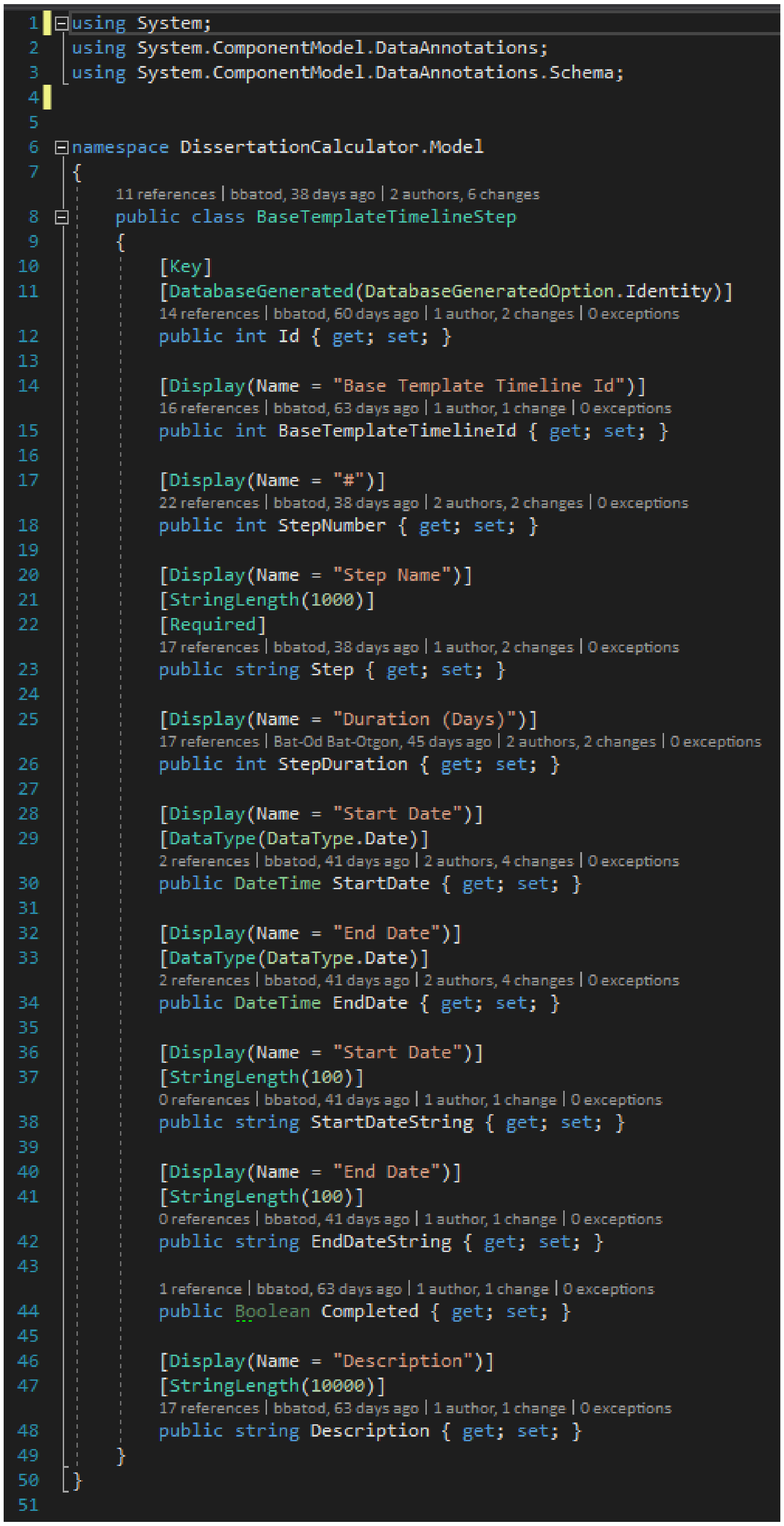
```
using System;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace DissertationCalculator.Model
{
        9 references | bbatod, 14 days ago | 1 author, 1 change
    public class AcademicCalendarTermDate
    {
        [Key]
        11 references | bbatod, 14 days ago | 1 author, 1 change | 0 exceptions
        public int Id { get; set; }

        8 references | bbatod, 14 days ago | 1 author, 1 change | 0 exceptions
        public int AcademicCalendarTermId { get; set; }

        [Display(Name = "Date")]
        [DataType(DataType.Date)]
        18 references | bbatod, 14 days ago | 1 author, 1 change | 0 exceptions
        public DateTime Date { get; set; }

        [DisplayName("Date")]
        [StringLength(50)]
        2 references | bbatod, 14 days ago | 1 author, 1 change | 0 exceptions
        public string DateString { get; set; }

        [Display(Name = "Day")]
        12 references | bbatod, 14 days ago | 1 author, 1 change | 0 exceptions
        public DayOfWeek DayOfWeek { get; set; }

        [Display(Name = "Day")]
        [StringLength(50)]
        2 references | bbatod, 14 days ago | 1 author, 1 change | 0 exceptions
        public string DayOfWeekString { get; set; }

        [Display(Name = "Description")]
        [StringLength(1000)]
        14 references | bbatod, 14 days ago | 1 author, 1 change | 0 exceptions
        public string Description { get; set; }
    }
}
```

