МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет електроніки та інформаційних технологій

Кафедра комп’ютерних наук

Кваліфікаційна робота магістра

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПРОЕКТУВАННЯ ВЕБ-ПЛАТФОРМИ ДЛЯ ПРОВЕДЕННЯ ОЛІМПІАД ПІД КЕРІВНИЦТВОМ МІНІСТЕРСТВА ОСВІТИ І НАУКИ УКРАЇНИ

Здобувач освіти гр. ІН.м-01н Є.А. Шимко

Науковий керівник,
кандидат ф.-м. наук, доцент О.Б. Проценко

Завідувач кафедри
доктор технічних наук, професор. А.С. Довбиш

Суми 2022

Сумський державний університет
(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*
Спеціальність *122 «Комп'ютерні науки»*

Затверджую:
зав.кафедрою _____
_____
" _____" _____ 20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Шимко Євгену Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Інформаційна технологія проектування веб-платформи для проведення олімпіад під керівництвом Міністерства освіти і науки України*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____

3. Вхідні данні до проекту (роботи) _____
_____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Інформаційний огляд 2) Вибір програмних засобів 3) Практична реалізація*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
_____
_____
_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	*Інформаційний огляд*		
2.	*Вибір програмних засобів*		
3.	*Практична реалізація*		
4.	*Оформлення кваліфікаційної магістерської роботи*		

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 71 стор., 28 рис., 2 табл., 1 додаток, 16 джерел.

Об’єкт дослідження — процес створення наукової веб-платформи за допомогою Django.

Мета роботи — розробка та проектування веб-платформи для проведення олімпіад під керівництвом Міністерства освіти і науки України

Методи дослідження — методи проектування веб-патформ.

Результати — розроблено наукову платформу для проведення олімпіад під керівництвом Міністерства освіти та науки України. В роботі використовується фреймворк Django для швидкої розробки із нуля, із можливостями масштабування проекту. Розроблений веб-сайт реалізовано мовою програмування Python.

ВЕБ-ПЛАТФОРМА ДЛЯ ПРОВЕДЕННЯ ОЛІМПІАД ІЗ ТОЧНИХ НАУК, PYTHON, DJANGO

ЗМІСТ

ВСТУП

Проблеми потребують рішень. Це не лозунг чи цитата, а аксіома. Так само фактом є те, що без інтернету сучасний світ неможливий. Світова мережа знищує кордони та відстань, поступово витісняє і замінює все більше сервісів та побутових дрібниць.

Інтернет спрощує життя, дозволяє економити час та виконувати кілька справ в одночасно. Це безмежне джерело інформації та можливостей.

Кожна людина так чи інакше зацікавлена в глобальній мережі. Хтось для розваг, інші ж будують в ньому бізнес, кар'єру і навіть життя.

Чим молодше покоління, тим раніше воно приєднується до культури інформаційного простору. Дані які отримують діти напряму впливають на формування їх особистості і світогляду. Тож страшне покоління не може бути не зацікавленим у контенті який споживають діти.

В просторі перенасичення інформацією дуже складно привернути погляд до себе та втримати увагу дітей. Треба брати до розрахунку сотні факторів, від віку і психологічних особливостей покоління до елементарних трендів. Скласти успішний проект який був би ефективним пасивно - не можливо, освіта не може відставати від епохи і має йти за нею крок в крок. Справи складаються так, що навіть з існуючим профіцитом варіантів всіх можливих сайтів, неможливо знайти два однакових, навіть якщо вони були створені по одному шаблону. Сайти створюються задля різноманітних цілей: інформаційні сторінки, реклама, сервіси, магазини, - кожен з них потребує особливого бачення та унікального підходу до створення. В залежності від цілей та амбіцій сайтів, вони потребують різних ресурсів, дизайну та витрат. Наразі, найпопулярнішими типами сайтів є блоги, корпоративні сайти, брошури, портали, соціальні мережі, форуми, портфоліо. Яким би не був сайт, він має бути виконаним якісно. Важливість та потрібність електронної комерції не піддається сумнівам і лише зростає кожного дня і кожної години,

і навіть перманентний зріст потоку технологій не може задовільнити постійно прогресуючого попиту.

Раніше вже порівнювалися веб сайт та візитівка, але буде злочином замовчати про набагато більшу інформаційність сторінок над мінімалістичним картками. Якщо мова йде про навчальний ресурс то коректно буде порівняти його з книгою або посібником. Сайти такого типу дають користувачам змогу ознайомитися з завданнями та записати їх рішення в спеціальні поля. такі сайти дуже зручні для освітніх установ, адже можуть поєднувати найрізноманітніший функціонал, від елементарного моніторингу часу до повноцінного відстежування екрану учня задля запобігання шахрайства при виконанні завдань та задач. Реалізація найрізноманітніших функцій обмежена лише потребами та фантазією замовника.

Важко сперечатися з тим що 21 століття - епоха шаленого ритму життя, коли час проминає ніби на прискореному перемотуванні далі, вперед. Аби пристосуватися до подібного темпу треба відповідати йому та мати можливість грамотно розподілити свій розклад, аби використати кожну хвилину максимально ефективно. Вчитися цьому потрібно з дитинства, привчаючи до дисципліни та грамотному тайм менеджменту, який стане в пригоді в дорослому житті. Навчання це стосується безпосередньо в першу чергу. Оскільки освіта - це основне заняття молодого покоління, дуже важливим є його корисність та результативність. Сформувавши процес вірно ми матимемо змогу не тільки швидко та зручно навчати та перевіряти знання, я й будувати в розумах зростаючого покоління ефективний стиль мислення. Крім того, маючи електронні платформи для навчання, учні, та студенти, зможуть отримувати знання в будь якому місці, де б вони не знаходилися. Правда світу така, що наразі не кожна дитина може відвідувати навчальний заклад, але це не повинно стати причиною відсутності в неї знань. Кожен має право на освіту. Навіть більше, зараз, як ніколи до цього ми бачимо важливість та необхідність освіченості населення. Діти мають право та

повинні дізнаватися свою історію, культуру, та розвиватися всебічно для свого кращого майбутнього і їхнє місцеположення не має ставати перепоною.

В реаліях наших днів багато дітей зіштовхнулися з необхідністю тимчасового виїзду за кордон, тисячі школярів мають проблеми з завершенням отримання освіти що ставить під сумнів не лише якість їх знань а й можливість в майбутньому отримати вищу освіту в Україні чи за кордоном. В цьому випадку, електронний ресурс для навчання та перевірки знань міг би стати панацеєю, здатною. врятувати учнів та студентів зі складного положення. Доступ до інтернету та електрики - ось і все що має бути потрібним сучасній людині для отримання знань. Задача ж освітян в тому щоб грамотно розпорядитися наявними ресурсами та побудувати навчальну програму згідно з можливістю епохи.

Саме тому ця робота присвячена гострій проблемі освіти, а саме, необхідності допомогти учням та студентам розвивати свої здібності критичного мислення, логіки та розуміння точних наук. Тести для дітей будь якого віку, задачі на розвиток логіки та інші вправи можуть бути представлені на освітньому ресурсі.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Аналіз предметної області

Еволюціонування у сфері веб-розробок відбувається дуже швидко у порівнянні з іншими науково-технічними галузями. Усього за кілька років примітивні статичні веб-сторінки, написані на "чистому" html, перетворилися на надскладні, багатофункціональні, інтегровані з іншими програмами веб-системи.

Останнім часом стало очевидно, що сайт - це не просто електронна візитка або онлайн-каталог, а зручний і дуже ефективний інструмент. Але, ефективним сайт стає лише у випадку якісної розробки та грамотного просування.

Оцінка якості сайту - не таке просте завдання, як може здатися на перший погляд. Як показує досвід, більшість людей взагалі не уявляють, що ще можна оцінювати крім дизайну. Хтось, звичайно, згадує про зручність структури сайту, але як саме визначити – чи для всіх вона зручна, вже не розуміють. Про решту важливих моментів згадують дуже рідко.

А тим часом, незважаючи на те, що дизайн та структура сайту дуже важливі, існують і інші важливі критерії оцінки сайту.

Якісний сайт повинен не тільки відповідати усім сучасним технічним вимогам та побажанням цільовий аудиторії, але й враховувати внутрішні та зовнішні фактори ранжування (критерії якості сайту з погляду пошукових систем).

Розглянемо основні критерії оцінки якості сучасного сайту.

Для того, щоб створити якісний, сучасний та ефективний сайт необхідно:

Розробити ефективний та привабливий дизайн, відповідний усім очікуванням цільовий аудиторії та витриманий в єдиному стилі. Якщо сайт

виглядає добре, то відвідувачі вважають його ефективним, корисним та цікавим.

Ретельно продумати "юзабіліті" (структура, зручність користування сайтом). Тільки одного красивого дизайну недостатньо. Потрібно, щоб кожна сторінка сайту максимально відповідала побажанням відвідувача сайту, щоб всі елементи сторінки були розташовані в найзручніших місцях, щоб інформація на сайті читалася легко, а навігація по сайту та пошук - були інтуїтивно зрозумілими та простими. Відвідувач за мінімальну кількість часу повинен отримувати те, що він хотів отримати від сайту. [10]

Використати сучасні методи розробки сайту. В даний час для розробки сайту краще всього використовувати HTML5, CSS3, PHP (або ASP, Java, Python, Pearl, Ruby тощо), MySQL (або OracleDatabase, Microsoft SQL Server тощо). [8]

Налаштувати автоматичну трансформацію (адаптивність) сайту під будь-які мобільні пристрої (смартфони, планшети та ін.). Адаптація сайту під різні екрани – вже давно перейшла до розряду обов'язкових вимог.

Починати розробку з аналізу конкурентів та складання плану пошукової оптимізації (SEO). В даний час існують сотні мільйонів сайтів, з якими усім знову створюваним сайтам доведеться конкурувати в пошуковій видачі. Для того, щоб простіше було просувати сайт у ТОП - бажано ще до початку розробки сайту визначитися за якими ключовим словам сайт буде просуватися. Сучасні алгоритми пошукових систем більше не працюють за формулами, розробленими для них людьми. Формула ранжування для кожного запиту та кожного сайту визначається самою пошуковою системою у процесі аналізу конкурентів по кожному ключовому слову. [9]

Створювати якісні "посадкові" сторінки для кожного пошукового запиту. Про те, що спрямовувати всіх відвідувачів на головну сторінку сайту - нерозумно і неефективно - широко відомо, але мало хто замислюється над тим,

що в ідеалі - для кожного контекстного оголошення, для кожної фрази, яка просувається в пошуковій видачі - на сайті повинна існувати так звана "посадкова" сторінка, або сторінка "приземлення". Ця сторінка повинна максимально швидко і всеосяжно відповідати на запит користувача, за яким він перейшов на сайт. Після відвідування цієї сторінки у відвідувача більше не повинно виникати спокуси повернутися до пошукової видачі та подивитись інші подібні сайти (сайти конкурентів). Саме тому потрібно чітко уявляти, що саме хоче побачити відвідувач по кожному ключовому запиту та створити під цей запит окрему "спеціальну" сторінку. [7]

Розміщувати на сайті виключно унікальний та якісний контент. Тільки сайт з цікавим, корисним та на 100% унікальним контентом має шанс не тільки потрапити в ТОП, але й утриматись у ньому надовго. Саме контент - є ключовим фактором успішного просування сайту.

Грамотно використовувати ключові слова. Ключові слова обов'язково повинні бути у заголовках сторінок (Title), у заголовках документа (h1-h6), в адресах документів (URL), при цьому необхідно використовувати ЧПУ (людинозрозумілі URL), ну і, звичайно, в самих текстах документів, і в текстах інших сторінок сайту. Крім цього необхідно використовувати внутрішню "перелінковку" з використанням ключових слів.

Забезпечити інтеграцію з соцмережами та можливість залишення відгуків. На сайті повинна бути реалізована можливість проставляти "лайки" та залишати відгуки про компанію чи проект. Відгуки, залишені на сторінках сайту, дозволяють підвищити унікальність однотипних сторінок, які пошукові системи часто приймають за неточні копії один одного і просто виключають їх з індексної бази. Наявність посилань на соцмережі, лайків та відгуків - дозволяє пошуковий системі прийняти рішення про соціальну активності компанії, і в нагороду за це - "підняти" сайт в пошуковій видачі. [13-15]

Забезпечити високу швидкість завантаження сторінок сайту, захист від спаму, злому та вірусів. Сучасні пошукові системи велику увагу приділяють

технічному стану сайту. Якщо сайт заражений вірусом, швидкість завантаження його сторінок невелика, або використовуються застарілі технології створення сайту - високих позицій у видачі пошуковий системи очікувати не варто. [2]

Після запуску сайту проводити глибоку аналітичну роботу і безперервно допрацьовувати сайт з урахуванням виявлених технічних, структурних, функціональних та інших недоліків, а також поведінкових особливостей використання сайту.

1.2 Актуальність розробки сайту

У цілому світі інформаційні технології дедалі більше входять у життя людини. Інформатизація стосується всіх сфер життєдіяльності. І одним з пріоритетних напрямів інформатизації є інформатизація освіти.

В даний час практично кожна людина має в наявності комп'ютер, ноутбук, мобільний телефон з виходом в Інтернет. Мета створення сайту полягає у передачі корисної інформації користувачеві, яка дозволить популяризувати точні науки серед учнів, абітурієнтів та студентів.

Актуальність створення сайту полягає в тому, щоб довести відомості певної тематики до необмеженого, різнорідного кола людей.

Мета даної роботи - створити Web-ресурс для підвищення інтересу до такого предмета, як математика.

Перспективним напрямом інформатизації організаційно-методичної та освітньої діяльності є розробка сучасних ресурсів для учнів, на яких будуть розміщуватися цікаві завдання та інформація з різних дисциплін.

Сайт, що розробляється, дозволить учням, абітурієнтам та студентам шукати цікаві завдання з математики, алгебри, геометрії та інших точних наук.

Проектований веб-ресурс виконується на замовлення Міністерства освіти та науки України. Можна виділити дві найважливіші функції, які він покликан реалізовувати. По-перше, сайт дозволить учням швидко та легко знаходити потрібну корисну інформацію та завдання з точних наук, використовуючи розумний пошук, фільтрацію та інші найсучасніші інструменти.

По-друге, сайт може бути використаний як інструмент навчального процесу, що значно полегшить завдання вчителям.

Таким чином, актуальність даної роботи обумовлена тим, що на сьогоднішній день веб-ресурс для учнів може стати ефективним інструментом для удосконалення освітнього процесу, організації дозвілля школярів, підвищення популярності математики та точних наук.

1.3 Аналіз аналогів

На даний момент у всесвітній мережі існує велика кількість веб-ресурсів, освітніх сайтів та порталів, і щодня їх кількість зростає. Завдяки ним користувач може знайти потрібну інформацію.

Перед розробкою веб-сайту було проведено огляд вже існуючих сайтів, а саме:

- zadachi.mccme.ru
- problems.ru

На сайті zadachi.mccme.ru представлений об'ємний список математичних завдань. Реалізовано наступні можливості:

- здійснювати тематичний та текстовий пошук;
- здійснювати перехід за номером завдання або сторінки;
- сортувати списки завдань;

- обробляти вибрані завдання та завантажувати їх на свій комп'ютер у форматі pdf.

Для кожного представленого завдання користувач може переглянути відповідь, вказівки з підказками, повне докладне рішення. Зовнішній вигляд сайту представлено на рис. 1.1.

