

A Real-Time Web-Based Academic Discussion Platform

Mariya Zhekova¹, Teodosi Kehayov¹, Silvia Gaftandzhieva²

¹ University of Food Technology, 26 Maritsa Blvd, 4000 Plovdiv, Bulgaria

² University of Plovdiv "Paisii Hilendarski", 24 Tzar Assen Str, Plovdiv, Bulgaria

Abstract – The paper presents the development of a web-based academic platform designed to discuss and find answers to academic questions, where users can create topics in various admin-approved categories, comment on topics created by other users, and customize their profile with a biography, photo, and free text description. To achieve this goal, the logical processes of creating and managing a discussion forum were studied, the existing techniques and technologies for creating web apps were analysed, tools for the implementation were chosen, the database was built, and the client interface was made, as well as the functionality of the system and numerous tests and experiments. The application has been tested by students and teachers and the conducted experiments demonstrate the main functionalities of the application and prove its reliability. The research facilitates and encourages research partnerships at personal and university levels; disseminates ideas and solutions to a wider audience; creates a network of scholars and trainees for interaction and discusses research problems, experiences, and good practices; encourages collaboration on common projects; supports young scholars in achieving their academic goals, and offers open-access resources for reading, downloading, sharing, etc.

Keywords – Discussion platform, academic forum, information exchange, discussions.

DOI: 10.18421/TEM131-62

<https://doi.org/10.18421/TEM131-62>

Corresponding author: Mariya Zhekova,
University of Food Technology, 26 Maritsa Blvd, 4000
Plovdiv, Bulgaria


Email: m.jekova@uft-plovdiv.bg

Received: 11 October 2023.

Revised: 19 January 2024.

Accepted: 25 January 2024.

Published: 27 February 2024.

 © 2024 Mariya Zhekova, Teodosi Kehayov & Silvia Gaftandzhieva; published by UIKTEN. This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 4.0 License.

The article is published with Open Access at
<https://www.temjournal.com/>

1. Introduction

The continuous increase in the volume of information in recent years, as well as the need for educational interaction, sharing, discussion and interdisciplinary exchange, encourages scientists to create collaborative spaces in which they can freely exchange ideas, experiences, recommendations, and good practices.

By creating opportunities for dialogue between academics and users of educational services – primarily learners (students, adolescents) – a research approach serving as a bridge for cooperation and development in the field of higher education is provided, in order to create solutions needed in the rapidly evolving global network. This exchange of information takes place through academic events, conferences, and inclusive centres, but for the most part, it takes place virtually in online public spaces (discussion forums).

The article presents the process of creating a real-time web-based discussion platform (forum) for exchanging ideas, asking questions, and collaborating on projects that benefit students and teachers. In the application, which is a kind of academic forum, users can generate and share new knowledge, facilitate their work, direct discussions in a desired direction, solve problems, and find answers to their questions on a wide range of topics. The forum could contain topics, opinions and categories (groups) of academic subjects, host online meetings and events and discuss topics and materials relevant to researchers.

The research aims to create a real-time online discussion platform that will:

- facilitate and encourage research partnerships at personal and university levels;
- disseminate ideas and solutions to a wider audience;
- create a network of scientists and trainees with common interests;
- provide an environment for educational interaction;
- improve efficiency in the educational process between teachers and students;

- provide a field for sharing and discussing problems;
- help share accumulated experience and good practices;
- facilitate dialogue among colleagues;
- foster interdisciplinary research in institutions;
- provide a public forum for expression;
- provide a question–answer system;
- enable cooperation on common projects;
- provide resources for open-access reading, downloading, sharing of educational resources, etc.

Last but not least, the creation of a real-time web-based discussion platform for the field of higher education aims at supporting young scholars in achieving their academic goals. This could improve their academic outcomes and scientific and technical awareness, deepen their cultural sensitivities and help them accumulate and share new knowledge while also finding answers to their questions.

2. Related Work

Reading and posting ideas, views, and comments on discussion platforms is becoming more and more common. The idea of creating such an online forum was born from the need for a flexible and convenient tool that provides teachers and students with the opportunity to manage the learning process, share knowledge, and discuss.

The flexibility to reply to messages and post at a convenient and preferred time, as well as to follow conversations at any time, is what makes discussion forums so popular [1]. Xiaoling [2] argued that the discussion forum plays an important role in higher education. In addition to eliminating the communication barriers that exist between instructors and students, it also provides an opportunity for fruitful discussions between students and their peers. The connections created in this way could also engage learners outside the university environment, encouraging them to interact, give, and receive feedback.

The discussion platform can also motivate students to freely and confidently express their thoughts and ideas. Online discussion forums produce good discussions that learners continue to confirm and share in a real environment as well [3]. In their study [4], the scientists claimed that learners who participated in a discussion forum had better results than those who did not participate in the forum. Therefore, the online discussion forum has a positive effect on the learning process as well. According to [1], experience and satisfaction with learning obtained from online forums influence both active learning and independent learning. Scientists also claim that online forums are an effective training ground for deeper learning, cultivating original thought, and applying data acquired in a course [1].

