

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Освітній портал»

Виконав студент групи ПЗк-21-1
СРЕМЕНКО Артем Ігорович

Керівник роботи:
ЛЕВЧЕНКО Антон Юрійович

Консультант:
ЄФРЕМОВ Юрій Миколайович

Рецензент:
ЛОБАНЧИКОВА Надія Миколаївна

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ЄРЕМЕНКО** Артем Ігорович

Керівник роботи: **ЛЕВЧЕНКО** Антон Юрійович

Тема роботи: **«Освітній портал»**,

затверджена Наказом закладу вищої освіти від **«23» лютого 2024 р. №74с**

Вихідні дані для роботи: **розробка освітнього порталу**.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів,
що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Єфремов Ю.М., Левченко А.Ю.	01.03.2024р.	01.03.2024р.
2	Єфремов Ю.М., Левченко А.Ю.	03.04.2024р.	03.04.2024р.
3	Єфремов Ю.М., Левченко А.Ю.	02.05.2024р.	02.05.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота складається з програмної реалізації освітнього порталу, а також пояснювальної записки. Пояснювальна записка до дипломної роботи містить 49 сторінок, 32 ілюстрації, 10 таблиць, 4 додатки та 20 літературних джерел.

Метою роботи є дослідження та розробка системи освітнього порталу, яка надасть користувачам прозорий та зручний інтерфейс для керування навчальним процесом.

В роботі описано процес аналізу технічного завдання та вимог системи, оглянуто існуюче програмне забезпечення за тематикою проекту. Також описано процес розробки бази даних та реалізації системи освітнього порталу. Далі зазначено порядок встановлення та налаштування системи, приклади тестування та опис інтерфейсу додатку.

КЛЮЧОВІ СЛОВА: ВЕБСЕРВЕР, НАВЧАННЯ, ОСВІТА, ОСВІТНІЙ ПРОЦЕС, FLASK, JINJA2.

					ІПЗ.КР.Б – 121 – 24 – ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Єременко А.І.			«Освітній портал» Пояснювальна записка			Літ.	Арк.	Аркушів	
Керівник		Левченко А.Ю.								3	49
								Житомирська політехніка, група ІПЗк-21-1			
Рецензент		Лобанчикова Н.М.									
Зав. каф.		Вакалюк Т.А.									

ABSTRACT

The graduation qualification thesis consists of the software implementation of the educational portal, as well as an explanatory note. The explanatory note to the thesis contains 49 pages, 32 illustrations, 10 tables, 4 appendices and 20 literary sources.

The purpose of the work is to research and develop an educational portal system that will provide users with a transparent and convenient interface for managing the educational process.

The work describes the process of analyzing the technical task and system requirements, and reviews the existing software on the subject of the project. The process of database development and implementation of the educational portal system is also described. The procedure for installing and setting up the system, testing examples and a description of the application interface are listed below.

KEY WORDS: WEBSERVER, LEARNING, EDUCATION, EDUCATIONAL PROCESS, FLASK, JINJA2.

		Єременко А.І.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Левченко А.Ю.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ ВИМОГ ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ WEB-ОРІЄНТОВАНОЇ СИСТЕМИ ОСВІТНЬОГО ПОРТАЛУ	8
1.1 Технічне завдання на розробку системи.....	8
1.2 Аналіз існуючого програмного забезпечення за тематикою проекту	11
1.3 Аналіз вимог системи та користувачів освітнього порталу	15
1.4 Обґрунтування вибору інструментальних засобів та архітектури системи освітнього порталу	17
Висновки до першого розділу.....	20
РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ОСВІТНЬОГО ПОРТАЛУ ..	21
2.1 Опис акторів та алгоритмів роботи.....	21
2.2 Розробка бази даних	24
2.3 Комунікації і послідовність взаємодії об'єктів системи.....	28
2.4 Реалізація системи освітнього порталу	31
Висновки до другого розділу	34
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ОСВІТНЬОГО ПОРТАЛУ	35
3.1 Порядок встановлення та налаштування системи.....	35
3.2 Тестування системи освітнього порталу	37
3.3 Опис інтерфейсу програмного продукту.....	39
Висновки до третього розділу	46
ВИСНОВКИ	47
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	48

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – База даних;

ПЗ – Програмне забезпечення;

СКБД – Система керування базами даних;

MVC – схема розділення даних додатків і керуючої логіки на три окремі компоненти Model-View-Controller.

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

У сучасному світі, де комплексність проблем зростає і ресурси є обмеженими, ефективне управління різними процесами освіти стає важливою задачею для багатьох навчальних закладів. Одним з основних інструментів, що допомагають вирішувати такі задачі, є інформаційні системи, які дозволяють організувати та покращити навчальні процеси. Впровадження таких систем є актуальним способом поліпшення навчання у освітніх закладах.

Мета цієї роботи полягає у розробці Web-орієнтованої системи освітнього порталу.

Встановлена мета обумовлює наступні завдання:

- Розробка технічного завдання на розробку системи.
- Аналіз існуючого програмного забезпечення.
- Аналіз вимог системи та користувача.
- Проектування освітнього порталу.
- Реалізація системи.
- Опис інтерфейсу.
- Тестування платформи.
- Налаштування проекту для встановлення.

Об'єктом дослідження є WEB-орієнтована система освітнього порталу, що дозволяє покращити навчальний процес та надає можливості для дистанційного навчання.

Предметом дослідження є використання Web-орієнтованих технологій для проектування та розробки системи освітнього порталу.

Апробація. Було проведено дослідження та порівняння процесу навчання штучного інтелекту з процесом навчання людей [1].

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ВИМОГ ТА КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ WEB-ОРІЄНТОВАНОЇ СИСТЕМИ ОСВІТНЬОГО ПОРТАЛУ

1.1 Технічне завдання на розробку системи

Назва системи – WEB-орієнтована система освітнього порталу.

Мета – розробка та впровадження системи освітнього порталу, яка надає інструменти для поліпшення та автоматизації навчального процесу. Дана система може бути застосована в умовах дистанційного навчання для покращення навчання у коледжах, університетах та школах.

Актуальність – використання системи освітнього порталу дозволить викладачам викладати навчальний матеріал у зручному середовищі, поширювати його між студентами та перевіряти результати виконаних завдань. Студенти, у свою чергу, зможуть швидко відшукати необхідний матеріал, подивитися на оцінки та здати виконану роботу.

Технічні вимоги:

- Система повинна бути здатна підтримувати мінімум 300 одночасно працюючих користувачів, пов'язаних із загальною базою даних.
- Час відгуку для типових задач - не більше 5 секунд, для складних завдань - не більше 20 секунд.
- Система повинна відповідати всім стандартам інтерфейсу користувача Microsoft® Windows® або ОС UNIX та сучасним нормам проектування WEB - додатків.
- Інтерфейс додатку повинен бути представлений у вигляді WEB – сайту, сервер повинен обробляти запити, які надходять від користувачів, та відображати їх на спеціальних сторінках.

Функціональні вимоги.

Авторизація та аутентифікація користувачів в системі.

В системі повинен бути представлений довідник користувачів. Повинна бути можливість реєстрації користувача і призначення користувачеві ролі.

Ведення рейтингу студентів.

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

Викладач повинен мати можливість вести рейтинг студентів виставляючи бали за здані роботи. Студенти перейшовши до сторінки курсу зможуть завантажити звіт до певного розділу курсу, після чого викладач за допомогою спеціальної сторінки зможе вивантажити його, перевірити та виставити оцінку. Студенти зможуть переглядати свої оцінки за обраними курсами.

Ведення довідника курсів та студентів.

В системі повинні бути представлені засоби управління курсами та списками студентів. Курси повинні бути поділені на розділи, кожен з яких має назву, опис та прикріплений файл. При редагування курсу можна обрати тип розділу, або лекція або практична, та додати навчальний матеріал. Курси також можуть бути закритими або відкритими. Запрошення студентів до закритого курсу повинно відбуватися за групами. Можливість встановлювати дату закінчення курсу та перегляд архіву оцінок.

Застосовність:

1. Зручність використання:

Інтерфейс повинен бути у вигляді WEB - додатку та мати сучасний інтерфейс. Головними вимогами до інтерфейсу є зручність та прозорість.

Надійність:

1. Доступність:

WEB - додаток повинен бути доступний протягом усього часу, виключенням є непередбачені ситуації та обслуговування.

2. Напрацювання на відмову:

Середній час безвідмовної роботи - 30 робочих днів.

3. Норма дефектів:

Максимальна норма помилок або дефектів - 1 помилка на десять тисяч рядків коду.

Продуктивність:

1. Одночасно працюють користувачі:

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

Система повинна бути здатна підтримувати мінімум 300 одночасно працюючих користувачів, пов'язаних із загальною базою даних.

2. Час відгуку:

Час відгуку для типових задач - не більше 5 секунд, для складних завдань - не більше 20 секунд.

Придатність до експлуатації:

1. Масштабованість:

Система повинна бути здатна підтримувати мінімум 300 одночасно працюючих користувачів, пов'язаних із загальною базою даних і мати можливість збільшити їх кількість на випадок збільшення штату співробітників підприємства. Кількість студентів та викладачів постійно змінюється, тому точно сказати кількість користувачів важко, тому система повинна мати можливість обробляти велику кількість користувачів.

2. Оновлення версій:

Оновлення версій має здійснюватися в автоматизованому режимі на основі системи контролю версій і системи (сервера) оновлення версій на робочих місцях користувачів.

Обмеження проектування:

1. Застосовувані стандарти.:

Система повинна відповідати всім стандартам інтерфейсу користувача Microsoft® Windows® або ОС UNIX та сучасним нормам проектування WEB - додатків.

2. Вимоги до середовища виконання.:

Система повинна задовольняти зазначеним вище вимогам на комп'ютері в наступній мінімальній комплектації:

- 2048 МБ пам'яті
- 10 ГБ вільного дискового простору
- процесор з тактовою частотою 2 ГГц
- Операційна система Windows або Linux
- Інтернет з'єднання

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

3. Вимоги до СКБД і доступу до даних:

У ядрі системи повинна бути представлена промислова СКБД реляційного доступу. Всі звернення до інформації повинні здійснюватися через SQLite.

Антивимоги:

1. Не використовує інформацію, заборонену нормами моралі та законодавством.
2. Не використовує інформацію, що є недійсною або може зашкодити психіці користувачів.
3. Не містить реклами.

Бізнес логіка:

Реклама нашого сайту на інших сайтах. Умова: власні кошти.

- Продукти та послуги: проект надає послуги для оформлення освітнього порталу для вашого навчального закладу.
- Користування платформою буде доступно після купівлі ліцензії на використання, після чого покупцю будуть відправлені файли для хостингу та модифікування проекту за бажанням.

1.2 Аналіз існуючого програмного забезпечення за тематикою проекту

На даний момент існує велика кількість Web-додатків, що реалізують різноманітні взаємодії з користувачем. Більшість із навчальних площадок направлені на деяку цільову аудиторію. Розроблюємий додаток буде розглядатися в контексті вже існуючих аналогів, які мають велику популярність.

Moodle (learn) [2].

Платформа Moodle є освітньою системою з відкритим кодом, яка допомагає створити структуру для дистанційного навчання. Платформа дозволяє створювати курси, налаштовувати їх, працювати з користувачами, підключати плагіни, які можуть доповнити функціональність. Також має

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

можливості масштабування, що дозволяє використовувати її у навчальних закладах різного розміру (рис 1.1).

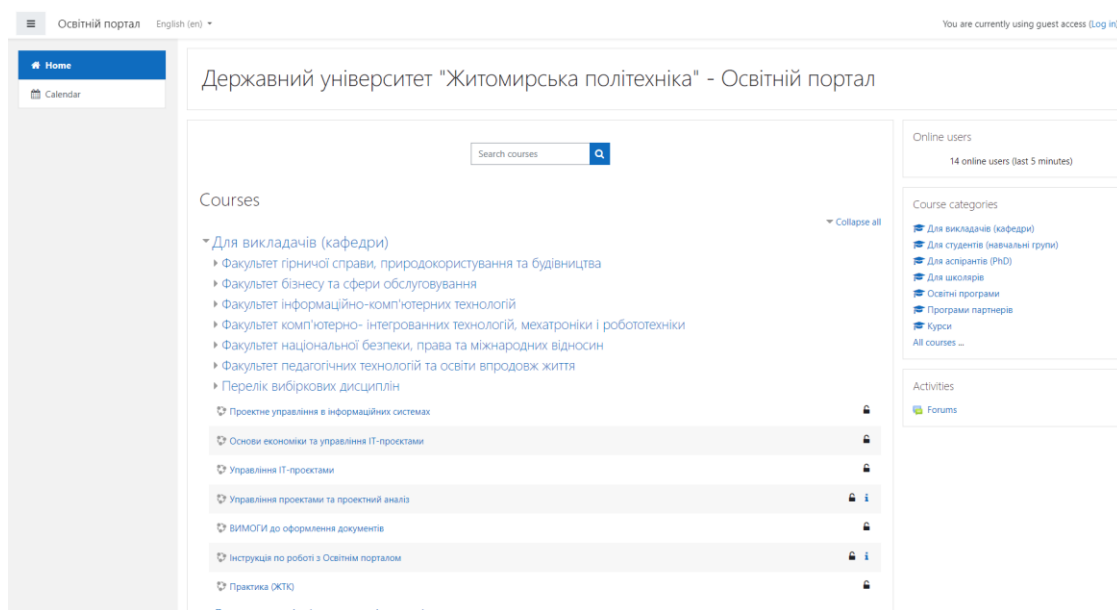


Рис. 1.1 – Інтерфейс Moodle

Для огляду було обрано реалізацію для Державного університету "Житомирська політехніка".

