Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Андрющенко Кристини Сергіївни*
(ПІБ)

академічної групи *121-19-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка програмного забезпечення для дистанційної освіти на основі мов Python/JavaScript та фреймворку Django*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Ширін А.Л.*			
розділів:				
спеціальний	*доц. Ширін А.Л.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент	*доц. Шедловський І.А.*			
Нормоконтролер	*ст.викл. Мартиненко А.А.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » _____ 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-19-1* *Андрющенко Кристини Сергіївни*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка програмного забезпечення для дистанційної освіти на основі мов Python/JavaScript та фреймворку Django*

затверджена наказом ректора НТУ «ДП» від *16.05.2023* № *350-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ *доц. Ширін А.Л.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Андрющенко К.С.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *12.06.2023 р.*

РЕФЕРАТ

Пояснювальна записка: 87 с., 48 рис., 8 табл., 3 дод., 25 джерел.

Об'єкт розробки: веборієнтоване програмне забезпечення для дистанційної освіти.

Метою кваліфікаційної роботи є розробка програмного забезпечення для дистанційної освіти на основі мов Python/JavaScript та фреймворку Django.

У вступі розглядається актуальність вирішення проблеми, опис галузі, де може використовуватись розроблене програмне забезпечення та мета кваліфікаційної роботи.

У першому розділі роботи проводиться аналіз предметної області, у якій проводитиметься дослідження та розробка. Також проведено огляд наявних робіт, теоретичних засад і практичних аспектів, пов'язаних із темою дослідження.

У другому розділі роботи проводиться аналіз наявних рішень, пов'язаних із розглянутою проблемою та темою дослідження. Досліджуються різні підходи, методи та інструменти, які були застосовані раніше для вирішення аналогічних завдань.

В економічному розділі роботи проводиться оцінка трудовитрат на розробку продукту. Визначається кількість місяців, яка потрібна для виконання кожного етапу проекту. Також проводиться аналіз ресурсів, необхідних для розробки, таких як людські ресурси, апаратне й програмне забезпечення та інші витрати.

Практичне значення полягає у створенні програмного забезпечення для дистанційної освіти за допомогою якого студенти можуть переглядати лекції, завдання до практичних робіт, підписавшись на курс викладача та здавати роботи. Також є швидкий доступ до зв’язку через внутрішній чат або пошту. Викладачі мають можливість швидко отримувати всі оцінки з курсу, та вивантажувати їх у файл формату MS Excel.

Актуальність даного програмного продукту визначається великим попитом на подібні розроблені програми, що дає змогу без перешкод здобувати дистанційну освіту.

Список ключових слів: ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, РОЗРОБКА, ДИСТАНЦІЙНА ОСВІТА, PYTHON, ПЛАТФОРМА, ВЕБДОДАТОК.

ABSTRACT

Explanatory note: 87 p., 48 figures, 8 tables, 3 appendixes, 25 sources.

Object of development: software for distance education.

The purpose of the qualification work is to develop software for distance education based on Python/JavaScript and Django framework.

The introduction discusses the purpose of the qualification work, the relevance of solving the problem, and a description of the industry where the developed software application can be used.

The first chapter of the work analyses the subject area in which the research or development will be carried out. It also reviewed existing works, theoretical foundations and practical aspects related to the research topic.

The second section of the paper analysed existing solutions related to the problem under consideration and the research topic. Various approaches, methods and tools that have been used in the past to solve similar problems were studied.

The economic section of the paper estimates the labour costs of product development. It determines the number of months required to complete each stage of the project. It also analyses the resources required for development, such as human resources, hardware and software, and other costs.

The practical significance lies in the creation of software for distance education, with the help of which students can view lectures, assignments for practical work, subscribe to the teacher's course, and submit work. There is also quick access to communication via internal chat or email. Teachers can quickly get all the grades from the course by uploading them to an Excel file.

The relevance of this software product is determined by the high demand for such developed programmes, which makes it possible to get an education without any obstacles.

List of keywords: SOFTWARE, DEVELOPMENT, DISTANCE EDUCATION, PYTHON, PLATFORM, WEB APPLICATION.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ДО - дистанційна освіта

ПЗ - програмне забезпечення

БД – база даних

JS - JavaScript

HTML - HyperText Markup Language

CSS - Cascading Style Sheets

SSL - Secure Sockets Layer

CSRF - Cross-Site Request Forgery

XSS - Cross-Site Scripting

HDD - hard disk drive

SSD - Solid-State Drive

MVC - Model-View-Controller

FBV – Function Based Views

ЗМІСТ

ВСТУП

Дистанційне навчання, завжди було популярне, але в часи карантину через COVID воно стало стрімко набирати популярність не тільки серед різних приватних шкіл певної сфери діяльності, а й серед державних навчальних закладів.

Дистанційна освіта (ДО) сприяє підвищенню доступності освіти. Україна – велика країна з різноманітними регіонами, і не всім студентам зручно чи можливо фізично відвідувати традиційні навчальні заклади. ДО дозволяє подолати географічні та соціальні обмеження, надаючи можливість навчання людям із різних куточків країни.

Одна з головних переваг дистанційної освіти - гнучкість. Студенти можуть навчатися у зручний для них час та місце, не прив'язуючись до фізичної присутності в аудиторії. Вони можуть самі обирати темп та порядок вивчення матеріалів, а також взаємодіяти з навчальними ресурсами та завданнями відповідно до власного графіка [1].

Зараз є велика кількість різних платформ для дистанційного навчання, наприклад Moodle, Coursera, Udemy та інші.

На платформі «EASY STUDY» студенти із легкістю можуть підписуватись на курси та спілкуватися з викладачами щодо кожної з робіт та матеріалів.

Додатково, програмне забезпечення «EASY STUDY» дозволяє викладачу легко вивантажити всі оцінки на курсі в Excel-форматі, що дозволяє працювати й мати резервну копію незалежно від того є підключення до Інтернету чи ні.

Переваги платформ дистанційної освіти є невід'ємною частиною сучасного процесу творення. Вони відкривають нові горизонти для студентів та викладачів, забезпечуючи гнучкість, доступність та ефективність навчання.

РОЗДІЛ І
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Дистанційне навчання, або онлайн-навчання, відноситься до способу навчання, при якому учні та викладачі можуть знаходитися в різних точках земної кулі та взаємодіють в основному через онлайн-платформи та цифрові ресурси. Це дозволяє людям здобувати освіту, без присутності фізично.

Дистанційне навчання стає все більш популярним і доступним завдяки великому просуванню в галузі технологій та Інтернету [1].

Воно пропонує ряд переваг, таких як:

Гнучкість. Студенти можуть отримати доступ до навчальних матеріалів. Вони можуть навчатися з будь-якого місця, де є підключення до Інтернету, що забезпечує більшу гнучкість в управлінні роботою, сім'єю та іншими зобов'язаннями.

Доступність. Дистанційне навчання робить освіту більш доступною для осіб, які можуть мати географічні, фізичні або інші обмеження, що не дозволяють відвідувати традиційні очні заняття [2].

Персоналізоване навчання. Студенти можуть просуватися у своєму власному темпі, переглядати матеріали в міру потреби та отримувати доступ до додаткових ресурсів залежно від їх індивідуальних потреб та стилів навчання.

Широкий спектр курсів та програм. Дистанційне навчання пропонує широкий спектр курсів та програм на здобуття ступеня з різних дисциплін. Студенти можуть вибирати курс з ширшого спектру варіантів освіти, включаючи спеціалізовані області або нішові предмети, які можуть бути недоступними на місцевому рівні.

Співробітництво та взаємодія. Платформи онлайн навчання зазвичай надають інструменти для спільної роботи та взаємодії, такі як дошки обговорень, відеоконференції та віртуальні класи. Студенти можуть взаємодіяти з викладачами та дізнаватися більше інформації про предмет [3].

Тим не менш, важливо визнати деякі проблеми, пов'язані з дистанційним навчанням, такі як:

Обмежена особиста взаємодія. На відміну від очних класів, дистанційне навчання часто зменшує рівень зворотного зв'язку, співробітництва та групових дискусій, що може бути важливим аспектом навчання.

Технічні проблеми. Для участі у дистанційному навчанні необхідний доступ до інтернет-з'єднання та комп'ютера або смартфона. Технічні збої або недостатні навички роботи з технологіями можуть стати перешкодою для ефективної участі у процесі навчання.

Відсутність суворої структури та спостереження. У дистанційному навчанні студентам часто не вистачає зовнішньої мотивації та дисципліни, оскільки їм необхідно самостійно встановлювати графік занять та справлятися з навчальними завданнями. Деяким людям може бути складно організуватись та підтримувати постійну роботу без присутності викладачів чи тих, хто навчається.

Обмежений доступ до ресурсів. У деяких випадках дистанційне навчання може обмежувати доступ до певних ресурсів, таких як лабораторії, бібліотеки або спеціалізоване обладнання. Це може обмежити практичний досвід та застосування теоретичних знань [2.]

Платформи для дистанційного навчання можна розділити на два види залежно від їхньої форми реалізації: програмне забезпечення (ПЗ) та вебдодатки.

Програмне забезпечення – це спеціалізовані програми, які встановлюються на комп'ютер або сервер. Ці програми можуть бути розроблені індивідуально для використання всередині навчальних закладів або як окремий продукт для дистанційного навчання. Приклади таких програм: Blackboard, Moodle, Canvas, Adobe Connect, Microsoft Teams.

Вебдодатки – це платформи, доступні через веббраузер, які не потребують встановлення додаткового програмного забезпечення на комп'ютер. Вебзастосунки зазвичай мають багатофункціональні можливості і можуть бути доступні на різних пристроях. Приклади таких веб-застосунків: Google Classroom, Edmodo, Schoology, Coursera, Udemy.

Обидва види платформ мають свої переваги та недоліки. Вибір між ПЗ та вебдодатком залежить від потреб та вимог освітніх закладів, викладачів та студентів.

1.2. Призначення розробки та галузь застосування

В якості програмного забезпечення у роботі розглянуто вебдодаток, в якому викладачі можуть додавати свої курси та матеріали до цих курсів, студенти можуть підписатися на ці курси та виконувати призначені їм завдання.

Крім цього, викладачі можуть оцінювати зроблені роботи в таблиці оцінок, які закріплюються. На кожному курсі студенти, якщо у них виникають запитання, мають можливість розмовляти з викладачем за допомогою чату на сторінці матеріалу курсу.

Розроблений вебдодаток призначений для:

−зручного виставлення оцінок викладачами;

−зручного перегляду оцінок студентами;

−експорту оцінок викладачами у файл розширення Excel, що дає швидку можливість збирати успішність усіх студентів з усіх курсів викладача.

1.3. Підстава для розробки

Підставами для розробки та виконання кваліфікаційної роботи є:

− освітня програма 121 Інженерія програмного забезпечення;

− навчальний план та графік навчального процесу;

− наказ ректора Національного технічного університету «Дніпровська політехніка» № 350-с від 16.05.2023 р;

− завдання на кваліфікаційну роботу на тему «Розробка програмного забезпечення для дистанційної освіти на основі мов Python/JavaScript та фреймворку Django».

1.4. Постановка завдання

Завданням роботи є розробка програмного забезпечення для дистанційної освіти на основі мов Python/JavaScript та фреймворку Django. Функціонал ПЗ, який має бути реалізований при користуванні викладачем:

−реєстрація користувача;

−зміна своїх даних у профілі;

−додавання курсу;

−зміна курсу;

−додавання матеріалу;

−зміна матеріалу;

−перевірка робіт студентів;

−виставлення оцінок;

−перегляд всіх студентів на курсі;

−видалення студентів з курсу;

−чат зі студентом під матеріалом курсу;

−завантаження всіх оцінок у файл Excel.

При користуванні студентом:

−реєстрація користувача;

−зміна своїх даних у профілі;

−завантаження робіт під матеріалом курсу;

−чат з викладачем;

−перегляд оцінок з курсів;

−можливість завантаження матеріалів курсу.

Можливості адміністратора:

−перегляд та зміна усієї інформації про користувачів;

−надання особливих прав користувачам;

−перегляд та зміна інформації про курс;

−перегляд усіх файлів.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Програмне забезпечення має бути написано мовою програмування Python та відкриватися за допомогою веббраузеру, наприклад, Chrome, Microsoft Edge, Opera та інші.

Для успішного функціонування поставленого завдання потрібно за допомогою фреймворку Django, мови JS, HTML та CSS реалізувати наступне:

−система повинна мати веб-сервер, який прийматиме запити та обробляти їх;

−мати адаптивний дизайн під кожен вид пристроїв, щоб давати можливість заходити на сайт із різних пристроїв;

−збереження даних користувача в реляційній базі даних SQLite.

1.5.2 Вимоги до інформаційної безпеки

Важливо забезпечити інформаційну безпеку для будь-якої системи, особливо для даної, оскільки вона пов'язана з цілим навчальним закладом, де зберігається багато даних. Для цього була реалізована можливість реєструватися лише з корпоративної пошти університету.

Також для безпеки було використано:

− шифрування даних за допомогою SSL. При передачі даних між клієнтом і сервером встановлюється зашифроване з'єднання, і ніхто не зможе перехопити і прочитати інформацію, що передається;

− захист від CSRF та XSS атак.

Також користувач, який не зайшов у систему може заходити на перелік курсів, але не може бачити матеріали та всю іншу інформацію про курси. Це було зроблено для захисту авторських прав викладачів та студентів.

1.5.3 Вимоги до складу та параметрів технічних засобів

Для правильного функціонування ПЗ необхідно мати комп’ютер або ноутбук, який буде відповідати наступним характеристикам:

- процесор Intel Core i3 або вище;
- частота процесора не менше 3.6 ГГц;
- підтримка 32- або 64-розрядного процесора та операційної системи;
- вільного місця на HDD чи SSD 10 гб або більше;
- оперативна пам'ять об'ємом 4 ГБ або вище;
- мережеве підключення;
- клавіатура та мишка.

1.5.4 Вимоги до інформаційної та програмної сумісності

Для запуску та функціонування вебдодатку, програмне забезпечення, яке встановлене на комп’ютері або ноутбуці, повинно мати наступні характеристики:

- операційна система Windows 7 або вище/MacOS 10.13 або вище /Linux;
- веббраузер Chrome, Microsoft Edge, Opera, Safari.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Платформа для дистанційного навчання «EASY STUDY» розроблено для студентів і викладачів з метою забезпечити зручне навчання та перевірку засвоєних знань. Викладачі мають змогу створювати курси та розміщувати на них різноманітні матеріали, тоді як студенти можуть виконувати завдання та подавати їх для оцінювання впродовж усього курсу.

Програма має наступне призначення:

- забезпечення здобуття освіти через Інтернет;
- легке використання викладачем та студентом;
- швидка комунікація між викладачем та студентом;
- зручне експортування викладачем всіх оцінок на курсі.

Також програма має наступні функціональні можливості:

- гість може увійти в систему/зареєструватися
- викладач може: створити/змінити курс, виставити/змінити матеріал, переглядати свої/всі курси, змінювати особисту інформацію в профілі, спілкуватися в чаті
- студент може: переглядати курси, змінювати особисту інформацію, здавати роботу під практичним завданням, спілкуватися з викладачем у чаті.

2.2. Опис застосованих математичних методів

При написанні даного веб-орієнтованого програмного додатку не використовувалися математичні методи.

2.3. Опис використаної архітектури та шаблонів проектування

MVC (Model-View-Controller) є архітектурним патерном розробки, який дозволяє подолати недоліки традиційного підходу, де весь код знаходиться в одному файлі. Архітектура MVC базується на розділенні коду на окремі компоненти, які відповідають за різні аспекти веб-програми або веб-сайту [5].

Шаблон MVC складається з трьох основних компонентів: модель (Model), представлення (View) та контролер (Controller) (рис. 2.1.).

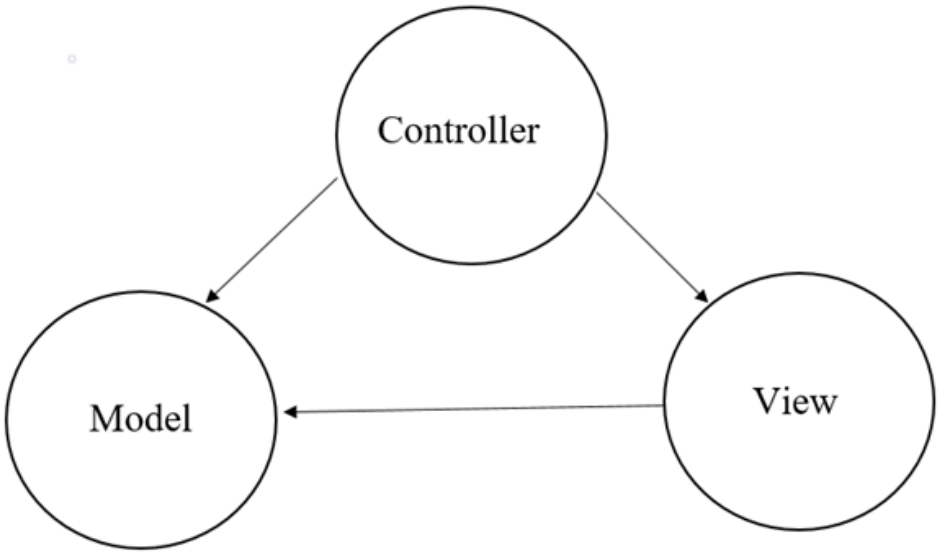


Рис. 2.1. Шаблон MVC

Такий поділ дозволяє розробнику фокусуватися на конкретному аспекті розробки веб-додатку та створювати код вищої якості, який краще організований, проходить ефективніше тестування, налагодження та забезпечує можливість масштабування [6].

Модель

Модель є компонентом архітектури Django, який містить у собі бізнес-логіку програми.

Модель є невід'ємною частиною вебдодатку, яка виконує важливу роль провідника між базою даних та інтерфейсом вебсайту. З технічної точки зору, модель є об'єктом, який реалізує логіку предметної області додатку. Існують

випадки, коли програма здатна обробляти дані безпосередньо з певного набору даних і передавати їх безпосередньо в подання (компонент інтерфейсу користувача), без необхідності використання бази даних. У цьому випадку цей набір даних можна розглядати як модель [5].

Однак у сучасній веброзробці, при створенні практично будь-якого вебсайту, потрібна наявність бази даних, оскільки ми зазвичай потребуємо можливості отримувати введення користувача.

Представлення

У Django, представлення (view) є функцією або класом Python, яка обробляє веб-запити і повертає веб-відповіді. Ця відповідь може бути представлена у вигляді HTML-вмісту веб-сторінки, перенаправлення, помилки 404, XML-документа, зображення або будь-якого іншого вмісту, який може бути відображено веб-браузером. Уявлення визначають логіку обробки запитів та взаємодії з моделями та шаблонами [5].

Функціональні уявлення (Function-Based Views) є найпростішим способом визначення уявлень у Django. Вони є звичайними функціями Python, які приймають об'єкт запиту (request) як аргумент і повертають об'єкт відповіді (response) [9].

Контролер

В рамках архітектури MVC, контролер виконує важливу роль в обробці та управлінні потоком даних між поданням та моделлю, а також в оновленні даних моделі. З використанням програмної логіки, контролер визначає, які дані мають бути вилучені з бази даних через модель та передані до подання, а також отримує інформацію від користувача через подання. Контролер реалізує задану логіку шляхом зміни уявлення чи оновлення даних через модель. У простих термінах, контролер можна розглядати як посередника між введенням користувача і вебдодатком. Він приймає введення користувача, визначає, яка логіка повинна бути виконана, і взаємодіє з моделлю і поданням для обробки запитів і оновлення даних [5].

Мета контролера в архітектурі MVC полягає в управлінні логікою програми, забезпеченні її керованості та поділу, що сприяє спрощенню супроводу та розширення коду. Він також допомагає уникнути змішування бізнес-логіки з кодом подання та моделі, що сприяє більш організованому та зрозумілому коду, а також його легкій підтримці.

У проекті також були використані шаблони Django. Головна мета шаблонів в Django полягає в розділенні логіки програми та представлення даних. Шаблони надають зручний спосіб визначення структури та зовнішнього вигляду вебсторінки, а також вставки змінних, умов та циклів для динамічного відображення даних.

Django надає шаблони з можливістю використання спеціальних тегів та фільтрів, які значно розширюють їхню функціональність. Теги дозволяють виконувати умовні операції, цикли, включати інші шаблони та багато іншого. Фільтри, у свою чергу, забезпечують можливість форматування та обробки даних перед їх відображенням [8].

Для ефективного використання шаблонів у Django потрібно визначити відповідні файли шаблонів і вказати їх шлях в уявленнях. Це дозволяє застосовувати шаблони для різних завдань, таких як відображення веб-сторінок, надсилання електронної пошти, створення PDF-документів та інші методи виведення даних. Шляхом поділу логіки програми та уявлення від даних, шаблони забезпечують гнучкість та перевикористовуваність коду. При використанні шаблонів у Django розробники можуть ефективно організовувати відображення даних та керувати їх форматуванням та обробкою.