Figure A.2. AcademicCalendarTermDate.cs



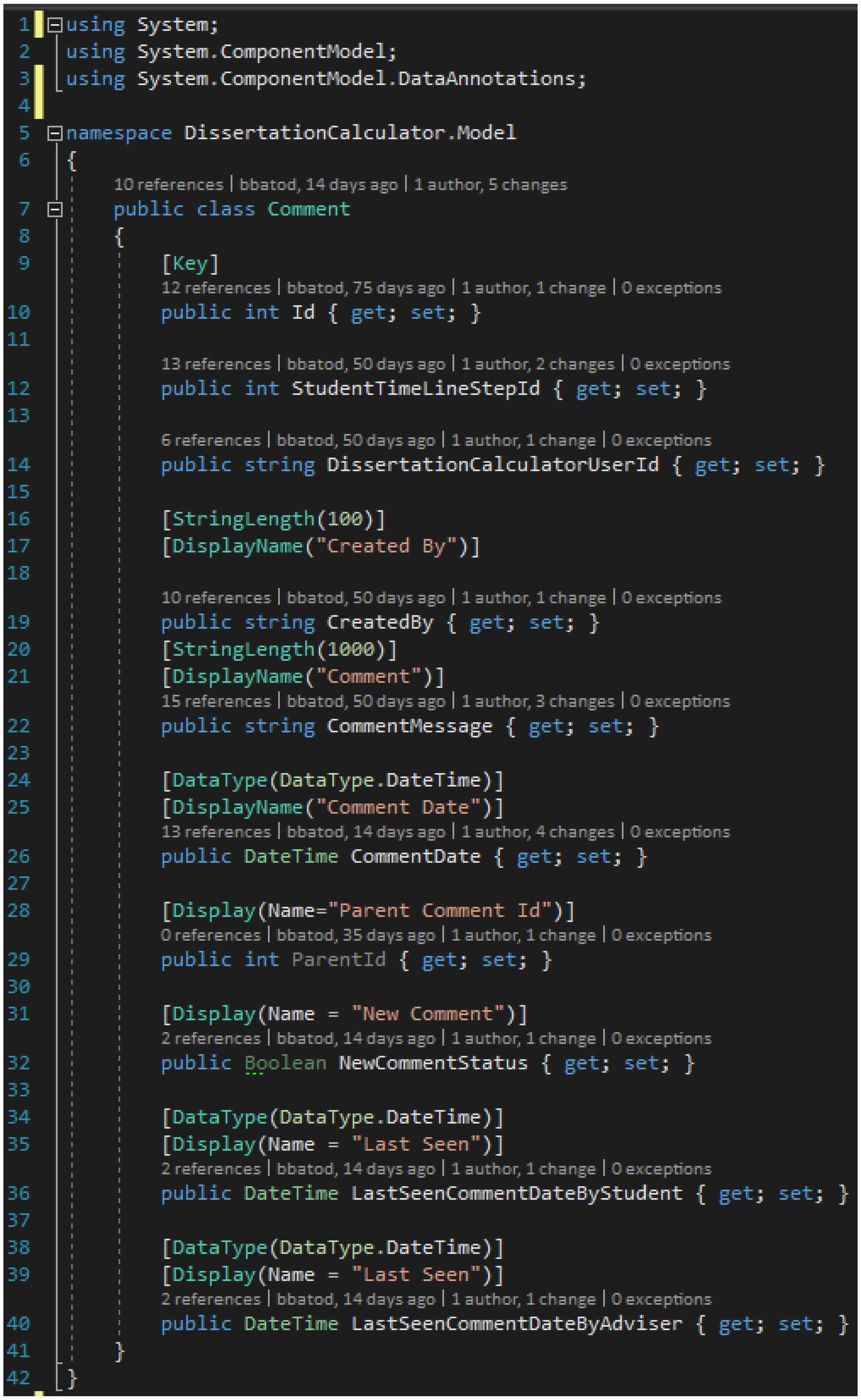
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace DissertationCalculator.Model
{
    11 references | bbatod, 45 days ago | 1 author, 6 changes
    public class BaseTemplateTimeline
    {
        [Key]
        14 references | bbatod, 74 days ago | 1 author, 1 change | 0 exceptions
        public int Id { get; set; }

        2 references | bbatod, 60 days ago | 1 author, 2 changes | 0 exceptions
        public string DissertationCalculatorUserId { get; set; }

        [StringLength(100)]
        [DisplayName("Created By")]
        3 references | bbatod, 50 days ago | 1 author, 1 change | 0 exceptions
        public string CreatedBy { get; set; }

        [StringLength(1000)]
        [DisplayName("Base Template Timeline Name")]
        14 references | bbatod, 63 days ago | 1 author, 3 changes | 0 exceptions
        public string BaseTemplateTimelineName { get; set; }

        [DisplayName("Created Date")]
        [DataType(DataType.Date)]
        9 references | bbatod, 45 days ago | 1 author, 2 changes | 0 exceptions
        public DateTime CreatedDateTime { get; set; }

        0 references | bbatod, 63 days ago | 1 author, 1 change | 0 exceptions
        public List<BaseTemplateTimelineStep> BaseTemplateTimelineSteps { get; set; }

    }
}
```

Figure A.3. BaseTemplateTimeline.cs



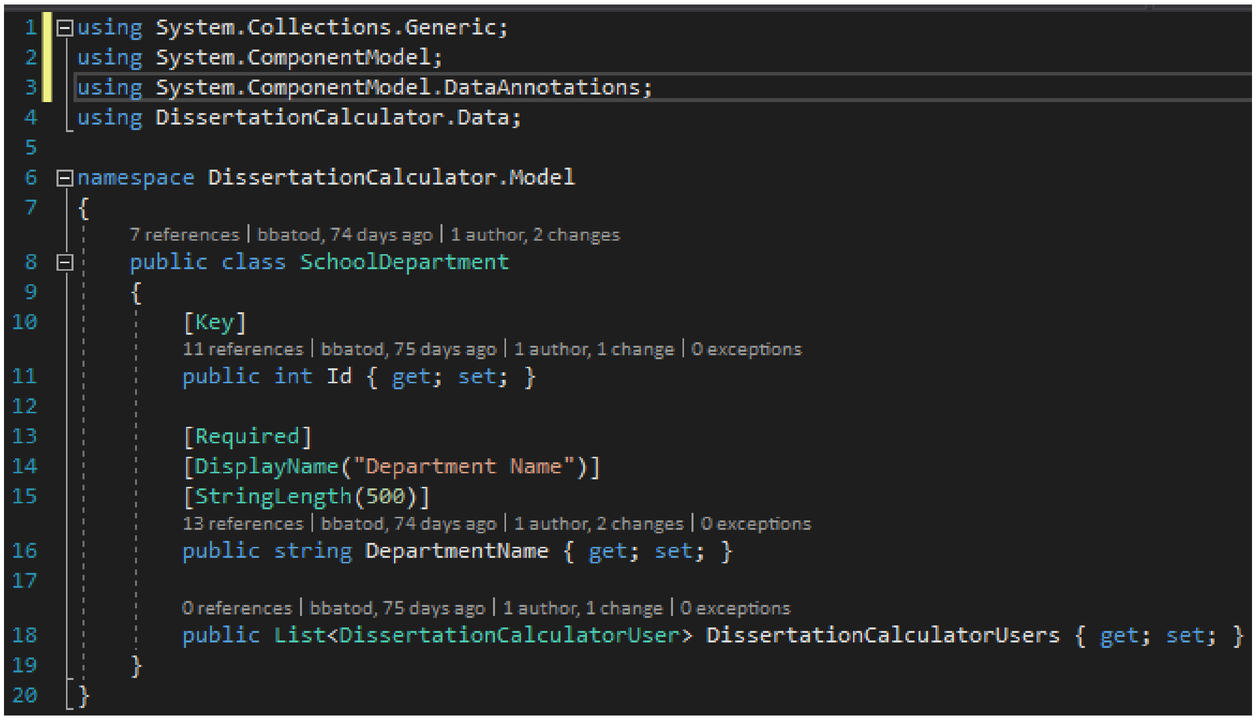
```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;


namespace DissertationCalculator.Model
{
    11 references | bbatod, 38 days ago | 2 authors, 6 changes
    public class BaseTemplateTimelineStep
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        14 references | bbatod, 60 days ago | 1 author, 2 changes | 0 exceptions
        public int Id { get; set; }