Переваги:

- Гнучке налаштування.
- Система доступу до курсів.
- Система оцінювання.
- Система тестувань.

Недоліки:

- Нагромадження великої кількості елементів інтерфейсу.
- Відсутність єдиної структури курсів.

Udemy [3].

Платформа Udemy сфокусована на завантаження окремих незалежних курсів, які створюються різними людьми та компаніями. Дана система дозволяє створювати свої курси, але лише підтвердженим преміум користувачам (рис 1.2).

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

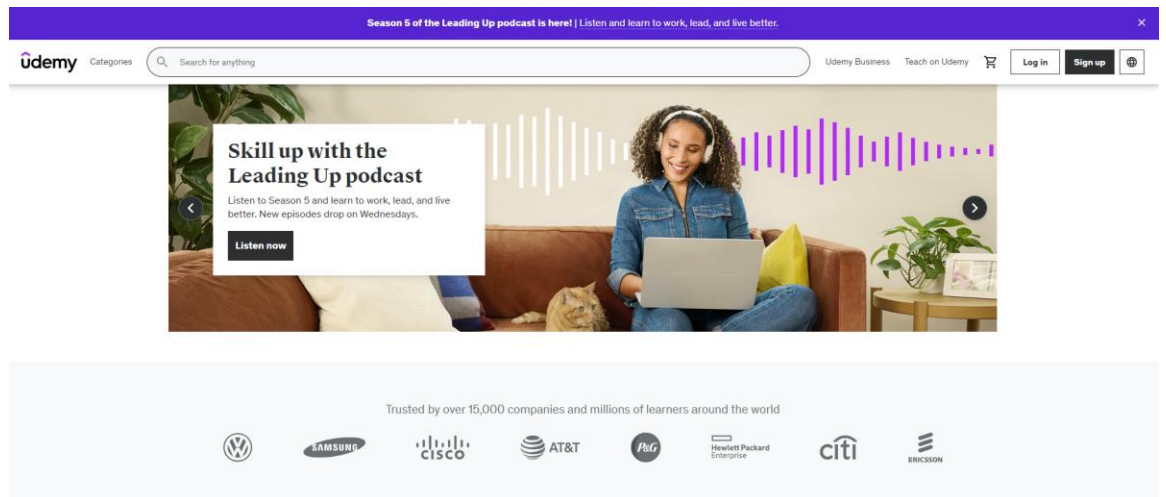


Рис. 1.2 – Інтерфейс Udemy

Переваги:

- Можливість навчатися у партнерів Udemy.
- Зручний інтерфейс курсів.
- Отримання сертифікатів при завершенні курсів.

Недоліки:

- Фокус на персональне навчання.
- Неможливість реалізації структури для навчальних закладів.
- Нагромадження інформації при пошуку курсів.
- Неможливість додавання власних курсів.
- Платне навчання.

Brilliant [4].

Дана система надає можливість навчатися за декількома напрямками, кожен з яких містить велику кількість інтерактивних завдань. Усі курси розроблені вручну та недоступні без передоплати, також не можливо створювати курсу в ручну (рис 1.3).

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

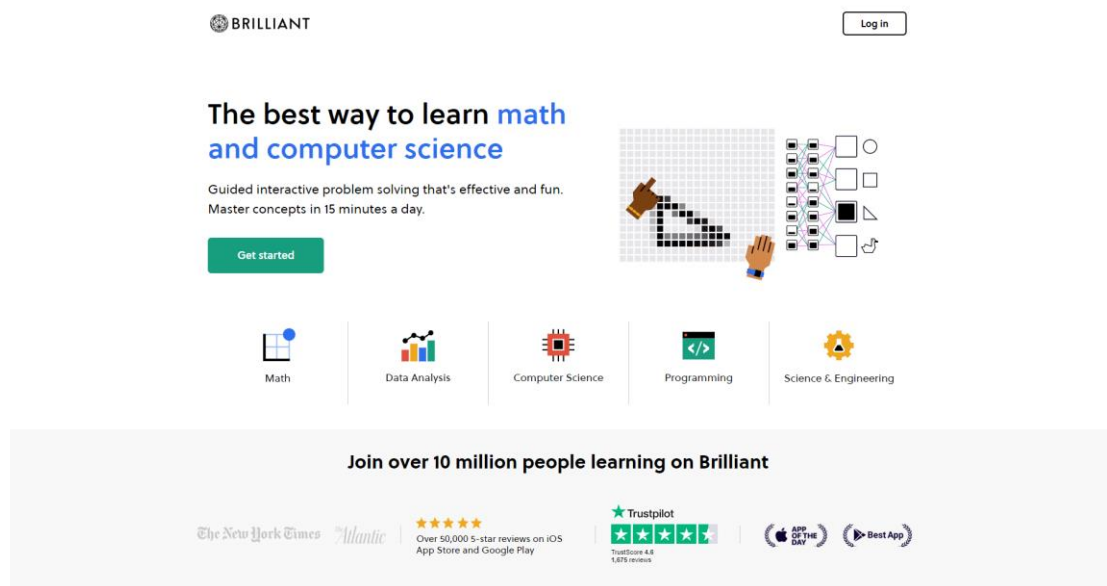


Рис. 1.3 – Інтерфейс Brilliant

Переваги:

- Інтерактивний підхід до курсів.
- Зручний інтерфейс.
- Можливість навчатися у своєму темпі

Недоліки:

- Фокус на персональне навчання.
- Неможливість реалізації структури для навчальних закладів.
- Неможливість додавання власних курсів.
- Лише платне навчання.

Перераховані платформи надають можливість навчатися за допомогою різного функціоналу та методів, але кожна з них відрізняється підходами та реалізацією. Також важливим фактором є цільова аудиторія, кожна з платформ має різний фокус, Moodle спрямована на організацію процесу навчання певних груп людей, в той час, Udemu та Brilliant фокусуються на персональному навчанні.

Для кращого розуміння розглянутих систем, порівняємо їх функціональні можливості (табл. 1.1).

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

Функціональні можливості

Назва	Moodle	Udemy	Briliant	Освітній портал
Можливість створення власних курсів	+	-	-	+
Можливість введення рейтингу студентів	+	-	-	+
Можливість реалізації структури для навчального закладу	+	-	-	+
Прозорий інтерфейс	-	-	+	+
Безкоштовний доступ до курсів	+	-	-	+
Отримання сертифікатів	-	+	-	-

1.3 Аналіз вимог системи та користувачів освітнього порталу

Мета аналізу вимог полягає в тому, щоб зібрати, проаналізувати та визначити високорівневі потреби і можливості системи освітнього порталу. Вимоги акцентують увагу на можливостях, необхідних співвласникам і цільовим користувачам, і на те, чому ці потреби існують.

Подробиці того, як система освітнього порталу виконує ці потреби, будуть деталізовані в прецедентах і додаткових специфікаціях у додатку А.

Бізнес-вимоги:

1. Основні цілі: проект створюється з метою надання функціоналу для проведення навчання студентів та контролю навчального процесу.
2. Спрощення роботи викладача: проект надає зручні інтерфейс та засоби для оприлюднення навчального матеріалу та контролю успішності студентів.
3. Представлення проекту: проект буде реалізовано у вигляді сайту, що містить набір інструментів для налаштування та проведення процесу навчання.

4. Прискорення пошуку інформації: Надання зручного інтерфейсу для пошуку необхідної інформації для студентів.

5. Підтримка великої кількості користувачів: Сервер буде обробляти усіх користувачів та зберігати усю необхідну інформацію.

Вимоги користувачів:

Студент:

1. Має доступ до перегляду курсів свого начального закладу.

2. Має можливість здавати роботи.

3. Має можливість переглядати оцінки за обраними курсами.

Викладач:

1. Має можливість редагувати вміст свого курсу.

2. Має можливість додавати та видаляти навчальний матеріал.

3. Має можливість змінювати доступ до курсу, роблячи його доступним для всіх, або по запрошенню.

4. Має можливість переглядати та змінювати оцінки студентів.

5. Має можливість налаштовувати дати закінчення курсу та переглядати архів оцінок.

Адміністратор:

1. Має можливість створювати та редагувати профілі користувачів.

2. Має можливість редагувати структури курсів.

3. Має можливість переглядати зміст курсів свого навчального закладу.

4. Має можливість переглядати оцінки студентів.

5. Має можливість переглядати архів оцінок.

На основі проаналізованих вимог була створена діаграма варіантів використання (рис 1.4).

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

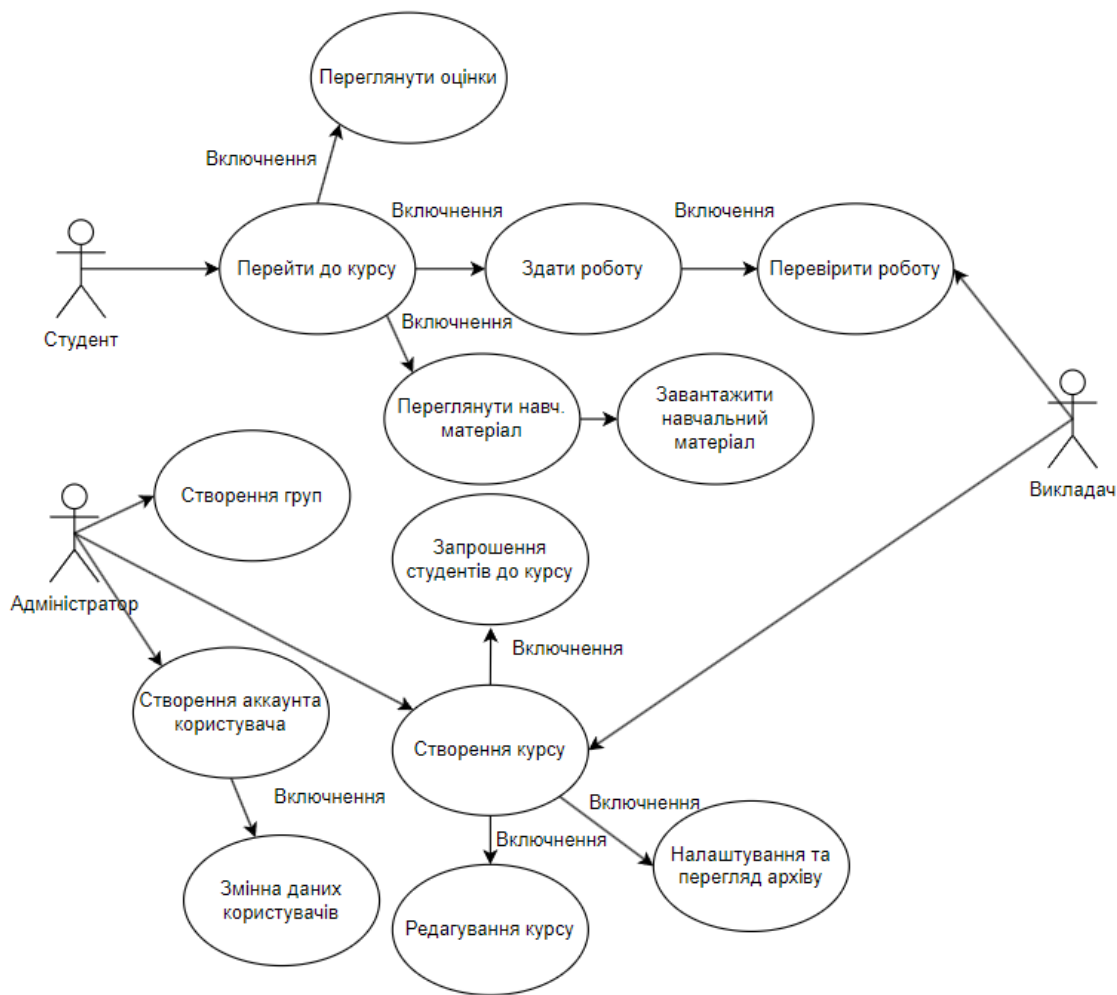


Рис. 1.4 – Діаграма варіантів використання

Діаграма варіантів використання відображає у собі проаналізовані вимоги користувачів [5] [17].

1.4 Обґрунтування вибору інструментальних засобів та архітектури системи освітнього порталу

Існує велика кількість архітектур додатків, серед яких було обрано клієнт-серверну. Вона поділяється на наступні компоненти:

- Сервер – оброблює запити користувачів на надсилає відповіді.
- Клієнти – використовують застосунки, які дозволяють спілкуватися з сервером, у даній реалізації використовується браузер.
- Мережа – зв'язок відбувається у просторі інтернет мережі.
- База даних – відповідає за збереження даних.