Ось приклад Template Rendering змінних у програмному додатку, що розробляється (рис. 2.2. – 2.7.):

```
<span class="course_author">
    <span style="font-weight: bold;">
        {% if course.author.first_name and course.author.last_name %}
            {{course.author.first_name}} {{course.author.last_name}}
        {% elif course.author.first_name %}
            {{course.author.first_name}}
        {% elif course.author.last_name %}
            {{course.author.last_name}}
        {% else %}
            {{course.author.get_first_part_email}}
        {% endif %}
    </span>
     - {{course.get_rubric_display}}</span>
```

Рис. 2.2. Виведення значення з контексту (context) - словник ключів зі значень.

```
<form action="{% url 'app:add_course' %}" method="post">
    {% csrf_token %}
    <h1>{% translate 'Добавление курса' %}</h1>
    {{form.name}}
    {{form.description}}
    {{form.rubric}}
    <input type="submit" value="{% translate 'Добавить' %}" class="submit-button">
</form>
```

Рис. 2.3. Виведення полів об'єкта форми

```
{% if request.user.is_authenticated %}
    {% if course.author == request.user %}
        <div style="display: flex; justify-content: end;">
            <a class="course_add_material" href="{% url 'app:edit_course' course.slug %}">{% translate 'Изменить курс' %}</a>
            <a class="course_add_material" href="{% url 'app:add_material' %}?from={{course.id}}">{% translate 'Добавить материал' %} +</a>
            <a class="course_add_material" href="{% url 'app:students' course.slug %}">{% translate 'Студенты на курсе' %} {{course.students.count}}</a>
        </div>
    {% endif %}
{% endif %}
```

Рис. 2.4. Створення умовних блоків (if else)

```
{% for course in request.user.get_courses %}
<div class="course" style="width: 100%;">
    <a href="{% url 'app:course' course.slug %}" class="course_name">{{course.name}}</a>
    <span class="course_description">{{course.description}}</span>
    <span class="course_author"><span style="font-weight: bold;">{{course.get_rubric_display}}</span>
</div>
{% empty %}
    <h3 class="center">{% translate 'Пока ваших курсов нет.' %} <a href="{% url 'app:add_course' %}">{% translate 'Добавьте один.' %}</a></h3>
{% endfor %}
```

Рис. 2.5. Створення циклів виведення об'єктів/значень (for)

```
<div class="course_desc"><strong>{% translate 'Последняя дата сдачи' %}</strong>: {{material.last_due_date|date:"j E Y"}}</div>
```

Рис. 2.6. Використання django-filters (у даному випадку виведення дати в потрібному форматі)

```
<p>{% translate "Имя" %}:</p>
<p>{% translate "Фамилия" %}:</p>
<p>{% translate "Отчество" %}:</p>
<p>{% translate "Номер телефона" %}:</p>
```

Рис. 2.7. Можливість перекладу кількома мовами

2.4. Опис використаних технологій та мов програмування

Веб-орієнтований додаток було реалізовано на мовах JavaScript та Python.

JavaScript (JS) – це мова програмування, яка застосовується для розробки динамічних та інтерактивних веб-сайтів та програм. Його унікальність полягає у можливості працювати безпосередньо у браузері, а не лише на сервері [12].

Разом з гіпертекстовою мовою розмітки (HTML) та каскадними таблицями стилів (CSS), JavaScript є однією з найпоширеніших мов програмування в Інтернеті.

Розробники широко застосовують JavaScript для додавання анімації, спливаючих вікон, панелей пошуку, кнопок, аудіо та відео, віджетів чату та інших інтерактивних елементів на веб-сторінку. Ця мова також дозволяє оновлювати вміст сторінки в режимі реального часу без перезавантаження.

Приклади веб-сайтів, які використовують JavaScript для додавання динамічних елементів, включають сайти з відображенням актуальних курсів акцій, рядком новин, що біжить, а також наданням інформації про наявність товарів у режимі реального часу. Зовнішній інтерфейс (фронтенд) майже повністю ґрунтується на JavaScript багатьох популярних вебсайтів, таких як Google, Twitter, Facebook та Wikipedia [13].

Нижче наведено кілька основних варіантів використання JavaScript:

– створення веб-серверів та серверних програм;

– додавання динаміки веб-сторінкам;

– розробка ігор.

– JavaScript має низку переваг. Нижче наведено деякі з цих переваг:

−оновлення - команда розробників JavaScript та ECMA International регулярно випускає оновлення, щоб гарантувати їхню актуальність в індустрії;

−масштабування - JavaScript дозволяє розробляти як маленькі та прості веб-додатки, так і складні проекти великого масштабу;

−універсальність - JavaScript сумісний з іншими мовами, такими як PHP, Perl та Java. Він також широко використовується як на клієнтській, так і на серверній стороні, що дозволяє розробникам реалізовувати веб-додатки однією мовою;

−зниження навантаження на сервер – робота на стороні клієнта з використанням JavaScript має ще одну перевагу. JavaScript дозволяє перевіряти дані безпосередньо у веббраузері, а оновлення застосовуються лише до конкретних розділів вебсторінки, що скорочує кількість запитів, що надсилаються на сервер [14].

JavaScript може бути вбудований безпосередньо на вебсторінку або завантажений з окремого файлу з розширенням .js. Коли користувач переходить на вебсторінку, браузер запускає сценарій JavaScript разом з HTML і CSS, що призводить до створення функціональної сторінки, яка відображається у вікні браузера.

Скрипт JavaScript завантажується на комп'ютери користувачів та обробляється там. Це відрізняється від серверної мови програмування, де скрипт обробляється на сервері перед надсиланням результату до браузера [13].

При виконанні JavaScript-коду в браузері блоки коду виявляються та виконуються у порядку зверху донизу. Порядок виконання має важливе значення, тому необхідно переконатися, що при зверненні до об'єктів або змінних вони вже були оголошені попередніми частинами коду. Використання змінних, яким не було надано значення, може призвести до помилок і невизначеної поведінки. Тому рекомендується правильно організовувати код та оголошувати всі необхідні змінні перед їх використанням [12].

Python - це універсальна об'єктно-орієнтована мова програмування, яка знаходить застосування в різних галузях, таких як розробка програмного

забезпечення, веб-розробка, наука про дані та автоматизація процесів. Він відрізняється динамічною семантикою, високорівневими вбудованими структурами даних, динамічною типізацією та динамічним зв'язуванням, що робить його однією з найпотужніших мов для швидкої розробки додатків [10].

Python має безліч можливостей і переваг:

–платформна незалежність: Python може бути використаний на різних платформах, включаючи Windows, Mac, Linux, Raspberry Pi та інші. Це дозволяє розробникам створювати програми, які можуть працювати на різних операційних системах без змін коду.

–простий синтаксис - Python має зчитуваний і зрозумілий синтаксис, який можна порівняти з англійською мовою. Це робить код Python більш лаконічним і дозволяє розробникам писати менше рядків коду в порівнянні з іншими мовами програмування. Це спрощує розробку та підтримку коду.

–багатофункціональність: Python підтримує різні програмні парадигми, включаючи процедурне, об'єктно-орієнтоване та функціональне програмування.

–велика стандартна бібліотека: Python поставляється з великою стандартною бібліотекою, яка надає різні модулі та інструменти для широкого спектру завдань. Це дозволяє розробникам швидко вирішувати типові завдання без необхідності писати код із нуля [11].

Ця мова програмування відрізняється простотою використання і може бути використана в багатьох технічних областях, таких як машинне навчання, наука про дані, автоматизація та інші.

На відміну від деяких інших мов програмування, які також здатні вирішувати завдання в цих галузях, Python вирізняється своєю здатністю розв'язувати задачі в кожній з них. Це пов'язано з його гнучкістю та багатим набором інструментів та бібліотек, спеціалізованих для різних областей.

У результаті, завдяки своїй універсальності, простоті використання та інтерпретованості, Python стає мовою вибору для багатьох розробників та фахівців у різних галузях техніки та науки [10].

2.5. Опис структури програми та алгоритмів її функціонування

Загальний огляд того, як функціонує додаток Django:

–Для початку роботи потрібно створити новий проект Django. Команда django-admin startproject створить кілька файлів і каталогів, які забезпечують базову структуру проекту.

–Усередині проекту потрібно створювати окремі додатки, які виконують певні функції. Кожен застосунок має свою сувору структуру, моделі, подання та шаблони.

–Крім цього потрібно створити моделі для взаємодії з БД. Вони визначають структуру даних додатка.

–Після визначення моделей потрібно створити міграції, які застосують ці зміни до бази даних. Міграції автоматично створюють і оновлюють схему бази даних на основі моделей.

–Views визначають, як додаток обробляє запити. За це відповідають функції або класи, які приймають запит, обробляють його та повертають відповідь. Views можуть взаємодіяти з моделями, шаблонами та іншими компонентами програми.

–Шаблони Django дають змогу відокремити логіку відображення від дизайну. Вони визначають, як дані мають бути відображені на сторінках. Шаблони можуть використовувати змінні, цикли, умовні оператори та інші конструкції для генерації динамічного вмісту.

–Проєкт використовує файли URL-конфігурації для визначення, які подання мають бути викликані під час обробки конкретних

–URL-запитів. У файлі urls.py проекту визначені відповідності між URL-адресами та Views. Django автоматично обробляє вхідні запити та спрямовує їх до відповідного View на основі визначених маршрутів.

Нижче надано скріншот директорії проекту, а також таблиця, яка пояснює, для чого призначена кожна папка та файл (рис. 2.8. , табл. 2.1.)

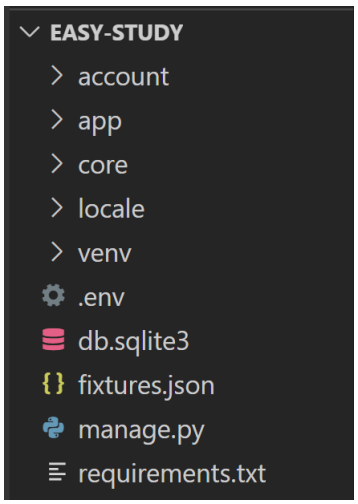


Рис. 2.8. Директорії проекту

Таблиця 2.1

Таблиця призначення каталогів та файлів проекту

Назва файлу / директорії	Зміст / призначення
account	Додаток Django, який реалізує логіку користувача
app	Головний додаток Django, містить у собі всю іншу логіку
core	Коренева директорія проекту, містить налаштування та інші конфігураційні файли проекту
locale	Містить .mo і .po файли, які роблять можливою багатомовність на сайті
venv	Містить віртуальне середовище python (virtual environment)
.env	Конфігураційний файл, який використовується для зберігання змінних середовища проєкту
db.sqlite3	Оскільки django за замовчуванням використовує цю базу даних для локальної розробки, вона присутня в каталозі. Насправді використовується PostgreSQL.

Продовження табл. 2.1

Назва файлу / директорії	Зміст / призначення
fixtures.json	Містить попередньо завантажені дані у форматі json
manage.py	Скрипт командного рядка, що забезпечує зручний інтерфейс для виконання різних завдань, пов'язаних з управлінням проектом.
requirements.txt	Список залежностей проекту, таких як бібліотеки та версії

2.6. Обґрунтування та організація вхідних та вихідних даних програми

На рисунках 2.9. – 2.13. наведено діаграму зв’язків між таблицями бази даних проекту (ER-таблиця БД).

Нижче в таблицях 2.1 – 2.7 подано поля БД, типи полів (як в ER-таблиці), які можуть зберігати в собі інформацію, надану від користувача та згенеровану системою, і описи їхніх призначень (за що вони відповідають).

Як вихідні дані надано всі дані, до яких є доступ у відповідних осіб, що мають доступ.

Якщо є дужки біля типу поля - це обмеження поля за кількістю символів/макс. значення.

Тип поля представлений не в традиційному форматі, а відповідно до django models (*from django.db import models*)