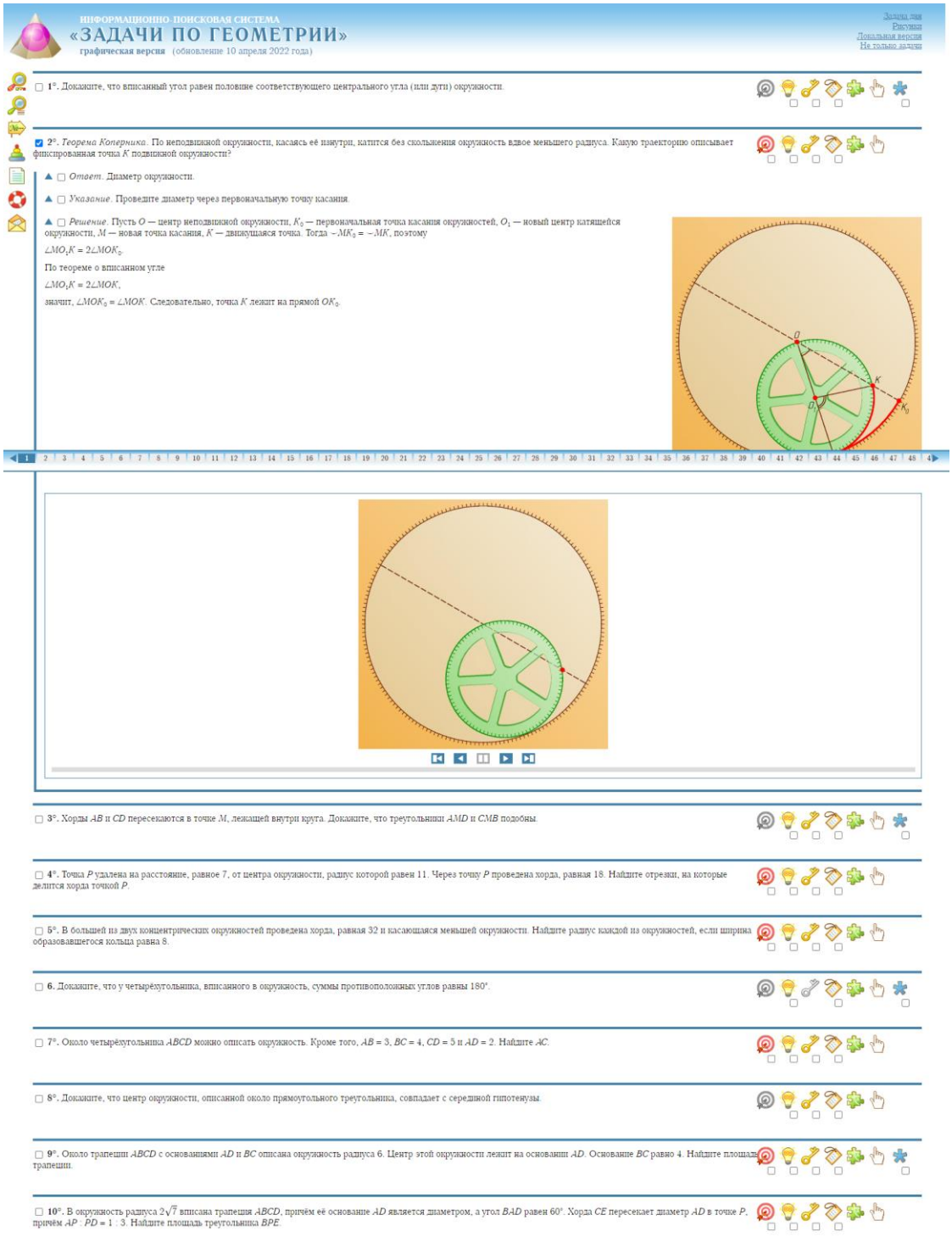

Рисунок 1.1 – Зовнішній вигляд сайту www.zadachi.mccme.ru

Сайт problems.ru містить:

- каталог задач по різним темам;
- каталог задач по різним джерелам;
- каталог задач по різним авторам;
- новинний розділ;
- можливість здійснювати пошук;
- можливість скористатися безпосередньо на сайті коротким довідником математичних термінів.

Вибравши в меню дисципліну та тематику завдань, користувач побачить список завдань (див. рис. 1.2), для багатьох з них можна також подивитися рішення (див. рис. 1.3).

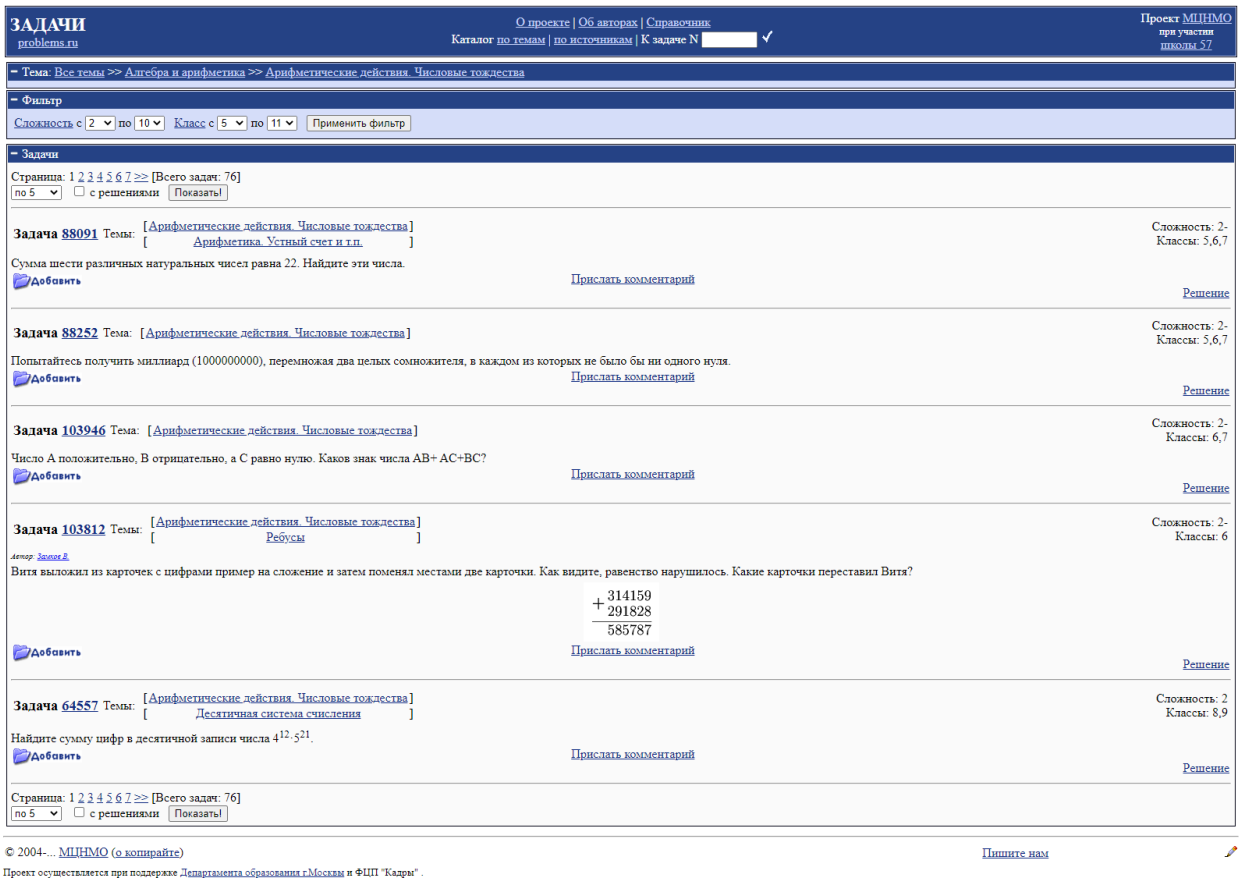

Рисунок 1.2 – Зовнішній вигляд сторінки зі списком завдань сайту problems.ru

Рисунок 1.3 – Вид сторінки із рішенням завдання сайту problems.ru

Обидва розглянуті сайти містять безліч дуже корисної для учнів інформації, однак є в них і важливий недолік - досить застарілий дизайн, незручний та незрозумілий для користувачів інтерфейс, а також не можливість створення масштабування можливостей цих сайтів через використання для розробки застарілих технологій.

Отже, було проаналізовано різні сучасні тенденції, завдяки яким освітній сайт буде більш ефективним, зручним та корисним для учнів загальноосвітніх шкіл та вищих навчальних закладів.

1.4 Шаблони проектування

У світі розробки програмного забезпечення архітектурним проектуванням можна назвати процес конструювання програми, в ході якого його прагнуть зробити якісним, надійним і таким, що буде добре піддаватися підтримці. При цьому патерни (шаблони) проектування дозволяють оперувати поняттями, що являють собою підходи до вирішення поширених проблем. Ці рішення можуть варіюватися від абстрактних, концептуальних, до гранично конкретних. Їх знання дозволяє розробникам ефективно один з одним спілкуватися.

Якщо хоча б два розробники в команді розбираються в патернах, розмова про вирішення проблем, що постають перед командою, стає дуже продуктивною.

У цій роботі буде розглянуто патерн проектування MVC.

Шаблон MVC (Модель-Вид-Контролер або Модель-Стан-Поведінка) представляє легкий варіант того, як можна будови структуру програми для того, щоб відокремити бізнес-логіку від користувацького інтерфейсу. Результатом є легке масштабування додатку, легке тестування, легкий супровід і реалізація. [4]

Бізнес-логіка програми зосереджена у моделі. Вибірка, обробка, надання певних даних - ці методи включені до моделі. За рахунок цього, вона часто буває досить товстою, але це є цілковитою нормою.

У моделі немає потреби безпосередньої взаємодії з користувачами. Обробка усіх змінних, які мають відношення до запитів користувачів, повинна проводитись у контролері.

Генерація HTML коду відображення (який змінюється залежно від потреб користувачів) не відбувається у моделі. Такий код оброблюється у видах. [1,5]

Використання, наприклад, моделі аутентифікації користувача можливе у обох частинах програми - і у користувальницькій, і в адміністративній. У таких випадках можна, ймовірно, провести винесення загального коду до окремого класу і здійснити успадкування від нього, назначаючи спадкоємцями методи, що являються специфічними для додатків.

Вид — застосовується, щоб задати як будуть відображатися дані зовнішнє, отримання яких здійснюється з контролера і моделі.

У видах міститься HTML-розмітка і невеликого обсягу фрагменти PHP-коду, необхідні для забезпечення обходу, форматування та відображення даних.

Види не здійснюють безпосереднього звернення до БД. Ця функція виконується моделлю.

Види не працюють з даними, які отримуються з запитів користувачів. Ця задача лежить на контролері.

Вид, для отримання готових до висновку даних, може здійснювати безпосереднє звернення до властивостей та методів контролера або моделей.

Види, як правило, поділяються на загальний шаблон, у якому міститься розмітка, загальна для всіх сторінок (наприклад, header та footer) та фрагменти шаблону, які використовуються, щоб відображати дані, які виводяться з моделі, або форми вводу даних.

Контролер покликаний сполучати між собою моделі, види та інші компоненти, для створення робочого додатку. Зона відповідальності контролера - обробка запитів користувачів. У контролері не місце для SQL-запитів. Найкраще, щоб вони розміщувалися у моделях. У контролері не місце для розміток (HTML тощо). Їх потрібно винести у види.

Якщо MVC додаток добре спроектований, то місткість контролеру будет складати не більше як кілька десятків строк коду, що робить його досить тонким.

А от у моделях міститься велика частина коду (для обробки даних), це пов'язано з специфікою структури даних та бізнес-логіки для конкретних програм.

На рис. 1.4 представлена одна з можливих схем роботи шаблонів MVC.

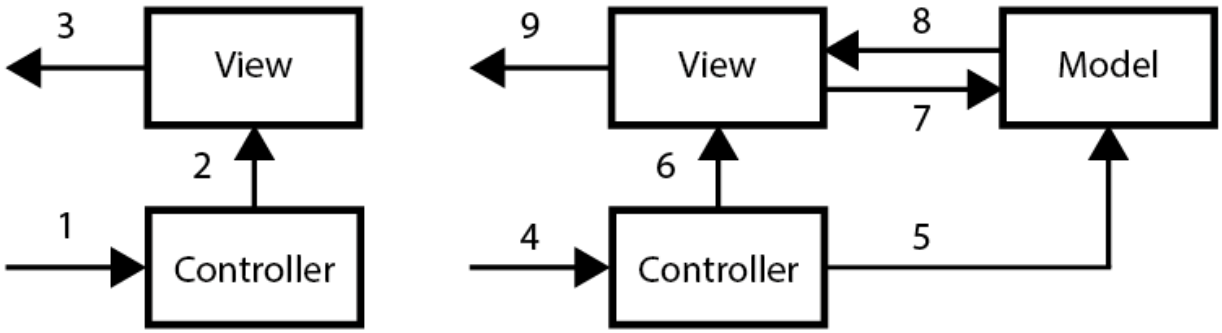

Рисунок 1.4 – Схема роботи програм MVC

1. Коли користувач заходить на веб-ресурс, скрипт ініціалізації створює приклад програми та запускає його на виконання;
2. Виконується дія index фронт-контролера, яка генерує уявлення головною сторінки;
3. Подання відображається користувачеві;
4. Після отримання додатком запита від користувача – відбувається створення екземпляру контролера, що був запитаний, і виклик вказаної дії;
5. У цій дії викликаються методи моделі, що змінюють її;
6. Генерується уявлення (або ж уявлення повідомляється про оновлення моделі);
7. Подання запитує дані для відображення;
8. Модель повертає запитані дані;
9. Подання відображає результати користувачеві.

Перші три кроки – це простий ланцюг, без використання моделі. Далі йде послідовність, де задіяна модель.

Зустрічається і інша схема – представлена на рис. 1.5.

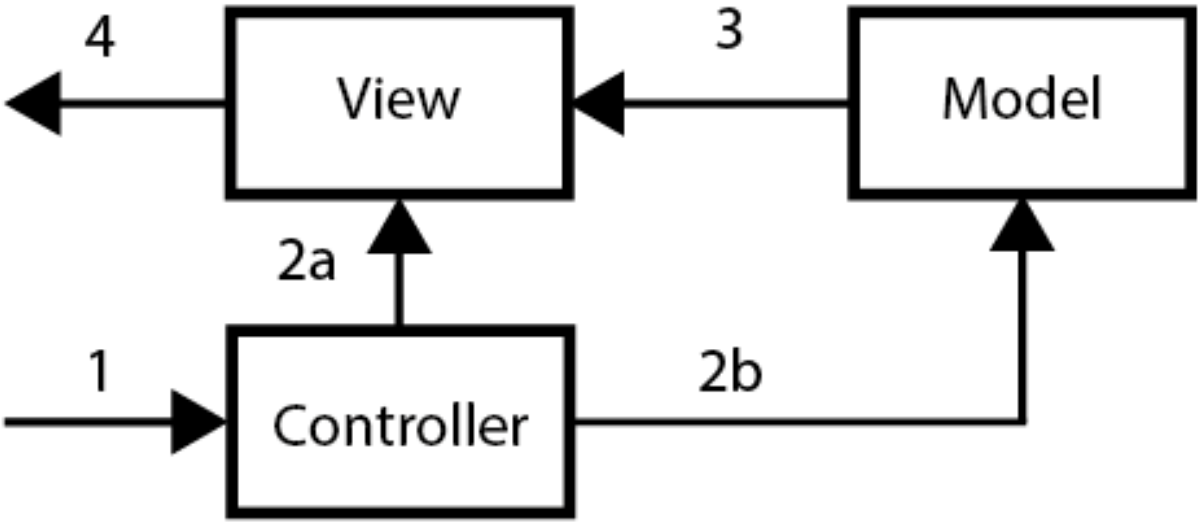

Рисунок 1.5 – Інша схема роботи шаблонів MVC.