Дана модель визначає три рівні операцій:

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

- Рівень представлення даних – інтерфейс за допомогою якого дані потрапляють до користувача.
- Прикладний рівень – реалізує логіку застосунку, займається обробкою даних, тобто сервер.
- Рівень управління даними – забезпечує зберігання та доступ до даних, реалізується за допомогою СКБД.

Обрану архітектуру необхідно реалізувати за допомогою поєднання інструментів розробки веб-інтерфейсу та веб-серверу, тому було обране наступне об'єднання засобів:

- Розмітка WEB-сторінок: HTML5 [6], CSS [7], JavaScript [8];
- Мова розробки серверної частини: Python [9] [18];
- Фреймворк: Flask [10];
- Шаблонізатор: Jinja2 [11];
- Бібліотека-драйвер для роботи з базою даних: SQLAlchemy [12];
- База даних: SQLite [13];
- Додаткові бібліотеки: Werkzeug [14], flask_login, OS [15], WTForms [16].

Сервер є найголовнішою частиною будь якого WEB-додатку, мова програмування Python є гнучкою мовою з можливістю реалізації алгоритмів різної складності, що дозволяє зручно розроблювати серверну частину.

Фреймворк Flask містить у собі інструменти для розробки веб додатків. Він дозволяє розмічати посилання для серверу, оброблювати запити, які до нього надходять, працювати з файлами, що є важливим для системи освітнього порталу та має великий набір інтеграцій з іншими бібліотеками.

Шаблонізатор Jinja2, є невід'ємною частиною Flask. Інтеграція даних бібліотек дозволяє готувати WEB-сторінки, наповнюючи їх даними з серверу, перед відправкою користувачу. За допомогою шаблонів, будуються структури WEB-сторінок, мова HTML5 дозволяє розмітити сторінки, після чого, вони використовуються як шаблони, у які Jinja2, вставляє данні використовуючи спеціальний синтаксис, схожий на Python.

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

Бібліотека-драйвер SQLAlchemy є популярним вибором для роботи з базами даних на мові Python. Вона дозволяє абстрагувати використання баз даних від додатка, таким чином заміна однієї бази даних на іншу вимагає лише зміну посилання доступу до ресурсу бази даних. Також дана бібліотека надає функціонал для розмітки бази даних на мові Python, що дозволяє швидко та ефективно використовувати бази даних. Вона також має інтеграцію з Flask, яка використовується для керування користувачами, системою авторизації та обробленням помилок.

Бібліотека Werkzeug, є набором інструментів для роботи, захисту та перевірки працездатності серверу. Вона дозволяє захищати паролі користувачів, за допомогою шифрування SHA256, захищати назви файлів, перед зберіганням на сервері та багато іншого.

Flask-login є додатковим модулем для Flask, який дозволяє оброблювати систему авторизації, та поєднати між Flask та SQLAlchemy.

WTForms це бібліотека для захисту, відображення та валідації форм. Вона підтримує різні шаблонізатори, за допомогою яких реалізується відображення форми. Сервер у свою чергу використовує інструменти для валідації даних та захисту за допомогою системи генерації сесійних токенів.

СКБД SQLite написана на мові C, та оформлена як бібліотека. Це дозволяє їй бути швидкою, надійною, само достатньою та повнофункціональною, при цьому зберігати невеликий розмір.

Характеристики SQLite:

- Повнофункціональність;
- Максимальний розмір: 281 ТБ;
- Максимальний розмір одного запису: 1 ГБ.
- Представлена у вигляді фізичного файлу.
- Відмінна стійкість на надійність;
- Ліцензія: немає (суспільне надбання)
- Кроссплатформність.

СКБД SQLite має додаткову підтримку у Flask, що доповнює її функціонал та пришвидшує налаштування бази даних для подальшого використання сервером.

Висновки до першого розділу

У першому розділі було проаналізовано завдання на кваліфікаційну роботу, поставлено ціль та мету, сформовано технічне завдання. Після чого були оглянуті та порівняні різні аналоги програмного продукту. Також було проаналізовано вимоги до системи освітнього порталу та розроблено діаграму використання, що надає глибшого розуміння системи. Далі було обрано та описано архітектуру та інструментальні засоби для реалізації системи.

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ОСВІТНЬОГО ПОРТАЛУ

2.1 Опис акторів та алгоритмів роботи

Наступним кроком розробки системи освітнього порталу є аналіз алгоритму роботи та створення діаграм активності.

Основними акторами є користувачі з наступними ролями: студент, викладач та адміністратор (табл. 2.1).

Таблиця 2.1

Користувачі

Актор	Короткий опис
Студент	Має можливість переглядати зміст курсів, завантажувати навчальний матеріал, виконувати та здавати роботи. Також має можливість переглядати оцінки за обраними курсами
Викладач	Має можливість створювати курси, редагувати їх, та наповнювати навчальним матеріалом. Також має можливість перевіряти здані студентами роботи та виставляти оцінки.
Адміністратор	Має доступ до усіх елементів системи, може створювати користувачів, редагувати їх інформацію. Також може створювати групи.

Актори будуть взаємодіяти з один з іншим за допомогою WEB - інтерфейсу. Розберемо основні алгоритми роботи за допомогою діаграм активності [19].

Процес створення курсу може почати, як і викладач, так і адміністратор. Після переходу до сторінки створення курсу, відкривається спеціальна форма для редагування вмісту курсу, після завершення наповнення матеріалом, форма відправляється до серверу, уся інформація зберігається (рис 2.1).

Адміністратор/Викладач: Користувач

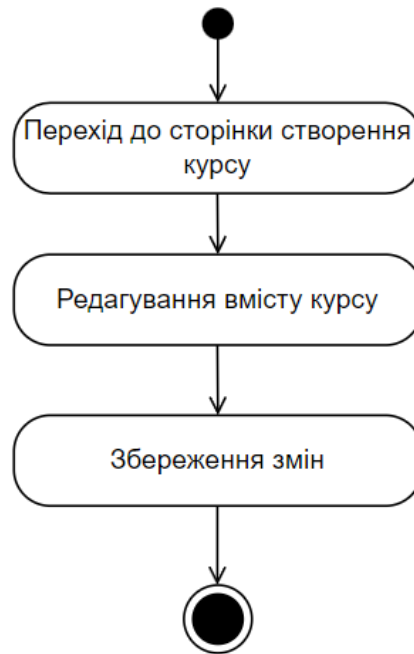


Рис. 2.1 – Діаграма активності для створення курсу

Процес запрошення студентів до закритого курсу може розпочати, як викладач, так і адміністратор. Вони обирають студентів, після чого система відсилає запрошення. Коли студент отримує запрошення, він може перейти до курсу (рис 2.2).

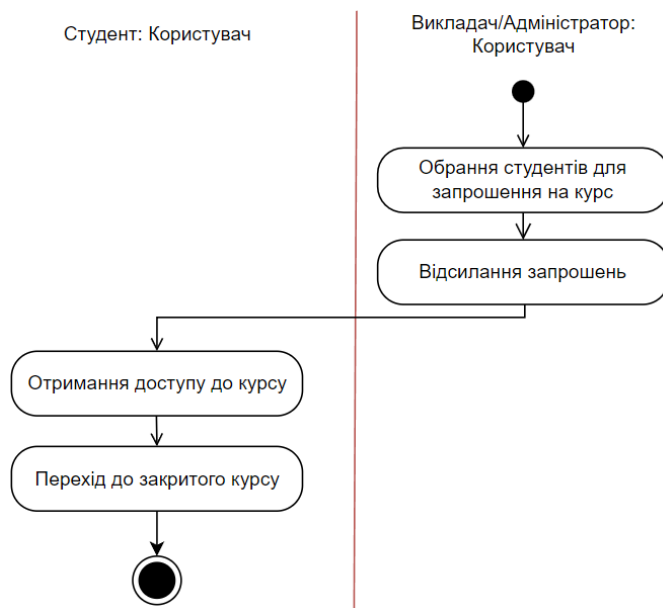


Рис. 2.2 – Діаграма активності для процесу відсилання запрошень на закритий курс

Процес перевірки зданої роботи розпочинається з того, що студент за допомогою спеціального інтерфейсу завантажує виконане завдання до системи освітнього порталу. Далі викладач на окремій сторінці може завантажити та перевірити завдання, після чого він вирішує чи виконане завдання вправно, та виставляє або не виставляє оцінку (рис 2.3).

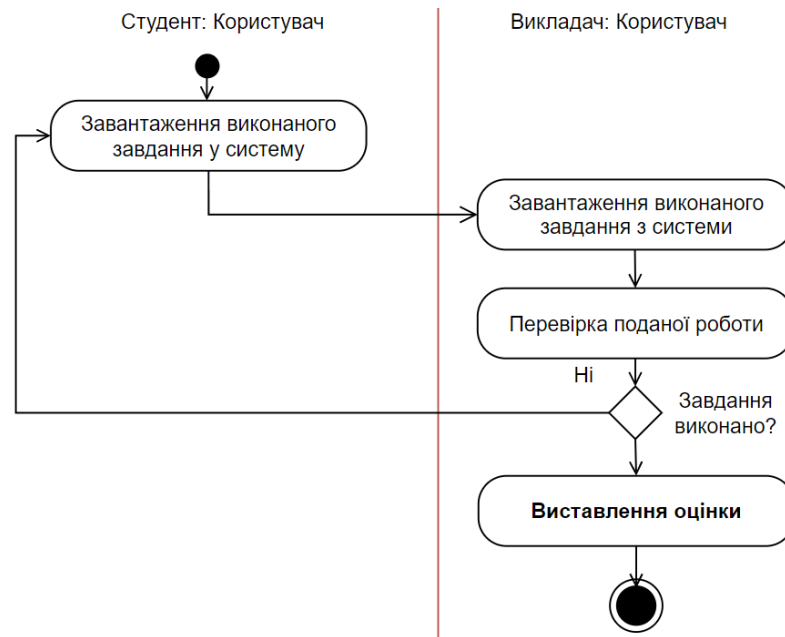


Рис. 2.3 – Діаграма активності для процесу перевірки зданої роботи

Процес редагування змісту курсу розпочинається з того, що викладач або адміністратор переходять до сторінки певного курсу, де обирають режим редагування, після чого змінюють зміст курсу та зберігають зміни (рис 2.4).

Адміністратор/Викладач: Користувач



Рис. 2.4 – Діаграма активності для процесу редагування курсу

2.2 Розробка бази даних

Наступним важливим кроком розробки системи освітнього порталу є опис та створення бази даних. База даних складається з 7 таблиць, розберемо їх (табл. 2.2).

Таблиця 2.2

Опис таблиць

Назва таблиці	Короткий опис
Course	Містить у собі курси та їх налаштування
Structure	Містить у собі структуру курсів, кожен запис відповідає одному блоку курсу.
Course_users	Допоміжна таблиця, для реалізації зв'язку багато до багатьох, між таблицями Course та User
Group	Містить у собі перелік груп.
User	Містить усю інформацію про користувачів
Mark	Містить усю інформацію про здані роботи та оцінки
Sqlite_master	Допоміжна таблиця, для організації роботи бази даних

Усі таблиці створені за допомогою бібліотеки – драйверу SQLAlchemy, це дозволяє описувати структуру бази даних у кодї на мові Python, що пришвидшує розробку бази даних, та надає зручний синтаксис для подальшого редагування та змін. Завдяки паттерну MVC, усі моделі зберігаються окремо, описані у вигляді класів, які наслідують об'єкт з бібліотеки SQLAlchemy.

Розберемо зв'язки таблиць за допомогою діаграми візуалізації бази даних (рис 2.5).

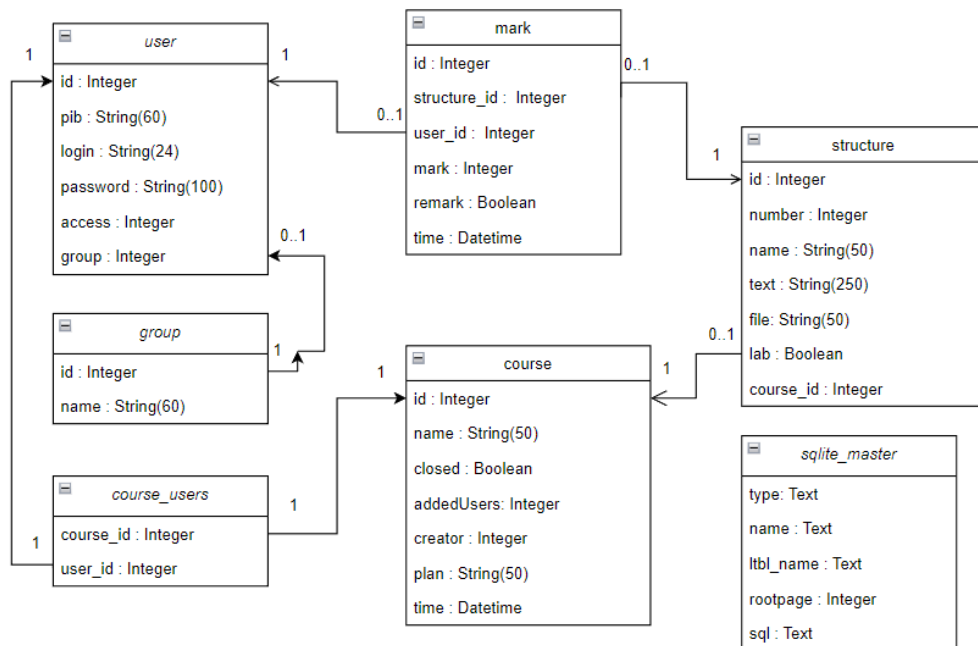


Рис. 2.5 – Візуалізація бази даних

Для кращого розуміння діаграми розберемо таблиці окремо.