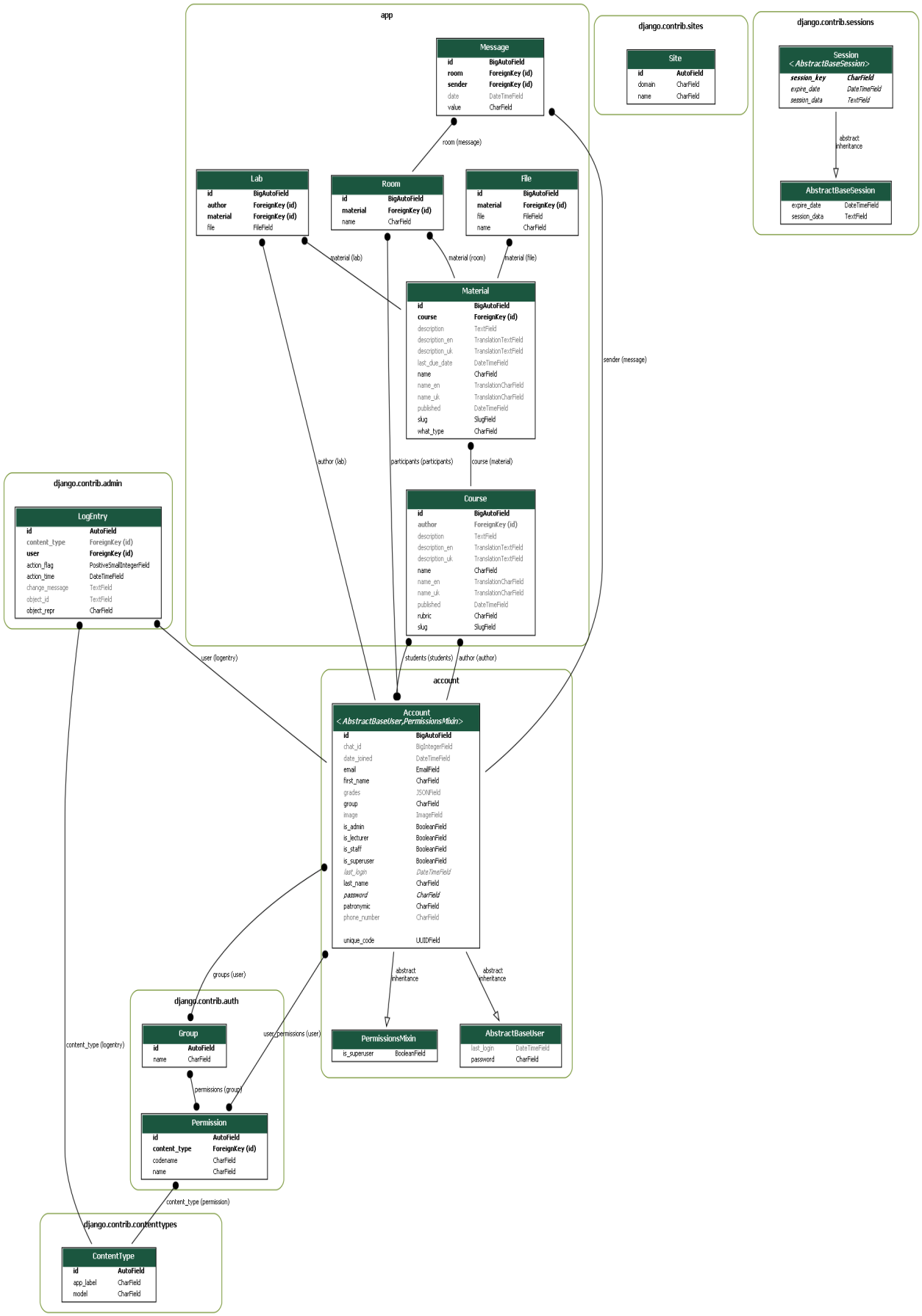


Рис. 2.9. ER-таблиця БД

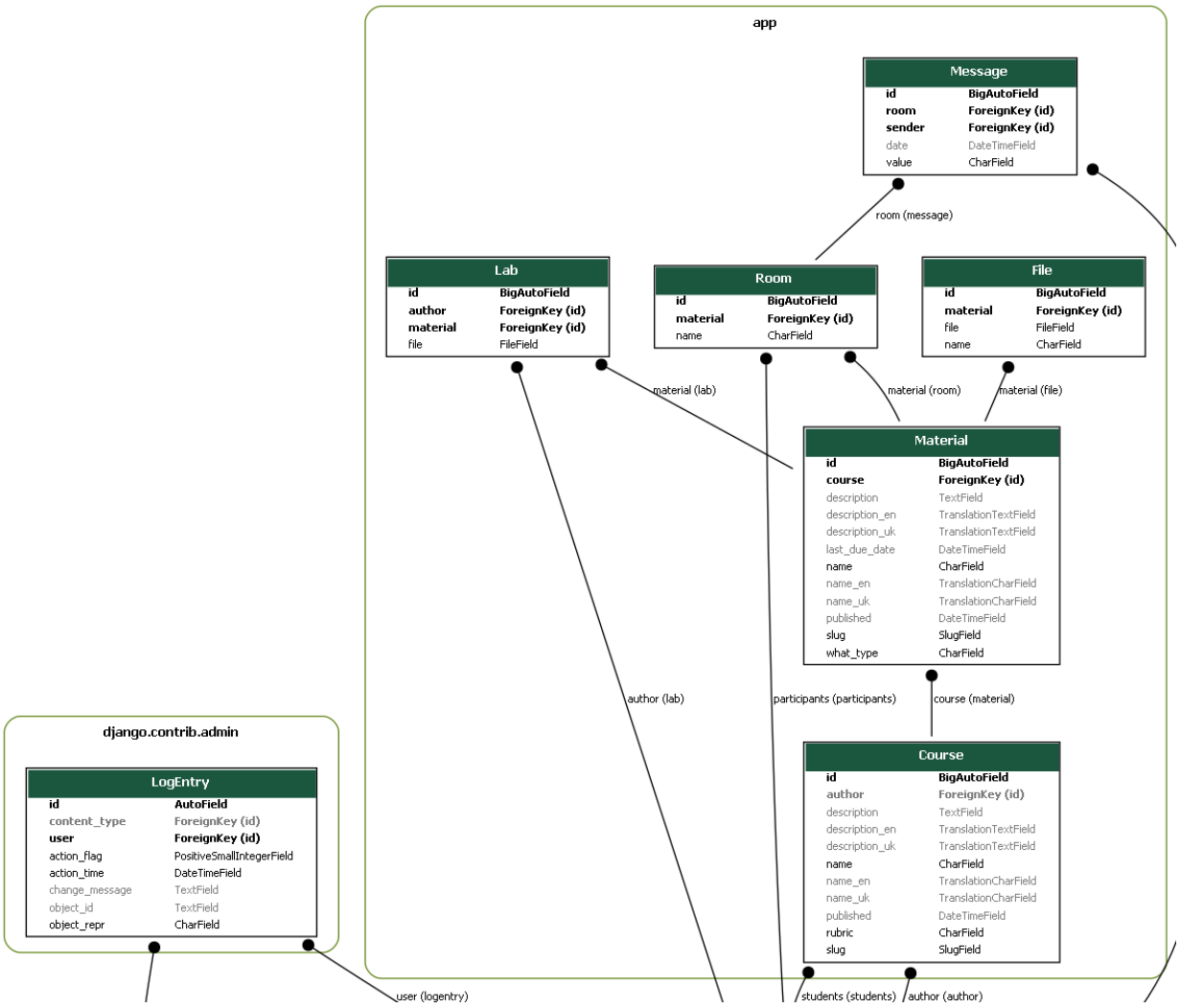


Рис. 2.10. Таблиці App та Django.contrib.admin

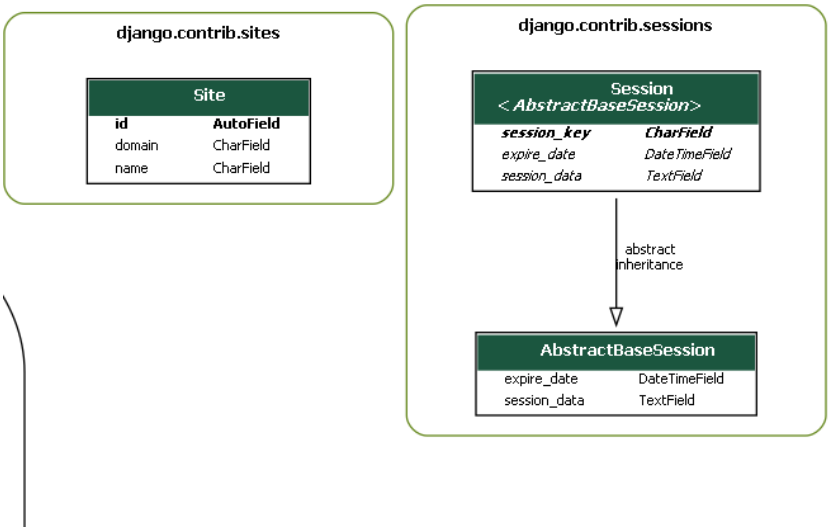


Рис. 2.11. Таблиці Django.contrib.sites та Django.contrib.sessions

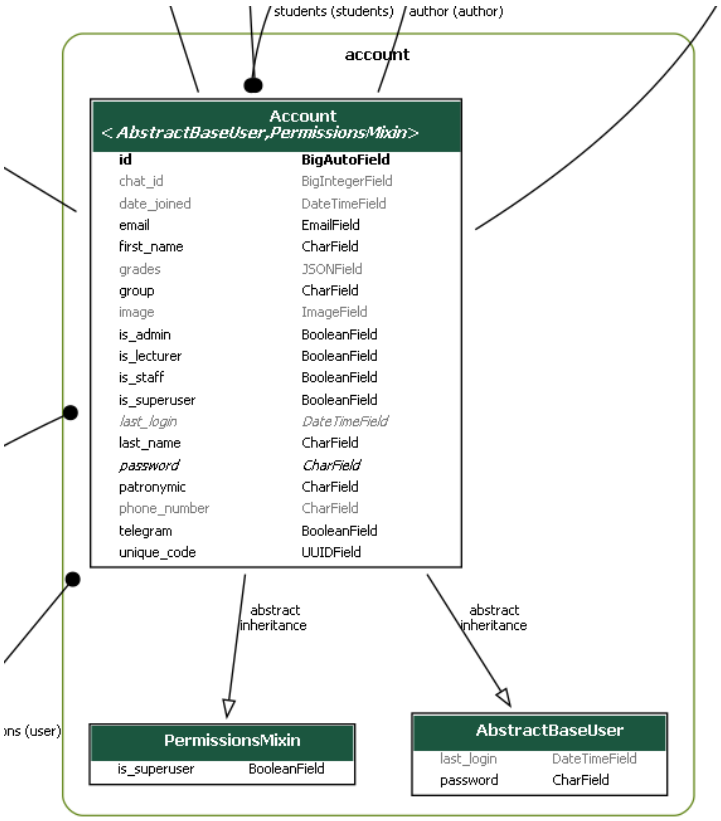


Рис. 2.12. Таблиця Account

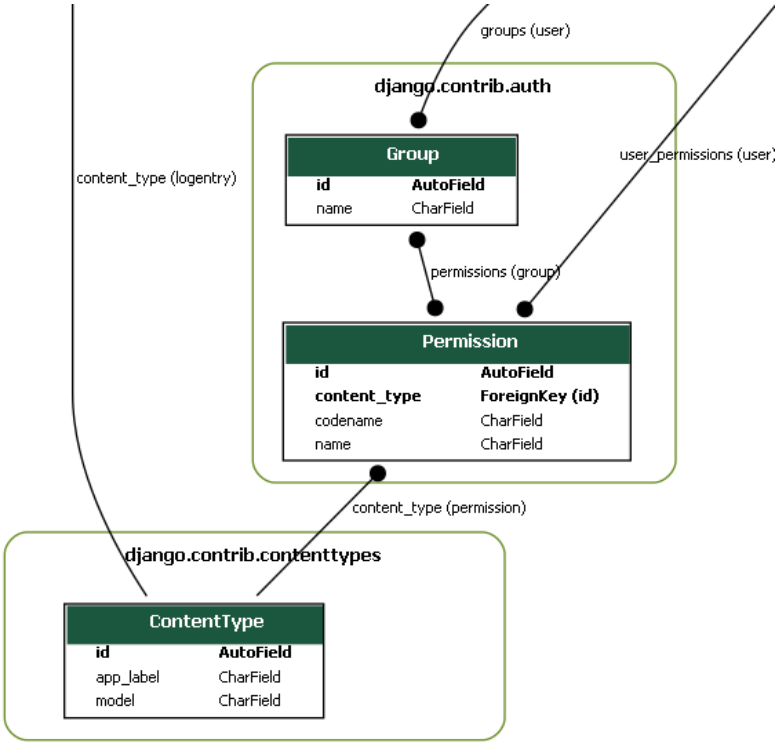


Рис. 2.13. Таблиці Django.contrib.auth та Django.contrib.contenttypes

Таблиця 2.2

База даних Account

Назва поля	Тип даних	Опис призначення поля
id	AutoField	Унікальний ідентифікатор запису в БД
image	ImageField	Відносний шлях до завантаженого зображення користувачем
email	EmailField(60)	Електронна пошта
phone_number	CharField(13)	Ім'я
first_name	CharField(30)	Прізвище
last_name	CharField(30)	По батькові
patronymic	CharField(30)	Група
group	CharField(10)	Id чату з викладачем
chat_id	BooleanField(False)	Дата реєстрації
date_joined	BigIntegerField	Чи є користувач адміністратором
is_admin	DateTimeField	Чи є користувач персоналом
is_staff	BooleanField(False)	Чи є користувач суперкористувачем
is_superuser	BooleanField(False)	Унікальний код (можливо знадобиться для майбутніх інтеграцій з microsoft)
unique_code	BooleanField(False)	Чи є користувач викладачем

Продовження табл. 2.2

Назва поля	Тип даних	Опис призначення поля
is_lecturer	UUIDField	Оцінки (якщо користувач - студент)
password	CharField	Пароль
grades	BooleanField(False)	Унікальний ідентифікатор запису в БД

Таблиця 2.3

База даних Course

Назва поля	Тип даних	Опис призначення поля
id	AutoField	Унікальний ідентифікатор запису в БД
name	CharField(100)	Назва курсу
description	TextField(2000)	Опис курсу
slug	SlugField(200)	Певне поле, яке відповідає за посилання до запису (у веб-браузері)
published	DateTimeField	Дата публікації
author	ForeignKey	Автор (викладач)
students	ManyToManyField	Студенти на курсі
rubric	CharField	Рубрика

Таблиця 2.4

База даних Material

Назва поля	Тип даних	Опис призначення поля
id	AutoField	Унікальний ідентифікатор запису в БД
name	CharField(100)	назва матеріалу
description	TextField(1000)	опис матеріалу
published	DateTimeField	дата публікації
last_due_date	DateTimeField	останній термін здачі
course	ForeignKey	курс
slug	SlugField	Певне поле, яке відповідає за посилання до запису (у веб-браузері)
what_type	CharField	тип матеріалу (лекція, практика, інше)

Таблиця 2.5

База даних File

Назва поля	Тип даних	Опис призначення поля
id	AutoField	Унікальний ідентифікатор запису в БД
file	FileField	Відносний шлях до завантаженого файлу користувачем

Продовження табл. 2.5

Назва поля	Тип даних	Опис призначення поля
name	CharField(100)	Назва файлу
material	ForeignKey	До якого матеріалу належить

Таблиця 2.6

База даних Lab

Назва поля	Тип даних	Опис призначення поля
id	AutoField	Унікальний ідентифікатор запису в БД
file	FileField	Відносний шлях до завантаженого файлу студентом
author	ForeignKey	Хто завантажив (автор-студент)
material	ForeignKey	До якого матеріалу належить

Таблиця 2.7

База даних Room

Назва поля	Тип даних	Опис призначення поля
id	AutoField	Унікальний ідентифікатор запису в БД

Продовження табл. 2.6

Назва поля	Тип даних	Опис призначення поля
name	CharField(500)	Назва чату
material	ForeignKey	До якого матеріалу належить
participants	ManyToManyField	Учасники чату (студент, викладач)

Таблиця 2.8

База даних Message

Назва поля	Тип даних	Опис призначення поля
id	AutoField	Унікальний ідентифікатор запису в БД
value	CharField(10000)	Текст повідомлення
room	ForeignKey	До якого чату належить
date	DateTimeField	Дата відправлення
sender	ForeignKey	Відправник

2.7. Опис розробленого програмного продукту

2.7.1. Використані технічні засоби

Для розробки було використано:

а) Ноутбук

1) Версія ОС - Windows 10

2) Процесор - Intel Core i7-8550U 1.99 GHz

3) Оперативаня пам'ять - 16 ГБ

4) Тип системи - 64-розрядна операційна система, процесор x64

б) Браузер - Chrome 113.0.5672.127

в) Комп'ютерна миша

2.7.2. Використані програмні засоби

Для розробки було використано:

а) Серверна частина:

1) мова програмування Python версії 3.10.4 (рис. 2.14.);
2) фреймворк Django версії 4.0.5 (рис. 2.14.);
3) бібліотека django-modeltranslation версіЇ 0.17.3;
4) бібліотека XlsxWriter версіЇ 3.1.0;
5) бібліотека psycopg2 версіЇ 2.9.2.

б) Візуальна частина:

1) HTML - є стандартизованою мовою розмітки документів, призначеною для відображення веб-сторінок у браузері. Її використовують для опису структури та вмісту веб-сторінки, включно з текстом, зображеннями, посиланнями, таблицями та іншими елементами [11];

2) CSS - (Cascading Style Sheets) - це мова, яка використовується для візуального оформлення вмісту HTML-документа. Вона надає можливість задавати різні стилі та зовнішній вигляд елементів веб-сторінки, включно з кольорами, шрифтами, відступами, розмірами та розташуванням. CSS дає змогу створювати узгоджений і привабливий дизайн для веб-сторінок, а також забезпечує поділ вмісту та представлення. Це означає, що HTML-код відповідає за структуру і вміст сторінки, тоді як CSS визначає, як ці елементи будуть відображатися на екрані [12];

3) мова програмування Javascript.

Рис. 2.14. Версії мови програмування Python та фреймворку Django

2.7.3. Виклик та завантаження програми

Для того, щоб запустити проєкт, потрібно здійснити такі кроки:

– Відкрити термінал (Win + R)

– Перейти в папку проекту cd easy-study

– Побудувати віртуальне оточення. Це потрібно для того, щоб ми не встановлювали всі необхідні залежності в кореневу директорію системи, а тримали їх в одному місці проєкту, щоб їх видалити, щоб вони не займали місця на дисковому простанстві python -m venv venv

– Запустити віртуальне оточення venv/scripts/activate

– Встановити всі залежності у віртуальне оточення pip install -r requirements.txt

– Запустити процес мігрування (для створення БД і таблиць) python manage.py migrate

– Завантажити фікстури. Це дані у форматі JSON, які завантажуються в БД, щоб відображати дані за замовчуванням. python manage.py loaddata fixtures.json

– Запустити проєкт на 80 порту. python manage.py runserver 0.0.0.0:80

– Перейти за адресою http://127.0.0.1/

2.7.4. Опис інтерфейсу користувача

На головній сторінці (рис. 2.15.) зареєстрований та незареєстрований користувач може бачити для чого було створено платформу. На верхній панелі є зміна мови (українська/англійська), а також реєстрація або вхід. Кнопка "Всі курси" доступна всім, але незареєстрований користувач не зможе побачити список курсів (рис. 2.16.).

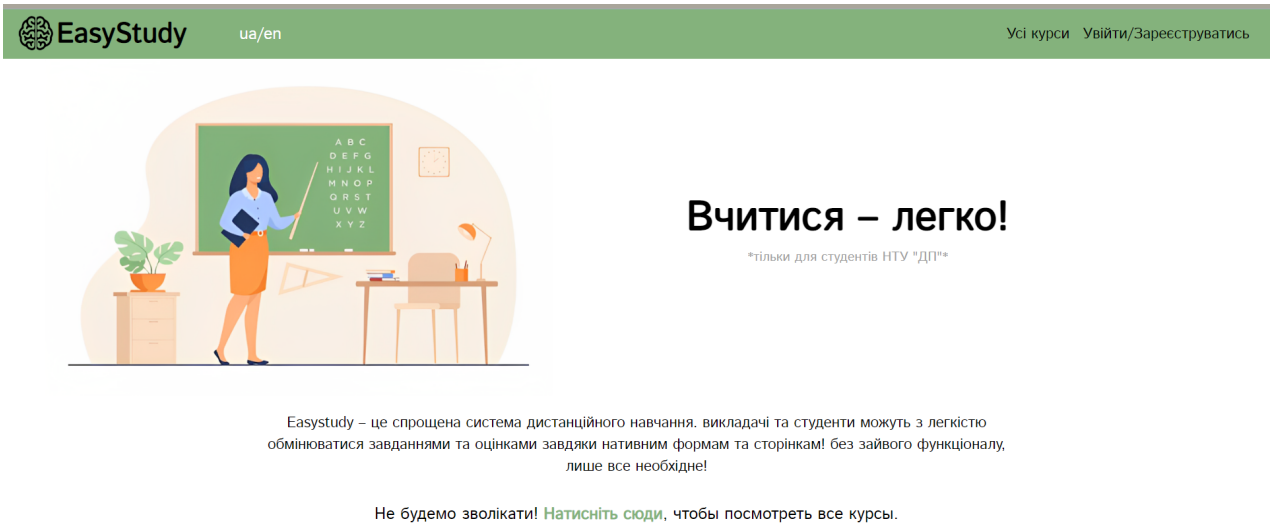


Рис. 2.15. Головна сторінка

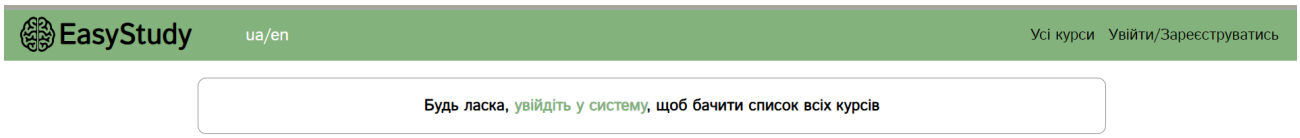


Рис. 2.16. Сторінка «Усі курси» для незареєстрованих користувачів

При переході на сторінку "увійти/зареєструватися" (рис. 2.17.) користувачеві відразу запропонують увійти вже на існуючий обліковий запис, але якщо такого ще немає, то внизу, натиснувши кнопку "створіть", можна зробити свій новий обліковий запис.

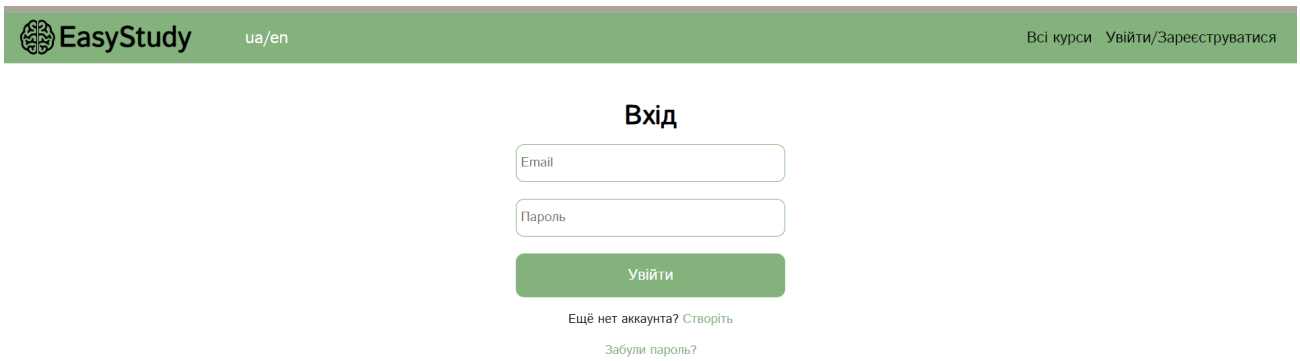


Рис.2.17. Сторінка входу

При реєстрації (рис. 2.18.) обов'язково потрібно вказувати корпоративну пошту університету НТУ "Дніпровська політехніка", оскільки додаток зроблений виключно для викладачів та студентів цього університету.

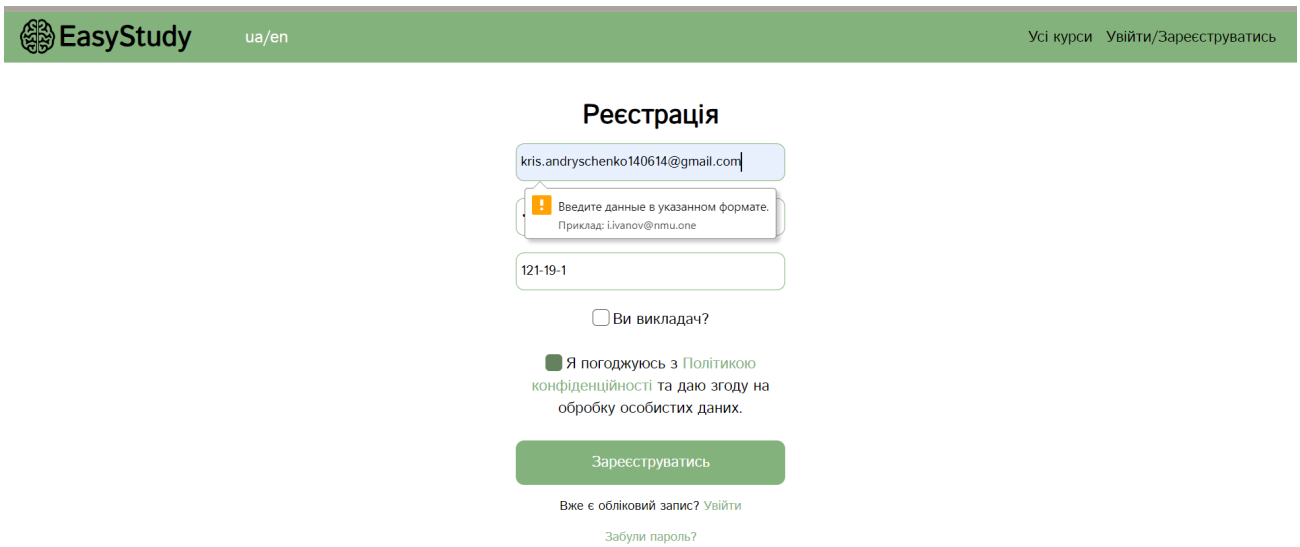


Рис. 2.18. Реєстрація користувача (студента) не під корпоративною поштою

Інтерфейс для викладача можна побачити на рис. 2.19. – 2.35.

У профілі викладача можна змінити дані або вийти з системи. Є також додавання курсів та перехід на них. З цієї сторінки можна перейти на всі курси через кнопку «Усі курси», на головну сторінку та змінити мову.

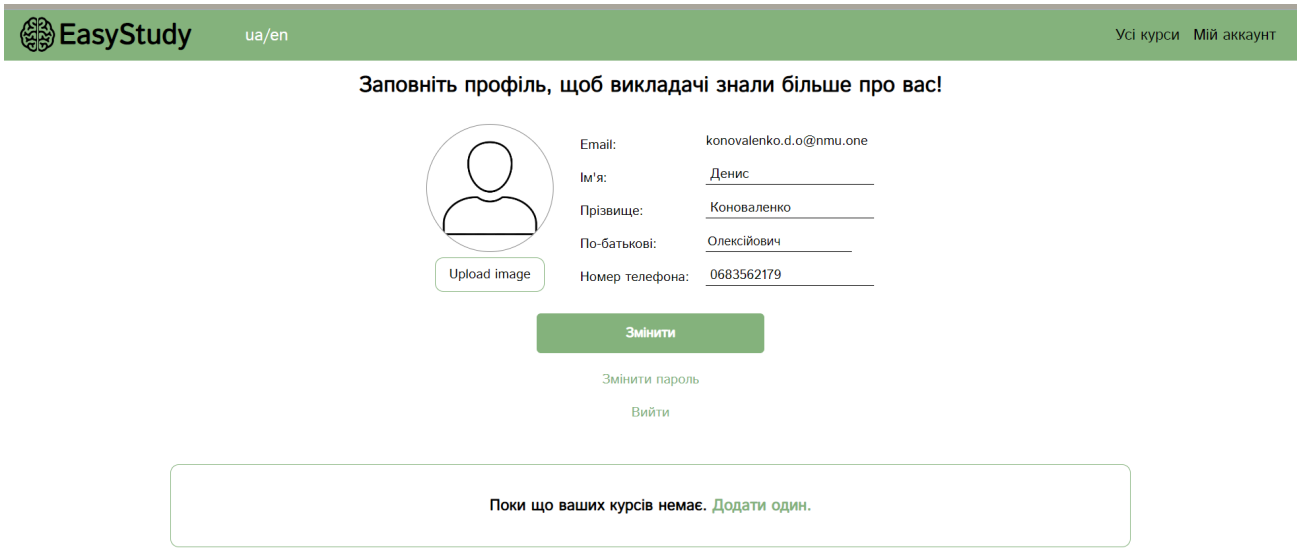


Рис. 2.19. Профіль викладача

Щоб додати фото користувачу, треба натиснути на «Upload image».