According to another study [5], the results of using an online forum in learning are increased grades and satisfaction on the part of learners. In [6], scientists consider the discussion forum as a place that consists of: - user groups in which users are grouped according to their access to the forum; - publications, which are the messages in the individual categories; - categories, these are the topics on which the user can post an opinion; - admin, a user who manages the forum's appearance, database, and rules.

The potential of this type of online discussion platform is great and if used properly it could achieve the set goals. This publication marks the beginning of an introduction to online discussion platforms and hopefully contributes to the development of interactivity in the educational process.

3. Tools for Creating a Web Platform

This section discusses the selection of the development environment, programming language, local storage server, and database management system, as well as tools (plugins, libraries, and extensions) for building the web-based application.

3.1. Programming Languages and Technologies

HTML (Hyper Text Markup Language) is used for creating web pages. It is the standard markup language on the Internet, and its rules are defined by the World Wide Web Consortium (W3C). Client browsers use HTML to figure out exactly how to present the received data to the user. The language also takes care of the appearance of the elements found on the web pages, although this role by and large is undertaken by CSS (Cascading Style Sheets), with HTML being used mainly to structure pages [7].

CSS is a separate language containing many tools for adding style (e.g., fonts, colors, and spacing) to enhance the appearance of HTML pages [8]. It can be seen as an upgrade to HTML. CSS provides significant convenience in building and controlling the appearance of HTML documents by requiring changes to only a single file, without the need to change the HTML code on each page. CSS is most often used in addition to plain HTML but is also applied to XML (Extensible Markup Language) web pages and documents. The CSS specification is officially maintained by the W3C. Before the CSS standards were established by the W3C, the content of the sites and the style of their design were written on the same HTML page [8].

JavaScript is an interpreted programming language distributed with most web browsers. It was created at Netscape in 1995. JavaScript is a dynamically scriptable, prototype-based, object-oriented language with premium features.

It is most often applied to add functionality and load data. It can also be used to write JSON (JavaScript Object Notation) server scripts, as well as many other applications. JavaScript supports event-driven, functional and imperative (including object-oriented and prototype-based) styles. Moreover, it provides the ability to validate client-side forms so that server-side data processing is faster [9].

Django is a framework for creating interactive web-based applications in the Python programming language. The system is free software developed by volunteers under the auspices of the Django Software Foundation and is based on a Model-Views-Template pattern. Django primarily facilitates the creation of complex web systems, almost always involving databases. Further, it offers a flexible ORM for modelling and a complete administrative interface for managing, creating, reading, updating, and deleting content. Django's main goal is to make it easy to create complex, database-driven websites.

Python is a high-level, interactive, object-oriented programming language created by Guido van Rossum in the early 1990s. Python's language constructs and its object-oriented approach aim to make code clear and logical, for both small and large projects [9].

In the discussion platform, Python is used as the language to code the logic models, classes and all the functionality that the website performs.

Visual Studio Code is a program code editor for Windows, Linux and OS X. It supports a rich set of development tools such as debugger, built-in Git control and IntelliSense, which allows simultaneous work on two files open side by side. The editor is a Microsoft product and is free, publicly available for review, and can be used for various programming languages such as C, C#, C++, Java, JavaScript, Node.js, Python, Fortran and Go. Visual Studio Code is based on Electron, which is based on Chromium, which is used to deploy io.js applications for desktops. Visual Studio Code can be supplemented with multiple extensions.

3.2. Django Libraries

Django-tinymce is a Django application that contains an executable tool for rendering form fields in the TinyMCE editor. The TinyMCE editor is integrated with django-tinymce to allow easy installation and use [10]. The library features are as follows:

- Usable as an executable form module or as an interface;
- Improved support for content languages;
- Integration with TinyMCE for spell checking;
- Facilitation of predefined link lists and images for dialogue windows;

- Support for Django-staticfiles;
- Integration with django-filebrowser;
- Compression of TinyMCE JavaScript code [10].

Django-resized resizes images to a specified size. The following options can be set: size – offers maximum width and height, for example, (640, 480); none – keeps the original size of the image; scale – scales the image; crop – resizes and cuts; quality – improves the quality of resized image 0..100; keep_meta – keeps EXIF and other metadata; default True, force-format – force format the resized image [11].

Django-hitcount is a counter that allows tracking the number of hits (views) for a certain object. Tracking a visit or view of a web page is a difficult task; in general, doing so relies upon tools such as Google Analytics. django-hitcount can track hits with HitCountDetailView and HitCountJSONView.

There are different methods of displaying hits: template tags, views, and models. Moreover, there are additional settings that can be used to customize Django-hitcount. These facilitate, for example, the number of days, weeks, months, hours, etc.; limiting the number of active visits from one IP address; and excluding visits from users from specific user groups [12].

Django-taggit is a reusable Django application designed to easily add effective tags (the Django way of tagging). It can add or remove tags to an object, return a list of objects tagged in a certain way, filter by specific or multiple tags, remove duplicate results, etc. [13].

3.3. Microsoft Visual Studio Code Extensions

Pylance is a fast, feature-rich extension that works together with Python in Visual Studio Code to provide efficient language support. Pylance is the official successor to Pyright, Microsoft's static type-checking tool. Pylance intelligently completes keywords, inserts expressions, quickly finds or renames all symbol references in code, reports code errors and warnings (diagnostics), supports code navigation and is compatible with other products [14].

