**CS-382 Visual Programming**

**Project Report**

***DEBATRON* – An Online Discussion Forum**



**Submitted To: Engr. Aatka Ali**

**Submitted By:** Laiba Faisal (213002), Khansa (213064)

**Department of Computer Science, Air University Multan Campus**

**TABLE OF CONTENTS**

[**TITLE** 3](#_Toc154569955)

[**OBJECTIVE** 4](#_Toc154569956)

[**DOMAIN MODEL** 5](#_Toc154569957)

[**WIREFRAMES** 6](#_Toc154569958)

[**DATABASE-FIRST APPROACH** 7](#_Toc154569959)

[**PROJECT TEMPLATE** 7](#_Toc154569960)

[**4.1. The ASP.NET MVC** 7](#_Toc154569961)

[**4.2. MVC – The Three-Tier Architecture** 7](#_Toc154569962)

[**RESTFUL API IMPLEMENTATION** 8](#_Toc154569963)

[**TOOLS DEPLOYED** 9](#_Toc154569964)

[**6.1. The Razor View Engine** 9](#_Toc154569965)

[**6.2. The Microsoft Entity Framework** 9](#_Toc154569966)

[**6.3. The CSS Template** 11](#_Toc154569967)

[**6.4. The SSMS** 11](#_Toc154569968)

[**SCREENSHOTS** 12](#_Toc154569969)

[**7.1. The Models** 12](#_Toc154569970)

[**7.2. The Views (Razor Views)** 14](#_Toc154569971)

[**7.3. The Controllers** 14](#_Toc154569972)

[**PROJECT VERSIONING** 15](#_Toc154569973)

[**CONCLUSION** 15](#_Toc154569974)

**TABLE OF FIGURES**

[**Figure 1.1 The Domain Model of Debatron** 5](#_Toc154569713)

[**Figure 2 The Home Page** 6](#_Toc154569714)

[**Figure 3 The SignUp Page** 6](#_Toc154569715)

[**Figure 4 The Three-Tier Architecture** 8](#_Toc154569716)

[**Figure 5 RESTful API** 9](#_Toc154569717)

[**Figure 6 The Entity Framework (EF)** 10](#_Toc154569718)

[**Figure 7 The ADO.NET Entity Data Model** 10](#_Toc154569719)

[**Figure 8 Debatron Data Model using the EF** 11](#_Toc154569720)

[**Figure 9 Debatron Database** 12](#_Toc154569721)

# **TITLE**

***DEBATRON – An Online Discussion Forum***

# **OBJECTIVE**

The primary objective of this project is to provide a dynamic and interactive platform for users to engage in meaningful discussions, share knowledge, and foster a sense of community. This web app aims to facilitate the exchange of ideas, information, and experiences among users with common interests. By leveraging the ASP.NET MVC framework, the project seeks to achieve a robust and scalable architecture, ensuring efficient data management, seamless user navigation, and a responsive user interface. The forum will feature user authentication, allowing participants to create profiles, post topics, reply to threads, and interact with each other in a user-friendly environment. Furthermore, the implementation of moderation tools will help maintain a positive and respectful online community. The overall goal is to create a valuable and engaging space where users can collaborate, learn, and contribute to a vibrant online community.