Таблиця User є одним із основних елементів у системі освітнього порталу, за допомогою цієї таблиці реалізовано автентифікацію, систему доступу та систему оцінювання. Дана таблиця на пряму працює з контролером, що відповідає за усю бізнес-логіку. Також система доступу до курсів, працює з даною таблицею, для перевірки аккаунту студента. Система автентифікації допоможе захистити дані від незнайомих користувачів та забезпечить надійність у виконанні інших, пов'язаних систем. Усі паролі користувачів захищені за допомогою алгоритму шифрування PBKDF2 (табл. 2.3) [20].

Таблиця 2.3

Користувачі

User		
Назва поля	Тип даних	PK/FK
ID	Integer	PK
PIB	String(60)	-
Password	String(100)	-
Login	String(24)	-

Продовження таблиці 2.3

Access	Integer	-
Group	Integer	FK

Таблиця оцінки містить у собі інформацію стосовно оцінок студентів за окремими курсами. Завдяки полям Structure_id та User_id дані будуть поєднуватися з іншими таблицями, таким чином відрізняючи оцінки студентів по окремим розділам курсів. Записи створюються при здачі робіт, містять у собі назву файлу, який був завантажений на сервер та стан оцінювання, якщо робота була перездана, то поле Remark буде змінено та робота знову повинна бути перевірена. Поле Time використовується для відображення в архіві (табл. 2.4).

Таблиця 2.4

Оцінки

Mark		
Назва поля	Тип даних	PK/FK
ID	Integer	PK
User_id	Integer	FK
Structure_id	Integer	FK
Mark	Integer	-
File	String(50)	-
Remark	Boolean	-
Time	Datetime	-

Таблиця Course містить у собі усю необхідну інформацію про курси. Поле AddedUsers вказує на записи таблиці User. Кожен з курсів має свою структуру, яка буде фактично його вмістом та побудована з блоків, вони впорядковані та з'єднані за допомогою поля ID, таким чином кожен курс буде містити унікальні блоки з унікальною інформацією. Поле Closed відповідає за

тип курсу, або відкритий, або закритий. Поле Time використовується для збереження часу закінчення курсу (табл. 2.5).

Таблиця 2.5

Курси

Course		
Назва поля	Тип даних	PK/FK
ID	Integer	PK
Name	String(50)	-
Closed	Bool	-
AddedUsers	Integer	FK
Creator	Integer	FK
Plan	String(50)	-
Time	Datetime	-

Таблиця Structure містить у собі частини курсу, з яких будується курс, таким чином можна буде редагувати розділи курсу та змінювати їх кількість. Окрім наповнення блоку, записи містять поле Number, яке використовується для впорядкування у правильній послідовності. Також поле Lab допомагає зрозуміти для системи та користувачів, чи є даний блок лекцією або завданням, яке потрібно виконати (табл. 2.6).

Таблиця 2.6

Структура курсу

Structure		
Назва поля	Тип даних	PK/FK
ID	Integer	PK
Number	Integer	-
Name	String(50)	-
Text	String(250)	-
File	Integer	-
Lab	Boolean	-
Course_id	Integer	FK

Таблиця Group містить у собі назви груп, які призначаються користувачам, за допомогою записів з даної таблиці, можна фільтрувати студентів за групами та додавати разом всю групу до курсу, замість обирання кожного студента окремо (табл. 2.7).

Таблиця 2.7

Групи

Group		
Назва поля	Тип даних	PK/FK
ID	Integer	PK
Name	String(60)	-

Course_users використовується для реалізації зв'язку багато до багатьох між таблицями Course та User. За допомогою SQLAlchemy можна отримати перелік зв'язаних таблиць за допомогою одного запиту, що є корисним під час розробки алгоритмів серверу (табл. 2.8).

Таблиця 2.8

Допоміжна таблиця між курсами та користувачами

Course_users		
Назва поля	Тип даних	PK/FK
Course_id	Integer	FK
User_id	Integer	FK

2.3 Комунікації і послідовність взаємодії об'єктів системи

Наступним кроком моделювання системи освітнього порталу є створення діаграм послідовностей, що допоможе покращити розуміння про комунікації компонентів у системі.

Діаграма послідовності переходу користувача до курсу починається з запиту користувача до серверу. Після чого контролер обробляє запит та звертається до моделі бази даних для отримання усіх необхідних даних. Після отримання даних, контролер звертається до шаблонізатору, який повертає

шаблон (розмічену HTML сторінку), яку потім надсилає користувачу (рис. 2.6).

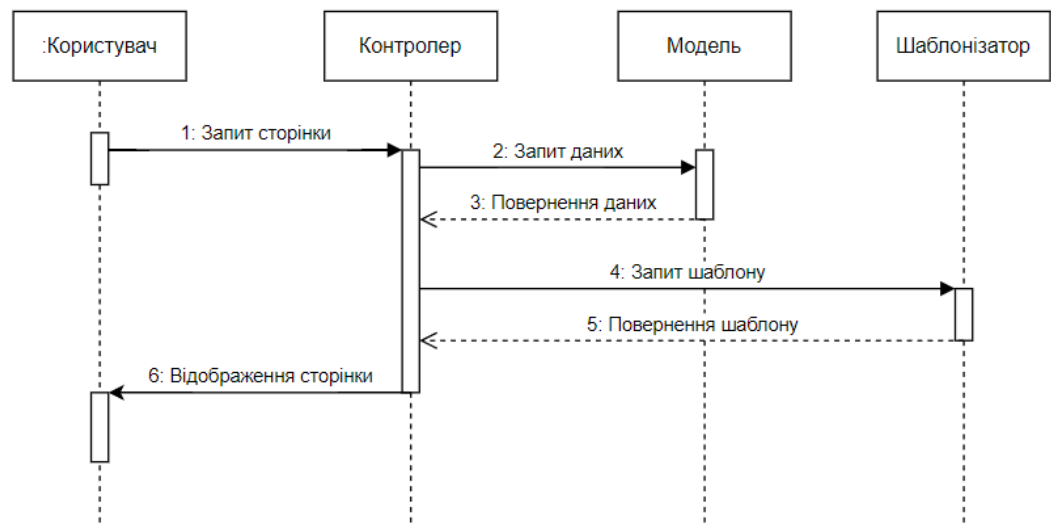


Рис. 2.6 – Діаграма послідовності переходу користувача до курсу

Процес створення нового курсу починається, з того що користувач обирає пункт створення курсу за допомогою інтерфейсу, після чого запит відсилається до контролера, який у свою чергу запитує шаблонізатор для отримання шаблону для налаштування курсу. Після чого даний шаблон відправляється користувачу, який вводить необхідні дані, та відсилає їх на сервер за допомогою форми. Далі контролер обробляє отриману інформацію, та відсилає запит, до бази даних, на створення курсу за певними параметрами, після отримання результатів, запитує шаблон, з створеним курсом, від шаблонізатора, який потім відображає користувачу (рис. 2.7).

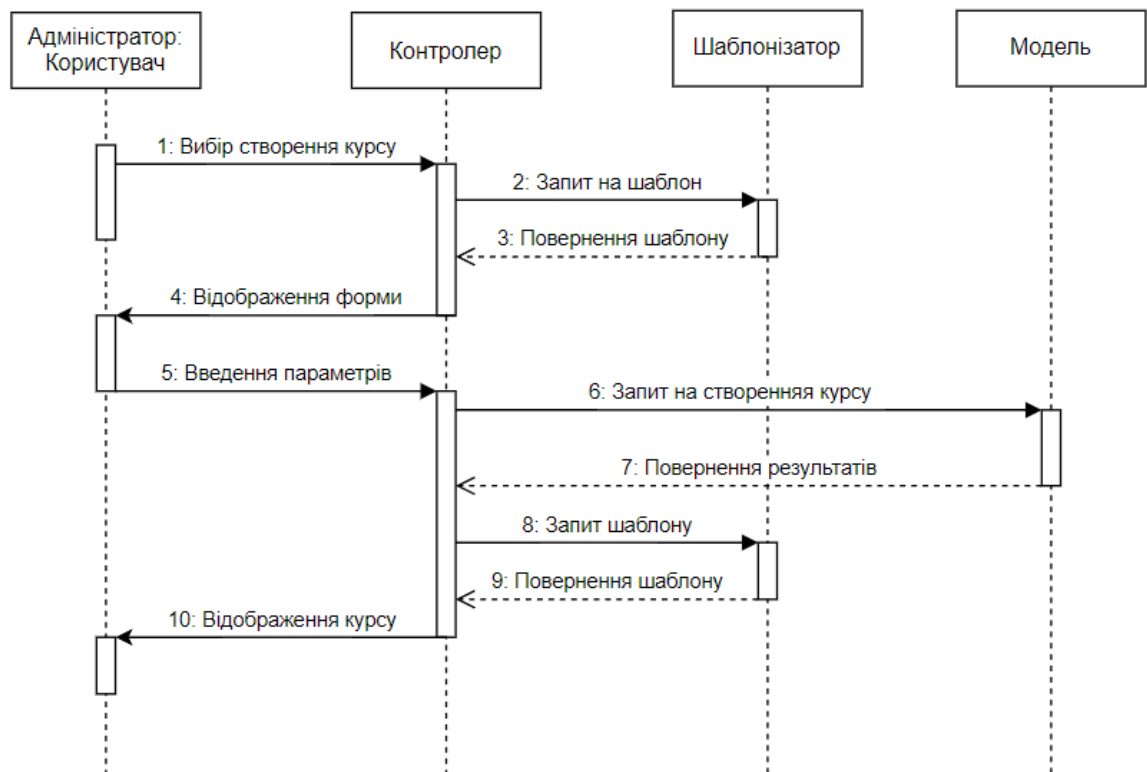


Рис. 2.7 – Діаграма послідовності створення нового курсу

Процес задачі роботи починається з того, що користувач за допомогою спеціального інтерфейсу завантажує роботу до серверу, після чого контролер, відправляє запит на оновлення даних до моделі, після повернення результатів, контролер запитує шаблонізатор, після отримання шаблону, він відображає результат операції користувачу (рис. 2.8).

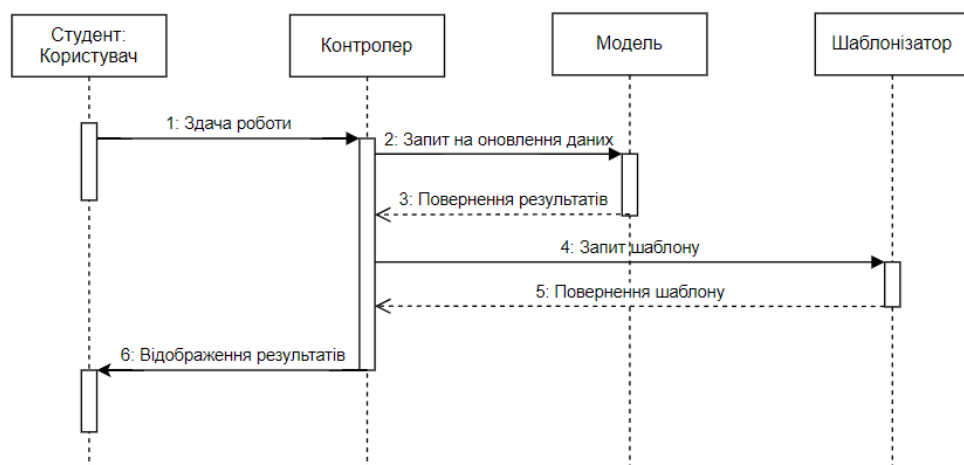


Рис. 2.8 – Діаграма послідовностей задачі роботи

Діаграма послідовностей перевірки роботи, починається з вибору викладачем роботи, яку він збирається перевірити, після чого запит на обрану роботу відсилається до серверу. Далі контролер запитує модель бази даних на отримання файлу, далі запитує шаблонізатор на отримання шаблону, на якому

буде розташоване посилання на завантаження файлу. Після чого викладач отримує сторінку з файлом, завантажує файл, перевіряє роботу, та за допомогою спеціального інтерфейсу виставляє оцінку. Виставлена оцінка відправляється до серверу, де контролер запитує базу даних на оновлення даних, після чого він запитує шаблонізатор на отримання шаблону з результатами, який потім надсилається користувачу (рис. 2.9).