Python is a Visual Studio Code extension with rich support for the Python language (for all actively supported versions of the language after 3.7), including features such as IntelliSense for auto-completion, formatting and code navigation (Pylance); linting; debugging; refactoring; variable explorer; and test explorer [14].

Django adds improved syntax with more default operators and keywords. It colors the syntax in the HTML code of the page.

Live Server is used to run a local development server with a live reload feature for static and dynamic pages.

4. Designing the Discussion Web Platform

The web discussion forum is designed to contain the following modules:

- Registration and login form
- Topic overview form and overview by category
- New topic creation form
- Form to create a comment
- Admin panel for overview and administration

HTML, CSS, JavaScript, Django framework, and Django libraries were used for their development, which contributed to the vision, functionality and intuitiveness of the project.

Logically, the application consists of interconnected pages, united in a common design.

The project consists of a registration and login page, a page for viewing topics, with the ability to search for a topic by category, keyword and/or tag, a page for creating and editing comments to topics, and a page for administering user data, as well as for viewing and creating various reports.

4.1. Design GUI of a Discussion Web Platform

In the academic discussion forum, anyone with an institutional (university) email can register, share opinions and seek answers and solutions to their questions. Additionally, users can create new topics, comment on topics created by other users, search by keyword, etc.

In this research, HTML, CSS, and JavaScript technologies were used to create the platform's GUI. The finished design of the online discussion forum is presented in Figure 1.

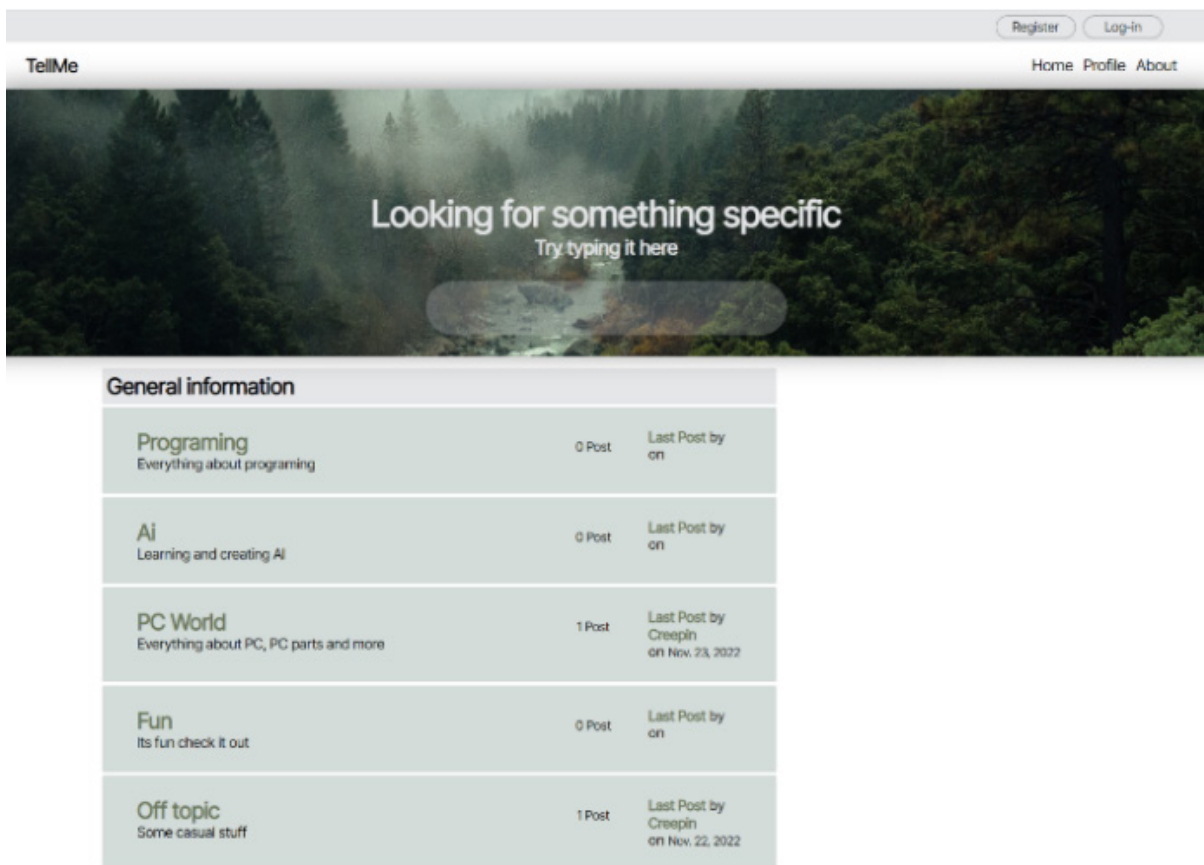


Figure 1. Overview of the platform GUI

4.1.1. Base HTML Page

The basis of the site is standard HTML code. Links to external CSS and JavaScript files have been added in the <head> section. In the main content <body> there are two sections to create a header

