

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕКОНОМІКИ І МЕНЕДЖМЕНТУ
Кафедра кібернетики та інформатики

КВАЛІФІКАЦІЙНА РОБОТА

освітній ступінь – «Бакалавр»

на тему:

**«РОЗРОБКА ВЕБЗАСТОСУНКУ ОРГАНІЗАЦІЇ
ДИСТАНЦІЙНИХ КУРСІВ
НЕФОРМАЛЬНОЇ ОСВІТИ»**

Виконав:

студент спеціальності

126 «Інформаційні системи та технології»

ОП «Інформаційні системи та технології»

Липченко Іван Володимирович

Науковий керівник:

Долгих Яна Володимирівна

доцент кафедри кібернетики та інформатики

Рецензент:

Коломієць Світлана Володимирівна

старший викладач кафедри економічної кібернетики
Сумського державного університету

Суми – 2024

СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

Факультет	Економіки і менеджменту
Кафедра	Кібернетики та інформатики
Освітній ступінь	«Бакалавр»
Спеціальність	<u>126 «Інформаційні системи та технології»</u> <u>ОП «Інформаційні системи та технології»</u>

Затверджую:
Завідувач кафедри Світлана АГАДЖАНОВА

« _____ » « _____ » 2024 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Лапченко Івану Володимировичу

1. Тема роботи: “Розробка вебзастосунку організації дистанційних курсів неформальної освіти”
2. База практичного дослідження: ТОВ “GoITeens”, Сумське відділення, м. Київ.

керівник роботи: Долгих Яна Володимирівна, к.е.н., доцент

затверджена наказом по університету від « _____ » _____

3. Строк подання студентом закінченого проєкту (роботи) «28» червня 2024 року

4. Вихідні дані до проєкту (роботи):
результати аналітичних досліджень зарубіжних та вітчизняних вчених, інтернет-джерела,

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

- виконати постановку задачі;
- вибрати середовище для реалізації;
- розробити технічне завдання;
- розробити програмний продукт відповідно до поставлених задач;

6. Дата видачі завдання: « _____ » _____ 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів проєкту	Примітка
1	Визначення об'єкту, предмету дослідження, формулювання мети та задач кваліфікаційної роботи, складання плану	січень 2024 р.	
2	Підбір та вивчення літературних джерел, технічної документації	січень 2024 р.	
3	Розробка технічного завдання	січень 2024 р.	
4	Розробка вебзастосунку	січень-травень 2024 р.	
5	Оформлення пояснювальної записки	до 30 травня 2024 р.	
6	Оформлення презентації	до 1 червня 2024 р.	
7	Опрацювання зауважень керівника, перевірка на автентичність	2 червня 2024 р.	
8	Підготовка доповіді та її попередній захист	6 червня 2024 р.	
9	Перевірка з академічної доброчесності кваліфікаційної роботи	10 червня 2024 р.	
10	Представлення кваліфікаційної роботи на кафедрі, деканат	23 червня 2024 р.	
11	Захист кваліфікаційної роботи	28 червня 2024р.	

Студент

_____ (підпис)

Іван ЛАПЧЕНКО

Керівник роботи

_____ (підпис)

Яна ДОЛГІХ

Перевірку на автентичність проведено

_____ (підпис)

Надія БАРАНІК

Кваліфікаційна робота допущена до захисту

_____ (підпис)

Маргарита ЛИШЕНКО

АНОТАЦІЯ

Лапченко І.В. Розробка вебзастосунку організації дистанційних курсів неформальної освіти.

Кваліфікаційна робота зі спеціальності 126 «Інформаційні системи та технології» ОП Інформаційні системи та технології, СНАУ, Суми-2024р. – Рукопис.

Ця робота присвячена створенню вебзастосунку організації дистанційних курсів неформальної освіти. Основною метою є розробка ефективного інструменту для викладачів та студентів, що дозволить точно оцінювати рівень знань та планувати відповідні освітні заходи.

У дослідженні проведено огляд існуючих платформ для дистанційної освіти, а також визначені ключові проблеми, які мають найпопулярніші з платформ для дистанційної освіти. Запропоновано архітектуру вебзастосунку, визначено мету та засоби реалізації.

Для реалізації вебзастосунку були використані сучасні технології програмування та баз даних, що забезпечують його швидке та надійне функціонування. Розроблено модулі для введення даних, виконання розрахунків, створення звітів та аналізу результатів. Проведено автоматизоване тестування вебзастосунку.

