

# EMBRACE

Elizabeth Pauley

## INTRODUCTION

Millions of people have immigrated to the United States and will continue to do so. When these individuals or families arrive, they can often experience a jarring transition period. Differences in language, culture, and other aspects of life, as well as the process of obtaining legal permission to live and work in the US, can all pose obstacles for those seeking to build happy, successful lives and raise families here.

There are a variety of ways one can attempt to address the language barrier, from using translation apps to learning a new language through courses or online resources. Unfortunately, translations can be inaccurate or lack the cultural connotation, and learning can take significant time, time that some families cannot afford, as navigating even basic daily tasks requires this skill almost immediately upon moving to the US. Obtaining visas or asylum is a long process in itself, but this is unfortunately required to obtain jobs and housing. This can leave a family in limbo with nothing to support them in the interim. Depending on where one lives, local resources might already be in place to help with this transition. However, these resources are not always as comprehensive as they should be. Furthermore, the true struggles immigrants face extend even further than the need for financial support or translation - feelings of isolation and hopelessness can arise when people feel unsupported and alone in their experiences, but it can be difficult to figure out how to connect to other individuals and families in a new community, especially in a new country.

Embrace is on a mission to bridge these gaps. This app provides both general and local resources in one centralized location. The general resources assist with difficulties common to immigrants across the country and can be used by anyone, no matter where they live. Our database of local resources provides solutions to problems specific to their new home, as well as where to find local resources and organizations available in one's area that offer a variety of supports relating to financial or legal aid, healthcare, education, and more. Regardless of where one comes from or where they are seeking to build a new life in the US, we want these individuals to feel that they are being welcomed into their new community with open arms, and as they face the challenges of adjusting to a new life, they will have access to the necessary support every step of the way.

# QUALIFICATIONS: ELIZABETH PAULEY

LinkedIn: [linkedin.com/in/elizabeth-pauley](https://www.linkedin.com/in/elizabeth-pauley)

## EDUCATION

**Montana State University** | Expected Graduation: May 2025

*Degree:* Bachelor of Science in Computer Science with minors in Data Science and Psychology

*Other education:* 5+ Years of Spanish Language Education, Adult Mental Health First Aid Certification

## PROFESSIONAL EXPERIENCE

**Intern at IntelliTect** | Spokane, WA

*May-Jul 2023, May-Jul 2024*

- Utilized PowerShell scripts and APIs to combine data from multiple databases for future analysis
- Performed data visualization and analysis using Microsoft Power BI
- Gained experience working with the Coalesce framework used to build web applications
- Implemented updates, bug fixes, and new features for an application used for company management
- Learned to use GitHub and engage in agile software development with other team members

## RELATED SKILLS

### Languages

Java

Python

C#

HTML & CSS

TypeScript

JavaScript

PowerShell

SQL

### Other Skills

Coalesce

Microsoft Power BI

Vue.js

EF Core

GitHub

## RELEVANT COURSES

Web Design

Database Systems

Data Mining

Software Engineering

Advanced Software

Engineering

## BACKGROUND

Using a basic search online, one can find endless resources targeted towards helping immigrants in the US in just about every area imaginable. However, most of these either provide a single type of resource, a comprehensive list of resources for a specific location, or an overall comprehensive list of resources, which is not very user-friendly when it comes to searching for a specific kind of resource, and can therefore be overwhelming. Additionally, the more comprehensive lists typically do not include location-specific resources, but rather more general ones.

The US government offers many resources for immigrants, but unfortunately, the system is not very centralized or easy to navigate. For example, the Department of Homeland Security has a page titled, “Find Citizenship Resources”. This page contains a link to the “Citizenship Resource Center”, which is a page on the US Citizenship and Immigration Services website. This landing page lists several categories of resources relating to citizenship, several of which are specifically targeted toward immigrants, while others are targeted toward educators or organizations seeking to support immigrants in their own settings. If a user clicks on the “Civic Integration” link, then the “Settling in the U.S.” link, this page lists out main resource categories such as “Education and Child Care”, “Employment”, “Money and Finance”, and more. Overall, while these sites do have search functionalities, they search the entire website rather than the available resources specifically, as resources are only one section of the site. The interface is not the most intuitive when users need to click link after link to find a specific resource, and mixing resources for immigrants and for those seeking to support immigrants makes the websites a little more overwhelming than they need to be (“Find Citizenship Resources: Homeland Security”).

Some great resources exist that focus on a specific kind of resource, such as the *Immigrant Legal Resource Center*. For immigrants seeking legal support, the interface is very simple and allows users to find legal aid for specific categories based on their specific location (“Immigrant Legal Resource Center”). *United We Dream* is another website that has a repository of resources with easy-to-use search functionality that has filtering and sorting by category. However, these resources are largely centered around providing all resources relating to DACA (Deferred Action for Childhood Arrivals), mental health support, education on immigrant rights, and how to become a more informed, politically active citizen who stands up for those rights (“Resources for Immigrants in the U.S.”). Overall, this website seems to be more focused on political activism and awareness and targeting immigrant youth in particular.

An example of a location-specific resource connector is *Bienvenidos*, which is an organization based in Bozeman, MT (the location of focus for Embrace’s demo described in the *Expected Results* section). This website does not necessarily have a database of resources a user can search through, but they have a program designed where “mentors” and volunteers are paired up with migrant families in the Gallatin Valley area, in order to connect these people with services and local community resources in a more personal way. This website is also specifically

targeted towards the Spanish-speaking community, which is the most prominent language among immigrants in Gallatin Valley (“Connecting with Resources”).

Overall, a plethora of resources exist to support immigrants in the US. However, as a new resident of the US, it might be difficult to know exactly how to sift through so many options and narrow down which ones are actually applicable to one’s needs. Embrace is a unique solution in that it will be more comprehensive in the types of resources it provides, offer more user-friendly search functionality, and allow for customization based on the user’s preferred language and physical location. This app also has a couple of additional features that set it apart from other solutions out there, including a communication tool that will allow users to directly connect with other users and a built-in document translation service.

## WORK SCHEDULE

This project started as a group endeavor but has since shifted to be an individual project completed by Elizabeth Pauley. This project will be developed using the Agile development approach to ensure the Minimum Viable Product is completed within the allotted time, with more complexity and additional features being added based on how much time is left in the provided time frame for this project.

The first major milestone of this project will be the initial setup of the project. This will include creating the ASP.NET Core project following the Model-View-Controller (MVC) design pattern and setting up the MySQL database. During this step, basic user authentication will be implemented so that users can successfully create an account, log in, and log out. After this, all of the necessary models and relationships for entities in the database will be defined, and database migrations will be set up using Entity Framework Core. Azure Blob Storage will also be set up to store files such as documents uploaded by users. Finally, it should be ensured that HTTPS is required for all requests and that all HTTP requests that might occur are redirected to HTTPS (“What Is ASP.NET Core?”; Anderson; Dykstra).

After the initial parts of the project are set up, the next milestones will be associated with the main features offered by Embrace. The most-touted feature of this app will be the resource database, so this feature should be the first one to be implemented. This will include creating the resource pages (both the list and detail views) with fully functional searching and filtering. Next, the basic functionality of the Connection Corner will be developed, which means users should be able to create, view, and comment on discussion boards. This step will also include the implementation of searching for boards that contain specific text or filtering by type of discussion. The final main feature is document translation, which should initially be created so that any text-based document (types accepted by the Google Cloud Translation API) can be uploaded and returned as a translated version in the user’s desired language. This step should also include secure storage of files in the database using encryption.

Following the main milestones, which are the initialization of the main features of Embrace, additional features and functionality can be developed. For the resource pages, this will include implementing the admin functionality for adding, editing, and deleting resources, and with this, handling image uploads and address verification using Azure Blob Storage and the USPS Addresses API (“Addresses 3.0”). A user dashboard should be created so that general users can view and update their account information and view saved resources. Searching and filtering functionality should also be implemented for the saved resources on the user’s dashboard. Additionally, the Organization Application feature can be added so that, if a specific resource is not displayed on Embrace, a user can submit an application for their resource to be added, then reviewed and either accepted or rejected by an admin user. If accepted, this will prompt the automatic creation of a new resource in the database. For the Connection Corner, additional functionality will include the ability for users to report discussions or comments (with the desire to remove them from the site). Finally, for the document translation feature, additional

functionality could be added for users to upload an image that contains text and have this file translated with the additional use of Google's Vision API, which uses optical character recognition (OCR) to extract text from images ("Detect Text in Images").

Following the expansion of pre-existing features, additional milestones will include the implementation of the Feedback Form for users to submit, and the option for users to save resources to their profiles for later viewing. Before deployment, some final milestones will include testing the UI/UX and addressing any bugs or design inconsistencies, although these should also be addressed along the way with each defined milestone. The final milestone for this project will be deploying the app to production.

# PROPOSAL STATEMENT

## TECHNOLOGY

We will be developing our web application using the ASP.NET Core framework, which provides a set of resources and tools that will help us build and manage our application more efficiently and in an organized manner. We will be implementing the Model-View-Controller (MVC) design pattern, which will be explained in more detail later on in this proposal. In general, this will involve using C# for our model and controller classes and a combination of HTML, CSS, JavaScript, Razor, and anything else needed to create the UI. HTML and CSS will be used to build out the structure and style of our app. JavaScript will be used to implement interactivity on the pages. Razor will allow us to integrate C# code into our HTML to create more dynamic views based on our data and user interactions.

We will be using MySQL as our relational database (tables with rows and columns) to store all information relating to users, discussion boards and comments, and any other related application data. Any documents that users upload will be stored as links to blob storage in the MySQL database, and an additional Azure Blob Storage account will be set up to store these larger files. Additionally, we will be using Entity Framework (EF) Core to map our C# model classes to tables in our MySQL database (Dykstra). Each model class represents an entity (or table) with properties that will correspond to columns in the aforementioned table in our database. For example, a Resource might be one entity that has attributes such as a Title, Description, Address, etc. While EF Core was not the only option explored, it is one of the most common choices for data access technologies used with ASP.NET Core and has some features that will overall make our process easier, such as automatic detection of relationships between entities in our model, simplified CRUD operations, and the option of migrations (this is largely based off of previous work experience listed in the Qualification section of this proposal). Since the main features of our web app are largely based on interacting with the database (uploading documents, viewing resources, posting on discussion boards), it will be especially helpful to have the built-in features of EF Core to manage interactions with and updates to the database.

The Google Cloud Translation API will be used to translate documents (“Cloud Translation API”). Later down the road, the Google Cloud Vision API may be used to extend this feature to accept files other than documents, such as images (“Detect Text in Images”). The USPS Addresses API will be used to verify addresses stored for users and resources. The ASP.NET Core Identity will be used to implement secure user authentication.



## REQUIREMENTS

### FUNCTIONAL REQUIREMENTS

#### *FR-001*

*Requirement Name:* Account Creation

*Requirement:* The system must allow users to create a new account if they do not have an existing one.

#### *FR-002*

*Requirement Name:* User Registration

*Requirement:* The system must allow new users to create an account by providing their email, username, password, address, and preferred language.

#### *FR-003*

*Requirement Name:* User Login

*Requirement:* The system must allow users to log in using their email and password.

#### *FR-004*

*Requirement Name:* User Logout

*Requirement:* The system must allow authenticated users to log out of their account.

#### *FR-005*

*Requirement Name:* Display List of Resources

*Requirement:* The system must display all resources stored in the database, as well as their associated overview information such as their name, description, and tags when in the list view.

#### *FR-006*

*Requirement Name:* Display Details of a Resource

*Requirement:* The system must display all information relating to a specific resource selected by a user when in the detail view.

#### *FR-007*

*Requirement Name:* Resource Saving

*Requirement:* The system should allow users to save a resource to their profile in either the list or detail view.

#### *FR-008*

*Requirement Name:* Resource Searching

*Requirement:* The system should allow users to search for resources by text that is contained in the resource name, description, or category.

*FR-009*

*Requirement Name:* Resource Filtering

*Requirement:* The system should allow for users to filter resources displayed from the database (in list view or on their profile) based on location, what type of resource they are looking for (in-person, online), or whether the user has saved a resource to their profile or not.

*FR-010*

*Requirement Name:* Resource Paging

*Requirement:* The system should display a certain number of resources on a given page, based on a number of predefined options provided to the user.

*FR-011*

*Requirement Name:* Resource Page Navigation

*Requirement:* The system should allow users to navigate back and forth between pages of resources if more than one page exists.

*FR-012*

*Requirement Name:* Resource Directions

*Requirement:* When in detail view, the system should allow users to click on a hyperlink tied to the address of a local resource and be redirected to Google Maps navigation to the resource.

*FR-013*

*Requirement Name:* Resource Website Access

*Requirement:* The system should allow users to click on a link associated with a resource's website and be redirected to said website.

*FR-014*

*Requirement Name:* Document Uploading

*Requirement:* The system must allow users to upload permitted document types to their profiles.

*FR-015*

*Requirement Name:* Document Translation

*Requirement:* The system should generate translated text from uploaded documents, provided the original and target languages.

*FR-016*

*Requirement Name:* Document Deletion

*Requirement:* The system should allow users to delete a previously uploaded document entity (deleting both the original and translated versions) stored on their account.

*FR-017*

*Requirement Name:* Display a User's Translated Documents

*Requirement:* The system should display all documents currently uploaded by the authenticated user on their dashboard.

*FR-018*

*Requirement Name:* Display List of Discussion Boards

*Requirement:* The system should display all discussion boards in the database, as well as their associated overview information, such as their title, description, type, and when it was last modified (date of most recent comment).

*FR-019*

*Requirement Name:* Display Details of a Discussion Board

*Requirement:* The system must display all information relating to a specific discussion board selected by a user when in the detail view, including the overview information and all comments posted by users.

*FR-020*

*Requirement Name:* Display Details of a Comment

*Requirement:* The system must display all information relating to each comment when viewing a discussion board in detail view, including the comment itself, the username of the user, and the date it was posted.

*FR-021*

*Requirement Name:* Error Handling

*Requirement:* The system must provide appropriate error messages if users provide invalid or incomplete inputs (login credentials, file types, etc.).

*FR-022*

*Requirement Name:* Organization Application Submission

*Requirement:* The system should allow a user to fill out an application for a new resource to be added by including all of the necessary information to create a new resource, as well as a preferred contact email.

*FR-023*

*Requirement Name:* Organization Application Review

*Requirement:* The system should allow an admin user to approve or reject an organization application.

*FR-024*

*Requirement Name:* Feedback Submission

*Requirement:* The system should allow a user to fill out a feedback form in relation to Embrace by providing a description of the feedback and a preferred contact email (optional).

*FR-025*

*Requirement Name:* Comment Reporting

*Requirement:* The system should allow a user to report a given comment on a discussion board by providing a description explaining what is wrong with the comment.

*FR-026*

*Requirement Name:* Discussion Board Reporting

*Requirement:* The system should allow a user to report a given discussion board by providing a description explaining what is wrong with the board.

## NON-FUNCTIONAL REQUIREMENTS

*NFR-001*

*Requirement Name:* Secure Transmission of Documents

*Requirement:* The system should use an HTTPS protocol when handling the uploading of a document by a user.

*NFR-002*

*Requirement Name:* Secure Translation of Documents

*Requirement:* The system should use a secure API in the process of translating user documents.

*NFR-003*

*Requirement Name:* Secure Storage of Documents

*Requirement:* The system should use encryption when storing documents in the database.

*NFR-004*

*Requirement Name:* Secure Storage of Documents

*Requirement:* The system should use an HTTPS protocol when handling the uploading of a document by a user.

*NFR-004*

*Requirement Name:* Secure Handling of User Information

*Requirement:* The system should use encryption during the transmission and storage of sensitive user information such as usernames and passwords.

*NFR-005*

*Requirement Name:* SQL Injection Prevention

*Requirement:* The database should be secured when accessed separately from the web app.

*NFR-006*

*Requirement Name:* Database Security

*Requirement:* The system must have security measures in place to prevent unauthorized access to the MySQL database when accessing the database directly.

*NFR-007*

*Requirement Name:* Usability of Different Web Pages

*Requirement:* The various pages of the web app should be easy to navigate back and forth.

*NFR-008*

*Requirement Name:* Presentation of the App's Purpose

*Requirement:* The main page of the site should provide the user with a clear understanding of what resources and services are available to them on the website.

*NFR-009*

*Requirement Name:* Page Load Response Time

*Requirement:* When switching from one page to another or navigating between various pages of resources listed on the site, the wait time before loading should be negligible (a few seconds at most).

*NFR-010*

*Requirement Name:* Translation Response Time

*Requirement:* When a user uploads a document for translation, it should take no longer than a few minutes for them to receive the translated version back.

*NFR-011*

*Requirement Name:* Throughput

*Requirement:* Users should be permitted to upload/request translation for one document at a time.

*NFR-012*

*Requirement Name:* Capacity

*Requirement:* A given user will be limited to a maximum of three translated documents on their profile at one time.

*NFR-013*

*Requirement Name:* Accessibility

*Requirement:* The web app should follow accessibility guidelines to ensure that those with disabilities can still navigate the site (font size, color schemes, etc.)

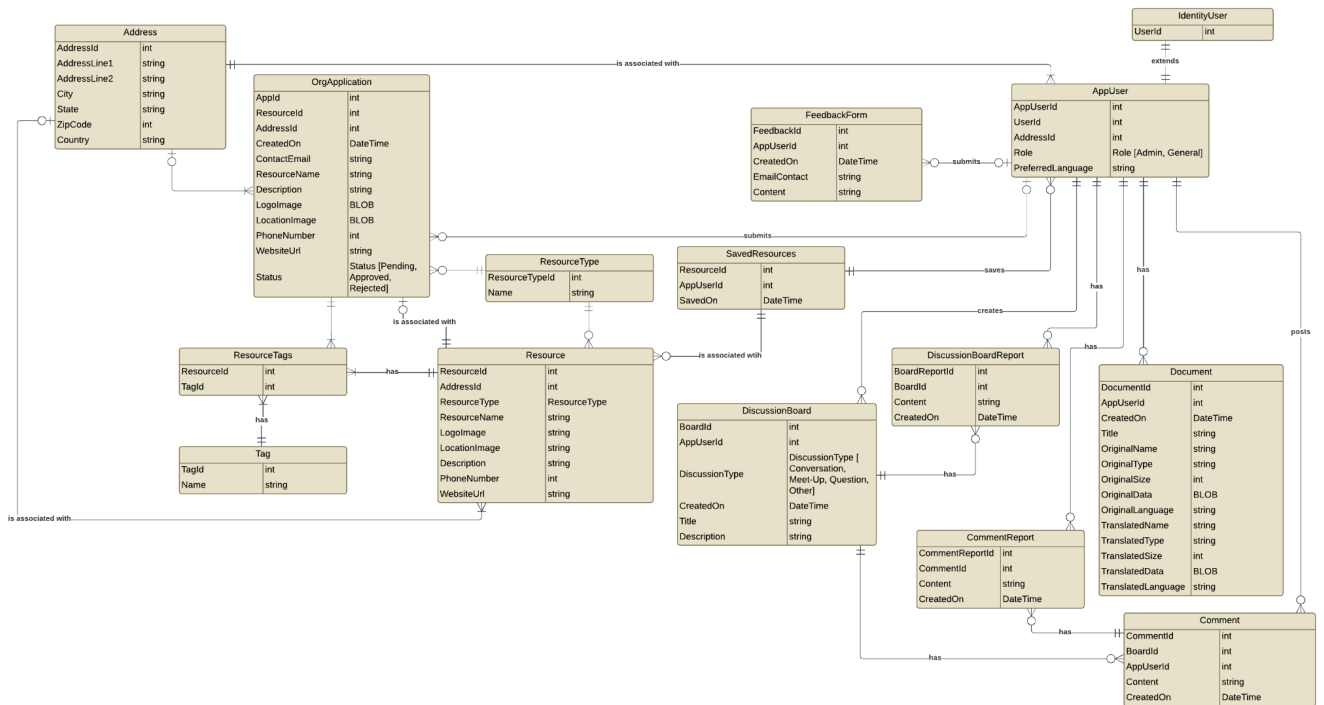
*NFR-014*

*Requirement Name:* Display of Data

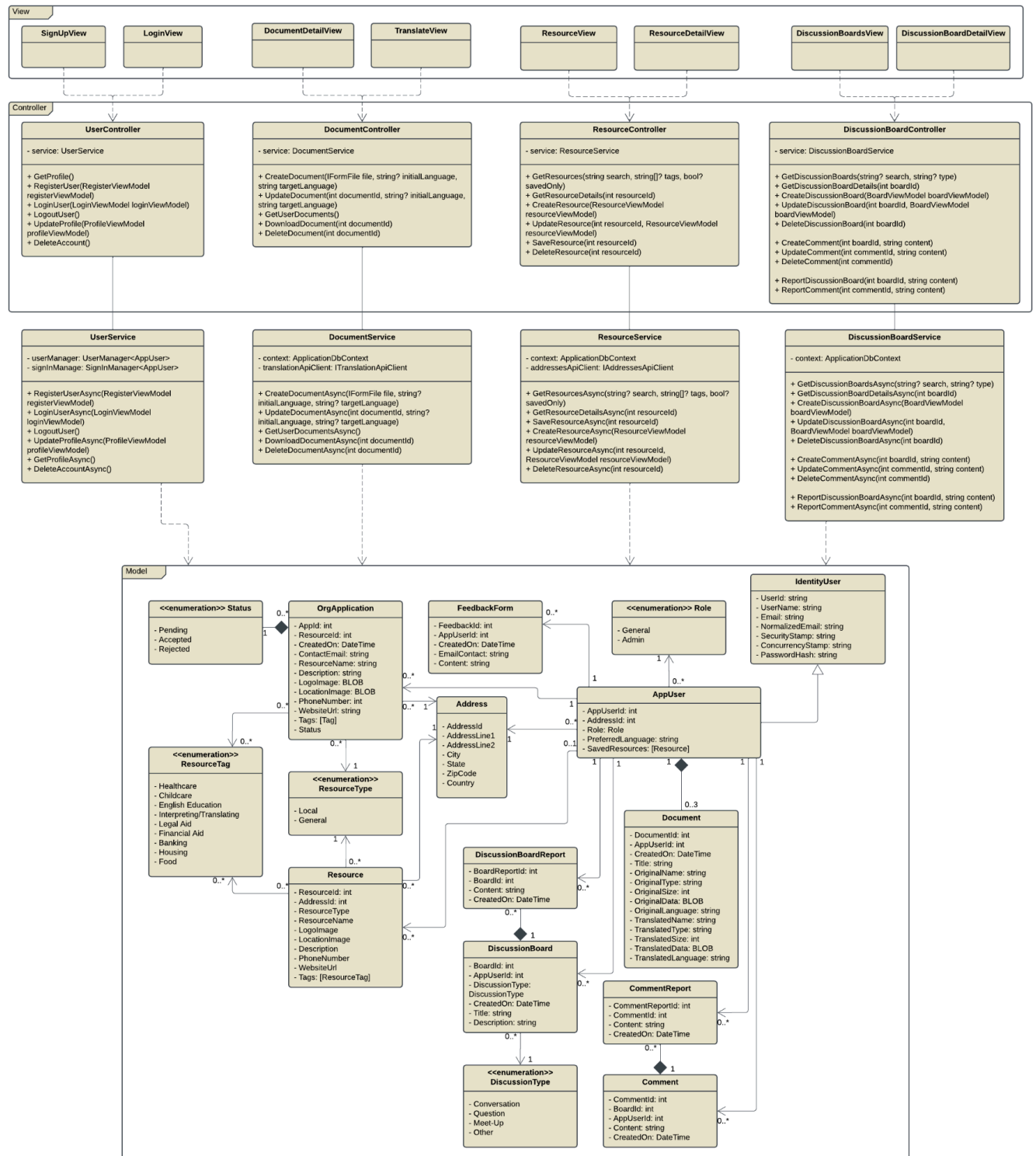
*Requirement:* The web app should have a flexible layout that adjusts according to screen size.