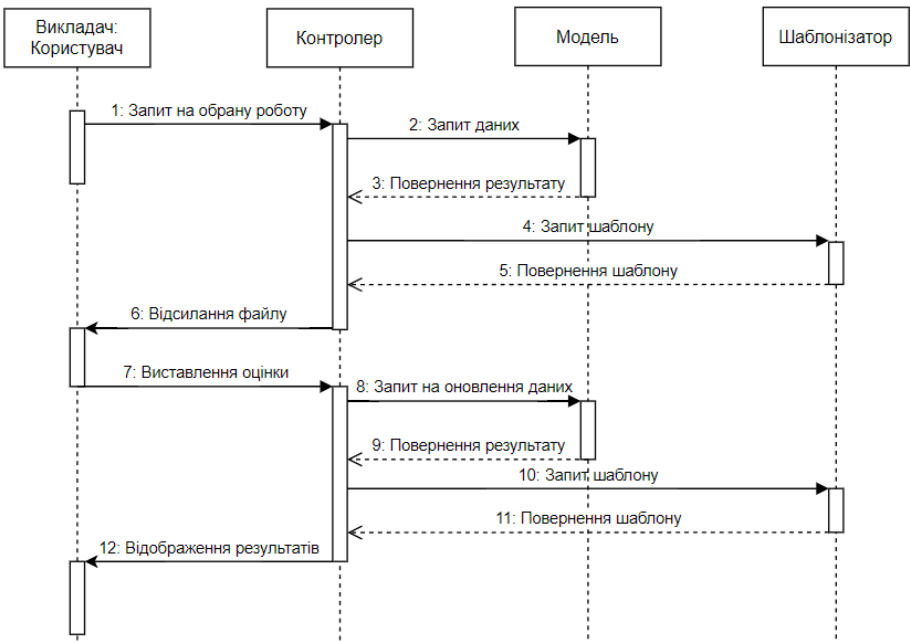


Рис. 2.9 – Діаграма послідовностей перевірки роботи

2.4 Реалізація системи освітнього порталу

Систему можна умовно поділити на 3 частини: контролери, які опрацьовують запити, які надходять до сервера; шаблони, які використовуються для побудування сторінок; моделі, які описують будову бази даних та допомагають складати запити до неї.

Спочатку розглянемо реалізацію функції, яка обробляє запити на авторизацію користувачів.

```

@app.route('/login', methods=['GET', 'POST'])
def login_page():
    form = LoginForm()
    if form.validate_on_submit():
        if form.login.data and form.password.data:
            user = User.query.filter_by(login=form.login.data).first()
            if user and check_password_hash(user.password,
            str(form.password.data)):
                login_user(user)
                next_page = request.args.get('next')
                if next_page:
                    return redirect(next_page)
                else:
                    return redirect('/')
        else:
            flash('Помилка у введенних даних')
            return render_template('login.html', form=form)
    else:
        flash('Поля повинні бути заповненими')
        return render_template('login.html', form=form)
    else:
        return render_template('login.html', form=form)

```

На початку використовується декоратор, він надається бібліотекою Flask, у параметри задається шлях, при отриманні запиту на який, функція буде спрацьовувати та перелік методів, які дозволено отримувати за обраним шляхом.

Далі за допомогою WTForms, створюється захищена форма, яка містить у собі токен, для захисту форми від потенційних атак. Після чого, форма перевіряється, чи вона містить данні та була правильно відправлена користувачем. Далі за допомогою моделі ми отримуємо дані користувача з бази даних та перевіряємо, чи співпадають паролі за хешем, це необхідно для забезпечення безпеки даних користувачів. Якщо всі умови виконались, сервер авторизує користувача та пересилає його на головну сторінку. Також необхідно звернути увагу на функцію `render_template`, вона використовується для формування шаблонів, у параметри необхідно вказати назву файлу шаблону та данні, які будуть передані до нього.

Таким чином описуються усі функції у контролерах, кожна містить посилання, яке вона обробляє, визначений набір методів, які приймаються сервером та алгоритм для обробки даних, які потрібно надіслати користувачу, або обробити та зберегти.

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				32
Змн.	Арк.	№ докум.	Підпис	Дата		

Розглянемо приклад шаблону сторінки авторизації. Вона будується після виконання усіх передумов описаних у контролері, після чого відправляється користувачу.

```
{% block body %}
<div>
  <h1 style="text-align: center;">Логін</h1>
  <div style="width: 191px; text-align: center; background-color: red;
color: white;">
    {% with messages = get_flashed_messages() %}
      {% if messages %}
        {% for message in messages %}
          {{ message }}
        {% endfor %}
      {% endif %}
    {% endwith %}
  </div>
  <form method="post" class="forma" name="login">
    {{ form.hidden_tag() }}
    <div class="logpassblock">
      {{ form.login.label }}
      {{ form.login(size=24,class="loginpole") }}
      {{ form.password.label }}
      {{ form.password(size=24,class="loginpole") }}
    </div>
    <hr size="1px" noshade color="grey"></hr>
    <div class="regblock">
      <a href="/register" class="reg">Регістрація</a>
      <input type="submit" value="Увійти" class="reg">
    </div>
  </form>
</div>
{% endblock %}
```

На початку знаходиться синтаксис шаблонізатору Jinja2, це дозволяє нам розширити шаблон за допомогою інших шаблонів, у даному випадку додається шапка сайту та підвал, таким чином усі сторінки будуть мати спільні стилі та елементи інтерфейсу. Далі використовується структура HTML для розмічення сторінки, поєднуючи її з шаблонізатором, можна виводити дані, які були передані від контролеру. У даному випадку, ми виділяємо окреме місце для виводу помилок, які можуть виникнути при відмові авторизації, також виводимо захищену форму. Шаблонізатор також дозволяє використовувати цикли, для розташування великої кількості об'єктів та інші функції, які можуть бути передані контролером.

Далі оглянемо модель таблиці User, вона надає розуміння SQLAlchemy, як побудована таблиця. Описана модель відповідає за зберігання даних та виконання запитів, пов'язаних з користувачами.

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

```

class User(db.Model, UserMixin):
    __tablename__ = "user"
    id = db.Column(db.Integer, primary_key=True)
    login = db.Column(db.String(24), nullable=False, unique=True)
    password = db.Column(db.String(100), nullable=False)
    access = db.Column(db.Integer, default=0)
    pib = db.Column(db.String(60), nullable=False)
    group = db.Column(db.Integer, db.ForeignKey(Group.id), nullable=True)

    def __repr__(self):
        return '<User %r>' % self.id

```

Усі моделі описані, як класи та наслідують об'єкт db.Model, що містить набір необхідного функціоналу. У класі вказуються поля таблиці, їх тип, параметри, ключі, то що. Також існує перевантаження стандартної функції __repr__(), яка відповідає, за вид даних, які повертаються при звертанні до класу.

Таким чином розроблені усі таблиці, перевагою даного підходу є універсальність при виборі двигуна бази даних, фактично можна замінити двигун, на будь який, що створений з підтримкою SQL мови, та усі таблиці будуть автоматично створюватися під час першого запуску.

Висновки до другого розділу

У даному розділі було проаналізовано акторів системи та послідовність виконання операцій. Після чого було спроектовано структуру бази даних та створено діаграму візуалізації. Також було проаналізовано комунікації і послідовність взаємодії об'єктів системи та розроблено діаграми послідовностей. На основі проведеного проектування було розроблено систему освітнього порталу.

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				34
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ОСВІТНЬОГО ПОРТАЛУ

3.1 Порядок встановлення та налаштування системи

Для розуміння взаємодії компонентів у системі освітнього порталу та подальшого налаштування була розроблена діаграма компонентів, яка відображає у собі залежності та зв'язки між різними компонентами у системі (рис. 3.1).

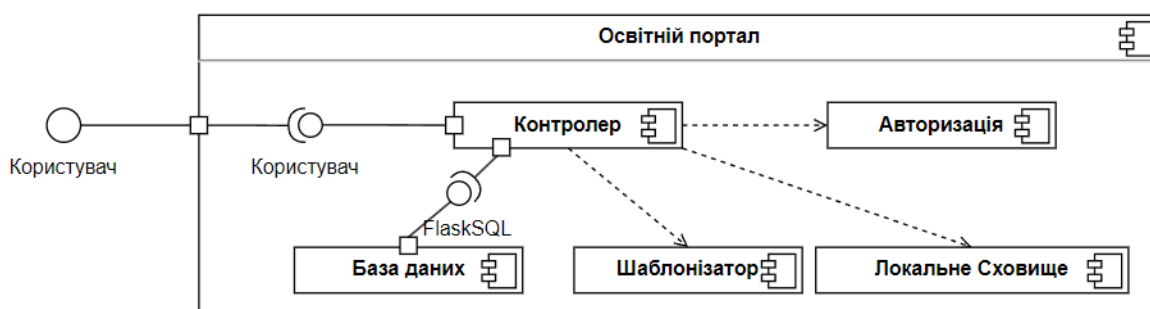


Рис. 3.1 – Діаграма компонентів

Користувач є основним джерелом даних, він використовує систему за допомогою інтерфейсу «Користувач», який обробляє отримані дані через мережу інтернет. Далі дані надходять до контролеру, який є основним компонентом у даній системі, він вирішує, що робити з цими даними, наприклад, якщо був створений новий курс, контролер за допомогою інтерфейсу «SQLAlchemy», оновлює дані у базі даних. Коли контролеру необхідно отримати шаблон для відображенні інформації користувачу він використовує компонент шаблонізатор. Якщо контролеру необхідно завантажити файл, то він звертається до локального сховища. Також контролер співпрацює з компонентом авторизації, якій відповідає за надання доступу користувачам до відповідних функцій, наприклад, тільки користувач з правами доступу адміністратора може створити новий курс.

Таким чином усі компоненти системи взаємодіють один з одним для виконання операцій.

Також було розроблено діаграму розгортання, яка надає інформацію про фізичну будову системи (рис. 3.2).

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

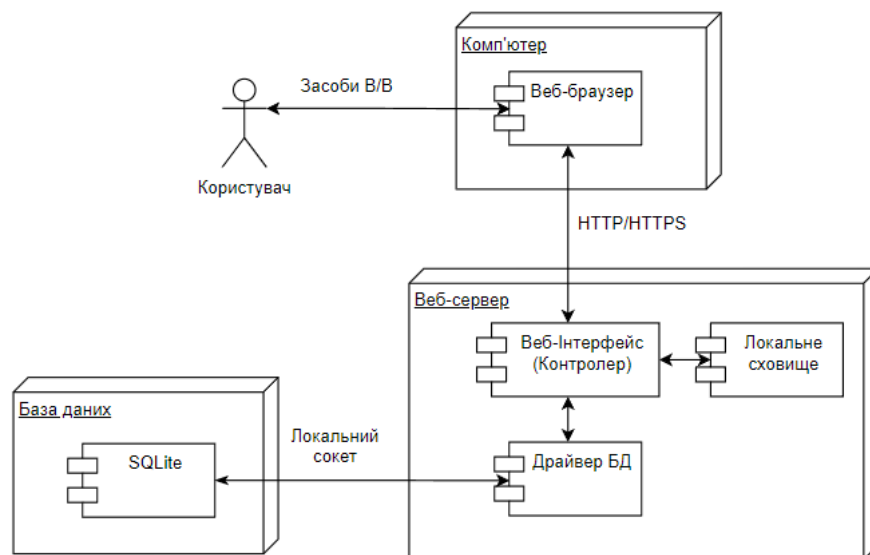


Рис. 3.2 – Діаграма розгортання

На діаграмі зображено комунікації користувача, за допомогою комп'ютера та мережі інтернет, він обмінюється інформацією з веб-сервером. У свою чергу компонент веб-інтерфейс (контролер) оброблює інформацію та звертається до інших компонентів системи. Локальне сховище, є частиною веб-серверу, воно буде зберігати у собі шаблони та завантажені файли. Також за допомогою драйверу бази даних (SQLAlchemy), контролер зв'язується з базою даних.

Після створення діаграми розгортання стає зрозуміло, що для запуску системи освітнього порталу, знадобиться хостинг для серверу, з деякою кількістю міста у локальному сховищі, також необхідно буде запустити базу даних, яка може розміщена на тому ж хостингу, або за допомогою іншого інструментарію. Необхідно звернути увагу, що для використання системи, користувачу знадобиться комп'ютер з встановленим браузером, за допомогою якого він буде зв'язуватися з веб-сервером.

Для запуску системи на хостингу потрібно підготувати проект до розгортання. Першим кроком є встановлення серверу, це виконується за допомогою команди «`pip install gunicorn`». Gunicorn — це сервер, розроблений за стандартом WSGI на чистому Python із простою конфігурацією та кількома робочими реалізаціями для налаштування продуктивності. Далі необхідно зберегти усі встановлені модулі у файлі, це виконується за командою «`pip`

freeze >> requirements.txt». За допомогою цього файлу, сервер під час запуску на хостингу, самостійно завантажить усі необхідні модулі. Далі необхідно змінити налаштування серверу, а саме ір адресу та порт, для того щоб сервер працював на адресою хостингу, за це відповідає команда: «app.run(host='0.0.0.0', port='80')». На даному етапі сервер готових до розгортання на хостингу, залишається лише запустити його за допомогою команди: «gunicorn -w 4 -b 0.0.0.0:80 app:app». Дана команда налаштовує сервер для роботи у режимі з 4 працівниками та прив'язує його до адреси хостинга.