Результати тестування підтвердили коректність функціонування вебзастосунку та його відповідність вимогам і очікуванням користувачів. Отримані результати дозволяють зробити висновок про доцільність впровадження розробленого вебзастосунку організації дистанційних курсів неформальної освіти. Він сприятиме росту якості надання освітніх послуг ІТ-академією GoITeans, а також забезпечить зручний доступ до необхідної інформації для прийняття обґрунтованих рішень щодо планування освітнього процесу.

Ключові слова: Python, HTML, JS, ORM, Docker, вебзастосунок.

SUMMARY

Lapchenko I.V. Development of a web application for the organization of distance courses of non-formal education.

Qualification work on specialty 126 "Information systems and technologies" OP Information systems and technologies, SNAU, Sumy-2024. - Manuscript.

This work is devoted to the creation of a web application for the organization of distance courses of non-formal education. The main goal is to develop an effective tool for teachers and students that will allow to accurately assess the level of knowledge and plan appropriate educational activities.

The study provides an overview of existing distance education platforms and identifies key challenges faced by the most popular distance education platforms. The architecture of the web application is proposed, the purpose and means of implementation are determined.

To implement the web application, modern programming and database technologies were used, which ensure its fast and reliable functioning. Modules for entering data, performing calculations, creating reports and analyzing results have been developed. Automated testing of the web application was carried out.

The test results confirmed the correct functioning of the web application and its compliance with the requirements and expectations of users. The obtained results allow us to draw a conclusion about the expediency of implementing the developed web application for the organization of distance courses of informal education. It will contribute to the growth of the quality of the provision of educational services by the GoITeens IT Academy, as well as provide convenient access to the necessary information for making informed decisions regarding the planning of the educational process.

Keywords: Python, HTML, JS, ORM, Docker, web application.

ЗМІСТ

ВСТУП.....	6
РОЗДІЛ 1 Аналіз предметної області.....	10
1.1 Аналіз існуючих платформ, технологій та напрямків розвитку неформальної дистанційної освіти.....	10
1.2 Огляд методології розробки вебзастосунків	14
РОЗДІЛ 2 Постановка задачі та вибір засобів реалізації	18
2.1 Постановка задачі.....	18
2.2 Вибір засобів реалізації	Error! Bookmark not defined.
РОЗДІЛ 3 Проектування та реалізація вебзастосунку організації дистанційних курсів неформальної освіти.....	23
3.1 Розробка функціональних діаграм вебзастосунку для організації дистанційних курсів неформальної освіти	23
3.2 Проектування бази даних вебзастосунку для організації дистанційних курсів неформальної освіти	25
3.3 Розробка прототипу інтерфейсу користувача	28
3.4 Фізична реалізація бази даних та програмна реалізація інтерфейсу користувача.....	30
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	60
ДОДАТКИ.....	62
Додаток А Технічне завдання	63
Додаток Б Планування робіт.....	69
Додаток В Копії публікацій	72
Додаток Г Лістинги програм.....	78

ВСТУП

Традиційні формати навчання не можуть повністю задовольнити потреби у освіті, що зростають. Важливу роль у трансформації навчання відіграє неформальна освіта, яка пропонує гнучкий та індивідуальний підхід до набуття знань та навичок, а також дає можливість здобувачам будь-якого віку, з різним досвідом та потребами набувати та вдосконалювати професійні компетенції.

Сучасні тенденції в освіті свідчать про стрімке зростання попиту на дистанційне навчання. В умовах пандемії COVID-19, військового стану потреба у вебзастосунках, які забезпечують організацію дистанційних курсів, значно зросла. Такі платформи як Moodle, Schoology та інші довели свою ефективність, але існує попит на створення нових, більш адаптованих і зручних рішень для різних категорій користувачів. Отже, розробка вебзастосунку з використанням сучасних технологій програмування, який дозволить ефективно організовувати дистанційні курси неформальної освіти, є *актуальним*.

Метою дослідження є розробка вебзастосунку для організації дистанційних курсів неформальної освіти, який характеризується зручністю використання, широкими функціональними можливостями та адаптивністю до потреб різних користувачів. Вебзастосунок розроблюється на замовлення онлайн ІТ-академії GoITeens.

Завдання дослідження:

- проаналізувати існуючі платформи дистанційного навчання, визначити їх переваги та недоліки;
- спроектувати архітектуру вебзастосунку для організації дистанційних курсів неформальної освіти;
- спроектувати базу даних вебзастосунку для організації дистанційних курсів неформальної освіти;
- реалізувати архітектуру вебзастосунку, застосовуючи сучасні технології програмування;
- розробити прототип інтерфейсу користувача;

- фізично реалізувати базу даних вебзастосунку для організації дистанційних курсів неформальної освіти;
- протестувати функціональність створеного вебзастосунку;
- оцінити результати розробки та надати рекомендації щодо її подальшого удосконалення.