# METHODOLOGY

## ENTITY RELATIONSHIP DIAGRAM

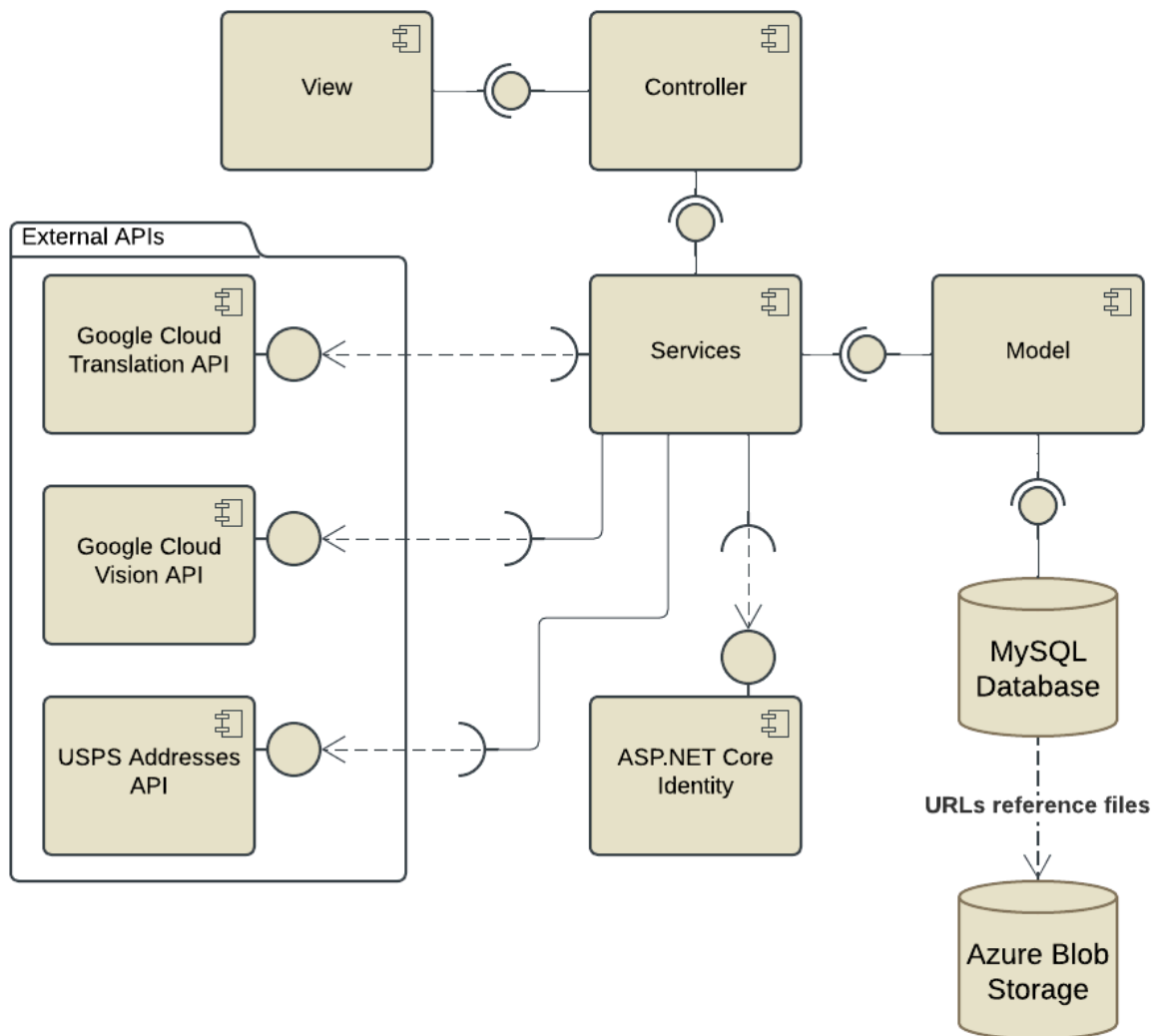


## CLASS DIAGRAM



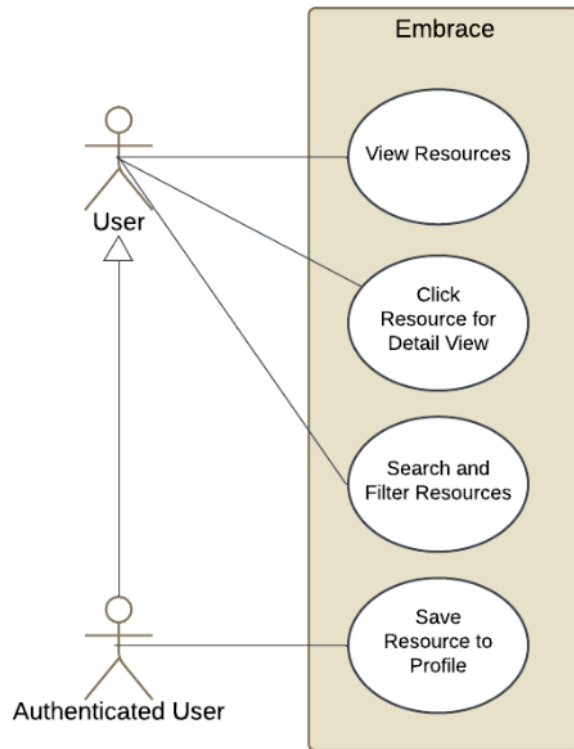


## COMPONENT DIAGRAM

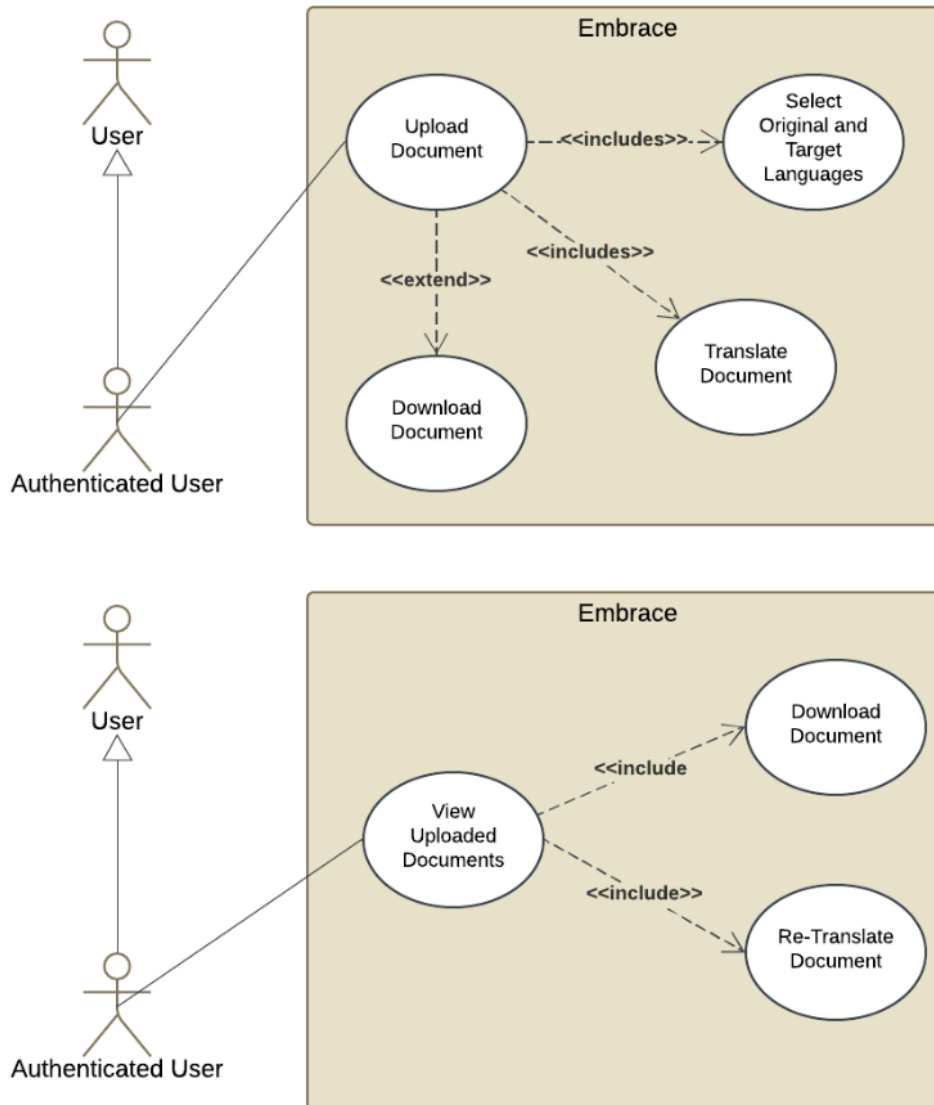


## USE CASE DIAGRAMS

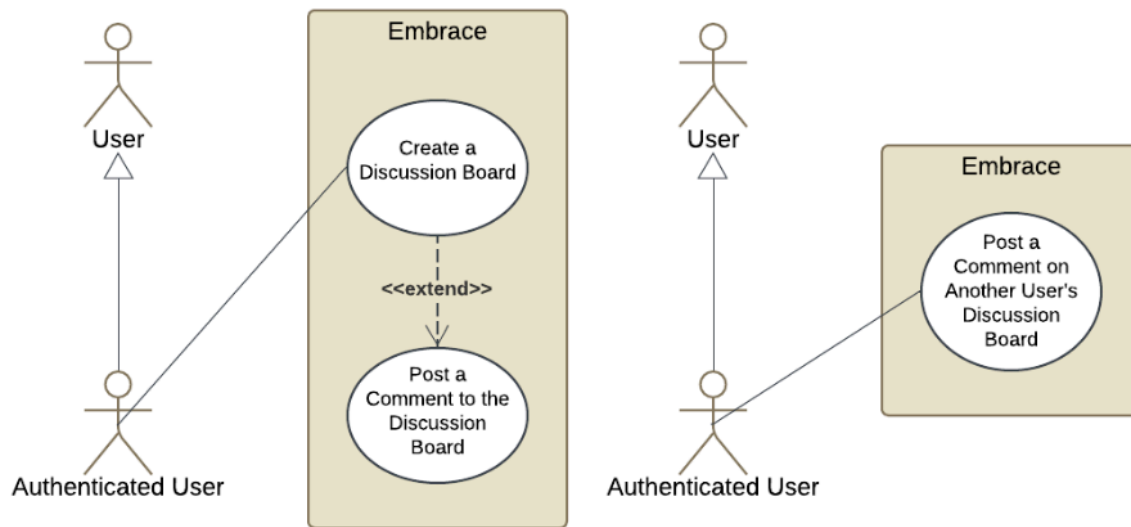
### FEATURE 1: RESOURCE HUB



## FEATURE 2: DOCUMENT TRANSLATION



## FEATURE 3: CONNECTION CORNER



## DESIGN PATTERN

Embrace will implement the Model View Controller (MVC) design pattern. Using the MVC design pattern will allow for the separation of the main aspects of the app, and is also a commonly used design pattern for this type of app.

Models will encompass all classes that represent the data used by the app. They are used to define entity attributes and relationships between other entities in the database. Model objects will also retrieve and store model states in the database. If new attributes or entities need to be added to the database, this can be done by creating a new model class or updating an existing one, then using EF Core to migrate that update to the MySQL database (Dykstra).

Controllers are classes that handle browser requests and call on Service classes, which interact with models to retrieve, create, update, or delete data, as well as handle more complex operations involving external services such as APIs or additional data manipulation. Controllers then provide responses in the form of data, status codes, etc., back to the user. In general, controllers act as an intermediary between the user and the model, controlling how the app behaves in response to user interactions, and calling on services to help handle logic relating to authentication, permissions, database access, and more.

Views are responsible for displaying data to and taking input from the user, then passing this input to the associated controllers.

The MVC pattern provides a standardized method of organization, where the model directly manages the data, the controller updates models and views based on how the user interacts with the app, and the views manage the display of data and other elements. There is a

clear separation between the front end and back end, making the overall development process much smoother (Chauhan; Anderson; “MVC Design Pattern”).

## TRADEOFFS AND DECISIONS

In the design process, the original plan was to use the MERN stack with MongoDB. However, this plan was later switched to using the ASP.NET Core framework with a MySQL database. The original MERN stack offered easy security and integration with some APIs. However, MongoDB as a non-relational database was not really a suitable choice for this app in particular. Additionally, ASP.NET Core offers built-in user authentication and allows for the efficient implementation of the necessary security practices for handling sensitive data on this app (Galvan).

Through this process, the Google Cloud Translation API was chosen to implement the document translation. While this API does have associated fees after a certain number of characters have been translated, it seems a better option to go with a well-known source such as Google, for reliability and security, since this API uses HTTPS (“Cloud Translation API”). Additionally, the API operates on a pay-as-you-go scale, which is a better option than a service that charges flat rates for different amounts of translations, as it is hard to predict how many users will want to use this feature at first. Additionally, given that it operates on a pay-as-you-go system, it could work well if the app needs to switch to making this an extra feature that users can pay for. While the goal is for this service to be free, even if users end up having to pay, this service is still in a centralized location among other resources and services on the Embrace app, which still fits into the overall purpose of Embrace (“Cloud Translation API”).

## EXPECTED RESULTS

Embrace is expected to be a useful and effective tool for immigrants in the United States, in that it can provide some extra assistance and support as they seek to find stability in their new life. Users will be able to translate documents, connect and communicate with one another, and locate resources that can help them navigate everyday tasks or various issues that might arise during the immigration process. Embrace will also be a useful tool for organizations looking to give their programs more exposure, as the app will make it easier for people to locate all the resources they need using a single application.

One expected feature of Embrace is its large database of local and general resources that will be provided to its users based on their location. This will be seen in our demo as a smaller database of resources local to Montana, which we will still be able to refine to specific cities, i.e. Bozeman or Billings. Some example general resources will be included as well, although not to the full scale that the final product would have. For the resources that are available in the demo, full searching and filtering functionality will be implemented.

The second feature expected from Embrace is the document translation service. This feature should be working at least for the accepted document types by the Google Cloud Translation API. OCR implementation such that images can also be uploaded will likely be a feature that can be added later on, after the demo is complete.

Embrace will also feature a Connection Corner page, where users can create boards for a desired topic and comment on one another's discussion boards. Users will also be able to search for specific discussions in a search bar and filter by type of discussion. This message board will be featured in our demo. Finally, our demo will also have a document translation feature, in which the user can submit a document as a PDF file to the web app, which will then return a PDF file with translations of the document in a selected target language.

The organization application, feedback forms, discussion board, and comment reporting, user and admin dashboards are all additional features that can be implemented later on down the road.

## RESULTS

The final prototype for Embrace included a welcome page, resource list and detail pages, a document translation page, and discussion board list and detail pages. User registration was also implemented. Related code for these features can be found under Appendix A.

For the resources feature, the database is populated with four example resources that can be filtered by resource type and service categories and searched by text within titles. Users can visit the detail page for each resource to see more in-depth information and links to related websites if available. For the document translation page, users can successfully upload, translate, and store documents on their profile in a wide variety of languages made available by the Google Cloud Translate API. One slight modification to the original proposal for this project is that Google Cloud Storage was used rather than Azure Storage, as this is the more standardized way for storing documents uploaded for translation. For the discussion board page, users can create discussion boards for a variety of categories and comment on both their own discussion boards and others.

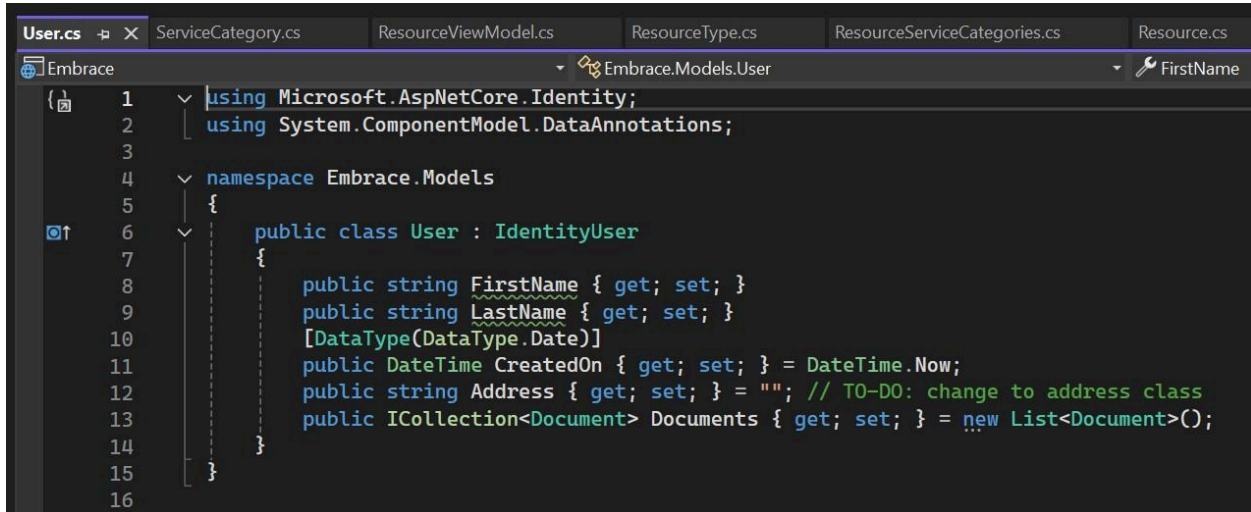
## DISCUSSION

Looking forward, many other features could be added to this project to further serve the community, as discussed earlier in the proposal. Given the time constraints of this project, only the three main features were developed to demonstrate the main purpose of the app. Additionally, more security measures would need to be implemented before making this project public, given that potentially very sensitive user information is being stored. Finally, some investigation into potential funding to keep the document translation feature free would be valuable so as to maintain accessibility for everyone. While all of this would require more time, effort, and money, it would be a worthwhile endeavor to create a safe and inviting space for immigrants and limited English-speaking individuals in our country to find the necessary tools and support they need to build strong foundations for their lives and integrate into their communities.

# APPENDIX A

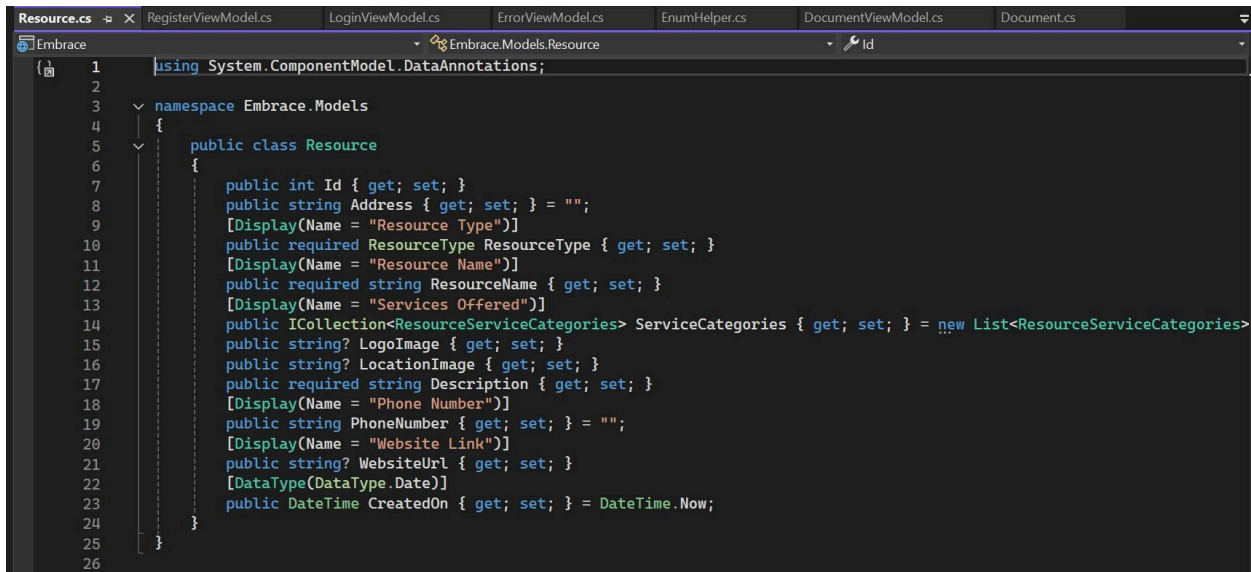
## SOURCE CODE

### MODELS



```
1 using Microsoft.AspNetCore.Identity;
2 using System.ComponentModel.DataAnnotations;
3
4 namespace Embrace.Models
5 {
6     public class User : IdentityUser
7     {
8         public string FirstName { get; set; }
9         public string LastName { get; set; }
10        [DataType(DataType.Date)]
11        public DateTime CreatedOn { get; set; } = DateTime.Now;
12        public string Address { get; set; } = ""; // TO-DO: change to address class
13        public ICollection<Document> Documents { get; set; } = new List<Document>();
14    }
15 }
16
```

Figure 1: User Model



```
1 using System.ComponentModel.DataAnnotations;
2
3 namespace Embrace.Models
4 {
5     public class Resource
6     {
7         public int Id { get; set; }
8         public string Address { get; set; } = "";
9         [Display(Name = "Resource Type")]
10        public required ResourceType ResourceType { get; set; }
11        [Display(Name = "Resource Name")]
12        public required string ResourceName { get; set; }
13        [Display(Name = "Services Offered")]
14        public ICollection<ResourceServiceCategories> ServiceCategories { get; set; } = new List<ResourceServiceCategories>();
15        public string? LogoImage { get; set; }
16        public string? LocationImage { get; set; }
17        public required string Description { get; set; }
18        [Display(Name = "Phone Number")]
19        public string PhoneNumber { get; set; } = "";
20        [Display(Name = "Website Link")]
21        public string? WebsiteUrl { get; set; }
22        [DataType(DataType.Date)]
23        public DateTime CreatedOn { get; set; } = DateTime.Now;
24    }
25 }
26
```