**Chapter 1**

# **DOMAIN MODEL**

A computer screen shot of a diagram

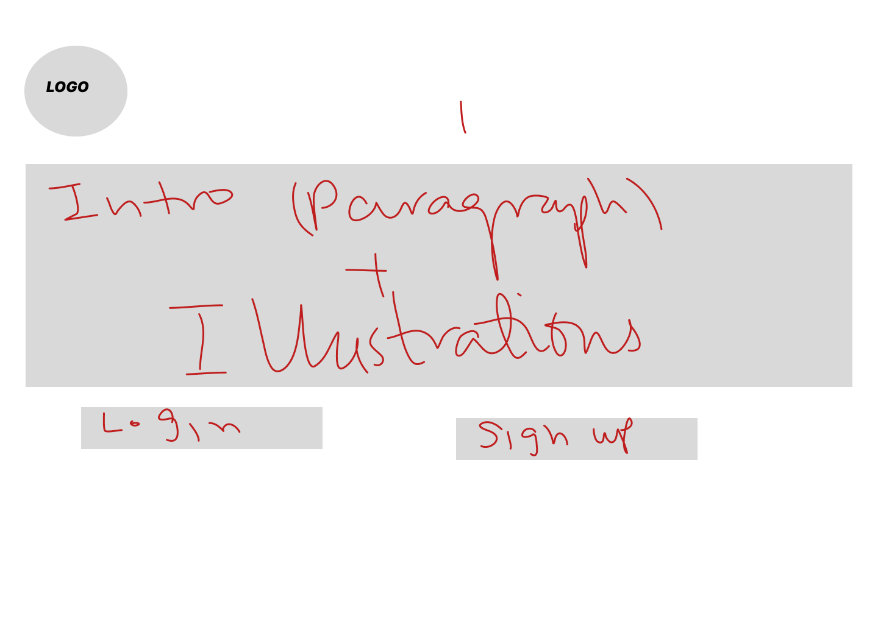
Description automatically generated

**Figure 1.1 The Domain Model of Debatron**

**Chapter 2**

# **WIREFRAMES**

**Homepage**



**Figure 2 The Home Page**

**Signup Page**

A white background with black border

Description automatically generated

**Figure 3 The SignUp Page**

**Chapter 3**

# **DATABASE-FIRST APPROACH**

We have deployed the database-first approach. Under this approach, we created the database in the SQL management studio and then designed the models in our project. The models were designed through coding in csharp in a few versions of the project while in the other versions, the Microsoft Entity Framework had auto generated the models.

An alternate approach is the code-first approach in which we code for the model classes in our project and then create a corresponding database using the Entity Framework's Code First Migrations.

**Chapter 4**

# **PROJECT TEMPLATE**

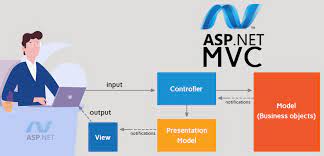
## **4.1. The ASP.NET MVC**

We have made use of the ASP.NET MVC (Model-View-Controller) template for this project. It allows us to have more control over how our website looks and works. It comes with built-in features that make it easier and faster to develop our project.

## **4.2. MVC – The Three-Tier Architecture**

MVC stands for Model-View-Controller, and it is a way to organize and build software applications.

* Model (M): This is where the application’s data and business logic live. It contains a set of classes, each of which corresponds to an entity in the database.
* View (V): The view is what the user sees and interacts with. It's like the face of your application.
* Controller (C): The controller is like the manager between the Model and the View. It takes user input from the View, processes it (with the help of the Model), and decides what the user should see next.

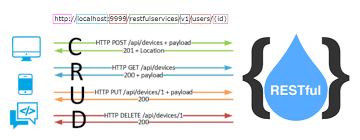


**Figure 4 The Three-Tier Architecture**

**Chapter 5**

# **RESTFUL API IMPLEMENTATION**

We have integrated a RESTful API to facilitate the CRUD operations through HTTP POST and GET methods, seamlessly interacting with the backend database managed by Entity Framework. This REST API adheres to the principles of Representational State Transfer (REST), providing a standardized and stateless communication mechanism. We have utilized the HTTP POST method for creating new records and the HTTP GET method for retrieving data, ensuring a clear and consistent approach to managing resources.



**Figure 5 RESTful API**

**Chapter 6**

# **TOOLS DEPLOYED**

## **6.1. The Razor View Engine**

We chose the Razor view engine due to its simplicity and readability, allowing for efficient HTML integration with C# code, enhancing developer productivity and maintaining clean, easily understandable views. Moreover, the boiler plate code was available for the razor views, making our job easier. All our views have been designed using razor pages and therefore, our views have the extension ***“.cshtml”.***