<header class="first_reg_log"> and navigation hyperlinks <nav class="navbar">. Between them is a wrapper, which serves as a gap between the sections. (Fig. 2 and Fig. 3) The base HTML page thus obtained is used as the basis for all other pages in the application.

```

<!--Register and Loggin bar & buttons-->
<section class="nav-reg-log">
  <header class="first-reg-log">
    <div class="reg-log-bar">

      {% if user.is_authenticated %}
      <div class="inform-greetings">Wellcome, {{user}}</div>
      <div class="btn-logout fade-in">
      <a href="{% url 'logout' %}"><button>Logout</button></a>
      </div>

      {% else %}
      <!--Buttons-->
      <div class="btn-register fade-in">
        <a href="{% url 'signup' %}"><button>Register</button></a>
      </div>
      <div class="btn-login fade-in">
        <a href="{% url 'signin' %}"><button>Log-in</button></a>
      </div>
      <!--END BUTTONS-->
    </div>
  </header>{% endif %}
</section>
<!--END OF REGISTER AND LOGGIN SECTION-->

```

Figure 2. Header section HTML code fragment

```

<section class="nav-links"> <!--Nav Links-->
  <nav class="navbar">
    <div class="container">
      <div class="navbar_main">
        <a href="{% url 'home' %}"><h1>TellMe</h1></a>
      </div>
    </div>
    <div class="navbar-sub-items">
      <ul>
        <li><a href="{% url 'home' %}">Home</a></li>
        {% if user.is_authenticated %}
        <li><a href="{% url 'update_profile' %}">Profile</a></li>

        {% else %}
        <li><a href="{% url 'signin' %}">Profile</a></li>

        {% endif %}

        {% if user.is_authenticated %}
        <li><a href="{% url 'create_post' %}">New Post</a></li>
        {% endif %}

        <li><a href="">About</a></li>
      </ul>
    </div>
  </nav>
</section> <!--END Nav bar-->

```

Figure 3. Navigation hyperlinks HTML code fragment

The base HTML page is formatted with site-wide parameters. The buttons, navigation hyperlinks and animation are obtained using the CSS code presented in Figure 4.

```

    .btn-register{
        height:10px;
        width:100px;
        margin: -11px;
        position:absolute;
        top:50%;
        left:84%;
        background-color: Transparent;
    }
    .btn-register button {
        background: transparent;
        height: 23px;
        line-height: 10px;
        border: 1px solid black;
        border-radius: 20px ;
        border-color: #8a8e90;
        color: #293031;
        display: inline-block;
        float: none;
        text-align: center;
        width: 80px;
    }
    .btn-login{
        height:10px;
        width:100px;
        margin: -11px 60px;
        position:absolute;
        top:50%;
        left:85%;
        background-color: Transparent;
    }
    .btn-login button{
        background: transparent;
        height: 23px;
        line-height: 10px;
        border: 1px solid black;
        border-radius: 20px ;
        border-color: #8a8e90;
        color: #293031;
        display: inline-block;
    }
    .nav-links{
        margin-left: 2%;
        margin-right: 2%;
    }
    .navbar {
        display: flex;
        justify-content: space-between;
        align-items: center;
    }
    .navbar h2 {
        font-size: 220%;
    }
    .navbar-sub-items ul {
        display: flex;
    }
    .navbar-sub-items ul li {
        margin-right: 10px;
    }
    .navbar-sub-items ul li a {
        display: inline-block;
        position: relative;
        color: black;
    }
    .navbar-sub-items ul li a:after {
        content: '';
        position: absolute;
        width: 100%;
        height: 2px;
        bottom: -3px;
        left: 0;
        background-color: #E5E6E8;
        transform-origin: bottom right;
        transition: transform 0.30s ease-out;
    }
    .navbar-sub-items ul li a:hover:after {
        transform: scaleX(1);
        transform-origin: bottom left;
    }

```

Figure 4. CSS code for buttons, hyperlinks, animation

The HTML page is illustrated in Fig. 5.



Figure 5. Base HTML preview page

4.1.2. Main HTML Page

Once the base HTML page is created, it can be used as a template for other pages, starting with the main page. The search box and main page background have been added in a new section. (Fig. 6).