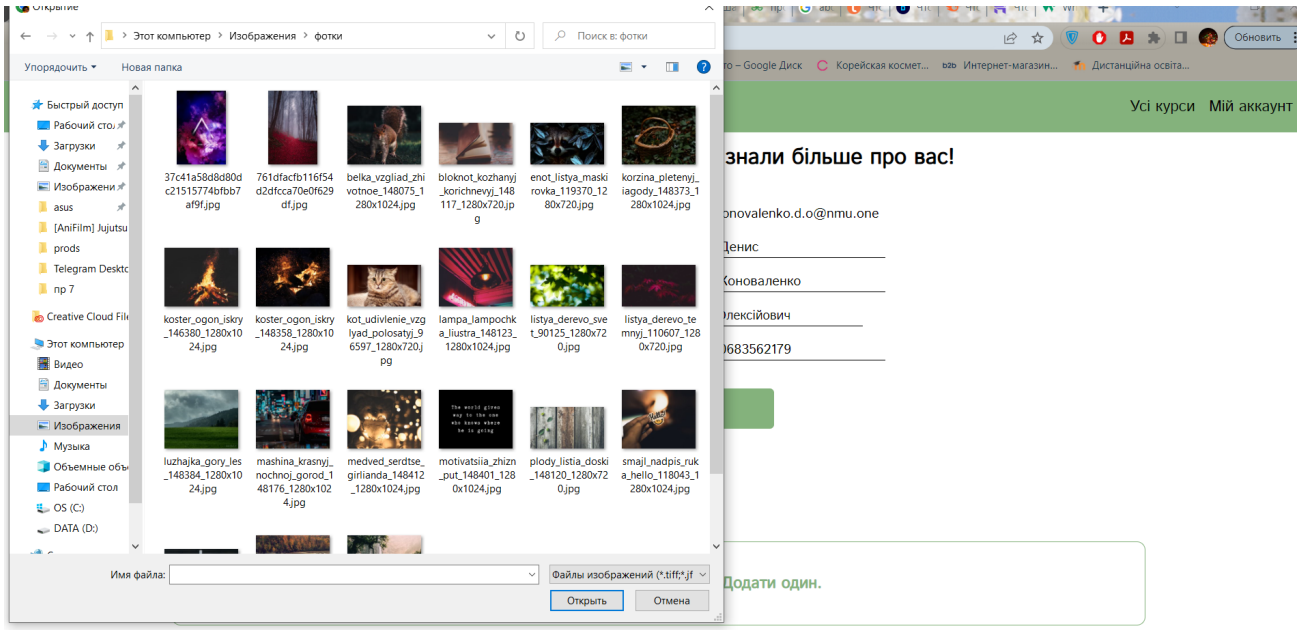


Рис. 2.20. Завантаження фото

Додати курс може тільки викладач зі свого профілю.

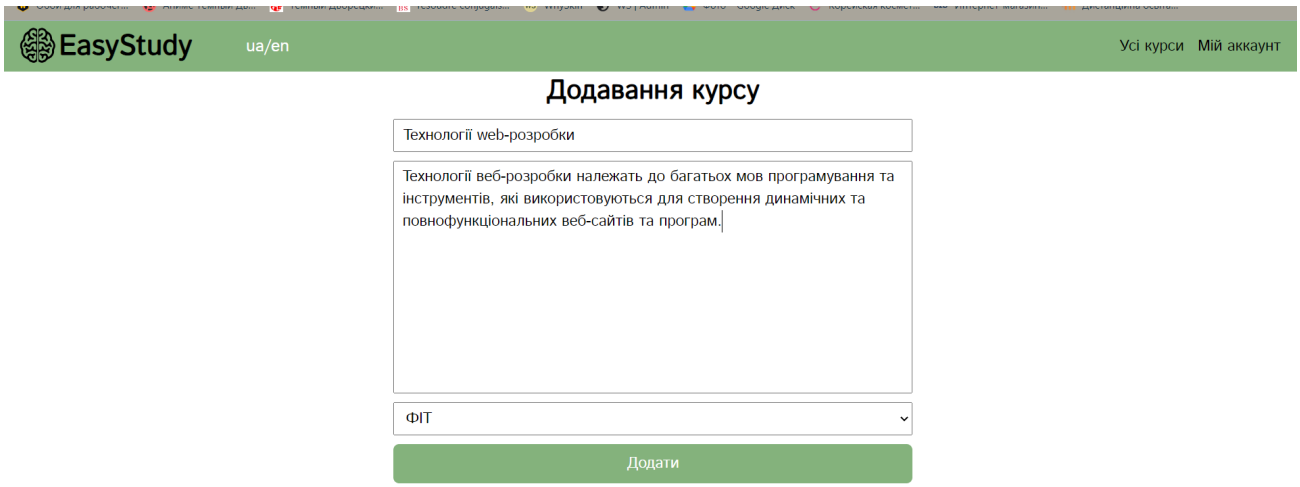


Рис. 2.21. Додавання курсу

Далі він з’явився у переліку всіх курсів, які є на платформі.

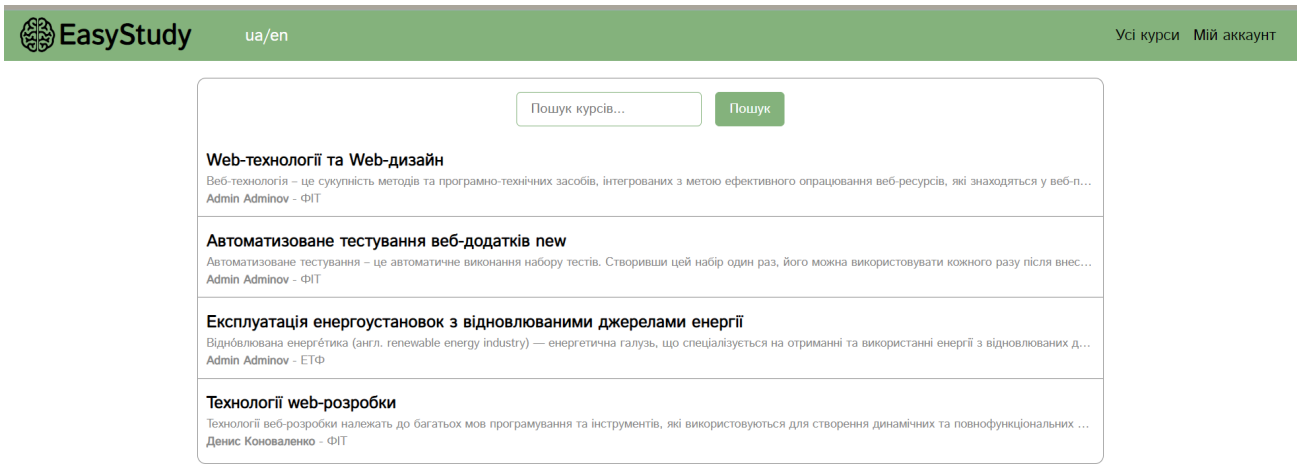


Рис. 2.22. Сторінка з усіма курсами

З цієї сторінки можна здійснити пошук певного курсу по його назві. З'явиться перелік всіх курсів, де є подібність до того, що писав користувач у пошуку.

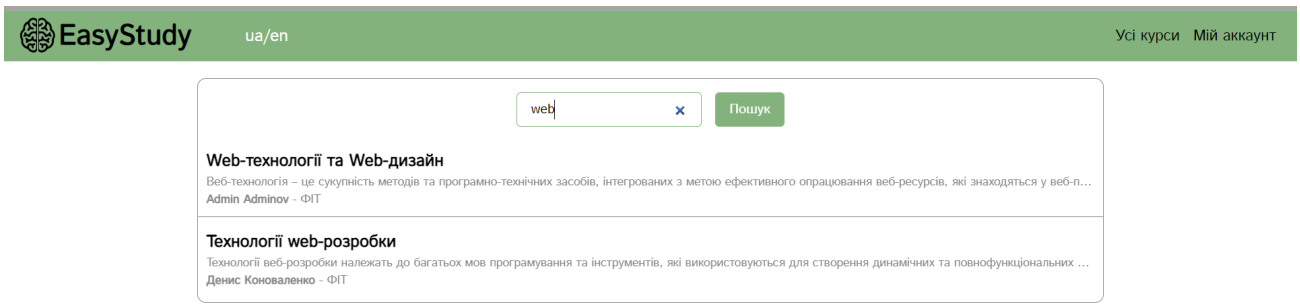


Рис. 2.23. Пошук курсу по назві

На сторінці курсу викладач може додавати матеріали, як практичні, так і лекційні. Повинен бути опис матеріалу, його назва та прикріплений файл з інформацією. «Лекція» та «Практика» виділені різними кольорами для того щоб користувачі могли швидко зорієнтуватися де якийсь матеріал.

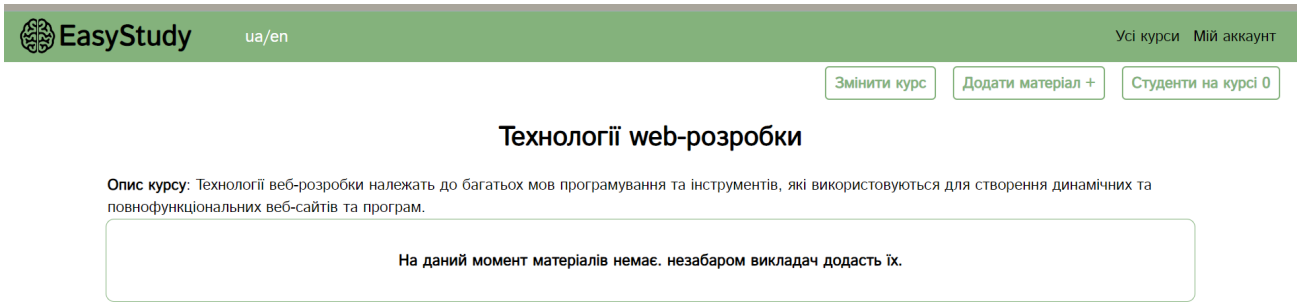


Рис. 2.24. Сторінка курсу

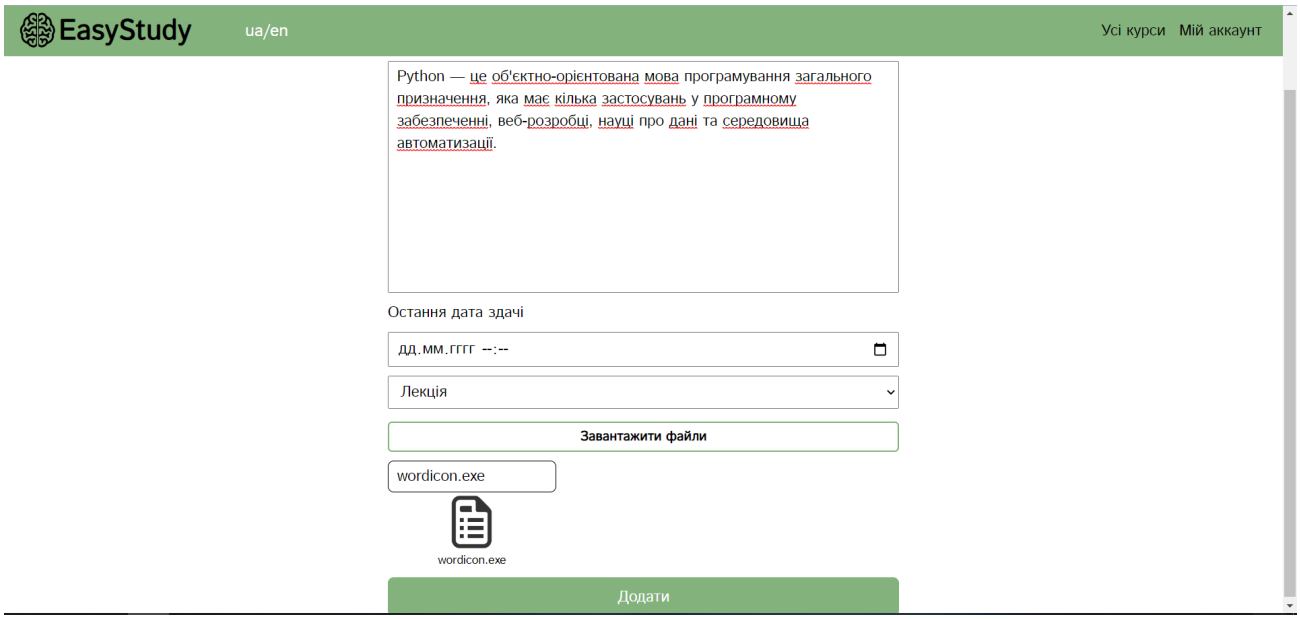


Рис. 2.25. Додавання лекційного матеріалу

При додаванні практичного завдання можна вказати термін його здачі за допомогою календарю.

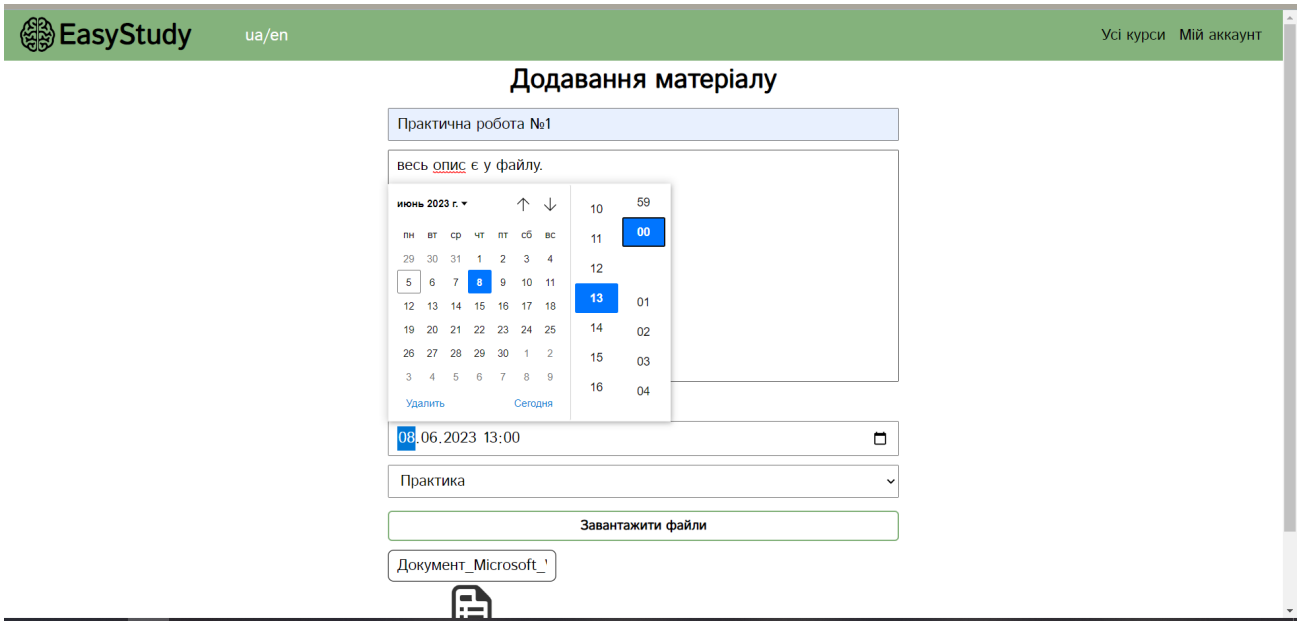


Рис. 2.26. Додавання практичного завдання

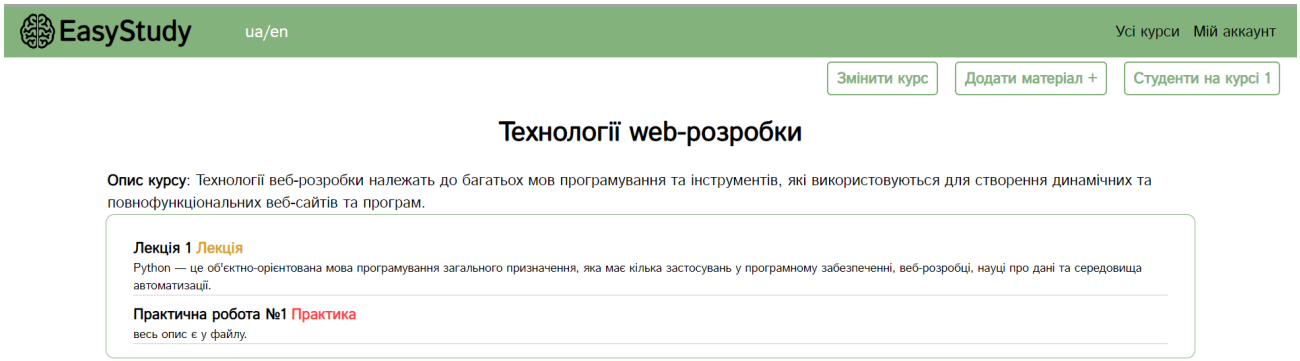


Рис. 2.27. Сторінка курсу зі всіма матеріалами

Через кнопку «Студенти на курсі» можна подивитися кількість та які саме студенти записані на цей курс. Їм можна написати повідомлення на пошту натиснувши «Написати». Автоматично в полях «відправник» та «одержувач» будуть пошти викладача та студента.

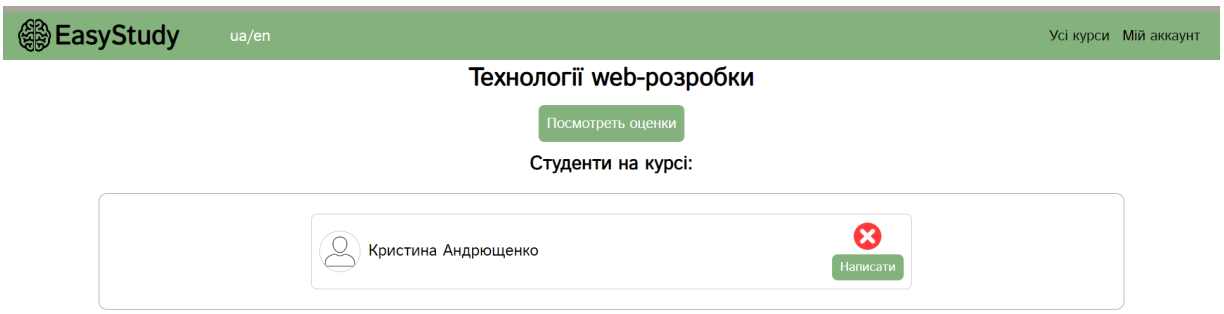


Рис. 2.28. Список студентів, які записані на курс

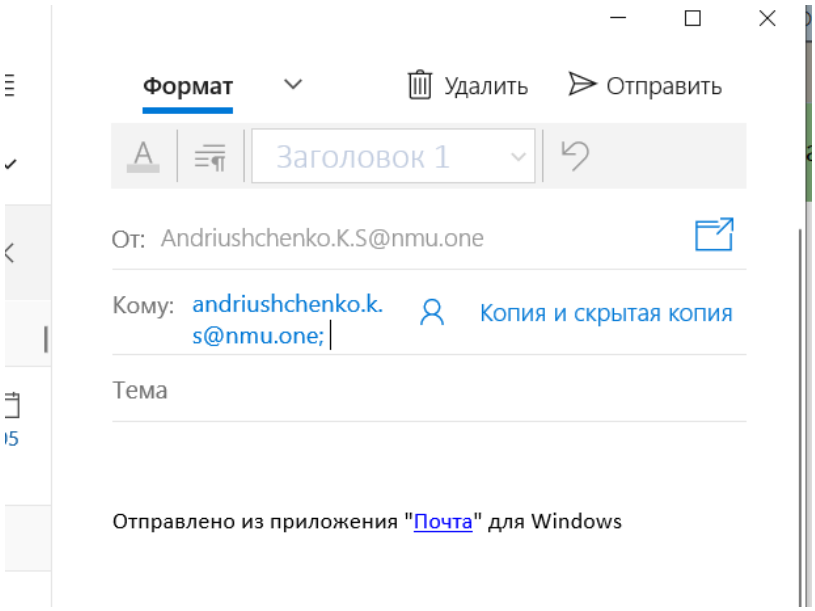


Рис. 2.29. Створення листа студенту

При переході на сторінку з практичним завданням можна побачити список студентів з їх роботами. Ці файли можна завантажити на перевірку та виставити оцінку перейшовши на іншу сторінку «Виставити оцінки».