1. Контролер отримує запит від користувача;

2. Далі, залежно від внутрішньої логіки:

2a. Формується уявлення якоїсь сторінки;

2b. Або викликаються методи моделі;

3. Модель повідомляє про зміни;

4. Подання оновлюється (якщо в ланцюжку була задіяна модель) та відображається користувачеві.

1.5 Постановка задачі

Метою дипломної роботи є створення веб платформи для проведення олімпіад з фізики для Міністерства освіти та науки України. Дану платформу потенційно можна буде розвинути для використання в майбутньому в більш широкому спектрі, розширивши його базу даних та доповнивши функціонал.

На сайті реалізувати функції фільтрації, пошуку, пагінації. Розробити адміністративну панель з можливістю додавання та редагування категорій, тегів для завдань. додавання та редагування завдань, додавання зображень до умов задач, реалізувати відображення формул, реалізувати функціоналу візуального редактору.

Для реалізації сайту та розробки вище вказаних цілей необхідно зробити наступні кроки:

- Провести пошук необхідної інформації, відповідно до теми.
- Провести аналіз аналогів даної веб платформи для визначення переваг та недоліків конкурентів, для більш успішної реалізації нашого продукту.
- Оглянути та обрати сучасні засоби реалізації.
- Спроектувати модель сайту: його адміністративну частину та користуватську.
- Створити бекенд частину
- Створення фронтенд частини
- Провести тестування та аналіз результату

Даний сайт буде інформаційною платформою для проведення тестувань по програмі з точних наук - алгебра, геометрія, фізика. Він розробляється для поліпшення умов перевірки знань учнів і студентів та полегшення роботи викладачів освітніх закладів. Найважливішою задачею ресурсу є полегшення умов оцінки рівня знань.

2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ

2.1 Вибір фреймворку Django

Python — універсальна мова програмування та її можна використовувати для реалізації будь-якого класу завдань від простого сценарію автоматизації до системного програмування, від розробки ігор до наукових графічних та веб-додатків. [6]

Веб-програма складається з двох частин: клієнтської та серверної. Серверна частина, як правило, є найбільш складною та описує всю бізнес-логіку програми. Розробка серверної частини без фреймворку втомлива, треба написати код для кожної рутинної задачі, що повторюються. Тому фреймворки і закріпилися як основа веб-розробки.

Веб-фреймворк — це заздалегідь написаний код для виконання рутинних завдань та надання абстракцій для власної програми. Наприклад, автентифікація користувачів - вбудована функція фреймворку і майже скрізь надається абстрактний рівень використання баз даних. Отже, веб-фреймворк допомагає розробляти програму швидше та ефективніше з добре написаними модулями та функціональними можливостями. Python — це проект з відкритим вихідним кодом і багато його веб-фреймворків також відкриті і безкоштовні, наприклад, Flask, Django, web2py, bottle та інші.

Порівнюючи основні фреймворки, їх можливості, функціонал (див. таблицю 2.1), популярність серед веб-розробників, було прийнято рішення про використання для реалізації проекту саме фреймворку Django.

Таблиця 2.1 – Порівняльні характеристики різних фреймворків

Параметр	Джанго	Web2py	Колба	Пляшка	CherryPy
Перший випуск	2005	2007	2010	2009	2002
Актуальна версія	2.2, квітень 2019 року	2.17.2 жовтень 2018 року	1.0.2, травень 2018 року	16 грудня 2018 року	18.1.0, грудень 2018 року
Тип	Повний	Повний	мікро	мікро	мікро
Розроб.	Адріан Головатий, Саймон Віллісон	Массімо Ді П'єрро	Армін Ронахер	Марсель Хелкамп	Ремі Делон, Роберт Брюер
Ліцензія	BSD [a]	GNU LGPL [a]	BSD [a]	MIT [a]	BSD [a]
Версія Python	Python 2.x або Python 3.x (Django 2.0 та вище)	Python 2.7 та Python 3.5+	Обидві Python 2.x та Python 3.x	Python 2.5+ та 3.x	Python 2.x та Python 3.x
Приклади	Instagram [a], Disqus [a]	Instant Press, Ourway	Pinterest [a], LinkedIn [a]	-	Splunk Enterprise

При виборі фреймворку були розглянуті деякі характеристики, отримані для цих систем у відкритому доступі з GitHub, Stack Overflow та їм подібних. GitHub та Stack Overflow є майданчиками для розробників. GitHub дає можливість ставити зірки проектам, розміщеним там, а на Stack Overflow можна ставити запитання щодо використання того чи іншого програмного забезпечення. У таблиці 2.2 представлені чотири метрики цих сайтів, які можуть допомогти зрозуміти популярність веб-фреймворків:

- Зірки Github: загальна кількість зірок проекту, виставлених користувачам.
- Релізи Github: кількість релізів кожного проекту, що відображає активність та зрілість проекту.
- Запитання Stack-overflow: кількість запитань, заданих по певній темі.
- Вакансії: кількість вакансій, пов'язаних із технологіями або IT-компетенціями.

Таблиця 2.2 – порівняння популярності фреймворків серед веб-розробників

Метрика	Джанго	Web2py	Колба	Пляшка	CherryPy
	(Повний стек)			*(Мікрофреймворк)*	
Зірки Github	46 528	1 832	48 385	6 594	1 130
Релізи Github	272	72	30	75	127
Запитання Stack-overflow	217 030	2094	32 621	1 371	1300
Вакансії	42	0	18	4	0

Django - веб-фреймворк повного стека, що підходить для великих і складних проектів. Допомагає швидко та чисто зробити веб-проект. Django є безкоштовним, з відкритим вихідним кодом. Створено і підтримується сотнею досвідчених веб-розробників. Згідно з документацією Django деякі з перерахованих у ній функцій максимально швидкі, надійно захищені, надзвичайно масштабовані та універсальні. Django підтримує роботу з основними базами даних - MySQL, SQLite, PostgreSQL та Oracle. Особливості Django – це його автентифікація, маршрутизація URL-адрес, механізм шаблонів, об'єктно-реляційний картограф (ORM) та міграція схем БД (Django v.3.0+). [16]

Сайти на Django будуються з одного або декількох додатків, котрі краще робити відчужуваними та такими, що підключаються. Це являю собою одну із важливих архітектурних відмінностей даного веб-фреймворку від деяких інших (наприклад, Ruby on Rails). Один із головних принципів веб-фреймворку - DRY (англ. Don't repeat yourself). [11]

Основні переваги фреймворку:

Надає рівень абстракції («моделі») для структурування та управління даними веб-додатку.

Реалізована концепція «представлень» для інкапсуляції логіки, відповідальної за обробку запиту користувача та повернення відповіді.

Реалізован принцип DRY (don't repeat yourself). Завдяки чому зменшується час створення веб-сайтів. А це означає, що відпадає необхідність у переписуванні одного й того ж коду.

Фреймворк надає змогу розроблювати сайти з компонентів.

Підходить для створення веб-програм з високим навантаженням.

Шаблони забезпечують зручний для дизайнера синтаксис для візуалізації інформації, яка буде представлена користувачеві.

Надає багату основу для полегшення створення форм та маніпулювання даними форм.

Наявність безлічі різних компонентів та інструментів, які допомагають у розробці та тестуванні додатків.

Наявність автоматизованого інтерфейсу адміністратора. Адміністративна панель автоматично генерується під час створення програми. Це позбавляє розробника необхідності створювати адмінку вручну.

Безпека є темою першорядної важливості при розробці веб-застосунків, і Django надає безліч інструментів та механізмів захисту.

Пропонує надійну систему інтернаціоналізації та локалізації, яка допомагає у розробці програм для різних мов та регіонів світу.

Існує безліч методик та інструментів, які можуть допомогти зробити код більш ефективним - швидшим і використовуючи менше системних ресурсів.

Географічний фреймворк GeoDjango має намір стати географічною веб-інфраструктурою світового класу. Його мета – максимально спростити

створення веб-додатків GIS та використовувати можливості просторових даних.

Розширюваність. Функціональність Django розширюється за допомогою плагінів. Це програмні модулі, що надають змогу швидко додати на сайт необхідні функції. Офіційний каталог містить дуже багато плагінів, які дозволяють легко реалізувати на сайті sitemap.xml, керувати доступати, підключити платіжну систему Stripe і таке інше.

Підтримує використання бібліотек під час створення веб-застосунків. До популярних бібліотек входять: Django REST Framework, Django CMS, Django-allauth.

SEO-дружність. Написаний на Python код виходить читабельним та зрозумілим навіть непідготовленим людям. Це один з факторів, завдяки яким веб-програми на Python вважаються SEO-дружніми. Django генерує семантичні URL-адреси. Їх також називають людино-зрозумілими URL або ЧПК. У програмах Django легко реалізуються інші функції, необхідні для пошукової оптимізації.

Реалізовано об'єктно-реляційне відображення (ORM), яке забезпечує взаємодію програми з базами даних (БД). ORM передбачає автоматичну передачу даних з БД в об'єкти, що задіяні в коді програми. ORM пришвидшує процес розробки прототипів та готових веб-додатків. Розробнику навіть не потрібно знати мову, яка використовується для взаємодії з базами даних. Також ORM дозволяє швидко перемикатися між базами даних із мінімальними змінами коду.

Django був представлений у 2005 році. За 14 років існування він зазнав суттєвих змін та вдосконалився. Фреймворк постійно розширюється новими можливостями.

Django пропонує кілька інструментів, які зазвичай необхідні для розробки веб-додатків:

- Аутентифікація;
- Кешування;
- Логування;
- Надсилання електронних листів;
- Канали синдикації (RSS/ Atom);
- Пагінація;
- Система повідомлень;
- Серіалізація;
- Сесії;
- Карти сайту (sitemaps);
- Управління статичними файлами;
- Валідація даних.

Інші основні функціональні можливості фреймворку:

- Умовна обробка контенту;
- Типи вмісту та спільні відношення;
- Прості сторінки (flatpages);
- Перенаправлення (redirects);
- Сигнали;
- Фреймворк перевірки системи;

Фреймворк Django написаний мовою програмування Python, тому його структура відповідає особливостям мови. Виробники створили в Django патерн MVC, і він задіяний в поточній версії фреймворку.

Архітектура MVC надає змогу розробникам працювати з візуальним поданням та бізнес-логікою програми окремо. До речі, під час роботи з Django фахівці частіше використовують термін MVT — Model-View-Template або модель-вистава-шаблон. Компоненти MVT можна використовувати незалежно один від одного. Схема архітектури MVT Django представлена на рис. 2.1.

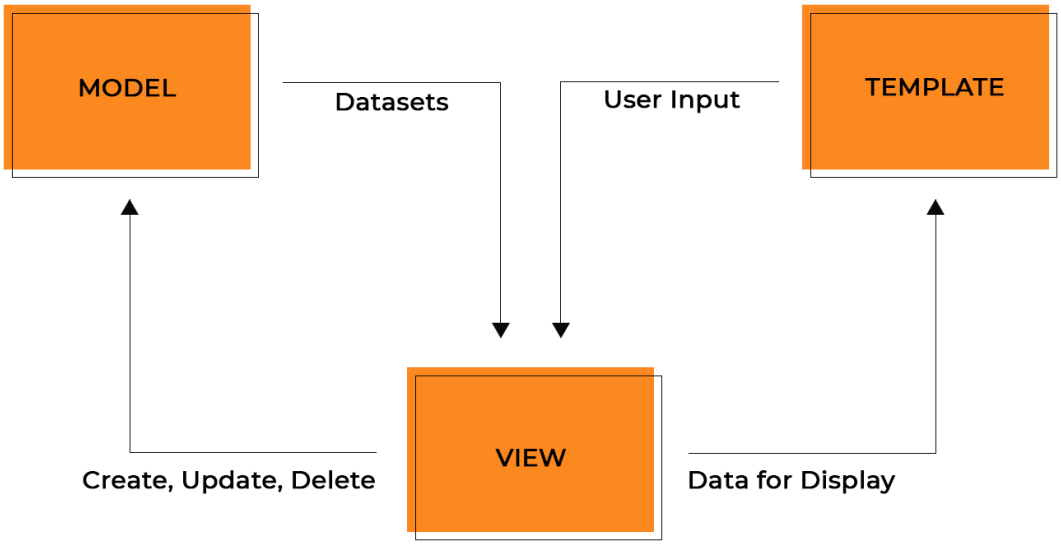

Рисунок 2.1 - Схема архітектури MVT в Django

У Документації Django визначено модель (model) як «інформаційне джерело стосовно даних, яке містить ключові поля та поведінку даних». Частіше за все, одна модель має вказувати на одну таблицю у БД.

У моделях зберігається інформація про дані, що є представленими атрибутами чи полями. Через те, що модель є простим класом, вона не має знань про інші рівні фреймворка. Рівні взаємодіють між собою за посередництва API.

Бізнес-логіка, методи, властивості та інші елементи, що є пов'язаними з маніпуляціями даними - все це є відповідальністю моделі. Крім того, моделі надають змогу розробникам створювати, читати, оновлювати та видаляти об'єкти в базі даних.

Представлення (view) допомагає вирішити три головні завдання: приймання HTTP-запитів, реалізацію бізнес-логіки, визначеної методами та властивостями, відправлення HTTP-відповіді у відповідь на запити. Після отримання даних від моделі, здійснюється надання представленням шаблонам доступа до даних або попереднє оброблення даних, а вже далі - надається доступ до них шаблонів.

У Django реалізований потужний двигун шаблонів та власна мова розмітки. Шаблони (див. рис. 2.2) є файлами з HTML-кодом, за допомогою якого здійснюється відображення даних. Файли можуть мати статичний або динамічний вміст. Бізнес-логіка в шаблонах не міститься. Їх функція - лише відображення даних.

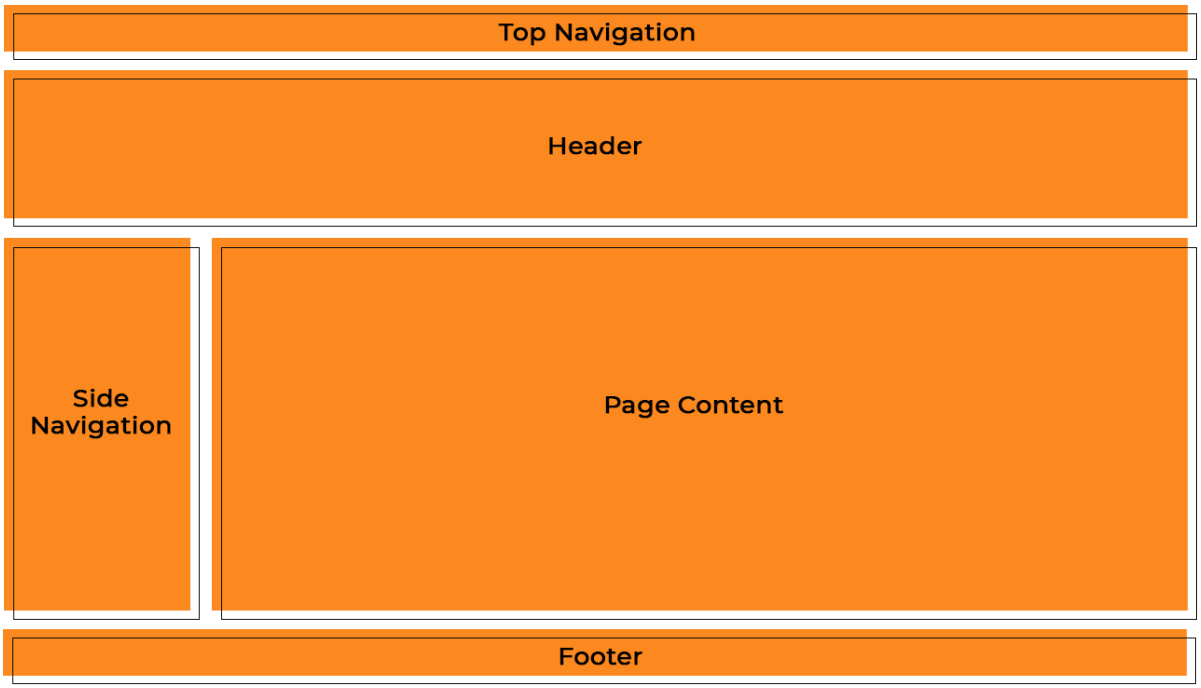

Рисунок 2.2 - Шаблон сторінки

Така архітектура дозволяє Django відмінно вирішувати різноманітні задачі.