3.2 Тестування системи освітнього порталу

Тестування є невід'ємною складовою розробки будь-якої програмної системи, особливо коли йдеться про освітні портали, які повинні забезпечувати надійний та безперебійний доступ до освітніх ресурсів. Якість освітнього процесу значною мірою залежить від стабільності та функціональності програмного забезпечення, яке використовують викладачі та студенти. Тому тестування системи освітнього порталу має ключове значення для досягнення поставлених цілей.

Для перевірки роботи освітнього порталу були розроблені тестові випадки, які описують предмет тестування, кроки, вхідні дані та очікуваний результат, ось деякі з них (табл. 3.1).

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

Тестові випадки

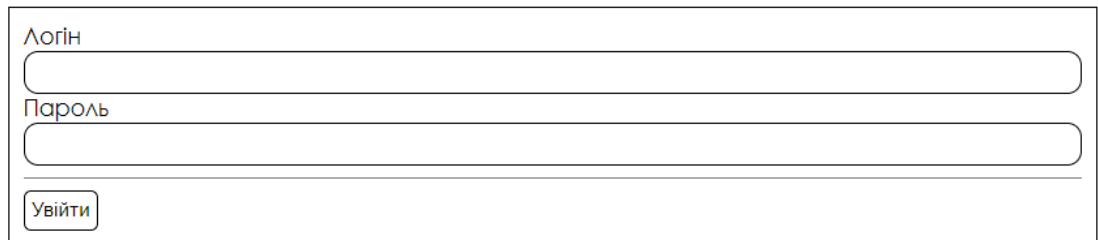
Опис тестового випадку	Кроки	Вхідні дані	Очікуваний результат	Фактичний результат	Пройшло/ Не пройшло
Успішна авторизація з правильними даними	1. Відкрити сторінку авторизації. 2. Ввести логін та пароль. 3. Натиснути кнопку «Увійти»	Логін: Artish Пароль: 1111	Успішна авторизація користувача	Успішна авторизація користувача	Пройшло
Авторизація з неправильними даними	1. Відкрити сторінку авторизації. 2. Ввести логін та пароль. 3. Натиснути кнопку «Увійти»	Логін: Artish Пароль: 1234	Повідомлення про помилку	Повідомлення про помилку	Пройшло
Авторизація з пустими полями	1. Відкрити сторінку авторизації. 2. Натиснути кнопку «Увійти»	Логін: Пароль:	Повідомлення про помилку	Повідомлення про помилку	Пройшло
Створення курсу	1. Відкрити сторінку створення курсу 2. Заповнити необхідними даними 3. Натиснути кнопку «Завантажити»	Назва курсу: Тестовий Назва блоку: Тестовий Опис блоку: Тестовий Файл: Лекційний матеріал: Так	Успішне створення курсу та додавання до бази даних	Успішне створення курсу та додавання до бази даних	Пройшло
Спроба створення курсу з пустими полями	1. Відкрити сторінку створення курсу 2. Натиснути кнопку «Завантажити»	Назва курсу: Назва блоку: Опис блоку: Файл: Лекційний матеріал: Так	Повідомлення про незаповнені поля	Повідомлення про незаповнені поля	Пройшло

Проведене тестування дозволило виявити та усунути недоліки, що значно знижує ризики під час експлуатації системи. Це, в свою чергу, сприяє підвищенню задоволеності користувачів та забезпеченню стабільної роботи порталу, особливо в критичні моменти, такі як екзаменаційні сесії або періоди інтенсивного використання.

3.3 Опис інтерфейсу програмного продукту

Під час першого переходу до сайту, відкривається форма авторизації, таким чином, щоб отримати доступ до функціоналу освітнього порталу, користувач повинен мати створений обліковий запис (рис 3.3).

Логін



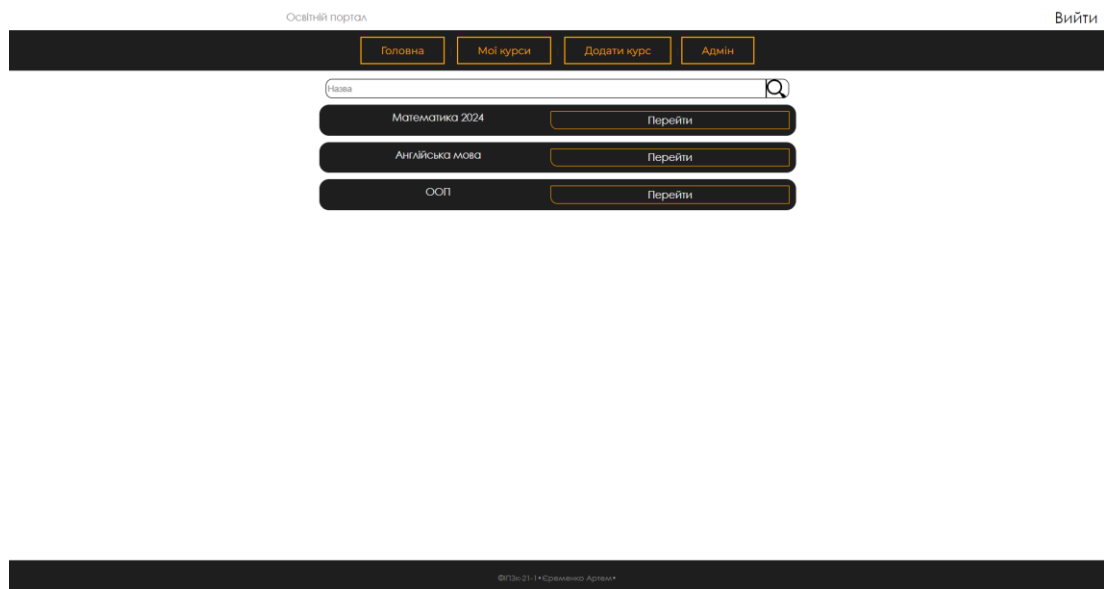
Логін

Пароль

Увійти

Рис. 3.3 – Форма авторизації

При відкритті головної сторінки, відображаються усі відкриті курси, у шапці сайту знаходяться посилання з доступом до різного функціоналу, кнопки з'являються за на основі ролі користувача, наприклад у студента не буде можливості перейти до створення курсу. Також присутній функціонал пошуку по назві (рис 3.4).



Освітній портал

Вийти

Головна

Міі курси

Додати курс

Адмін

Пошук

Математика 2024

Перейти

Англійська мова

Перейти

ООП

Перейти

ВПС-21 "Єременко Артем"

Рис. 3.4 – Головна сторінка

Після переходу до курсу, відображається його вміст, кнопки з налаштуваннями та в низу доступ до архіву та приєднання до курсу (рис 3.5).

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

Оцінки	Закрити курс	Додати користувачів
Перевірити здані завдання	Редагувати вміст курсу	Додати навчальний план

Математика 2024

Навчальний план
Завантажити файл

Лабораторна робота 1

Додаткова інформація.

Завантажити файл

Перездати завдання

Лабораторна робота 2

Додаткова інформація.

Завантажити файл

Здати завдання

Рис. 3.5 – Сторінка курсу

При натисканні на кнопку «Оцінки», відкривається форма з вибором групи, та методом оцінювання (рис 3.6).

Вибір групи

Назва Групи:

Метод оцінювання:

Обрати

Рис. 3.6 – Вибір групи

Після вибору необхідних параметрів відображається таблиця з оцінками, червоним кольором позначені роботи які чекають перевірки (рис 3.7).

	Лабораторна робота 1	Лабораторна робота 2	Лабораторна робота 3	Лабораторна робота 4	Лабораторна робота 5	Лабораторна робота 6	Сума балів
Єременко Артем Ігорович	5	0					5
Петров Петро Петрович	5						5

Рис. 3.7 – Таблиця з оцінками

При натисканні на кнопку «Перевірити здані завдання», відкривається сторінка, яка відображає усі завдання, які чекають на перевірку (рис 3.8).

Лабораторна робота 2 Єременко Артем Ігорович

Лабораторна робота 3 Єременко Артем Ігорович

Лабораторна робота 4 Єременко Артем Ігорович

Рис. 3.8 – Список зданих завдань

Після переходу до певного завдання, відкривається форма з інформацією про користувача, який здав роботу, до якого блоку дана робота відноситься, кнопка для завантаження файлу, та поле для виставлення оцінки (рис 3.9).

Перевірити завдання

Завдання:
Лабораторна робота 2

Студент:
Єременко Артем Ігорович

Файл:

Оцінка

Рис. 3.9 – Перевірка зданих завдань

Після натискання на кнопку закриття курсу, його тип зміниться на закритий, тобто доступ до нього можливо отримати, лише якщо викладач приєднає користувача, також кнопка зміниться на «Відкрити курс».

При натисканні на кнопку «Редагувати вміст курсу», відкривається форма для наповнення курсу, можна змінити назву курсу, інформацію про роботи, додати нові роботи та видалити старі (рис 3.10).

Редагування курсу

Рис 3.10 – Редагування курсу

При натисканні на кнопку «Додати користувачів», відкривається форма для вибору груп, користувачі з яких автоматично будуть додані до курсу (рис 3.11).

Приєднати групу до курсу

Рис. 3.11 – Приєднання користувачів

При натисканні на кнопку «Додати навчальний план», відкривається форма з файлом, доданий навчальний план буде окремо відображений на початку курсу (рис 3.12).

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

Завантажити навчальний план

Файл:

Choose File
No file chosen

Завантажити

Рис. 3.12 – Завантаження навчального плану

При натисканні кнопки «Мої курси», відкривається список курсів, до яких приєднаний користувач. Сірим кольором позначені закриті курси, при натисканні на кнопку «Переглянути оцінки», відкривається таблиця з оцінками самого користувача, таким чином студенти не зможуть переглянути оцінки інших студентів (рис 3.13).

Англійська мова	Перейти	Переглянути оцінки
ООП	Перейти	Переглянути оцінки
2	Перейти	Переглянути оцінки
Математика 2024	Перейти	Переглянути оцінки

Рис. 3.13 – Мої курси

При натисканні на кнопку «Адмін», відкривається сторінка адміністратора, де він може додавати та редагувати користувачів та групи (рис 3.14).

Групи	Додати	Переглянути
Користувачі	Додати	Переглянути

Рис. 3.14 – Сторінка адміністратора

Після натискання на кнопку «Переглянути» у пункті «Користувачі», відкривається список усіх користувачів, з можливістю фільтрувати за групами. Усіх користувачів можна редагувати, але видалити можна лише викладачів та студентів (рис 3.15).

Групи

Єременко Артем Ігорович	Редагувати
Петров Петро Петрович	Редагувати Видалити

Рис. 3.15 – Сторінка перегляду користувачів

При натисканні на кнопку «Редагувати», відкривається форма для редагування даних про користувача, форма додавання нових користувачів виглядає так само (рис 3.16).

Оновлення користувача

Логін

Artish2

Пароль

Повторіть пароль

ПІБ

Петров Петро Петрович

ІПЗк-21-1

Роль викладача

☐ Так

☒ Ні

Завантажити

Рис. 3.16 – Редагування користувача

При натисканні на кнопку «Переглянути» у пункті «Групи», відкривається список усіх груп з можливістю їх видалення (рис 3.17).

ІПЗк-22-1	Видалити
ІПЗк-21-1	Видалити
ПЛ-4	Видалити
ПЛ-3	Видалити
КН-21	Видалити

Рис. 3.17 – Перегляд груп

При натисканні на кнопку «Додати курс», яка знаходиться на шапці сайту, відкривається форма для створення курсу (рис 3.18).

Створення курсу

Назва Курсу:

Назва:

Текст:

Файл:

Choose File

No file chosen

Лекційний матеріал

☐ Так
 ☒ Ні

Додати блок

Видалити блок

Завантажити

Рис. 3.18 – Створення курсу

Також у кожного курсу є функціонал архіву. Для того щоб отримати доступ до нього, необхідно натиснути на кнопку «Архів», яка знаходиться у низу сторінки, під час перегляду вмісту курсу (рис 3.19).

Архів

Дата завершення курсу

05 / 31 / 2024

Завантажити

Переглянути оцінки за

2024-05-30

Метод оцінювання:

Середнє арифметичне

Обрати

Рис. 3.19 – Архів

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

Після налаштування дати завершення курсу, усі оцінки будуть помічені, при завершенні курсу, усіх користувачів автоматично від'єднає та перенесе до архіву, тоді можна буде змінити дату на іншу, наприклад на кінець наступного семестру. Також на даній сторінці можна переглянути оцінки за обраний період часу.

Висновки до третього розділу

У даному розділі було проаналізовано та спроектовано фізичну структуру системи освітнього порталу за допомогою діаграм компонентів та розгортання. Після чого було підготовлено та налаштовано систему до розташування на сервері. Далі було описано процес тестування основних функціональних компонентів системи. Також було описано користувацький інтерфейс програмного продукту та порядок роботи з ним.

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				46
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

При виконанні кваліфікаційної роботи у першому розділі досліджувалося технічне завдання до проектованої системи освітнього порталу. Також було розглянуто та проаналізовано декілька аналогів розроблюємого продукту. За допомогою засобів моделювання було створено діаграму варіантів використання, також були проаналізовані вимоги до програмного продукту.