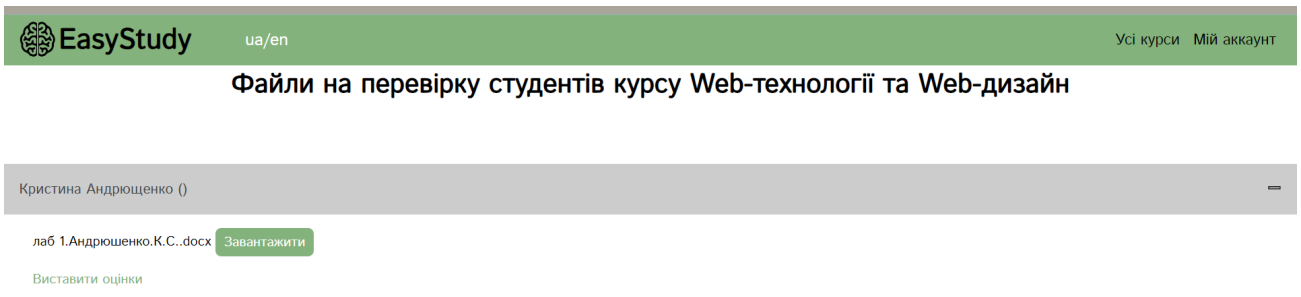


Рис. 2.30. Сторінка з файлами студентів

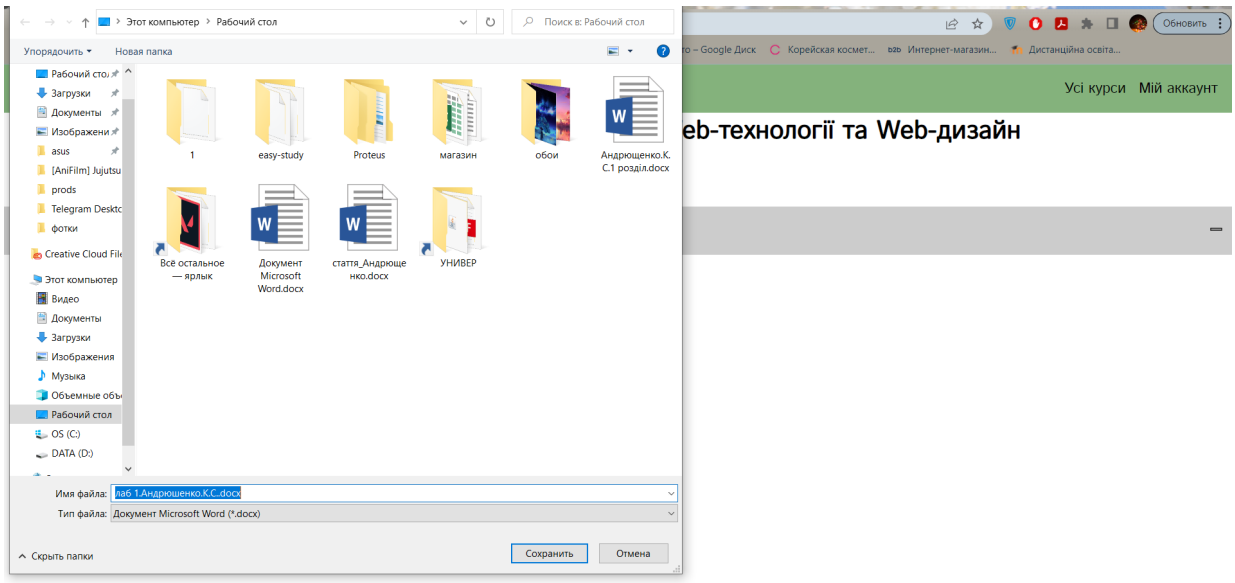
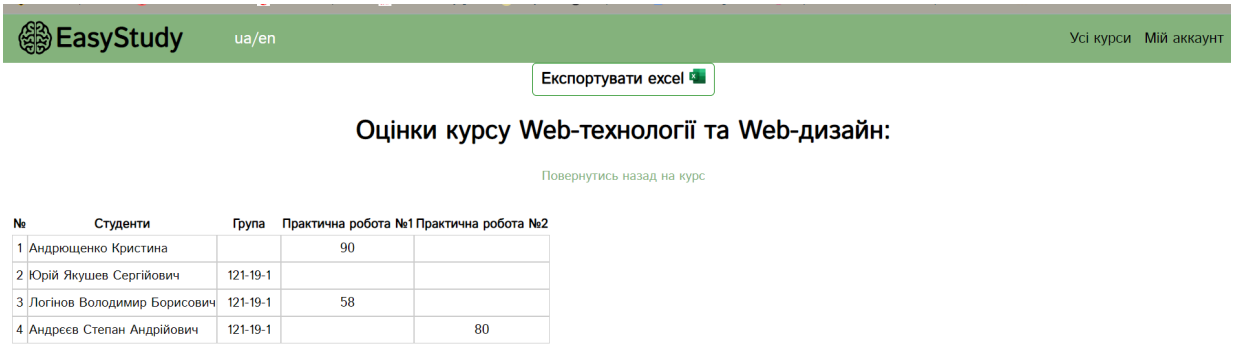


Рис. 2.31. Завантаження роботи студента



№	Студенти	Група	Практична робота №1	Практична робота №2
1	Андрющенко Кристина		90	
2	Юрій Якушев Сергійович	121-19-1		
3	Логінов Володимир Борисович	121-19-1	58	
4	Андрєєв Степан Андрійович	121-19-1		80

Рис. 2.32. Сторінка з оцінками студентів

Після виставлення всіх оцінок, викладач може експортувати їх у Excel файл.

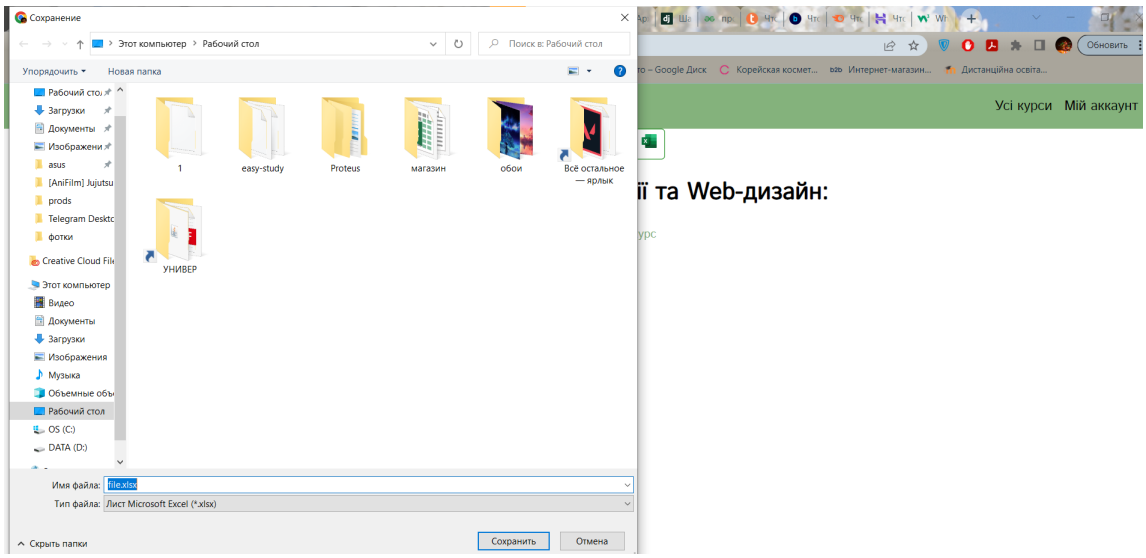
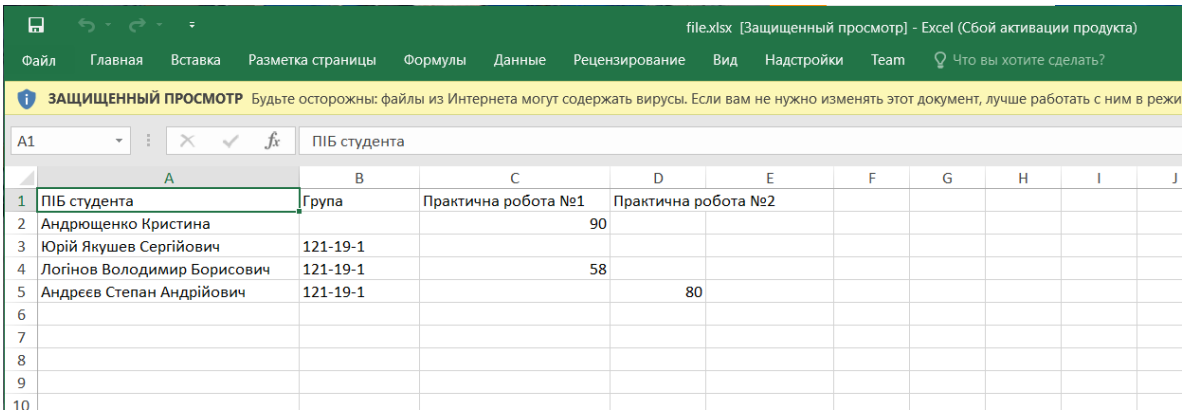


Рис. 2.33. Експортування Excel файлу



	A	B	C	D	E	F	G	H	I	J
1	ПІБ студента	Група	Практична робота №1	Практична робота №2						
2	Андрющенко Кристина		90							
3	Юрій Якушев Сергійович	121-19-1								
4	Логінов Володимир Борисович	121-19-1	58							
5	Андрєєв Степан Андрійович	121-19-1		80						
6										
7										
8										
9										
10										

Рис. 2.34. Excel файл з оцінками

На сторінці практичного завдання є список студентів, які написали викладачу. Цей чат створений для бистрої комунікації між викладачем та студентом.

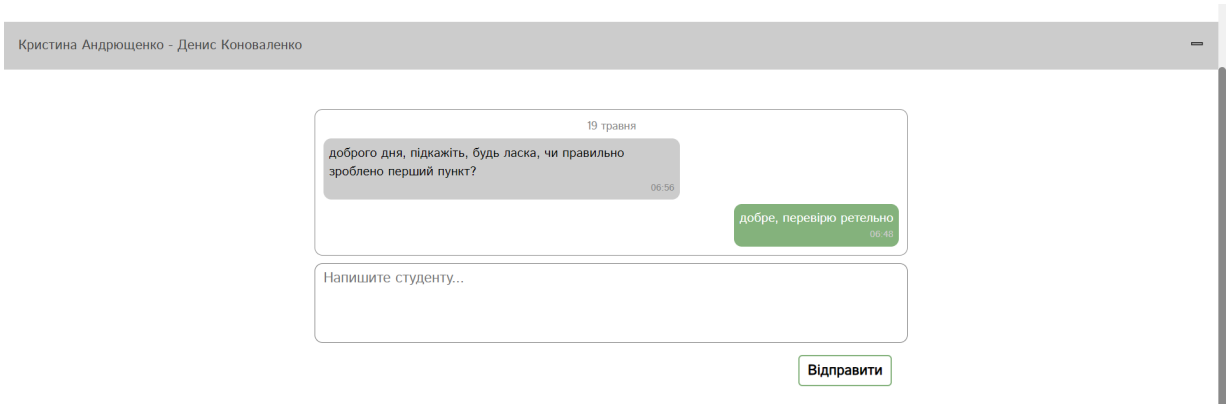


Рис. 2.35. Інтерфейс чату зі студентом

Інтерфейс для студента можна побачити на рис. 2.36. – 2.39.

Профіль студента відрізняється тим, що немає доступу до створення курсів, але є їх перегляд та можливість переглянути оцінки.

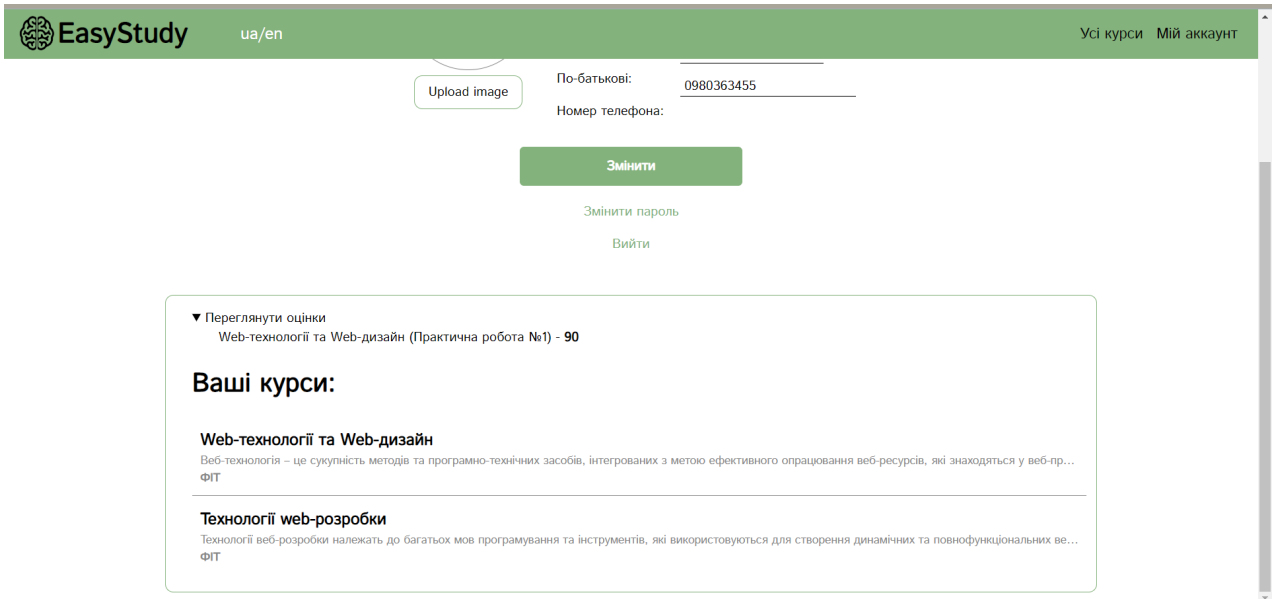


Рис. 2.36. Список курсів, на які записан студент

На сторінці матеріалу студент може завантажити файл, який розмістив викладач.

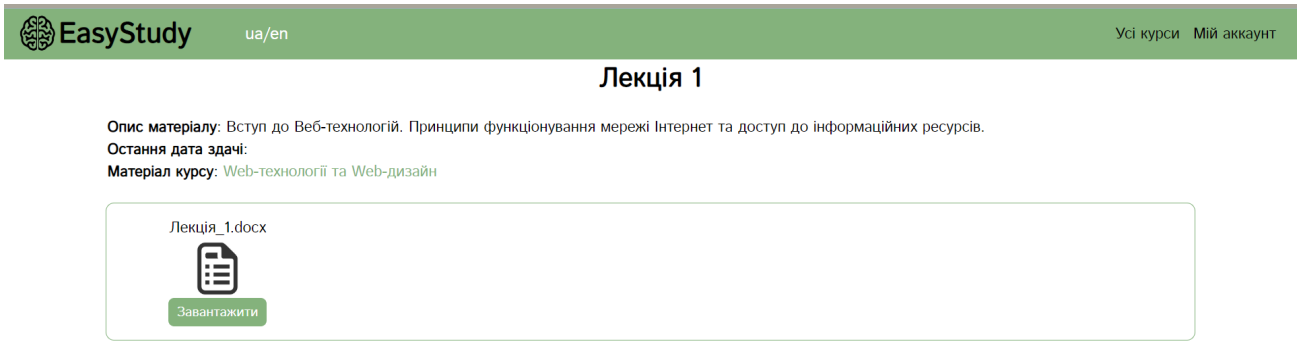


Рис. 2.37. Сторінка лекційного матеріалу

На сторінці з практичним завданням студент може здати свою роботу.

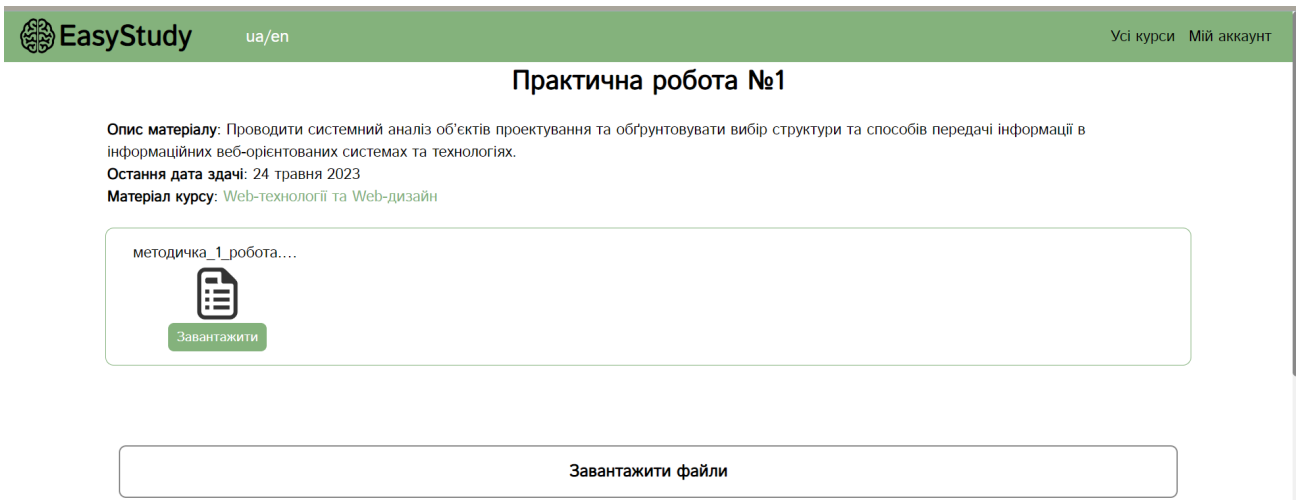


Рис. 2.38. Сторінка з практичними матеріалами

Нижче видно розташований файл та чат з викладачем.

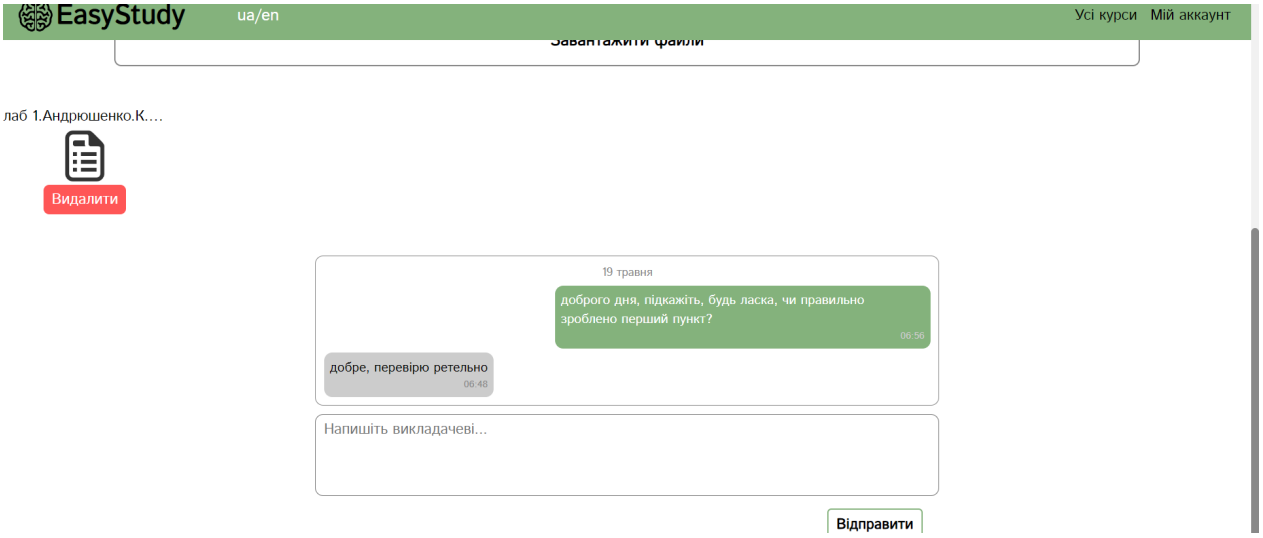


Рис. 2.39. Чат з викладачем

Інтерфейс адмін-панелі

На адмін-панель може заходити тільки користувач з правами адміна. Він може дивитися всю інформацію, що є на платформі, видаляти її та користувачів (рис. 2.40. – 2.48.).

