06.06.2024

Industrial Informatics
Semester Project

Students Unite
A student flavored forum website

Team members:

Eker Kara Deniz

Denes Tams Attila

Cocea Vlad Stefa

Contents

Chapter 1. Introduction

Overview

In the frames of the Industrial Informatics subject's semester project, we were given the task to develop a website in a team with functional roles. The topic of the website was free to choose, thus my team has chosen to develop a forum website with a theme for students and so we came up with StudentsUnite, a site where anyone can share the idea, the opinion, or the experience that is on their mind, easily for others to see. Although the idea seems promising and a very complex website with many features in general and specially tailored ones for students could be added to this project, we remained on the more simple and humble side of the development because we considered this project mainly as an exercise for learning and treated it as such. Our team, I believe, learned many things in the making of this project and not just about the used technologies but also about working in a group and about ourselves as well.

Project Team

As we mentioned earlier the respective project was meant to be a team effort, this year's teams being on the smaller size of three members we needed to put in some additional effort each as such we did each other's task if we had time to help each other out. But most of the task of the different roles mainly were finished under their own hands.

The roles and the people occupying it were the following:

- *Denes Tamas-Attila*: team lead
- *Cocea Vlad-Stefa*: software developer
- *Eker Kara Deniz*: software tester

Each role consisting of the following proposed responsibilities:

Team lead: establishes the project goals and the functional specifications of the application, as well as the structure of the GUIs and of the source code, assigns the tasks of each team member, coordinates the team during the project development, checks the functionality of the application and takes actions to meet its requirements and specifications, elaborates the documentation and presents the project

Software developer: implement the source code and add comments according to the project specifications, design and implement the database, perform other tasks assigned to them by the project manager

Software tester: checks the project functionality, reports the identified errors, performs other tasks assigned to him/her by the project manager

We can see that only the software developers' goal was to implement the application but we all took part in it, thus accelerating the development process.

The projects' goal

The projects' main goal is the development of an application which complies with the proposed requirements[1]:

- Layered application
- Database with at least 4 tables/collections/structures
- Minimum of six GUIs
- Authentication of at least two types of users

These requirements could have been met within four different types of applications. From these four we choose Model-View-Controller, MVC, ASP.NET Web Application.

As we will see, we achieved each of the proposed requirements.

In addition to the project's requirements, we wanted to create something that could help us and our fellow students, for this a forum like application was the one that came instantly in our mind, where we can express ourselves. We mentioned in the beginning that is has a theme for students, this is because we developed an application that with a little effort can be turned into any themed forum, thus make it more general.

The features of the forum that we wanted to have, were the following:

- adding/deleting/editing/viewing discussions
- adding/deleting/editing/viewing comments to discussions
- adding/deleting/editing/viewing replies to comments
- some search functionality
- authentication
- account management
- administration through special account: deleting/editing discussions, comments or replies; blocking users
- easy intuitive UI

[1] https://users.utcluj.ro/~tsanislav/files/II/II_Project.pdf

Chapter 2. Application design

This chapter describes the applications design, the projects' structure, the database design and such element, basically the projects design part where after the projects goal and the requirements were understood and well defined the design team or in our case mostly the team lead will think of the application design that will respect the requirements while still reaching the proposed goals.

The MVC design pattern

This design patter is widely used to make webpages, although reactive API based web applications are in the fashion today. MVC consist of three main elements: model, view and controller. The main idea of this separation is to make the applications structures simpler and introduce some type of code reuse. Each part of this pattern has a specific role: the model contains pure application data, it does not hold logic about how should it present the data to the user; the view has the mentioned logic, it has the logic to display the data but it does not know anything about the data; the controller exits between the model and the view, it listens to events and executes the appropriate reaction tot these events, most of the time a method within the controller that is called on the method, since the view and the model are connected the result will automatically effect the view[2]. Figure 2.1 illustrates the basic structure of the MVC design pattern, which is not something strict, most of the time it is changed to suite the applications need.

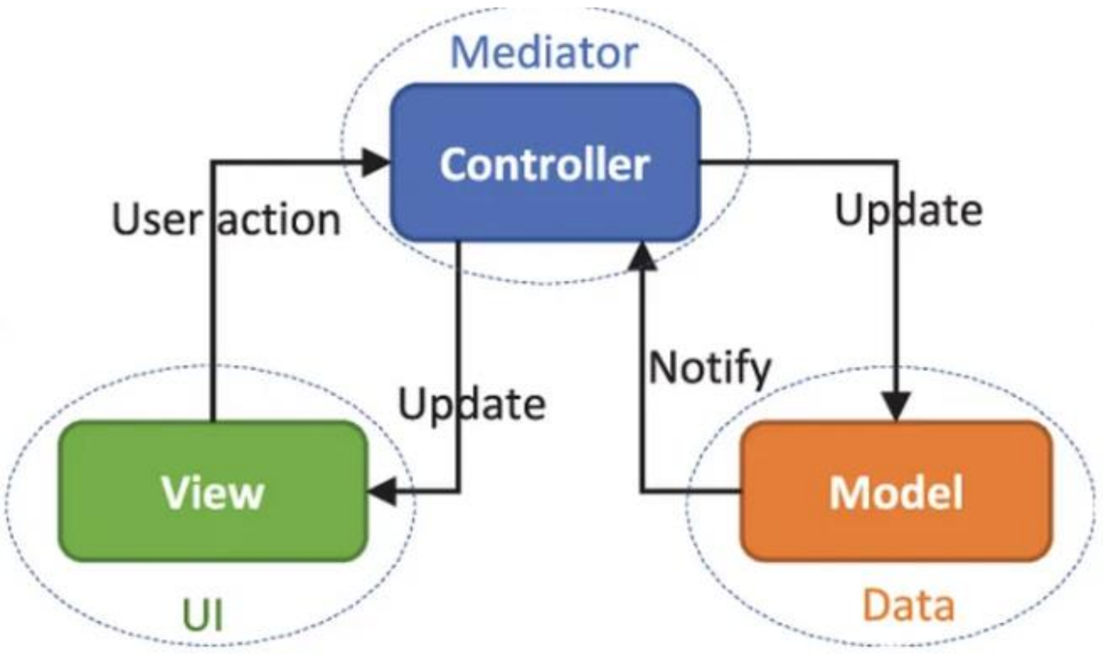


Figure 2.1 basic MVC structure[3]

[2] https://help.hcltechsw.com/commerce/9.1.0/developer/concepts/csdmvcdespat.html

[3] https://medium.com/@rhodunda/mvc-design-pattern-fe76175a01de

In the following the applications design will be explained and illustrated with the help of UML diagrams and we will observe that our design resembles the MVC design pattern.

The design of the application

In the design process, after we determined the features, we need our application to have, the next step would be to see, how could the user interact with the application to access those features. For this we can use the use case diagram to have a graphical representation of the user's interaction with the application.

Figure 2.2 shows the use case diagram for a user that does not have an account on the forum, lets name this actor guest, the guests have only limited functionalities. Whereas figure 2.3 illustrates the use case diagram for a user which has an account, this user has much more access of the offered features, but not as many as the admin which has the greatest number of features, because additionally to the accessible features of the registered user he also has admin functionalities as shown in figure 2.4.