Фреймворк Django справляється з величезним числом завдань та високими навантаженнями. Його застосовують для створення:

- CRM-систем;

- CMS;
- Комунікаційних платформ;
- Сервісів для здійснення бронювання номерів.
- Платформ керування документообігом.

Також Django є підходящим для створення генераторів алгоритмів, платформ для електронних розсилок, верифікаційних систем, фільтраційних систем з динамічними правилами та складними параметрами, платформ для аналізу даних та складних обчислень, машинного навчання.

Тисячі сайтів у різних країнах світу створені на Django. Цей фреймворк чудово підходить для розробки веб-застосунків. Даний веб-фреймворк був обраний мною для реалізації проекту завдяки наявності в ньому найважливіших функцій і можливостей – розширюваності, розвинутій інфраструктурі, великій кількості бібліотек та плагінів, SEO-дружності, поділу бізнес-логіки та візуальної частини на рівні архітектури.

2.2 Вибір середовища розробки PyCharm

PyCharm – це інтегроване середовище розробки, створене спеціально для Python. Розроблена компанією JetBrains. Оскільки редактор розроблений спеціально для Python, він має широкий набір можливостей, таких як автозавершення та інспекції коду, підсвічування помилок, виправлення, налагодження, система контролю версій та рефакторинг. IDE доступна на Microsoft Windows, Linux та MacOS. Доступні безкоштовна та платна професійна версії. Професійна IDE має кілька додаткових функцій, але безкоштовної версії достатньо для виконання більшості завдань. Вигляд середовища розробки розробки див. рис. 2.3.

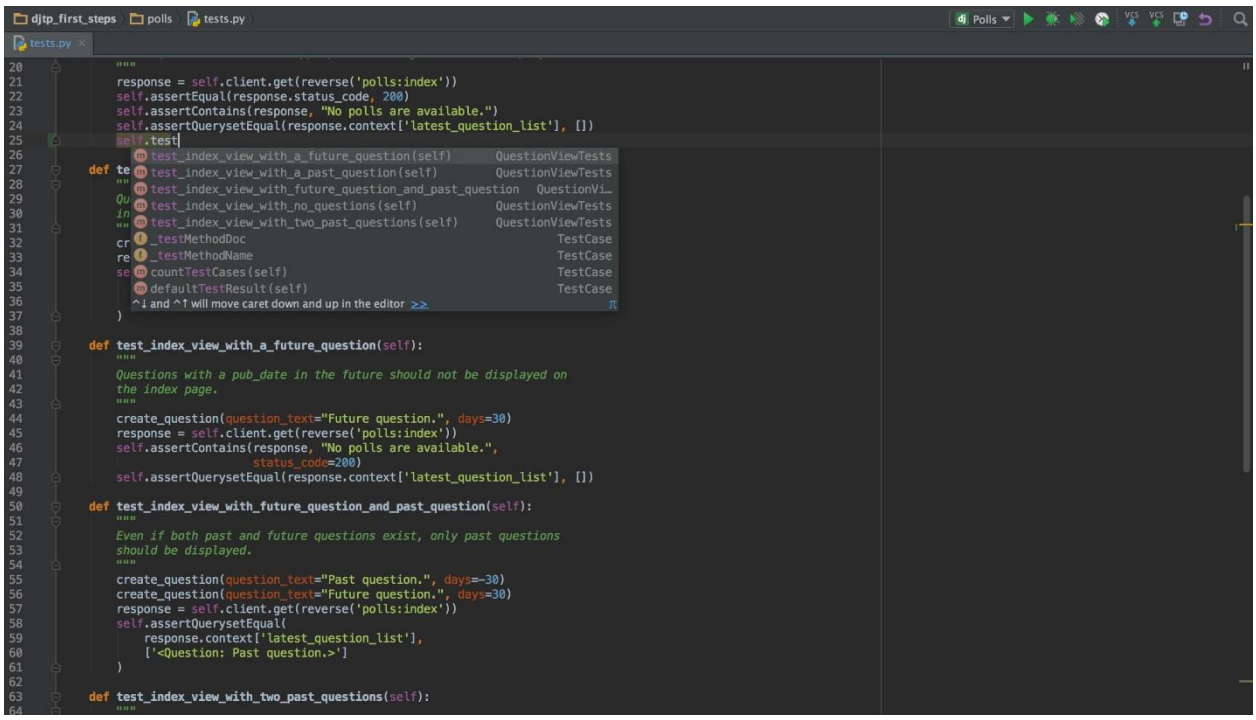

Рисунок 2.3 - Вигляд вікна IDE PyCharm

Основні переваги середовища розробки PyCharm :

1. Середовище забезпечує підтримку популярних веб-фреймворків, таких як Django, Flask, Google App Engine, Pyramid та web2py;
2. IDE забезпечує першокласну підтримку мов JavaScript, CoffeeScript, TypeScript, HTML та CSS, а також їх сучасних наступників. Відладчик JavaScript включений у PyCharm та інтегрований з Run-конфігурацією запуску сервера Django; [3]
3. Наявність функції Live Editing Preview, яка дозволяє відкрити редактор і браузер одночасно і слідкувати за результатами внесених в код змін на веб-сторінці. PyCharm зберігає зміни автоматично і вони миттєво відображаються в браузері без перезавантаження сторінки;
4. Інтерактивна консоль REPL для Python, яка має багато переваг над стандартною консоллю. Серед них перевірка синтаксису на льоту за допомогою інспекцій, зіставлення дужок та лапок та автодоповнення.

5. Підтримує Pandas, Numpy, Matplotlib та інші бібліотеки для наукових обчислень. IDE забезпечує розумне редагування, дозволяє переглядати набори даних у вигляді графіків та у табличній формі;
6. PyCharm - крос-платформна IDE, що настроюється, яку можна встановити за допомогою одного ліцензійного ключа на декількох комп'ютерах в операційних системах Windows, Mac OS або Linux - всі можливості IDE будуть доступні на будь-якому з них;
7. Є можливість налаштовувати робоче середовище: вибирати відповідну колірну схему та зручні поєднання клавіш, включати режим емуляції VIM.

PyCharm робить розробку максимально продуктивною завдяки функціям автодоповнення та аналізу коду, миттєвому підсвічуванню помилок та швидким виправленням. Автоматичні рефакторинги допомагають ефективно редагувати код, а зручна навігація дозволяє миттєво переміщатися за проектом.

У даному IDE легко редагувати код завдяки безлічі функцій, а саме:

Автоматична розстановка відступів та форматування. Відступи додаються автоматично на початку нового рядка. Перевірка коректності відступів та автоматичне переформатування виконуються відповідно до налаштувань стилю коду проекту.

Автодоповнення – редактор пропонує варіанти автодоповнення для ключових слів, класів та змінних, коли друкується код. Автодоповнення враховує контекст та пропонує найбільш підходящі варіанти.

Форматування коду. Можливості форматування та налаштування стилю коду допомагають писати зрозумілий код, який легко підтримувати. У PyCharm є вбудований форматор коду, що відповідає рекомендаціям стандарту PEP-8 для Python, а також інших стандартів для мов, що підтримуються.

Сніппети – дозволяють прискорити роботу.

Згортання коду. Доступна функція згортання блоків коду, автоматичне розміщення дужок і лапок, підсвічування парних дужок тощо.

Підсвічування помилок на льоту. Помилки відображаються під час набору коду. Вбудована перевірка орфографії попередить про помилки у текстах ідентифікаторів та коментарів.

Аналіз коду. Численні інспекції перевіряють код прямо у режимі редагування, а також дозволяють проаналізувати весь проект на наявність помилок та проблем у структурі коду.

Швидкі виправлення. Для більшості інспекцій доступні швидкі виправлення, які дозволяють відкоригувати код миттєво.

Пошук дублікатів у коді. За допомогою розумного детектора дублікатів редактор перевіряє код на наявність фрагментів, що дублюються. IDE запропонує список фрагментів, які слід перетворити, а рефакторинги допоможуть позбутися повторюваного коду.

Мовні вставки, що конфігуруються, дозволяють редагувати код, написаний не на Python, всередині рядкових літералів. При цьому доступні функції автодоповнення, підсвічування помилок та інші можливості IDE.

Автоматична генерація коду з використанням за допомогою швидких виправлень, docstrings, верифікації узгодження коду; автооновлення коду при рефакторингу. Автоматична генерація шаблонів docstrings (для reStructuredText, Epytext, Google та NumPy).

Підсвічування синтаксису. Код читається легко завдяки можливостям налаштування кольорів підсвічування синтаксису Python та шаблонів Django.(див. рис. 2.4)

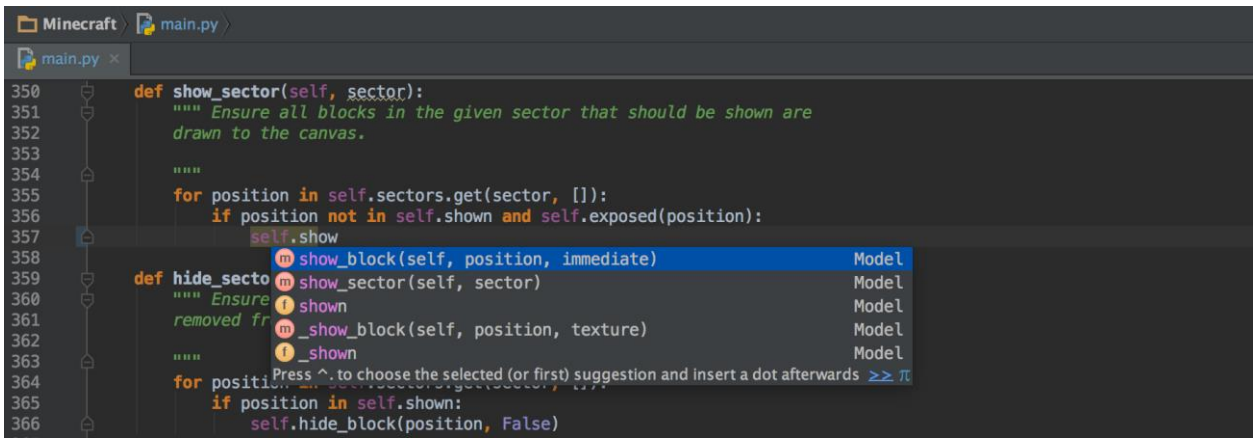

Рисунок 2.4 - Підсвічування _ синтаксису в PyCharm

Ще одна незаперечна перевага цієї IDE – зручність навігації. Так, PyCharm містить у собі:

- фукцію Search Everywhere, яка допоможе знайти клас, файл, дію чи елемент інтерфейсу IDE, як показано на рис. 2.5;
- можливість переходу до класу, файлу або символу - операції Go to class / file / symbol (необхідні для швидкої навігації за проектом).
- функцію Go to declaration, яка відкриває відповідний файл та переходить до оголошення символу.
- функцію Find Usages, яка знаходить входження будь-якого символу (класу, методу, поля тощо) у поточному файлі або по всьому проекту.

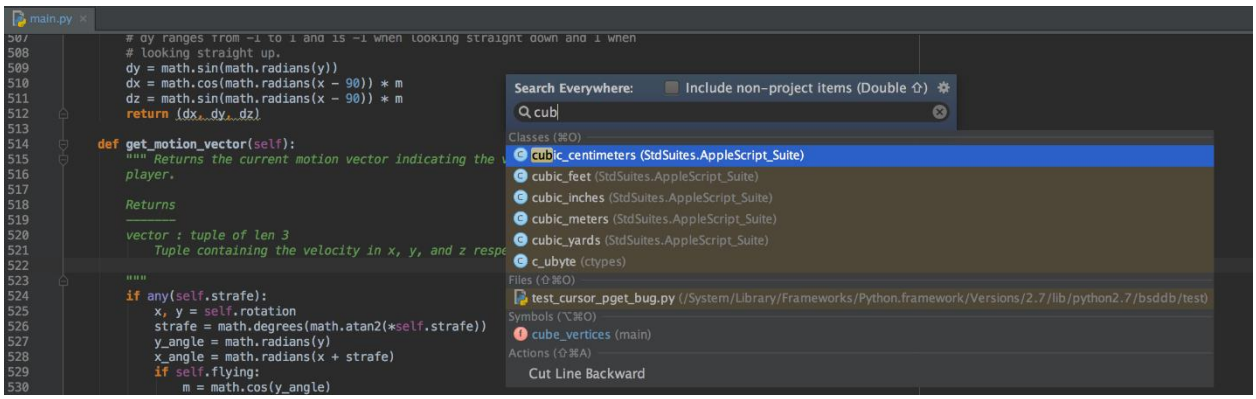

Рисунок 2.5 - Робота функції Search Everywhere в PyCharm

PyCharm допомагає вносити глобальні зміни у проект просто та безпечно. Локальні зміни відбуваються миттєво. Рефакторинги доступні для проектів на Python, а також Django, Flask, Pyramid та інших фреймворків.

Рефакторинги Rename та Move застосовні для файлів, функцій, констант, класів, властивостей, методів, параметрів, локальних та глобальних змінних. (див. рис. 2.6)

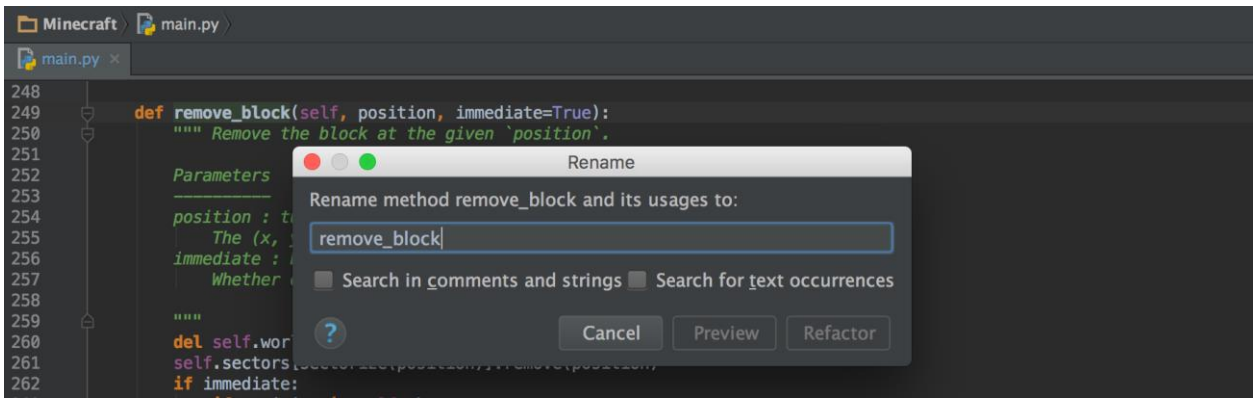

Рисунок 2.6 - Робота Рефакторингу Rename в PyCharm

Рефакторинги вилучення змінної/поля/константи/параметри та підстановки локальної змінної використовуються для покращення структури коду всередині методу.