У другому розділі було створено діаграми активності та послідовностей, що ілюструють поведінку системи та розглянуто акторів. Після чого було описано та розроблену базу даних. Далі було описано процес розробки системи освітнього порталу

У третьому розділі було спроектовано діаграми компонентів та розгортання, на основі діаграм було описано процес налаштування системи для розгортання на сервері. Після чого було описано процес тестування системи та наведено приклади тестових випадків. Також було описано користувацький інтерфейс системи та порядок роботи з ним.

В результаті виконання кваліфікаційної роботи було проаналізовано, спроектовано та розроблено програмний продукт, який відповідає функціональним вимогам поставленим у проекті. Під час виконання роботи було вивчено багато нової інформації відносно моделювання та створення програмного забезпечення, яка допоможе мені стати кращим програмістом.

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Yeremenko A. I. Core Concepts of Machine Learning / A. I. Yeremenko, S. S. Kukharonok, U. M. Yefremov. – Zhytomyr: Zhytomyr Polytechnic State University, 2024. – 214 с. – (Zhytomyr Polytechnic State University). – (Current Trends in Young Scientists' Research).
2. Освітня платформа Moodle [Електронний ресурс] – Режим доступу до ресурсу: <https://moodle.org/?lang=uk>
3. Освітня платформа Udemу [Електронний ресурс] – Режим доступу до ресурсу: <https://www.udemy.com>
4. Освітня платформа Brilliant [Електронний ресурс] – Режим доступу до ресурсу: <https://brilliant.org>
5. Використання drawio.com [Електронний ресурс] – Режим доступу до ресурсу: <https://www.drawio.com/blog/features>
6. Документація HTML [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/HTML>
7. Документація CSS [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/CSS>
8. Документація JavaScript [Електронний ресурс] – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
9. Документація Python [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.python.org/3.11/>
10. Документація Flask [Електронний ресурс] – Режим доступу до ресурсу: <https://flask.palletsprojects.com/en/2.0.x/>
11. Документація Jinja2 [Електронний ресурс] – Режим доступу до ресурсу: <https://jinja.palletsprojects.com/en/3.0.x/>
12. Документація SQLAlchemy [Електронний ресурс] – Режим доступу до ресурсу: <https://www.sqlalchemy.org>
13. Документація SQLite [Електронний ресурс] – Режим доступу до ресурсу: <https://www.sqlite.org/docs.html>

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

- 14.Документація Werkzeug [Електронний ресурс] – Режим доступу до ресурсу: <https://werkzeug.palletsprojects.com/en/3.0.x/>
- 15.Документація Python OS [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.python.org/3/library/os.html>
- 16.Документація WTForms [Електронний ресурс] – Режим доступу до ресурсу: <https://wtforms.readthedocs.io/en/3.1.x/>
- 17.Документація UML [Електронний ресурс] – Режим доступу до ресурсу: <https://www.uml.org>
- 18.Beazley D. Python Distilled (Developer's Library) / David Beazley., 2021. – 352 с.
- 19.Bhargava A. Grokking Algorithms, Second Edition / Aditya Bhargava., 2024. – 320 с. – (2).
- 20.Deite P. Intro to Python for Computer Science and Data Science: Learning to Program with AI, Big Data and The Cloud / P. Deite, H. Deitel., 2019. – 880 с.

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

Вміст файлу «main_controller.py»

```

import datetime
import os
import flask_login
from flask import Flask, render_template, url_for, request, redirect, flash,
abort, send_from_directory
from flask_login import login_required
from werkzeug.utils import secure_filename
from cfg import app, manager, db
from models.course import Course, course_users
from models.group import Group
from models.structure import Structure
from models.mark import Mark
from models.user import User
import math

@app.route('/')
def index():
    return redirect('/0')

@app.route('/<int:id>', methods=['GET', 'POST'])
@login_required
def search_courses(id=0):
    if request.method == 'GET':
        courses = Course.query.filter_by(closed=0).offset(id * 20).limit(20)
        count = Course.query.filter_by(closed=0).count()
        return render_template("index.html", count=math.ceil(count/20),
courses=courses, id=id, post=False)
    else:
        courses =
Course.query.filter_by(closed=0).filter(Course.name.contains(request.form['se
arch'])).all()
        return render_template("index.html", count=0, courses=courses, id=id,
post=True)

@app.route('/addcourse', methods=['GET', 'POST'])
@login_required
def new_course():
    if request.method == 'GET':
        return render_template("add_course.html")
    else:
        course = Course(name=request.form['coursename'],
creator=flask_login.current_user.id)
        try:
            db.session.add(course)
            db.session.commit()
        except:
            db.session.rollback()
            flash('При додаванні виникла помилка.')
            return render_template("add_course.html")
        name = ""
        number = 1
        file = ""
        lab = 0
        for item in request.form:
            if item == 'coursename':
                continue
            if item.__contains__('name'):

```

		Єременко А.І.			ІІЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        name = request.form[item]
        if item.__contains__('text'):
            if request.files['file' + str(number)].filename:
                file = request.files['file' + str(number)]
                filename = secure_filename(file.filename)
                file.save(os.path.join(app.config['UPLOAD_FOLDER'],
filename))

                file = filename
            if request.form['lab' + str(number)] == "True":
                lab = True
            else:
                lab = False
        try:
            db.session.add(Structure(number=number, name=name,
text=request.form[item], course_id=course.id,
                                file=file, lab=lab))

            db.session.commit()
            number += 1
            file = ""
        except:
            db.session.rollback()
            flash('При додаванні виникла помилка.')
            return render_template("add_course.html")
        user = User.query.filter_by(id=flask_login.current_user.id).first()
        course.addedUsers.append(user)
        db.session.commit()
        return redirect('/')

@app.route('/course/<int:id>', methods=['GET', 'POST'])
@login_required
def display_course(id=0):
    course = Course.query.filter_by(id=id).first()
    if course.time:
        today = datetime.datetime.today()
        if today > course.time:
            users = course.addedUsers
            for user in users:
                course.addedUsers.remove(user)
            if (flask_login.current_user in course.addedUsers and course.closed) or
not course.closed or \
                flask_login.current_user.id == course.creator:
                structure =
                Structure.query.filter_by(course_id=id).order_by(Structure.number).all()
                mark = []
                for el in structure:

mark.append(Mark.query.filter_by(user_id=flask_login.current_user.id,
structure_id=el.id).first())
                joined = False
                if flask_login.current_user in course.addedUsers:
                    joined = True
                return render_template("display_course.html", structure=structure,
course=course, id=id, mark=mark, joined=joined)
            else:
                return render_template('404.html'), 404

@app.route('/course/<int:id>/join')
@login_required
def join(id=0):
    course = Course.query.filter_by(id=id).first()
    user = User.query.filter_by(id=flask_login.current_user.id).first()
    try:

```

		Єременко А.І.			ІІЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        course.addedUsers.append(user)
        db.session.commit()
    except:
        db.session.rollback()
        flash('При додаванні виникла помилка.')
        return redirect('/course/' + str(id))
    return redirect('/course/'+str(id))

@app.route('/course/<int:id>/close')
@login_required
def close(id=0):
    course = Course.query.filter_by(id=id).first()
    try:
        if course.closed == 0:
            course.closed = 1
        else:
            course.closed = 0
        db.session.commit()
    except:
        db.session.rollback()
        flash('При додаванні виникла помилка.')
        return redirect('/course/' + str(id))
    return redirect('/course/'+str(id))

@app.route('/course/<int:id>/leave')
@login_required
def leave(id=0):
    course = Course.query.filter_by(id=id).first()
    user = User.query.filter_by(id=flask_login.current_user.id).first()
    try:
        course.addedUsers.remove(user)
        db.session.commit()
    except:
        db.session.rollback()
        flash('При додаванні виникла помилка.')
        return redirect('/course/' + str(id))
    return redirect('/course/'+str(id))

@app.route('/course/<int:id>/addUsers', methods=['GET', 'POST'])
@login_required
def joinUsers(id=0):
    if flask_login.current_user.access > 0:
        if request.method == 'GET':
            groups = Group.query.all()
            return render_template('add_users.html', groups=groups)
        else:
            course = Course.query.filter_by(id=id).first()
            users = []
            for el in request.form.getlist('group'):
                tmp = User.query.filter_by(group=el).all()
                if flask_login.current_user not in course.addedUsers:
                    users.append(tmp)
            try:
                for el in users:
                    course.addedUsers.append(el[0])
                db.session.commit()
            except:
                db.session.rollback()
                flash('При додаванні виникла помилка.')
                groups = Group.query.all()
                return render_template('add_users.html', groups=groups)

```

		Єременко А.І.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Левченко А.Ю.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return redirect('/course/'+str(id))
    else:
        return render_template('404.html'), 404

@app.route('/course/<int:id>/send')
@login_required
def send(id=0):
    stucture = Structure.query.filter_by(id=id).first()
    return send_from_directory(app.config["UPLOAD_FOLDER"], stucture.file)

@app.route('/course/<int:id>/send_plan')
@login_required
def send_plan(id=0):
    course = Course.query.filter_by(id=id).first()
    return send_from_directory(app.config["UPLOAD_FOLDER"], course.plan)

@app.route('/course/<int:id>/update', methods=['GET', 'POST'])
@login_required
def update_course(id=0):
    if request.method == 'GET':
        course = Course.query.filter_by(id=id).first()
        structure =
        Structure.query.filter_by(course_id=id).order_by(Structure.number).all()
        return render_template("update_course.html", course=course,
        structure=structure, length=len(structure), id=id)
    else:
        course = Course.query.get(id)
        course.name = request.form['coursename']
        try:
            db.session.commit()
        except:
            db.session.rollback()
            flash('При оновлені виникла помилка.')
            return render_template("update_course.html")
        name = ""
        structures =
        Structure.query.filter_by(course_id=id).order_by(Structure.number).all()
        maxnumber = structures[-1].number
        number = 1
        lab = 0
        for item in request.form:
            file = ""
            if item == 'coursename':
                continue
            if item.__contains__('name'):
                name = request.form[item]
            if item.__contains__('text'):
                if request.files['file' + str(number)].filename:
                    file = request.files['file' + str(number)]
                    filename = secure_filename(file.filename)
                    file.save(os.path.join(app.config['UPLOAD_FOLDER'],
filename))

                    file = filename
                    if request.form['lab' + str(number)] == "True":
                        lab = True
                    else:
                        lab = False
                try:
                    if number <= maxnumber:
                        structure = Structure.query.filter_by(course_id=id,
number=number).first()
                        structure.name = name

```

		Єременко А.І.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Левченко А.Ю.				54
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        structure.text = request.form[item]
        structure.lab = lab
        if (structure.file != file) and (file != ""):
            structure.file = file
        db.session.commit()
    else:
        db.session.add(Structure(number=number, name=name,
text=request.form[item], course_id=course.id,
                                file=file, lab=lab))
        db.session.commit()
        number += 1
    except:
        db.session.rollback()
        flash('При додаванні виникла помилка.')
        return render_template("add_course.html")
if(number-1 <= len(structures)):
    for i in range(number-1, len(structures)):
        db.session.delete(structures[i])
        db.session.commit()
return redirect('/')

@app.route('/course/<int:id>/plan', methods=['GET', 'POST'])
@login_required
def plan_course(id=0):
    if request.method == 'GET':
        return render_template("plan.html")
    else:
        course = Course.query.get(id)
        file = ""
        if request.files['file'].filename:
            file = request.files['file']
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
            file = filename
        course.plan = file
        db.session.commit()
        return redirect('/')

@app.route('/course/<int:id>/archive', methods=['GET', 'POST'])
@login_required
def archive_course(id=0):
    if request.method == 'GET':
        course = Course.query.get(id)
        times = []
        if course.time:
            structures = Structure.query.filter_by(course_id=id,
lab=True).order_by(Structure.number).all()
            for st in structures:
                tmp = Mark.query.filter_by(structure_id=st.id).all()
                for el in tmp:
                    if el.time.date() not in times:
                        times.append(el.time.date())
        return render_template("archive.html", course=course, times=times)
    else:
        course = Course.query.get(id)
        if 'date' in request.form:
            date_out = datetime.datetime(*[int(v) for v in
request.form['date'].replace('T', '-').replace(':', '-').split('-')])
            course.time = date_out
            db.session.commit()
            structures = Structure.query.filter_by(course_id=id,
lab=True).order_by(Structure.number).all()

```

		Єременко А.І.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Левченко А.Ю.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        marks = []
        for st in stuctures:
            tmp = Mark.query.filter_by(time=None,
structure_id=st.id).all()
            for el in tmp:
                marks.append(el)
        for el in marks:
            el.time = course.time
        db.session.commit()
    else:
        course = Course.query.filter_by(id=id).first()
        stuctures = Structure.query.filter_by(course_id=id,
lab=True).order_by(Structure.number).all()
        marks = []
        users_id = []
        date_out = datetime.datetime(*[int(v) for v in
request.form['times'].replace('T', '-').replace(':', '-').split('-')])
        for st in stuctures:
            tmp = Mark.query.filter_by(time=date_out,
structure_id=st.id).all()
            for el in tmp:
                marks.append(el)
        for el in marks:
            if el.user_id not in users_id:
                users_id.append(el.user_id)
        users = []
        for el in users_id:
            users.append(User.query.filter_by(id=el).first())
        marks = []
        for us in users:
            usmarks = []
            for st in stuctures:
                tmp = Mark.query.filter_by(user_id=us.id,
structure_id=st.id).first()
                usmarks.append(tmp)
            marks.append(usmarks)
        return render_template("marks_table.html", users=users,
marks=marks, stuctures=stuctures,
                                calc=request.form['calc'])

    return redirect('/')