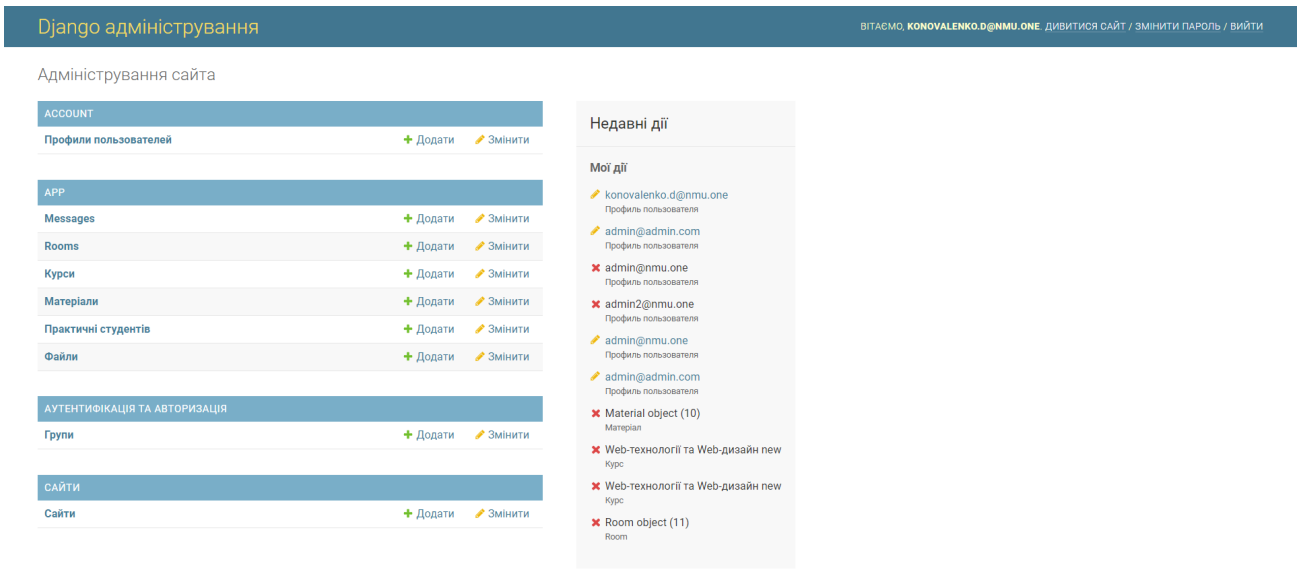


Рис. 2.40. Головна сторінка

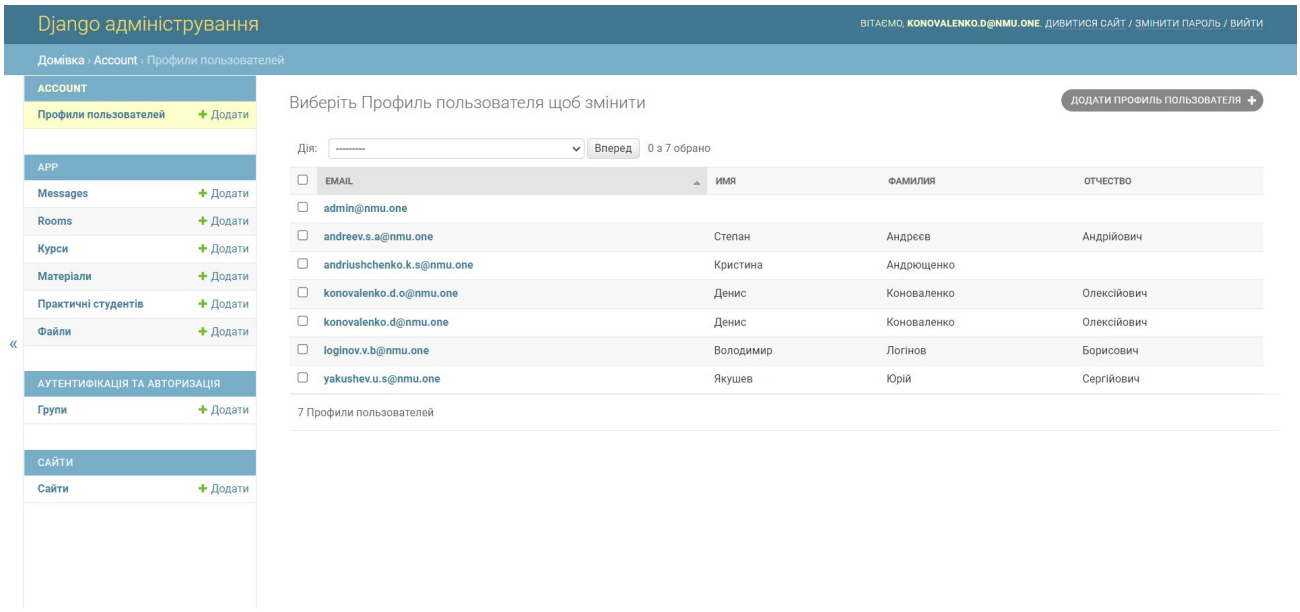


Рис.2.41. Перелік користувачів

Перейшовши на сторінку користувача (рис. 2.42.) можна побачити всю інформацію про нього: останній вхід, його дані, фотографію. Пароль хешується і зберігається в базі даних у такому вигляді. Він потрібен для того, щоб не зберігати пароль у тому вигляді, в якому його ввів користувач. Коли користувач захоче увійти в систему він веде свій оригінальний незахешований пароль, він захешується і порівняється з тим, що у вже зберігається в базі даних. Якщо вони рівні, тоді користувач ввів правильний пароль, якщо ні тоді програма не дозволяє вхід в систему.

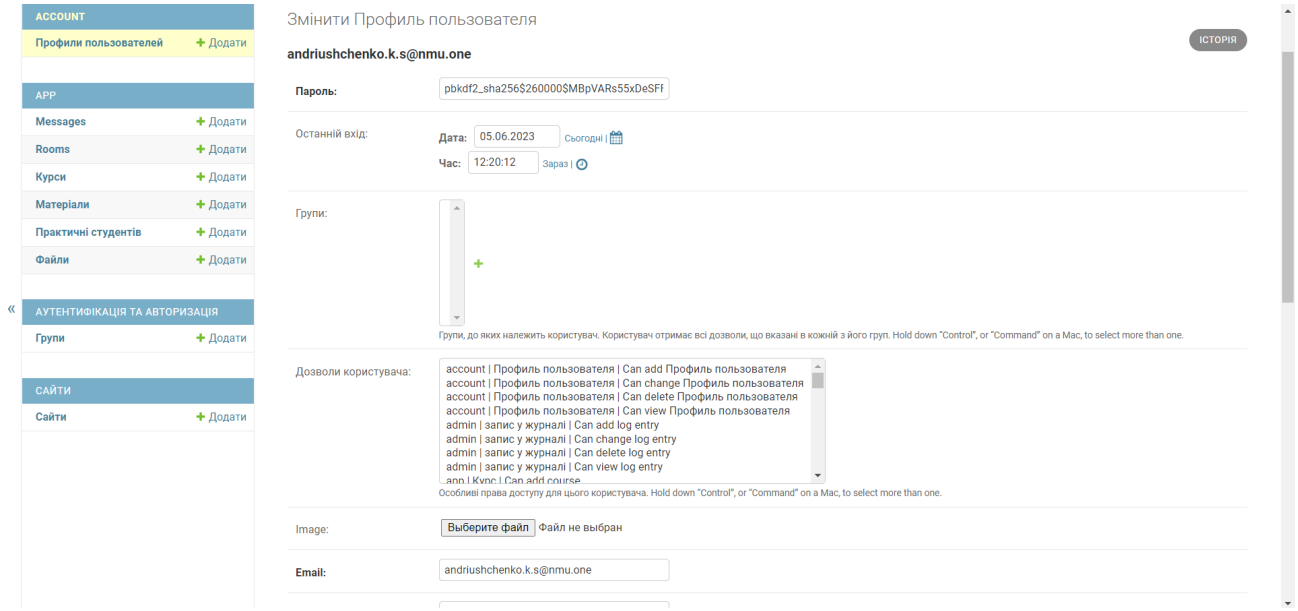


Рис.2.42. Сторінка з всією інформацією про користувача

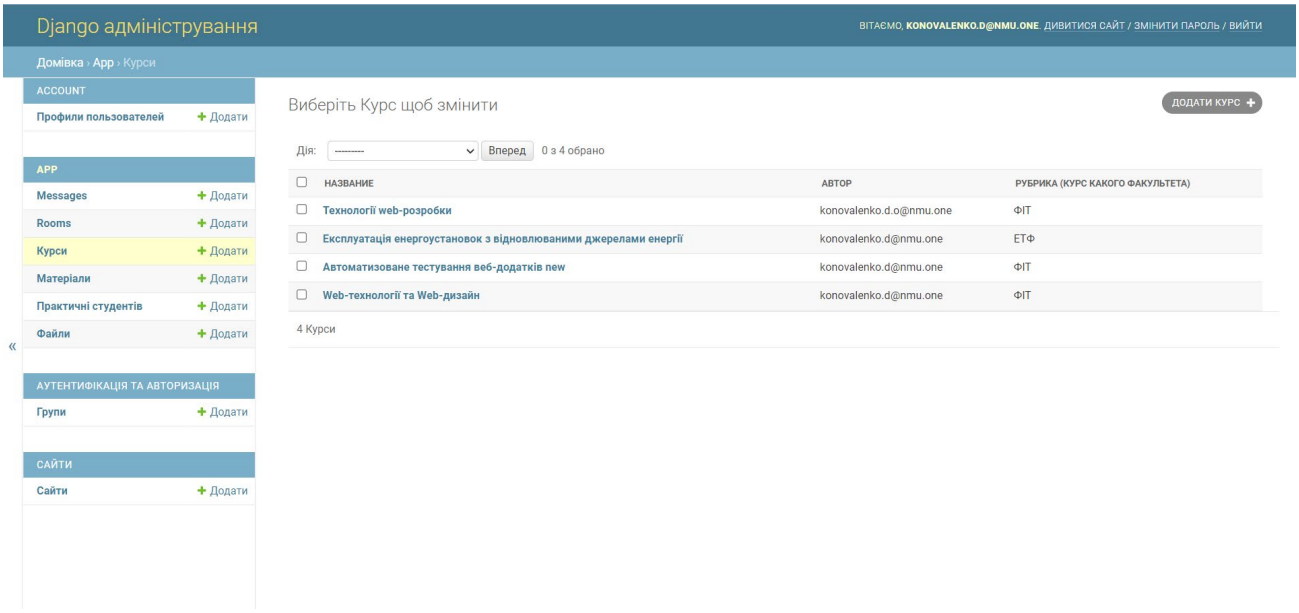


Рис.2.43. Сторінка зі всіма курсами на платформі

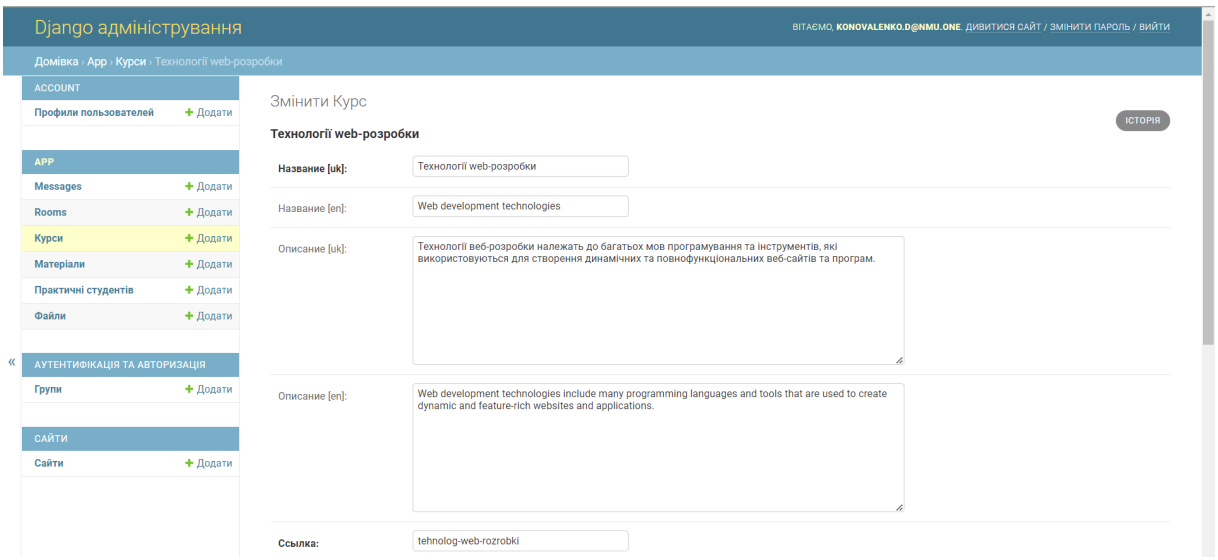


Рис.2.44. Інформація про курс

На сторінці з практичними роботами є перелік всіх зданих робіт.

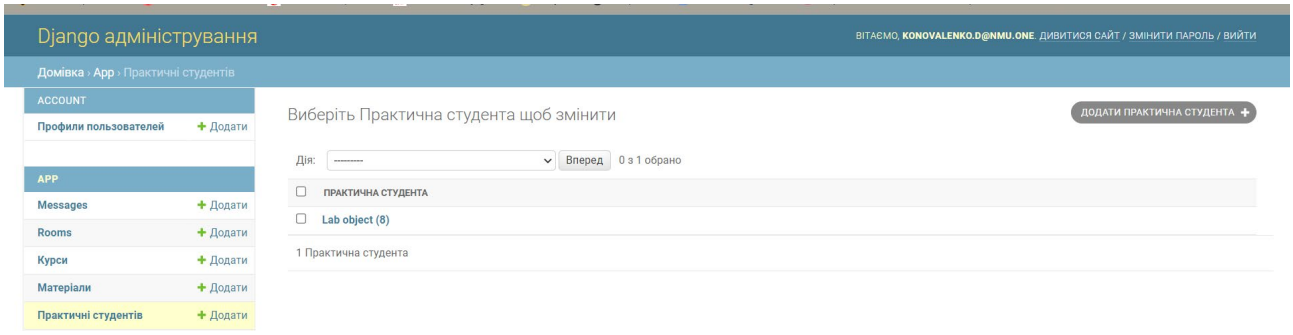


Рис. 2.45. Перелік зданих файлів

На сторінці з інформацією роботи адміністратор може бачити що це за файл, де він був опублікован та ким. Також його можна видалити.

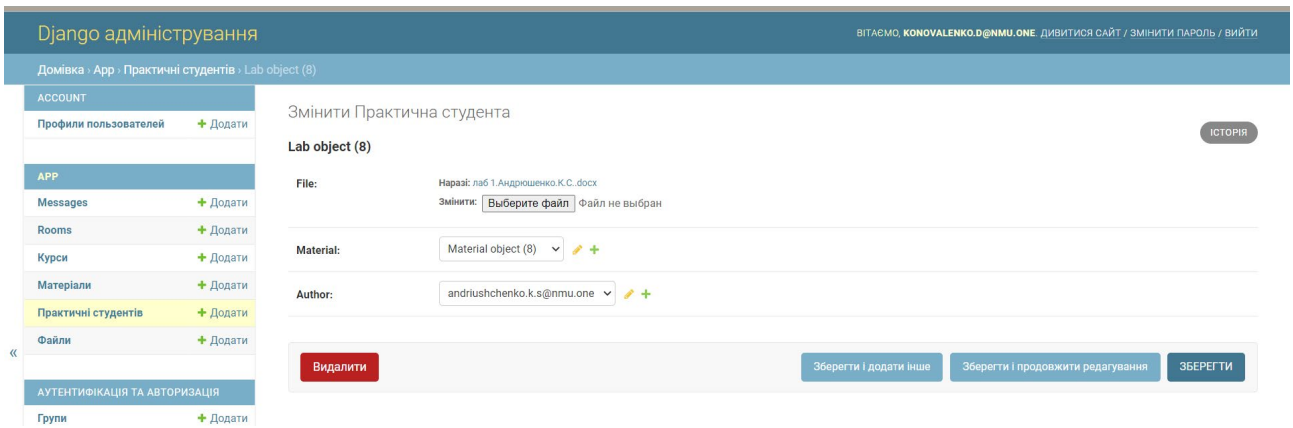


Рис. 2.46. Сторінка з інформацією про файл

Перелік всіх текстових повідомлень через чат можна подивитись на сторінці Messages.

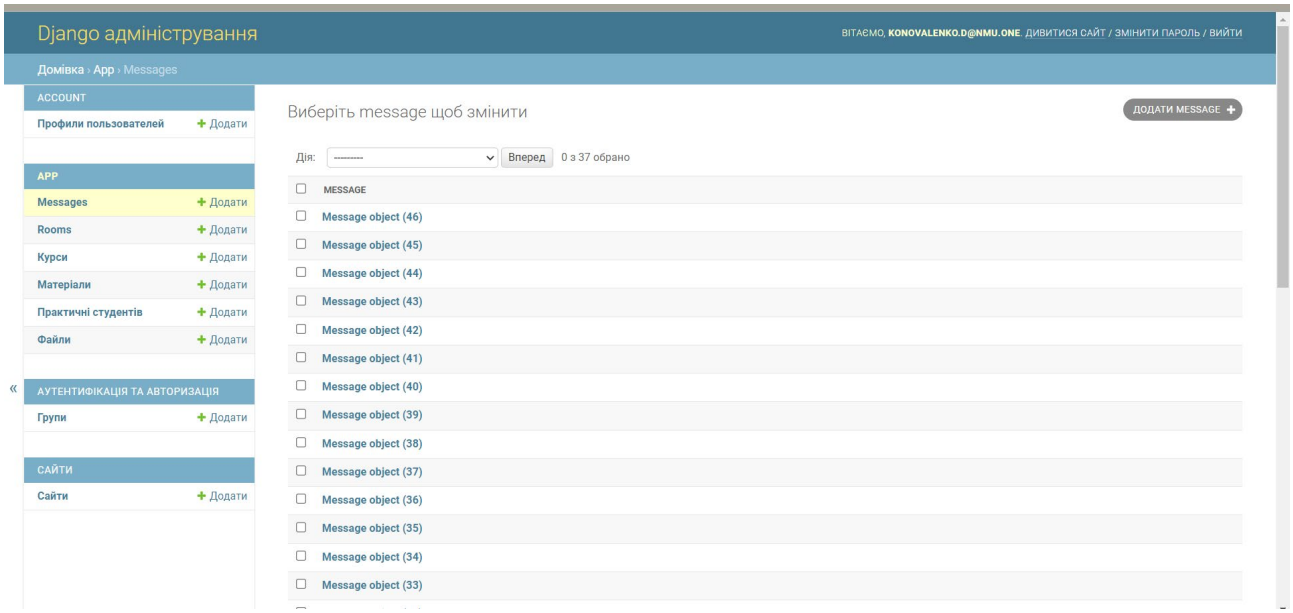


Рис. 2.47. Список всіх повідомлень

На сторінці з керуванням повідомлення можна видалити його та подивиться коли воно було відправлене та у якому чаті.

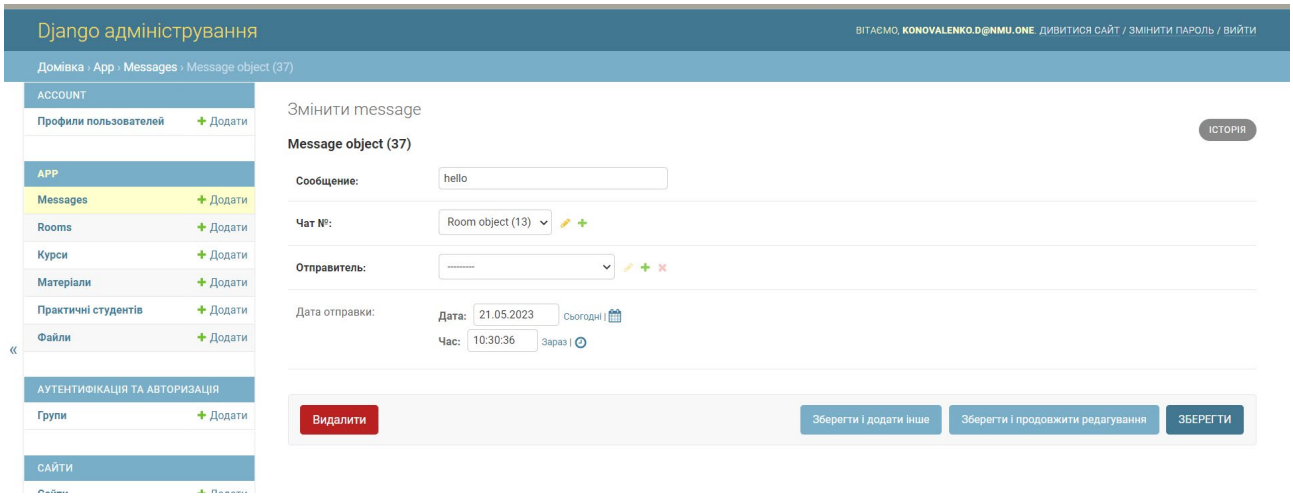


Рис. 2.48. Інформація про повідомлення

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості розробки програмного забезпечення

Вихідні дані:

1. передбачуване число операторів програми – 2370;
2. коефіцієнт складності програми – 1,4;
3. коефіцієнт корекції програми в ході її розробки – 0,2;
4. годинна заробітна плата програміста – 110 грн/год; [21]

Проаналізувавши вакансії представлені на популярній платформі для пошуку роботи Work.ua можна побачити, що середня зарплата Junior Python developer становить 500$. Курс долара [22] НБУ на момент написання записки становить 37 гривень за 1 долар, відповідно 500 * 37 = 18500 гривень на місяць. Звичайний робочий графік програміста становить 21 робочий день по 8 годин на добу. Отже погодинна ставка становить 18500 / (8 * 21) = 110 грн/год.

5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,3;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,1;
7. вартість машино-години ЕОМ – 0,87 грн/год.

Для написання кваліфікованої роботи додаткове обладнання або приміщення не знадобилося, тому для розрахунку вартості машино-години ЕОМ узято витрати електроенергії та домашнього інтернету. За даними мінфіну на момент написання роботи вартість 1 кВт/год. = 2,64 грн. [23] Домашній інтернет коштує 120 грн/міс. Ноутбук споживає 60 Вт, тому за місяць користування у робочий час вартість за електроенергію буде 60 * 2,64 * 8 * 21 / 1000 = 26,61 грн/міс. Тому вартість машино-години ЕОМ – (120 + 26,61) / 168 = 0,87 грн/міс.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може

бути розрахована на основі системи моделей з різною точністю оцінки. Трудомісткість розробки ПЗ можна розрахувати за формулою (3.1):

$$t = t_о + t_и + t_а + t_п + t_{отл} + t_д \text{, людино-годин,} \quad (3.1)$$

де $t_о$ - витрати праці на підготовку й опис поставленої задачі (приймається 50);

$t_и$ - витрати праці на дослідження алгоритму рішення задачі;

$t_а$ - витрати праці на розробку блок-схеми алгоритму;

$t_п$ - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм) (3.2):

$$Q = q \cdot C \cdot (1 + p) \quad (3.2)$$

де q - передбачуване число операторів;

C - коефіцієнт складності програми;

p - коефіцієнт кореляції програми в ході її розробки.