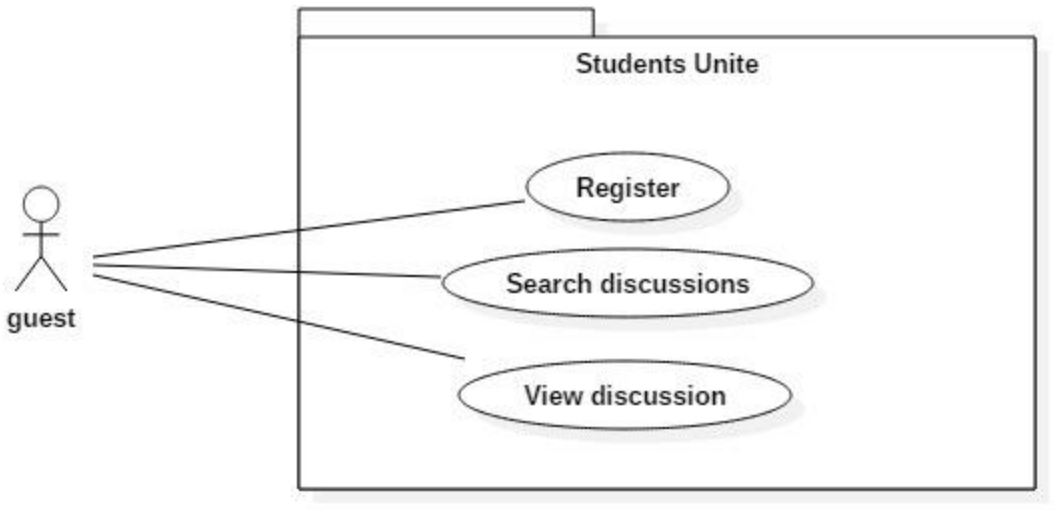


Figure 2.2 Guests use case diagram

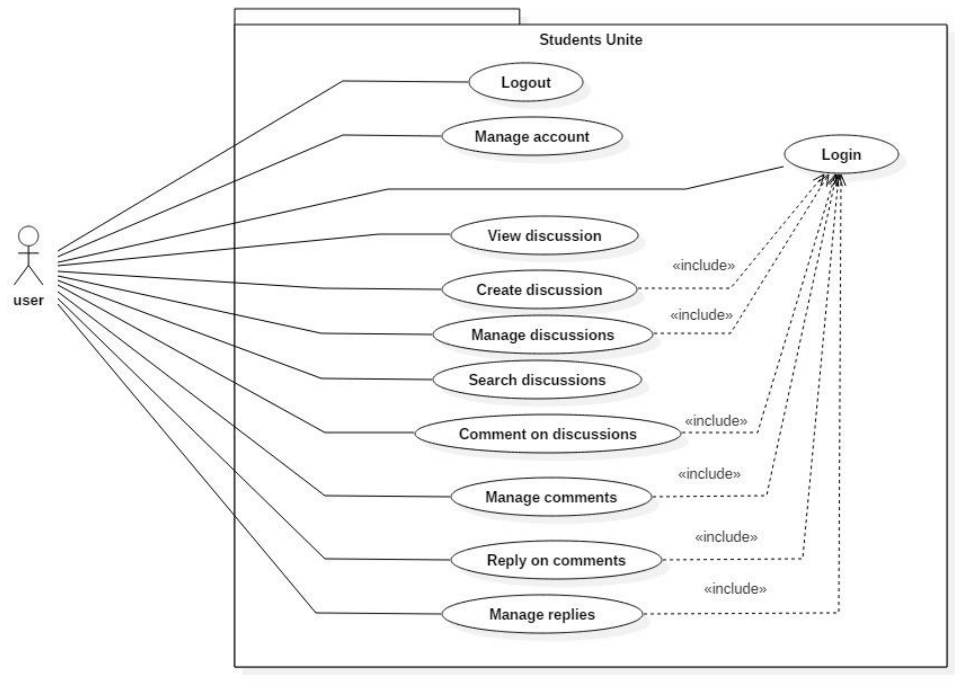


Figure 2.3 Users use case diagram

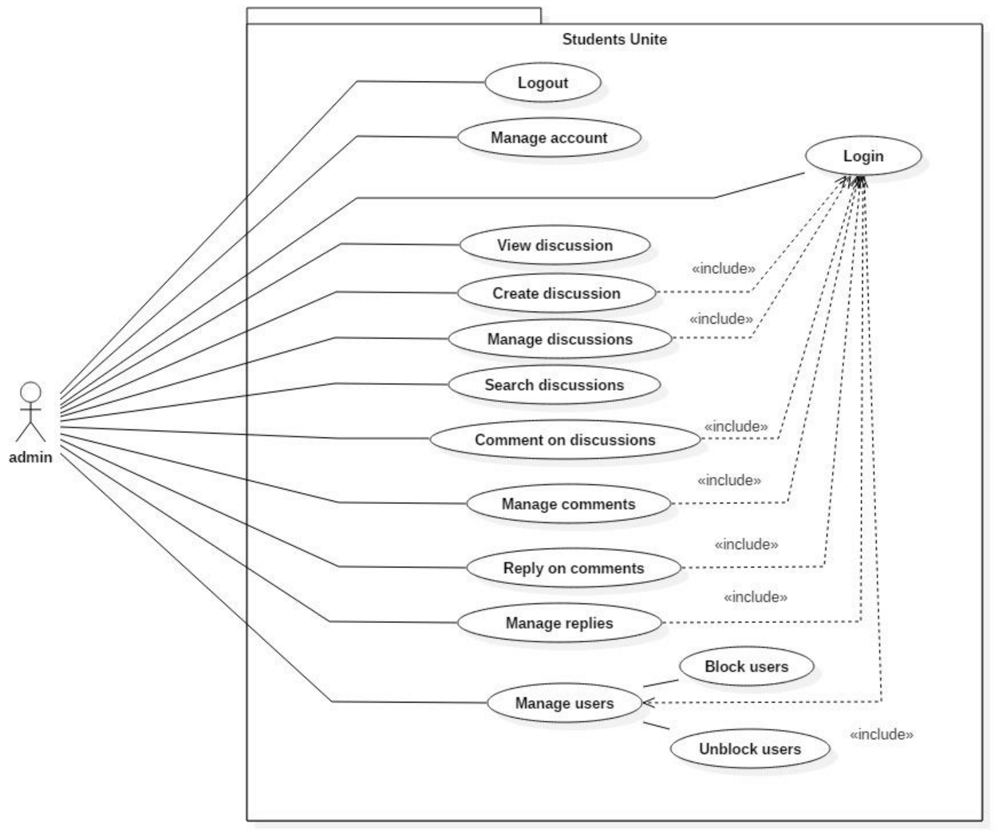


Figure 2.4 Admin use case diagram

The use case diagrams were followed by the class diagrams, here we will be able to observe the MVC design pattern of our application, initially we did not have the final design of our classes and so the initial class diagram design had been trough quite a number of alterations and changes, what we will show here is the final class diagrams. But first let us talk about class diagrams, this diagram is one of the most important diagrams when it comes to software development because it shows the structure of the classes used in an OOP application and the relationship between them, furthermore it also shows the necessary attributes and methods of these classes. Our application having multiple classes, many of them being complex ones implementing classes from frameworks, we will only show the relatable class diagrams, if the reader would like to have a fuller picture of the classes' interaction, annex 1. will offer it as a more complete class diagram.

Figure from figure 2.5 to figure 2.8 are showing the class diagrams of the main parts of our project: the models and the controllers mainly. On this diagram we can see that the different controllers have a center binding and interfacing role between models. Furthermore, we could not have inserted views everywhere because they do not behave as classes but we can observe in some places view models that are used to get information from the view that is not as full as the targeted model, when adding a new entity, for example the id should not be inputted by the user in the view (UI). With this abstraction in mind, we can think of the view as a hidden class, which is receiving and sending information on the controller's request, which is managing it all.

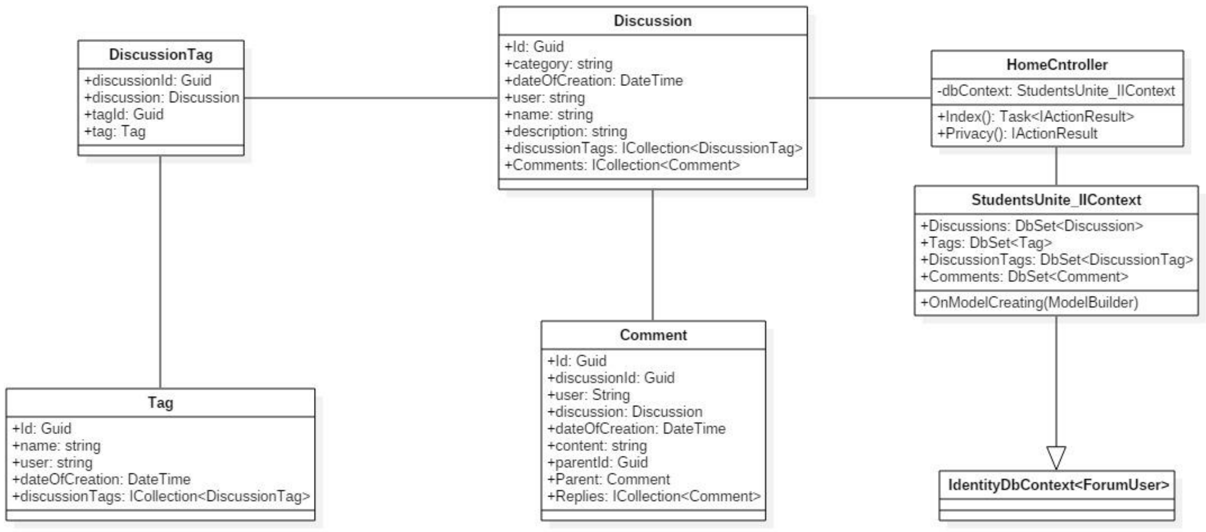

DiscussionTag
+discussionId: Guid
+discussion: Discussion
+tagId: Guid
+tag: Tag
Discussion
+Id: Guid
+category: string
+dateOfCreation: DateTime
+user: string
+name: string
+description: string
+discussionTags: ICollection<DiscussionTag>
+Comments: ICollection<Comment>
HomeCntroller
-dbContext: StudentsUnite_IIContext
+Index(): Task<IActionResult>
+Privacy(): IActionResult
StudentsUnite_IIContext
+Discussions: DbSet<Discussion>
+Tags: DbSet<Tag>
+DiscussionTags: DbSet<DiscussionTag>
+Comments: DbSet<Comment>
+OnModelCreating(ModelBuilder)
Tag
+Id: Guid
+name: string
+user: string
+dateOfCreation: DateTime
+discussionTags: ICollection<DiscussionTag>
Comment
+Id: Guid
+discussionId: Guid
+user: String
+discussion: Discussion
+dateOfCreation: DateTime
+content: string
+parentId: Guid
+Parent: Comment
+Replies: ICollection<Comment>
IdentityDbContext<ForumUser>

Figure 2.5 HomeController class diagram

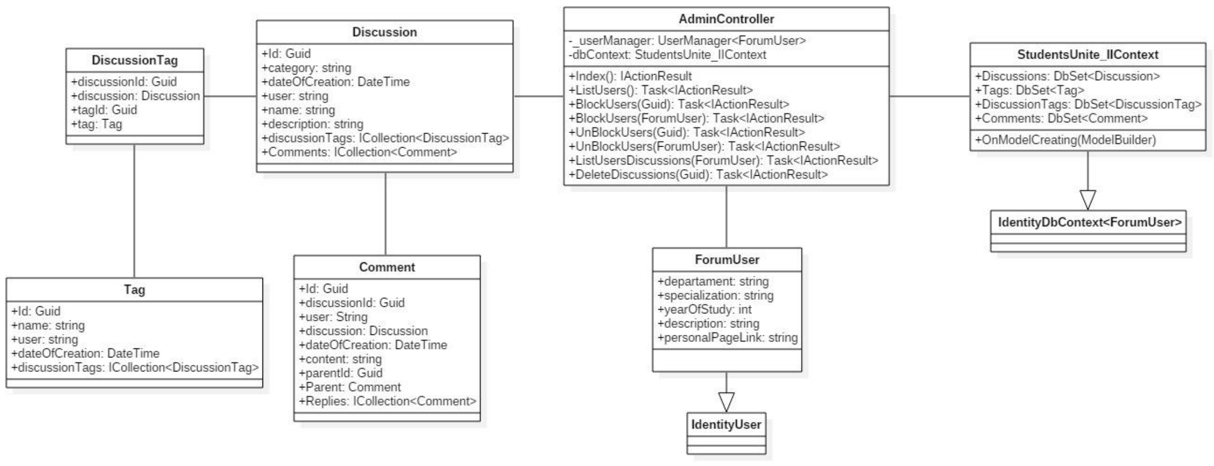

DiscussionTag
+discussionId: Guid
+discussion: Discussion
+tagId: Guid
+tag: Tag
Discussion
+Id: Guid
+category: string
+dateOfCreation: DateTime
+user: string
+name: string
+description: string
+discussionTags: ICollection<DiscussionTag>
+Comments: ICollection<Comment>
AdminController
-_userManager: UserManager<ForumUser>
-dbContext: StudentsUnite_IIContext
+Index(): IActionResult
+ListUsers(): Task<IActionResult>
+BlockUsers(Guid): Task<IActionResult>
+BlockUsers(ForumUser): Task<IActionResult>
+UnBlockUsers(Guid): Task<IActionResult>
+UnBlockUsers(ForumUser): Task<IActionResult>
+ListUsersDiscussions(ForumUser): Task<IActionResult>
+DeleteDiscussions(Guid): Task<IActionResult>
StudentsUnite_IIContext
+Discussions: DbSet<Discussion>
+Tags: DbSet<Tag>
+DiscussionTags: DbSet<DiscussionTag>
+Comments: DbSet<Comment>
+OnModelCreating(ModelBuilder)
IdentityDbContext<ForumUser>
Tag
+Id: Guid
+name: string
+user: string
+dateOfCreation: DateTime
+discussionTags: ICollection<DiscussionTag>
Comment
+Id: Guid
+discussionId: Guid
+user: String
+discussion: Discussion
+dateOfCreation: DateTime
+content: string
+parentId: Guid
+Parent: Comment
+Replies: ICollection<Comment>
ForumUser
+departament: string
+specialization: string
+yearOfStudy: int
+description: string
+personalPageLink: string
IdentityUser