Об'єктом дослідження є процес розробки вебзастосунку для організації дистанційних курсів у сфері неформальної освіти.

Предметом дослідження є інформаційні технології розробки вебзастосунку для організації дистанційних курсів у сфері неформальної освіти.

Наукова новизна полягає у розробці вебзастосунку, який забезпечить ефективну організацію дистанційних курсів неформальної освіти, враховуючи сучасні вимоги користувачів і технологічні тенденції. Розробка ініційована з причини відсутності певних рішень на ринку, тому є унікальною та науково цінною з технічної точки зору. Запропоноване рішення орієнтовано на зручність використання, інтуїтивно зрозумілий інтерфейс, адаптивний дизайн та можливість інтеграції з іншими сервісами.

Практичне значення одержаних результатів полягає у можливості використання вебзастосунку для організації дистанційних курсів у різних сферах неформальної освіти, включаючи професійні курси, тренінги та самостійне навчання.

У дослідженні застосовані такі методи:

- використано метод експерименту, котрий відноситься до емпіричних методів, що застосовувався під час тестування вебзастосунку;
- метод моделювання, котрий є комплексним методом наукового дослідження, був використаний для створення проєкту вебзастосунку та його бази даних;
- метод абстрагування, що також відноситься до комплексних методів, застосований при аналізі існуючих на ринку вебзастосунків;
- метод сходження від абстрактного до конкретного застосовано при аналізі технологій, що були використані у процесі розробки вебзастосунку. Даний метод

наукового дослідження відноситься до теоретичних.

Апробація матеріалів дослідження здійснювалася на міжнародній науково-практичній конференції “*Технології XXI сторіччя*”, 20-22 листопада 2023 р., Суми: СНАУ та щорічної науково-практичної конференції викладачів, аспірантів та студентів Сумського національного аграрного університету, 14-16 травня 2024 р., Суми: СНАУ, а також на Всеукраїнському конкурсі студентських наукових робіт у галузі «Інформаційно-комунікаційні технології в освіті» (МДПУ імені Богдана Хмельницького, 5 березня -19 квітня 2024р.).

Структура і обсяг роботи. Кваліфікаційна робота складається із вступу, трьох розділів, висновків та пропозицій, списку використаних джерел з 20 найменувань, додатків. Основний текст викладений на 88 сторінках комп’ютерного тексту, робота містить 20 рисунків, 4 додатки.

.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз існуючих платформ, технологій та напрямків розвитку неформальної дистанційної освіти

Дистанційна освіта стає все більш популярною. За аналізом існуючих платформ дистанційної освіти визначено їх переваги та недоліки. Переваги платформ дистанційної освіти:

- доступні з будь-якого місця, де є доступ до Інтернету;
- користувачі можуть навчатися у власному темпі, в зручний для них час;
- пропонують різноманітні інтерактивні методи навчання;
- надають можливість адаптувати навчальний контент до індивідуальних потреб користувачів.

Дослідження виявило ряд таких недоліків застосування платформ дистанційної освіти:

- платформи можуть бути платними;
- користувачі можуть стикатися з технічними проблемами, такими як недостатній функціонал, недоступність деяких функцій або платформи;
- користувачам може не вистачати мотивації для самостійного навчання.

Наведемо характеристику існуючих платформ навчання.

Moodle – безкоштовна платформа з відкритим кодом, яка використовується для організації онлайн-курсів. Moodle має широкий спектр функцій, таких як форуми, вікторини, чати, вебінари та багато іншого.

Canvas – платформа для онлайн-навчання, яка використовується в багатьох навчальних закладах. Canvas має зручний інтерфейс та пропонує широкий спектр функцій, таких як створення курсів, оцінювання знань, спілкування з учнями та багато іншого.

Udemy – платформа для онлайн-курсів, де можна знайти курси з будь-якої

теми. Udemy пропонує платні та безкоштовні курси, а також можливість створювати власні курси.

Coursera – платформа для онлайн-курсів, яка пропонує курси від кращих університетів світу. Coursera пропонує платні та безкоштовні курси, а також можливість отримати сертифікат про закінчення курсу [1].

EdX – платформа для онлайн-курсів, яка пропонує курси від кращих університетів світу. EdX пропонує платні та безкоштовні курси, а також можливість отримати сертифікат про закінчення курсу.

Нижче (див. рисунок 1.1) наведені функціональні можливості зазначених вище платформ дистанційного навчання.