Figure 2: Resource Model



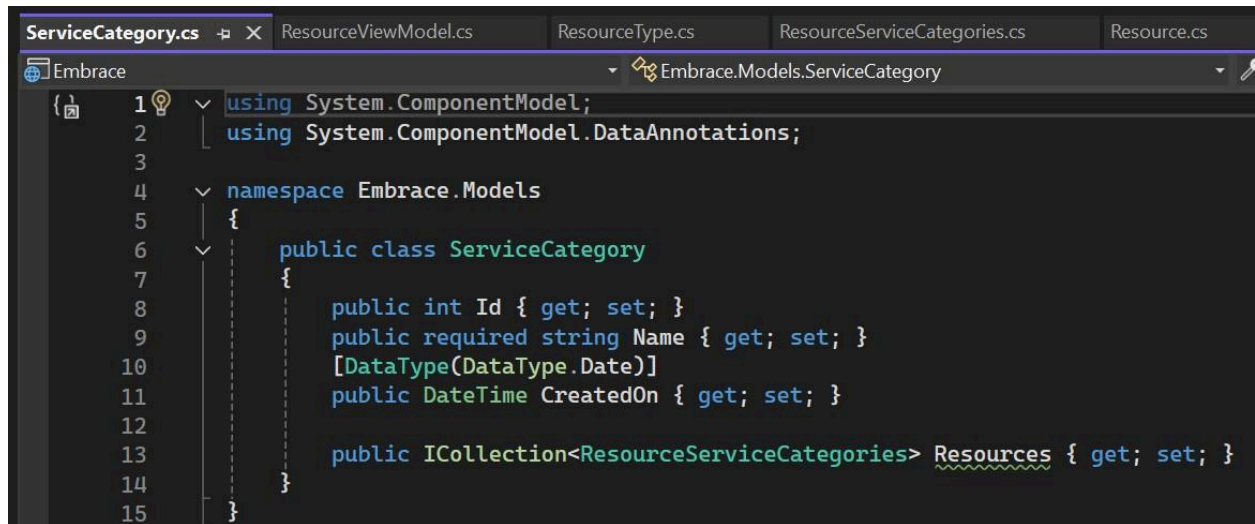


Figure 3: Service Category Model

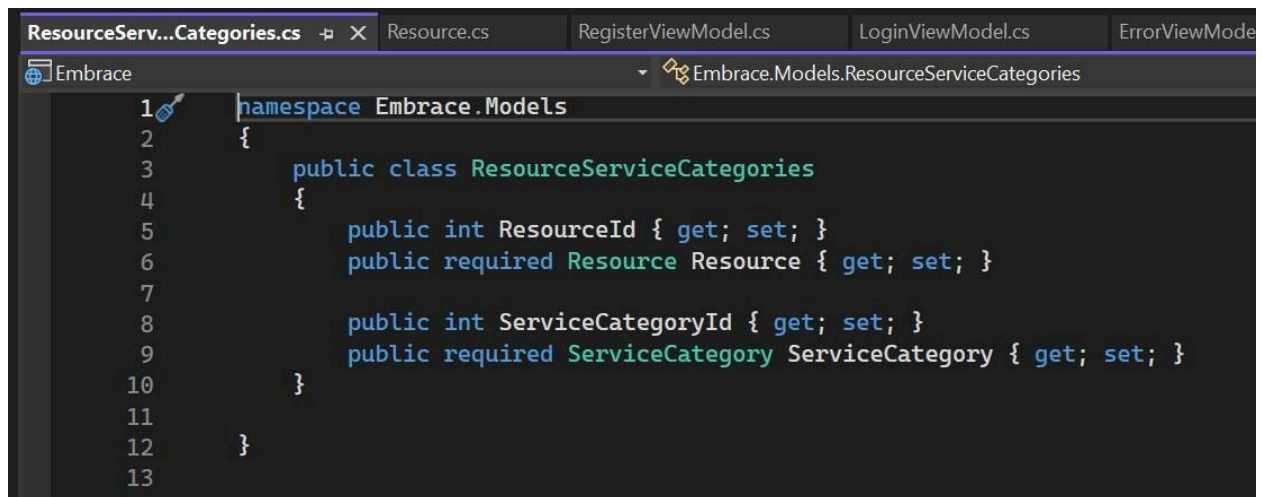


Figure 4: Resource Service Categories Model

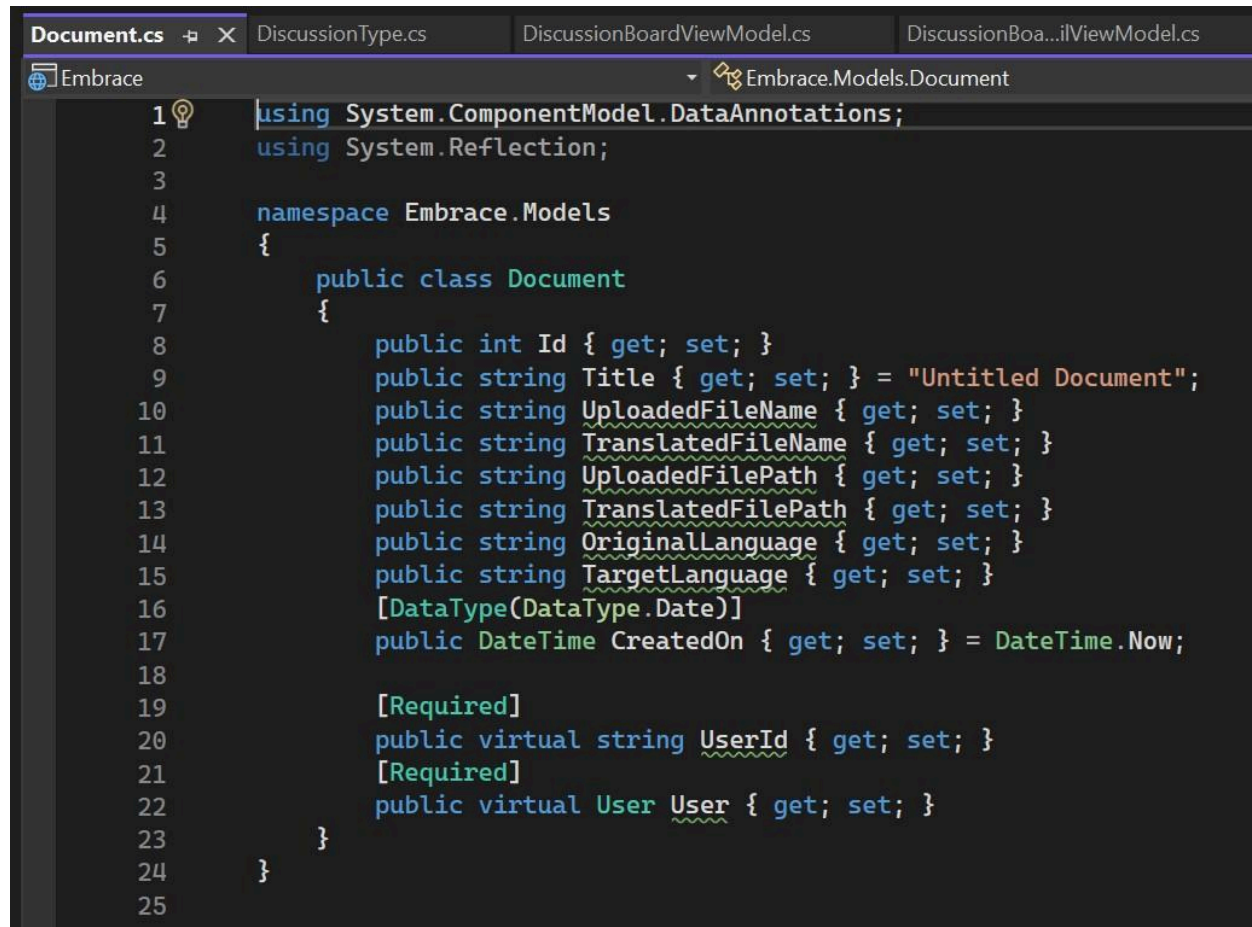


Figure 5: Document Model

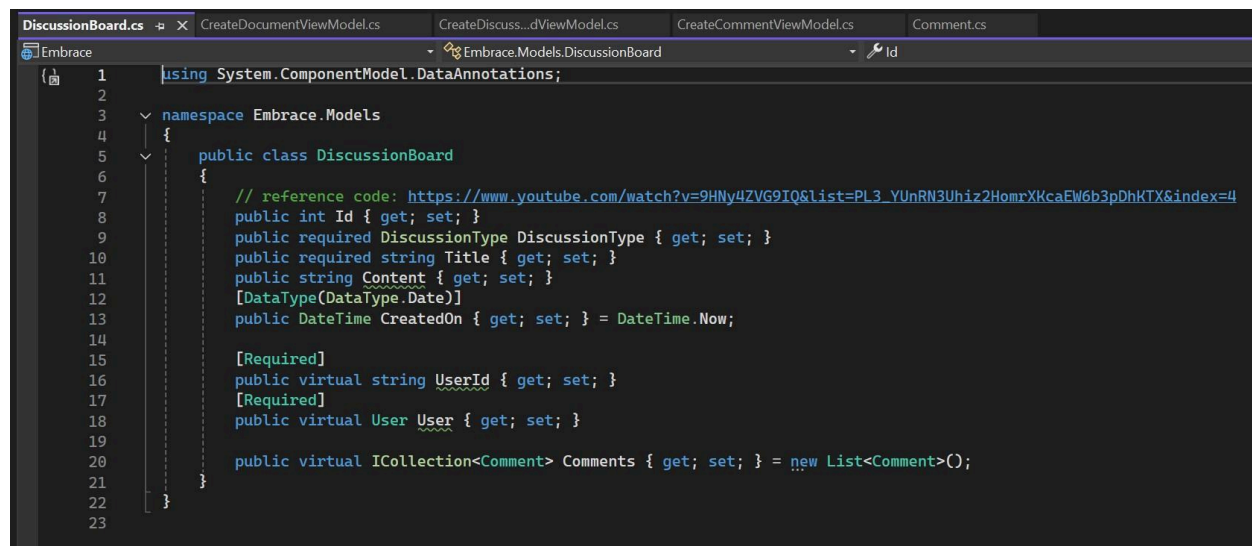


Figure 6: Discussion Board Model

```
Comment.cs
Embrace
using System.ComponentModel.DataAnnotations;

namespace Embrace.Models
{
    public class Comment
    {
        public int Id { get; set; }
        public required string Content { get; set; }

        [DataType(DataType.Date)]
        public DateTime CreatedOn { get; set; } = DateTime.Now;

        [Required]
        public virtual User User { get; set; }
        [Required]
        public virtual DiscussionBoard DiscussionBoard { get; set; }

        [Required]
        public virtual string UserId { get; set; }
        [Required]
        public virtual int DiscussionBoardId { get; set; }
    }
}
```

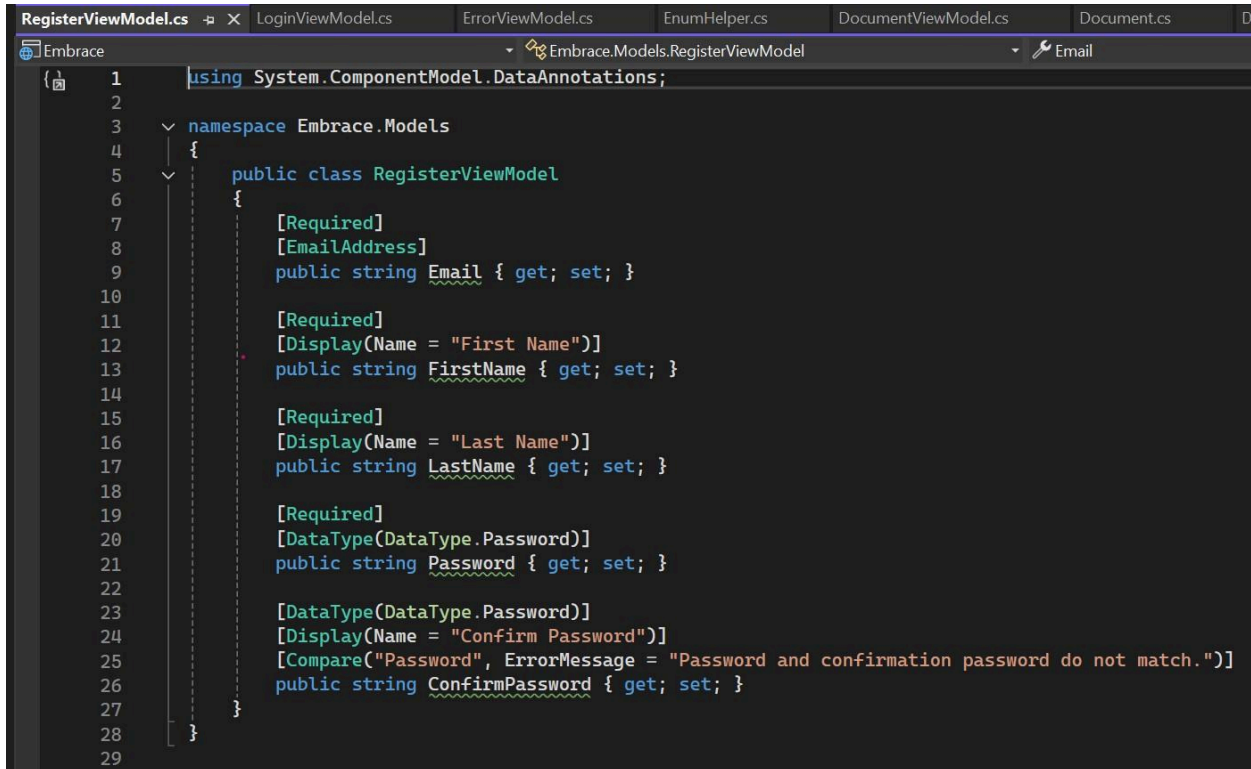
Figure 7: Comment Model

## VIEW MODELS

```
ResourceViewModel.cs
ResourceType.cs
ResourceServiceCategories.cs
Resource.cs
Embrace
using Microsoft.AspNetCore.Mvc.Rendering;

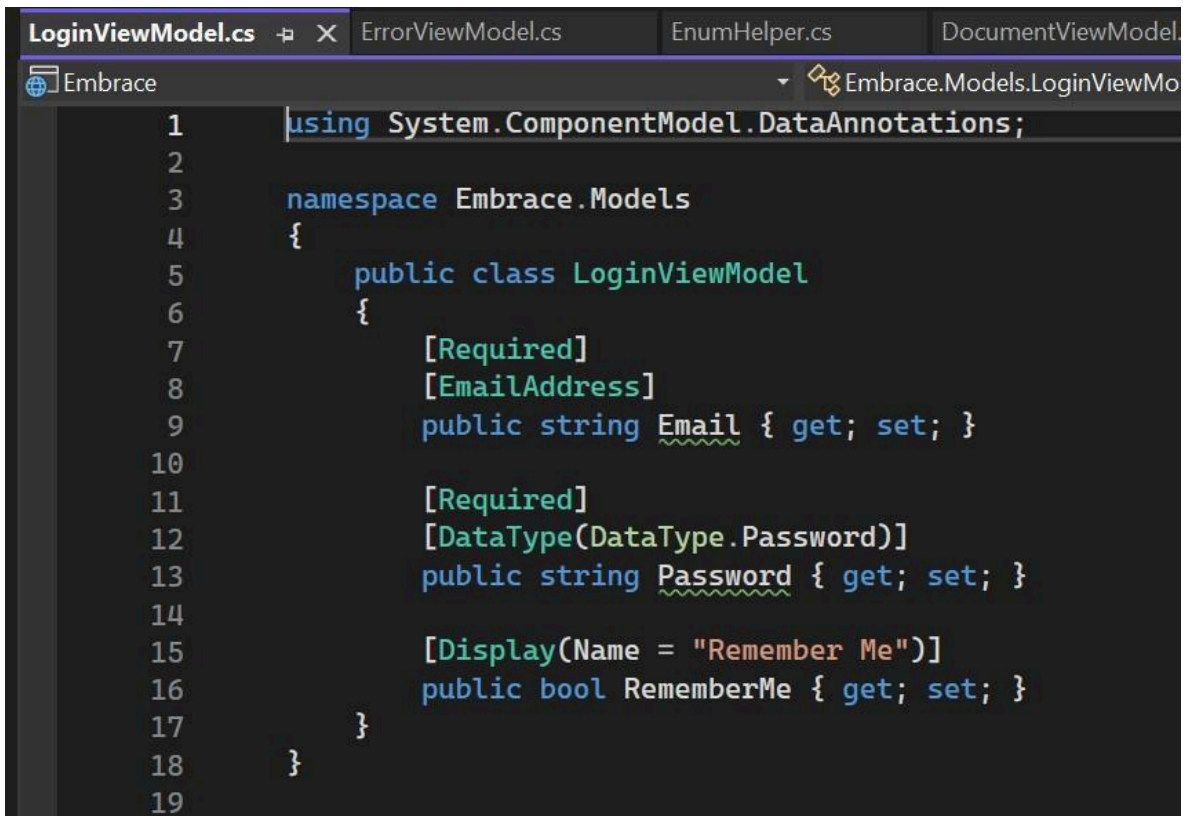
namespace Embrace.Models
{
    public class ResourceViewModel
    {
        public List<Resource>? Resources { get; set; }
        public ResourceType? ResourceType { get; set; }
        public int? ServiceCategoryId { get; set; }
        public SelectList? ResourceTypes { get; set; }
        public SelectList? ServiceCategories { get; set; }
        public string? SearchString { get; set; }
    }
}
```

Figure 8: Resource View Model



```
1 using System.ComponentModel.DataAnnotations;
2
3 namespace Embrace.Models
4 {
5     public class RegisterViewModel
6     {
7         [Required]
7         [EmailAddress]
8         public string Email { get; set; }
9
10
11         [Required]
11         [Display(Name = "First Name")]
12         public string FirstName { get; set; }
13
14
15         [Required]
15         [Display(Name = "Last Name")]
16         public string LastName { get; set; }
17
18
19         [Required]
19         [DataType(DataType.Password)]
20         public string Password { get; set; }
21
22
23         [DataType(DataType.Password)]
23         [Display(Name = "Confirm Password")]
24         [Compare("Password", ErrorMessage = "Password and confirmation password do not match.")]
25         public string ConfirmPassword { get; set; }
26     }
27 }
28
29
```

Figure 9: Register View Model



```
1 using System.ComponentModel.DataAnnotations;
2
3 namespace Embrace.Models
4 {
5     public class LoginViewModel
6     {
7         [Required]
7         [EmailAddress]
8         public string Email { get; set; }
9
10
11         [Required]
11         [DataType(DataType.Password)]
12         public string Password { get; set; }
13
14
15         [Display(Name = "Remember Me")]
16         public bool RememberMe { get; set; }
17     }
18 }
19
```