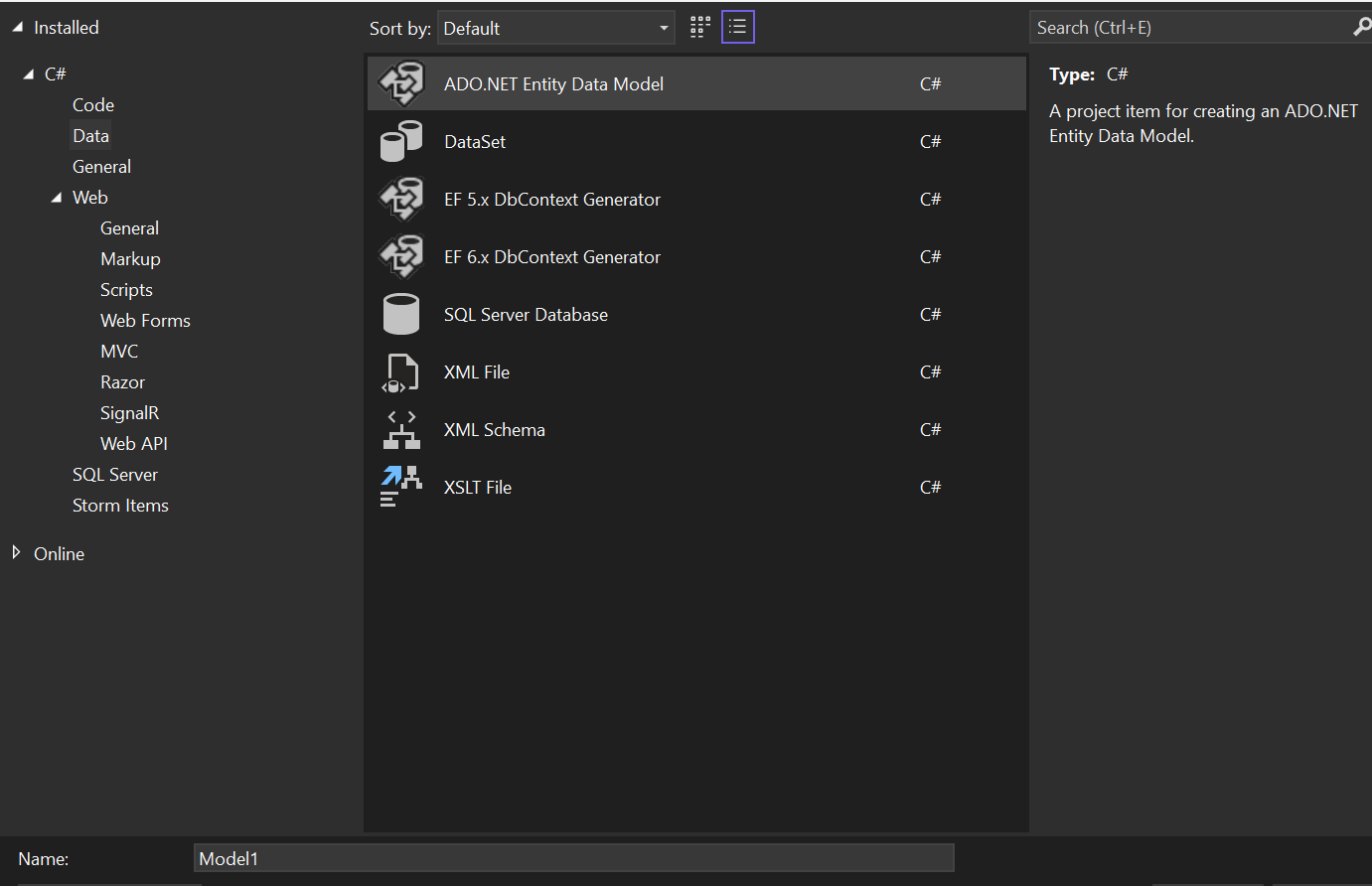
## **6.2. The Microsoft Entity Framework**

We integrated Microsoft Entity Framework (ADO.NET Entity Data Model) into our project for streamlined database interactions, enabling seamless object-relational mapping and reducing the need for complex SQL queries. This decision enhances development efficiency, code maintainability, and database abstraction, ensuring a more robust and scalable application architecture.