Figure 2.6 AdminController class diagram

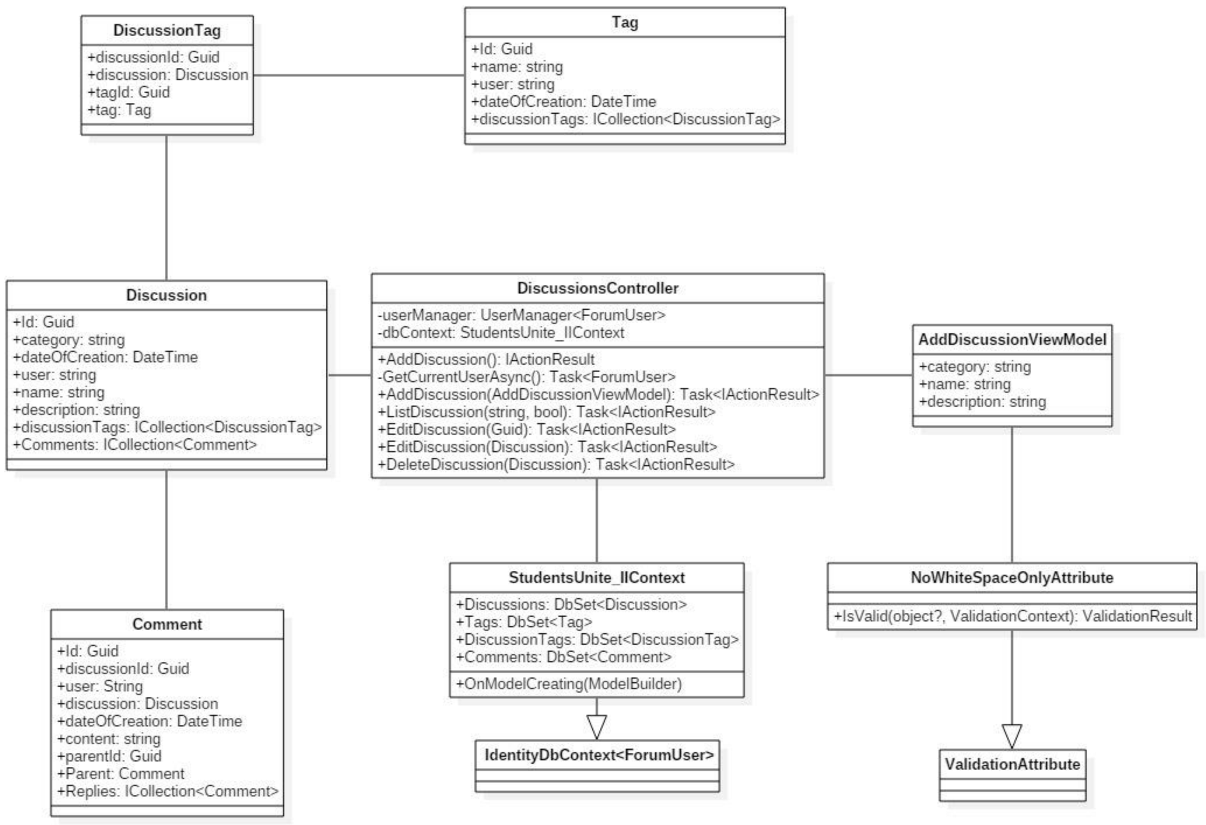

DiscussionTag
+discussionId: Guid
+discussion: Discussion
+tagId: Guid
+tag: Tag
Tag
+Id: Guid
+name: string
+user: string
+dateOfCreation: DateTime
+discussionTags: ICollection<DiscussionTag>
Discussion
+Id: Guid
+category: string
+dateOfCreation: DateTime
+user: string
+name: string
+description: string
+discussionTags: ICollection<DiscussionTag>
+Comments: ICollection<Comment>
DiscussionsController
-userManager: UserManager<ForumUser>
-dbContext: StudentsUnite_IIContext
+AddDiscussion(): IActionResult
-GetCurrentUserAsync(): Task<ForumUser>
+AddDiscussion(AddDiscussionViewModel): Task<IActionResult>
+ListDiscussion(string, bool): Task<IActionResult>
+EditDiscussion(Guid): Task<IActionResult>
+EditDiscussion(Discussion): Task<IActionResult>
+DeleteDiscussion(Discussion): Task<IActionResult>
AddDiscussionViewModel
+category: string
+name: string
+description: string
Comment
+Id: Guid
+discussionId: Guid
+user: String
+discussion: Discussion
+dateOfCreation: DateTime
+content: string
+parentId: Guid
+Parent: Comment
+Replies: ICollection<Comment>
StudentsUnite_IIContext
+Discussions: DbSet<Discussion>
+Tags: DbSet<Tag>
+DiscussionTags: DbSet<DiscussionTag>
+Comments: DbSet<Comment>
+OnModelCreating(ModelBuilder)
IdentityDbContext<ForumUser>
NoWhiteSpaceOnlyAttribute
+IsValid(object?, ValidationContext): ValidationResult
ValidationAttribute

Figure 2.7 DiscussionsController class diagram

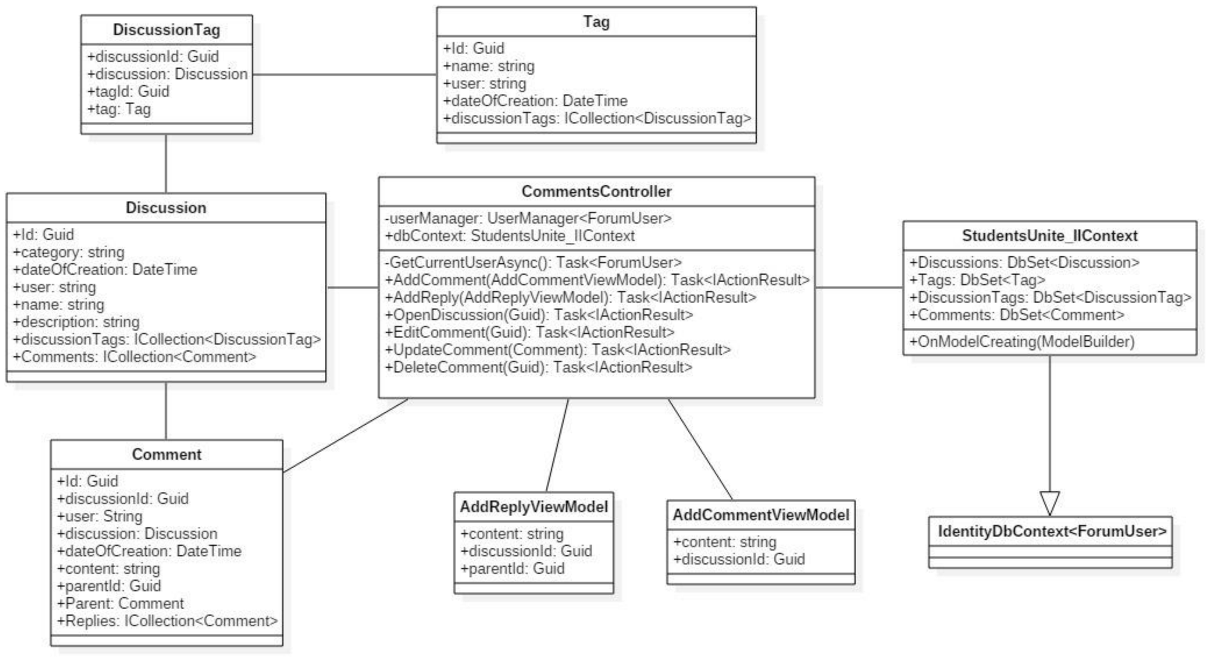

DiscussionTag
+discussionId: Guid
+discussion: Discussion
+tagId: Guid
+tag: Tag
Tag
+Id: Guid
+name: string
+user: string
+dateOfCreation: DateTime
+discussionTags: ICollection<DiscussionTag>
Discussion
+Id: Guid
+category: string
+dateOfCreation: DateTime
+user: string
+name: string
+description: string
+discussionTags: ICollection<DiscussionTag>
+Comments: ICollection<Comment>
CommentsController
-userManager: UserManager<ForumUser>
+dbContext: StudentsUnite_IIContext
-GetCurrentUserAsync(): Task<ForumUser>
+AddComment(AddCommentViewModel): Task<IActionResult>
+AddReply(AddReplyViewModel): Task<IActionResult>
+OpenDiscussion(Guid): Task<IActionResult>
+EditComment(Guid): Task<IActionResult>
+UpdateComment(Comment): Task<IActionResult>
+DeleteComment(Guid): Task<IActionResult>
StudentsUnite_IIContext
+Discussions: DbSet<Discussion>
+Tags: DbSet<Tag>
+DiscussionTags: DbSet<DiscussionTag>
+Comments: DbSet<Comment>
+OnModelCreating(ModelBuilder)
Comment
+Id: Guid
+discussionId: Guid
+user: String
+discussion: Discussion
+dateOfCreation: DateTime
+content: string
+parentId: Guid
+Parent: Comment
+Replies: ICollection<Comment>
AddReplyViewModel
+content: string
+discussionId: Guid
+parentId: Guid
AddCommentViewModel
+content: string
+discussionId: Guid
IdentityDbContext<ForumUser>

Figure 2.8 CommentsController class diagram

Database design

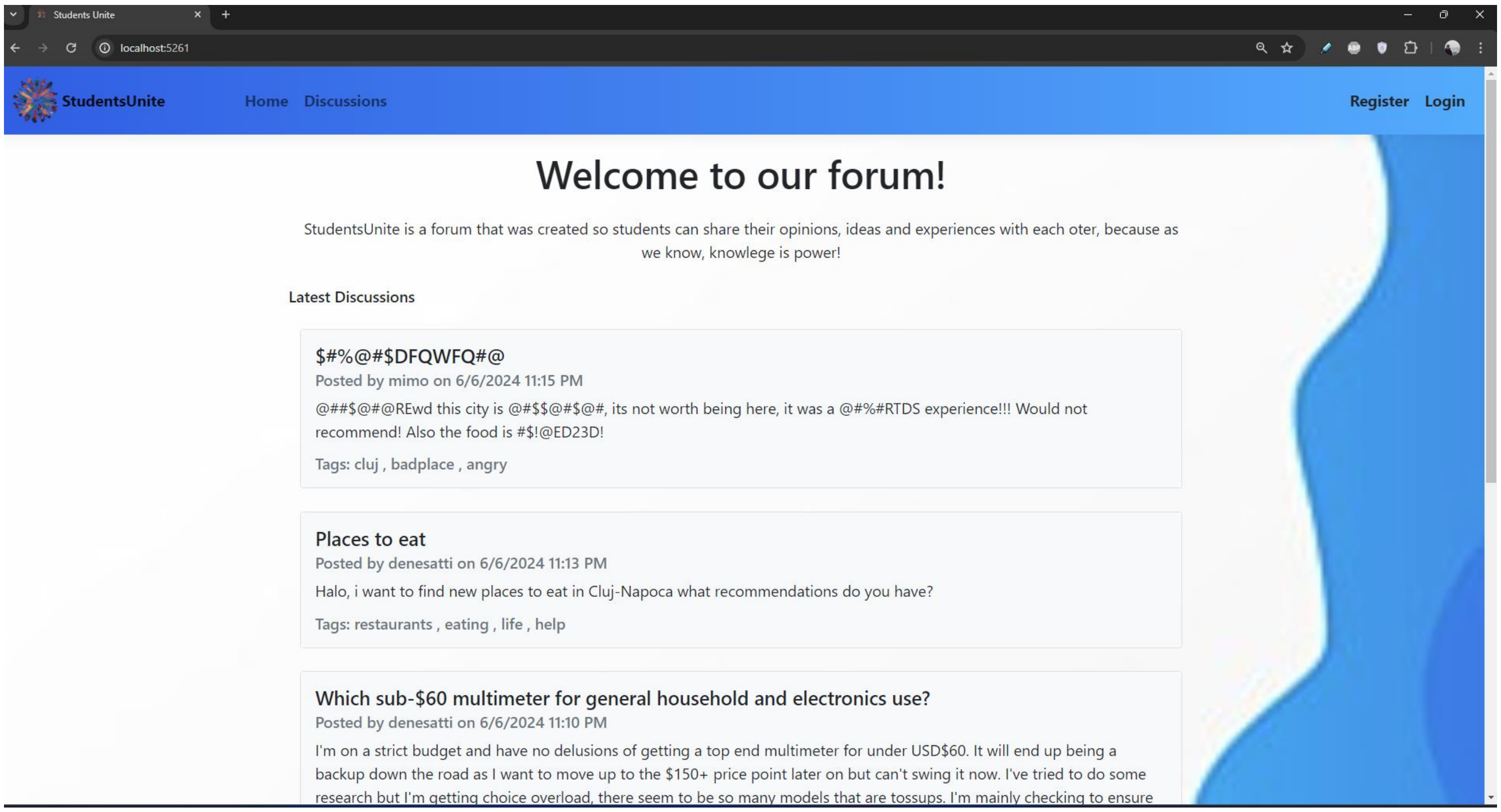
The models are abstractions which were used to make it easier to work with the database. We decided that a relational database will be ideal for this project because it is somewhat easier to implement different relationships than in NoSQL type databases, so we decided on SQL Server as our database and on a code first approach which means that we write our models first and with a helper object like an already seen DbContext interface implemented object we generate migration which are generated SQL queries with which we can update our database, meaning that running these scripts tables will be created and modified; they can even be populated with migrations. Because the database generating code will be autogenerated, annex 2. that's shows the design of the database will be autogenerated using the SQL Server Management Studio software's tool.