Figure 10: Login View Model



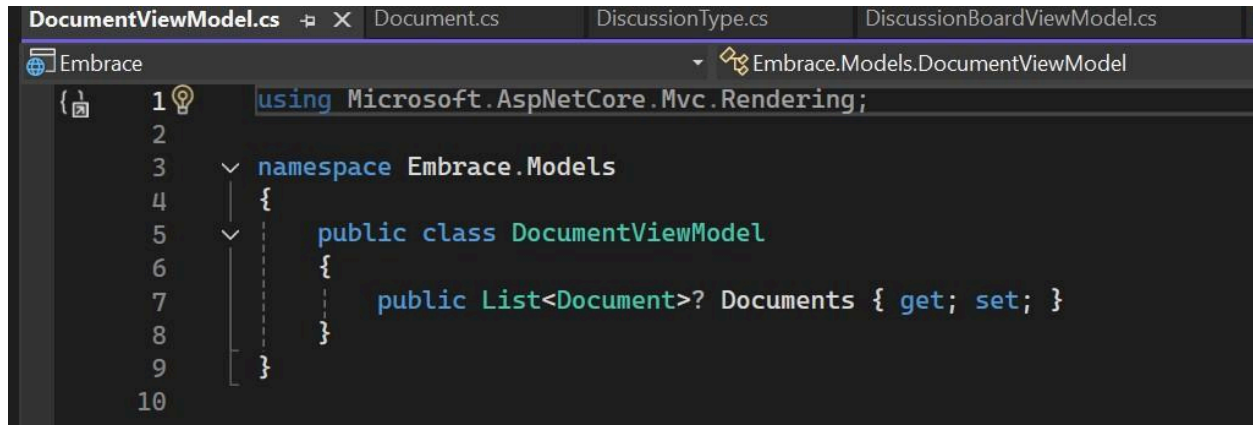


Figure 11: Document View Model

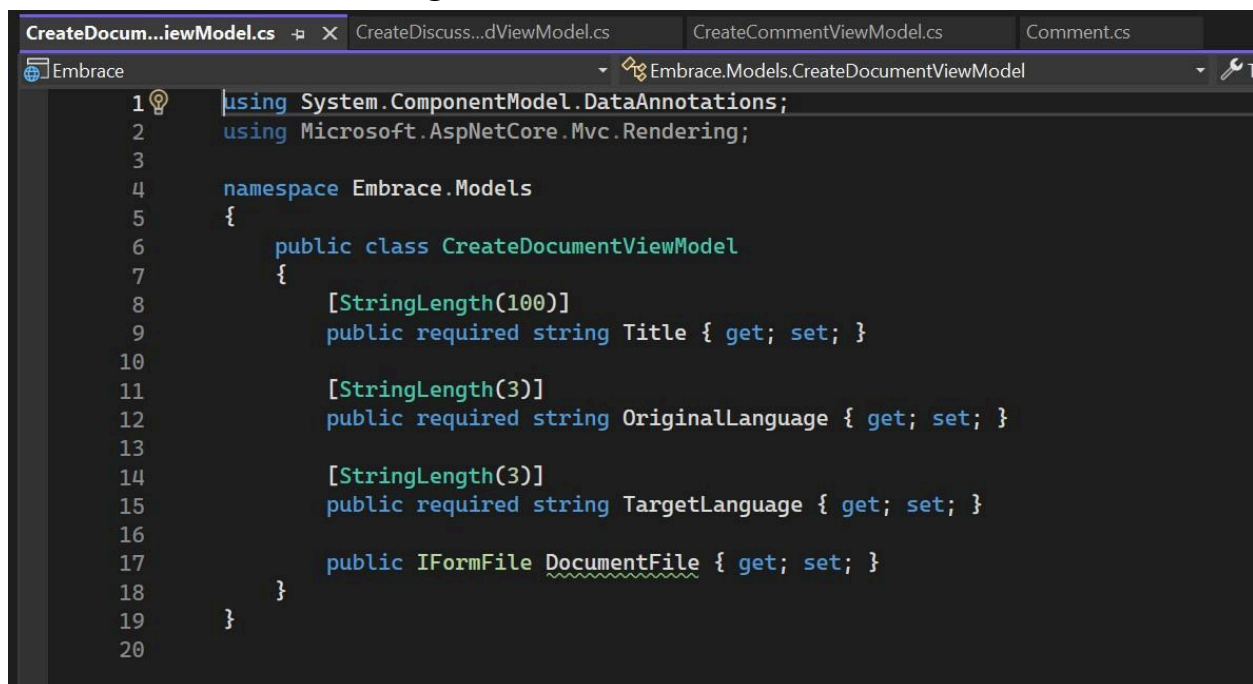


Figure 12: Create Document View Model

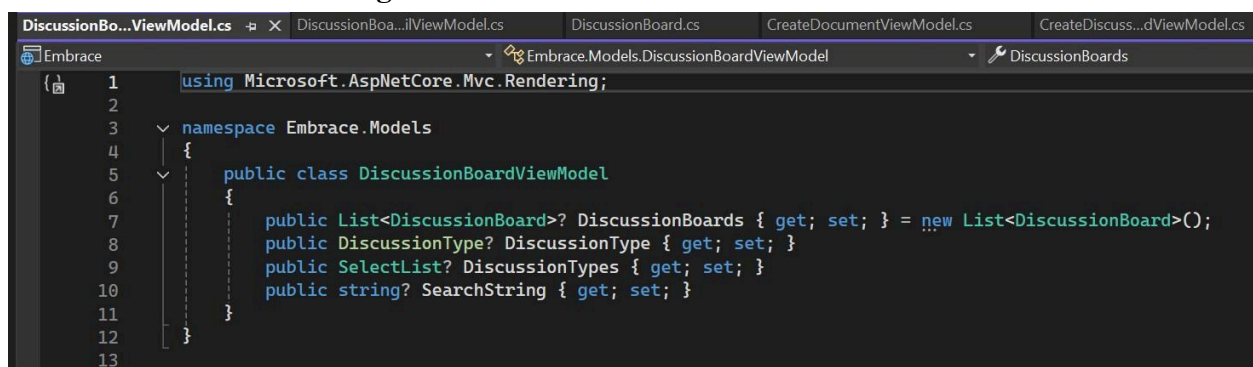
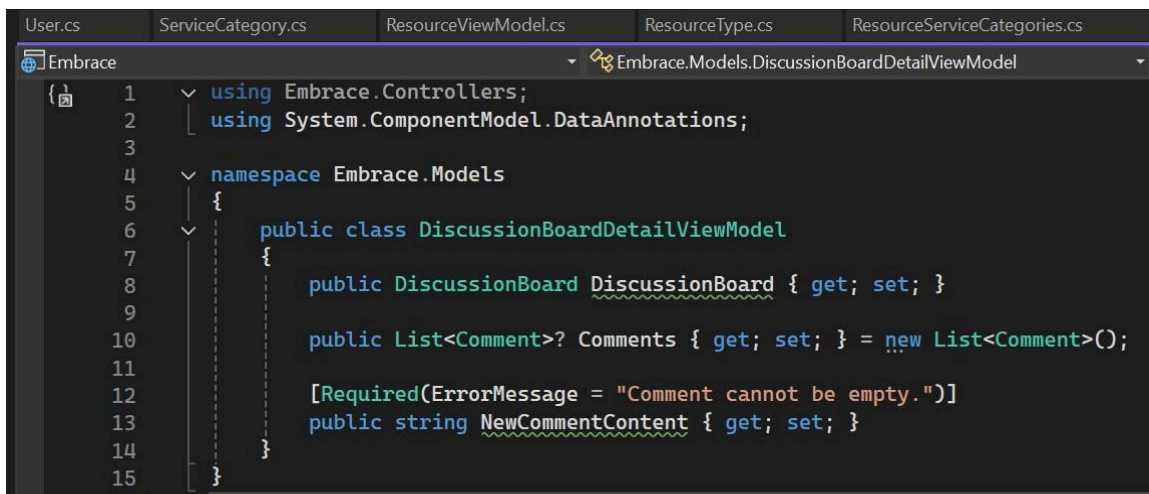


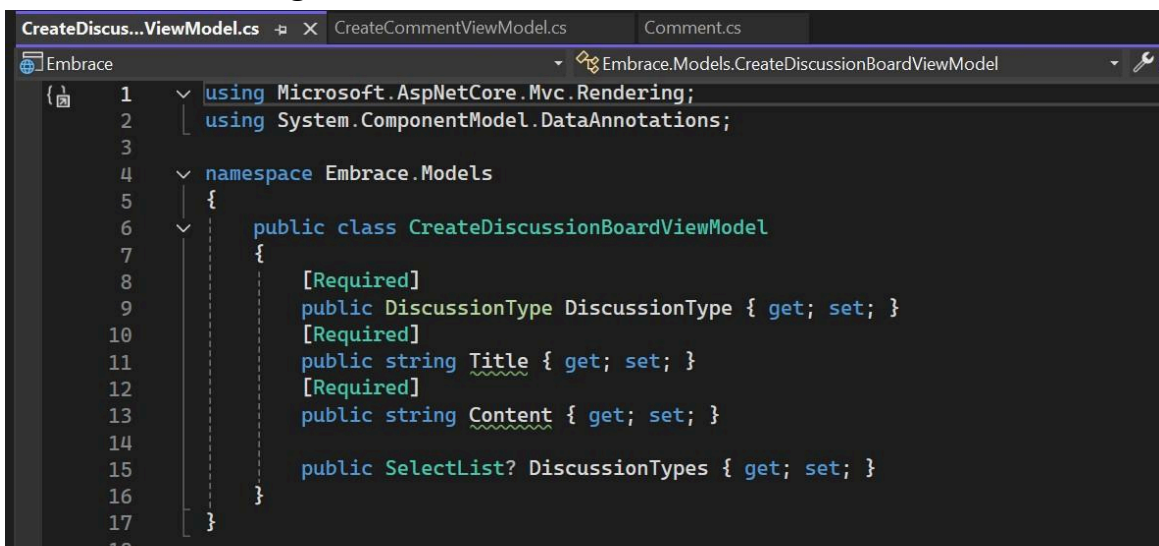
Figure 13: Discussion Board View Model



```

1  using Embrace.Controllers;
2  using System.ComponentModel.DataAnnotations;
3
4  namespace Embrace.Models
5  {
6      public class DiscussionBoardDetailViewModel
7      {
8          public DiscussionBoard DiscussionBoard { get; set; }
9
10         public List<Comment>? Comments { get; set; } = new List<Comment>();
11
12         [Required(ErrorMessage = "Comment cannot be empty.")]
13         public string NewCommentContent { get; set; }
14     }
15 }
    
```

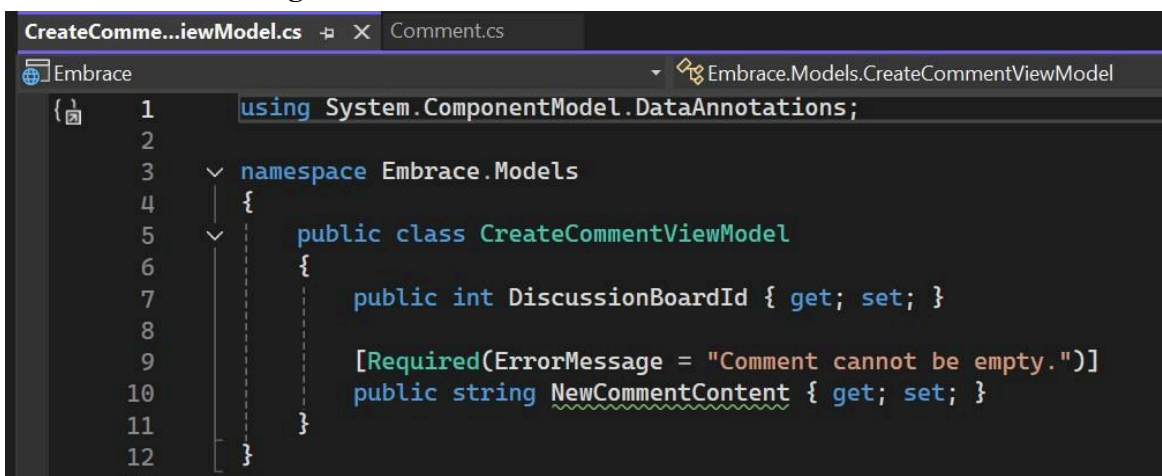
Figure 14: Discussion Board Detail View Model



```

1  using Microsoft.AspNetCore.Mvc.Rendering;
2  using System.ComponentModel.DataAnnotations;
3
4  namespace Embrace.Models
5  {
6      public class CreateDiscussionBoardViewModel
7      {
8          [Required]
9          public DiscussionType DiscussionType { get; set; }
10         [Required]
11         public string Title { get; set; }
12         [Required]
13         public string Content { get; set; }
14
15         public SelectList? DiscussionTypes { get; set; }
16     }
17 }
    
```

Figure 15: Create Discussion Board View Model



```

1  using System.ComponentModel.DataAnnotations;
2
3  namespace Embrace.Models
4  {
5      public class CreateCommentViewModel
6      {
7          public int DiscussionBoardId { get; set; }
8
9          [Required(ErrorMessage = "Comment cannot be empty.")]
10         public string NewCommentContent { get; set; }
11     }
12 }
    
```

Figure 16: Create Comment View Model

## CONTROLLERS

*Note: The following methods were added or modified within the generated controller code.*

```
73 // GET: Resources1/Details/5
74 public async Task<IActionResult> Details(int? id)
75 {
76     if (id == null)
77     {
78         return NotFound();
79     }
80
81     var resource = await _context.Resources
82         .Include(r => r.ServiceCategories)
83         .ThenInclude(rc => rc.ServiceCategory)
84         .FirstOrDefaultAsync(m => m.Id == id);
85
86     if (resource == null)
87     {
88         return NotFound();
89     }
90
91     return View(resource);
92 }
93
94 // GET: Resources1/Create
95 public IActionResult Create()
96 {
97     return View();
98 }
```

**Figure 17: Resource Detail Method**

```

25 // GET: Resources
26 public async Task<IActionResult> Index(ResourceType? resourceType, int? serviceCategoryId, string? searchString)
27 {
28     var resourcesQuery = _context.Resources
29         .Include(r => r.ServiceCategories)
30         .ThenInclude(rc => rc.ServiceCategory)
31         .AsQueryable();
32
33     // Filter by search term
34     if (!string.IsNullOrEmpty(searchString))
35     {
36         resourcesQuery = resourcesQuery.Where(x => x.ResourceName!.ToUpper().Contains(searchString.ToUpper()));
37     }
38
39     // Filter by resource type
40     if (resourceType != null)
41     {
42         resourcesQuery = resourcesQuery.Where(x => x.ResourceType == resourceType);
43     }
44
45     // Filter by service category
46     if (serviceCategoryId.HasValue)
47     {
48         resourcesQuery = resourcesQuery.Where(x => x.ServiceCategories
49             .Any(sc => sc.ServiceCategory.Id == serviceCategoryId));
50     }
51
52     var resources = await resourcesQuery.ToListAsync();
53
54     // Use LINQ to get list of resource types + tags
55     IQueryable<ResourceType> resourceTypeQuery = from r in _context.Resources
56         orderby r.ResourceType
57         select r.ResourceType;
58
59     IOrderedQueryable<ServiceCategory> resourceServiceCategoryQuery = from sc in _context.ServiceCategories
60         orderby sc.Name
61         select sc;
62
63     var resourceVM = new ResourceViewModel
64     {
65         ResourceTypes = new SelectList(await resourceTypeQuery.Distinct().ToListAsync()),
66         ServiceCategories = new SelectList(await resourceServiceCategoryQuery.Distinct().ToListAsync(), "Id", "Name"),
67         Resources = resources
68     };
69
70     return View(resourceVM);
71 }

```

Figure 18: Resource Index Method

```

34
35 public DocumentsController(ApplicationDbContext context, IWebHostEnvironment hostEnvironment, UserManager<User> userManager, IConfiguration configuration)
36 {
37     _context = context;
38     _userManager = userManager;
39     _hostEnvironment = hostEnvironment;
40     _uploadedDocumentsBucket = "embrace-uploaded-documents";
41     _translatedDocumentsBucket = "embrace-translated-documents";
42
43     _configuration = configuration;
44
45     var serviceAccountKeyPath = _configuration["GoogleCloud:ServiceAccountKey"];
46     if (!string.IsNullOrEmpty(serviceAccountKeyPath)) // add later: && File.Exists(serviceAccountKeyPath) - was throwing an error for some reason
47     {
48         Environment.SetEnvironmentVariable("GOOGLE_APPLICATION_CREDENTIALS", serviceAccountKeyPath);
49     }
50
51     _client = TranslationServiceClient.Create();
52 }
53
54 // GET: Documents
55 public async Task<IActionResult> Index()
56 {
57     var userId = _userManager.GetUser(User);
58     var applicationDbContext = _context.Documents.Include(d => d.User)
59         .Where(x => x.UserId == userId);
60     var documents = await applicationDbContext.AsQueryable().ToListAsync();
61
62     var documentVM = new DocumentViewModel
63     {
64         Documents = documents
65     };
66
67     return View(documentVM);
68 }

```

Figure 19: Document Controller Setup Code



```
292 // GET: Documents/ViewDocument/5
293
294 public async Task<IActionResult> ViewDocument(int? id)
295 {
296     if (id == null)
297     {
298         return NotFound();
299     }
300
301     var document = await _context.Documents
302         .Include(d => d.User)
303         .FirstOrDefaultAsync(m => m.Id == id);
304     if (document == null)
305     {
306         return NotFound();
307     }
308
309     if (string.IsNullOrEmpty(document.TranslatedFilePath))
310     {
311         return NotFound("No translated file path is available.");
312     }
313
314     try
315     {
316         var storageClient = StorageClient.Create();
317
318         // Download the document into a MemoryStream.
319         var memoryStream = new MemoryStream();
320         await storageClient.DownloadObjectAsync(_translatedDocumentsBucket, document.TranslatedFileName, memoryStream);
321         memoryStream.Position = 0;
322
323         string contentType = "application/pdf";
324
325         return File(memoryStream, contentType);
326     }
327     catch (Exception ex)
328     {
329         // Log the exception and return an error view or message.
330         Console.WriteLine($"Error downloading file from GCS: {ex.Message}");
331         throw;
332     }
333 }
```

Figure 20: View Document Method

```

99 [HttpPost]
100 [ValidateAntiForgeryToken]
101 public async Task<IActionResult> Create(CreateDocumentViewModel vm)
102 {
103     if (!ModelState.IsValid)
104     {
105         return View(vm);
106     }
107
108     // Get currently logged in user
109     var userId = _userManager.GetUserId(User);
110
111     var document = new Models.Document
112     {
113         Title = vm.Title,
114         OriginalLanguage = vm.OriginalLanguage,
115         TargetLanguage = vm.TargetLanguage,
116         UserId = userId,
117     };
118
119     if (vm.DocumentFile != null && vm.DocumentFile.Length > 0)
120     {
121         // Generate a unique file name and ensure it has a .pdf extension.
122         var uniqueFileName = $"{Guid.NewGuid()}_{document.Title.Replace(" ", "")}.pdf";
123         var gcsUploadedFileUri = $"gs://{_uploadedDocumentsBucket}/{uniqueFileName}.pdf";
124         document.UploadedFileName = uniqueFileName + ".pdf";
125         document.UploadedFilePath = gcsUploadedFileUri;
126         document.TranslatedFilePath = "";
127
128         // Build the output URI prefix for translated documents
129         var gcsTranslatedFileUriPrefix = $"gs://{_translatedDocumentsBucket}/";
130
131         // Upload the file to Google Cloud Storage using the StorageClient
132         var storageClient = StorageClient.Create();
133         using (var stream = vm.DocumentFile.OpenReadStream())
134         {
135             var uploadObject = await storageClient.UploadObjectAsync(
136                 _uploadedDocumentsBucket,
137                 uniqueFileName + ".pdf",
138                 "application/pdf",
139                 stream
140             );
141
142             if (uploadObject != null)
143             {
144                 document.UploadedFilePath = gcsUploadedFileUri;
145             }
146             else
147             {
148                 Console.WriteLine("Upload Failed.");
149                 return View(vm);
150             }
151         }
152     }

```

Figure 21: Create Document Method

```

152
153 // TRANSLATE DOCUMENT
154 var request = new TranslateDocumentRequest
155 {
156     Parent = $"projects/embrace-456119/locations/global",
157     SourceLanguageCode = vm.OriginalLanguage,
158     TargetLanguageCode = vm.TargetLanguage,
159     DocumentInputConfig = new DocumentInputConfig
160     {
161         GcsSource = new GcsSource
162         {
163             InputUri = gcsUploadedFileUri
164         },
165         MimeType = "application/pdf"
166     },
167     DocumentOutputConfig = new DocumentOutputConfig
168     {
169         GcsDestination = new GcsDestination
170         {
171             // Provide the output destination; OutputUriPrefix means the file will get saved with a generated name
172             OutputUriPrefix = gcsTranslatedFileUriPrefix,
173         },
174         MimeType = "application/pdf"
175     },
176     IsTranslateNativePdfOnly = false
177 };
178 var response = await _client.TranslateDocumentAsync(request);
179 Console.WriteLine($"Document translated. Response: {response}");
180
181 // save output URI to access document later
182 var prefix = "embrace-uploaded-documents_" + uniqueFileName;
183 var objects = storageClient.ListObjects(_translatedDocumentsBucket, prefix);
184 var finalObject = objects.FirstOrDefault();
185 document.TranslatedFileName = finalObject.Name;
186
187 if (finalObject != null)
188 {
189     string finalOutputUri = $"gs://{_translatedDocumentsBucket}/{finalObject.Name}";
190     Console.WriteLine($"Final translated document URI: {finalOutputUri}");
191     document.TranslatedFilePath = finalOutputUri;
192 }
193 else
194 {
195     Console.WriteLine("Translated file not found.");
196 }
197
198
199 // Save the document record to the database
200 _context.Documents.Add(document);
201 await _context.SaveChangesAsync();
202 return RedirectToAction(nameof(Index));
203

```

Figure 22: Translate Document Code

```

100
101 // POST: DiscussionBoards/Create
102 // To protect from overposting attacks, enable the specific properties you want to bind to.
103 // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
104 [HttpPost]
105 [ValidateAntiForgeryToken]
106 public async Task<ActionResult> Create([Bind("Title,Content,DiscussionType")] CreateDiscussionBoardViewModel vm)
107 {
108     var userId = _userManager.GetUserId(User);
109     var user = await _context.Users.FindAsync(userId);
110
111     if (ModelState.IsValid)
112     {
113         var discussionBoard = new DiscussionBoard
114         {
115             Title = vm.Title,
116             Content = vm.Content,
117             DiscussionType = vm.DiscussionType,
118             UserId = userId,
119             User = user,
120             CreatedOn = DateTime.Now
121         };
122
123         _context.Add(discussionBoard);
124         await _context.SaveChangesAsync();
125         return RedirectToAction(nameof(Index));
126     }
127
128     // If validation fails, repopulate the SelectList so the dropdown gets rendered
129     vm.DiscussionTypes = new SelectList(Enum.GetValues(typeof(DiscussionType)).Cast<DiscussionType>());
130
131     ViewData["UserId"] = new SelectList(_context.Users, "Id", "Id", userId);
132
133     return View(vm);
134 }

```

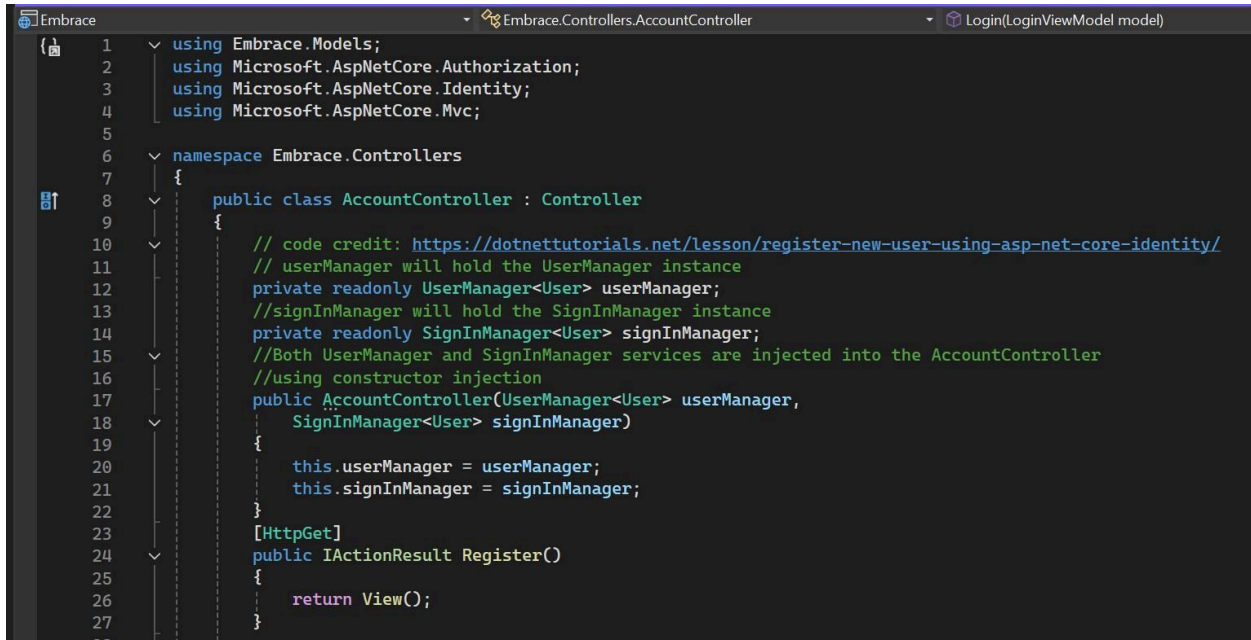
Figure 23: Create Discussion Board Method

```

24
25 // GET: DiscussionBoards
26 public async Task<ActionResult> Index(DiscussionType? discussionType, string? searchString)
27 {
28     var discussionBoardQuery = _context.DiscussionBoards
29         .Include(d => d.User)
30         .Include(d => d.Comments)
31         .AsQueryable();
32
33     // Filter by search term
34     if (!string.IsNullOrEmpty(searchString))
35     {
36         discussionBoardQuery = discussionBoardQuery.Where(x => x.Title!.ToUpper().Contains(searchString.ToUpper()) ||
37             x.Content!.ToUpper().Contains(searchString.ToUpper()));
38     }
39
40     // Filter by discussion type
41     if (discussionType != null)
42     {
43         discussionBoardQuery = discussionBoardQuery.Where(x => x.DiscussionType == discussionType);
44     }
45
46     var discussionBoards = await discussionBoardQuery.ToListAsync();
47
48     // Use LINQ to get list of discussion types + tags
49     IQueryable<DiscussionType> discussionTypeQuery = from r in _context.DiscussionBoards
50         orderby r.DiscussionType
51         select r.DiscussionType;
52
53     var discussionBoardVM = new DiscussionBoardViewModel
54     {
55         DiscussionTypes = new SelectList(await discussionTypeQuery.Distinct().ToListAsync()),
56         DiscussionBoards = discussionBoards
57     };
58
59     return View(discussionBoardVM);
60 }
61