        [Display(Name = "Base Template Timeline Id")]
        16 references | bbatod, 63 days ago | 1 author, 1 change | 0 exceptions
        public int BaseTemplateTimelineId { get; set; }

        [Display(Name = "#")]
        22 references | bbatod, 38 days ago | 2 authors, 2 changes | 0 exceptions
        public int StepNumber { get; set; }

        [Display(Name = "Step Name")]
        [StringLength(1000)]
        [Required]
        17 references | bbatod, 38 days ago | 1 author, 2 changes | 0 exceptions
        public string Step { get; set; }

        [Display(Name = "Duration (Days)")]
        17 references | Bat-Od Bat-Otgon, 45 days ago | 2 authors, 2 changes | 0 exceptions
        public int StepDuration { get; set; }

        [Display(Name = "Start Date")]
        [DataType(DataType.Date)]
        2 references | bbatod, 41 days ago | 2 authors, 4 changes | 0 exceptions
        public DateTime StartDate { get; set; }

        [Display(Name = "End Date")]
        [DataType(DataType.Date)]
        2 references | bbatod, 41 days ago | 2 authors, 4 changes | 0 exceptions
        public DateTime EndDate { get; set; }

        [Display(Name = "Start Date")]
        [StringLength(100)]
        0 references | bbatod, 41 days ago | 1 author, 1 change | 0 exceptions
        public string StartDateString { get; set; }

        [Display(Name = "End Date")]
        [StringLength(100)]
        0 references | bbatod, 41 days ago | 1 author, 1 change | 0 exceptions
        public string EndDateString { get; set; }

        1 reference | bbatod, 63 days ago | 1 author, 1 change | 0 exceptions
        public Boolean Completed { get; set; }

        [Display(Name = "Description")]
        [StringLength(10000)]
        17 references | bbatod, 63 days ago | 1 author, 1 change | 0 exceptions
        public string Description { get; set; }
    }
}
```

Figure A.4. BaseTemplateTimelineSteps.cs



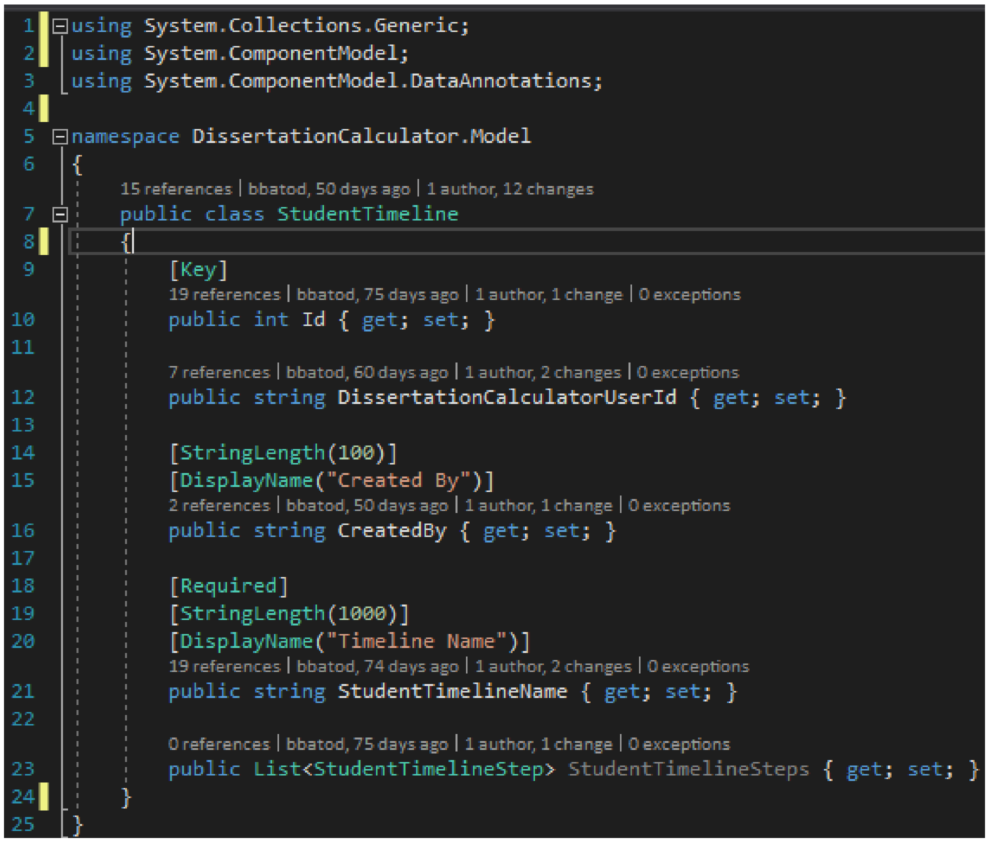
```
using System;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace DissertationCalculator.Model
{
    10 references | bbatod, 14 days ago | 1 author, 5 changes
    public class Comment
    {
        [Key]
        12 references | bbatod, 75 days ago | 1 author, 1 change | 0 exceptions
        public int Id { get; set; }

        13 references | bbatod, 50 days ago | 1 author, 2 changes | 0 exceptions
        public int StudentTimeLineStepId { get; set; }

        6 references | bbatod, 50 days ago | 1 author, 1 change | 0 exceptions
        public string DissertationCalculatorUserId { get; set; }

        [StringLength(100)]
        [DisplayName("Created By")]

        10 references | bbatod, 50 days ago | 1 author, 1 change | 0 exceptions
        public string CreatedBy { get; set; }
        [StringLength(1000)]
        [DisplayName("Comment")]
        15 references | bbatod, 50 days ago | 1 author, 3 changes | 0 exceptions
        public string CommentMessage { get; set; }

        [DataType(DataType.DateTime)]
        [DisplayName("Comment Date")]
        13 references | bbatod, 14 days ago | 1 author, 4 changes | 0 exceptions
        public DateTime CommentDate { get; set; }

        [Display(Name="Parent Comment Id")]
        0 references | bbatod, 35 days ago | 1 author, 1 change | 0 exceptions
        public int ParentId { get; set; }

        [Display(Name = "New Comment")]
        2 references | bbatod, 14 days ago | 1 author, 1 change | 0 exceptions
        public Boolean NewCommentStatus { get; set; }

        [DataType(DataType.DateTime)]
        [Display(Name = "Last Seen")]
        2 references | bbatod, 14 days ago | 1 author, 1 change | 0 exceptions
        public DateTime LastSeenCommentDateByStudent { get; set; }