Платформа	Функціональні можливості
Moodle	Створення та управління онлайн-курсами, додавання різноманітних навчальних матеріалів, використання інтерактивних елементів, створення та оцінювання завдань, спілкування з учнями, відстеження прогресу навчання
Canvas	Створення та управління онлайн-курсами, додавання різноманітних навчальних матеріалів, використання інтерактивних елементів, створення та оцінювання завдань, спілкування з учнями, відстеження прогресу навчання, інтеграція з іншими системами
Udemy	Доступ до великої бібліотеки онлайн-курсів з різних тем, платні та безкоштовні курси, можливість створювати власні курси, спілкування з інструкторами та іншими учнями, сертифікати про закінчення курсів
Coursera	Онлайн-курси від кращих університетів світу, платні та безкоштовні курси, можливість отримати сертифікат про закінчення курсів, спеціалізації та магістерські програми
EdX	Онлайн-курси від кращих університетів світу, платні та безкоштовні курси, можливість отримати сертифікат про закінчення курсів, спеціалізації та магістерські програми

Рисунок 1.1 – Порівняльна характеристика платформ для дистанційного навчання

При розробці вебзастосунків важливим є вибір оптимальних технологій. Ефективність, швидкість, масштабованість, безпека та зручність інтерфейсу – це лише кілька аспектів, на які впливають обрані технології. Наведемо огляд

технології за функціональними групами.

Front-end:

- HTML використовується для створення структури вебсторінок [2];
- CSS використовується для стилізації вебсторінок [3];
- JavaScript використовується для додавання інтерактивності на вебсторінки;
- React, Angular, Vue.js – це JavaScript-фреймворки, які використовуються для створення складних вебзастосунків [4].

Back-end:

- універсальна мова програмування Python, яка часто використовується для розробки вебзастосунків;
- популярна мова програмування Java, яка використовується для розробки масштабованих вебзастосунків;
- мова програмування PHP, яка використовується для розробки динамічних вебсторінок;
- фреймворк програмування Ruby on Rails, який використовується для розробки вебзастосунків на Ruby;
- платформа JavaScript Node.js, яка використовується для розробки веб-серверів.

Бази даних:

- популярна реляційна база даних MySQL, яка часто використовується для вебзастосунків;
- реляційна база даних з відкритим кодом PostgreSQL, яка використовується для вебзастосунків;
- нереляційна база даних MongoDB, яка використовується для вебзастосунків, які потребують гнучкості.

Хмарні платформи:

- Amazon Web Services (AWS) є лідером з широким спектром послуг, глобальним охопленням та активною спільнотою;
- Google Cloud Platform (GCP) є інноваційною платформою з фокусом на аналітиці даних, машинному навчанні та масштабованих обчислювальних

ресурсах[5];

- Microsoft Azure є комплексною платформою з різноманітними сервісами, сильною інтеграцією з продуктами Microsoft та рішеннями гібридної хмари.

Вибір технологій для розробки вебзастосунків дистанційної освіти визначає якість та успіх навчального процесу. Оптимальні технології мають враховувати потреби студентів та викладачів, забезпечуючи зручність, ефективність та безпеку.

Серед сучасних напрямків розвитку неформальної освіти відмітимо такі:

- персоналізація, яка враховує індивідуальні потреби та інтереси здобувачів та реалізується через адаптивні навчальні програми, персоналізовані рекомендації та оцінювання на основі компетенцій;

- гнучкість, яка враховує потреби здобувачів з різним способом життя та реалізується через онлайн-курси, мобільні навчальні програми, вечірні та weekend-курси;

- доступність освіти для здобувачів з різним доходом, освітою та географічним розташуванням, що реалізується через безкоштовні онлайн-курси, стипендії та гранти, а також курси, які проводяться в громадських центрах та бібліотеках [6];

- мікронавчання, яке передбачає розбиття навчального контенту на короткі, легко засвоювані модулі та може бути корисним для здобувачів, які мають обмежений час або які хочуть зосередитися на конкретних навичках;

- соціальне навчання, яке передбачає співпрацю та взаємодію з іншими учнями та може бути корисним для людей, які хочуть отримати зворотний зв'язок, поділитися своїми ідеями та побудувати мережу зв'язків;

- практичне навчання, яке передбачає застосування знань та навичок у реальних проєктах та може бути корисним для людей, які хочуть отримати досвід роботи та підготуватися до роботи [7].

Проте, наразі неформальна освіта стикається з рядом таких викликів:

- не фінансується з боку держави, отже повністю залежна від нестабільних приватних інвестицій, що може призвести до нестачі ресурсів, низької оплати праці викладачів та обмеженого доступу до курсів;

- не завжди регулюється так само, як формальна освіта, що може призвести до неякісних курсів та низьких стандартів;
- не завжди визнається роботодавцями або навчальними закладами, що може ускладнити людям отримання роботи або зарахування до вищого навчального закладу;
- має ускладнення доступу через те, що не всі здобувачі мають доступ до Інтернету або комп'ютерів;
- не всі мотивовані до самостійного навчання, що може призвести до низьких показників успішності.