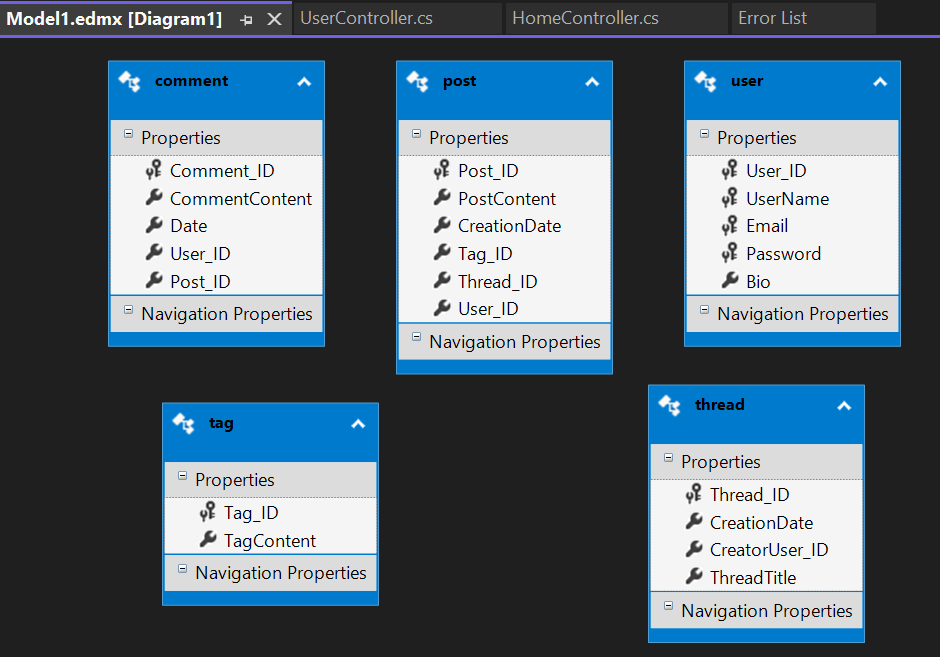
However, we have faced certain challenges owing to version compatibility and thus, have created a version without using the entity framework too which makes use of the connection string configuration for performing the simple CRUD operations.



**Figure 6 The Entity Framework (EF)**



**Figure 7 The ADO.NET Entity Data Model**



**Figure 8 Debatron Data Model using the EF**

## **6.3. The CSS Template**

The ASP.NET MVC comes with a built-in bootsrap.css template that enhances the look and feel of the applications. We have made use of this CSS file to design our website’s layout.

## **6.4. The SSMS**

We have used the SQL Server Management Studio (2014) to create the database to store the application data. The database, named “*Debatron-v1.0*” has been created on the localhost. It has been designed in line with the domain model (chapter 1).