        [DataType(DataType.DateTime)]
        [Display(Name = "Last Seen")]
        2 references | bbatod, 14 days ago | 1 author, 1 change | 0 exceptions
        public DateTime LastSeenCommentDateByAdviser { get; set; }
    }
}
```

Figure A.5. Comment.cs



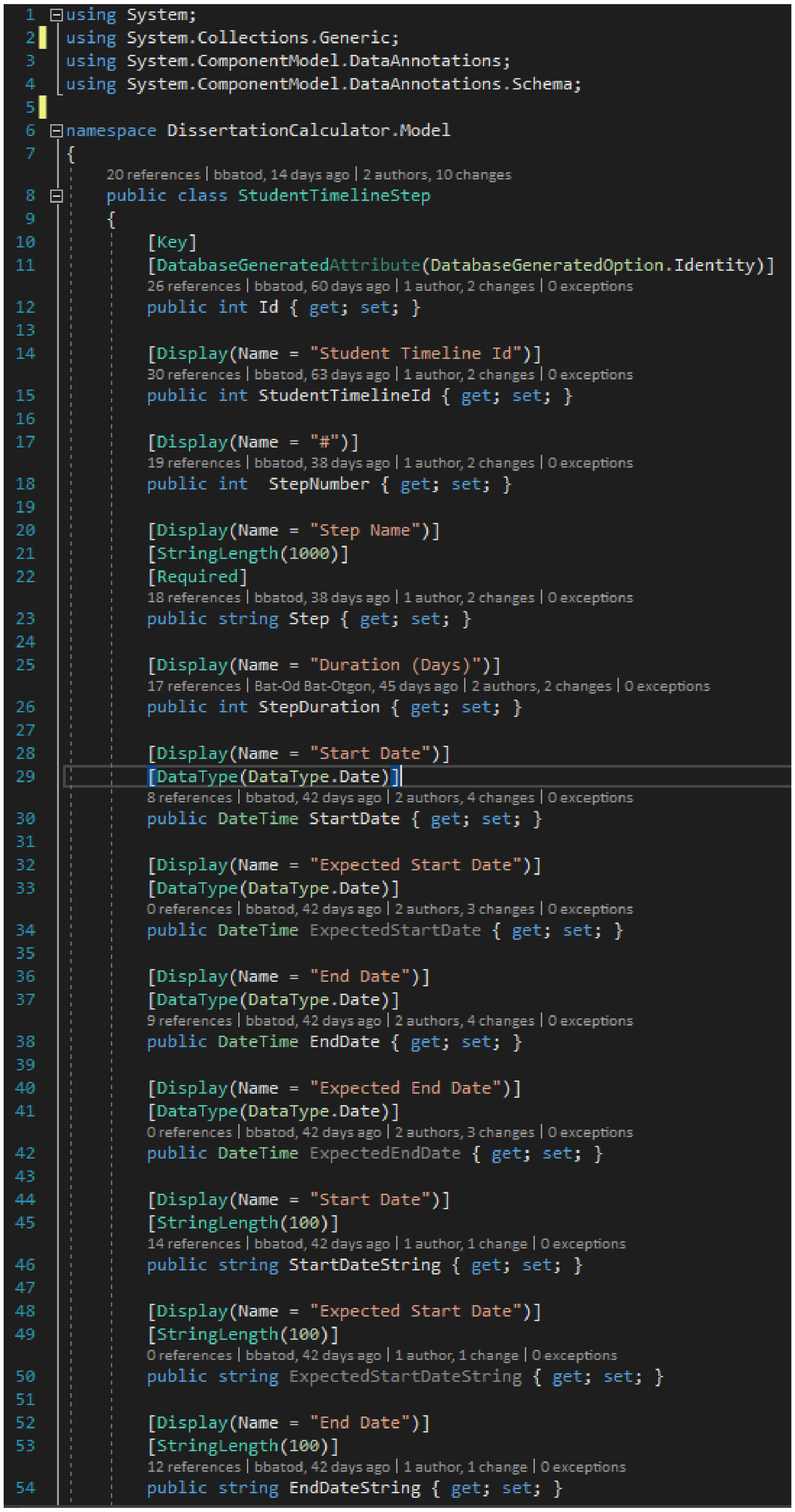
```
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using DissertationCalculator.Data;

namespace DissertationCalculator.Model
{
    7 references | bbatod, 74 days ago | 1 author, 2 changes
    public class SchoolDepartment
    {
        [Key]
        11 references | bbatod, 75 days ago | 1 author, 1 change | 0 exceptions
        public int Id { get; set; }

        [Required]
        [DisplayName("Department Name")]
        [StringLength(500)]
        13 references | bbatod, 74 days ago | 1 author, 2 changes | 0 exceptions
        public string DepartmentName { get; set; }

        0 references | bbatod, 75 days ago | 1 author, 1 change | 0 exceptions
        public List<DissertationCalculatorUser> DissertationCalculatorUsers { get; set; }
    }
}
```

Figure A.6. SchoolDepartment.cs



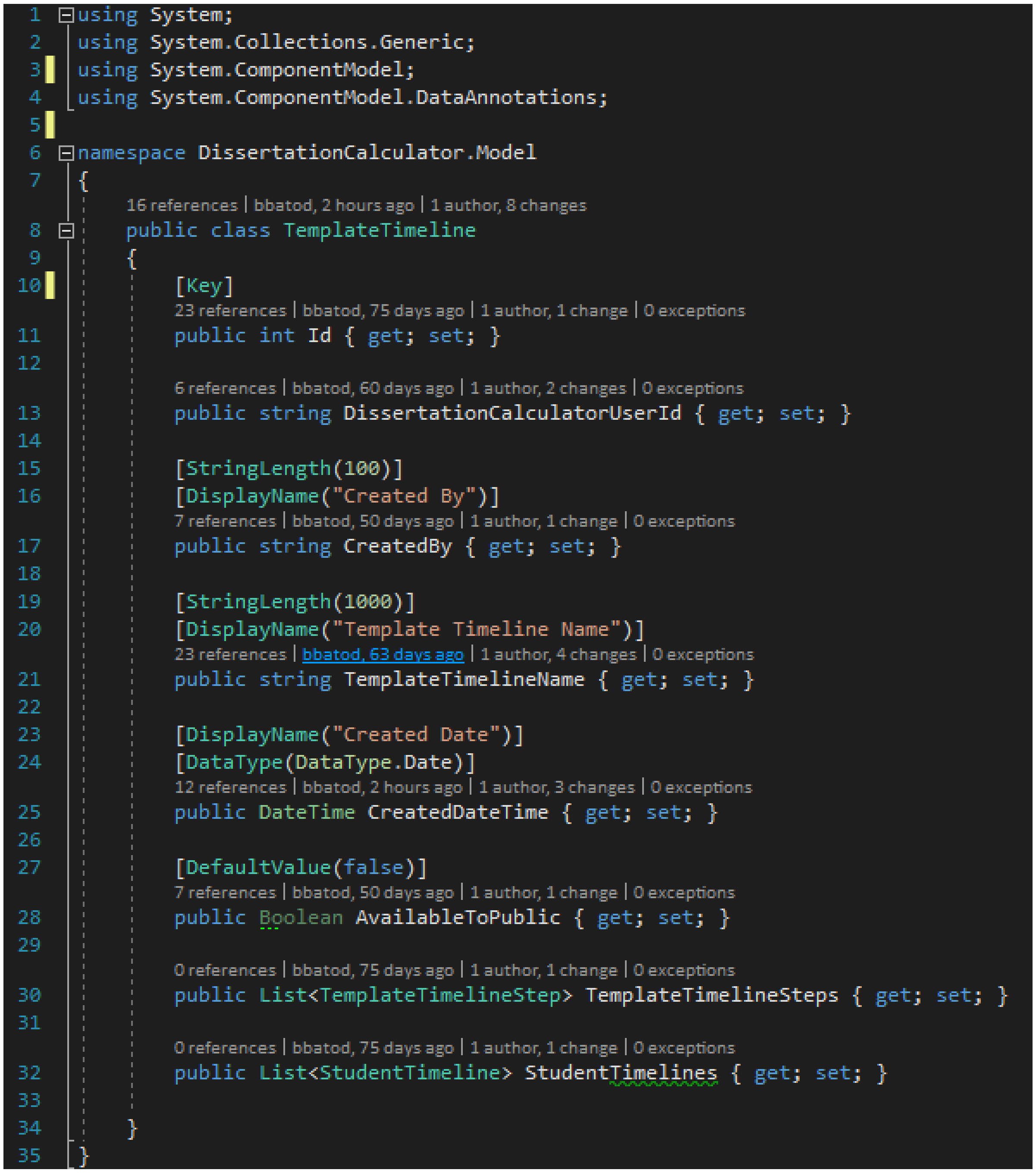
```
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