Отже, неформальна освіта має великий потенціал, однак стикається з низкою проблем, які необхідно вирішити. Деякі можливі шляхи вирішення цих проблем:

- збільшення державного та приватного фінансування неформальної освіти;
- розробка стандартів якості для неформальної освіти;
- сприяння визнанню неформальної освіти роботодавцями та навчальними закладами;
- розширення доступу до Інтернету та комп'ютерів;
- розробка стратегій мотивації людей до самостійного навчання.

Важливо зазначити, що це лише деякі з можливих шляхів вирішення проблем, з якими стикається неформальна освіта. Існує багато інших можливих підходів, і найкращий шлях вирішення проблем буде залежати від конкретного контексту.

Неформальна освіта – це динамічна та швидко зростаюча сфера. Тренди та виклики, з якими вона стикається, постійно змінюються. Важливо досліджувати та вивчати цю сферу для надання здобувачам можливості ефективно вчитися та розвиватися.

1.2 Огляд методології розробки вебзастосунків

Розробка вебзастосунку для неформальної освіти здійснюється за гібридною методологією, що комбінує підходи Agile та Waterfall.

На першому етапі визначаються основні вимоги та завдання проєкту.

Наступним етапом є аналіз існуючих платформ для дистанційної освіти та визначаються ключові функціональні та нефункціональні вимоги. Цей етап дозволяє зрозуміти основні вимоги користувачів та визначити напрямок розробки.

На етапі проєктування розробляється архітектурний дизайн вебзастосунку. Особлива увага приділяється використанню Helm та Kubernetes для забезпечення масштабованості та стабільності системи. В даній роботі ці технології гратимуть теоретичну роль, адже фізичне розгортання вебзастосунку потребує фінансових інвестицій для придбання обчислювальних потужностей. Також визначається структура бази даних та дизайн інтерфейсу користувача.

На етапі реалізації використовується мова програмування Python для написання логіки вебзастосунку. Крім того, відбувається інтеграція існуючого коду з розробленими функціями тестування для забезпечення якості продукту.

Етап тестування включає в себе використання різних методів, таких як модульні та інтеграційні тести. Оцінюються результати тестування, виявлені проблеми та застосовані вирішення. Також проводиться тестування прийняття користувача шляхом опитування осіб, що приймали участь у тестуванні застосунку.

Після введення в експлуатацію здійснюється оптимізація продукту. Виявлені вузькі місця у продуктивності виправляються, алгоритми та код постійно оптимізуються для досягнення найвищої ефективності.

Наведений вище методичний підхід визначається гнучкістю управління процесом розробки, інкрементальним підходом до впровадження нового функціоналу та стратегічним тестуванням для забезпечення високої якості розробленого вебзастосунку.

Виходячи з аналізу технологій, що можуть бути використані при розробці вебзастосунку та враховуючи вимоги до вебзастосунку, для реалізації вебзастосунку будуть використані такі технології:

- Python для написання логіки вебзастосунку, оскільки Python відомий своєю простотою та ефективністю, що сприяє швидкому розвитку та підтримці

коду;

- технологія Helm для керування пакунками (charts) Kubernetes, що дозволяє просто та ефективно розгортати, оновлювати та керувати конфігурацією застосунку в середовищі Kubernetes;

- Kubernetes для оркестрації контейнерів та управління розгортанням та масштабуванням вебзастосунку, що забезпечує високу доступність та стабільність системи;

- SQLite в якості системи управління базами даних, оскільки ідеально підходить для невеликих та середніх проєктів.

В якості методології буде використано гібрид Agile та Waterfall, щоб поєднати переваги гнучкості Agile з деталізацією та структурою Waterfall. Це дозволить ефективно управляти процесом розробки, зберігаючи важливість якості та строгості плану.

Буде використана стратегія тестування, яка включає модульне тестування, інтеграційне тестування та тестування прийняття користувача. Це забезпечить високий рівень якості та надійності вебзастосунку.

Обрані інструменти, технології та методології допомагають забезпечити ефективний розвиток вебзастосунку, враховуючи особливості його функціоналу та інфраструктури за рахунок певних переваг кожного інструменту, що наведені нижче.