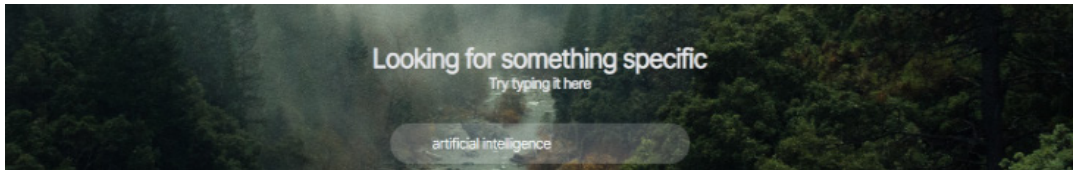


Figure 6. Search box on the main page

Fig. 7 shows a fragment of the main content of the base page.

```
<section class="content">
  <!--Subforum-->
  <div class="subforum"> <!--Main Content-->

    <div class="subforum-title"> <!--title of the forum-->
      <h1>General information</h1>
    </div> <!--End forum title-->

    {% for forum in forums %}
    <div class="subforum-row subforum-column" > <!--Inside content-->
      <div class="subforum-icon subforum-column">
        <i class="blank"></i>
      </div>
      <div class="subforum-description subforum-column">
        <h1><a href="{{forum.get_url}}">{{forum.title}}</a></h1>
        <p>{{forum.description}}</p>
      </div>

      <div style="text-align: center; margin-top:15px;" class="subforum-stats subforum-column">
        <span>{{forum.num_posts}} Post<br></span>
      </div>
      <div class="subforum-info subforum-column">
        <b><a href="{{forum.last_post.get_url}}">Last Post</a></b> by
        <a href="">{{forum.last_post.user.fullname|title}}</a><br> on
        <small>{{forum.last_post.date|date}}</small>
      </div>
    </div> <!--END Inside Content-->
    {% endfor %}
  </div> <!--END subforum Title and card-->
</section><!--END Sub-forum-->
```

Figure 7. Fragment of the main content of the base page

The keyword search box and platform categories are formatted using the CSS snippets, as shown in Fig. 8.

```
.text {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  background-color: transparent; }

.text h1 {
  font-size: 36px;
  color: #E5E6E8;
  background-color: transparent; }

.text h3 {
  font-size: 20px;
  color: #E5E6E8;
  background-color: transparent; }

.box {
  position: absolute;
  top: 80%;
  left: 50%;
  width: 20%;
  display: flex;
  transform: translate(-50%, -50%);
  background: rgba(255, 255, 255, 0.2);
  align-items: center;
  border-radius: 60px;
  padding: 15px 60px;
  backdrop-filter: blur(6px) saturation 180%; }

.box input {
  background: transparent;
  flex: 1;
  border: 0;
  outline: none;
  padding: 6px -20px;
  font-size: 20px;
  color: white; }

.content{
  margin-left: 4%;
  margin-right: 4%; }

.subforum {
  box-sizing: border-box;
  width: 60%; }

.content a {
  color: rgb(109, 121, 92);
  font-weight: bolder;
  background-color: transparent; }

.content p {
  background-color: transparent; }

.content h1{
  font-size: 25px;
  font-weight: bolder;
  background-color: transparent; }

.content span{
  font-size: 90%; }

.subforum-title{
  background-color: #E5E6E8;
  padding: 5px;
  border-radius: 2px;
  margin: 4px; }

.subforum-row {
  display: grid;
  grid-template-columns: 2% 65% 13% 20%; }

.subforum-column{
  font-family: 'Nanum Gothic', sans-serif;
  padding: 10px;
  margin: 4px;
  border-radius: 2px;
  background-color: rgb(211, 220, 217); }
```

Figure 8. Search box CSS code

The main page (main.html) inherits the base page (base.html) and adds the previously created content specific to it. The beginning of the main page, as with any other page, must state clearly whether it inherits another page. Moreover, if any files are being read locally, it must be mentioned where they are taken from.

```
{% extends 'base.html' %}
{% load static %}
```

Block content is created for the rest of the code. The principle is similar for all pages.

4.2. Install and Get Ready to Work with Django

```
INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "main",
    "rest_framework",
    "corsheaders",
    "django_unicorn",
    'tinymce',
    'hitcount',
    "taggit",
    'register',
]

from django.db import models
from django.utils.text import slugify
from django.contrib.auth import get_user_model
from django_resized import ResizedImageField
from tinymce.models import HTMLField
from hitcount.models import HitCountMixin, HitCount
from django.contrib.contenttypes.fields import GenericRelation
from taggit.managers import TaggableManager
from django.shortcuts import reverse

User = get_user_model()
```

Figure 9. Installed apps and libraries

The libraries from Fig. 9 are imported into the web-based platform project in MAIN>>main>>models.py.

4.3. Database Structure

The database on which the reasoning and tests in the research are focused is built from the following tables (objects): Users, Author, Category, Comment, Post, Reply, and Tags.

Django and Python extensions have been added to the Visual Studio Code development environment. A ready-made automated script <https://github.com/SelmiAbderrahim/AutoDjango.git> from Github has been downloaded and run, which prepares an easy-to-use programming environment. A virtual environment has been created in which Python AutoDjango.py-venv is started.

Django libraries are required to be installed. For this purpose, the libraries shown in Fig. 9 have been added in the project folder MAIN>>project>>settings.py, INSTALLED_APPS.

Further, the relationships between them are based on the phenomena and processes taking place in the online discussion platform.

Fig. 10 presents the information model of the Users table, which is filled in when a user registers on the platform. It consists of the following fields: username, password, and email address. After initial registration, users can enter additional information about themselves, such as first_name, last_name, fullname, tag (slug), status (user role) and profile picture.

USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
Koleva_Denica	koleva_denica@abv.bg	Denica	Koleva	
admin				
asd	supp3man@gmail.com	Stoyko	Mihaylov	
mimijekowa	mimijekowa@abv.bg	Мария	Жекова	
test123	teodosi.kehayov@gmail.com	Teodosi	Kehayov	
tonisosa	stu34983@uft-plovdiv.bg	Anton	Savov	

6 users

Figure 10. Table Users

The Category information model is populated when creating a forum topic category.