Рефакторинг Extract Method дозволяє витягти метод із фрагмента коду, а рефакторинги Extract Superclass, Push Up, Pull Down допомагають з реорганізацією ієрархії класів та методів у проекті.

PyCharm надає широкі можливості налагодження коду на Python / Django та JavaScript :

- Можливість розставляти точки зупинки і задавати умови їх спрацьовування прямо в редакторі, перевіряти контекстно-залежні локальні змінні та зумовлені користувачем watches, включаючи масиви та складні об'єкти, редагувати значення на льоту;

- Вбудований відладчик. У режимі вбудованого налагодження значення змінних, параметрів функцій та інших об'єктів доступні у вікні редактора.
- Трасування тільки за кодом проекту. Функція Step into My Code дозволяє в режимі налагодження здійснювати трасування тільки за кодом проекту, не заглиблюючись у бібліотечні джерела.
- Налагодження кількох процесів. Дане середовище розробки вміє налагоджувати програми, які породжують кілька процесів Python, наприклад, програми Django, які не запускаються в режимі no-reload, або програми, які використовують інші веб - фреймворки, в яких реалізований аналогічний підхід до автоматичного перезавантаження коду.
- Конфігурації Run / Debug. При кожному виконанні скрипта/тесту або налагодження створюється спеціальна конфігурація запуску/налагодження, яку можна змінити та використовувати повторно.

PyCharm підтримує всі основні реалізації мови Python: Python 2.x та 3.x, Jython, IronPython, PyPy та Cython, забезпечуючи:

- Підсвічування синтаксису, перевірку коду на помилки, можливості форматування;
- Автодоповнення коду, що враховує контекст;
- Навігацію за кодом та перегляд структури;
- Швидкий пошук входжень та інструменти рефакторингу;
- Інспекції коду та багато іншого;
- Юніт-тестування.

PyCharm дозволяє легко виконувати модульне тестування завдяки інтеграції з популярними тестовими фреймворками: doctests, nose та attest.

У IDE можна запустити тестовий файл, тест для окремого класу або методу, або всі тести з папки одночасно. Результати тестів зручно переглядати в графічному інтерфейсі інструменту запуску тестів, який відображає статистику їх виконання і дозволяє швидко переміщатися за кодом, що тестується.

Редактор інтергований із інструментом аналізу покриття коду Coverage.py. Він відстежує, які частини програми були виконані, та аналізує вихідний код, щоб визначити, які фрагменти коду могли бути виконані, але цього не сталося. Результати відображаються у наочному форматі для подальшого аналізу та швидкої навігації за кодом.

PyCharm підтримує два найбільш популярні інструменти для розробки з використанням BDD - Behave і Lettuce. Це дає можливість писати людинозрозумілий код тестів, що описує поведінку програми. [12]

Візуалізація паралельних потоків допомагає контролювати багатопотокові програми.

Підтримка Git, SVN, Mercurial, Perforce та інших систем контролю версій допомагає керувати локальними змінами та здійснювати складні операції з гілками.

IDE відстежує будь-які зміни у вихідних файлах, захищає від випадкової втрати даних і не дає іншим програмам вносити зміни до проекту. Завдяки чому завжди є можливість переглянути історію змін файлу чи каталогу та відкотитися до будь-якої попередньої версії.

При редагуванні баз даних (див. рис. 2.7), доступ до Oracle, SQL Server, PostgreSQL, MySQL та інших баз даних здійснюється прямо з IDE. PyCharm допомагає редагувати SQL-код, переглядати дані, змінювати схеми та таблиці, виконувати запити та аналізувати схеми за допомогою UML-діаграм.

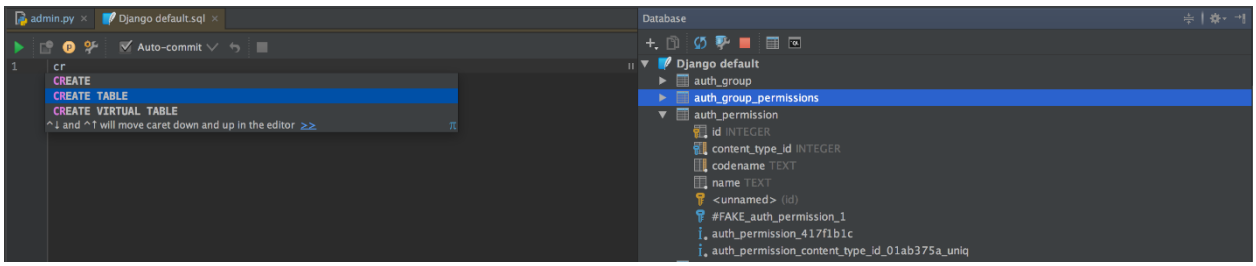

Рисунок 2.7 - Робота з базами даних PyCharm

Вбудована SSH-консоль дозволяє підключатися до будь-якого віддаленого комп'ютера та виконувати різні дії через SSH. Крім того, можна настроїти SSH Remote Tools для запуску будь-якого віддаленого інструменту безпосередньо з IDE одним натисканням клавіші. Налаштування віддаленої роботи зображено на рис. 2.8.

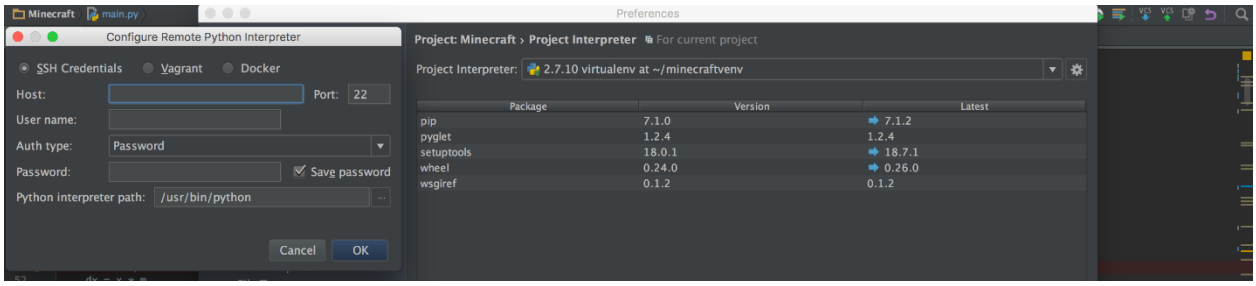

Рисунок 2.8 - Можливості віддаленої розробки в PyCharm

Редактор інтегрований з Docker, популярною платформою для розподілених додатків та контейнерної віртуалізації.

Єдина проблема, з якою можна зіткнутися під час роботи з PyCharm: низька швидкість роботи при недостатньому обсязі оперативної пам'яті комп'ютера. Для оптимальної роботи IDE необхідно мінімум 8 Гб оперативної пам'яті.

В цілому, PyCharm – це зручне, сучасне, зрозуміле, надійне та продуктивне середовище для програмування, розроблена спеціально для Python. Дана IDE містить у собі всі найважливіші вбудовані функції. Саме тому PyCharm був обраний для реалізації практичної частини даної роботи.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Інформаційна модель та структура веб-сайту

Щоб краще розуміти завдання, була розроблена спеціальна схема (див. рис. 3.1) на якій зображено структуру веб-сайту для звичайних користувачів. На схемі чітко видно, які сторінки доступні звичайному гостю сайта, це лише початкова схема, завдяки великими можливостям Django, сайт легко можна розширити, і додати нові сторінки, якщо буде потрібно.

Рисунок 3.1 - Структура сайту для звичайних користувачів

Далі було розроблено структуру адміністративної панелі веб-сайту (див. рис. 3.2), була спроектована можливість керувати групами користувачів, створювати ці групи або видаляти, тобто наприклад можна створити декілька груп користувачів, наприклад можна додати групу модераторів, які будуть додавати завдання на сайт, редагувати їх, але не мати можливості керувати адміністраторами сайту, таким чином можна гнучко налаштувати веб-сайт, і наймати працівників для роботи, не боячись що вони можуть все зіпсувати. Всі ці та інші можливості були закладені у сайт, на моменті його проектування.

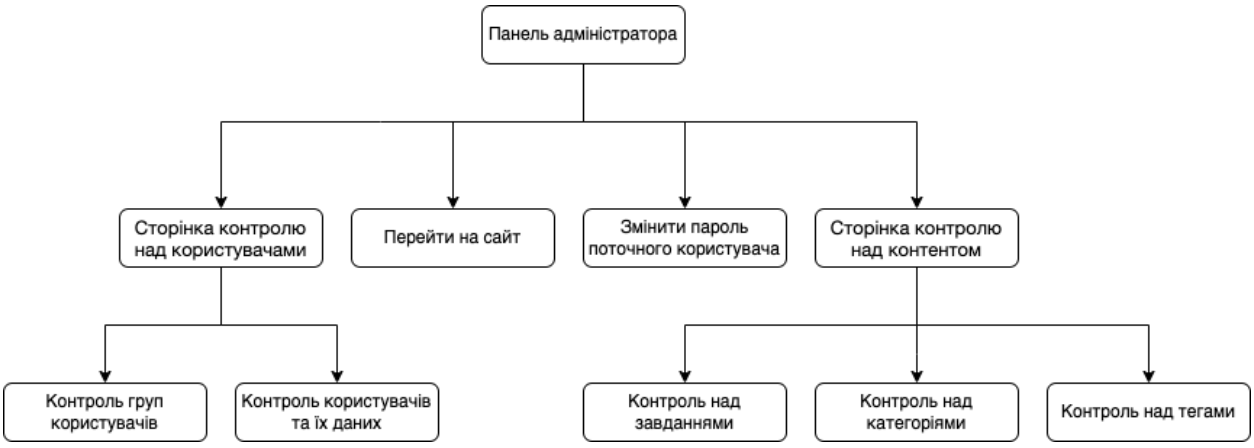

Рисунок 3.2 - Структура сайту для адміністратора

3.2 Огляд реалізованого функціоналу

Так як на сайті буде багато різних завдань, із математики, фізики, та при бажанні інших точних наук, із самого початку проектування сайту було поставлене завдання для реалізації фільтрації на сайті, для зручного пошуку необхідної задачі. (див. рис. 3.3)

Рисунок 3.3 - Вигляд полей для фільтрації завдань

За допомогою цього коду це було реалізовано:

```
import django_filters
```

```
from .models import *
class TaskFilter(django_filters.FilterSet):
    class Meta:
        model = Task
        fields = {'question': ['icontains'],
                  'difficulty': ['exact'],
                  'algebra': ['exact'],
                  'basic': ['exact'],
                  'beautiful': [,exact']}
```

Також для реалізації цього треба було підключити дві бібліотеки у файлі «settings.py», а саме «django_filters» та «bootstrapform».

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'blog.apps.BlogConfig',
    'debug_toolbar',
    'ckeditor',
    'django_filters',
    'bootstrapform',
]
```

Для зручного додавання зображень, форматування тексту, додавання посилань та багато іншого прямо через спеціальну форму у адміністративній панелі, був підключений розумний візуальний редактор типу WYSIWYG (розшифровується абревіатура як What You See Is What You Get, а якщо перекласти на українську вийде «що бачиш, то і отримаєш»), для підключення було використано готову бібліотеку для Django під назвою ckeditor. (див. рис. 3.4)

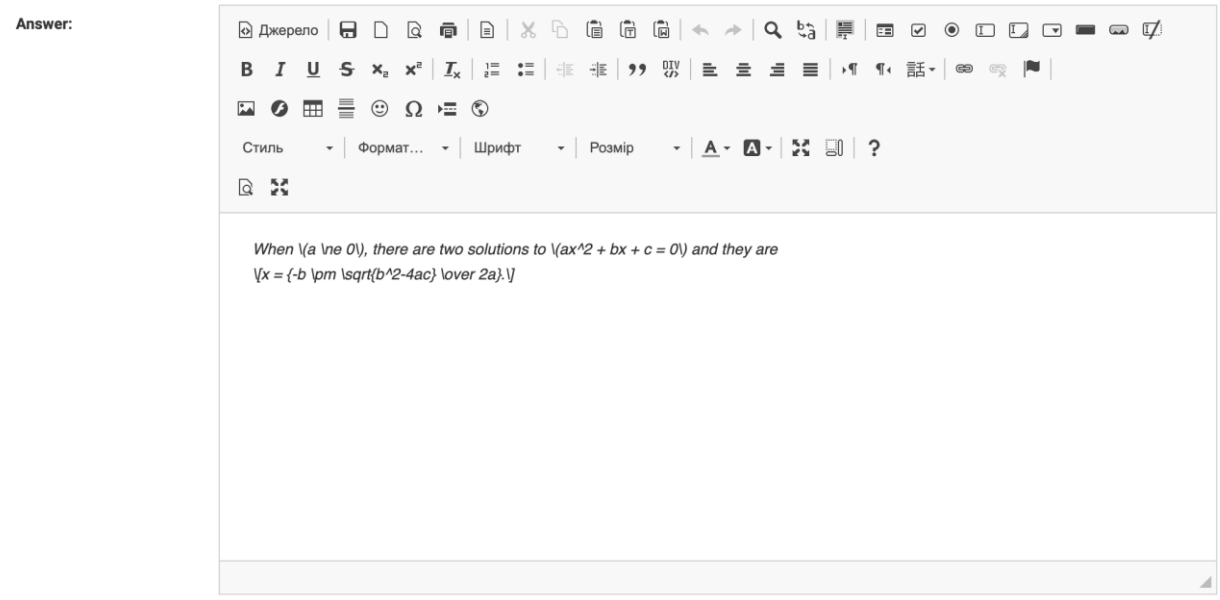

Рисунок 3.4 - Застосування візуального редактора WYSIWYG на прикладі

Для реалізації цього функціоналу був написаний наступний код:

```
from django.contrib import admin
from django import forms
from .models import *
from ckeditor_uploader.widgets import CKEditorUploadingWidget
class TaskAdminForm(forms.ModelForm):
    question = forms.CharField(widget=CKEditorUploadingWidget())
    answer = forms.CharField(widget=CKEditorUploadingWidget())
    solution = forms.CharField(widget=CKEditorUploadingWidget())
    class Meta:
        model = Task
        fields = '__all__'
class TaskAdmin(admin.ModelAdmin):
    # prepopulated_fields = {"slug": ('id',s))}
    # save_as = True
    form = TaskAdminForm
    list_display = ('id', 'title', 'slug', 'category',
'created_at')
    list_display_links = ('id', 'title')
```