Після підставлення значень умовне число операторів дорівнює:

$$Q = 2370 \cdot 1{,}4 \cdot (1 + 0{,}2) = 3981{,}6$$

Витрати праці на вивчення опису задачі $t_и$ визначається з урахуванням уточнення опису і кваліфікації програміста (3.3):

$$t_и = \frac{Q*B}{(75..85)*k} \text{, людино-годин.} \quad (3.3)$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності, стаж роботи 1 рік, тому коефіцієнт кваліфікації програміста = 1,1.

Будемо вважати збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1,3).

Після підставлення значень маємо:

t_u = (3981,6 · 1,3) / (85 · 1,1) = 66,98 людино-годин.

Витрати праці на розробку алгоритму рішення задачі (3.4):

$$t_a = \frac{Q}{(20..25)*k}, \text{ людино-годин.} \quad (3.4)$$

Підставивши значення:

t_a = 3981,6 / (23 · 1,1) = 190,42 людино-годин.

Витрати на складання програми по готовій блок-схемі розраховається за формулою (3.5):

$$t_п = \frac{Q}{(20..25)*k}, \text{ людино-годин.} \quad (3.5)$$

Підставивши значення отримуємо витрати на складання програми по готовій блок-схемі:

$t_п$ = 3981,6 / (25 · 1,1) = 175,19 людино-годин.

Витрати праці на налагодження програми на ЕОМ:

–за умови автономного налагодження одного завдання (3.6):

$$t_{отл} = \frac{Q}{(4..5) \cdot k}, \text{ людино-годин.} \quad (3.6)$$

Підставивши значення:

$t_{отл}$ = 3981,6/ (5 · 1,1) = 875,95 людино-годин.

–за умови комплексного налагодження завдання(3.7):

$$t^k_{отл} = 1{,}5 \cdot t_{отл}, \text{ людино-годин.} \quad (3.7)$$

$t^k_{отл}$ = 1,5 · 875,95 = 1313,928 людино-годин.

Витрати праці на підготовку документації (3.8):

$$t_д = t_{др} + t_{до}, \text{ людино-годин,} \quad (3.8)$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису (3.9):

$$t_{др} = \frac{Q}{(15..20) \cdot k}, \text{ людино-годин.} \quad (3.9)$$

$t_{др}$ = 3981,6/ (20 · 1,1) = 218,98 людино-годин.

$T_{до}$ - трудомісткість редагування, печатки й оформлення документації (3.10):

$$t_{до} = 0,75 \cdot t_{др}$$

, людино-годин. (3.10)

$t_{до} = 0,75 \cdot 218,98 = 164,241$ людино-годин.

Виходить що витрати праці на підготовку документації:

$t_{д} = 218,99 + 164,24 = 383,23$ людино-годин.

Отже підставивши всі найдені значення у першу формулу (3.1) маємо:

$t = 50 + 66,98 + 190,42 + 175,19 + 875,95 + 383,23 = 1741,78$ людино-години.

3.2. Розрахунок витрат на створення програмного забезпечення

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ. (3.11)

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.} \quad (3.11)$$

Заробітна плата виконавців визначається за формулою (3.12):

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн,} \quad (3.12)$$

де: t - загальна трудомісткість, людино-годин;

$C_{пр}$ - середня годинна заробітна плата програміста, грн/година

Середня плата за одну годинну роботи програміста становить 110 грн., тому:

$З_{ЗП} = 1741,78 \cdot 110 = 191595,83$ грн.

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою (3.13):

$$З_{мв} = t_{отл} \cdot C_{мч} \text{ грн,} \quad (3.13)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$С_{мч}$ - вартість машино-години ЕОМ, грн/год (0,87 грн/год).

Підставивши значення:

$$З_{мв} = 875,95 \cdot 0,87 = 764,43 \text{ грн.}$$

Отже витрати на створення програмного продукту будуть складати:

$$К_{ПО} = 191595,83 + 764,43 = 192360,27 \text{ грн.}$$

Формула для розрахунку очікуваного періоду створення ПЗ (3.14):

$$T = \frac{t}{B_k \cdot F_p} \text{міс,} \qquad (3.14)$$

де B_k - число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Очікуваний період створення ПЗ:

$$T = 1741,78 / 1 \cdot 176 \approx 9 \text{ міс.}$$

Висновок: Мінімальна вартість даного програмного забезпечення дорівнює остаточним витратам на створення цього програмного продукту - 192360,27 грн. Також, згідно розрахункам, термін створення становить 9 місяців. В цей термін входить також час на переробку помилок або удосконалення завдань, у зв'язку з їх неточністю.

ВИСНОВКИ

Метою кваліфікаційної роботи зробити веб-орієнтований програмний додаток у якості програмного забезпечення для дистанційної освіти, щоб надати можливість легко та доступно навчатися та навчати. Поставлена задача завжди буде актуальною, тому що платформи ДО дозволяють з різних кутків світу отримувати знання в бажаних закладах освіти.

Додаток створено виключно для користувачів нашого університету, реєстрація та вхід у систему виконується тільки через корпоративну пошту.

Основні функціональні можливості:

−швидке та легке публікування курсів та матеріалів до них – лекції, практичні завдання та їх пояснення, встановлення термінів здачі;

−на сторінці матеріалу курсу є чат між студентом та викладачем, що дає можливість швидко зв’язатись;

−завантаження всіх оцінок на курсі в Excel-форматі, що дозволяє швидко організувати журнали з оцінками;

−здача робіт під матеріалами практики та можливість отримання коментаря через чат.

Структура програми являє собою веб-орієнтований додаток, реалізований на мовах програмування JavaScript, Python та фреймворку Django зі збереженням даних користувача в реляційній базі даних SQLite.

Під час розробки було побудовано алгоритм роботи над продуктом та спроектовано інтуїтивно зрозумілий дизайн.

В «Економічному розділі» завданням було визначити трудомісткість розробки програмного забезпечення, яке в результаті дорівнює 1741,78 людино-години, витрати на створення програмного забезпечення - 192360,27 грн., та очікуваний період створення ПЗ – 9 місяців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Distance education [Електронний ресурс]. URL: https://en.wikipedia.org/wiki/Distance_education (дата звернення: 18.05.2023).
2. Що таке дистанційна освіта? [Електронний ресурс]. URL: https://kubg.edu.ua/servisi/48-struktura/pidrozdili/ndl-informatizatsiyi-osviti/262-scho-take-distantsijna-osvita.html (дата звернення: 23.05.2023).
3. Ryan knott. What is Distance Learning? The Complete Guide [Електронний ресурс]. URL: https://www.techsmith.com/blog/distance-learning/#future (дата звернення: 31.05.2023).
4. Золотар О.О. Трубін І. О. Класифікація загроз інформаційній безпеці [Електронний ресурс]. URL: http://ippi.org.ua/sites/default/files/13zoozib.pdf (дата звернення: 02.06.2023).
5. Ada Nduka Oyom. Understanding the MVC pattern in Django [Електронний ресурс]. URL: https://medium.com/shecodeafrica/understanding-the-mvc-pattern-in-django-edda05b9f43f (дата звернення: 03.06.2023).
6. Django Book: концепція розробки MVC - Model, View, Controller. [Електронний ресурс]. URL: https://rtfm.co.ua/ru/django-book-model-razrabotki-mtc-model-view-controller/ (дата звернення: 20.05.2023).
7. Django Architecture – 3 Major Components of MVC Pattern [Електронний ресурс]. URL: https://data-flair.training/blogs/django-architecture/ (дата звернення: 17.05.2023).
8. Templates [Електронний ресурс]. URL: https://docs.djangoproject.com/en/4.2/topics/templates/#:~:text=The%20Django%20template%20language%20doesn,an%20equivalent%20of%20Jinja2%20tests (дата звернення: 26.05.2023).
9. Views In Django | Python [Електронний ресурс]. URL: https://www.geeksforgeeks.org/views-in-django-python/ (дата звернення: 18.05.2023).
10. What is Python? [Електронний ресурс]. URL: https://www.teradata.com/Glossary/What-is-Python (дата звернення: 18.05.2023).

11. HTML [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/HTML (дата звернення: 12.06.2023).

12. ЩО ТАКЕ CSS? [Електронний ресурс]. URL: https://avada-media.ua/ua/css/ (дата звернення: 12.06.2023).

13. Anthony Corbo. What Is Python? A Basic Guide, Dec. 29, 2022 [Електронний ресурс]. URL: https://builtin.com/software-engineering-perspectives/python (дата звернення: 27.05.2023).

14. Zach Paruch. What Is JavaScript & What Is It Used For? A Basic Guide to JS, Mar 21, 2023 [Електронний ресурс]. URL: https://www.semrush.com/blog/javascript/ (дата звернення: 05.06.2023).

15. Jordana A. What Is JavaScript? A Basic Introduction to JS for Beginners, Jan 31, 2023 [Електронний ресурс]. URL: https://www.hostinger.com/tutorials/what-is-javascript (дата звернення: 01.06.2023).

16. What is JavaScript? [Електронний ресурс]. URL: https://www.w3schools.com/whatis/whatis_js.asp (дата звернення: 27.05.2023).

17. Django introduction [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction (дата звернення: 08.06.2023).

18. Aruan Gupta. All You Need To Know About Django Framework, Feb 23, 2023 [Електронний ресурс]. URL: https://www.simplilearn.com/tutorials/django-tutorial/what-is-django-python (дата звернення: 27.05.2023).

19. JavaScript language overview, May 4, 2023 [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language_overview (дата звернення: 26.05.2023).

20. Eric Gazoni, Charlie Clark. Openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files, Mar 11, 2023 [Електронний ресурс]. URL: https://openpyxl.readthedocs.io/en/stable/ (дата звернення: 29.05.2023).

21. Vijay Singh Khatri. Flask vs Django: Which Python Web Framework to Use in 2023?, 28 Mar, 2023 [Електронний ресурс]. URL: https://hackr.io/blog/flask-vs-django (дата звернення: 28.05.2023).

22. PLANEKS. What Is the Difference Between Django and Flask Framework?, March 6, 2021 [Електронний ресурс]. URL: https://www.planeks.net/python-web-development-django-vs-flask-the-ultimate-dilemma/ (дата звернення: 18.05.2023).

23. Junior python developer salary [Електронний ресурс]. URL: https://www.work.ua/ru/jobs-junior+python+developer/ (дата звернення: 26.05.2023).

24. Курс долара до гривні [Електронний ресурс]. URL: https://minfin.com.ua/currency/usd/ (дата звернення: 17.05.2023).

25. Тарифи на електроенергію [Електронний ресурс]. URL: https://index.minfin.com.ua/tariff/electric/index.php/#:~:text=30.05.2023%20%D0%BD%D0%B0%20%D0%B7%D0%B0%D1%81%D0%B5%D0%B4%D0%B0%D0%BD%D0%B8%D0%B8%20%D0%9F%D1%80%D0%B0%D0%B2%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%B0,%E2%84%96%20544%20%D0%BE%D1%82%2030.05.2023 (дата звернення: 20.05.2023).

26. Скільки електроенергії споживає ноутбук [Електронний ресурс]. URL: https://geoid.org.ua/skolko-elektroenergii-potreblyaet-noutbuk#:~:text=%D0%92%20%D1%81%D1%80%D0%B5%D0%B4%D0%BD%D0%B5%D0%BC%20%D1%81%D0%BE%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5%20%D0%BD%D0%BE%D1%83%D1%82%D0%B1%D1%83%D0%BA%D0%B8%20%D1%81,%D0%BF%D0%BE%D1%82%D1%80%D0%B5%D0%B1%D0%BB%D1%8F%D1%82%D1%8C%20150%20%D0%92%D1%82%20%D0%B8%20%D0%B1%D0%BE%D0%BB%D1%8C%D1%88%D0%B5 (дата звернення: 03.06.2023).

27. Методичні рекомендації до виконання кваліфікаційних робіт здобувачів першого рівня вищої освіти спеціальності 121 Інженерія програмного забезпечення / В.В. Спірінцев, О.С. Шевцова, І.М. Удовик; Д : НТУ «Дніпровська політехніка», 2022. – 60 с.

ДОДАТОК А

КОД ПРОГРАМИ

Account/models.py

```
from distutils.command.upload import upload
import re
from django.db import models
from django.contrib.auth.models import AbstractBaseUser, BaseUserManager
import uuid
from django.contrib.auth.models import PermissionsMixin
from app.models import Course, Material

class MyAccountManager(BaseUserManager):
    def create_user(self, email, password=None, *args, **kwargs):
        if not email:
            raise ValueError('Введите email')

        user = self.model(
            email=self.normalize_email(email),
            **kwargs
        )

        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_superuser(self, email, password):
        user = self.create_user(
            email=self.normalize_email(email),
            password=password,
        )
        user.is_admin = True
        user.is_staff = True
        user.is_superuser = True
        user.save(using=self._db)
        return user


def validate_email(email):
    if not re.search(r"^[A-Za-z0-9_!#$%&'*+\/=?`{|}~^.-]+@nmu.one$", email):
        print(f"The email address {email} is not valid")
        return False
class Account(AbstractBaseUser, PermissionsMixin):
    image                                   = models.ImageField(upload_to='files/', null=True,
blank=True)
    email                                   = models.EmailField(verbose_name="email",
max_length=60, unique=True, validators=[validate_email])
    phone_number                    = models.CharField(max_length=13, verbose_name='Номер
телефона', blank=True, unique=True, null=True)
    first_name                      = models.CharField(max_length=30, default='', null=True,
verbose_name='Ім’я')
    last_name                       = models.CharField(max_length=30, default='', null=True,
verbose_name=' Прізвище')
    patronymic                      = models.CharField(max_length=30, default='', null=True,
verbose_name='По батькові')
    group                           = models.CharField(max_length=10, default='', null=True,
verbose_name='Група')
```

```
    chat_id = models.BigIntegerField(null=True, blank=True, verbose_name='chat id', editable=False)

    date_joined                 = models.DateTimeField(verbose_name='Дата реєстрації',
auto_now_add=True)
    is_admin                    = models.BooleanField(default=False)
    is_staff                    = models.BooleanField(default=False)
    is_superuser                = models.BooleanField(default=False)
    unique_code                 = models.UUIDField(default=uuid.uuid4, unique=True,
editable=False)

    is_lecturer                 = models.BooleanField(default=False)

    grades              = models.JSONField(verbose_name='grades', default=dict, blank=True)

    USERNAME_FIELD = 'email'
    objects = MyAccountManager()

    class Meta:
        verbose_name_plural='Профілі користувачів'
        verbose_name= 'Профіль користувача'
        ordering=['email']

    def __str__(self):
        return self.email

    def get_first_part_email(self):
        return self.email.split('@')[0]

    def get_courses(self):
        if self.is_lecturer:
            return Course.objects.filter(author=self)
        else:
            return Course.objects.filter(students__in=[self])

    def get_profile_grades(self):
        labs = Material.objects.filter(course__students__in=[self], what_type='practice')
        if labs:
            students_info = []
            for lab in labs:
                if self.grades.get(f'course_{lab.course.id}', {}).get(f'lab_{lab.id}', ''):
                    grades_info_list = {}
                    grades_info_list['course_name'] = lab.course.name
                    grades_info_list['material_name'] = lab.name
                    grades_info_list['grade'] = self.grades.get(f'course_{lab.course.id}', {}).get(f'lab_{lab.id}', '')

                    students_info.append(grades_info_list)
            return students_info
        else:
            return []


    # def save(self, *args, **kwargs):
    #     if not self.uuid: self.uuid = uuid.uuid4()
    #     return super(Material, self).save(*args, **kwargs)
```

Account/views.py

```
from django.shortcuts import redirect, render
from account.forms import ProfileEditForm

from .models import Account
from django.contrib.auth import authenticate, login, logout
from django.contrib.sites.shortcuts import get_current_site
from django.utils.encoding import force_bytes, force_str, DjangoUnicodeDecodeError
from django.utils.http import urlsafe_base64_encode, urlsafe_base64_decode
from django.urls import reverse
from .utils import token_generator
from django.conf import settings
from django.core.mail import EmailMultiAlternatives
from django.template.loader import get_template
import threading
from django.views.generic import View
from django.contrib.auth.tokens import PasswordResetTokenGenerator
from django.contrib.auth.decorators import login_required

def user_login(request):
    if request.user.is_authenticated:
        return redirect('app:index')
    else:
        if request.method == 'POST':
            email = request.POST.get('email')
            password = request.POST.get('password')

            try:
                # email
                user = authenticate(email=email, password=password)
            except:
                # номер телефону
                user = authenticate(email=Account.objects.get(phone_number=email).email, password=password)

            if user is not None:
                login(request, user)
                if 'next' in request.POST: return redirect(request.POST.get('next'))
                return redirect('account:profile')
            else:
                return render(request, 'account/login.html')
        else:
            return render(request, 'account/login.html')

def user_logout(request):
    logout(request)
    return redirect('app:index')



def registration(request):
    if request.method == 'POST':
        email = request.POST.get('email')
        password = request.POST.get('password')

        if Account.objects.filter(email=email).exists():
            try: user = authenticate(email=email, password=password)
            except: pass
```

```
            if user is not None:
                login(request, user)
                if 'next' in request.POST: return redirect(request.POST.get('next'))
                return redirect('app:profile')

        user_data = {
            'password': password,
            'email': email.lower(),
        }

        if request.POST.get('is_lecturer', False):
            user_data['is_lecturer'] = True

        if group := request.POST.get('group', False):
            user_data['group'] = group

        user = Account.objects.create_user(**user_data)
        login(request, user)


        return render(request, 'account/registration_success.html', {'email':email})

    else:
        return render(request, 'account/registration.html')



# view для того, щоб швидше надсилати email
class EmailThreading(threading.Thread):
    def __init__(self, email_message):
        self.email_message = email_message
        threading.Thread.__init__(self)

    def run(self):
        self.email_message.send()


class RequestResetEmailView(View):
    def get(self, request):
        return render(request, 'account/reset_email.html')

    def post(self, request):
        email = request.POST.get('email', None)

        if email is None or not Account.objects.filter(email=email).exists():
            return render(request, 'account/reset_email.html')

        user = Account.objects.filter(email=email)
        if user.exists():
            uidb64 =urlsafe_base64_encode(force_bytes(user[0].pk))

            domain = get_current_site(request).domain
            link = reverse('account:set_new_pswrd',
kwargs={'uidb64':uidb64,'token':token_generator.make_token(user[0])})

            renew_password_url='http://' + domain + link
```

```
            mail_title = " Оновлення пароля"
            variables = {
                'activate_url': renew_password_url,
            }
            html = get_template('email/email_reset_password.html').render(variables)
            text = f"

            msg = EmailMultiAlternatives(
                mail_title,
                text,
                settings.EMAIL_HOST_USER,
                [email])
            msg.attach_alternative(html, "text/html")
            EmailThreading(msg).start() # ми швидше надсилаємо email

            return render(request, 'account/reset_email_success.html')
        else:
            return redirect('account:registration')

class SetNewPswrdView(View):
    def get(self, request, uidb64, token):
        context = {
            'uidb64':uidb64,
            'token':token,
        }

        try:
            user_id = force_str(urlsafe_base64_decode(uidb64))
            user = Account.objects.get(pk=user_id)

            if PasswordResetTokenGenerator().check_token(user, token):
                return render(request, 'account/reset_email.html') # Пароль клікають 2-ий раз, так не можна.
Відшлемо на повторне відправлення
        except DjangoUnicodeDecodeError as identifier:
            return render(request, 'account/reset_email.html')
        return render(request, 'account/set_new_pswrd.html', context)

    def post(self, request, uidb64, token):
        context = {
            'uidb64':uidb64,
            'token':token,

        }

        password = request.POST.get('password', None)
        if password is None or len(password) < 4 or len(password) > 20:
            return render(request, 'account/set_new_pswrd.html', context)

        try:
            user_id = force_str(urlsafe_base64_decode(uidb64))

            user = Account.objects.get(pk=user_id)
            user.set_password(password)
            user.save()

            return redirect('/login/')
```

```
        except DjangoUnicodeDecodeError as identifier:
            return render(request, 'account/set_new_pswrd.html', context)


@login_required(login_url='/login/')
def profile(request):
    account = Account.objects.get(pk=request.user.pk)
    if request.method == 'POST':
        form = ProfileEditForm(request.POST, request.FILES, instance=account)
        if form.is_valid(): form.save()
        else : print(form.errors)
        return redirect('account:profile')


    form = ProfileEditForm(instance=account)
    return render(request, 'account/profile.html', {'form':form, 'account':account,})
```

