DEBATE AND CONNECT FORUM APPLICATION

Group : 33

Members : Arif Ozden, Tekin Bal, Ali Riza Keles

Table of Contents

1. Introduction

This document describes in detail the design, architecture and implementation of a discussion forum application built using ASP.NET Core 6.0. The application allows users to create and view forum topics, respond and evaluate responses. The application also includes user authentication and authorization, as well as various basic error handling functions.

2. Architecture

The application is built using a three-layer architecture:

Presentation layer: The presentation layer is responsible for rendering the user interface and handling user interactions.

Application layer: The application layer contains the working organization of the application, such as the logic for creating, reading, updating and deleting forum threads and replies. It is implemented using the Model-View-Controller (MVC) model.

Data layer: The data layer is responsible for storing and retrieving data from the database. It is implemented using Entity Framework Core, an object-relational mapper (ORM) that provides a high-level abstraction for database access.

2.1 Models

The application has two models:

ForumQuestions: This model represents a forum topic. It has the following properties:

- Id: The unique identifier of the topic.
- Title: The title of the topic.
- Description: The content of the topic.
- IdentityUserId: The name or nick of user
- User: The identy of user.

- Answers: List of answers

ForumAnswer: This model represents a forum reply. It has the following properties:

- Id: The unique identifier of the reply.(int)
- QuestionId: The identifier of the question the reply is associated with.(int)
- User: The identy of user.
- IdentityUserId: The name or nick of user
- Content: The content of the reply.(string)
- Question: The date and time the reply was created.

2.2 Views

The application has three views:

Home: This view makes web pages dynamic and user-friendly.

Questions: This view allows users to create a new forum topic and to post a reply to a forum topic

Shared: This view manages authentication and authorization processes such as user login and logout

2.3 Controllers

The application has two controllers:

1.Home Controller:

This class acts as the main controller. This controller contains three actions, ‘Index’, ‘Privacy’ and ‘Error’.

2.QuestionsController:

This controller contains different processes that implement the CRUD operations of "Question".

Index Gives a list of all questions asked by users.

Details: Returns the details of a previously created question.

Creation: Returns a view to create a new question.

Edit: Returns a view to update a previously created question.

Delete: Returns a view to delete a question that was previously created.

3. AnswerController

Details: Returns the answers of a previously created question.

2.4 Data access layer

The application uses the repository model to isolate the data access layer from the business logic layer. Repositories are tasked with performing CRUD (create, read, update, delete) operations for a specific entity type.

The application has two repositories:

QuestionRepository: This repository provides CRUD operations for forum topics.

AnswerRepository: This repository provides CRUD operations for forum replies.

3. Database

The application uses a relational database to store data. The database schema is as follows:

SQL

```
CREATE TABLE "Questions" (
    "Id" INTEGER NOT NULL CONSTRAINT "PK_Questions" PRIMARY KEY AUTOINCREMENT,
    "Title" TEXT NOT NULL,
    "Description" TEXT NOT NULL,
    "Category" TEXT NOT NULL,
    "IdentityUserId" TEXT NULL,
    CONSTRAINT "FK_Questions_AspNetUsers_IdentityUserId" FOREIGN KEY ("IdentityUserId") REFERENCES "AspNetUsers" ("Id")
```

```
CREATE TABLE "Answers" (
    "Id" INTEGER NOT NULL CONSTRAINT "PK_Answers" PRIMARY KEY AUTOINCREMENT,
    "Content" TEXT NOT NULL,
    "IdentityUserId" TEXT NULL,
    "QuestionId" INTEGER NULL,
    CONSTRAINT "FK_Answers_AspNetUsers_IdentityUserId" FOREIGN KEY ("IdentityUserId") REFERENCES "AspNetUsers" ("Id"),
```

CONSTRAINT "FK_Answers_Questions_QuestionId" FOREIGN KEY ("QuestionId") REFERENCES "Questions" ("Id")

Note: We got other sql-codes from packages.

4. Coding Details

4.1 Design

The application is designed using a modular approach, where the code is divided into multiple namespaces, each containing a defined set of classes. This method makes the code easier to understand and easier to maintain.

This app uses Bootstrap to create a responsive and user-friendly design. The app also uses Font Awesome for icons. The app has a simple and straightforward layout with a navigation bar at the top, a footer at the bottom, and a main content area in the middle. The app has a white background and uses a light blue color scheme with black text.

The app also uses jQuery to add interactivity and dynamic content to views. For example, the application uses jQuery to do the following features:

Search functionality: The app allows users to search for topics by title or creator using a search box in the navigation bar. In practice, thanks to jQuery, it is possible to send an AJAX request to the server with the search query and update the view with the search results without reloading the page.

Reply functionality: The app allows users to reply to comments by clicking on reply buttons. Again, thanks to jQuery, the application allows showing a response form under the comment without reloading the page.

4.2 Models

Models are defined in the Models folder of the project. Models use data annotations to specify validation rules and display attributes for each property. Models also use navigation properties to define relationships between them. For example, the Review model has a Topic property that references the topic it belongs to, and a Ratings property that references the list of ratings it has.

Models also implement some specific methods to implement some business logic. For example, the Review model has a GetRating method that calculates the total rating of a review by summing the values of its ratings.

4.3 CRUD operations

CRUD operations on questions and incoming answers are performed in QuestionController using various methods. The controller uses model binding to retrieve data from views and validate them using ModelState.IsValid. The controller also returns feedback such as error messages when there is incorrect input.

For example, the Create action retrieves a Question pattern from the view and validates it. If valid, it calls the Create method from the QuestionController to add it to the database and redirects to the Index action where the questions are located. If it is not valid, it remains in the Create view with an error message, waiting for the user to enter valid input.

All elements of CRUD are used here. That is, Create (Create), Read (Details), Update (Edit), Delete (Delete). Above these, questions are listed with a QuestionsIndex and a user-friendly interface is provided. The user can see the questions here with the Read (Details) function. You can filter the questions according to their categories with the Filter method, or you can search the

questions by entering the words you are looking for with the Search method. If he cannot find the topic he is looking for, he can ask a new question using the Create method. If it wants to make changes to the question it asks, it can edit its question using the Update (Edit) method or delete its question completely.

5. Conclusion

In conclusion, the goal of our discussion forum application is to create a sustainable and user-friendly platform for discussions and knowledge sharing between users in the simplest way possible. Here we have tried to demonstrate the capabilities of the .NET Core MVC framework. By adhering to established coding norms and applying current coding techniques in software development, we have tried to provide a stable and secure environment for users to engage in dialog and collaborative learning. The implementation of the project requirements with emphasis on user satisfaction and functionality reflects the team's commitment to deliver an efficient solution.

6. Contribution List

Arif Ozden, Tekin Bal and Ali Riza Keles did coding of all documents. Controllers, Models and Views. Arif Ozden did design and front-end, Arif Ozden, Tekin Bal and Ali Riza Keles did documentation.

Reference List

Chatgpt was used to write descriptions of some items.

https://getbootstrap.com

https://blog.jquery.com

https://www.w3schools.com/asp/default.asp

https://stackoverflow.com