```

Figure 24: Discussion Board Index Method

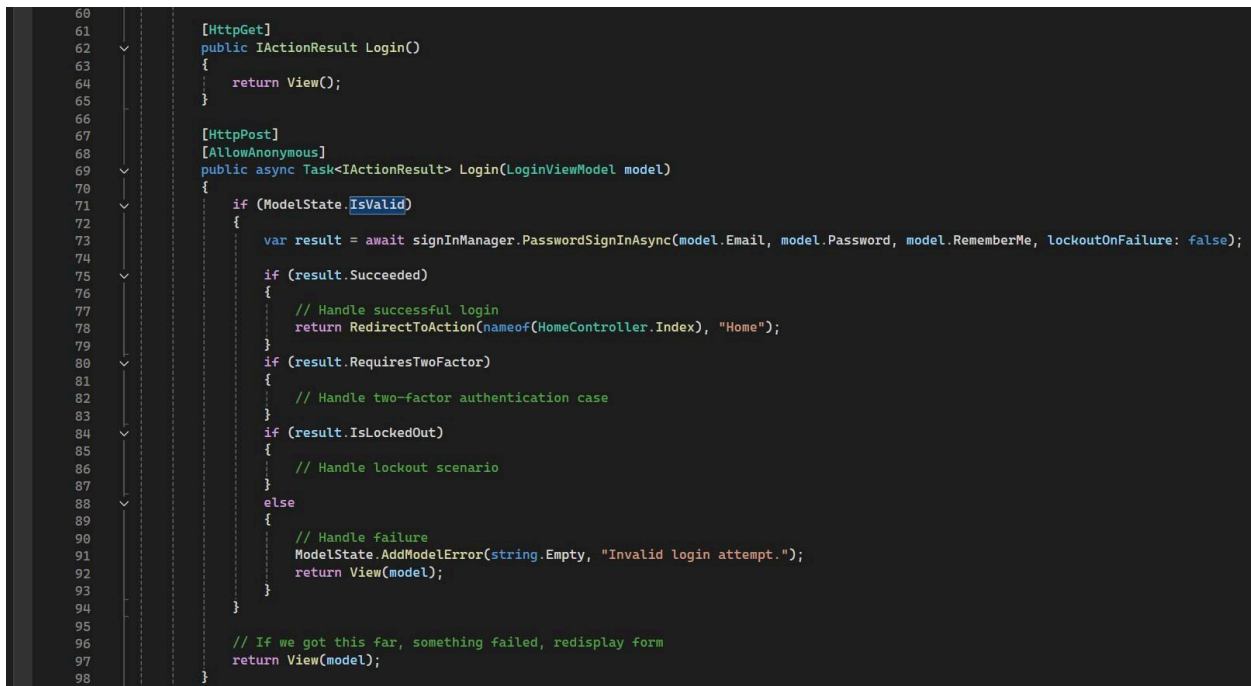


```

1  using Embrace.Models;
2  using Microsoft.AspNetCore.Authorization;
3  using Microsoft.AspNetCore.Identity;
4  using Microsoft.AspNetCore.Mvc;
5
6  namespace Embrace.Controllers
7  {
8      public class AccountController : Controller
9      {
10         // code credit: https://dotnettutorials.net/lesson/register-new-user-using-asp-net-core-identity/
11         // userManager will hold the UserManager instance
12         private readonly UserManager<User> userManager;
13         //signInManager will hold the SignInManager instance
14         private readonly SignInManager<User> signInManager;
15         //Both UserManager and SignInManager services are injected into the AccountController
16         //using constructor injection
17         public AccountController(UserManager<User> userManager,
18             SignInManager<User> signInManager)
19         {
20             this.userManager = userManager;
21             this.signInManager = signInManager;
22         }
23         [HttpGet]
24         public IActionResult Register()
25         {
26             return View();
27         }
28     }

```

Figure 25: Account Controller Setup



```

60
61 [HttpGet]
62 public IActionResult Login()
63 {
64     return View();
65 }
66
67 [HttpPost]
68 [AllowAnonymous]
69 public async Task<IActionResult> Login(LoginViewModel model)
70 {
71     if (ModelState.IsValid)
72     {
73         var result = await signInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, lockoutOnFailure: false);
74
75         if (result.Succeeded)
76         {
77             // Handle successful login
78             return RedirectToAction(nameof(HomeController.Index), "Home");
79         }
80         if (result.RequiresTwoFactor)
81         {
82             // Handle two-factor authentication case
83         }
84         if (result.IsLockedOut)
85         {
86             // Handle lockout scenario
87         }
88         else
89         {
90             // Handle failure
91             ModelState.AddModelError(string.Empty, "Invalid login attempt.");
92             return View(model);
93         }
94     }
95
96     // If we got this far, something failed, redisplay form
97     return View(model);
98 }

```

Figure 26: Login Code



```
28
29 [HttpPost]
30 public async Task<IActionResult> Register(RegisterViewModel model)
31 {
32     if (ModelState.IsValid)
33     {
34         // Copy data from RegisterViewModel to IdentityUser
35         var user = new User
36         {
37             UserName = model.Email,
38             FirstName = model.FirstName,
39             LastName = model.LastName,
40             Email = model.Email
41         };
42         // Store user data in AspNetUsers database table
43         var result = await userManager.CreateAsync(user, model.Password);
44         // If user is successfully created, sign-in the user using
45         // SignInManager and redirect to index action of HomeController
46         if (result.Succeeded)
47         {
48             await signInManager.SignInAsync(user, isPersistent: false);
49             return RedirectToAction("Index", "home");
50         }
51         // If there are any errors, add them to the ModelState object
52         // which will be displayed by the validation summary tag helper
53         foreach (var error in result.Errors)
54         {
55             ModelState.AddModelError(string.Empty, error.Description);
56         }
57     }
58     return View(model);
59 }
```

Figure 27: Register Code

```
99
100 // Logs out user and redirects them to the home page
101 [HttpPost]
102 public async Task<IActionResult> Logout()
103 {
104     await signInManager.SignOutAsync();
105     return RedirectToAction("index", "home");
106 }
107
108 }
```

Figure 28: Logout Code

## HELPERS AND ENUMS

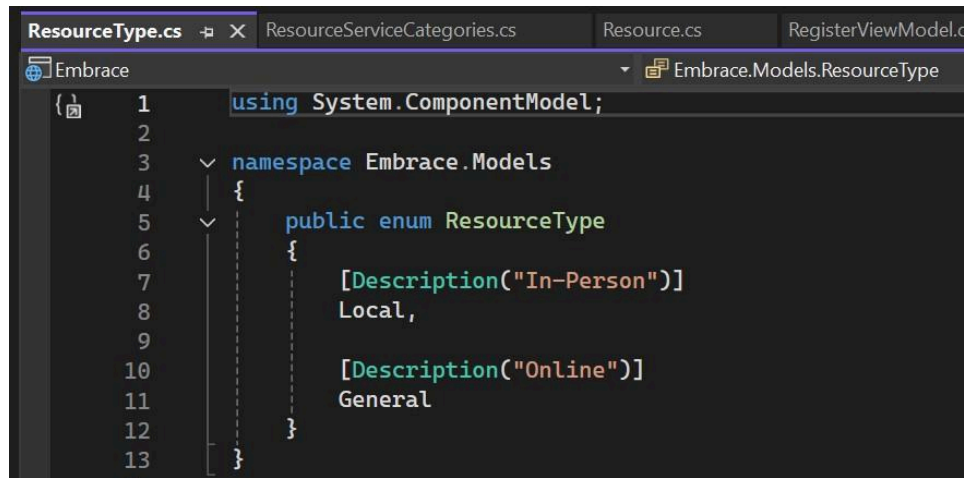


Figure 29: Resource Type Enum

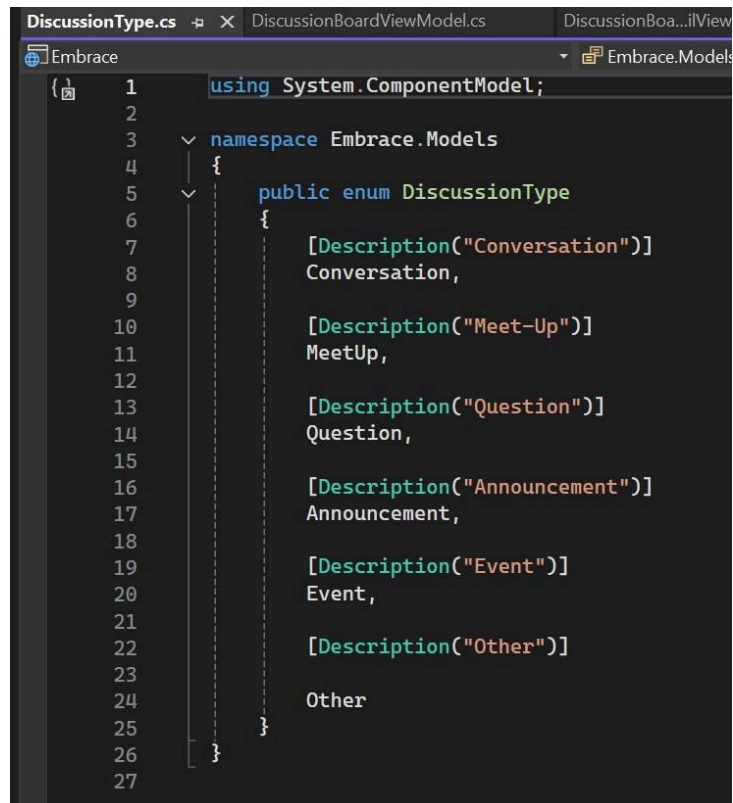
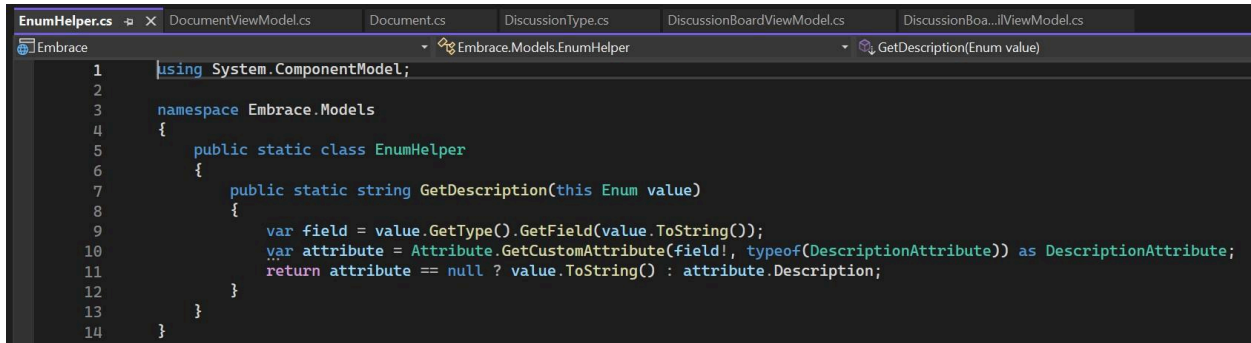


Figure 30: Discussion Type Enum



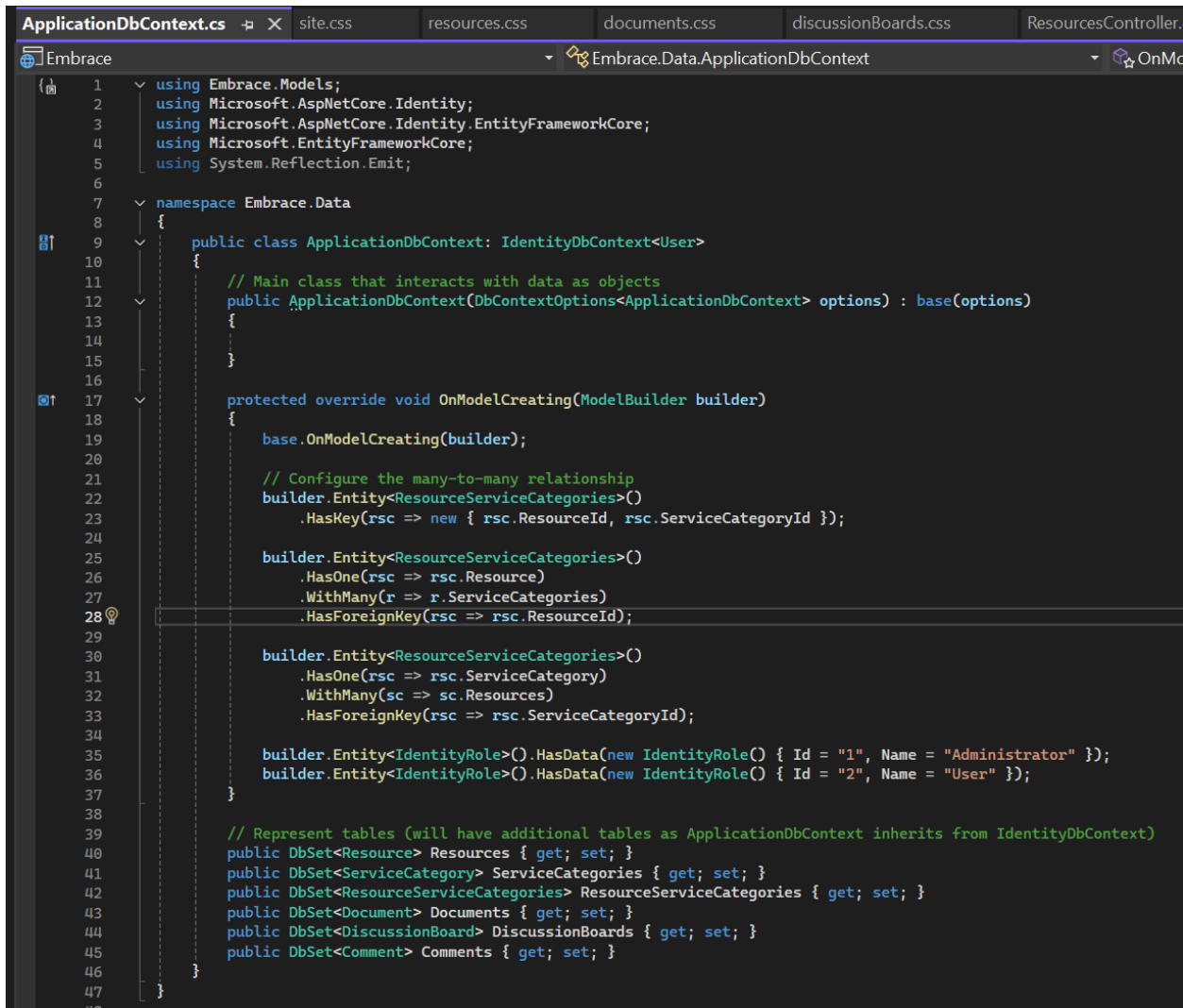
```

1  using System.ComponentModel;
2
3  namespace Embrace.Models
4  {
5      public static class EnumHelper
6      {
7          public static string GetDescription(this Enum value)
8          {
9              var field = value.GetType().GetField(value.ToString());
10             var attribute = Attribute.GetCustomAttribute(field!, typeof(DescriptionAttribute)) as DescriptionAttribute;
11             return attribute == null ? value.ToString() : attribute.Description;
12         }
13     }
14 }

```

Figure 31: Get Description Helper Method for Enums

## DATA



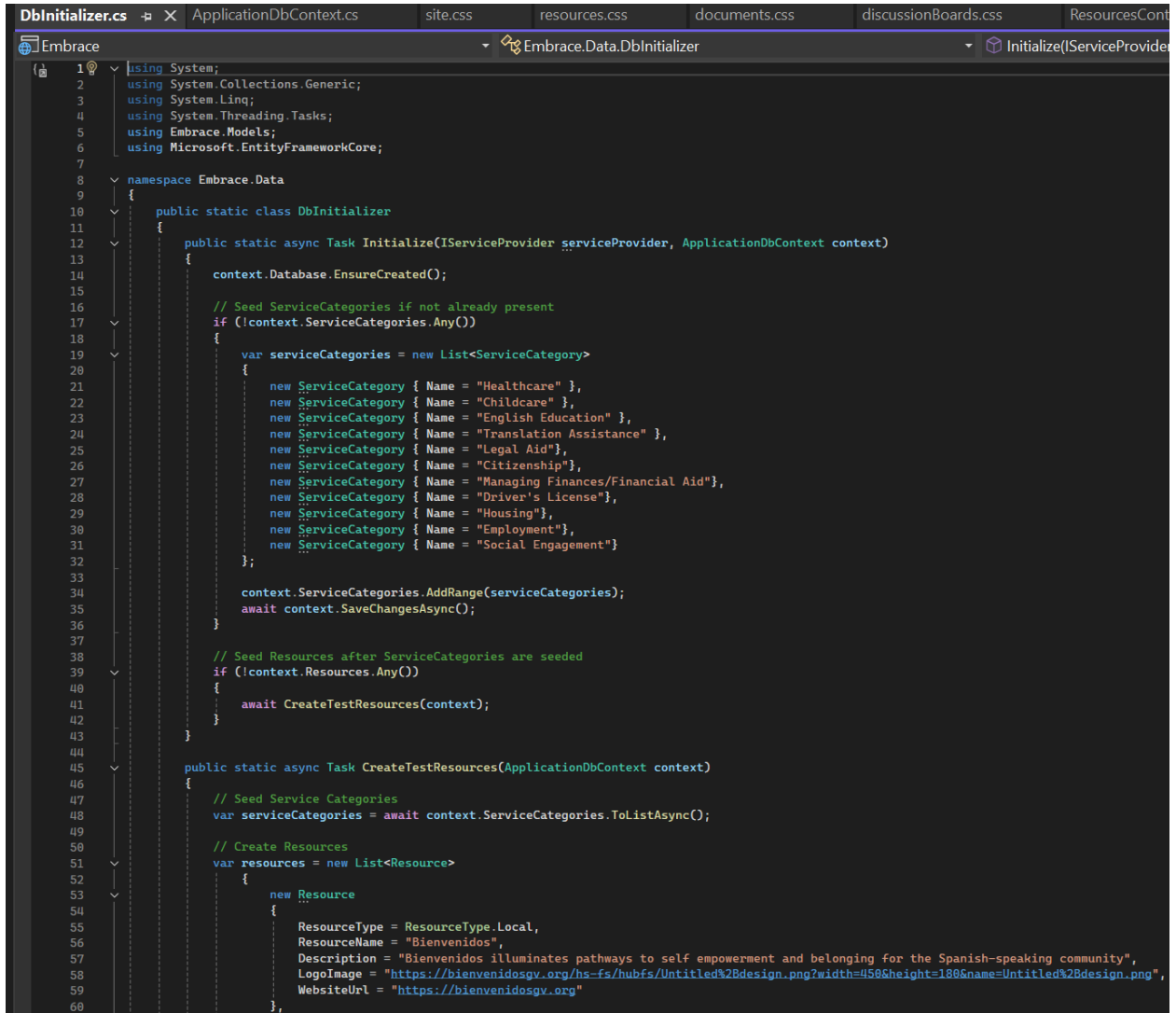
```

1  using Embrace.Models;
2  using Microsoft.AspNetCore.Identity;
3  using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
4  using Microsoft.EntityFrameworkCore;
5  using System.Reflection.Emit;
6
7  namespace Embrace.Data
8  {
9      public class ApplicationDbContext : IdentityDbContext<User>
10     {
11         // Main class that interacts with data as objects
12         public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options)
13         {
14         }
15
16         protected override void OnModelCreating(ModelBuilder builder)
17         {
18             base.OnModelCreating(builder);
19
20             // Configure the many-to-many relationship
21             builder.Entity<ResourceServiceCategories>()
22                 .HasKey(rsc => new { rsc.ResourceId, rsc.ServiceCategoryId });
23
24             builder.Entity<ResourceServiceCategories>()
25                 .HasOne(rsc => rsc.Resource)
26                 .WithMany(r => r.ServiceCategories)
27                 .HasForeignKey(rsc => rsc.ResourceId);
28
29             builder.Entity<ResourceServiceCategories>()
30                 .HasOne(rsc => rsc.ServiceCategory)
31                 .WithMany(sc => sc.Resources)
32                 .HasForeignKey(rsc => rsc.ServiceCategoryId);
33
34             builder.Entity<IdentityRole>().HasData(new IdentityRole() { Id = "1", Name = "Administrator" });
35             builder.Entity<IdentityRole>().HasData(new IdentityRole() { Id = "2", Name = "User" });
36
37         }
38
39         // Represent tables (will have additional tables as ApplicationDbContext inherits from IdentityDbContext)
40         public DbSet<Resource> Resources { get; set; }
41         public DbSet<ServiceCategory> ServiceCategories { get; set; }
42         public DbSet<ResourceServiceCategories> ResourceServiceCategories { get; set; }
43         public DbSet<Document> Documents { get; set; }
44         public DbSet<DiscussionBoard> DiscussionBoards { get; set; }
45         public DbSet<Comment> Comments { get; set; }
46     }
47 }

```

Figure 32: ApplicationDbContext File





```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using Embrace.Models;
6  using Microsoft.EntityFrameworkCore;
7
8  namespace Embrace.Data
9  {
10     public static class DbInitializer
11     {
12         public static async Task Initialize(IServiceProvider serviceProvider, ApplicationDbContext context)
13         {
14             context.Database.EnsureCreated();
15
16             // Seed ServiceCategories if not already present
17             if (!context.ServiceCategories.Any())
18             {
19                 var serviceCategories = new List<ServiceCategory>
20                 {
21                     new ServiceCategory { Name = "Healthcare" },
22                     new ServiceCategory { Name = "Childcare" },
23                     new ServiceCategory { Name = "English Education" },
24                     new ServiceCategory { Name = "Translation Assistance" },
25                     new ServiceCategory { Name = "Legal Aid" },
26                     new ServiceCategory { Name = "Citizenship" },
27                     new ServiceCategory { Name = "Managing Finances/Financial Aid" },
28                     new ServiceCategory { Name = "Driver's License" },
29                     new ServiceCategory { Name = "Housing" },
30                     new ServiceCategory { Name = "Employment" },
31                     new ServiceCategory { Name = "Social Engagement" }
32                 };
33
34                 context.ServiceCategories.AddRange(serviceCategories);
35                 await context.SaveChangesAsync();
36             }
37
38             // Seed Resources after ServiceCategories are seeded
39             if (!context.Resources.Any())
40             {
41                 await CreateTestResources(context);
42             }
43         }
44
45         public static async Task CreateTestResources(ApplicationDbContext context)
46         {
47             // Seed Service Categories
48             var serviceCategories = await context.ServiceCategories.ToListAsync();
49
50             // Create Resources
51             var resources = new List<Resource>
52             {
53                 new Resource
54                 {
55                     ResourceType = ResourceType.Local,
56                     ResourceName = "Bienvenidos",
57                     Description = "Bienvenidos illuminates pathways to self empowerment and belonging for the Spanish-speaking community",
58                     LogoImage = "https://bienvenidosgv.org/hs-fs/hubfs/Untitled%2Bdesign.png?width=458&height=188&name=Untitled%2Bdesign.png",
59                     WebsiteUrl = "https://bienvenidosgv.org"
60                 },