```
    search_fields = ('title',)
    list_filter = ('category', 'tags')
    save_on_top = True
    readonly_fields = ('views', 'created_at', 'slug',)
    fields = ('title', 'slug', 'question', 'solution', 'answer',
'difficulty', 'algebra', 'basic', 'beautiful',
              'category', 'tags', 'views', 'created_at')
class CategoryAdmin(admin.ModelAdmin):
    prepopulated_fields = {"slug": ("title",)}
class TagAdmin(admin.ModelAdmin):
    prepopulated_fields = {"slug": ("title",)}
admin.site.register(Category, CategoryAdmin)
admin.site.register(Tag, TagAdmin)
admin.site.register(Task, TaskAdmin)
```

Було реалізовано адміністративну панель, а саме розділ для керування завданнями, тут їх можна переглядати, додавати, редагувати, видаляти та багато іншого. (див. рис. 3.5)

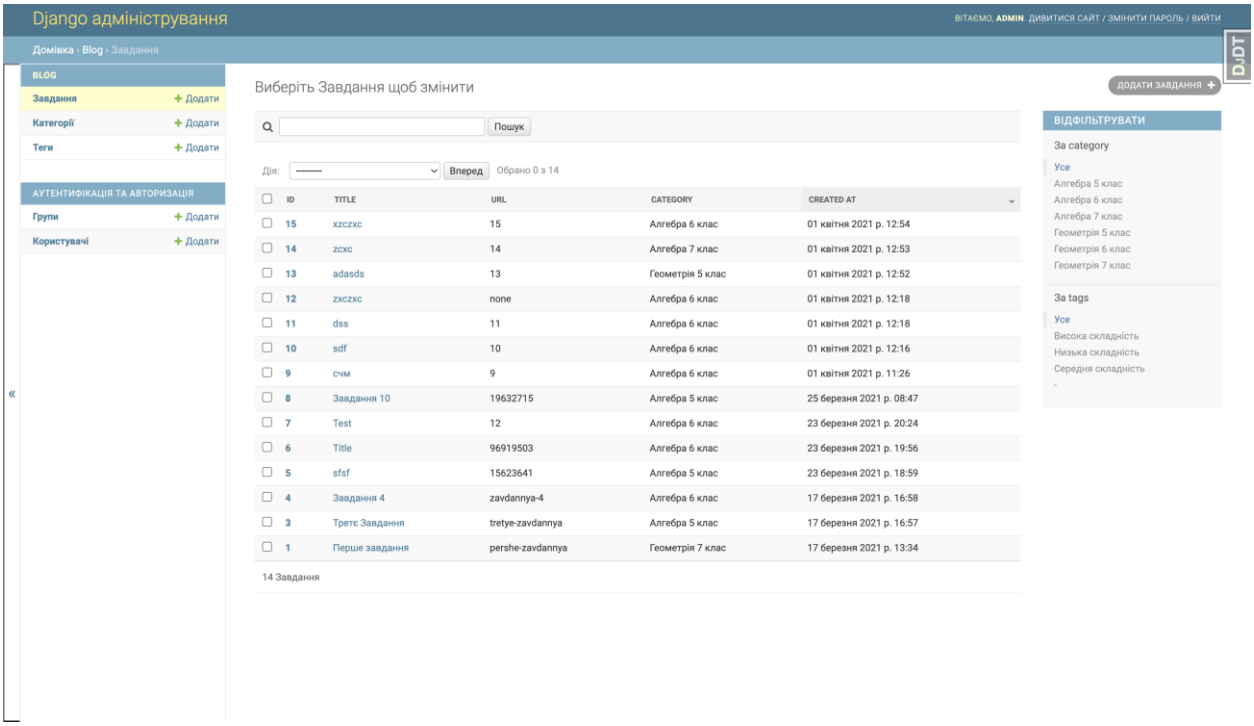

Рисунок 3.5 - Вигляд завдань в адміністративній панелі

Головна частина коду, яка за це відповідає:

```
class Task(models.Model):
    title = models.CharField(max_length=255)
    slug = models.SlugField(max_length=8, verbose_name='url',
unique=True)
    question = models.TextField(default="")
    solution = models.TextField(default="")
    answer = models.TextField(default="")
    difficulty = models.PositiveSmallIntegerField(default=1,
validators=[MinValueValidator(1), MaxValueValidator(100)])
    algebra = models.BooleanField(default=True)
    basic = models.BooleanField(default=True)
    beautiful = models.BooleanField(default=True)
    created_at = models.DateTimeField(auto_now_add=True)
    views = models.IntegerField(default=0)
    category = models.ForeignKey(Category, blank=True,
on_delete=models.PROTECT, related_name='tasks')
    tags = models.ManyToManyField(Tag, blank=True,
related_name='tasks')
    def __str__(self):
        return self.title
    def save(self, *args, **kwargs):
        super(Task, self).save(*args, **kwargs)
        if not self.slug:
            self.slug = str(self.id)
            self.save()
    def get_absolute_url(self):
        return reverse('task', kwargs={"slug": self.slug})
    class Meta:
        ordering = ['-created_at']
        verbose_name = 'Завдання'
        verbose_name_plural = 'Завдання'
```

Щоб робота над сайтом велася максимально зручно потрібно знайти багато даних про роботу сайту, саме тому був піключений плагін для дебагінга.

Так виглядає цей плагін на сайті, (див. рис. 3.6), це такий прямокутник, який прикріплений до правої верхньої частини сторінки, він видний лише адміністратору, тобто звичайні користувачі не зможуть ним користуватися. На цьому прямокутнику написано DjDT, що можна розшифрувати як Django Debug Tools, тобто це зручні інструменти для дебагінгу.

Рисунок 3.6 - Вигляд плагіна на сторінці

А ось так цей плагін виглядає, якщо натиснути на прямокутник, і розгорнути плагін (див. рис. 3.7), одразу стає видно багато даних, наприклад версія фреймворку Django, час який знадобився йому для того щоб згенерувати сторінку, який шаблон був використаний для зображення даних, скільки було запитів до бази даних та як довго база даних генерувала відповідь, кількість статичних файлів, дані про помилки, якщо вони є, та таке інше. Загалом тут зібрані базові дані необхідні для зручного дебагінгу та розробки сайту.

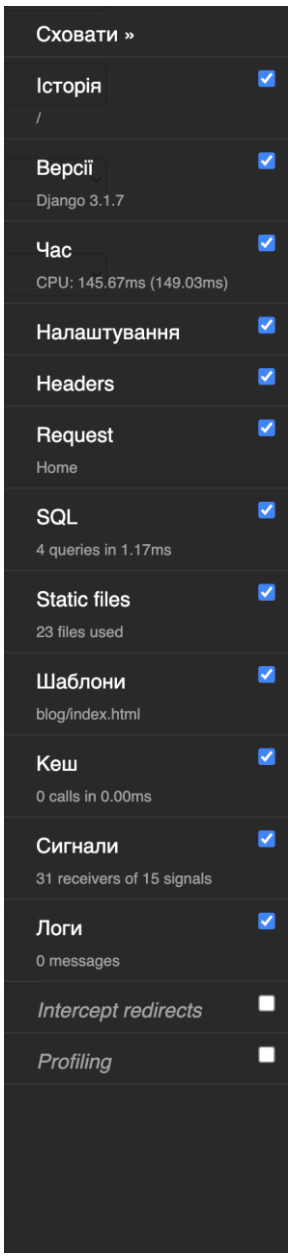

Рисунок 3.7 - Вигляд плагіна на сторінці якщо його розгорнути

Щоб увімкнути цей плагін треба спершу у файлі «settings.py» треба знайти значення «DEBUG» і змінити на «True»:

```
# SECURITY WARNING: don't run with debug turned on in
production!

DEBUG = True
```

Після цього необхідно у цьому ж файлі, підключити бібліотеку «debug_toolbar», шляхом додавання відповідного тексту у масив під назвою

«INSTALLED_APPS», потім треба перезапустити і інструмент буде встановлено.

```
INSTALLED_APPS = [
    ,debug_toolbar',
]
```

Для того щоб користувачі могли зручно корисуватися сайтом, була розроблена система, що дозволяє додавати теги і категорії, а потім додавати ці дані до математичних завдань на сайті, щоб завжди можна було зручно знайти завдання із відповідною категорією або тегом.

Процес вибору категорії чи додаванні тегів виглядає так, (див. рис. 3.8), користувач якому дали дозвіл на додавання чи редагування завдань, це може бути адміністратор чи модератор, може створити нове завдання на сайті, а також вибрати категорії, та додати теги до нього.

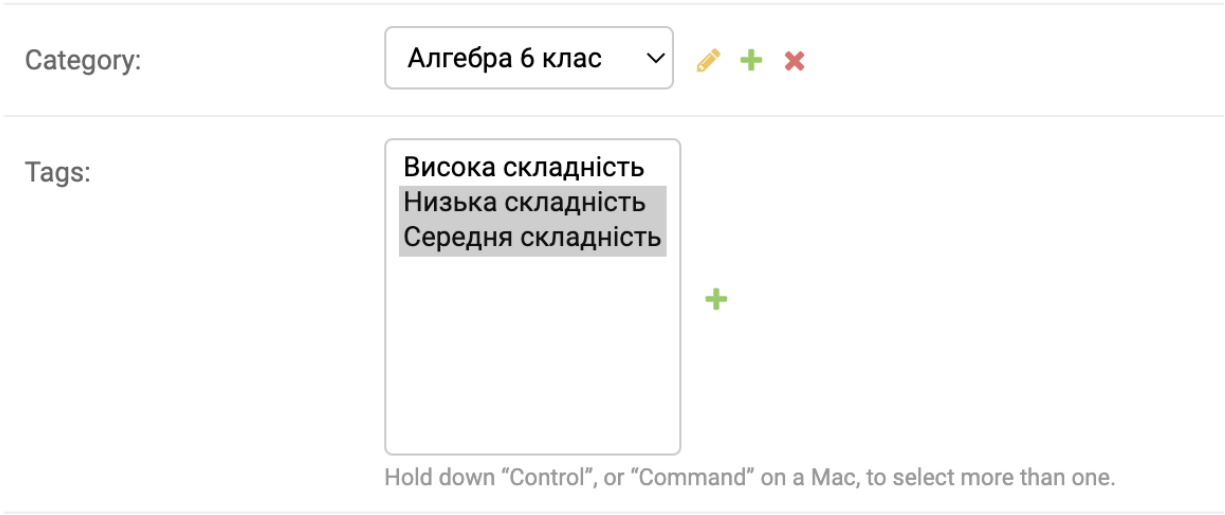

Рисунок 3.8 - Додавання категорій та тегів до завдання

Також у адміністративній панелі реалізована можливість фільтрації за тегом чи категорією, для зручності роботи. (див. рис. 3.9)

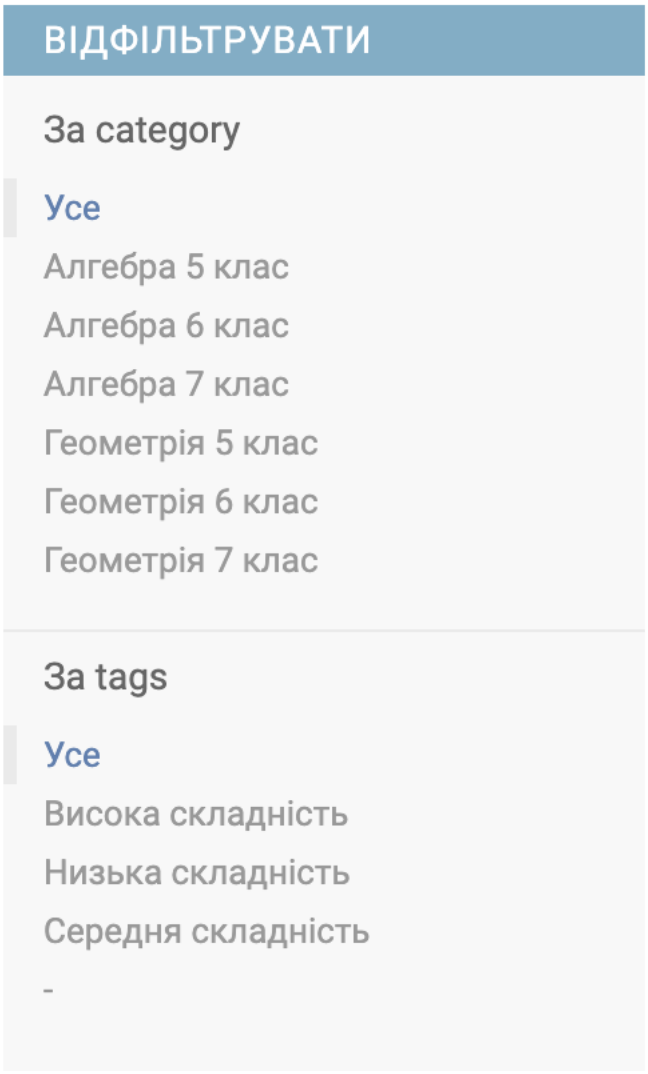

Рисунок 3.9 - Фільтрація за категорією чи тегом у адміністративній панелі

За допомогою цього коду, була реалізована можливість роботи із категоріями та тегами:

```
class Category(models.Model):
    title = models.CharField(max_length=255)
    slug = models.SlugField(max_length=255, verbose_name='url', unique=True)
    def __str__(self):
        return self.title
    class Meta:
        ordering = ['title']
        verbose_name = 'Категорія(ю)'
```