Вибір гібриду Agile та Waterfall обумовлено необхідністю застосування комбінації гнучкості Agile та структурної деталізації Waterfall. Це пояснюється тим, що неформальна освіта, яка включає в себе розробку вебзастосунку для дистанційного навчання, може зазнавати змін та вимагати швидкої адаптації до нових вимог. Гнучка методологія Agile дозволяє оперативно враховувати зміни та взаємодіяти з користувачами під час розробки. З іншого боку, Waterfall забезпечує структурований підхід до планування та деталізації завдань, що важливо для забезпечення якості та стабільності продукту.

Python обрано через простоту, лаконічність та широку підтримку у спільноті розробників. Python дозволяє швидку розробку, а його багата екосистема

допомагає в легкому інтегруванні нових функцій та бібліотек.

Для забезпечення масштабованості, високої доступності та ефективного управління інфраструктурою обрано Helm та Kubernetes. Також дані інструменти забезпечать оркестрацію контейнерів та управління конфігурацією. Helm дозволяє легко керувати конфігурацією та версіонуванням застосунку в середовищі Kubernetes.

Вибір SQLite як системи управління базами даних обумовлений легкістю та компактністю, що підходить для проєкту середнього розміру. SQLite не вимагає окремого сервера та легко вбудовується в застосунок, забезпечуючи простоту у використанні та підтримці.

Обрана методологія та технології спрямовані на забезпечення ефективного впровадження нового функціоналу вебзастосунку для неформальної освіти, враховуючи специфіку проєкту та його потреби.

РОЗДІЛ 2

ПОСТАНОВКА ЗАДАЧІ ТА ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Постановка задачі

У світі, де технології швидко змінюють наше сприйняття навчання та освіти, важливо використовувати інноваційні рішення. З таким усвідомленням важливо поставити перед собою завдання розробити вебзастосунок, який відповідатиме сучасним вимогам, забезпечуючи гнучкість, інтерактивність та персоналізацію навчального процесу.

Високий рівень мотивації та результативності навчання є ключовими аспектами в освітньому процесі. Саме тому розробка вебзастосунку має на меті підвищити ці показники, створюючи стимули для учнів та надаючи їм інструменти для ефективного вивчення матеріалу.

Одним із головних пріоритетів є зробити неформальну освіту більш доступною та зручною для учнів з різними потребами. Це означає створення середовища, де кожен учень може вільно розвиватися, отримуючи необхідні знання та навички, не залежно від своїх особистих обмежень чи умов.

Розроблений вебзастосунок може мати широкий спектр застосувань. Він має стати основою для організації дистанційних курсів сприяючи ефективній комунікації між вчителями та учнями, а також надаючи доступ до навчального матеріалу у будь-який час та в будь-якому місці. Крім того, цей вебзастосунок може бути використаний для вдосконалення існуючих платформ дистанційного навчання, допомагаючи їм пристосуватися до потреб сучасного користувача та вдосконалюючи якість навчального процесу.

Не менш важливою є розробка методичних рекомендацій щодо використання вебзастосунків в неформальній освіті. Ці рекомендації допоможуть освітнім установам та вчителям ефективно впроваджувати нові технології в навчальний процес та максимально використовувати їх потенціал.

Нижче наведено загальну постановку задачі або технічну пропозицію розробки вебзастосунку організації дистанційних курсів неформальної освіти.

Огляд проєкту:

- розробка вебзастосунку, який надає можливість організації дистанційних курсів неформальної освіти для онлайн IT-академії GoITeens;
- забезпечення доступу до інструментів, ресурсів та інформації, які допоможуть учням онлайн IT-академії GoITeens отримувати знання швидше та зроблять їх більш доступними, а викладачам допоможуть проводити більш об'єктивне оцінювання.

Функціональні вимоги:

- розробити вебзастосунок, який буде відповідати потребам онлайн-школи “GoITeens”;
- забезпечити гнучкість, інтерактивність та персоналізацію навчального процесу;
- підвищити мотивацію та результативність навчання;
- зробити неформальну освіту більш доступною та зручною для учнів з різними потребами.

Технічні вимоги:

- розробка серверної частини з використанням вебфреймворку, Flask, для обробки запитів користувачів та зберігання даних;
- використання мов програмування, таких як Javascript, для розробки фронтенду та бізнес-логіки;
- використання ORM для роботи з базою даних засобами мови програмування Python.

Вимоги до управління проєктом:

- розробка плану проєкту з чіткими критеріями успіху, графіком робіт та ресурсними вимогами;
- управління командою розробників, тестувальників та інших зацікавлених сторін для забезпечення виконання проєкту вчасно та відповідно до вимог.

Тестування на якість:

- розробка тестових сценаріїв та проведення системного, модульного та інтеграційного тестування для перевірки функціональності та якості вебзастосунку;

- забезпечення стабільності, швидкодії та ефективності системи шляхом оптимізації коду, ресурсів та інших аспектів розробки.