Chapter 3. Application implementation

The third chapter of the project's documentation will contain the actual implementation of the designs mentioned before in chapter two. We are going to present the applications implementation in two parts, firstly the we will focus on the graphical user interface and then we will focus on the actual code behind what the users see. Because some parts of the GUI and code repeats for different controllers we will explain one of them in greater detail and we will highlight if anything meaningful is different in others.

Application implementation

First of all, we will show the GUIs of a guest (not logged in) user. Figure 3.1 – 3.3 are the GUIs that a guest user can access. The rest of the GUIs are following the same design theme.



Welcome to our forum!

StudentsUnite is a forum that was created so students can share their opinions, ideas and experiences with each oter, because as we know, knowlege is power!

Latest Discussions

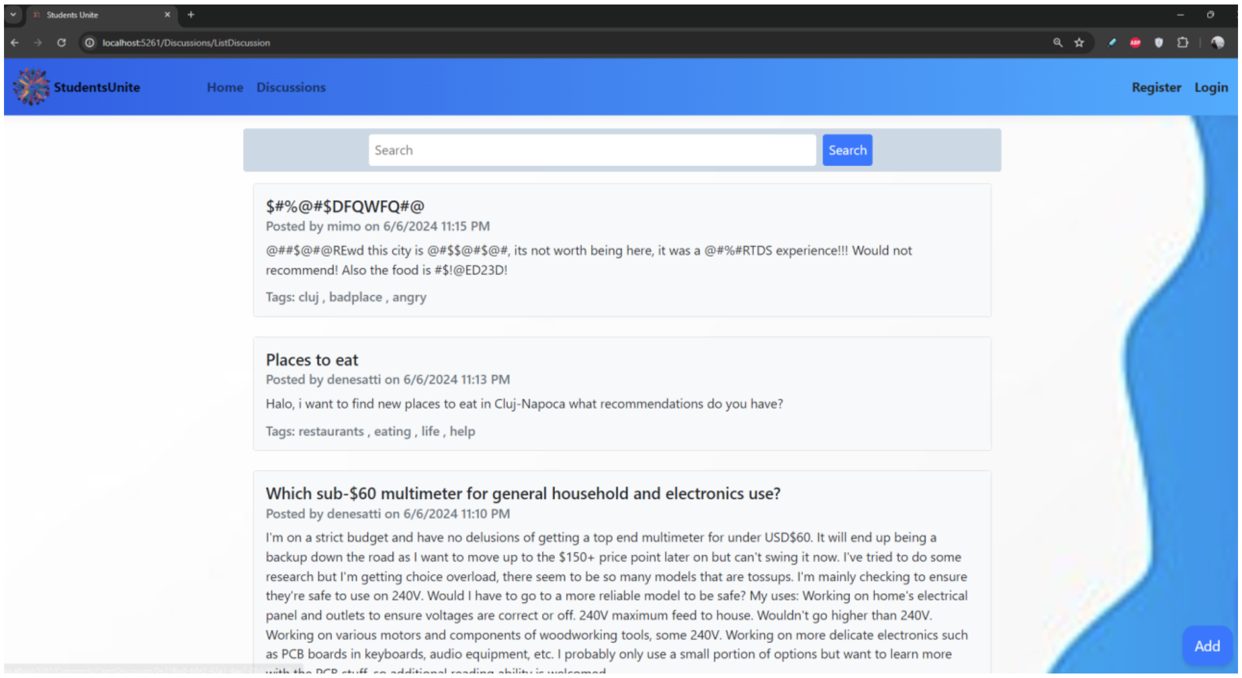
$#%@#$DFQWFQ#@
Posted by mimo on 6/6/2024 11:15 PM
@##$@#@REwd this city is @#$$@#$@#, its not worth being here, it was a @#%#RTDS experience!!! Would not recommend! Also the food is #$!@ED23D!
Tags: cluj , badplace , angry

Places to eat
Posted by denesatti on 6/6/2024 11:13 PM
Halo, i want to find new places to eat in Cluj-Napoca what recommendations do you have?
Tags: restaurants , eating , life , help

Which sub-$60 multimeter for general household and electronics use?
Posted by denesatti on 6/6/2024 11:10 PM
I'm on a strict budget and have no delusions of getting a top end multimeter for under USD$60. It will end up being a backup down the road as I want to move up to the $150+ price point later on but can't swing it now. I've tried to do some research but I'm getting choice overload, there seem to be so many models that are tossups. I'm mainly checking to ensure

Figure 3.1 Guest: Home page

The home page implements the same logic in the HomeController class that is used without the search functionality in the DiscussionsController ListDiscussion () method. That will be represented in the followings.

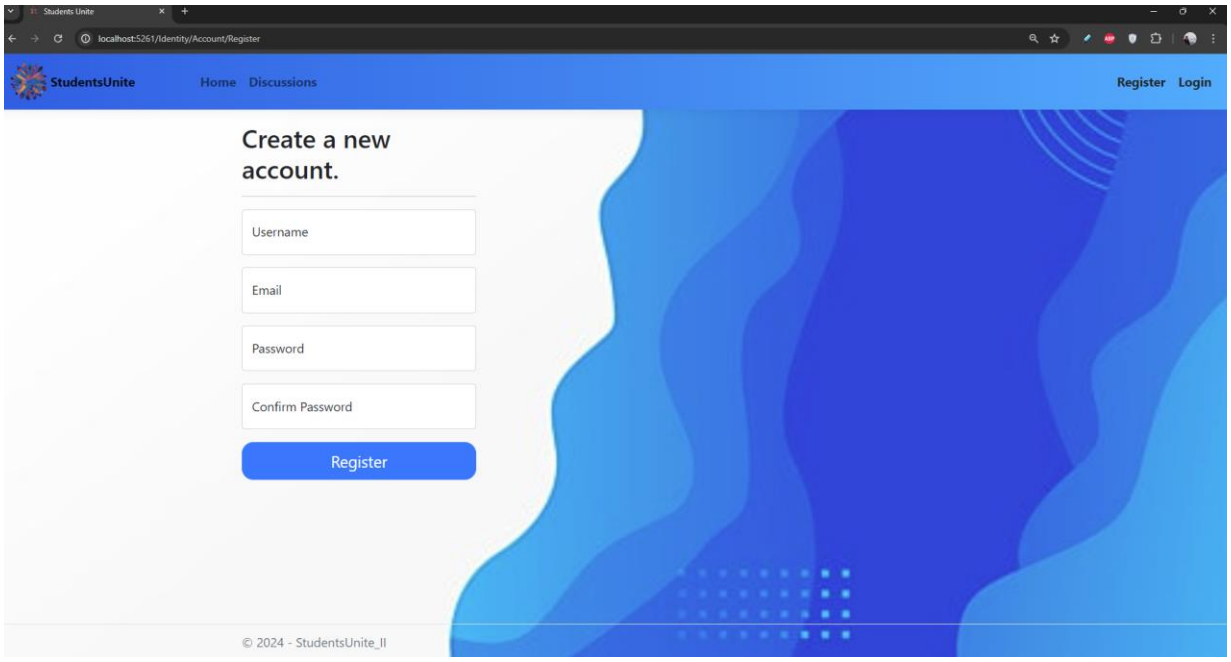


$#%@#$DFQWFQ#@
Posted by mimo on 6/6/2024 11:15 PM
@##$@#@REwd this city is @#$$@#$@#, its not worth being here, it was a @#%#RTDS experience!!! Would not recommend! Also the food is #$!@ED23D!
Tags: cluj , badplace , angry

Places to eat
Posted by denesatti on 6/6/2024 11:13 PM
Halo, i want to find new places to eat in Cluj-Napoca what recommendations do you have?
Tags: restaurants , eating , life , help

Which sub-$60 multimeter for general household and electronics use?
Posted by denesatti on 6/6/2024 11:10 PM
I'm on a strict budget and have no delusions of getting a top end multimeter for under USD$60. It will end up being a backup down the road as I want to move up to the $150+ price point later on but can't swing it now. I've tried to do some research but I'm getting choice overload, there seem to be so many models that are tossups. I'm mainly checking to ensure they're safe to use on 240V. Would I have to go to a more reliable model to be safe? My uses: Working on home's electrical panel and outlets to ensure voltages are correct or off. 240V maximum feed to house. Wouldn't go higher than 240V. Working on various motors and components of woodworking tools, some 240V. Working on more delicate electronics such as PCB boards in keyboards, audio equipment, etc. I probably only use a small portion of options but want to learn more

Figure 3.2 Guest: Discussions page

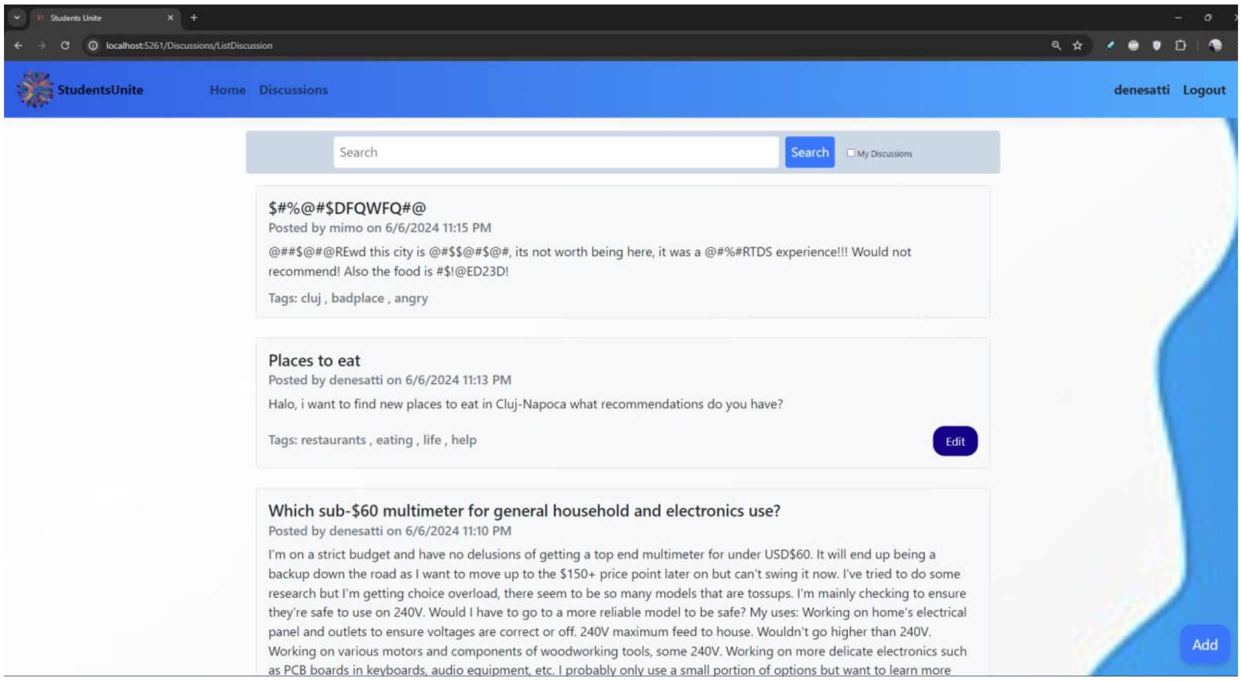


Create a new account.