The table comprises the following fields (attributes): category name (title), tag (slug) and description (Fig.11).

CATEGORY	TAG
Career	career
Development	development
Off topic	off-topic
Fun	fun
PC World	pc-world
AI	ai
Programming Languages	programming-languages

Figure 11. Table Categories

The Comment table is filled when writing a comment (post) in the discussion forum. It has the following attributes (fields): user – the username of the user who created the comment, content – the

content of the comment, date – date of creation of the comment (post), replies – replies to the comment, if any (Fig.12).

COMMENT	TAG
This is very good.	this-is-very-good
If you know something interesting, just tell me, please.	if-you-know-something-interesting-just-tell-me-please
yes	yes
is good	is-good

Figure 12. Table Comments

The Tags table contains information about keywords. It stores the post and category tags

(Fig.13). The data stored in the database are keyword name, tag (slug), and content type.

NAME	SLUG
academic	academic
ai	ai
article	article
degree	degree
discipline	discipline
extraction	extraction
games	games
programming	programming
python	python
recognition	recognition
research	research
syntheze	syntheze
translation	translation

Figure 13. Table Tags

The application provides for a user with the superuser role to access the admin panel to approve new posts.

5. Development of the Discuss Web Platform

The real-time academic discussion platform was created in the Visual Studio Code development environment, using Python programming language. Django, appropriate libraries, and extensions were used to create the GUI.

Python offers a wide range of libraries and is easy to maintain, read, and edit. It is preferred for scripting applications, web development, word processing and scientific computing, and is becoming increasingly popular in education [15].

```
class Author(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    fullname = models.CharField(max_length=40, blank=True)
    slug = slug = models.SlugField(max_length=400, unique=True, blank=True)
    bio = HTMLField()
    points = models.IntegerField(default=0)
    profile_pic = ResizedImageField(size=[50, 80], quality=100, upload_to="authors",
                                    default=None, null=True, blank=True)

    def __str__(self):
        return self.fullname

    def save(self, *args, **kwargs):
        if not self.slug:
            self.slug = slugify(self.fullname)
        super(Author, self).save(*args, **kwargs)
```

Figure 14. Fragment of Author class

The class that describes the categories in the discussion platform is Category (Fig.15). The following fields characterize a category:

- title – the name of the category, Text field;
- slug – short tag (label) of the category;
- description – description, Text field.

This class is unique in that each created category returns a URL via the `get_url` method. Only a user with the admin role can add and remove categories. The last two functions in the figure – `num_posts` and `last_posts` – return the number of posts in a given category and the date of the last activity in that category, respectively.

The Post class contains fields that characterize the posts (comments) (Fig. 16) in the forum:

- title – the name of the post, Text field;

5.1. Logic Models and Classes in the Application

The class that is the model for creating, processing and interacting with users in the web-based discussion platform is Author (Fig.14). The fields to be filled in when creating a user are as follows:

- user – username, Foreign key;
- fullname – full name of a user, maximum length up to 40 characters, optional field, Text field;
- slug – a short tag (label) of a user;
- bio – we use the TinyMCE library, which was installed with the HTMLField function, which gives a nicer text field;
- points – a counter that shows how many times a user's post has been seen by other users, Integer field;
- profile_pic – profile picture, which automatically changes to a size of 50x50 pixels, Image field.

- slug – short tag (label) of the comment;
- user – name of the user who created the post, Foreign key;
- content – the content of the post, Text field;
- categories – post category, ManyToMany field;
- date – date of creation of the post, DateTime field;
- approved – status of the post (whether it has been approved by an admin), Boolean field;
- hit_count_generic – number of times the post was opened;
- tags – public tag;
- comments – ManyToMany field;
- closed – status of the post (whether the post/topic is closed/on), Boolean field.

```

class Category(models.Model):
    title = models.CharField(max_length=50)
    slug = models.SlugField(max_length=400, unique=True, blank=True)
    description = models.TextField(default="description")

    class Meta:
        verbose_name_plural = "categories"
    def __str__(self):
        return self.title

    def save(self, *args, **kwargs):
        if not self.slug:
            self.slug = slugify(self.title)
        super(Category, self).save(*args, **kwargs)

    def get_url(self):
        return reverse("posts", kwargs={
            "slug":self.slug
        })
    @property
    def num_posts(self):
        return Post.objects.filter(categories=self).count()
    @property
    def last_post(self):
        return Post.objects.filter(categories=self).latest("date")

```

Figure 15. Fragment of Category class

```

class Post(models.Model):
    title = models.CharField(max_length=400)
    slug = models.SlugField(max_length=400, unique=True, blank=True)
    user = models.ForeignKey(Author, on_delete=models.CASCADE)
    content = HTMLField()
    categories = models.ManyToManyField(Category)
    date = models.DateTimeField(auto_now_add=True)
    approved = models.BooleanField(default=False)
    hit_count_generic = GenericRelation(HitCount, object_id_field='object_pk',
        related_query_name='hit_count_generic_relation' )
    tags = TaggableManager()
    comments = models.ManyToManyField(Comment, blank=True)
    closed = models.BooleanField(default=False)
    state = models.CharField(max_length=40, default="zero")
    def save(self, *args, **kwargs):
        if not self.slug:
            self.slug = slugify(self.title)
        super(Post, self).save(*args, **kwargs)
    def __str__(self):
        return self.title
    def get_url(self):
        return reverse("detail", kwargs={
            "slug":self.slug
        })
    @property
    def num_comments(self):
        return self.comments.count()
    @property
    def last_reply(self):
        return self.comments.latest("date")