Документація:

- розробка технічної документації, включаючи специфікації системи, діаграми бази даних, опис API та інструкції з розгортання та використання вебзастосунку;

- підтримка документації проєкту в актуальному стані та забезпечення доступу до неї для розробників та інших зацікавлених сторін.

Очікується, що розробка вебзастосунку для неформальної освіти буде мати значний вплив на розвиток онлайн-школи “GoITeens” та зробить неформальну освіту більш доступною та зручною для учнів з різними потребами.

Серед функціональних вимог слід зазначити можливість створення облікового запису, що дозволяє користувачам реєструватися та отримувати доступ до вебзастосунку; створення та редагування курсів, де викладачі можуть формувати та змінювати курси з неформальної освіти, вказуючи назву, опис, теми та інші важливі атрибути; створення та проходження навчальних тестів, що дає викладачам змогу створювати тести для своїх курсів, а студентам – проходити ці тести для оцінки свого рівня знань; обмін матеріалами, де користувачі можуть завантажувати та завантажувати матеріали для навчання, такі як лекції, презентації, відео та інше.

Нефункціональні вимоги включають забезпечення безпеки даних через конфіденційність та цілісність користувацьких даних за допомогою шифрування та інших заходів; масштабованість, що передбачає можливість системи збільшувати обсяги даних та кількість користувачів; високу доступність, яка гарантує найвищий рівень доступності вебзастосунку для користувачів без перебоїв у роботі; швидкодію, що забезпечує швидкість завантаження сторінок та відгуків на запити користувачів; сумісність та розширюваність, які гарантують сумісність

вебзастосунку з різними браузером та можливість легкої розширення функціоналу в майбутньому.

Визначення обсягу та обмежень проєкту є важливою частиною планування, що допомагає чітко визначити, що включено в проєкт, а також визначити рамки його виконання.

Обсяг проєкту включає в себе розробку та підтримку системи обліку облікових записів користувачів, створення, редагування та видалення курсів для викладачів, можливість створювати та проходити навчальні тести для студентів та викладачів, а також обмін матеріалами та ресурсами для навчання між користувачами. До адміністративних можливостей входить система управління користувачами та правами доступу, моніторинг активності користувачів та викладачів, а також можливість вирішення конфліктів та питань, пов'язаних із викладанням. Також передбачено автоматизовану систему рекомендацій для користувачів на основі їхньої історії та вибору курсів.

2.2 Вибір засобів реалізації

При розробці вебзастосунку для організації дистанційних курсів неформальної освіти було проведено всебічний аналіз можливих методів та засобів реалізації. Одним із ключових аспектів вибору засобів реалізації стало забезпечення зручності, багатофункціональності та адаптивності вебзастосунку до потреб різних категорій користувачів.

Для серверної частини було обрано мову програмування Python завдяки її простоті, гнучкості та широкому набору бібліотек. В якості системи управління базами даних (СУБД) було використано SQLite, що забезпечує простоту інтеграції та достатню продуктивність для цілей проєкту. Для взаємодії з базою даних застосовувалася ORM SQLAlchemy, яка дозволяє ефективно працювати з даними, абстрагуючи складні SQL-запити.

При проєктуванні інтерфейсу користувача основна увага приділялася зручності та інтуїтивній зрозумілості. Було вирішено використовувати сучасні

фронтенд-технології, такі як HTML, CSS і JavaScript, що забезпечують адаптивність і можливість роботи в різних браузерах.

Вибір засобів реалізації також враховував необхідність забезпечення безпеки та масштабованості вебзастосунку. Використання стандартних протоколів безпеки, таких як HTTPS, а також інтеграція з зовнішніми сервісами для автентифікації користувачів, дозволяють забезпечити надійний захист даних.

Проєкт повинен бути завершений в рамках обумовленого графіку та бюджету. Доступні ресурси для розробки обмежені, тому розробка повинна оптимально використовувати їх. Також є обмежена можливість масштабування та розширення системи, тому проєкт повинен бути готовий до зростання користувацької бази. Велика увага має бути приділена безпеці даних користувачів та системи загалом. Забезпечення сумісності вебзастосунку з різними браузерами та пристроями також є обмеженням. Використання Python, Helm, Kubernetes та SQLite обмежується технічними можливостями та інтеграційними обмеженнями цих технологій.