```

Figure 33: DbInitializer Part 1

```

61
62
63     new Resource
64     {
65         ResourceType = ResourceType.Local,
66         ResourceName = "Proyecto Salud",
67         Description = "An interdisciplinary team of researchers, students, and community members, seeks to reduce health disparities, improve health outcomes, and make a s
68         LogoImage = "https://www.montana.edu/nursing/salud/images/New%28Logo%28from%28Cass.png",
69         WebsiteUrl = "https://www.montana.edu/nursing/salud/"
70     },
71     new Resource
72     {
73         ResourceType = ResourceType.General,
74         ResourceName = "Immigrant Legal Resource Center",
75         Description = "Providing legal resources and education for immigrants",
76         LogoImage = "https://i.pinimg.com/originals/b4/44/fa/b444fa6d1d9clad1442878458c3cec45.png",
77         WebsiteUrl = "https://ilrc.org"
78     },
79     new Resource
80     {
81         ResourceType = ResourceType.General,
82         ResourceName = "U.S. Citizenship and Immigration Services",
83         Description = "USCIS offers helpful information about education, child care, employment, what to do in case of an emergency, and a number of popular topics that wa
84         LogoImage = "https://www.uscis.gov/sites/default/files/USCIS_Signature_PREFERRED_EC.png",
85         WebsiteUrl = "https://www.uscis.gov/citizenship/civic-integration/settling-in-the-us"
86     }
87 ];
88
89 context.Resources.AddRange(resources);
90 await context.SaveChangesAsync(); // Save resources to generate IDs
91
92 var resourceServiceCategories = new List<ResourceServiceCategories>
93 {
94     new ResourceServiceCategories
95     {
96         Resource = resources[0],
97         ServiceCategory = serviceCategories.First(sc => sc.Name == "Social Engagement")
98     },
99     new ResourceServiceCategories
100     {
101         Resource = resources[0],
102         ServiceCategory = serviceCategories.First(sc => sc.Name == "Translation Assistance")
103     },
104     new ResourceServiceCategories
105     {
106         Resource = resources[1],
107         ServiceCategory = serviceCategories.First(sc => sc.Name == "Healthcare")
108     },
109     new ResourceServiceCategories
110     {
111         Resource = resources[2],
112         ServiceCategory = serviceCategories.First(sc => sc.Name == "Legal Aid")
113     },
114     new ResourceServiceCategories
115     {
116         Resource = resources[2],
117         ServiceCategory = serviceCategories.First(sc => sc.Name == "Citizenship")
118     },
119     new ResourceServiceCategories
120 
```

Figure 34: DbInitializer Part 2

```

106     },
107     new ResourceServiceCategories
108     {
109         Resource = resources[2],
110         ServiceCategory = serviceCategories.First(sc => sc.Name == "Legal Aid")
111     },
112     new ResourceServiceCategories
113     {
114         Resource = resources[2],
115         ServiceCategory = serviceCategories.First(sc => sc.Name == "Citizenship")
116     },
117     new ResourceServiceCategories
118     {
119         Resource = resources[3],
120         ServiceCategory = serviceCategories.First(sc => sc.Name == "English Education")
121     },
122     new ResourceServiceCategories
123     {
124         Resource = resources[3],
125         ServiceCategory = serviceCategories.First(sc => sc.Name == "Childcare")
126     },
127     new ResourceServiceCategories
128     {
129         Resource = resources[3],
130         ServiceCategory = serviceCategories.First(sc => sc.Name == "Managing Finances/Financial Aid")
131     },
132     new ResourceServiceCategories
133     {
134         Resource = resources[3],
135         ServiceCategory = serviceCategories.First(sc => sc.Name == "Healthcare")
136     }
137 };
138
139 context.ResourceServiceCategories.AddRange(resourceServiceCategories);
140 await context.SaveChangesAsync();
141 }
142
143 }
144
145

```

Figure 35: DbInitializer Part 3

## VIEWS

```

Index.cshtml*  X  Index.cshtml  Register.cshtml  Login.cshtml  DbInitializer.cs  ApplicationDbContext.cs  site.css  resources.css  documents.css
Embrace
@{
    ViewData["Title"] = "Home Page";
}

<!-- Welcome Section -->
<section class="container text-center my-5">
    <div class="row align-items-center">
        <div class="col-md-6 text-md-start">
            <h1>Welcome!</h1>
            <p class="lead">
                Whether you're a first- or second-generation immigrant or come from a multilingual background,
                this can sometimes make it difficult to complete daily tasks or find people in your area that you can connect with. <br /><br />
                Embrace is here to help you navigate life in the U.S., offering all the information and tools you need to find the support you need,
                get answers to your questions, and connect with your community. Look below to learn more about what we have to offer.
            </p>
        </div>
        <div class="col-md-6 text-center">
            
        </div>
    </div>
</section>

<!-- Tile Section for Services Offered -->
<section class="container my-5">
    <h2 class="section-title">What We Offer</h2>
    <div class="row">
        <div class="col-md-4">
            <a class="card shadow-sm asp-area="" asp-controller="Resources" asp-action="Index">
                
                <div class="card-body">
                    <h3 class="card-title">Resource Hub</h3>
                    <p class="card-text">A collection of location-based and general resources that can assist you in a variety of categories such as
                        legal assistance, immigration support, housing, financial aid, healthcare, and more.</p>
                </div>
            </a>
        </div>
        <div class="col-md-4">
            <a class="card shadow-sm asp-area="" asp-controller="Documents" asp-action="Index">
                
                <div class="card-body">
                    <h3 class="card-title">Document Translation</h3>
                    <p class="card-text">A free service that allows you to safely upload PDFs or images and receive a translated version in your preferred language.</p>
                </div>
            </a>
        </div>
        <div class="col-md-4">
            <a class="card shadow-sm asp-area="" asp-controller="DiscussionBoards" asp-action="Index">
                
                <div class="card-body">
                    <h3 class="card-title">Connection Corner</h3>
                    <p class="card-text">A page where you can find people to meet up with in your community, ask questions, or simply start a conversation!</p>
                </div>
            </a>
        </div>
    </div>
</section>

```

Figure 36: Home Page

```

Embrace
using Microsoft.AspNetCore.Identity
@model Embrace.Models.ResourceViewModel
@inject SignInManager<User> SignInManager

@{
    ViewData["Title"] = "Resources";
}

<link rel="stylesheet" href="/css/resources.css" />
<h1>Resources</h1>
<!-- TO-DO: maybe update this page to use a carousel at the top to navigate through the different offerings and their links -->

<p class="resource-list-description">
    Use this page to search for resources that can help you in a variety of different categories. You can filter by the type of service offered, whether it is an in-person or online resource, and so on.
</p>
<div class="fw-bold">Definitions</div>
<p>In-Person - A resource that has an in-person location.</p>
<p>Online - A resource that can only be accessed online (although it might point you to more in-person resources on the website).</p>

<div class="container-text">
    <form asp-controller="Resources" asp-action="Index" method="get" class="search-container">
        @if (SignInManager.IsSignedIn(User) && User.IsInRole("Admin"))
        {
            <p>
                <a class="btn btn-create" asp-action="Create">Add a resource</a>
            </p>
        }
        <div class="d-flex">
            <select asp-for="ResourceType" asp-items="Model.ResourceTypes" class="form-select">
                <option value="" selected disabled>Select a resource type</option>
                <option value="">All</option>
            </select>
            <select asp-for="ServiceCategoryId" asp-items="Model.ServiceCategories" class="form-select">
                <option value="" selected disabled>Select a service category</option>
                <option value="">All</option>
            </select>
            <input type="text" name="SearchString" placeholder="Enter a searchword" class="form-control" />
            <input type="text" name="ZipSearchString" placeholder="Enter a zipcode" class="form-control" />
            <button type="submit" class="btn btn-search">Search</button>
        </div>
    </form>
</div>

```

Figure 37: Resource Main Page Part 1

```

<div style="overflow-x:auto;">
  <table class="table table-borderless table-sm">
    <thead>
      <tr class="resource-header">
        <th>
          <h4>Resource</h4>
        </th>
        <th>
          <h4>Type</h4>
        </th>
        <th>
          <h4>Service Categories</h4>
        </th>
      </tr>
    </thead>
    <tbody>
      @foreach (var item in Model.Resources!)
      {
        <tr class="resource-row">
          <td>
            <div class="resource-image">
              @if (!string.IsNullOrEmpty(item.LocationImage))
              {
                
              }
              else if (!string.IsNullOrEmpty(item.LogoImage))
              {
                
              }
              else
              {
                
              }
            </div>
          </td>
          <td>
            @Html.DisplayFor(modelItem => item.ResourceName)
          </td>
          <td>
            @Html.DisplayFor(modelItem => item.ResourceType)
          </td>
          <td>
            <ul class="service-categories">
              @foreach (var category in item.ServiceCategories)
              {
                <li class="checkmark-bullet">
                  @category.ServiceCategory.Name
                </li>
              }
            </ul>
          </td>
          <td>
            <div class="d-flex justify-content-between">
              <a class="btn btn-details resource-button" asp-action="Details" asp-route-id=@item.Id>Details</a>
              @if (SignInManager.IsSignedIn(User) && User.IsInRole("Admin"))
              {

```

Figure 38: Resource Main Page Part 2

```

            <div class="d-flex justify-content-between">
              <a class="btn btn-details resource-button" asp-action="Details" asp-route-id=@item.Id>Details</a>
              @if (SignInManager.IsSignedIn(User) && User.IsInRole("Admin"))
              {
                <a class="btn btn-success resource-button" asp-action="Edit" asp-route-id=@item.Id>Edit</a>
                <a class="btn btn-delete resource-button" asp-action="Delete" asp-route-id=@item.Id>Delete</a>
              }
              <a class="btn btn-save resource-button" asp-action="Save" asp-route-id=@item.Id>Save</a>
            </div>
          </td>
        </tr>
      }
    </tbody>
  </table>
</div>

```

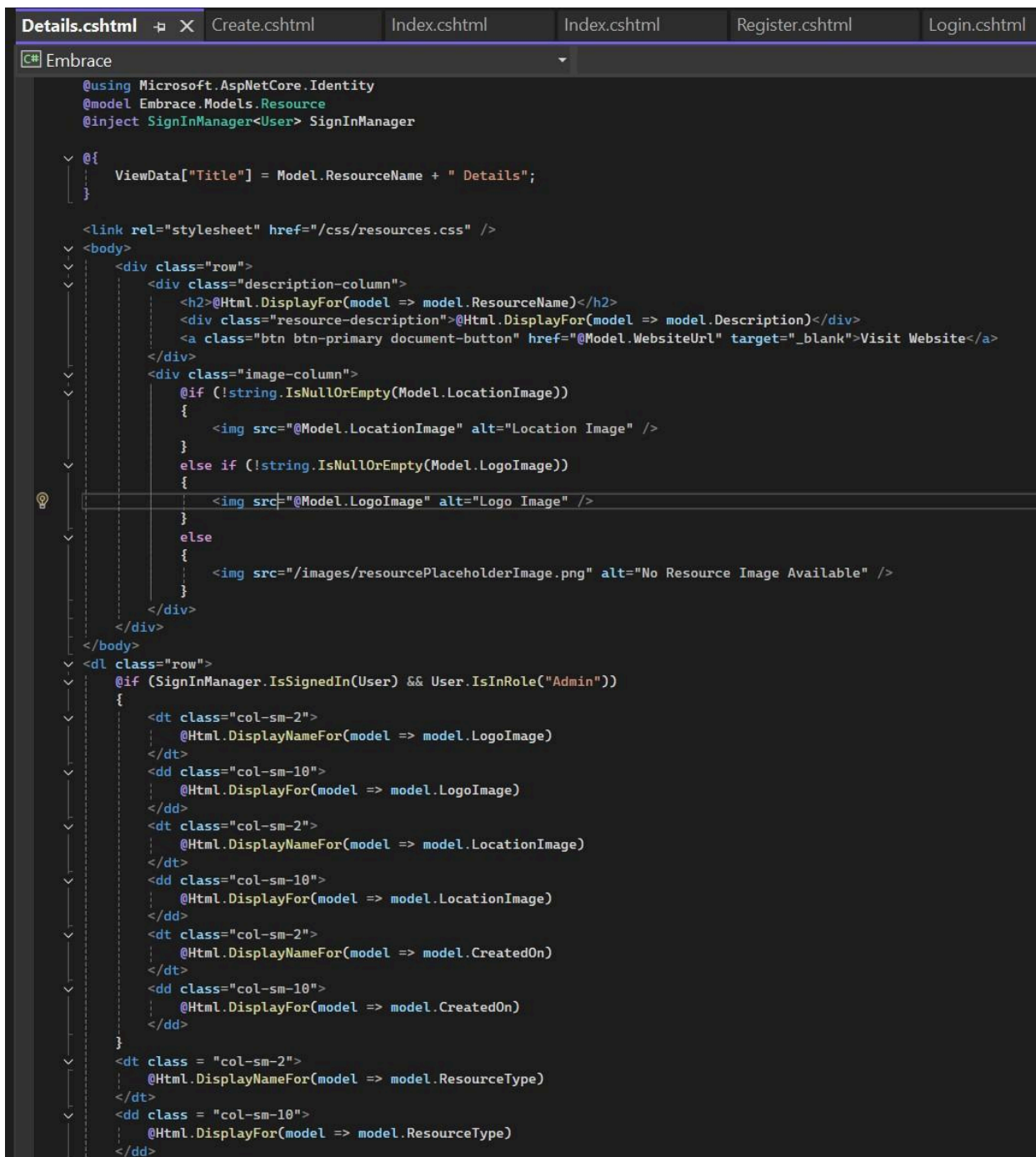
Figure 39: Resource Main Page Part 3

```

<dt class = "col-sm-2">
    @Html.DisplayNameFor(model => model.ServiceCategories)
</dt>
<dd class = "col-sm-10">
    @if (Model.ServiceCategories != null && Model.ServiceCategories.Any())
    {
        <ul class="service-categories">
            @foreach (var category in Model.ServiceCategories)
            {
                <li class="checkmark-bullet">
                    @category.ServiceCategory?.Name
                </li>
            }
        </ul>
    }
    else
    {
        <p>No service categories listed for @Model.ResourceName.</p>
    }
</dd>
@if (Model.ResourceType.Equals(ResourceType.Local))
{
    @if (!Model.Address.Equals(""))
    {
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Address)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.Address)
        </dd>
    }
    else
    {
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.Address)
        </dt>
        <dd class="col-sm-10">
            <p>No address currently exists for this resource. Consider visiting the organization website to learn more.</p>
        </dd>
    }
}
<dt class = "col-sm-2">
    @Html.DisplayNameFor(model => model.PhoneNumber)
</dt>
<dd class = "col-sm-10">
    @if (!Model.PhoneNumber.Equals(""))
    {
        @Html.DisplayFor(model => model.PhoneNumber)
    }
    else
    {
        <p>Phone number is not currently available for @Model.ResourceName. Consider visiting the organization website to learn more.</p>
    }
</dd>
</dl>

```

Figure 40: Resource Detail Page Part 1



```

Details.cshtml  Create.cshtml  Index.cshtml  Index.cshtml  Register.cshtml  Login.cshtml
Embrace
@using Microsoft.AspNetCore.Identity
@model Embrace.Models.Resource
@inject SignInManager<User> SignInManager

@{
    ViewData["Title"] = Model.ResourceName + " Details";
}

<link rel="stylesheet" href="/css/resources.css" />
<body>
    <div class="row">
        <div class="description-column">
            <h2>@Html.DisplayFor(model => model.ResourceName)</h2>
            <div class="resource-description">@Html.DisplayFor(model => model.Description)</div>
            <a class="btn btn-primary document-button" href="@Model.WebsiteUrl" target="_blank">Visit Website</a>
        </div>
        <div class="image-column">
            @if (!string.IsNullOrEmpty(Model.LocationImage))
            {
                
            }
            else if (!string.IsNullOrEmpty(Model.LogoImage))
            {
                
            }
            else
            {
                
            }
        </div>
    </div>
</body>
<dl class="row">
    @if (SignInManager.IsSignedIn(User) && User.IsInRole("Admin"))
    {
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.LogoImage)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.LogoImage)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.LocationImage)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.LocationImage)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.CreatedOn)
        </dt>
        <dd class="col-sm-10">
            @Html.DisplayFor(model => model.CreatedOn)
        </dd>
    }
    <dt class="col-sm-2">
        @Html.DisplayNameFor(model => model.ResourceType)
    </dt>
    <dd class="col-sm-10">
        @Html.DisplayFor(model => model.ResourceType)
    </dd>
</dl>

```

Figure 41: Resource Detail Page Part 2

```

    }
  }
</dd>
</dl>
<div>
  @if (SignInManager.IsSignedIn(User) && User.IsInRole("Admin"))
  {
    <a asp-action="Edit" asp-route-id="@Model?.Id">Edit</a>
  }
  <a asp-action="Index">Return</a>
</div>

```

Figure 42: Resource Detail Page Part 3

```

@if (SignInManager.IsSignedIn(User))
{
  <h1>My Documents</h1>
  <div class="row">
    <div class="col-md-6 doc-upload-left">
      <p class="doc-upload-description">
        Having a hard time translating something? Upload your document here, select your preferred language, and receive a translated version back!
      </p>
      <a class="btn btn-create" asp-action="Create">Upload a New Document</a>
    </div>
    <div style="overflow-x:auto;" class="col-md-6">
      <table class="table table-borderless table-sm">
        <tbody>
          <!-- check if user signed in, also if no documents then display text to say that-->
          @foreach (var item in Model.Documents)
          {
            <tr class="document-row">
              <td>
                <div class="document-image">
                  
                </div>
              </td>
              <td>
                @item.Title
              </td>
              <td>
                @item.CreatedOn.Date.ToShortDateString()
              </td>
              <td>
                <div class="d-flex flex-column justify-content-between gap-2">
                  @if (SignInManager.IsSignedIn(User) && User.IsInRole("Admin"))
                  {
                    <a class="btn btn-details document-button" asp-action="Details" asp-route-id=@item.Id>Details</a>
                    <a class="btn btn-primary document-button" asp-action="ViewDocument" asp-route-id=@item.Id>View</a>
                    <a class="btn btn-delete document-button" asp-action="Delete" asp-route-id=@item.Id>Delete</a>
                  }
                </div>
              </td>
            </tr>
          }
        </tbody>
      </table>
    </div>
  </div>
}
else
{
  <h1>Document Translation</h1>
  <div class="row">
    <div class="col-md-6 doc-upload-left">
      <p class="doc-upload-description">
        Having a hard time translating something? Please sign in then upload your document here, select your preferred language, and receive a translated version back!
      </p>
    </div>
  </div>
}

```

Figure 43: Document Translation Main Page



```
@using Microsoft.AspNetCore.Identity
@model Embrace.Models.DiscussionBoardViewModel
@inject SignInManager<User> SignInManager

@{
    ViewData["Title"] = "Connection Corner";
}

<h1>Connection Corner</h1>

<link rel="stylesheet" href="/css/discussionBoards.css" />

<p class="connection-corner-description">Use this page to start a conversation, ask a question, find people to meet up with in your community, and more! Either start your own chat board or comment on

<form asp-controller="DiscussionBoards" asp-action="Index" method="get" class="search-container">
    <div class="d-flex">
        @if (SignInManager.IsSignedIn(User))
        {
            <a class="btn btn-create" asp-action="Create">New Chat</a>

            <select asp-for="DiscussionType" asp-items="Model.DiscussionTypes" class="form-select">
                <option value="" selected disabled>Select a discussion type</option>
                <option value="">All</option>
            </select>
            <input type="text" name="SearchString" placeholder="Enter a searchword" class="form-control" />

            <button type="submit" class="btn btn-search">Search</button>
        }
    </div>
</form>

<section class="centered">
    @if (Model.DiscussionBoards!.Count == 0)
    {
        @if (SignInManager.IsSignedIn(User))
        {
            <p class="no-discussion-boards">There are currently no conversations! Click the "New Chat" button above to start your own.</p>
        }
        else
        {
            <p class="no-discussion-boards">There are currently no conversations! Please register or login to start your own.</p>
        }
    }
    else
    {
        <div class="cards">
            @foreach (var item in Model.DiscussionBoards)
            {
                <a class="card" asp-action="Details" asp-route-id=@item.Id>
                    <div class="card-header d-flex">
                        <div class="header-left">
                            <h5 class="card-title">@item.Title</h5>
                        </div>
                        <div class="header-right">
                            <div class="discussion-tag @GetTagClass(@item.DiscussionType)">@item.DiscussionType.GetDescription()</div>
                        </div>
                    </div>

                    <div class="card-body">
                        <p class="card-text">@item.Content</p>
                    </div>
                    <div class="discussion-board-card-buttons">