© 2024 - StudentsUnite_II

Figure 3.3 Guest: Register page

Now that we illustrated what a guest can do, we will show the difference between a guest GUI and an authenticated users GUI. Figure 3.4 shows the discussions page, where we can see the first difference next to the search bar, also if we have a discussion that we created or we are in admins role next to every discussion the edit button, but that’s not all, the way we can interact with a discussion, we can also open discussions by simply clicking on their body, this operation can be accessed by every types of actors, even guests, but the authenticated user can also further interact with the discussion, by adding a comment or replying on an already existing comment, this will be shown in figure 3.5.

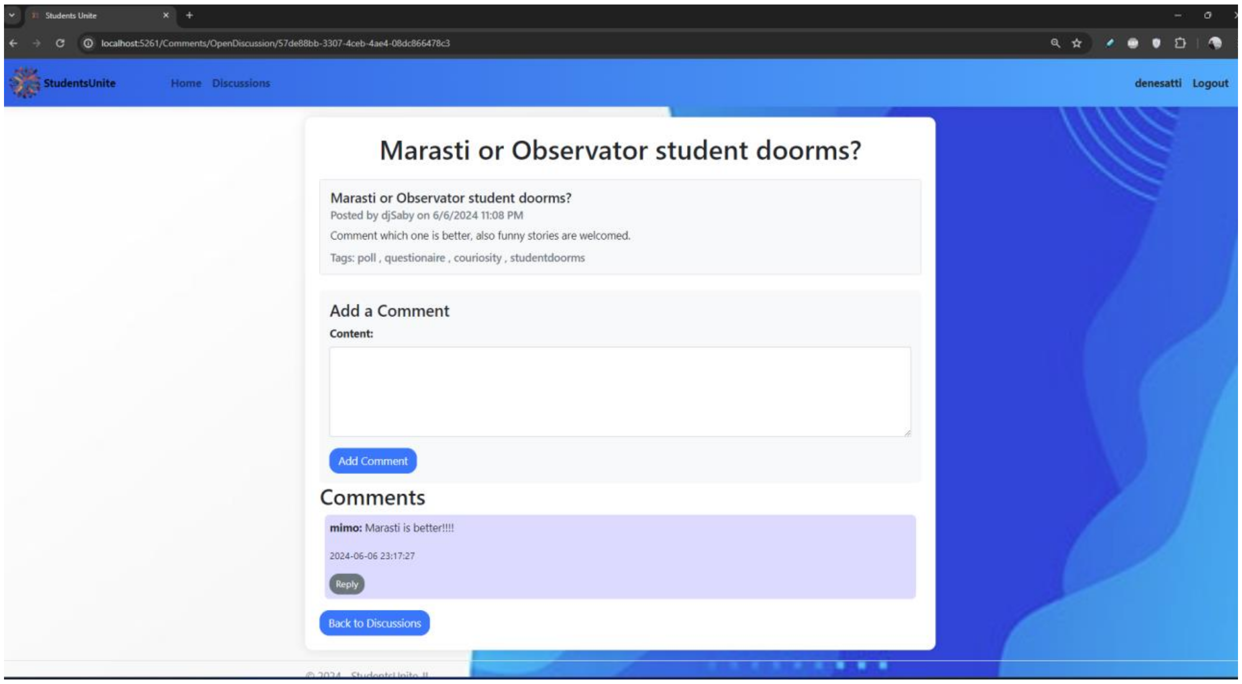


$#%@#$DFQWFQ#@
Posted by mimo on 6/6/2024 11:15 PM
@##$@#@REwd this city is @#$$@#$@#, its not worth being here, it was a @#%#RTDS experience!!! Would not recommend! Also the food is #$!@ED23D!
Tags: cluj , badplace , angry

Places to eat
Posted by denesatti on 6/6/2024 11:13 PM
Halo, i want to find new places to eat in Cluj-Napoca what recommendations do you have?
Tags: restaurants , eating , life , help

Which sub-$60 multimeter for general household and electronics use?
Posted by denesatti on 6/6/2024 11:10 PM
I'm on a strict budget and have no delusions of getting a top end multimeter for under USD$60. It will end up being a backup down the road as I want to move up to the $150+ price point later on but can't swing it now. I've tried to do some research but I'm getting choice overload, there seem to be so many models that are tossups. I'm mainly checking to ensure they're safe to use on 240V. Would I have to go to a more reliable model to be safe? My uses: Working on home's electrical panel and outlets to ensure voltages are correct or off. 240V maximum feed to house. Wouldn't go higher than 240V. Working on various motors and components of woodworking tools, some 240V. Working on more delicate electronics such as PCB boards in keyboards, audio equipment, etc. I probably only use a small portion of options but want to learn more

Figure 3.4 Authenticated user: Discussions Page


Students Unite

localhost:5261/Comments/OpenDiscussion/57de88bb-3307-4ceb-4ae4-08dc866478c3

StudentsUnite Home Discussions denesatti Logout

Marasti or Observator student doorms?

Marasti or Observator student doorms?
Posted by djSaby on 6/6/2024 11:08 PM
Comment which one is better, also funny stories are welcomed.
Tags: poll , questionaire , couriosity , studentdoorms

Add a Comment
Content:

Add Comment

Comments

mimo: Marasti is better!!!!
2024-06-06 23:17:27
Reply

Back to Discussions

Figure 3.5 Authenticated user: Opened Discussion (with 80% zoom)

We can also add our own discussion, the GUI for it will be shown on figure 3.6. Another important part of a forum application is the users page, unfortunately we did not implement a feature that allows users to view other users' information, but every user can add and update information about them in their profile, that is accessed by clicking on the authenticated user's username. Figure 3.7 will show us the mentioned profile page, that can also be used to set a new password or delete the account.

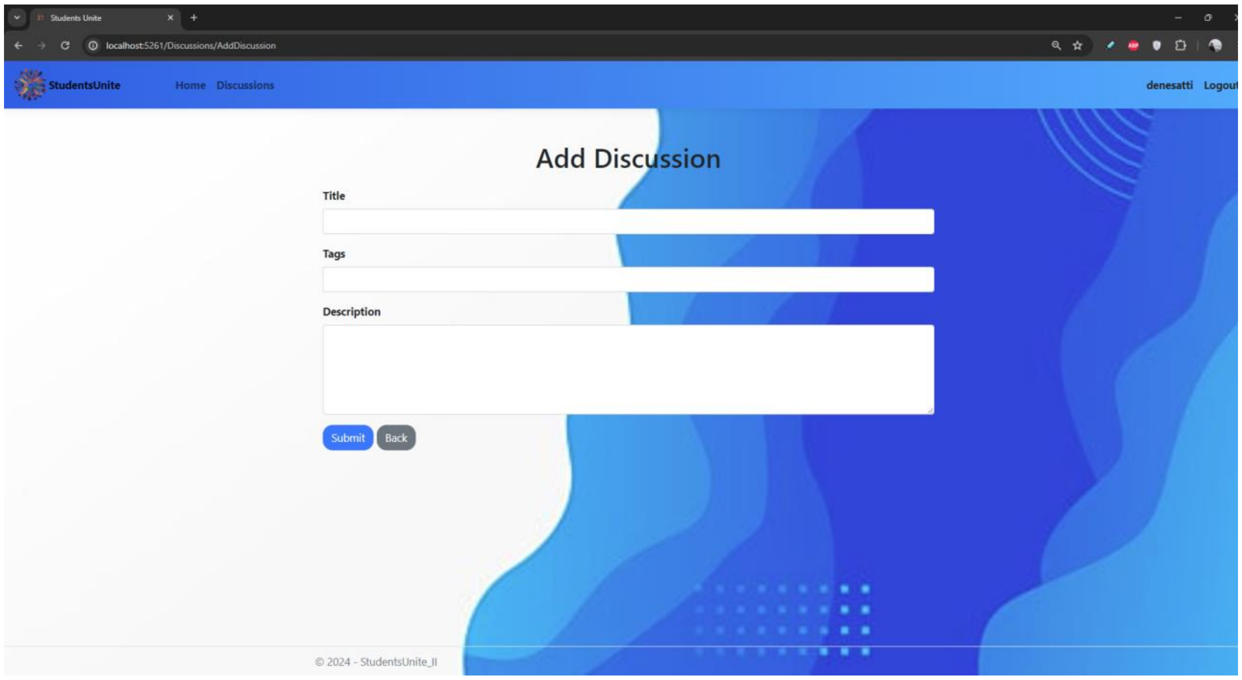
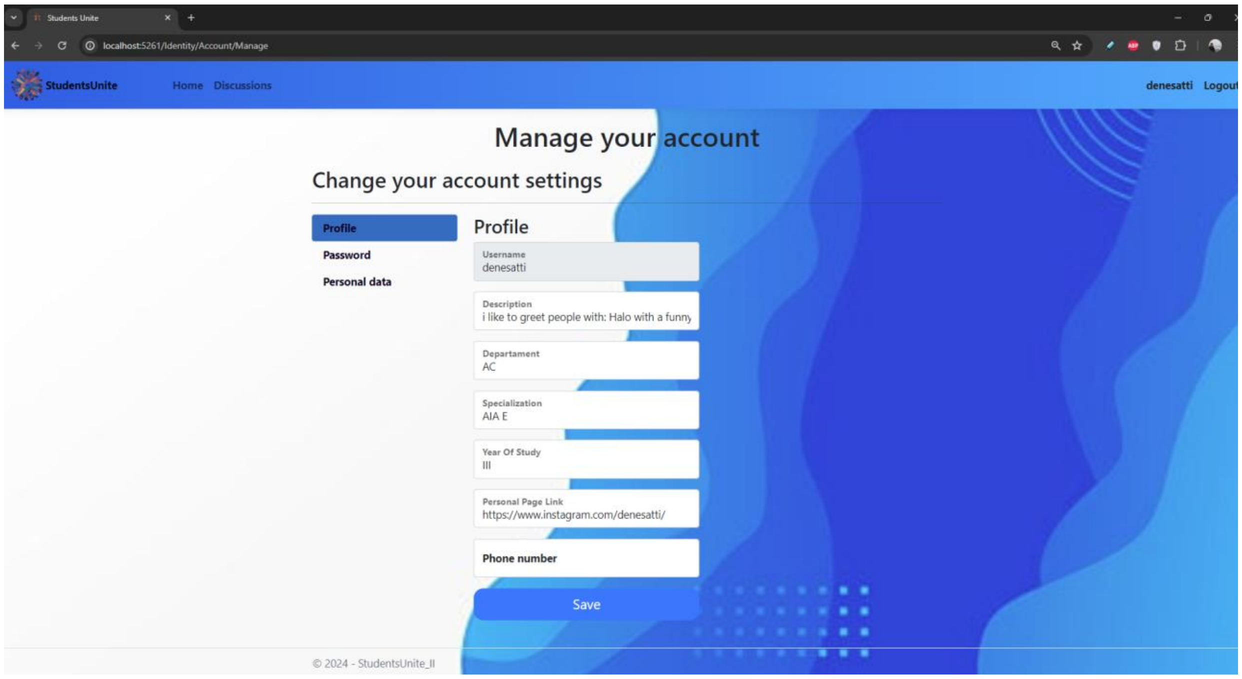