@app.route('/course/<int:id>/delete')
@login_required
def delete_course(id=0):
    course = Course.query.get(id)
    structure = Structure.query.filter_by(course_id=id)
    try:
        db.session.delete(course)
        for el in structure:
            if el.file != "":
                os.remove(os.path.join(app.config['UPLOAD_FOLDER'], el.file))
            marks = Mark.query.filter_by(structure_id=el.id).all()
            if marks:
                for mark in marks:
                    os.remove(os.path.join(app.config['UPLOAD_FOLDER'],
mark.file))
                    db.session.delete(mark)
        db.session.delete(el)
        db.session.commit()
    except:
        db.session.rollback()
        flash('При видаленні виникла помилка.')
        return redirect('/')

```

		Єременко А.І.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Левченко А.Ю.				56
Змн.	Арк.	№ докум.	Підпис	Дата		

```

return redirect('/')

@app.errorhandler(404)
def page_not_found(e):
    return render_template('404.html'), 404

```

ДОДАТОК Б

Вміст файлу «mark_controller.py»

```

import os
import flask_login
from flask import Flask, render_template, url_for, request, redirect, flash,
abort, send_from_directory
from flask_login import login_required
from werkzeug.utils import secure_filename
from cfg import app, manager, db
from models.course import Course
from models.group import Group
from models.user import User
from models.structure import Structure
from models.mark import Mark

@app.route('/task/<int:id>', methods=['GET', 'POST'])
@login_required
def task(id=0):
    if request.method == 'GET':
        return render_template("task.html")
    else:
        structure = Structure.query.filter_by(id=id).first()
        file = ""
        mark = Mark.query.filter_by(user_id=flask_login.current_user.id,
structure_id=id).first()
        course = Course.query.filter_by(id=structure.course_id).first()
        if request.files['file'].filename:
            file = request.files['file']
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
            file = filename
        else:
            flash('Ви не додали файл')
            return render_template("task.html")
        try:
            if mark:
                os.remove(os.path.join(app.config['UPLOAD_FOLDER'],
mark.file))
                mark.file = file
                mark.remark = True
                if course.time:
                    mark.time = course.time
                db.session.commit()
            else:
                if course.time:
                    db.session.add(Mark(user_id=flask_login.current_user.id,
structure_id=id, file=file, remark=True, time=course.time))
                else:
                    db.session.add(Mark(user_id=flask_login.current_user.id,
structure_id=id, file=file, remark=True))
                db.session.commit()
        except:
            db.session.rollback()

```

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        flash('При додаванні виникла помилка.')
        return render_template("task.html")
    return redirect('/course/'+str(structure.course_id))

@app.route('/all_tasks/<int:id>')
@login_required
def allTasks(id=0):
    if flask_login.current_user.access > 0:
        course = Course.query.filter_by(id=id).first()
        structure = Structure.query.filter_by(course_id=course.id).all()
        marks = []
        for el in structure:
            tmp = Mark.query.filter_by(structure_id=el.id, remark=True,
time=course.time).all()
            if tmp:
                for eli in tmp:
                    marks.append(eli)
        return render_template("all_tasks.html", marks=marks,
structure=structure, User=User)
    else:
        return render_template('404.html'), 404

@app.route('/check_task/<int:id>', methods=['GET', 'POST'])
@login_required
def check_task(id=0):
    if flask_login.current_user.access > 0:
        if request.method == 'GET':
            mark = Mark.query.filter_by(id=id).first()
            structure =
Structure.query.filter_by(id=mark.structure_id).first()
            return render_template("check_task.html", mark=mark, User=User,
structure=structure)
        else:
            mark = Mark.query.filter_by(id=id).first()
            structure =
Structure.query.filter_by(id=mark.structure_id).first()
            mark.mark = int(request.form['mark'])
            mark.remark = False
            db.session.commit()
            return redirect("/all_tasks/"+str(structure.course_id))
    else:
        return render_template("404.html"), 404

@app.route('/check_marks/<int:id>', methods=['GET', 'POST'])
@login_required
def check_marks(id=0):
    if flask_login.current_user.access > 0:
        course = Course.query.filter_by(id=id).first()
        stuctures = Structure.query.filter_by(course_id=id,
lab=True).order_by(Structure.number).all()
        users = course.addedUsers
        if request.method == 'GET':
            groups = []
            for el in users:
                tmp = Group.query.filter_by(id=el.group).first().name
                if el.group not in groups:
                    groups.append(tmp)
            return render_template("choose_group.html", groups=groups)
        else:
            name = request.form['group']
            group = Group.query.filter_by(name=name).first()

```

		Єременко А.І.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Левченко А.Ю.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        users = User.query.filter_by(group=group.id).all()
        marks = []
        for us in users:
            usmarks = []
            for st in stuctures:
                tmp = Mark.query.filter_by(user_id=us.id,
structure_id=st.id).first()
                usmarks.append(tmp)
            marks.append(usmarks)
        return render_template("marks_table.html", users=users,
marks=marks, stuctures=stuctures, calc=request.form['calc'])
    else:
        return render_template("404.html"), 404

@app.route('/my_marks/<int:id>', methods=['GET', 'POST'])
@login_required
def my_marks(id=0):
    if flask_login.current_user.access > 0:
        stuctures = Structure.query.filter_by(course_id=id,
lab=True).order_by(Structure.number).all()
        user = flask_login.current_user
        users = [user]
        marks = []
        usmarks = []
        for st in stuctures:
            tmp = Mark.query.filter_by(user_id=user.id,
structure_id=st.id).first()
            usmarks.append(tmp)
        marks.append(usmarks)
        return render_template("marks_table.html", users=users, marks=marks,
stuctures=stuctures)
    else:
        return render_template("404.html"), 404

@app.route('/task/<int:id>/send')
@login_required
def send_task(id=0):
    mark = Mark.query.filter_by(id=id).first()
    return send_from_directory(app.config["UPLOAD_FOLDER"], mark.file)

```

ДОДАТОК В

Вміст файлу «user_controller.py»

```

import os

import sqlalchemy.exc
import flask_login
from flask import flash, request, redirect, render_template, url_for, abort
from cfg import manager, app, db
from models.course import Course
from models.mark import Mark
from models.user import User
from werkzeug.security import check_password_hash, generate_password_hash
from flask_login import LoginManager, UserMixin, login_user, login_required,
logout_user
from models.LoginForm import LoginForm
from models.group import Group

@app.route('/login', methods=['GET', 'POST'])
def login_page():

```

		Єременко А.І.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Левченко А.Ю.				59
Змн.	Арк.	№ докум.	Підпис	Дата		

```

form = LoginForm()
if form.validate_on_submit():
    if form.login.data and form.password.data:
        user = User.query.filter_by(login=form.login.data).first()
        if user and check_password_hash(user.password,
str(form.password.data)):
            login_user(user)
            next_page = request.args.get('next')
            if next_page:
                return redirect(next_page)
            else:
                return redirect('/')
        else:
            flash('Помилка у введеннях даних')
            return render_template('login.html', form=form)
    else:
        flash('Поля повинні бути заповненими')
        return render_template('login.html', form=form)
else:
    return render_template('login.html', form=form)

@app.route('/logout', methods=['GET', 'POST'])
@login_required
def logout():
    logout_user()
    return redirect(url_for('index'))

@app.route('/cabinet')
@login_required
def cabinet():
    user = User.query.filter_by(id=flask_login.current_user.id).first()
    courses = Course.query.filter(Course.addedUsers.contains(user)).all()
    return render_template('cabinet.html', courses=courses)

@manager.user_loader
def load_user(user_id):
    return User.query.get(user_id)

```

ДОДАТОК Г

Вміст файлу «admin_controller.py»

```

import os
import flask_login
import sqlalchemy
from flask import Flask, render_template, url_for, request, redirect, flash,
abort, send_from_directory
from flask_login import login_required
from werkzeug.security import generate_password_hash
from werkzeug.utils import secure_filename
from cfg import app, manager, db
from models.course import Course
from models.group import Group
from models.structure import Structure
from models.mark import Mark
from models.user import User

@app.route('/admin')
@login_required
def admin():

```

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    if flask_login.current_user.access == 2:
        return render_template("admin.html")
    else:
        return render_template('404.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    if flask_login.current_user.access == 2:
        if request.method == 'GET':
            return render_template('register.html')
        else:
            if not (request.form['login'] or request.form['password1'] or
request.form['password2'] or request.form[
    'pib']):
                flash('Поля повинні бути заповненими')
                return render_template('register.html')
            elif request.form['password1'] != request.form['password2']:
                flash('Різні паролі')
                return render_template('register.html')
            else:
                hash_pwd = generate_password_hash(request.form['password1'])
                access = 0
                if request.form['access'] == "True":
                    access = 1
                new_user = User(login=request.form['login'],
password=hash_pwd, pib=request.form['pib'],
                    access=access)
                try:
                    db.session.add(new_user)
                    db.session.commit()
                except sqlalchemy.exc.IntegrityError:
                    db.session.rollback()
                    flash('Користувач вже зареєстрований')
                    return render_template('register.html')
                return render_template('admin.html')
    else:
        return render_template('404.html'), 404

@app.route('/users')
def all_users():
    if flask_login.current_user.access == 2:
        users = User.query.all()
        groups = Group.query.all()
        return render_template('users.html', users=users, groups=groups)

@app.route('/users/<int:id>')
def group_users(id=0):
    if flask_login.current_user.access == 2:
        users = User.query.filter_by(group=id).all()
        groups = Group.query.all()
        return render_template('users.html', users=users, groups=groups)
    else:
        return render_template('404.html'), 404

@app.route('/update_user/<int:id>', methods=['GET', 'POST'])
def update_users(id=0):
    if flask_login.current_user.access == 2:
        user = User.query.filter_by(id=id).first()
        groups = Group.query.all()
        if request.method == 'GET':

```

		Єременко А.І.			ІІЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return render_template('update_user.html', user=user,
groups=groups)
    else:
        if not (request.form['login'] or request.form['pib']):
            flash('Поля повинні бути заповненими')
            return render_template('update_user.html', user=user,
groups=groups)
        elif request.form['password1'] != request.form['password2']:
            flash('Пізнi паролі')
            return render_template('update_user.html', user=user,
groups=groups)
        else:
            if request.form['password1']:
                hash_pwd =
generate_password_hash(request.form['password1'])
            else:
                hash_pwd = user.password
                access = 0
            if request.form.__contains__('access'):
                if request.form['access'] == "True":
                    access = 1
            else:
                access = 2
            user.login = request.form['login']
            user.password = hash_pwd
            user.access = access
            user.pib = request.form['pib']
            user.group = request.form['group']
            try:
                db.session.commit()
            except sqlalchemy.exc.IntegrityError:
                db.session.rollback()
                flash('Користувач вже зареєстрований')
                return render_template('update_user.html', user=user,
groups=groups)
            return render_template('admin.html')
    else:
        return render_template('404.html'), 404

@app.route('/delete_user/<int:id>')
def delete_users(id=0):
    if flask_login.current_user.access == 2:
        user = User.query.filter_by(id=id).first()
        if Mark.query.filter_by(user_id=id).all():
            for el in Mark.query.filter_by(user_id=id).all():
                os.remove(os.path.join(app.config['UPLOAD_FOLDER'], el.file))
                db.session.delete(el)
            try:
                db.session.delete(user)
                db.session.commit()
            except:
                db.session.rollback()
                flash('Виникла помилка')
                render_template('admin.html')
            return redirect('/users')
    else:
        return render_template('404.html'), 404

@app.route('/group')
@login_required
def groups():

```

		Єременко А.І.			ІІЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if flask_login.current_user.access == 2:
    group = Group.query.all()
    return render_template("groups.html", groups=group)
else:
    return render_template('404.html'), 404

@app.route('/group/add', methods=['GET', 'POST'])
@login_required
def newGroup():
    if flask_login.current_user.access == 2:
        if request.method == 'GET':
            return render_template('add_group.html')
        else:
            try:
                db.session.add(Group(name=request.form['group']))
                db.session.commit()
            except:
                db.session.rollback()
                flash('При додаванні виникла помилка.')
                return render_template('add_group.html')
            return render_template('admin.html')
    else:
        return render_template('404.html'), 404

@app.route('/group/delete/<int:id>')
@login_required
def delGroup(id=0):
    if flask_login.current_user.access == 2:
        allGroups = Group.query.all()
        group = Group.query.filter_by(id=id).first()
        users = User.query.filter_by(group=id).all()
        for el in users:
            el.group = None
        try:
            db.session.delete(group)
            db.session.commit()
        except:
            db.session.rollback()
            flash('При видаленні виникла помилка')
            return render_template('groups.html', groups=allGroups)
        return render_template('groups.html', groups=allGroups)
    else:
        return render_template('404.html'), 404

```

		Єременко А.І.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Левченко А.Ю.				63
Змн.	Арк.	№ докум.	Підпис	Дата		