namespace DissertationCalculator.Model
{
    15 references | bbatod, 50 days ago | 1 author, 12 changes
    public class StudentTimeline
    {
        [Key]
        19 references | bbatod, 75 days ago | 1 author, 1 change | 0 exceptions
        public int Id { get; set; }

        7 references | bbatod, 60 days ago | 1 author, 2 changes | 0 exceptions
        public string DissertationCalculatorUserId { get; set; }

        [StringLength(100)]
        [DisplayName("Created By")]
        2 references | bbatod, 50 days ago | 1 author, 1 change | 0 exceptions
        public string CreatedBy { get; set; }

        [Required]
        [StringLength(1000)]
        [DisplayName("Timeline Name")]
        19 references | bbatod, 74 days ago | 1 author, 2 changes | 0 exceptions
        public string StudentTimelineName { get; set; }

        0 references | bbatod, 75 days ago | 1 author, 1 change | 0 exceptions
        public List<StudentTimelineStep> StudentTimelineSteps { get; set; }
    }
}
```

Figure A.7. StudentTimeline.cs



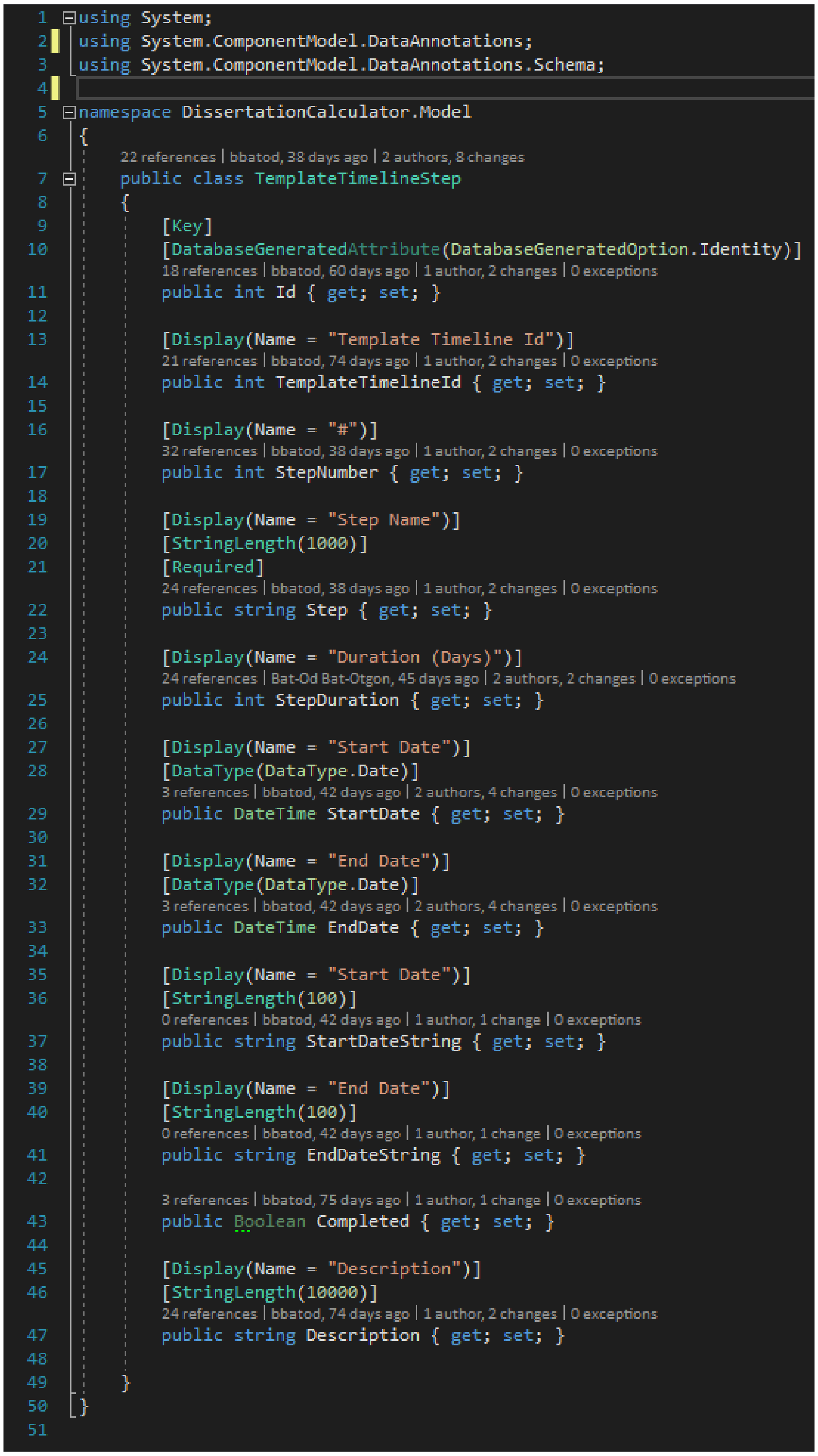
```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace DissertationCalculator.Model
{
    20 references | bbatod, 14 days ago | 2 authors, 10 changes
    public class StudentTimelineStep
    {
        [Key]
        [DatabaseGeneratedAttribute(DatabaseGeneratedOption.Identity)]
        26 references | bbatod, 60 days ago | 1 author, 2 changes | 0 exceptions
        public int Id { get; set; }