Figure 3.6 Add Discussion Page



Username
denesatti

Description
i like to greet people with: Halo with a funny

Departament
AC

Specialization
AIA E

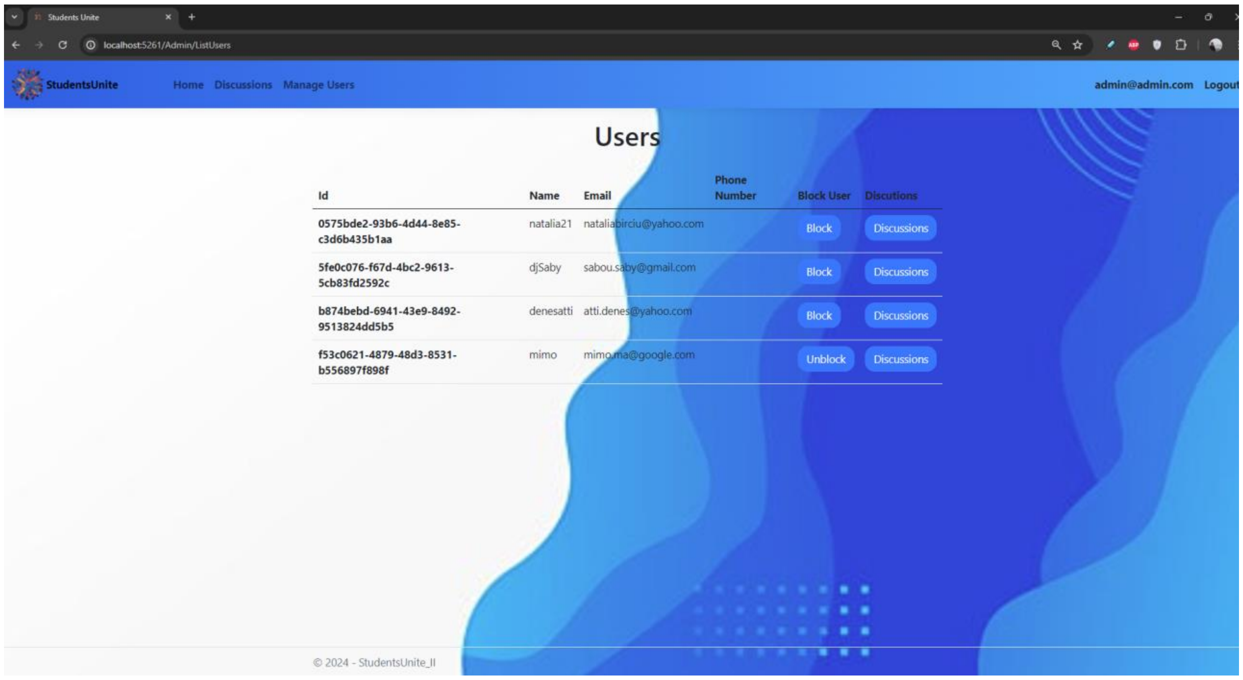
Year Of Study
III

Personal Page Link
https://www.instagram.com/denesatti/

Phone number

Figure 3.7 Profile Page

But not only the basic users should have features but also admins. We have admin functionalities implemented. An admin is a user with a special role. This role enables the admin to block or unblock users that stops them form signing in. Furthermore, the admins have right to edit and delete discussions but also to delete comments and replies. Figure 3.8 and figure 3.9 will illustrate the different features of the admin user.

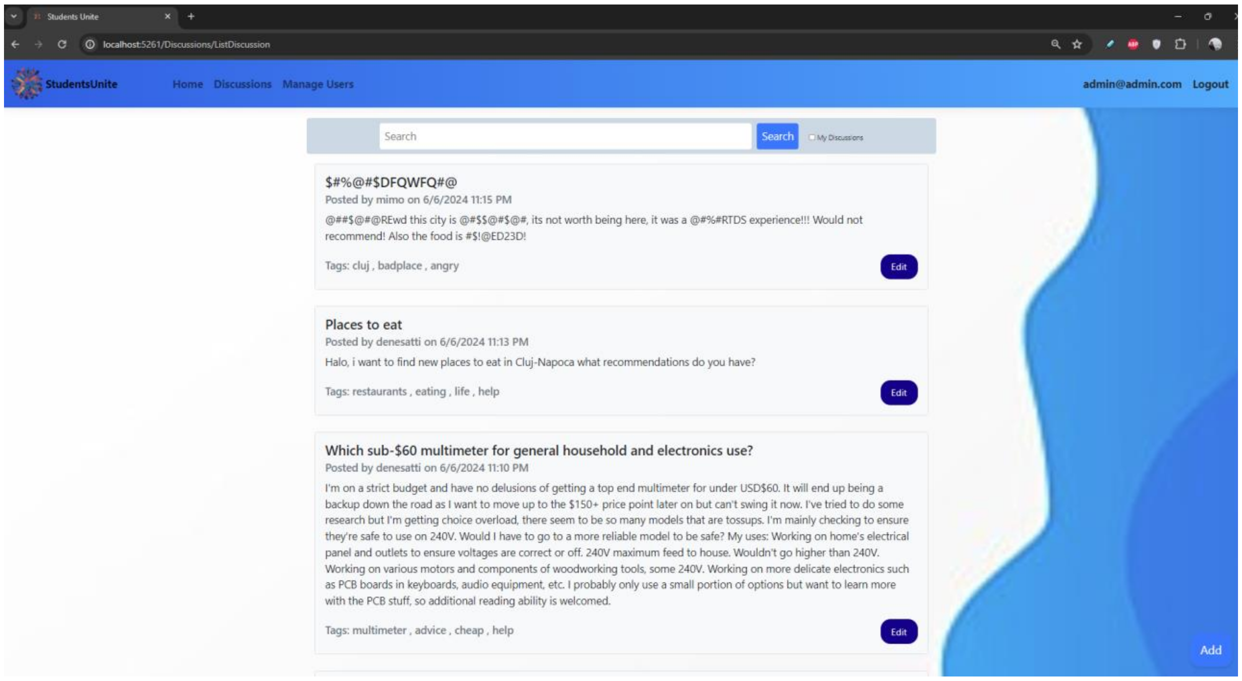


Users

Id	Name	Email	Phone Number	Block User	Discutions
0575bde2-93b6-4d44-8e85-c3d6b435b1aa	natalia21	nataliabirciu@yahoo.com		Block	Discussions
5fe0c076-f67d-4bc2-9613-5cb83fd2592c	djSaby	sabou.saby@gmail.com		Block	Discussions
b874bebd-6941-43e9-8492-9513824dd5b5	denesatti	atti.denes@yahoo.com		Block	Discussions
f53c0621-4879-48d3-8531-b556897f898f	mimo	mimo.ma@google.com		Unblock	Discussions

© 2024 - StudentsUnite_II

Figure 3.8 Manage Users Page



$#%@#$DFQWFQ#@
Posted by mimo on 6/6/2024 11:15 PM
@##$@#@REwd this city is @#$$@#$@#, its not worth being here, it was a @#%#RTDS experience!!! Would not recommend! Also the food is #$!@ED23D!
Tags: cluj , badplace , angry

Places to eat
Posted by denesatti on 6/6/2024 11:13 PM
Halo, i want to find new places to eat in Cluj-Napoca what recommendations do you have?
Tags: restaurants , eating , life , help

Which sub-$60 multimeter for general household and electronics use?
Posted by denesatti on 6/6/2024 11:10 PM
I'm on a strict budget and have no delusions of getting a top end multimeter for under USD$60. It will end up being a backup down the road as I want to move up to the $150+ price point later on but can't swing it now. I've tried to do some research but I'm getting choice overload, there seem to be so many models that are tossups. I'm mainly checking to ensure they're safe to use on 240V. Would I have to go to a more reliable model to be safe? My uses: Working on home's electrical panel and outlets to ensure voltages are correct or off. 240V maximum feed to house. Wouldn't go higher than 240V. Working on various motors and components of woodworking tools, some 240V. Working on more delicate electronics such as PCB boards in keyboards, audio equipment, etc. I probably only use a small portion of options but want to learn more with the PCB stuff, so additional reading ability is welcomed.
Tags: multimeter , advice , cheap , help