```
        verbose_name_plural = 'Категорії'
class Tag(models.Model):
    title = models.CharField(max_length=50)
    slug = models.SlugField(max_length=255, verbose_name='url',
unique=True)
    def __str__(self):
        return self.title
    class Meta:
        ordering = ['title']
        verbose_name = 'Тег(и)'
        verbose_name_plural = 'Теги'
```

3.3 Тестування та аналіз результатів

Для того щоб перевірити чи коректно працює сайт, додамо нове завдання із математики на нього, почнемо із авторизації, переходимо на сторінку для входу і вводимо логін та пароль адміністратора. (див. рис. 3.10)

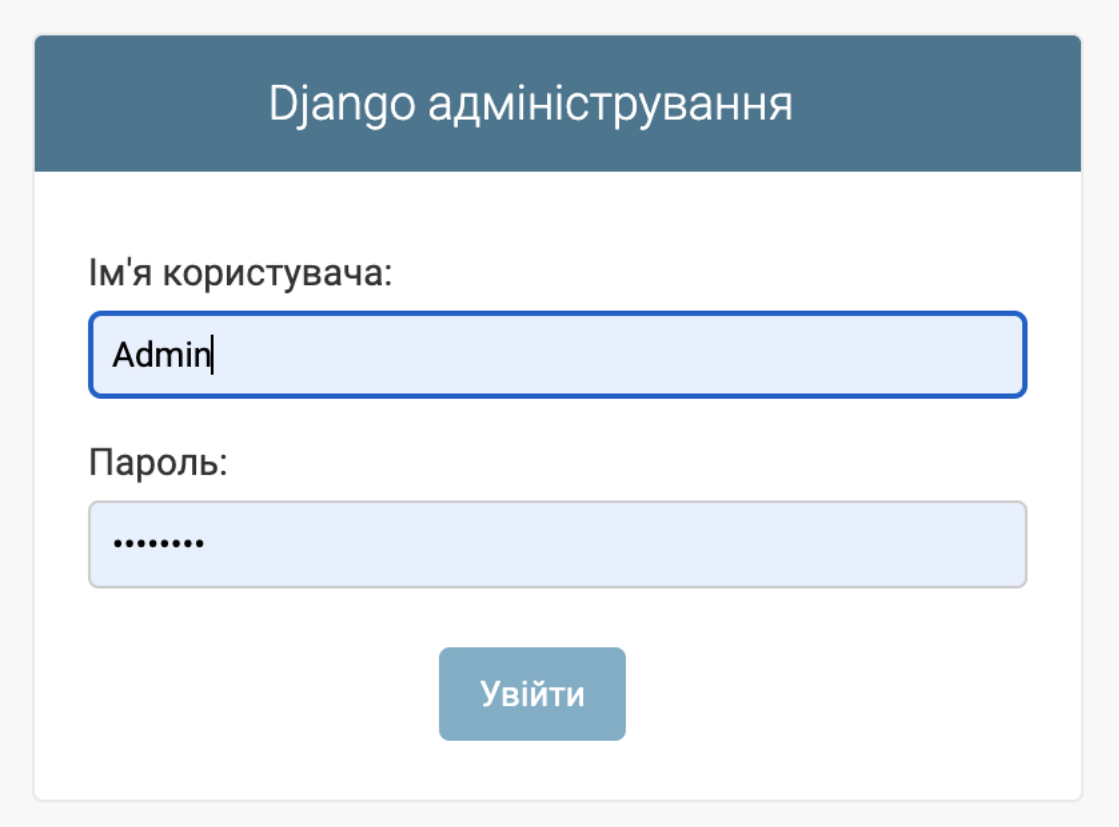

Рисунок 3.10 - Форми для авторизації

Після авторизації ми бачимо панель для адміністрування сайту із базовою структурою. (див. рис. 3.11)

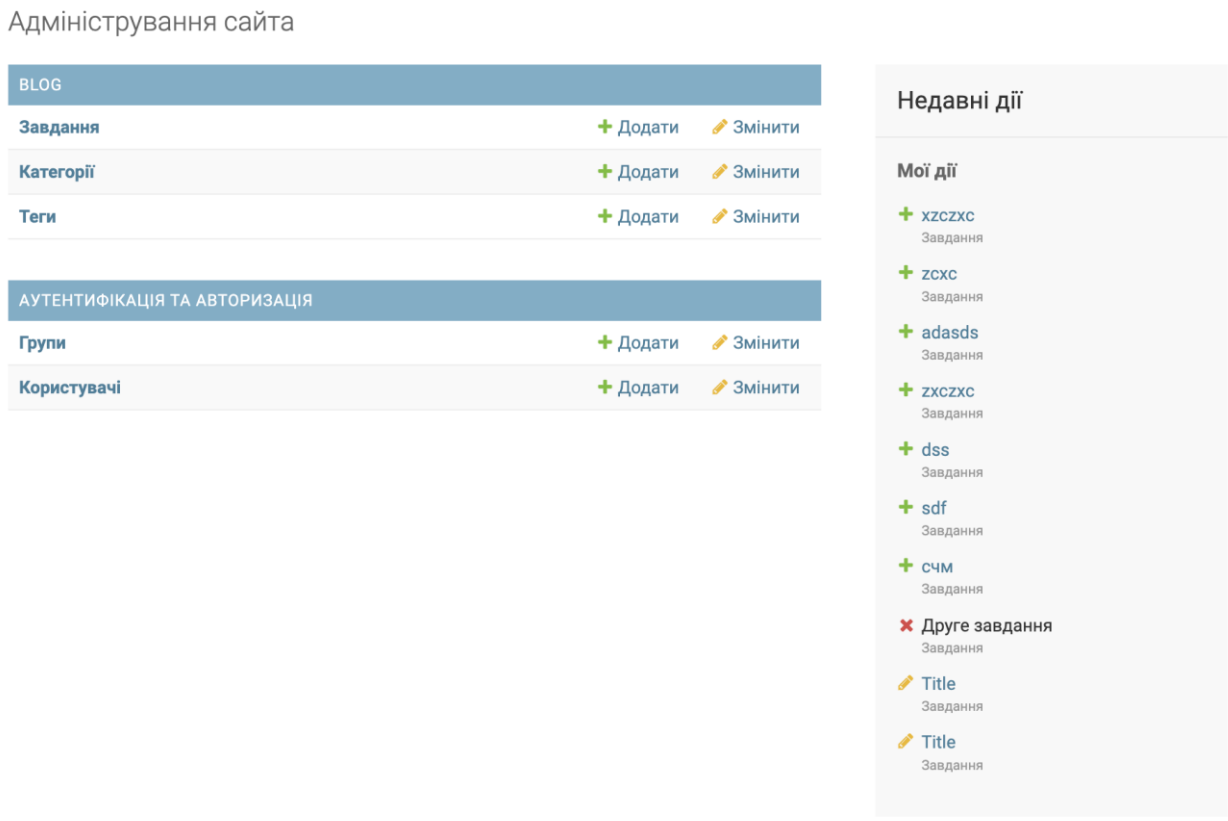

Рисунок 3.11 - Панель для адміністрування сайту

Шукаємо очима завдання, і в цьому рядку натискаємо кнопку «додати» біля неї розміщено зелений плюсик, щоб додати нове завдання. Далі ми бачимо, що відкрилася сторінка для додавання нового завдання, і нам доступні різні поля для введення. (див. рис. 3.12) Вводимо тестові дані.

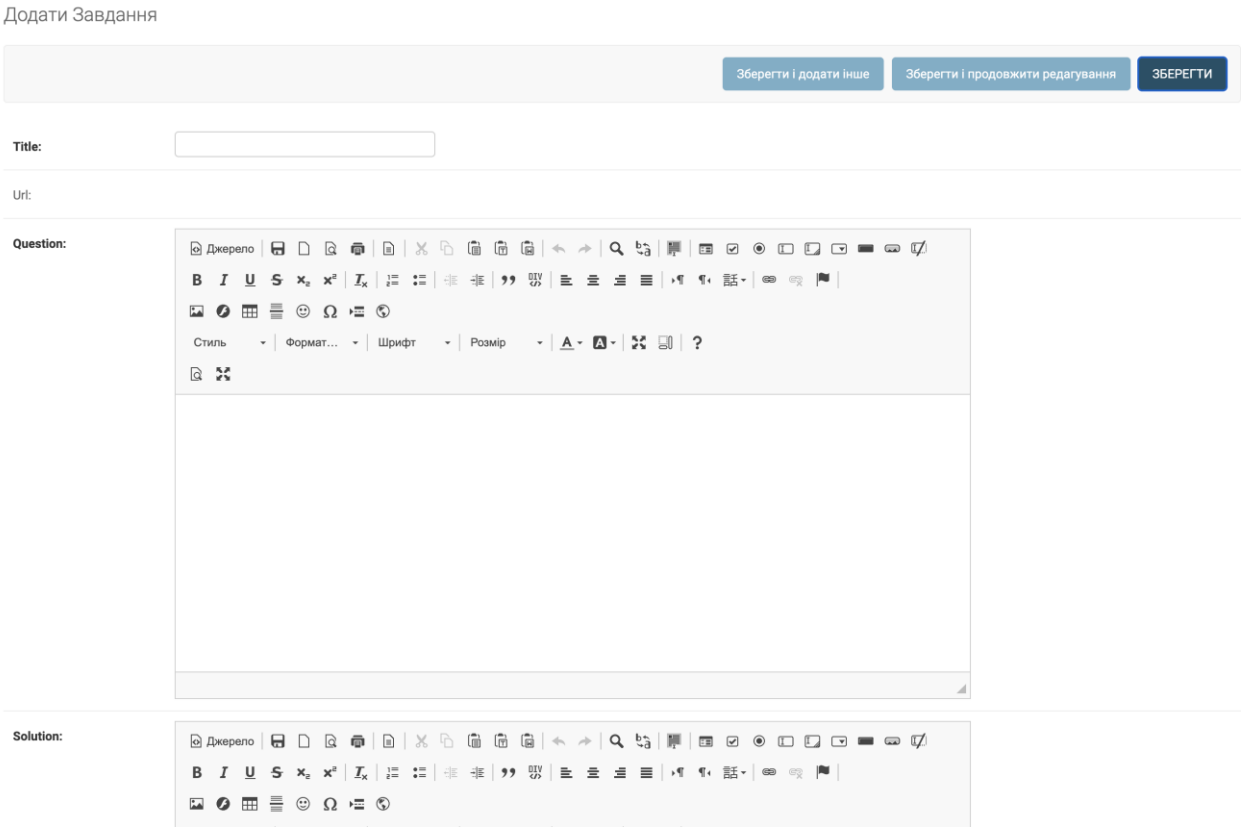

Рисунок 3.12 - Сторінка для додавання нового завдання

Далі коли дані було введено, перевіряємо правильність введення даних, шукаємо внизу сторінки кнопку із назвою «зберегти» і натискаємо її. (див. рис. 3.13)

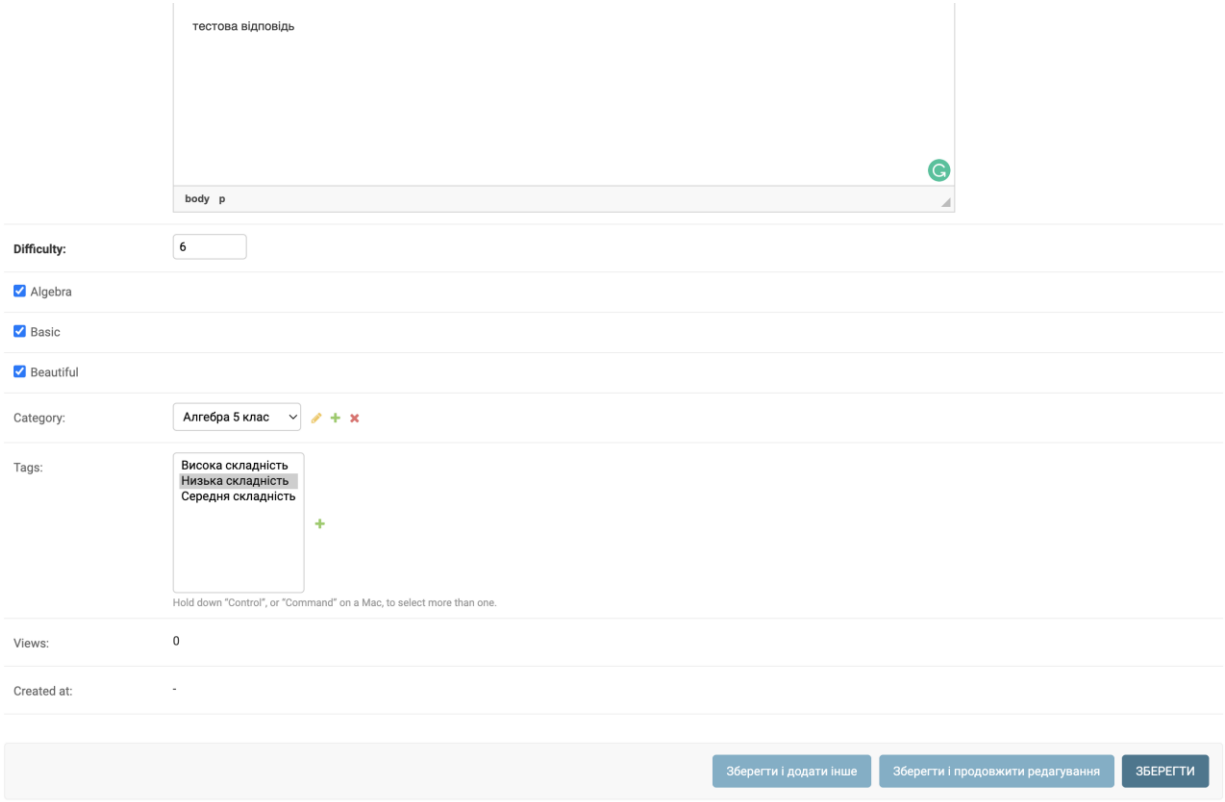

Рисунок 3.13 - Готове завдання до збереження

Якщо всі поля заповнено правильно, тобто не використано заборонених значень, то нас перекине на сторінку із списком завдання, вгорі буле написано на зеленому фоні, що завдання успішно додано, і ми можемо його бачити у списку із усіма іншими завданнями, так як список відсортований за датою створення, то нове завдання яке ми додали, заходиться на самому верху. (див. рис. 3.14)

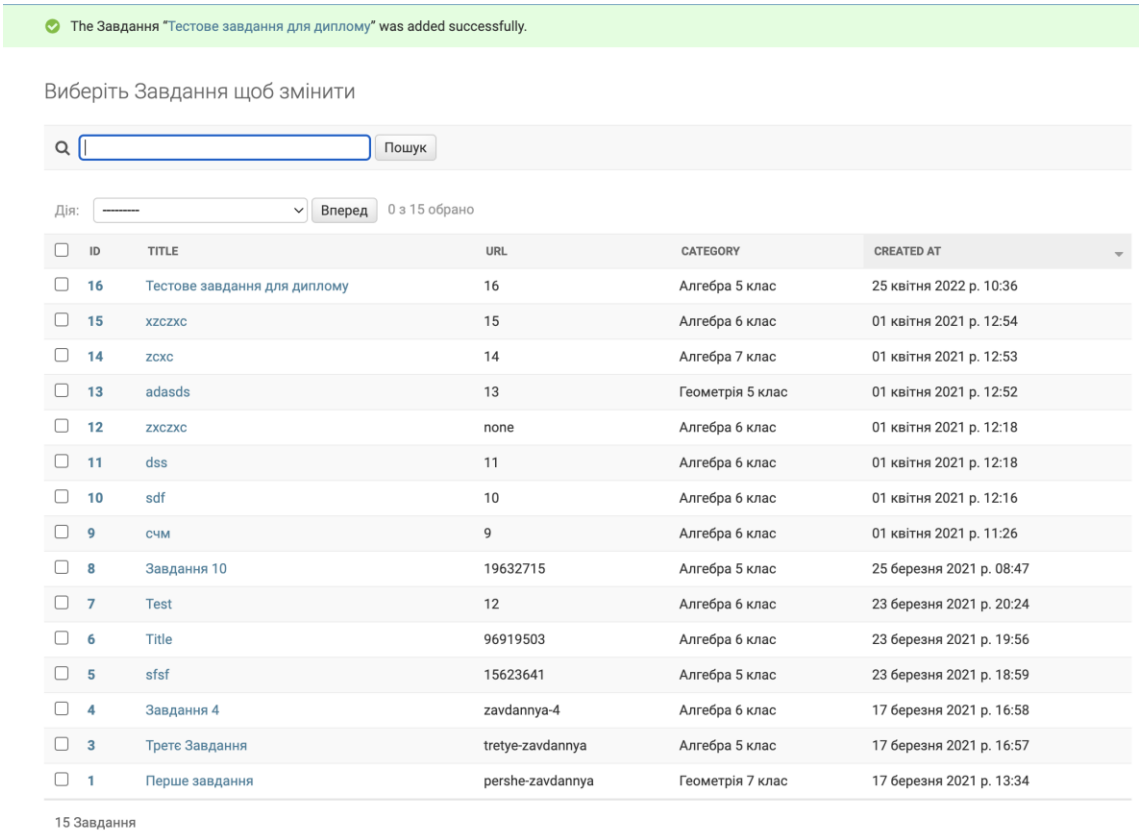

Рисунок 3.14 - Успішне додавання завдання на сайт

Якщо перейти на головну сторінку сайту, то можна побачити, що задача виконана, сайт працює як слід завдання було додане, і воно відображається на сайті. (див. рис. 3.15)

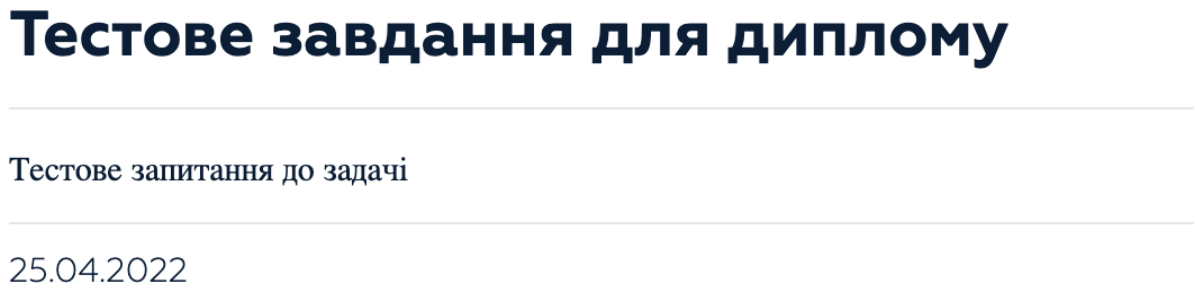

Рисунок 3.15 - Відображення щойно створеного завдання на головній сторінці сайту

ВИСНОВКИ

В даній дипломній роботи було розглянуто та створено веб платформу для проведення олімпіад з фізики для Міністерства освіти та науки України.

Платформа перспективна й в майбутньому підлягає вдосконаленню різних аспектів та функцій.

Було проведений пошук інформації та її обробка, аналіз конкурентів, обрано сучасні інструменти для реалізації веб-сайту, спроектована модель сайту, розроблено веб-сайт, і проведене тестування.

На сайті розроблено функції пагінації, пошуку, фільтрації. Було створено адміністративну панель та реалізовано можливість додавання та редагування категорій, додавання та редагування завдань, тегів для завдань, реалізовано додавання зображень до умов задач, відображення формул, функціоналу візуального редактору.

Даний сайт являється інформаційною платформою для проведення тестувань по програмі з точних наук - фізика, алгебра, геометрія. Його створено для поліпшення умов перевірки знань учнів і студентів та полегшення роботи викладачів освітніх закладів. Сайт буде максимально корисний для упередження кризового стану освіти країни в неординарних ситуаціях.

СПИСОК ЛІТЕРАТУРИ

1. Кеннеді Б., Мусciано Ч. HTML та XHTML. Детальне керівництво (HTML & HXTML. The Definitive Guide). – Sebastopol: O'Reilly Media, 2019. – 213 с.
2. Гуржій А. М., Поворознюк Н. І., Самсонов В. В. Інформатика та інформаційні технології. – Х.: Компанія СМІТ, 2003. – 372 с.
3. Девід Фленаган "JavaScript. Детальне керівництво (JavaScript. The Definitive Guide). – Sebastopol: O'Reilly Media, 2020. – 177 с.
4. Trygve M. Reenskaug H. The Model-View-Controller (MVC). Its Past and Present. – Орхус: JAOO, 2003. - 304 с.
5. Роббінс Д. HTML5, CSS3 і Javascript. Вичерпне керівництво. – Таллінн: Ексмо, 2019. – 308 с.
6. Bader D. Python Tricks: A Buffet of Awesome Python Features. – Ванкувер: Dan Bader, 2017. – 301 с.
7. Einstein M. Advertising: What Everyone Needs to Know. – Оксфорд: Oxford University Press, 2017. – 215 с.
8. Дакетт Д. HTML та CSS. Розробка та дизайн веб-сайтів. – Таллінн: Ексмо, 2020. – 302 с.
9. Enge E., Spencer S., Stricchiola J. The Art of SEO: Mastering Search Engine Optimization 3rd Edition. – Sebastopol: O'Reilly Media, 2015. – 994 с.
10. Connolly T., Begg C., Database Systems: A Practical Approach to Design, Implementation, and Management. – Лондон: Pearson, 2016. – 1440 с.
11. База Знань DJANGO [Електронний ресурс] – Режим доступа: URL: https://django.fun
12. Офіційна документація PyCharm [Електронний ресурс] – Режим доступа: URL: https://www.jetbrains.com/ru-ru/pycharm/learn/
13. McFarland D. CSS – The Missing Manual. – Sebastopol: O’Reilly Media, 2019. – 605 с.
14. Chaffey D. Digital Marketing – Лондон: Pearson, 2019. – 576 с.

15. Hart C. Conversion Rate Optimization Report. – London: Econsultancy, 2019. – 303 с.
16. William S. Django for Beginners: Build websites with Python and Django. – Лондон: WelcomeToCode 2018. – 317 с.

ДОДАТОК

Models.py