        [Display(Name = "Student Timeline Id")]
        30 references | bbatod, 63 days ago | 1 author, 2 changes | 0 exceptions
        public int StudentTimelineId { get; set; }

        [Display(Name = "#")]
        19 references | bbatod, 38 days ago | 1 author, 2 changes | 0 exceptions
        public int  StepNumber { get; set; }

        [Display(Name = "Step Name")]
        [StringLength(1000)]
        [Required]
        18 references | bbatod, 38 days ago | 1 author, 2 changes | 0 exceptions
        public string Step { get; set; }

        [Display(Name = "Duration (Days)")]
        17 references | Bat-Od Bat-Otgon, 45 days ago | 2 authors, 2 changes | 0 exceptions
        public int StepDuration { get; set; }

        [Display(Name = "Start Date")]
        [DataType(DataType.Date)]
        8 references | bbatod, 42 days ago | 2 authors, 4 changes | 0 exceptions
        public DateTime StartDate { get; set; }

        [Display(Name = "Expected Start Date")]
        [DataType(DataType.Date)]
        0 references | bbatod, 42 days ago | 2 authors, 3 changes | 0 exceptions
        public DateTime ExpectedStartDate { get; set; }

        [Display(Name = "End Date")]
        [DataType(DataType.Date)]
        9 references | bbatod, 42 days ago | 2 authors, 4 changes | 0 exceptions
        public DateTime EndDate { get; set; }

        [Display(Name = "Expected End Date")]
        [DataType(DataType.Date)]
        0 references | bbatod, 42 days ago | 2 authors, 3 changes | 0 exceptions
        public DateTime ExpectedEndDate { get; set; }

        [Display(Name = "Start Date")]
        [StringLength(100)]
        14 references | bbatod, 42 days ago | 1 author, 1 change | 0 exceptions
        public string StartDateString { get; set; }

        [Display(Name = "Expected Start Date")]
        [StringLength(100)]
        0 references | bbatod, 42 days ago | 1 author, 1 change | 0 exceptions
        public string ExpectedStartDateString { get; set; }

        [Display(Name = "End Date")]
        [StringLength(100)]
        12 references | bbatod, 42 days ago | 1 author, 1 change | 0 exceptions
        public string EndDateString { get; set; }
```

Figure A.8. StudentTimelineStep.cs



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

namespace DissertationCalculator.Model
{
    16 references | bbatod, 2 hours ago | 1 author, 8 changes
    public class TemplateTimeline
    {
        [Key]
        23 references | bbatod, 75 days ago | 1 author, 1 change | 0 exceptions
        public int Id { get; set; }

        6 references | bbatod, 60 days ago | 1 author, 2 changes | 0 exceptions
        public string DissertationCalculatorUserId { get; set; }

        [StringLength(100)]
        [DisplayName("Created By")]
        7 references | bbatod, 50 days ago | 1 author, 1 change | 0 exceptions
        public string CreatedBy { get; set; }

        [StringLength(1000)]
        [DisplayName("Template Timeline Name")]
        23 references | bbatod, 63 days ago | 1 author, 4 changes | 0 exceptions
        public string TemplateTimelineName { get; set; }

        [DisplayName("Created Date")]
        [DataType(DataType.Date)]
        12 references | bbatod, 2 hours ago | 1 author, 3 changes | 0 exceptions
        public DateTime CreatedDateTime { get; set; }

        [DefaultValue(false)]
        7 references | bbatod, 50 days ago | 1 author, 1 change | 0 exceptions
        public Boolean AvailableToPublic { get; set; }

        0 references | bbatod, 75 days ago | 1 author, 1 change | 0 exceptions
        public List<TemplateTimelineStep> TemplateTimelineSteps { get; set; }

        0 references | bbatod, 75 days ago | 1 author, 1 change | 0 exceptions
        public List<StudentTimeline> StudentTimelines { get; set; }

    }
}
```