```

Figure 16. Fragment of Post class

The class that describes the comments in the discussion platform is Comment. (Fig. 17). It contains information about

- user – username of the person who creates the comment, Foreign key;

- content – the content of the comment, Text field;
- date – date of creation of a comment, DateTime field;
- replies – replies to comment, ManyToMany field.

```
class Comment(models.Model):
    user = models.ForeignKey(Author, on_delete=models.CASCADE)
    content = models.TextField()
    date = models.DateTimeField(auto_now_add=True)
    replies = models.ManyToManyField(Reply, blank=True)

    def __str__(self):
        return self.content[:100]
```

Figure 17. Fragment of Comment class

The Reply class stores information about

- user – user who wrote the reply, Foreign key;
- content – content of the reply, Text field;
- date – date of the reply, DateTime field as presented in (Fig. 18).

```
class Reply(models.Model):
    user = models.ForeignKey(Author, on_delete=models.CASCADE)
    content = models.TextField()
    date = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.content[:100]

class Meta:
    verbose_name_plural = "replies"
```

Figure 18. Fragment of Reply class

Each of the classes contains the special `__str__` method, which is used to generate a textual representation of the objects: comment/post, reply to comment, category name, post, and user.

5.2. Registration Form

The file forms.py in the application folder contains the following:

- model for registration form – **UserCreationForm**;
- authentication form model – **AuthenticationForm**;
- model for changing a user's account, for example, name, photo and password – **UpdateForm**.

Figure 19. Register and Login form

In the `SignUp` and `SignIn` classes, there are standard functions for initializing object values `__init__`. (Fig. 19), These are class constructors that Python automatically calls whenever it creates new

objects of the corresponding classes. Methods that initialize objects expect an input parameter `self`, which refers to the object itself. (Fig. 20).

```

class SignUpForm(UserCreationForm):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.fields['username'].widget.attrs.update({
            'type': 'text',
            'class': 'input-field',
            'placeholder': "Username",
            'required': ''
        })
        self.fields['email'].widget.attrs.update({
            'type': 'email',
            'class': 'input-field',
            'placeholder': "E-mail",
            'required': ''
        })
        self.fields['password1'].widget.attrs.update({
            'type': 'password',
            'class': 'input-field',
            'placeholder': "Password",
            'required': ''
        })
        self.fields['password2'].widget.attrs.update({
            'type': 'password',
            'class': 'input-field',
            'placeholder': "Repeat Password",
            'required': ''
        })

class Meta:
    model = User
    fields = ('username', 'email', 'password1', 'password2')
    
```

Figure 20. Fragment of UserCreation form

6. Experiments

The paper presents the development process of a web platform designed for sharing experiences and ideas, discussions, and finding answers to academic questions, where users can create topics in various admin-approved categories, comment on topics created by other users, and customize their profile with a biography, photo, and free text description.

Multiple users with different access roles were registered through the platform's registration form. Several categories (forum topics) were created – artificial intelligence (AI), programming languages, development, PC world, career, fun, and off-topic.

The comments and their responses were created by different users in more than one category. It became clear that the counter correctly counts the views of posts (comments), the date of the last post and the last seen post by a user. (Fig. 21).

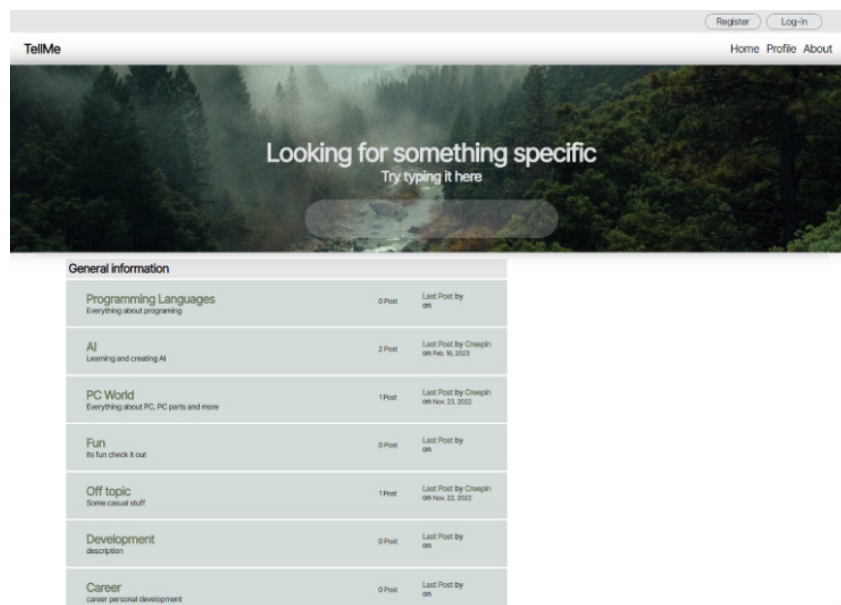


Figure 21. General Appearance

Fig. 22. shows how an Off-Topic comment is displayed, how many views and replies that comment

has and the date and user who last replied to the comment.