Figure 3.9 Authenticated as Admin: Discussions Page

In the following we will show the DiscussionsController class so that we can better understand the workings of the controllers:

```
public class DiscussionsController : Controller
{
    private readonly UserManager<ForumUser> _userManager;
    private readonly StudentsUnite_IIContext dbContext;
    public DiscussionsController(
        StudentsUnite_IIContext dbContext,
        UserManager<ForumUser> userManager
        )
    {
        _userManager = userManager;
        this.dbContext = dbContext;
    }

    [HttpGet]
    [Authorize]
    public IActionResult AddDiscussion()
    {
        return View();
    }

    private Task<ForumUser> GetCurrentUserAsync() =>
_userManager.GetUserAsync(HttpContext.User);

    [HttpPost]
    [Authorize]
    public async Task<IActionResult> AddDiscussion(AddDiscussionViewModel
addDiscussionViewModel)
    {
        if (ModelState.IsValid) {
            var tagNames = addDiscussionViewModel.category.TrimStart('
').Split(',');
            var tags = new List<Tag>();
            foreach(var tagName in tagNames)
            {
                var tagNameTrimmed = tagName.Trim();
                if (!tagNameTrimmed.IsNullOrEmpty())
                {
                    var tag = await dbContext.Tags.FirstOrDefaultAsync(t => t.name
== tagNameTrimmed);
                    if (tag == null)
                    {
                        tag = new Tag
                        {
                            name = tagNameTrimmed,
                            dateOfCreation = DateTime.Now,
                            user = await
_userManager.GetUserNameAsync(GetCurrentUserAsync().Result)

                        };
                        await dbContext.Tags.AddAsync(tag);
                    }
                    tags.Add(tag);
                }

            }
            var discussion = new Discussion
            {
                name = addDiscussionViewModel.name.TrimStart(' '),
                category = string.Join(" , ",tagNames),
                description = addDiscussionViewModel.description.TrimStart(' '),
                dateOfCreation = DateTime.Now,
                user = await
_userManager.GetUserNameAsync(GetCurrentUserAsync().Result)
                };

            await dbContext.Discussions.AddAsync(discussion);
            await dbContext.SaveChangesAsync();
            foreach(var tag in tags)
            {
               var discussionTag = new DiscussionTag
                {
                    discussionId = discussion.Id,
                    tagId = tag.Id
                };
                await dbContext.AddAsync(discussionTag);
            }
            await dbContext.SaveChangesAsync();
            return RedirectToAction("ListDiscussion", "Discussions");
        }
        else {
            return View(addDiscussionViewModel);
        }

    }


    [HttpGet]
    public async Task<IActionResult> ListDiscussion(string searchString, bool
myDiscussions = false)
    {
        var discussions = await dbContext.Discussions.OrderByDescending(d =>
d.dateOfCreation).ToListAsync();

        if (!String.IsNullOrEmpty(searchString))
        {
            discussions = await dbContext.Discussions.Where(d =>
d.name.Contains(searchString)).OrderByDescending(d =>
d.dateOfCreation).ToListAsync();

            if (myDiscussions)
            {
                var userName = await
_userManager.GetUserNameAsync(GetCurrentUserAsync().Result);
                discussions = await dbContext.Discussions.Where(d =>
d.name.Contains(searchString) && d.user == userName).OrderByDescending(d =>
d.dateOfCreation).ToListAsync();
            }
        }

        if (myDiscussions)
        {
            var userName = await
_userManager.GetUserNameAsync(GetCurrentUserAsync().Result);
            discussions = await dbContext.Discussions.Where(d=>  d.user ==
userName).OrderByDescending(d => d.dateOfCreation).ToListAsync();
        }
        ViewData["CurrentFilter"] = searchString;
        ViewData["MyDiscussions"] = myDiscussions;

        return View(discussions);
    }

    [HttpGet]
    [Authorize]
    public async Task<IActionResult> EditDiscussion(Guid Id)
    {
        var discussion = await dbContext.Discussions.FindAsync(Id);
        return View(discussion);
    }

    [HttpPost]
    [Authorize]
    public async Task<IActionResult> EditDiscussion(Discussion viewModel)
    {
        var discussion = await dbContext.Discussions.FindAsync(viewModel.Id);
        string[] tagNames;
        var tags = new List<Tag>();

        if (discussion != null)
        {
            discussion.name = viewModel.name;

            if (discussion.description.Contains("[Edited]"))
            {
                discussion.description = viewModel.description;
            }
            else
            {
                 discussion.description = "[Edited]\n" + viewModel.description;
            }

            discussion.category = viewModel.category;

            tagNames = viewModel.category.Split(',');

            foreach (var tagName in tagNames)
            {
                var tagNameTrimmed = tagName.Trim();
                if (!tagNameTrimmed.IsNullOrEmpty())
                {
                    var tag = await dbContext.Tags.FirstOrDefaultAsync(t => t.name
== tagNameTrimmed);
                    if (tag == null)
                    {
                        tag = new Tag
                        {
                            name = tagNameTrimmed,
                            dateOfCreation = DateTime.Now,
                            user = await
_userManager.GetUserNameAsync(GetCurrentUserAsync().Result)
```

```
                        };
                        await dbContext.Tags.AddAsync(tag);
                    }
                    tags.Add(tag);
                }

            }

            var disussionTagsToRemove = await dbContext.DiscussionTags.Where(dt =>
dt.discussionId == discussion.Id).ToListAsync();

            dbContext.DiscussionTags.RemoveRange(disussionTagsToRemove);

            foreach (var tag in tags)
            {
                var discussionTag = new DiscussionTag
                {
                    discussionId = discussion.Id,
                    tagId = tag.Id
                };
                await dbContext.AddAsync(discussionTag);
            }

            await dbContext.SaveChangesAsync();
        }


        return RedirectToAction("ListDiscussion", "Discussions");
    }

    [HttpPost]
    public async Task<IActionResult> DeleteDiscussion(Discussion viewModel)
    {
        var discussion = await
dbContext.Discussions.AsNoTracking().FirstOrDefaultAsync(x => x.name ==
viewModel.name);

        if (discussion != null)
        {
            dbContext.Discussions.Remove(discussion);
            await dbContext.SaveChangesAsync();
        }

        return RedirectToAction("ListDiscussion", "Discussions");
    }

}
```

We can observe that some method has the same name, that is because we have POST and GET methods that are called on the same functions, these functions are overloaded. The POST method retrieves information from the view, while the GET method returns for the view a model. We have models that makes us easier to work with the database but most of the time we do not want to send all the information to the view we have about a model, so we send a special model called view model which is a stripped version of one of our models. We will give an example in the following:

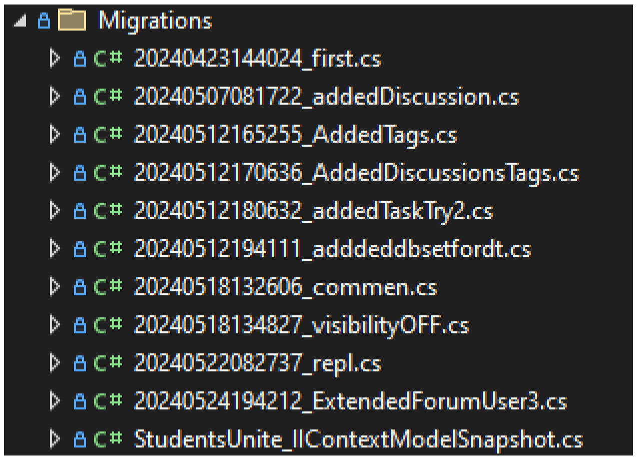
```
public class Discussion
{
    public Guid Id { get; set; }


    public string category { get; set; }

    [DataType(DataType.DateTime)]
    public DateTime dateOfCreation { get; set; }

    [ForeignKey("Id")] public string user { get; set; }

    public string name { get; set; }

    public string description { get; set; }

    public ICollection<DiscussionTag> discussionTags { get; set; }
    public ICollection<Comment> Comments { get; set; }

}
```

```
public class AddDiscussionViewModel
{
    [Required]
    [NoWhiteSpaceOnly]
    public string category { get; set; }

    [Required]
    [NoWhiteSpaceOnly]
    public string name { get; set; }

    [Required]
    [NoWhiteSpaceOnly]
    public string description { get; set; }

}
```

Models make it more straight forward to work with data, which is mostly transported between the database and the controller. In the following we will talk about the database's implementation.

Database implementation

We decided on a code first approach, so our database was constructed using built in tools from Microsoft's ASP.Net. We were adding different migration as we constructed our project and updated our database with it, figure 3.10 shows the migrations we used, some times we had an error and we manually needed to manipulate the database and the migrations became corrupt in a way, we needed to modify them so they work as intended.



```
Migrations
  20240423144024_first.cs
  20240507081722_addedDiscussion.cs
  20240512165255_AddedTags.cs
  20240512170636_AddedDiscussionsTags.cs
  20240512180632_addedTaskTry2.cs
  20240512194111_adddeddbsetfordt.cs
  20240518132606_commen.cs
  20240518134827_visibilityOFF.cs
  20240522082737_repl.cs
  20240524194212_ExtendedForumUser3.cs
  StudentsUnite_IIContextModelSnapshot.cs
```

Figure 3.11 Migrations

Chapter 4. Application testing

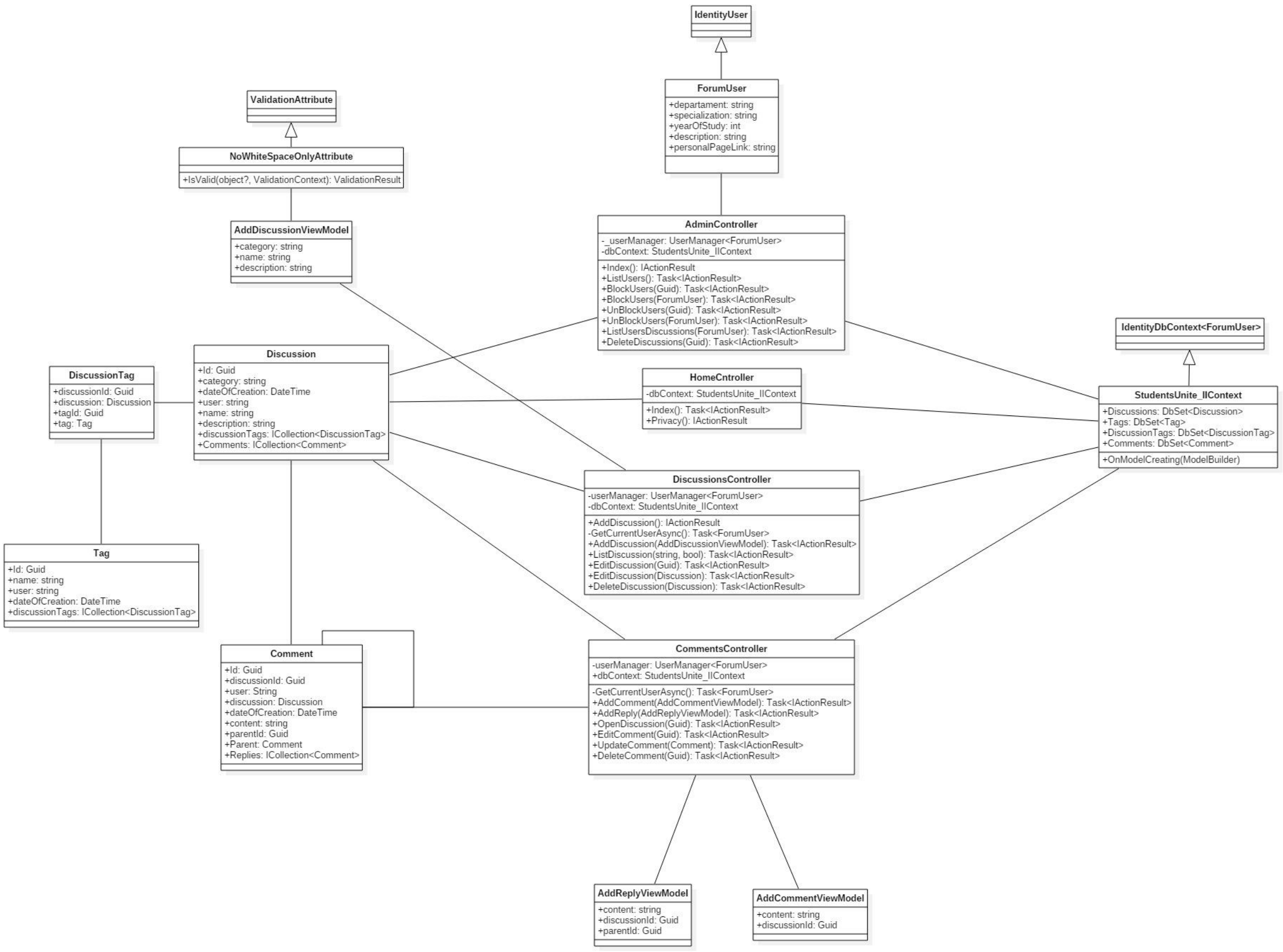
After we started to have reasonable progress in the development phase, we begin testing our components along the way to ensure that they work and work how we want to, although we might have a good eye for subtill changes we can not anticipate that a user would use our application as we intended to, that why our tester, Deniz, occupied with breaking the application so we can know for sure that it could perform in the real world, if we decide to deploy it of course. With the help of Deniz we encountered luckily only a few bugs in our application so we did not have to do a lot of bug fixes. He came up with a table system to write down each test he made, he did manual testing, he wrote in details the situation of the tests, that is where, what, when, after what, so we could reproduce them which is a key step in quick and easy bug fixing. This table will be attached to annex 3. We can observe in the attached table that every test passed, that is because we resolved the respective bug if we found one and he re did the respective that and the application passed it.