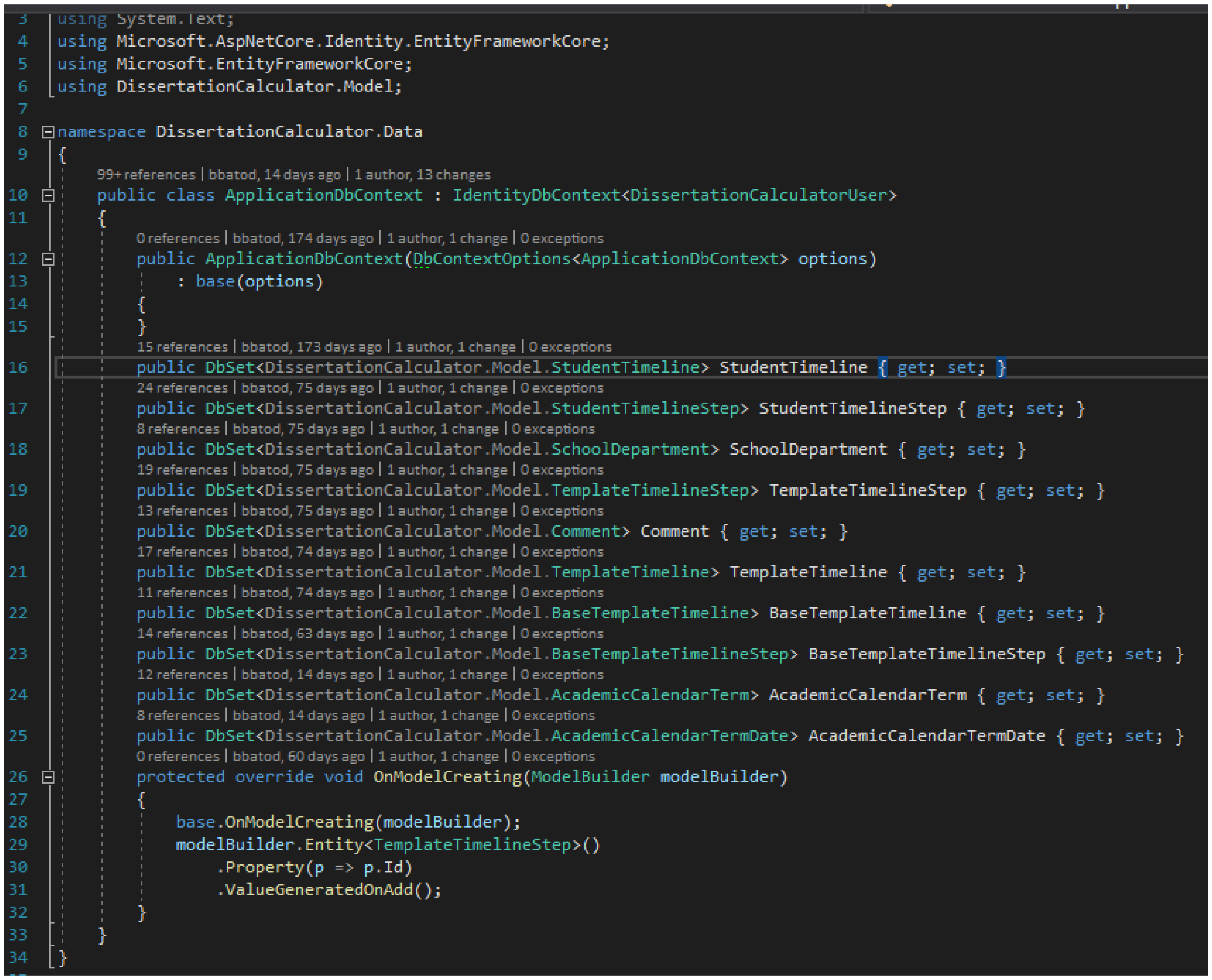
Figure A.9. TemplateTimeline.cs



```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace DissertationCalculator.Model
{
    22 references | bbatod, 38 days ago | 2 authors, 8 changes
    public class TemplateTimelineStep
    {
        [Key]
        [DatabaseGeneratedAttribute(DatabaseGeneratedOption.Identity)]
        18 references | bbatod, 60 days ago | 1 author, 2 changes | 0 exceptions
        public int Id { get; set; }

        [Display(Name = "Template Timeline Id")]
        21 references | bbatod, 74 days ago | 1 author, 2 changes | 0 exceptions
        public int TemplateTimelineId { get; set; }

        [Display(Name = "#")]
        32 references | bbatod, 38 days ago | 1 author, 2 changes | 0 exceptions
        public int StepNumber { get; set; }

        [Display(Name = "Step Name")]
        [StringLength(1000)]
        [Required]
        24 references | bbatod, 38 days ago | 1 author, 2 changes | 0 exceptions
        public string Step { get; set; }

        [Display(Name = "Duration (Days)")]
        24 references | Bat-Od Bat-Otgon, 45 days ago | 2 authors, 2 changes | 0 exceptions
        public int StepDuration { get; set; }

        [Display(Name = "Start Date")]
        [DataType(DataType.Date)]
        3 references | bbatod, 42 days ago | 2 authors, 4 changes | 0 exceptions
        public DateTime StartDate { get; set; }

        [Display(Name = "End Date")]
        [DataType(DataType.Date)]
        3 references | bbatod, 42 days ago | 2 authors, 4 changes | 0 exceptions
        public DateTime EndDate { get; set; }

        [Display(Name = "Start Date")]
        [StringLength(100)]
        0 references | bbatod, 42 days ago | 1 author, 1 change | 0 exceptions
        public string StartDateString { get; set; }

        [Display(Name = "End Date")]
        [StringLength(100)]
        0 references | bbatod, 42 days ago | 1 author, 1 change | 0 exceptions
        public string EndDateString { get; set; }

        3 references | bbatod, 75 days ago | 1 author, 1 change | 0 exceptions
        public Boolean Completed { get; set; }

        [Display(Name = "Description")]
        [StringLength(10000)]
        24 references | bbatod, 74 days ago | 1 author, 2 changes | 0 exceptions
        public string Description { get; set; }

    }
}

```

Figure A.10. TemplateTimelineStep.cs

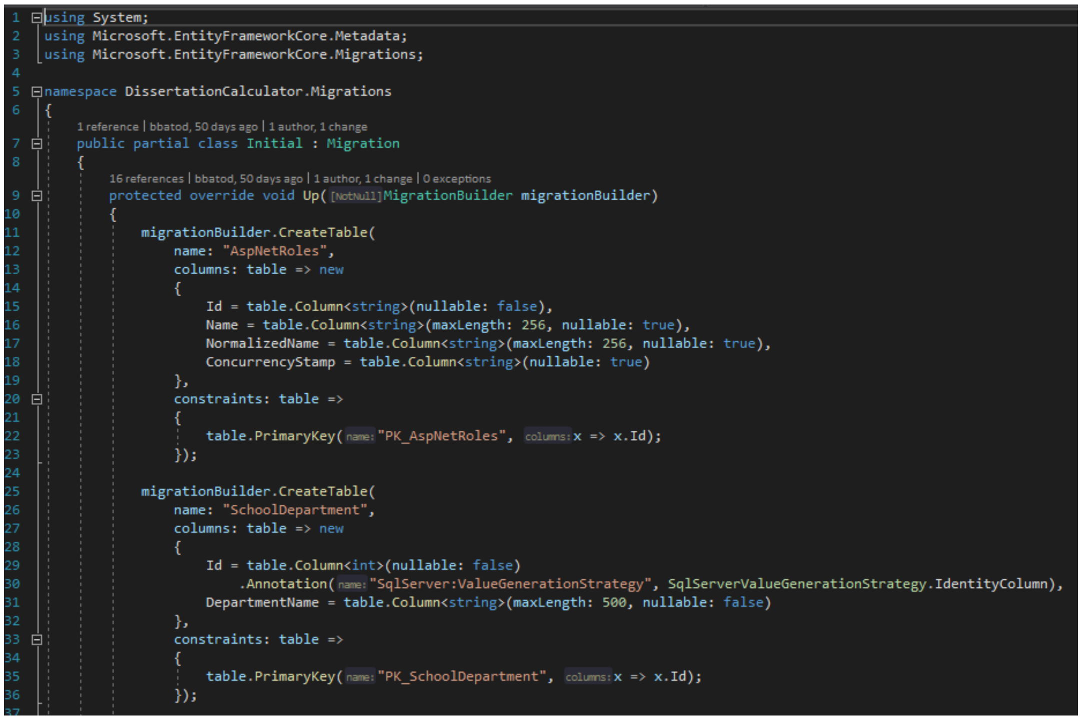
The models are initialized in a database context like below and the Entity Framework Core takes care of the database creation using Fluent Modeling with a process called migration.



```
using System.Text;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using DissertationCalculator.Model;

namespace DissertationCalculator.Data
{
    99+ references | bbatod, 14 days ago | 1 author, 13 changes
    public class ApplicationDbContext : IdentityDbContext<DissertationCalculatorUser>
    {
        0 references | bbatod, 174 days ago | 1 author, 1 change | 0 exceptions
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
            : base(options)
        {
        }
        15 references | bbatod, 173 days ago | 1 author, 1 change | 0 exceptions
        public DbSet<DissertationCalculator.Model.StudentTimeline> StudentTimeline { get; set; }
        24 references | bbatod, 75 days ago | 1 author, 1 change | 0 exceptions
        public DbSet<DissertationCalculator.Model.StudentTimelineStep> StudentTimelineStep { get; set; }
        8 references | bbatod, 75 days ago | 1 author, 1 change | 0 exceptions
        public DbSet<DissertationCalculator.Model.SchoolDepartment> SchoolDepartment { get; set; }
        19 references | bbatod, 75 days ago | 1 author, 1 change | 0 exceptions
        public DbSet<DissertationCalculator.Model.TemplateTimelineStep> TemplateTimelineStep { get; set; }
        13 references | bbatod, 75 days ago | 1 author, 1 change | 0 exceptions
        public DbSet<DissertationCalculator.Model.Comment> Comment { get; set; }
        17 references | bbatod, 74 days ago | 1 author, 1 change | 0 exceptions
        public DbSet<DissertationCalculator.Model.TemplateTimeline> TemplateTimeline { get; set; }
        11 references | bbatod, 74 days ago | 1 author, 1 change | 0 exceptions
        public DbSet<DissertationCalculator.Model.BaseTemplateTimeline> BaseTemplateTimeline { get; set; }
        14 references | bbatod, 63 days ago | 1 author, 1 change | 0 exceptions
        public DbSet<DissertationCalculator.Model.BaseTemplateTimelineStep> BaseTemplateTimelineStep { get; set; }
        12 references | bbatod, 14 days ago | 1 author, 1 change | 0 exceptions
        public DbSet<DissertationCalculator.Model.AcademicCalendarTerm> AcademicCalendarTerm { get; set; }
        8 references | bbatod, 14 days ago | 1 author, 1 change | 0 exceptions
        public DbSet<DissertationCalculator.Model.AcademicCalendarTermDate> AcademicCalendarTermDate { get; set; }
        0 references | bbatod, 60 days ago | 1 author, 1 change | 0 exceptions
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);
            modelBuilder.Entity<TemplateTimelineStep>()
                .Property(p => p.Id)
                .ValueGeneratedOnAdd();
        }
    }
}
```