```

Figure 45: Discussion Board Main Page Part 1

```

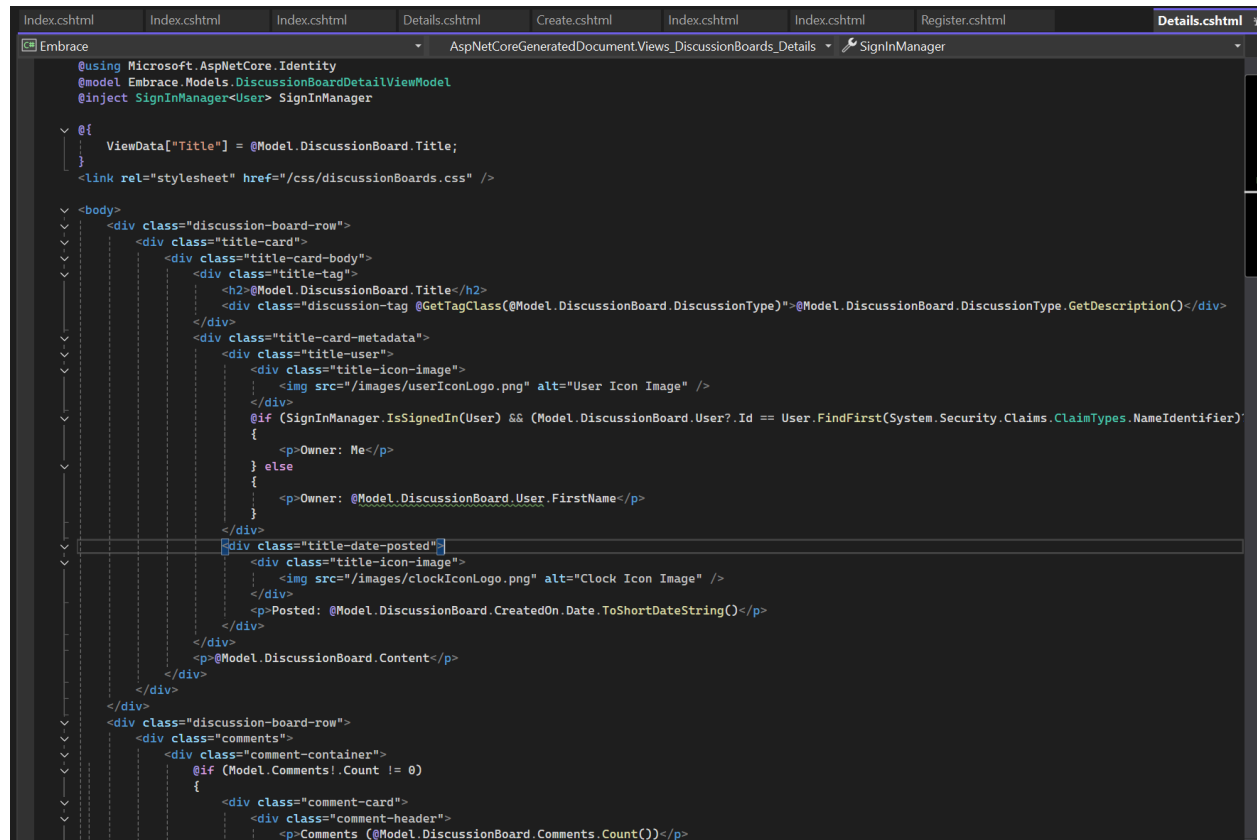
<p class="card-text">@item.Content</p>
</div>
<div class="discussion-board-card-buttons">
  @if (SignInManager.IsSignedIn(User) && User.IsInRole("Admin"))
  {
    <a class="btn btn-success resource-button" asp-action="Edit" asp-route-id="@item.Id">Edit</a>
    <a class="btn btn-delete resource-button" asp-action="Delete" asp-route-id="@item.Id">Delete</a>
  }
</div>
<div class="card-footer d-flex justify-content-between">
  <span class="card-footer-text">Comments: @item.Comments.Count</span>
  <span class="card-footer-text">@item.CreatedOn.Date.ToShortDateString()</span>
</div>
</a>
</div>
</div>
</section>

<script>
  document.addEventListener("DOMContentLoaded", function() {
    const paragraphs = document.querySelectorAll('.card-body p');
    paragraphs.forEach(function(p) {
      if (p.scrollHeight > p.clientHeight) {
        p.classList.add('fade-if-overflow');
      } else {
        p.classList.remove('fade-if-overflow');
      }
    });
  });
</script>

@functions {
  // returns a CSS class based on the discussion type for coloring purposes
  private string GetTagClass(DiscussionType discussionType)
  {
    return discussionType switch
    {
      DiscussionType.Question => "question-tag",
      DiscussionType.Conversation => "conversation-tag",
      DiscussionType.MeetUp => "meetup-tag",
      DiscussionType.Announcement => "announcement-tag",
      DiscussionType.Event => "event-tag",
      DiscussionType.Other => "other-tag",
      _ => ""
    };
  }
}

```

Figure 46: Discussion Board Main Page Part 2



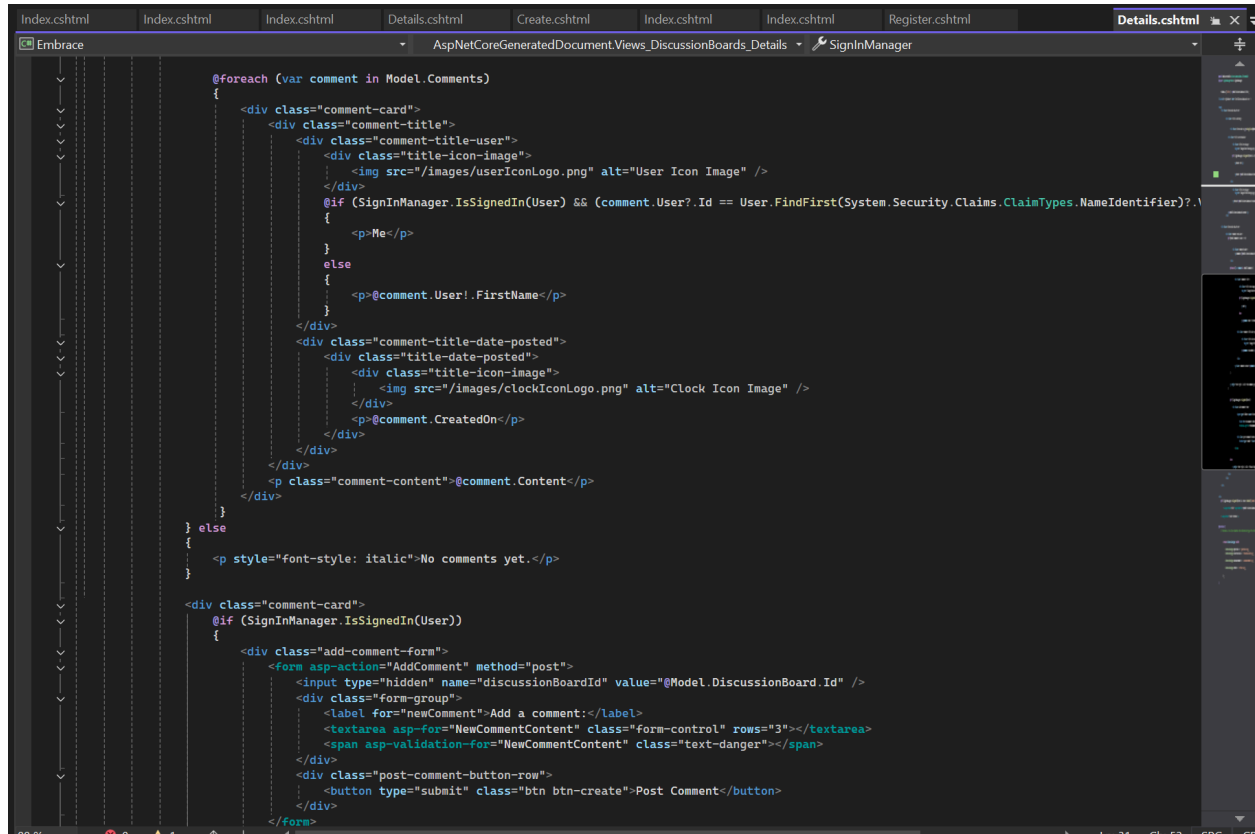
```
Index.cshtml Index.cshtml Index.cshtml Details.cshtml Create.cshtml Index.cshtml Index.cshtml Register.cshtml Details.cshtml
Embrace ASP.NETCoreGeneratedDocument.Views.DiscussionBoards.Details SignInManager

@using Microsoft.AspNetCore.Identity
@model Embrace.Models.DiscussionBoardDetailViewModel
@inject SignInManager<User> SignInManager

@{
    ViewData["Title"] = @Model.DiscussionBoard.Title;
}
<link rel="stylesheet" href="/css/discussionBoards.css" />

<body>
    <div class="discussion-board-row">
        <div class="title-card">
            <div class="title-card-body">
                <div class="title-tag">
                    <h2>@Model.DiscussionBoard.Title</h2>
                </div>
                <div class="discussion-tag @GetTagClass(@Model.DiscussionBoard.DiscussionType)">@Model.DiscussionBoard.DiscussionType.GetDescription()</div>
            </div>
            <div class="title-card-metadata">
                <div class="title-user">
                    <div class="title-icon-image">
                        
                    </div>
                    @if (SignInManager.IsSignedIn(User) && (Model.DiscussionBoard.User?.Id == User.FindFirst(System.Security.Claims.ClaimTypes.NameIdentifier)?.Id))
                    {
                        <p>Owner: Me</p>
                    }
                    else
                    {
                        <p>Owner: @Model.DiscussionBoard.User.FirstName</p>
                    }
                </div>
                <div class="title-date-posted">
                    <div class="title-icon-image">
                        
                    </div>
                    <p>Posted: @Model.DiscussionBoard.CreatedOn.Date.ToShortDateString()</p>
                </div>
            </div>
            <p>@Model.DiscussionBoard.Content</p>
        </div>
        <div class="discussion-board-row">
            <div class="comments">
                <div class="comment-container">
                    @if (Model.Comments!.Count != 0)
                    {
                        <div class="comment-card">
                            <div class="comment-header">
                                <p>Comments (@Model.DiscussionBoard.Comments.Count())</p>
                            </div>
                        </div>
                    }
                </div>
            </div>
        </div>
    </div>
</body>
```

Figure 47: Discussion Board Detail Page Part 1

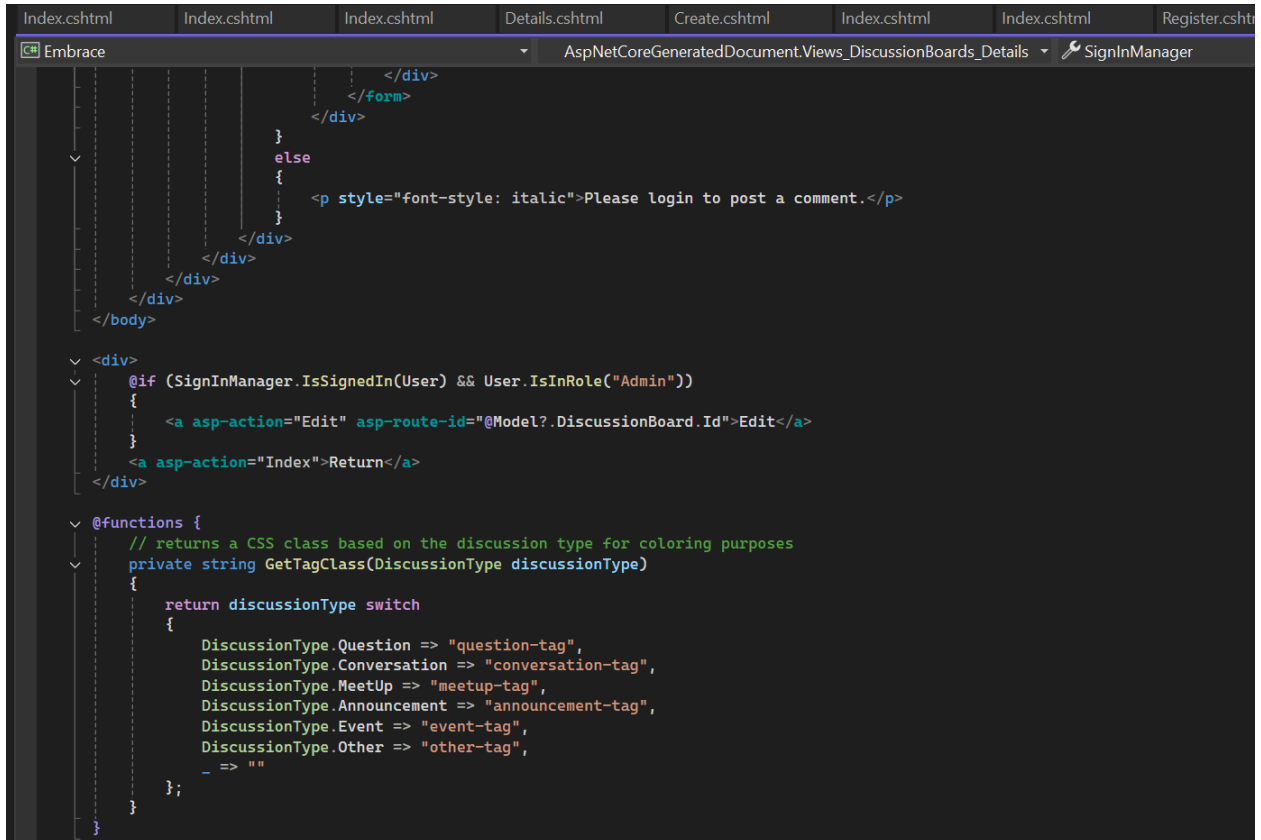


The screenshot shows a code editor with a dark theme. The top of the editor has several tabs: 'Index.cshtml', 'Index.cshtml', 'Index.cshtml', 'Details.cshtml', 'Create.cshtml', 'Index.cshtml', 'Index.cshtml', 'Register.cshtml', and 'Details.cshtml'. The active tab is 'Details.cshtml'. The code is written in C# and Razor syntax. It starts with a `@foreach (var comment in Model.Comments)` loop. Inside the loop, there's a `<div class="comment-card">` block. This block contains a `<div class="comment-title">` section with a user icon and name, and a `<div class="comment-title-date-posted">` section with a clock icon and the creation date. Below these is the comment content: `<p class="comment-content">@comment.Content</p>`. After the loop, there's an `else` block for when there are no comments: `<p style="font-style: italic;">No comments yet.</p>`. Finally, there's an `<div class="add-comment-form">` block for users who are signed in. This block contains a form with a hidden input for the discussion board ID, a text area for the new comment content, and a submit button labeled 'Post Comment'.

```
@foreach (var comment in Model.Comments)
{
    <div class="comment-card">
        <div class="comment-title">
            <div class="comment-title-user">
                <div class="title-icon-image">
                    
                </div>
                @if (SignInManager.IsSignedIn(User) && (comment.User?.Id == User.FindFirst(System.Security.Claims.ClaimTypes.NameIdentifier)?.Value))
                {
                    <p>Me</p>
                }
                else
                {
                    <p>@comment.User!.FirstName</p>
                }
            </div>
            <div class="comment-title-date-posted">
                <div class="title-date-posted">
                    <div class="title-icon-image">
                        
                    </div>
                    <p>@comment.CreatedOn</p>
                </div>
            </div>
            <p class="comment-content">@comment.Content</p>
        </div>
    }
}
else
{
    <p style="font-style: italic;">No comments yet.</p>
}

<div class="add-comment-form">
    @if (SignInManager.IsSignedIn(User))
    {
        <form asp-action="AddComment" method="post">
            <input type="hidden" name="discussionBoardId" value="@Model.DiscussionBoard.Id" />
            <div class="form-group">
                <label for="newComment">Add a comment:</label>
                <textarea asp-for="NewCommentContent" class="form-control" rows="3"></textarea>
                <span asp-validation-for="NewCommentContent" class="text-danger"></span>
            </div>
            <div class="post-comment-button-row">
                <button type="submit" class="btn btn-create">Post Comment</button>
            </div>
        </form>
    }
}
```

Figure 48: Discussion Board Detail Page Part 2



```
Index.cshtml Index.cshtml Index.cshtml Details.cshtml Create.cshtml Index.cshtml Index.cshtml Register.cshtml
Embrace ASP.NETCoreGeneratedDocument.Views_DiscussionBoards_Details SignInManager

    </div>
    </form>
    </div>
  }
  else
  {
    <p style="font-style: italic">Please login to post a comment.</p>
  }
}
</div>
</div>
</div>
</body>

<div>
  @if (SignInManager.IsSignedIn(User) && User.IsInRole("Admin"))
  {
    <a asp-action="Edit" asp-route-id="@Model?.DiscussionBoard.Id">Edit</a>
  }
  <a asp-action="Index">Return</a>
</div>

@functions {
  // returns a CSS class based on the discussion type for coloring purposes
  private string GetTagClass(DiscussionType discussionType)
  {
    return discussionType switch
    {
      DiscussionType.Question => "question-tag",
      DiscussionType.Conversation => "conversation-tag",
      DiscussionType.MeetUp => "meetup-tag",
      DiscussionType.Announcement => "announcement-tag",
      DiscussionType.Event => "event-tag",
      DiscussionType.Other => "other-tag",
      _ => ""
    };
  }
}
```

Figure 49: Discussion Board Detail Page Part 3

```

@model RegisterViewModel

@{
    ViewBag.Title = "User Registration";
}

<h1>User Registration</h1>

<div class="row create-page">
    <div class="col-md-4">
        <form method="post">
            <div asp-validation-summary="All" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="FirstName"></label>
                <input asp-for="FirstName" class="form-control" />
                <span asp-validation-for="FirstName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="LastName"></label>
                <input asp-for="LastName" class="form-control" />
                <span asp-validation-for="LastName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Email"></label>
                <input asp-for="Email" class="form-control" />
                <span asp-validation-for="Email" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Password"></label>
                <input asp-for="Password" class="form-control" />
                <span asp-validation-for="Password" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="ConfirmPassword"></label>
                <input asp-for="ConfirmPassword" class="form-control" />
                <span asp-validation-for="ConfirmPassword" class="text-danger"></span>
            </div>
            <button type="submit" class="btn btn-primary">Register</button>
        </form>
    </div>
</div>

```

Figure 50: Register Page

```

Create.cshtml  Create.cshtml  Index.cshtml  Index.cshtml  Index.cshtml  Details.cshtml
Embrace
@model LoginViewModel

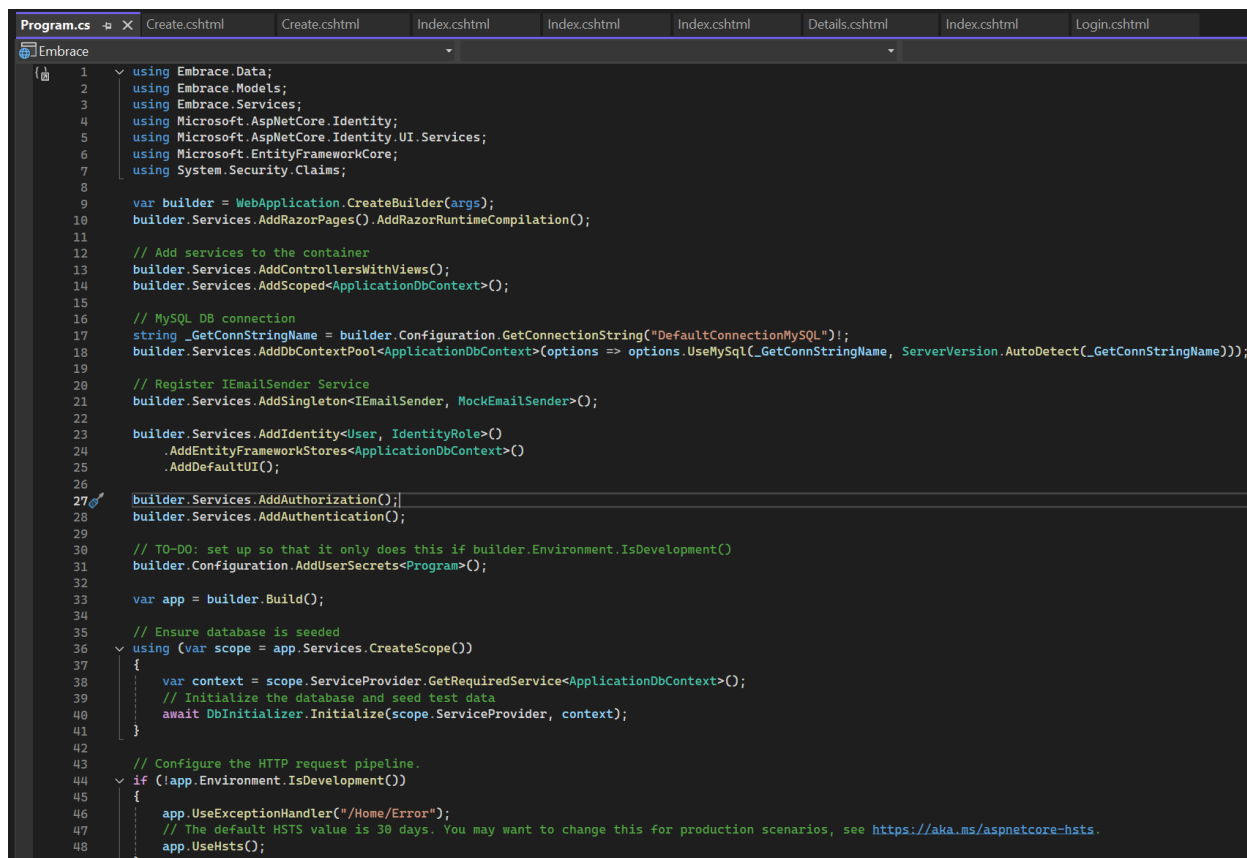
@{
    ViewBag.Title = "User Login";
}

<h1>User Login</h1>

<div class="row">
    <div class="col-md-12">
        <form method="post">
            <div asp-validation-summary="All" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Email"></label>
                <input asp-for="Email" class="form-control" />
                <span asp-validation-for="Email" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Password"></label>
                <input asp-for="Password" class="form-control" />
                <span asp-validation-for="Password" class="text-danger"></span>
            </div>
            <div class="form-group">
                <div class="checkbox">
                    <label asp-for="RememberMe">
                        <input asp-for="RememberMe" />
                        @Html.DisplayNameFor(m => m.RememberMe)
                    </label>
                </div>
            </div>
            <button type="submit" class="btn btn-primary">Login</button>
        </form>
    </div>
</div>

```

Figure 51: Login Page

A screenshot of a Visual Studio code editor showing the first part of a C# program file named Program.cs. The file is open in a tab labeled 'Program.cs'. The code includes various using statements for Embrace, Microsoft.AspNetCore.Identity, and Microsoft.EntityFrameworkCore. It defines a builder, adds services, and configures the application. The code is as follows:

```
1 using Embrace.Data;
2 using Embrace.Models;
3 using Embrace.Services;
4 using Microsoft.AspNetCore.Identity;
5 using Microsoft.AspNetCore.Identity.UI.Services;
6 using Microsoft.EntityFrameworkCore;
7 using System.Security.Claims;
8
9 var builder = WebApplication.CreateBuilder(args);
10 builder.Services.AddRazorPages().AddRazorRuntimeCompilation();
11
12 // Add services to the container
13 builder.Services.AddControllersWithViews();
14 builder.Services.AddScoped<ApplicationDbContext>();
15
16 // MySQL DB connection
17 string _GetConnStringName = builder.Configuration.GetConnectionString("DefaultConnectionMySQL");
18 builder.Services.AddDbContextPool<ApplicationDbContext>(options => options.UseMySQL(_GetConnStringName, ServerVersion.AutoDetect(_GetConnStringName)));
19
20 // Register IEmailSender Service
21 builder.Services.AddSingleton<IEmailSender, MockEmailSender>();
22
23 builder.Services.AddIdentity<User, IdentityRole>()
24     .AddEntityFrameworkStores<ApplicationDbContext>()
25     .AddDefaultUI();
26
27 builder.Services.AddAuthorization();
28 builder.Services.AddAuthentication();
29
30 // TODO: set up so that it only does this if builder.Environment.IsDevelopment()
31 builder.Configuration.AddUserSecrets<Program>();
32
33 var app = builder.Build();
34
35 // Ensure database is seeded
36 using (var scope = app.Services.CreateScope())
37 {
38     var context = scope.ServiceProvider.GetRequiredService<ApplicationDbContext>();
39     // Initialize the database and seed test data
40     await DbInitializer.Initialize(scope.ServiceProvider, context);
41 }
42
43 // Configure the HTTP request pipeline.
44 if (!app.Environment.IsDevelopment())
45 {
46     app.UseExceptionHandler("/Home/Error");
47     // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
48     app.UseHsts();
49 }
```