**A screenshot of a computer

Description automatically generated**

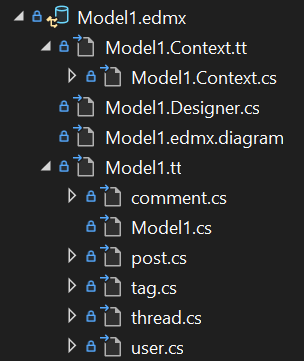
**Figure 9 Debatron Database**

**Chapter 7**

# **SCREENSHOTS**

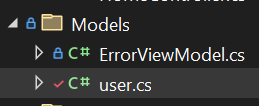
## **7.1. The Models**

* **Designed by the Entity Framework:**



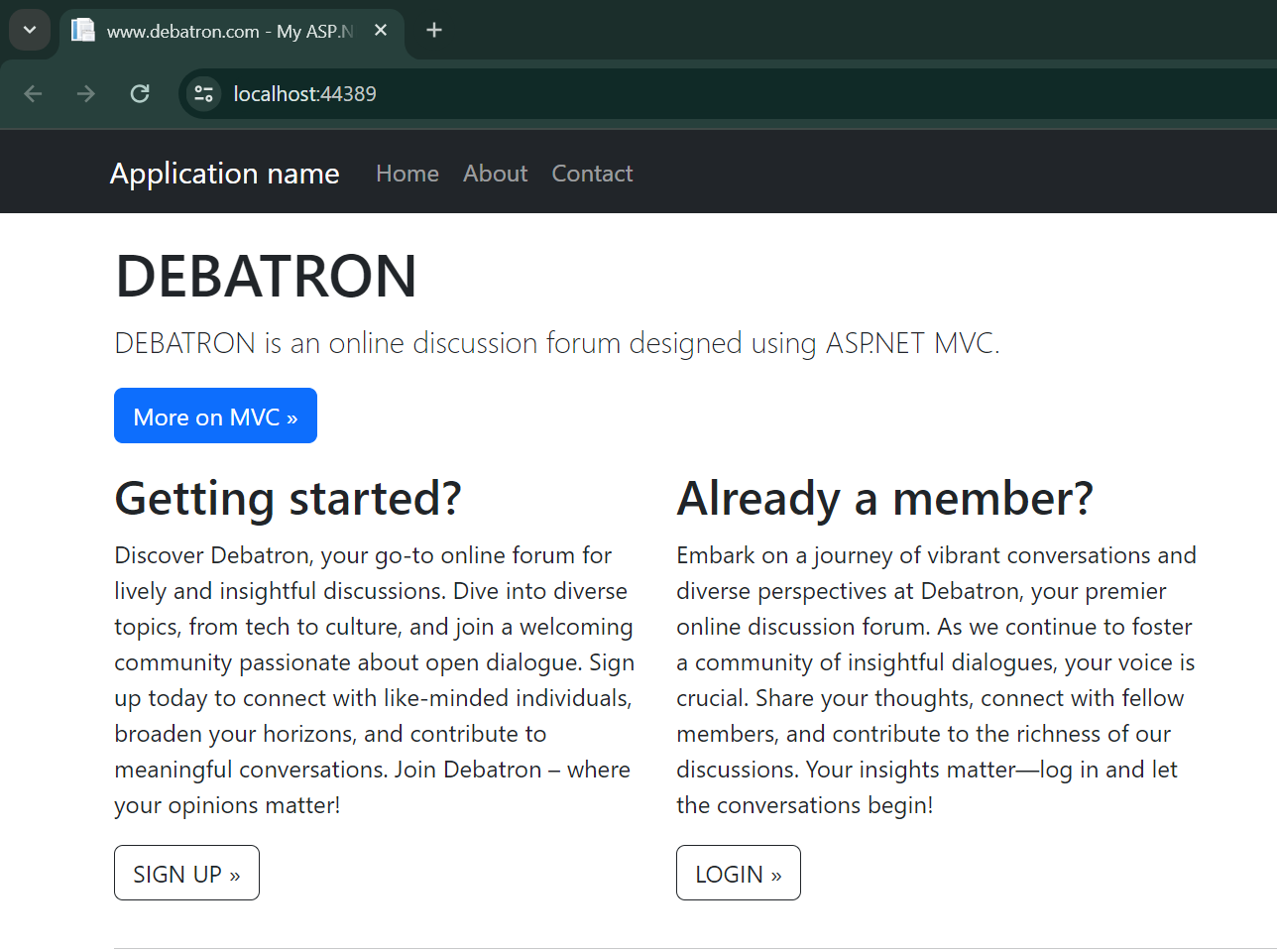
* **Designed without the Entity Framework:**

A screen shot of a computer

Description automatically generated

## **7.2. The Views (Razor Views)**

A screenshot of a computer

Description automatically generated

## **7.3. The Controllers**

A screenshot of a computer screen

Description automatically generated

**Chapter 8**

# **PROJECT VERSIONING**

We have created different variants of the project to achieve the desired outcomes. There is a total of seven versions of the project. Initially, we tried to use the entity framework in the ASP.NET Core MVC template and faced some challenges. After making three attempts to resolve these challenges (in versions 1,2 and 3 of Debatron), we switched to the ASP.NET MVC template and successfully created versions four, five, six and seven, with and without using the entity framework. The version conflicts in the EF compelled us to create this many versions. Finally, the version seven (Debatron-v7) had been chosen as the beta version of the project.

**Chapter 9**

# **CONCLUSION**

In conclusion, the development journey of Debatron, has been marked by continuous refinement and adaptability. We explored variations, including seven versions with and without Entity Framework, and transitioned from ASP.NET Core MVC to ASP.NET MVC. Despite encountering version conflicts with Entity Framework and other challenges (such as the use of the razor views, designing of the views and database integration), we achieved significant milestones by implementing essential features such as user authentication through login and signup functionalities. Additionally, the incorporation of a RESTful API further elevated the project's functionality, contributing to a robust and dynamic online community platform.

We made an attempt to make use of HTTP cookies to keep the users logged in in version 1 but we could not resolve the errors.

We look forward to continuing the development of the project and incorporating other features and functionalities such as voting the comments under the posts, reporting the comments, creating notifications for the users, and allowing the moderators to play their part in maintaining a good community.