Figure 22. Create a post in a category

Fig. 23. presents a comment and the discussion that took place below it.

the topic and their profile pictures, roles, and accumulated points.

You can see the different users who commented on



Figure 23. Comment with replies to it

7. Conclusion

The web-based discussion platform, created to exchange experiences and ideas as well as to discuss and find answers to academic questions, bring together technologies such as HTML, CSS, JavaScript, Django, and Python. Using the platform, users can freely create topics in various admin-approved categories, comment on topics created by other users and customize their profile with a bio, photo, and free text description. The platform is designed with an easy-to-navigate and simple interface.

Disadvantages can be that there are no classes for defining and catching exceptions and classes for various platform behaviors – sending emails, encoding passwords, etc. We have planned to store passwords as plain text in the database, as user data becomes vulnerable in the event of a potential breach.

In the future, the discussion platform has the possibility of being extended and improved by adding functionalities for rating topics/projects, providing user feedback and sharing public and private posts and improving the front-end part using AngularJS.

The online forum, created for use in academic circles, promotes development, coordination, and dialogue among users of educational services – primarily teachers and trainees (doctoral and other students). The created real-time discussion platform could stimulate academic impact and help the evolution of higher education by providing a forum for sharing and discussing research problems, creating a network of scholars and learners with common interests and disseminating interesting ideas and solutions to a wider academic audience.

Acknowledgements

This paper is financed by the European Union-NextGenerationEU, through the National Recovery and Resilience Plan of the Republic of Bulgaria, project № BG-RRP-2.004-0001-C01. The paper reflects only the author's view and the Agency is not responsible for any use that may be made of the information it contains.

References:

- [1]. Sivanandan, P., Rajandram, K. V. & Chan H. (2014). Online forum: A platform that affects students' learning. *American International Journal of Social Science*, 3(7), 107-116.
- [2]. Xiaoling, L. (2018). The Effectiveness of Online Discussion Forums and Recommendations for Chinese Higher Education. *A Project Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Education*, Department of Curriculum and Instruction, University of Victoria.
- [3]. Jeff, K. (2016). *5 Online Discussion Tools Fuel Student Engagement, Common Sense Education*, Retrieved from: <https://thejournal.com/articles/2016/04/05/5-online-discussion-tools-to-fuel-student-engagement.aspx> [accessed: 02 October 2023].
- [4]. Cheng, C. K., Paré, D. E., Collimore, L. M., & Joordens, S. (2011). Assessing the effectiveness of a voluntary online discussion forum on improving students' course performance. *Computers & Education*, 56(1), 253-261.
- [5]. AlJeraisy, M. N., Mohammad, H., Fayyoumi, A., & Alrashideh, W. (2015). Web 2.0 in education: The impact of discussion board on student performance and satisfaction. *Turkish Online Journal of Educational Technology-TOJET*, 14(2), 247-258.
- [6]. Biriya, A. H., & Thomas, E. V. (2014). Online discussion forum: A tool for effective student-teacher interaction. *International Journal of Applied Science*, 1(3), 111-116.
- [7]. Rouse, M. (2016) *Hypertext Markup Language*. Techopedia. Retrieved from: <https://www.techopedia.com/definition/1892/hypertext-markup-language-html> [accessed: 03 October 2023].
- [8]. Nixon, R. (2014). *Learning PHP, MySQL & JavaScript: With JQuery, CSS & HTML5*. O'Reilly Media.
- [9]. Shyam, A., & Mukesh, N. (2020). A Django based educational resource sharing website: Shreic. *Journal of scientific research*, 64(1), 138-152.
- [10]. Django-tinymce. (n.d.). *Welcome to the django-tinymce documentation — django-tinymce 2.3 documentation*. Django-tinymce. Retrieved from: <https://django-tinymce.readthedocs.io> [accessed: 04 October 2023].
- [11]. Pypi. (n.d.). *Django-resized*. Pypi. Retrieved from: <https://pypi.org/project/django-resized> [accessed: 05 October 2023].
- [12]. django-hitcount. (n.d.). *Django-hitcount, Installation and Usage, Damon Timm*. django-hitcount. Retrieved from: <https://django-hitcount.readthedocs.io/en/latest> [accessed: 07 October 2023].
- [13]. Alex Gaynor. (n.d.). *Django-taggit's Documentation*. Alex Gaynor Copyright. Retrieved from: <https://djangotaggit.readthedocs.io/en/latest> [accessed: 07 October 2023].
- [14]. Steenman, D., (2023). *10 Must Have VS Code extensions for Python developers*. Towards the cloud. Retrieved from: <https://towardsthecloud.com/best-vscode-extensions-python> [accessed: 08 October 2023].
- [15]. Donaldson, T. (2013). *Python: Visual QuickStart Guide 3rd Edition*. Peachpit Press.