```
from django.db import models
from django.template.defaultfilters import slugify
from django.urls import reverse
from django.core.validators import MaxValueValidator,
MinValueValidator
from django.utils.crypto import get_random_string



def get_random_string_my():
    return get_random_string(8, '0123456789')



class Category(models.Model):
    title = models.CharField(max_length=255)
    slug = models.SlugField(max_length=255, verbose_name='url',
unique=True)

    def __str__(self):
        return self.title

    class Meta:
        ordering = ['title']
        verbose_name = 'Категорія(ю)'
        verbose_name_plural = 'Категорії'



class Tag(models.Model):
    title = models.CharField(max_length=50)
    slug = models.SlugField(max_length=255, verbose_name='url',
unique=True)
```

```
    def __str__(self):
        return self.title

    class Meta:
        ordering = ['title']
        verbose_name = 'Тег(и)'
        verbose_name_plural = 'Теги'


class Task(models.Model):
    title = models.CharField(max_length=255)
    slug = models.SlugField(max_length=8, verbose_name='url',
unique=True)
    question = models.TextField(default="")
    solution = models.TextField(default="")
    answer = models.TextField(default="")
    difficulty = models.PositiveSmallIntegerField(default=1,
validators=[MinValueValidator(1), MaxValueValidator(100)])
    algebra = models.BooleanField(default=True)
    basic = models.BooleanField(default=True)
    beautiful = models.BooleanField(default=True)
    created_at = models.DateTimeField(auto_now_add=True)
    views = models.IntegerField(default=0)
    category = models.ForeignKey(Category, blank=True,
on_delete=models.PROTECT, related_name='tasks')
    tags = models.ManyToManyField(Tag, blank=True,
related_name='tasks')

    def __str__(self):

        return self.title
```

```
    def save(self, *args, **kwargs):
        super(Task, self).save(*args, **kwargs)
        if not self.slug:
            self.slug = str(self.id)
            self.save()


    # def save(self):
    #     # calling super so that the instance will get created
and self.id will be accessible.
    #     super(Task, self).save()
    #     if not self.slug:
    #         slug = slugify(self.title)
    #         try:
    #             post_obj = Task.objects.get(slug=slug)
    #             slug += "-" + str(self.id)
    #         except Task.DoesNotExist:
    #             pass
    #         self.slug = slug
    #         self.save()
    # def save(self, *args, **kwargs):
    #     self.slug = slugify(self.id)
    #     super(Task, self).save(*args, **kwargs)


    def get_absolute_url(self):
        return reverse('task', kwargs={"slug": self.slug})


    # def save(self, **kwargs):
    #     slug_str = get_random_string(8, '0123456789')
    #     unique_slugify(self, slug_str)
    #     super(Task, self).save(**kwargs)


    # def save(self, *args, **kwargs):
```

```
    #     if not self.id:
    #         self.slug = slugify(self.name)
    #
    #     super(Gadget, self).save(*args, **kwargs)

    class Meta:
        ordering = ['-created_at']
        verbose_name = 'Завдання'
        verbose_name_plural = 'Завдання'
```

Settings.py

```
"""
Django settings for siteblog project.

Generated by 'django-admin startproject' using Django 3.1.7.

For more information on this file, see
https://docs.djangoproject.com/en/3.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.1/ref/settings/
"""

from pathlib import Path
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
```

```
# See
https://docs.djangoproject.com/en/3.1/howto/deployment/checklist
/


# SECURITY WARNING: keep the secret key used in production
secret!
SECRET_KEY =
'hx0u9+o^je0vhghchlc1iomyqt%h18(9(k_gv!hbw67u(@yd14#'


# SECURITY WARNING: don't run with debug turned on in
production!
DEBUG = True


ALLOWED_HOSTS = []



# Application definition


INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'blog.apps.BlogConfig',
    'debug_toolbar',
    'ckeditor',
    'django_filters',
    'bootstrapform',
]


MIDDLEWARE = [
```

```
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'debug_toolbar.middleware.DebugToolbarMiddleware',
]

ROOT_URLCONF = 'siteblog.urls'

TEMPLATES = [
    {
        'BACKEND':
'django.template.backends.django.DjangoTemplates',
        'DIRS': [
            os.path.join(BASE_DIR, 'templates'),
        ],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',

'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

```
WSGI_APPLICATION = 'siteblog.wsgi.application'



# Database

# https://docs.djangoproject.com/en/3.1/ref/settings/#databases


DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': BASE_DIR / 'db.sqlite3',

    }

}



# Password validation

# https://docs.djangoproject.com/en/3.1/ref/settings/#auth-
password-validators


AUTH_PASSWORD_VALIDATORS = [

    {

        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarity
Validator',

    },

    {

        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator'
,

    },

    {

        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator
',

    },
```

```
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidato
r',
    },
]



# Internationalization
# https://docs.djangoproject.com/en/3.1/topics/i18n/

LANGUAGE_CODE = 'uk'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True



# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.1/howto/static-files/

STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'siteblog/static'),
]

MEDIA_URL = '/media/'
```

```
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')


INTERNAL_IPS = [
    '127.0.0.1',
]


CKEDITOR_UPLOAD_PATH = "uploads/"


CKEDITOR_CONFIGS = {
    'default': {
        'skin': 'moono-lisa',
        # 'skin': 'office2013',
        'toolbar_Basic': [
            ['Source', '-', 'Bold', 'Italic']
        ],
        'toolbar_YourCustomToolbarConfig': [
            {'name': 'document', 'items': ['Source', '-',
'Save', 'NewPage', 'Preview', 'Print', '-', 'Templates']},
            {'name': 'clipboard', 'items': ['Cut', 'Copy',
'Paste', 'PasteText', 'PasteFromWord', '-', 'Undo', 'Redo']},
            {'name': 'editing', 'items': ['Find', 'Replace', '-
', 'SelectAll']},
            {'name': 'forms',
             'items': ['Form', 'Checkbox', 'Radio', 'TextField',
'Textarea', 'Select', 'Button', 'ImageButton',
                       'HiddenField']},
            '/',
            {'name': 'basicstyles',
             'items': ['Bold', 'Italic', 'Underline', 'Strike',
'Subscript', 'Superscript', '-', 'RemoveFormat']},
            {'name': 'paragraph',
             'items': ['NumberedList', 'BulletedList', '-',
'Outdent', 'Indent', '-', 'Blockquote', 'CreateDiv', '-',
```

```
                    'JustifyLeft', 'JustifyCenter',
'JustifyRight', 'JustifyBlock', '-', 'BidiLtr', 'BidiRtl',
                    'Language']},
            {'name': 'links', 'items': ['Link', 'Unlink',
'Anchor']},
            {'name': 'insert',
             'items': ['Image', 'Flash', 'Table',
'HorizontalRule', 'Smiley', 'SpecialChar', 'PageBreak',
'Iframe']},
            '/',
            {'name': 'styles', 'items': ['Styles', 'Format',
'Font', 'FontSize']},
            {'name': 'colors', 'items': ['TextColor',
'BGColor']},
            {'name': 'tools', 'items': ['Maximize',
'ShowBlocks']},
            {'name': 'about', 'items': ['About']},
            '/',  # put this to force next toolbar on new line
            {'name': 'yourcustomtools', 'items': [
                # put the name of your editor.ui.addButton here
                'Preview',
                'Maximize',

            ]},
        ],
        'toolbar': 'YourCustomToolbarConfig',  # put selected
toolbar config here
        # 'toolbarGroups': [{ 'name': 'document', 'groups': [
'mode', 'document', 'doctools' ] }],
        # 'height': 291,
        # 'width': '100%',
        # 'filebrowserWindowHeight': 725,
        # 'filebrowserWindowWidth': 940,
        # 'toolbarCanCollapse': True,
```

```
        # 'mathJaxLib': '//cdn.mathjax.org/mathjax/2.2-
latest/MathJax.js?config=TeX-AMS_HTML',
        'tabSpaces': 4,
        'extraPlugins': ','.join([
            'uploadimage', # the upload image feature
            # your extra plugins here
            'div',
            'autolink',
            'autoembed',
            'embedsemantic',
            'autogrow',
            'mathjax',
            # 'devtools',
            'widget',
            'lineutils',
            'clipboard',
            'dialog',
            'dialogui',
            'elementspath'
        ]),
    }
}
```

Views.py

```
from django.shortcuts import render
from django.views.generic import ListView, DetailView
from .models import Task, Category, Tag
from django.db.models import F
from .filters import TaskFilter



# class Home(ListView):
```

```
#     model = Task
#     template_name = 'blog/index.html'
#     context_object_name = 'tasks'
#     paginate_by = 3
#
#     def get_context_data(self, *, object_list=None, **kwargs):
#         context = super().get_context_data(**kwargs)
#         context['title'] = 'Title'
#         return context

class Home(ListView):
    model = Task
    template_name = 'blog/index.html'
    context_object_name = 'tasks'
    paginate_by = 3

    def get_context_data(self, **kwargs):
        # myFilter = TaskFilter()
        # context = {'myFilter': myFilter}
        context = super().get_context_data(**kwargs)
        context['filter'] = TaskFilter(self.request.GET,
queryset=self.get_queryset())
        # context['s'] = f"s={self.request.GET.get('s')}&"
        return context


def index(request):
    return render(request, 'blog/index.html')


# def get_task(request, slug):
#     return render(request, 'blog/single.html')
```

```
class GetTask(DetailView):
    model = Task
    template_name = 'blog/single.html'
    context_object_name = 'task'
    task_id = Task.id

    def get_context_data(self, *, object_list=None, **kwargs):
        context = super().get_context_data(**kwargs)
        context['title'] = 'Task page'
        self.object.views = F('views') + 1
        self.object.save()
        self.object.refresh_from_db()
        return context


class Search(ListView):
    template_name = 'blog/search.html'
    context_object_name = 'tasks'
    paginate_by = 1

    def get_queryset(self):
        return
Task.objects.filter(question__icontains=self.request.GET.get('s'
))

    def get_context_data(self, *, object_list=None, **kwargs):
        context = super().get_context_data(**kwargs)
        context['s'] = f"s={self.request.GET.get('s')}&"
        return context
```

Admin.py

```
from django.contrib import admin
from django import forms
from .models import *
from ckeditor_uploader.widgets import CKEditorUploadingWidget


class TaskAdminForm(forms.ModelForm):
    question = forms.CharField(widget=CKEditorUploadingWidget())
    answer = forms.CharField(widget=CKEditorUploadingWidget())
    solution = forms.CharField(widget=CKEditorUploadingWidget())

    class Meta:
        model = Task
        fields = '__all__'


class TaskAdmin(admin.ModelAdmin):
    # prepopulated_fields = {"slug": ('id',s))}
    # save_as = True
    form = TaskAdminForm
    list_display = ('id', 'title', 'slug', 'category',
'created_at')
    list_display_links = ('id', 'title')
    search_fields = ('title',)
    list_filter = ('category', 'tags')
    save_on_top = True
    readonly_fields = ('views', 'created_at', 'slug',)
    fields = ('title', 'slug', 'question', 'solution', 'answer',
'difficulty', 'algebra', 'basic', 'beautiful',
              'category', 'tags', 'views', 'created_at')
```

```
class CategoryAdmin(admin.ModelAdmin):
    prepopulated_fields = {"slug": ("title",)}


class TagAdmin(admin.ModelAdmin):
    prepopulated_fields = {"slug": ("title",)}


admin.site.register(Category, CategoryAdmin)
admin.site.register(Tag, TagAdmin)
admin.site.register(Task, TaskAdmin)
```