**Figure 52: Program File Part 1**

*Generated code was modified to implement user authentication.*

A screenshot of a Visual Studio code editor showing the second part of a C# program file named Program.cs. The code continues from the previous part, adding HTTPS redirection, mapping the Identity API, and configuring the application. The code is as follows:

```
50
51 app.UseHttpsRedirection();
52 app.MapIdentityApi<User>();
53
54 // Add protected endpoint for API; require auth before accessing endpoint
55 app.MapGet("Users/Profile", async (ClaimsPrincipal claims, ApplicationDbContext context) =>
56 {
57     string userId = claims.Claims.First(claims => claims.Type == ClaimTypes.NameIdentifier).Value;
58     return await context.UserClaims.FindAsync(userId);
59 })
60 .RequireAuthorization();
61
62 app.UseStaticFiles();
63
64 app.UseRouting();
65
66 app.UseAuthorization();
67
68 app.MapControllerRoute(
69     name: "default",
70     pattern: "{controller=Home}/{action=Index}/{id?}");
71
72 app.Run();
73
```

**Figure 53: Program File Part 2**

*Generated code was modified to implement user authentication.*



## CSS

### Discussion Board CSS

```
.connection-corner-description {
  padding-top: 0.8em;
}

:root {
  --base: 1.35; /* multiplier across elements */
  --background: hsl(360 100% 100%);
}

.no-discussion-boards {
  margin: 4em;
  text-align: center;
  justify-content: center;
}

/* card styling reference code: https://codepen.io/collection/XLRrqW/
*/
/* fading effect reference code:
https://css-tricks.com/recreating-mdns-truncated-text-effect/*/
.cards {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  gap: .5em;
}

.discussion-board-row {
  justify-items: center;
}

.title-card {
  background-color: #FFFFFF;
  border-radius: 8px;
  width: 75%;
  height: min-content;
  text-align: left;
}

.comments {
  justify-items: center;
  background-color: #FFFFFF;
  width: 75%;
```

```
    margin-top: .2em;
}

.comment-container {
    width: 90%;
    padding: .2em .2em .2em .2em;
}

.comment-card {
    margin-top: .2em;
    padding: .5em .5em .5em .5em;
    background-color: #f8f9fa;
    border-radius: 8px;
    width: 100%;
    height: min-content;
    text-align: left;
}

.comment-header {
    background-color: #f8f9fa;
    justify-content: left;
    font-weight: bold;
}

.comment-card-body {
    padding: 0.5rem 1rem;
}

.comment-card p {
    font-size: .8em;
}

.comment-content {
    padding-top: 1em;
}

.title-card-body {
    padding: 0.5rem 1rem;
}

.title-tag {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-between;
    align-items: center;
```

```
    width: 100%;
}

.title-card-metadata {
    padding-bottom: 1em;
}

.title-card-metadata p {
    font-style: italic;
}

.title-user, .title-date-posted, .comment-title {
    display: flex;
    align-items: center;
    font-style: italic;
}

.comment-title-user {
    display: flex;
    width: fit-content;
    padding-right: 1em;
}

.title-user p, .title-date-posted p, .comment-title p {
    margin: 0px;
}

.title-icon-image {
    display: flex;
    justify-content: center;
    align-items: center;
}

.title-icon-image img {
    max-width: 1.2em;
    max-height: 1.2em;
    margin-right: .3em;
}

.card {
    flex: 1 0 100%;
    height: 15em;
}

@media screen and (min-width: 62em) {
    .cards {
```

```
        display: flex;
        flex-wrap: wrap;
        justify-content: flex-start;
        gap: .45em;
    }

    .card {
        flex: 0 1 calc(49.5%);
        height: 15em;
    }
}

@media screen and (min-width: 88em) {
    .cards {
        display: flex;
        flex-wrap: wrap;
        justify-content: flex-start;
        gap: .45em;
    }

    .card {
        flex: 0 1 calc(32.85%);
        height: 15em;
    }
}

.card-header, .card-header:hover {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-between;
    align-items: center;
    font-size: 1.1rem;
    width: 100%;
    padding: 0.5em;
    text-decoration: none;
    color: #000000;
}

.card-header p {
    margin: 0px;
}

.header-left {
    display: flex;
    flex-wrap: wrap;
```

```
    justify-content: left;
    align-items: center;
    margin: 0px;
    gap: 0.5rem;
}

.header-right {
    width: fit-content;
    justify-content: center;
    justify-items: right;
}

.post-comment-button-row {
    text-align: right;
    margin-top: .4em;
}

.discussion-tag {
    top: 0.5rem;
    right: 0.5rem;
    border-radius: 4px;
    padding: 0.25rem 0.5rem;
    font-size: 0.85rem;
    font-weight: 600;
    white-space: nowrap;
    object-fit: contain;
    width: fit-content;
    flex-shrink: 0;
}

.question-tag {
    background-color: #dafad2; /* green */
    color: #285c1b;
    border: 1px solid #285c1b;
}

.conversation-tag {
    background-color: #d6e3f4; /* blue */
    color: #3178c6;
    border: 1px solid #3178c6;
}

.meetup-tag {
    background-color: #fadeb9; /* orange */
    color: #cb7b11;
```

```
border: 1px solid #cb7b11;
}

.announcement-tag {
  background-color: #e3dfff; /* purple */
  color: #46355f;
  border: 1px solid #46355f;
}

.event-tag {
  background-color: #fff5ff; /* pink */
  color: #800065;
  border: 1px solid #800065;
}

.other-tag {
  background-color: #d8d8d8; /* gray */
  color: #515151;
  border: 1px solid #515151;
}

.card:hover {
  transform: scale(1.02);
}

.card-title {
  margin: 0px;
  text-align: left;
}

.card-body p, .add-comment-form, .form-control {
  max-height: calc(8rem * var(--base));
  overflow: hidden; /* truncate text so that it stays within the
body area */
  position: relative;
  font-size: .9em;
}

.form-control {
  width: 99%;
}

.card-body p.fade-if-overflow:after {
  content: "";
```

```
background: linear-gradient(to right, transparent,
var(--background) 80%);
display: block;
height: calc(1rem * var(--base) + 10px);
inset-block-end: 0;
position: absolute;
bottom: 0;
left: 0;
width: 100%;
}
```

```
.card-icon {
width: 30px;
height: 30px;
object-fit: contain;
margin-right: 0.5rem;
}
```

```
.card-text {
margin: 0px;
}
```

```
.card-footer {
display: flex;
justify-content: space-between;
padding: 0.5rem 1rem;
background-color: #f8f9fa;
font-size: 0.9rem;
width: 100%;
}
```

```
.card-count {
font-weight: bold;
}
```

```
.discussion-board-card-buttons {
width: 100%;
margin: 0 18px 10px 0;
text-align: right;
}
```

## Document CSS

```
.document-image img {
max-width: 7em;
```

```
    max-height: 7em;
    object-fit: cover;
    padding: 0px;
}

.tbody {
    padding-top: 0px;
}

.table .document-row {
    height: auto;
    padding-top: 0;
}

.doc-upload-left {
    padding-top: .6em;
}

.doc-upload-description {
    padding-bottom: 1em;
}

.language-row th, .language-row td {
    padding: 0px 0px 0px 0px;
}

.lang-selection-title {
    font-weight: bold;
    padding: 1em 0em 0em 0em;
    margin: 0px;
}

.translate-btn {
    margin-top: .4em;
}
```

### Resource CSS

```
.resource-list-description {
    padding-top: .8em;
}

.resource-image {
    width: 8em;
}
```



```
.resource-image img {
    max-width: 100%;
    max-height: 200px;
    width: auto;
    height: auto;
}

.service-categories {
    list-style-type: none; /* Remove default bullet points */
    padding: 0;
    margin: 0;
}

.checkmark-bullet {
    display: flex;
    align-items: center;
    margin-bottom: 5px;
    padding-left: 25px;
    background-image: url('/images/checkmarkIcon.png');
    background-size: 16px 16px;
    background-repeat: no-repeat;
    background-position: left center;
}

.row {
    padding-bottom: 1em;
}

.description-column {
    text-align: left;
    width: 60%;
}

.image-column {
    width: 40%;
    display: flex;
    justify-content: center;
}

.image-column img {
    width: 100%;
    max-height: 10em;
    object-fit: contain;
}
```

```
.resource-description {  
    font-style: italic;  
    text-align: left;  
    padding-bottom: 1em;  
    width: 100%;  
}
```

## Site CSS

```
html {  
    font-size: 1em;  
}  
  
@media (min-width: 768px) {  
    html {  
        font-size: 1em;  
    }  
}  
  
.search-container .d-flex {  
    display: flex;  
    gap: 0.5rem;  
    margin-bottom: .5rem;  
}  
  
.search-container .btn,  
.search-container .form-select,  
.search-container .form-control {  
    flex: 1 1 auto; /* they can grow/shrink as needed */  
    min-width: 6em; /* prevents them from becoming too narrow */  
}  
  
.form-select, .form-control {  
    font-size: inherit;  
    margin: 2px;  
}  
  
@media (max-width: 62em) {  
    .search-container .d-flex {  
        flex-wrap: wrap;  
    }  
}  
  
/* button styling */  
.btn:focus, .btn:active:focus, .btn-link.nav-link:focus,  
.form-control:focus, .form-check-input:focus {
```

```
    box-shadow: 0 0 0 0.1rem white, 0 0 0 0.25rem #258cfb;
}

.btn-primary {
    margin-top: .3em;
}

.btn {
    color: white;
    white-space: nowrap;
    margin: 2px;
}

.btn-create {
    background-color: #9656a2;
}

.btn-create:hover {
    background-color: #81488c;
    color: white;
}

.btn-search {
    background-color: #17a2b8;
}

.btn-search:hover {
    background-color: #13919d;
    color: white;
}

.btn-details {
    background-color: #dec000;
}

.btn-details:hover {
    background-color: #c4a700;
    color: white;
}

.btn-save {
    background-color: #EB9E46;
}

.btn-save:hover {
```

```
        background-color: #D58932;
        color: white;
    }

    .btn-delete {
        background-color: #de324c;
    }

    .btn-delete:hover {
        background-color: #c42a42;
        color: white;
    }
/* end button styling */

html {
    position: relative;
    min-height: 100%;
}

body {
    margin-bottom: 60px;
    background-color: #E1F2F7;
    font-family: 'Lato', serif;
    font-size: 1.3em;
}

.lead {
    font-weight: normal;
}

h1, h2 {
    padding-top: .5em;
}

h1, h2, h3, h4, h5, h6 {
    font-weight: bold;
    font-family: 'Lustria', serif;
}

.navbar {
    background-color: #FFFFFF;
}

.navbar-brand {
    font-family: 'Lustria', serif;
```

```
    font-size: 1em;
}

.navbar-nav, .btn {
    font-family: 'Lustria', serif;
    font-size: 1em;
}

.nav-item {
    padding-right: .8em;
}

.welcome-text {
    max-width: 600px;
    margin: 0 auto;
    text-align: center;
}

.section-title {
    font-weight: bold;
    margin-bottom: 20px;
}

/* card styling */
.card {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: space-between;
    text-align: left;
    height: 100%;
}

.col-md-4 {
    padding-bottom: 1em;
}

.card-body {
    flex-grow: 1;
    display: flex;
    flex-direction: column;
    height: 100%;
    width: 90%;
    text-align: left;
}
```

```
.card, .card:hover {
    background-color: #FFFFFF;
    box-shadow: rgba(0,0,0,0,4);
    text-decoration: none;
    color: #000000;
    border-radius: 8px;
}

.card-img {
    height: 210px;
    width: 100%;
    object-fit: cover;
    display: block;
    margin: auto;
}

.card-img-doc {
    height: 210px;
    padding-top: 10px;
}

.card-title {
    font-weight: bold;
    font-size: 1em;
}

.card-text {
    font-size: 1em;
    font-weight: normal;
}

/* table styling */
.table {
    border-spacing: 0px 10px;
    border-collapse: separate;
}

.table td, .table th {
    padding: .7em;
    align-content: center;
    text-align: left;
}

.table th h4 {
```

```
    color: white;
    padding: 2px;
}

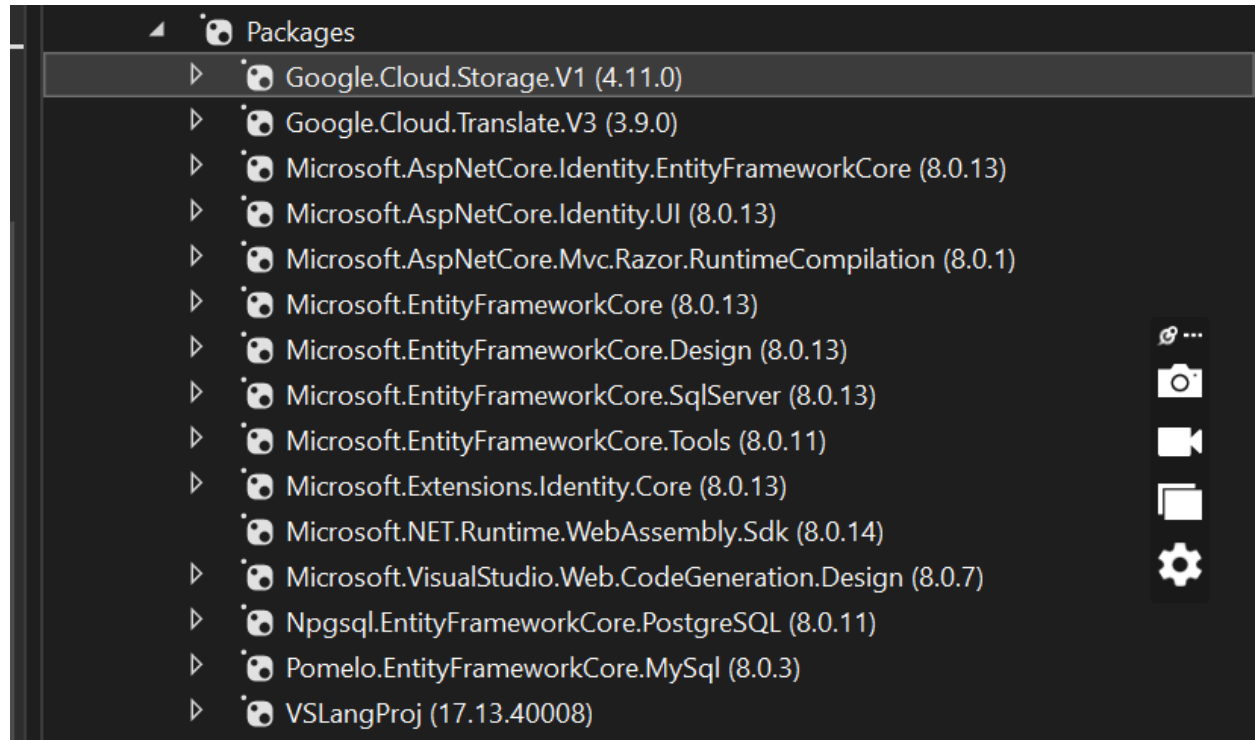
.resource-header {
    background-color: #004080;
    box-shadow: 0px 0px 9px 0px rgba(0,0,0,0.1);
}

.resource-row, .document-row {
    background-color: #ffffff;
    border-radius: 50px;
    box-shadow: 0px 0px 9px 0px rgba(0,0,0,0.1);
}

.footer {
    position: absolute;
    bottom: 1em;
    margin-bottom: 0px;
    width: 100%;
}

.create-page {
    padding-top: .8em;
}
```

## PACKAGES USED



**Figure 54: List of Project Dependencies**



## REFERENCES

“Addresses 3.0.” Addresses 3.0 | Devportal, [developer.usps.com/addressesv3](https://developer.usps.com/addressesv3). Accessed 6 Dec. 2024.

Anderson, Rick. “Get Started with ASP.NET Core MVC.” Microsoft Learn, [learn.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/start-mvc?view=aspnetcore-8.0&tabs=visual-studio](https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/start-mvc?view=aspnetcore-8.0&tabs=visual-studio). Accessed 5 Dec. 2024.

Ardalis. “Common Web Application Architectures - .NET.” .NET | Microsoft Learn, [learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures](https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures). Accessed 5 Dec. 2024.

“ASP.NET MVC Architecture.” *TutorialsTeacher*, [www.tutorialsteacher.com/mvc/mvc-architecture](https://www.tutorialsteacher.com/mvc/mvc-architecture). Accessed 31 Mar. 2025.

Chauhan, Shailendra. “Understanding MVC, MVP and MVVM Design Patterns.” Learn to Code, Prepare for Interviews, and Get Hired, ScholarHat, 27 July 2024, [www.scholarhat.com/tutorial/designpatterns/understanding-mvc-mvp-and-mvvm-design-pattern](https://www.scholarhat.com/tutorial/designpatterns/understanding-mvc-mvp-and-mvvm-design-pattern).

“Cloud Translation API | Google Cloud.” Google, Google, [cloud.google.com/translate/docs/reference/rest](https://cloud.google.com/translate/docs/reference/rest). Accessed 6 Dec. 2024.

“Component-Based Diagram - Unified Modeling Language (UML).” GeeksforGeeks, GeeksforGeeks, 4 Nov. 2024, [www.geeksforgeeks.org/component-based-diagram/](https://www.geeksforgeeks.org/component-based-diagram/).

“Component Diagram Tutorial.” Lucidchart, [www.lucidchart.com/pages/uml-component-diagram](https://www.lucidchart.com/pages/uml-component-diagram). Accessed 6 Dec. 2024.

“Connecting with Resources.” Bienvenidos a Gallatin Valley, [bienvenidosgv.org/connecting-with-resources](https://bienvenidosgv.org/connecting-with-resources). Accessed 4 Dec. 2024.

“Detect Text in Images | Cloud Vision API | Google Cloud.” Google, Google, [cloud.google.com/vision/docs/ocr](https://cloud.google.com/vision/docs/ocr). Accessed 6 Dec. 2024.

Dot Net Tutorials. “Register New User Using ASP.NET Core Identity.” *Dot Net Tutorials*, 22 May 2024, [dotnettutorials.net/lesson/register-new-user-using-asp-net-core-identity/](https://dotnettutorials.net/lesson/register-new-user-using-asp-net-core-identity/).

Dot Net Tutorials. “Connecting Mysql in ASP.NET Core via 2 Ways.” Dot Net Tutorials, 4 Apr. 2024, [dotnettutorials.net/connecting-mysql-in-asp-net-core-via-2-ways/](https://dotnettutorials.net/connecting-mysql-in-asp-net-core-via-2-ways/).

Doyle, Wes. “ASP.NET Core 2 MVC Forum | 03 | Creating Models.” *YouTube*, YouTube, [www.youtube.com/watch?v=9HNY4ZVG9IQ&list=PL3\\_YUnRN3Uhiz2HomrXKcaEW6b3pDhKTX&index=5](https://www.youtube.com/watch?v=9HNY4ZVG9IQ&list=PL3_YUnRN3Uhiz2HomrXKcaEW6b3pDhKTX&index=5). Accessed 2 May 2025.

Dykstra, Tom, and Rick Anderson. “Tutorial: Get Started with EF Core in an ASP.NET MVC Web App.” Microsoft Learn, [learn.microsoft.com/en-us/aspnet/core/data/ef-mvc/intro?view=aspnetcore-9.0](https://learn.microsoft.com/en-us/aspnet/core/data/ef-mvc/intro?view=aspnetcore-9.0). Accessed 6 Dec. 2024.

Fakhroutdinov, Kirill. “Sentinel Hasp Licensing Components.” UML Graphical Notation Overview, Examples, and Reference., 16 Apr. 2013, [www.uml-diagrams.org/software-licensing-component-diagram-example.html?context=cmp-examples](https://www.uml-diagrams.org/software-licensing-component-diagram-example.html?context=cmp-examples).

“Find Citizenship Resources: Homeland Security.” U.S. Department of Homeland Security, [www.dhs.gov/find-citizenship-resources](http://www.dhs.gov/find-citizenship-resources). Accessed 4 Dec. 2024.

Galvan, David, and Rick Anderson. “ENFORCE HTTPS in ASP.NET Core.” Microsoft Learn, [learn.microsoft.com/en-us/aspnet/core/security/enforcing-ssl?view=aspnetcore-8.0&tabs=visual-studio%2Clinux-sles](https://learn.microsoft.com/en-us/aspnet/core/security/enforcing-ssl?view=aspnetcore-8.0&tabs=visual-studio%2Clinux-sles). Accessed 6 Dec. 2024.

“How to Create UML Component Diagram Effortlessly.” Edraw, [www.edrawsoft.com/create-uml-component-diagram.html](https://www.edrawsoft.com/create-uml-component-diagram.html). Accessed 6 Dec. 2024.

“Immigrant Legal Resource Center.” ILRC, [www.ilrc.org/](http://www.ilrc.org/). Accessed 4 Dec. 2024.

“Login and Logout in ASP.NET Core Identity.” Dot Net Tutorials, 22 May 2024, [dotnettutorials.net/lesson/login-and-logout-in-asp-net-core-identity/](https://dotnettutorials.net/lesson/login-and-logout-in-asp-net-core-identity/).

“MVC Design Pattern.” GeeksforGeeks, GeeksforGeeks, 11 Oct. 2024, [www.geeksforgeeks.org/mvc-design-pattern/](https://www.geeksforgeeks.org/mvc-design-pattern/).

“Resources for Immigrants in the U.S.” United We Dream, 16 Oct. 2024, [unitedwedream.org/resources/](https://unitedwedream.org/resources/).

Rick-Anderson. “Introduction to Identity on Asp.Net Core.” Microsoft Learn, [learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-8.0&tabs=visual-studio](https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-8.0&tabs=visual-studio). Accessed 6 Dec. 2024.

Teja, Weerayut. “Building Your First .Net Core Web App: A Step-by-Step Guide.” Medium, Itthirit Technology, 13 Aug. 2023, [medium.com/itthirit-technology/building-your-first-net-core-web-app-a-step-by-step-guide-a69af3f55105](https://medium.com/itthirit-technology/building-your-first-net-core-web-app-a-step-by-step-guide-a69af3f55105).

“UML Use Case Diagram Tutorial.” Lucidchart, [www.lucidchart.com/pages/uml-use-case-diagram](https://www.lucidchart.com/pages/uml-use-case-diagram). Accessed 6 Dec. 2024.

“UML Use Case Examples of Common Scenarios: EdrawMax.” Edrawsoft, [www.edrawsoft.com/article/use-case-diagram-examples.html?refid=0FOF24644909943311156](https://www.edrawsoft.com/article/use-case-diagram-examples.html?refid=0FOF24644909943311156). Accessed 6 Dec. 2024.

“Use Case Diagram - Unified Modeling Language (UML).” GeeksforGeeks, GeeksforGeeks, 14 Oct. 2024, [www.geeksforgeeks.org/use-case-diagram/](https://www.geeksforgeeks.org/use-case-diagram/).

“What Is ASP.NET Core?: .NET.” Microsoft, [dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet-core](https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet-core). Accessed 5 Dec. 2024.

What Is Component Diagram?, [www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/](https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/). Accessed 6 Dec. 2024.

Zang, Assis. “ASP.NET Core Basics: Working with a Database.” Telerik Blogs, Telerik, 8 Dec. 2023, [www.telerik.com/blogs/aspnet-core-basics-working-database](https://www.telerik.com/blogs/aspnet-core-basics-working-database).