Course/models.py

```
from datetime import datetime
from django.db import models
from pytils.translit import slugify
import uuid
from django.utils.translation import gettext_lazy as _
from django.utils import timezone

class Course(models.Model):
    name = models.CharField(max_length=100, verbose_name='Назва')
    description = models.TextField(max_length=2000, null=True, blank=True, verbose_name='Опис')
    slug = models.SlugField(max_length=200, unique = True,verbose_name='Посилання')
    published = models.DateTimeField(auto_now_add=True, db_index=True,
verbose_name='Опубліковано')
    author = models.ForeignKey('account.Account', related_name='author', null=True,
on_delete=models.CASCADE, verbose_name='Автор', blank=True)
    students = models.ManyToManyField('account.Account', related_name='students', default=None,
blank=True, verbose_name = "Зараховані студенти")

    RUBRIC_CHOICES = (
        ('fit', 'ФІТ'),
        ('etf', 'ЕТФ'),
    )
    # to display runbric in template, should: get_<fieldName>_display(get_rubric_display)
    rubric = models.CharField(max_length=10, choices=RUBRIC_CHOICES, default='draft'
,verbose_name='Рубрика (курс якого факультету)')

    class Meta:
        verbose_name = _('Курс')
        verbose_name_plural = _('Курси')

    def save(self,  *args, **kwargs):
        if not self.pk:
            if Course.objects.filter(name=self.name).exists():
                self.slug = slugify(self.name) + "-" + str(uuid.uuid4())
            else:
                self.slug = slugify(self.name)
            super(Course, self).save(*args, **kwargs)
        super(Course, self).save(*args, **kwargs)
```

```
    def __str__(self):
        return str(self.name)


class Material(models.Model):
    name = models.CharField(max_length=100, verbose_name='Назва')
    description = models.TextField(max_length=1000, null=True, blank=True, verbose_name='Опис')
    published = models.DateTimeField(auto_now_add=True, db_index=True,
verbose_name='"Опубліковано')
    last_due_date = models.DateTimeField(verbose_name=' Остання дата виконання', null=True,
blank=True)
    course = models.ForeignKey(Course, on_delete=models.CASCADE)
    slug = models.SlugField(null=True, default=str(uuid.uuid4()))
    TYPE_CHOICES = (
        ('lecture', _('Лекція')),
        ('practice', _('Практика')),
        ('others', _('Інше')),
    )
    what_type = models.CharField(max_length=10, choices=TYPE_CHOICES, default='lecture'
,verbose_name='Тип матеріалу')

    class Meta:
        verbose_name = _('Матеріал')
        verbose_name_plural = _('Матеріали')

    def get_files(self):
        return File.objects.filter(material=self)

    # def save(self, *args, **kwargs):
    #     if not self.uuid: self.uuid = uuid.uuid4()
    #     return super(Material, self).save(*args, **kwargs)


class File(models.Model):
    file = models.FileField(upload_to='files')
    name = models.CharField(max_length=100, verbose_name='Назва')
    material = models.ForeignKey(Material, on_delete=models.CASCADE, null=True)

    class Meta:
        verbose_name = _('Файл')
        verbose_name_plural = _('Файли')


class Lab(models.Model):
    file = models.FileField(upload_to='files')
    material = models.ForeignKey(Material, on_delete=models.CASCADE, null=True)
    author = models.ForeignKey('account.Account', on_delete=models.CASCADE, null=True)

    class Meta:
        verbose_name = _('Практична студента')
        verbose_name_plural = _('Практичні студенти')




class Room(models.Model):
```

```
    name = models.CharField(max_length = 500, verbose_name='Чат')
    material = models.ForeignKey(Material, on_delete=models.CASCADE, verbose_name='Практична робота')
    participants = models.ManyToManyField('account.Account', related_name="participants", default=None, blank=True, verbose_name = "Учасники чату")

    def get_messages(self):
        return Message.objects.filter(room=self)

    def last_msg_sender(self):
        return Message.objects.filter(room=self).last().sender.email

class Message(models.Model):
    value = models.CharField(max_length = 10000, verbose_name='Повідомлення')
    room = models.ForeignKey(Room, on_delete=models.CASCADE, verbose_name='Чат №', null=True)
    date = models.DateTimeField(default=timezone.now, blank = True, verbose_name='Дата відправлення') # auto_now_add=True
    sender = models.ForeignKey('account.Account', on_delete=models.SET_NULL, verbose_name='Відправник', null=True)

    class Meta:
        get_latest_by = 'id'
```

Course/views.py

```
import functools
from django.shortcuts import render, redirect
from .models import Course
from django.contrib.admin.views import decorators
from django.views import View
from .forms import *
from django.utils.decorators import method_decorator
from googletrans import Translator # googletrans==3.1.0a0
from django.shortcuts import get_object_or_404
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt
from django.contrib.auth.decorators import login_required
from account.models import Account
import xlsxwriter
import io
from django.http import HttpResponse, HttpResponseRedirect
from django.db.models import Q, Count


def lecturer_required(view_func): # перевизначимо декоратор
    def _checklogin(request, *args, **kwargs):
        if request.user.is_active and request.user.is_lecturer or request.user.is_superuser:
            return view_func(request, *args, **kwargs)
        else:
            return redirect('app:index')
    return functools.wraps(view_func)(_checklogin)

def translate_str(string):
    this_lang = Translator().detect(text=string, confidence=1).lang
    dest_lang = 'uk' if this_lang == 'en' else 'en'
    string_trans = ''
    try:
        translator = Translator()
        string_trans = translator.translate(string, dest=dest_lang).text
```

```
    except Exception as e: print(e)

    return {'uk':string, 'en':string_trans} if this_lang == 'uk' else {'uk':string_trans, 'en':string}



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

def all_courses(request):
    if search := request.GET.get('search'):
        courses = Course.objects.filter(name__icontains = search.lower())
    else:
        courses = Course.objects.all()
    return render(request, 'course/all_courses.html', {'courses':courses})

class AddCourseView(View):
    @method_decorator(lecturer_required)
    def get(self, request, *args, **kwargs):
        form = CourseForm()
        return render(request, 'course/add_course.html', {'form':form})

    @method_decorator(lecturer_required)
    def post(self, request, *args, **kwargs):
        form = CourseForm(request.POST)
        if form.is_valid():
            course = form.save(commit=False)
            course.author = request.user
            trans_name = translate_str(course.name)
            trans_description = translate_str(course.description)
            course.name_uk = trans_name['uk']
            course.name_en = trans_name['en']
            course.description_uk = trans_description['uk']
            course.description_en = trans_description['en']
            course.save()
        else:
            print(form.errors)
            return redirect('app:add_course')
        return redirect('app:all_courses')


def course(request, slug):
    course = get_object_or_404(Course, slug=slug)
    materials = Material.objects.filter(course=course)
    context = {
        'course':course,
        'materials': materials,
    }
    return render(request, 'course/course.html', context)

@login_required
@csrf_exempt
def course_enroll(request, course_slug):
    student = get_object_or_404(Account, email=request.user.email)
    course = get_object_or_404(Course, slug=course_slug)
    labs = Material.objects.filter(course=course)
```

```
    course.students.add(student)
    grades = {}
    # for lab in labs: grades[f'lab_{lab.id}'] = ''
    student.grades[f'course_{course.id}'] = grades
    student.save()

    return redirect('app:course', slug=course_slug)


class AddMaterialView(View):
    @method_decorator(lecturer_required)
    def get(self, request, *args, **kwargs):
        form = MaterialForm(request.user)
        return render(request, 'course/add_material.html', {'form':form})

    @method_decorator(lecturer_required)
    def post(self, request, *args, **kwargs):
        form = MaterialForm(request.user, request.POST)
        if form.is_valid():
            material = form.save(commit=False)
            trans_name = translate_str(material.name)
            trans_description = translate_str(material.description)
            material.name_uk = trans_name['uk']
            material.name_en = trans_name['en']
            material.description_uk = trans_description['uk']
            material.description_en = trans_description['en']
            material.save()

            file_names = request.POST.getlist('uploadedFileName')
            for index, _id in enumerate(request.POST.getlist('uploadedFileId')):
                File.objects.filter(id=_id).update(name=file_names[index], material=material)

            if request.POST.get('from', None) and request.POST.get('from', None) != '': return
redirect('app:course', slug=Course.objects.get(id=request.POST.get('from')).slug)
        else:
            print(form.errors)
            return redirect('app:add_material')
        return redirect('app:all_courses')

@csrf_exempt
def add_file(request):
    files = request.FILES.getlist('files')
    if files:
        uploaded_files = {}
        for val in files:
            uploaded_file = File.objects.create(file=val)
            uploaded_files[uploaded_file.id] = str(uploaded_file.file.name.split('/')[1])

        return JsonResponse(data={
            'uploaded_files':uploaded_files,
        })
    return JsonResponse({}, status=500)


class StudentsView(View):
    @method_decorator(lecturer_required)
```

```
    def get(self, request, slug, *args, **kwargs):
        course = get_object_or_404(Course, slug=slug)
        students = course.students.all()
        return render(request, 'course/students.html', {'students':students, 'course':course})

    @csrf_exempt
    @method_decorator(lecturer_required)
    def post(self, request, *args, **kwargs):
        student = request.POST.get('student', None)
        course = request.POST.get('course', None)
        if student and course:
            course = get_object_or_404(Course, slug=course)
            student = get_object_or_404(Account, email=student)
            course.students.remove(student)

            return JsonResponse({}, status=200)
        return JsonResponse({}, status=500)

class GradesView(View):
    @method_decorator(lecturer_required)
    def get(self, request, course_slug, *args, **kwargs):
        course = get_object_or_404(Course, slug=course_slug)
        students = course.students.all().order_by('group')
        labs = Material.objects.filter(course=course, what_type='practice')

        students_info = []
        for student in students:
            student_info = {}
            student_info['id'] = student.id
            student_info['first_name'] = student.first_name
            student_info['last_name'] = student.last_name
            student_info['patronymic'] = student.patronymic
            student_info['group'] = student.group
            student_info['grades'] = {}
            for lab in labs:
                grade = student.grades.get(f'course_{course.id}', {}).get(f'lab_{lab.id}', '')
                student_info['grades'][f'{lab.id}'] = grade

            students_info.append(student_info)

        return render(request, 'course/grades.html', {'students_info':students_info, 'course':course, 'labs':labs})

    @method_decorator(lecturer_required)
    def post(self, request, *args, **kwargs):
        grade = request.POST.get('grade', None)
        course = request.POST.get('course', None)
        field = request.POST.get('field', None)
        if grade and course and field:
            course = get_object_or_404(Course, slug=course)
            student = get_object_or_404(Account, id=field.split('-')[0])
            lab = field.split('-')[1]
            student.grades[f'course_{course.id}'][f'lab_{lab}'] = grade
            student.save()

            return JsonResponse({}, status=200)
        return JsonResponse({}, status=500)
```

```
def get_grades_excel_data(course):
    table = []
    labs = Material.objects.filter(course=course, what_type='practice')

    # Firstly append row with labs
    row = []
    row.append('ПІБ студента')
    row.append('Група')
    row.extend([lab.name for lab in labs])
    table.append(row)

    for student in course.students.all().order_by('group'):
        row = []
        student_name = f'{student.last_name} {student.first_name} {student.patronymic}'
        student_group = f'{student.group}'
        row.append(student_name)
        row.append(student_group)
        for lab in labs:
            grade = student.grades.get(f'course_{course.id}', {}).get(f'lab_{lab.id}', '')
            try: grade = int(grade)
            except: pass
            row.append(grade)
        table.append(row)

    return table


# https://xlsxwriter.readthedocs.io/example_django_simple.html
def export_excel(request, course_slug):
    course = get_object_or_404(Course, slug=course_slug)
    if request.user != course.author: JsonResponse({}, status=500)
    output = io.BytesIO()
    workbook = xlsxwriter.Workbook(output, {'in_memory': True})
    worksheet = workbook.add_worksheet()

    data = get_grades_excel_data(course)
    for row_num, columns in enumerate(data):
        for col_num, cell_data in enumerate(columns):
            worksheet.write(row_num, col_num, cell_data)

    workbook.close()
    output.seek(0)

    filename = 'file.xlsx'
    response = HttpResponse(
        output,
        content_type='application/vnd.openxmlformats-officedocument.spreadsheetml.sheet'
    )
    response['Content-Disposition'] = 'attachment; filename=%s' % filename

    return response


class MaterialView(View):
    @method_decorator(login_required)
    def get(self, request, slug, *args, **kwargs):
```

```
        material = get_object_or_404(Material, slug=slug)
        labs = Lab.objects.filter(material=material, author=request.user)
        if request.user == material.course.author:
            room = Room.objects.filter(material=material)
        else:
            # if Room.objects.filter(participants__in=[request.user, material.course.author],
material=material).exists():
            #     room = Room.objects.filter(participants__in=[request.user, material.course.author],
material=material)[0]
            # if Room.objects.filter(participants__in=[request.user, material.course.author],
material=material).exists():
            #     room = Room.objects.filter(participants__in=[request.user, material.course.author],
material=material)[0]
            if Room.objects.filter(participants__in=[request.user, material.course.author],
material=material).annotate(c=Count('participants')).filter(c=2).exists():
                room = Room.objects.filter(participants__in=[request.user, material.course.author],
material=material).annotate(c=Count('participants')).filter(c=2)[0]
            else: room = None


        return render(request, 'course/material.html', {'material':material, 'labs':labs, 'room':room})

    @method_decorator(login_required)
    def post(self, request, *args, **kwargs):
        if request.POST.get('delete_lab_id'):
            if request.user == Account.objects.get(id=request.POST.get('this_user')) \
                and Lab.objects.filter(id=request.POST.get('delete_lab_id'), author=request.user).exists():
                Lab.objects.filter(id=request.POST.get('delete_lab_id')).delete()
                return JsonResponse({}, status=200)
            return JsonResponse({}, status=500)


        try:
            for lab in request.POST.getlist('labs'):
                Lab.objects.create(
                    file=lab,
                    author=request.user,
                    material=Material.objects.get(slug=request.POST.get('material_slug'))
                )
        except Exception as e: print(e)
        finally: return HttpResponseRedirect(request.path_info)



class ChatView(View):
    def get(self, request): pass

    @method_decorator(login_required)
    def post(self, request, *args, **kwargs):
        sender = request.POST.get('sender')
        value = request.POST.get('value')
        material = request.POST.get('material')
        room = request.POST.get('room', None)

        if not room or room == '':
            room = Room.objects.create(
                material = Material.objects.get(id=material)
```

```
            )
            room.participants.add(request.user, Material.objects.get(id=material).course.author)
            room.save()
            room = room.id

        Message.objects.create(
            room = Room.objects.get(id=room),
            value=value,
            sender = Account.objects.get(id=sender)
        )
        return JsonResponse({}, status=200)



class WorkView(View):
    @method_decorator(lecturer_required)
    def get(self, request, slug):
        material = get_object_or_404(Material, slug=slug)
        labs = []
        for student in material.course.students.all():
            if Lab.objects.filter(material=material, author=student).count() > 0:
                labs.append(Lab.objects.filter(material=material, author=student))
        print(labs)
        return render(request, 'course/works.html', {'all_labs':labs, 'material':material})

    @method_decorator(lecturer_required)
    def post(self, request, *args, **kwargs):
        sender = request.POST.get('sender')
        value = request.POST.get('value')
        material = request.POST.get('material')
        room = request.POST.get('room', None)

        if not room or room == '':
            room = Room.objects.create(
                material = Material.objects.get(id=material)
            )
            room.participants.add(request.user, Material.objects.get(id=material).course.author)
            room.save()
            room = room.id

        Message.objects.create(
            room = Room.objects.get(id=room),
            value=value,
            sender = Account.objects.get(id=sender)
        )
        return JsonResponse({}, status=200)

class EditCourseView(View):
    @method_decorator(lecturer_required)
    def get(self, request, slug, *args, **kwargs):
        course = get_object_or_404(Course, slug=slug)
        form = CourseForm(instance=course)
        return render(request, 'course/edit_course.html', {'form':form, 'course':course})

    @method_decorator(lecturer_required)
    def post(self, request, slug, *args, **kwargs):
        course = get_object_or_404(Course, slug=slug)
```

```
        form = CourseForm(request.POST, instance=course)
        if form.is_valid():
            course = form.save(commit=False)
            course.author = request.user
            trans_name = translate_str(course.name)
            trans_description = translate_str(course.description)
            course.name_uk = trans_name['uk']
            course.name_en = trans_name['en']
            course.description_uk = trans_description['uk']
            course.description_en = trans_description['en']
            course.save()
        else:
            print(form.errors)
            return redirect('app:add_course')
        return redirect('app:course', slug=slug)


class EditMaterialView(View):
    @method_decorator(lecturer_required)
    def get(self, request, slug, *args, **kwargs):
        material = get_object_or_404(Material, slug=slug)
        files = File.objects.filter(material=material)
        form = MaterialForm(request.user, instance=material)
        return render(request, 'course/edit_material.html', {'form':form, 'material':material, 'files':files})

    @method_decorator(lecturer_required)
    def post(self, request, slug, *args, **kwargs):
        material = get_object_or_404(Material, slug=slug)
        form = MaterialForm(request.user, request.POST, instance=material)
        if form.is_valid():
            material = form.save(commit=False)
            trans_name = translate_str(material.name)
            trans_description = translate_str(material.description)
            material.name_uk = trans_name['uk']
            material.name_en = trans_name['en']
            material.description_uk = trans_description['uk']
            material.description_en = trans_description['en']
            material.save()

            file_names = request.POST.getlist('uploadedFileName')
            for index, _id in enumerate(request.POST.getlist('uploadedFileId')):
                File.objects.filter(id=_id).update(name=file_names[index], material=material)

            if request.POST.get('from', None) and request.POST.get('from', None) != '': return
redirect('app:course', slug=Course.objects.get(id=request.POST.get('from')).slug)
        else:
            print(form.errors)
            return redirect('app:add_material')
        return redirect('app:all_courses')


def delete_material_id(request):
    if request.POST.get('delete_material_id'):
        File.objects.filter(id=request.POST.get('delete_material_id')).delete()
        return JsonResponse({}, status=200)
    return JsonResponse({}, status=500)
```

app\templates\course\course.html

```
{% extends 'app/base.html' %}
{% load static %}
{% load i18n %}


{% block title %}Easy Study{% endblock %}
{% block styles %}
  <link rel="stylesheet" href="{% static 'css/course.css' %}">
{% endblock %}


{% block content %}
  {% if request.user.is_authenticated %}
    {% if course.author == request.user %}
      <div style="display: flex; justify-content: end;">
        <a class="course_add_material" href="{% url 'app:edit_course' course.slug %}">{% translate
'Змінити курс' %}</a>
        <a class="course_add_material" href="{% url 'app:add_material' %}?from={{course.id}}">{%
translate 'Додати матеріал' %} +</a>
        <a class="course_add_material" href="{% url 'app:students' course.slug %}">{% translate
'Студенти на курсі' %} {{course.students.count}}</a>
      </div>
    {% endif %}
  {% endif %}
  <h1 class="center">{{course.name}}</h1>
  {% if request.user.is_authenticated %}
    {% if course.author != request.user %}
      {% if request.user not in course.students.all %}
        <h3 style="text-align: center;">
          <a class="course_add_material" href="{% url 'app:course_enroll' course.slug %}">{% translate
'Записатися на курс' %} ➞</a>
        </h3>
      {% endif %}
    {% endif %}

    <div class="course_desc"><strong>{% translate 'Опис курсу' %}</strong>:
{{course.description}}</div>
    <div class="course_body">
      {% for material in materials %}
        <a href="{% url 'app:material' material.slug %}" class="material">
          <div class="material_name">
            {{material.name}}
            <span
class="material_{{material.what_type}}">{{material.get_what_type_display}}</span>
          </div>
          <div class="material_description">{{material.description}}</div>
        </a>
      {% empty %}
        <h3 class="center">{% translate 'На даний момент матеріалів немає. Незабаром викладач
додасть їх.' %}</h3>
      {% endfor %}
    </div>
  {% endif %}
{% endblock %}
```

Весь код додається на диску.

ДОДАТОК Б

ВІДГУК

керівника економічного розділу

на кваліфікаційну роботу бакалавра

на тему:

" Розробка програмного забезпечення для дистанційної освіти на основі мов Python/JavaScript та фреймворку Django "

студентки групи 121-19-1 Андрющенко Кристини Сергіївни

Керівник економічного розділу доц. каф. ПЕП та ПУ, к.е.н **Л.В. Касьяненко**

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Пояснювальні документи	
Андрющенко.К.С.диплом.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Андрющенко.К.С.диплом.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Андрющенко.К.С.диплом.rar	Архів. Містить коди програми і скомпільовану програму
Презентація	
Андрющенко.К.С.диплом.ppt	Презентація кваліфікаційної роботи