Chapter 5. Conclusions

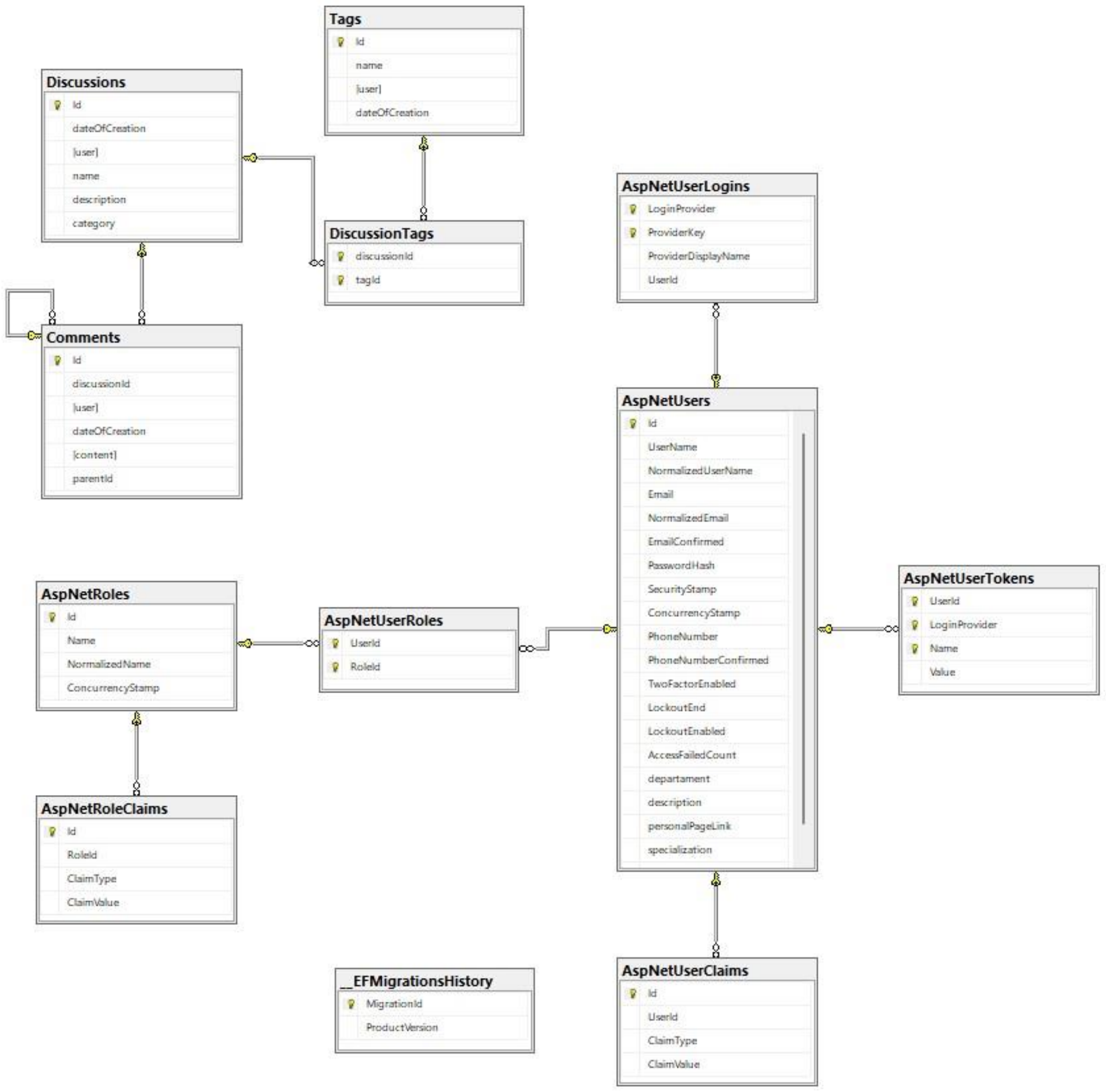
As we can see we implemented the basic functionalities for a forum app, the desired features were mostly met, but there is always room for improvement. In our opinion this project was a nice way to begin to understand what it means to work in a team, to work with new technologies, to write code that somebody else will reuse or build upon. All in all, we learned a lot and are happy with our application.

Annexes

Annex 1.


IdentityUser
ForumUser
+departament: string
+specialization: string
+yearOfStudy: int
+description: string
+personalPageLink: string
ValidationAttribute
NoWhiteSpaceOnlyAttribute
+IsValid(object?, ValidationContext): ValidationResult
AddDiscussionViewModel
+category: string
+name: string
+description: string
AdminController
-_userManager: UserManager<ForumUser>
-dbContext: StudentsUnite_IIContext
+Index(): IActionResult
+ListUsers(): Task<IActionResult>
+BlockUsers(Guid): Task<IActionResult>
+BlockUsers(ForumUser): Task<IActionResult>
+UnBlockUsers(Guid): Task<IActionResult>
+UnBlockUsers(ForumUser): Task<IActionResult>
+ListUsersDiscussions(ForumUser): Task<IActionResult>
+DeleteDiscussions(Guid): Task<IActionResult>
IdentityDbContext<ForumUser>
DiscussionTag
+discussionId: Guid
+discussion: Discussion
+tagId: Guid
+tag: Tag
Discussion
+Id: Guid
+category: string
+dateOfCreation: DateTime
+user: string
+name: string
+description: string
+discussionTags: ICollection<DiscussionTag>
+Comments: ICollection<Comment>
HomeCntroller
-dbContext: StudentsUnite_IIContext
+Index(): Task<IActionResult>
+Privacy(): IActionResult
StudentsUnite_IIContext
+Discussions: DbSet<Discussion>
+Tags: DbSet<Tag>
+DiscussionTags: DbSet<DiscussionTag>
+Comments: DbSet<Comment>
+OnModelCreating(ModelBuilder)
DiscussionsController
-userManager: UserManager<ForumUser>
-dbContext: StudentsUnite_IIContext
+AddDiscussion(): IActionResult
-GetCurrentUserAsync(): Task<ForumUser>
+AddDiscussion(AddDiscussionViewModel): Task<IActionResult>
+ListDiscussion(string, bool): Task<IActionResult>
+EditDiscussion(Guid): Task<IActionResult>
+EditDiscussion(Discussion): Task<IActionResult>
+DeleteDiscussion(Discussion): Task<IActionResult>
Tag
+Id: Guid
+name: string
+user: string
+dateOfCreation: DateTime
+discussionTags: ICollection<DiscussionTag>
Comment
+Id: Guid
+discussionId: Guid
+user: String
+discussion: Discussion
+dateOfCreation: DateTime
+content: string
+parentId: Guid
+Parent: Comment
+Replies: ICollection<Comment>
CommentsController
-userManager: UserManager<ForumUser>
+dbContext: StudentsUnite_IIContext
-GetCurrentUserAsync(): Task<ForumUser>
+AddComment(AddCommentViewModel): Task<IActionResult>
+AddReply(AddReplyViewModel): Task<IActionResult>
+OpenDiscussion(Guid): Task<IActionResult>
+EditComment(Guid): Task<IActionResult>
+UpdateComment(Comment): Task<IActionResult>
+DeleteComment(Guid): Task<IActionResult>
AddReplyViewModel
+content: string
+discussionId: Guid
+parentId: Guid
AddCommentViewModel
+content: string
+discussionId: Guid

Annex 2.


Tags
Id
name
[user]
dateOfCreation
Discussions
Id
dateOfCreation
[user]
name
description
category
DiscussionTags
discussionId
tagId
AspNetUserLogins
LoginProvider
ProviderKey
ProviderDisplayName
UserId
Comments
Id
discussionId
[user]
dateOfCreation
[content]
parentId
AspNetUsers
Id
UserName
NormalizedUserName
Email
NormalizedEmail
EmailConfirmed
PasswordHash
SecurityStamp
ConcurrencyStamp
PhoneNumber
PhoneNumberConfirmed
TwoFactorEnabled
LockoutEnd
LockoutEnabled
AccessFailedCount
departament
description
personalPageLink
specialization
AspNetRoles
Id
Name
NormalizedName
ConcurrencyStamp
AspNetUserRoles
UserId
RoleId
AspNetUserTokens
UserId
LoginProvider
Name
Value
AspNetRoleClaims
Id
RoleId
ClaimType
ClaimValue
__EFMigrationsHistory
MigrationId
ProductVersion
AspNetUserClaims
Id
UserId
ClaimType
ClaimValue

Annex 3.

test case id	objective	description/steps	expected result	actual result	status
Tc1	Validate login with valid credentials.	Open the login page. Enter a valid username and password. Click on the login button.	User is redirected to the home page and the user's name is displayed.	User is redirected to the home page and the user name is displayed	Test passed
Tc2	Validate login with invalid username.	Open the login page. Enter an invalid username and a valid password. Click on the login button.	An error message is displayed indicating that the username is incorrect, and the user remains on the login page.	An error message is displayed indicating that the username is incorrect, and the user remains on the login page.	Test passed
Tc3	Validate login with invalid password	Open the login page. Enter a valid username and an invalid password. Click on the login button.	An error message is displayed indicating that the password is incorrect, and the user remains on the login page.	An error message is displayed indicating that the password is incorrect, and the user remains on the login page.	Test passed
Tc4	Validate login with empty username and password fields.	Open the login page. Leave the username and password fields empty. Click on the login button.	Error messages are displayed indicating that both fields are required, and the user remains on the login page.	Error messages are displayed indicating that both fields are required, and the user remains on the login page.	Test passed
Tc5	Validate the logout functionality.	Log in to the system using valid credentials. Verify that the user is redirected to the home page. Click on the logout button/link.	The user is logged out and redirected to the login page. Attempting to access any authenticated page should redirect the user to the login page.	The user is logged out and redirected to the login page. Attempting to access any authenticated page should redirect the user	Test passed
Tc6	Validate changing the password of the user	Log in to the account. Go to profile. Click "change password" and enter the new one.	User's password is changed. Login works with the new password.	User's password is changed. Login works with the new password.	Test passed
Tc7	Validate adding a new discussion with the title containing just empty spaces	Go the the discussion page and click "Add". For the title /description/tags , put only empty spaces, the rest doesn't matter. Then click add	The page show a message regarding the title ,to enter a valid title / description / tags .	The page show a message regarding the title ,to enter a valid title / description / tags .	Test passed
Tc8	Validate changing the details of the profile(department, phone number etc)	Log in to the account. Go to profile. Edit the details you want to change and click Save.	The new details are saved and updated.	The new details are saved and updated.	Test passed
Tc9	Validate removing an account // Deleting an account with all its details	Being logged in, go to Profile tab. Click on Personal Data and click Delete. Enter the password and press Delete again.	It logges out , the account is no longer existing, or any of its post/details.	It logges out , the account is no longer existing, or any of its post/details.	Test passed
Tc10	Validate registering an account with the details containing empty spaces.	Click on register, and enter empty spaces for username/ password, etc.	The page show a message to enter a valid password/username.	The page show a message to enter a valid password/username.	Test passed