Figure A.11. ApplicationDbContext.cs

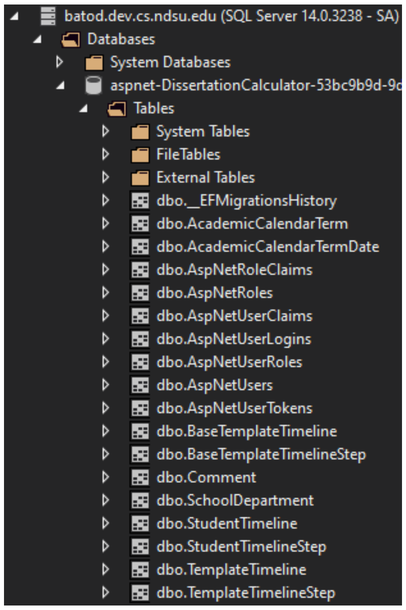
Migration Code example.



```
using System;
using Microsoft.EntityFrameworkCore.Metadata;
using Microsoft.EntityFrameworkCore.Migrations;

namespace DissertationCalculator.Migrations
{
    1 reference | bbatod, 50 days ago | 1 author, 1 change
    public partial class Initial : Migration
    {
        16 references | bbatod, 50 days ago | 1 author, 1 change | 0 exceptions
        protected override void Up([NotNull]MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "AspNetRoles",
                columns: table => new
                {
                    Id = table.Column<string>(nullable: false),
                    Name = table.Column<string>(maxLength: 256, nullable: true),
                    NormalizedName = table.Column<string>(maxLength: 256, nullable: true),
                    ConcurrencyStamp = table.Column<string>(nullable: true)
                },
                constraints: table =>
                {
                    table.PrimaryKey(name: "PK_AspNetRoles", columns: x => x.Id);
                });

            migrationBuilder.CreateTable(
                name: "SchoolDepartment",
                columns: table => new
                {
                    Id = table.Column<int>(nullable: false)
                        .Annotation(name: "SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn),
                    DepartmentName = table.Column<string>(maxLength: 500, nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey(name: "PK_SchoolDepartment", columns: x => x.Id);
                });
```

Figure A.12. Migration Code Example

The databases are then created on the SQL server with command Update-Database in Visual Studio which results in the following:



```
batod.dev.cs.ndsu.edu (SQL Server 14.0.3238 - SA)
  Databases
    System Databases
    aspnet-DissertationCalculator-53bc9b9d-9c
      Tables
        System Tables
        FileTables
        External Tables
        dbo.__EFMigrationsHistory
        dbo.AcademicCalendarTerm
        dbo.AcademicCalendarTermDate
        dbo.AspNetRoleClaims
        dbo.AspNetRoles
        dbo.AspNetUserClaims
        dbo.AspNetUserLogins
        dbo.AspNetUserRoles
        dbo.AspNetUsers
        dbo.AspNetUserTokens
        dbo.BaseTemplateTimeline
        dbo.BaseTemplateTimelineStep
        dbo.Comment
        dbo.SchoolDepartment
        dbo.StudentTimeline
        dbo.StudentTimelineStep
        dbo.TemplateTimeline
        dbo.TemplateTimelineStep
```

Figure A.13. Database Tables

Example code generated for SQL query for AcademicCalendarTerm table.

```
CREATE TABLE [dbo].[AcademicCalendarTerm] (

    [Id]           INT           IDENTITY (1, 1) NOT NULL,
    [CalendarName] NVARCHAR (50) NOT NULL,
    [TermName]     NVARCHAR (50) NOT NULL,
    [TermYear]     DATETIME2 (7) NOT NULL,
    [YearString]   NVARCHAR (20) NULL,
    CONSTRAINT [PK_AcademicCalendarTerm] PRIMARY KEY CLUSTERED ([Id] ASC)
);
```

AcademicCalendarTermDate table

```
CREATE TABLE [dbo].[AcademicCalendarTermDate] (
    [Id]                     INT             IDENTITY (1, 1) NOT NULL,
    [AcademicCalendarTermId] INT             NOT NULL,
    [Date]                   DATETIME2 (7)   NOT NULL,
    [DateString]             NVARCHAR (50)   NULL,
    [DayOfWeek]              INT             NOT NULL,
    [DayOfWeekString]        NVARCHAR (50)   NULL,
    [Description]            NVARCHAR (1000) NULL,
    CONSTRAINT [PK_AcademicCalendarTermDate] PRIMARY KEY CLUSTERED ([Id] ASC),
    CONSTRAINT [FK_AcademicCalendarTermDate_AcademicCalendarTerm_AcademicCalendarTermId]
FOREIGN KEY ([AcademicCalendarTermId]) REFERENCES [dbo].[AcademicCalendarTerm] ([Id]) ON
DELETE CASCADE
);


GO
CREATE NONCLUSTERED INDEX [IX_AcademicCalendarTermDate_AcademicCalendarTermId]
    ON [dbo].[AcademicCalendarTermDate]([AcademicCalendarTermId] ASC);
```

As you can see from the SQL query generated by the Entity Framework Core, it takes care of most of the modeling and data access automatically, so it makes it much simpler to develop an application that utilizes data access.