РОЗДІЛ 3

ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ОРГАНІЗАЦІЇ ДИСТАНЦІЙНИХ КУРСІВ НЕФОРМАЛЬНОЇ ОСВІТИ

3.1 Розробка функціональних діаграм вебзастосунку для організації дистанційних курсів неформальної освіти

Архітектурний дизайн вебзастосунку визначає його структуру та організацію компонентів для забезпечення ефективної роботи, масштабованості та зручного відтворення. У даному проєкті, буде використано архітектурний підхід MVC (Model-View-Controller) (див. рисунок 3.1) для розділення логіки програми, представлення та управління даними. Даний архітектурний підхід використовує наступні сутності:

- модель, що представляє собою базу даних, де зберігаються дані про користувачі, курси, тести та інші необхідні відомості. Використання бази даних SQLite для легкості вбудовування та забезпечення ефективного зберігання та отримання даних.

- представлення, що включає в себе інтерфейс користувача, який дозволяє викладачам та студентам взаємодіяти з вебзастосунком. Буде використано технології HTML/CSS/JavaScript для створення дружнього та інтуїтивного інтерфейсу.

- контролер, що відповідає за обробку запитів від користувачів та взаємодію з моделлю та представленням.

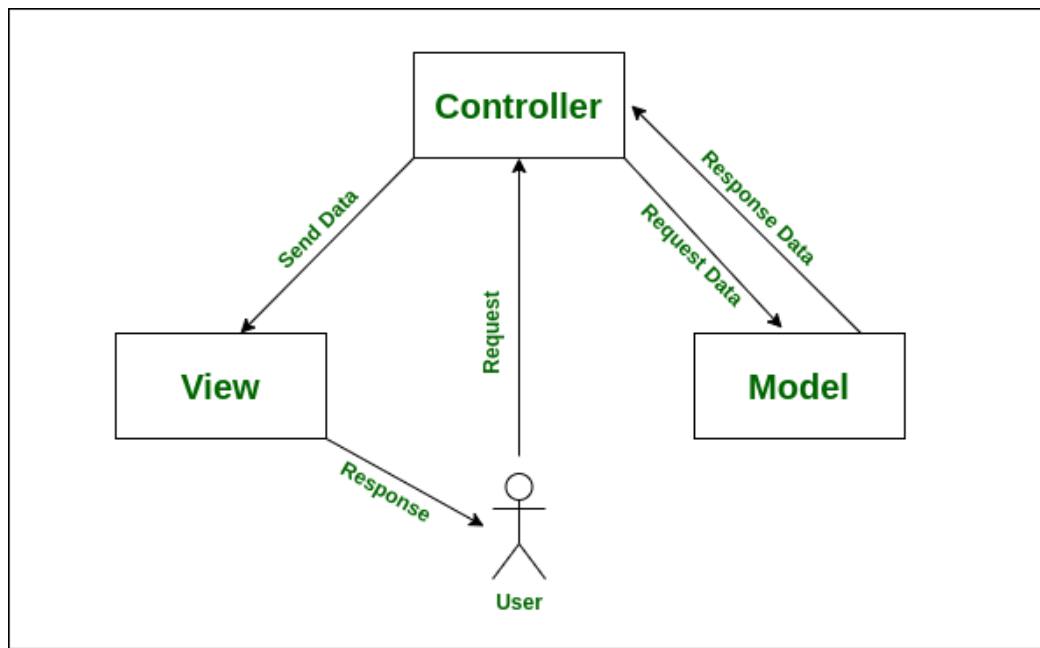


Рисунок 3.1 – Діаграма підходу MVC

Використання мови програмування Python та фреймворка Flask [8] для створення контролеру, який обробляє HTTP-запити та керує логікою вебзастосунку.

Користувачі взаємодіють через веб-інтерфейс, який передає запити до контролера. Контролер обробляє запити, взаємодіє з моделлю для отримання або збереження даних та оновлює представлення для відображення результатів користувачам.

Основні переваги архітектурного дизайну MVC:

- розділення відповідальностей, так як MVC дозволяє чітко розділити логіку додатку, представлення та управління даними, що спрощує розробку та підтримку коду;
- легкість розширення, позаяк застосування MVC полегшує внесення змін та розширення функціоналу без впливу на інші компоненти;
- підтримка тестування, оскільки кожен компонент може бути тестований окремо, це полегшує процес тестування та забезпечує високу якість коду.

Архітектурний дизайн вебзастосунку базується на ретельному аналізі вимог та враховує потреби як викладачів, так і студентів для створення ефективного та зручного інструменту для дистанційної неформальної освіти.

3.2 Проєктування бази даних вебзастосунку для організації дистанційних курсів неформальної освіти

При виборі технологій для створення бази даних були враховані ключові аспекти. SQLite обрано через його простоту та легкість використання, особливо в розробці та тестуванні. SQLite – це легка файлова СУБД, яка не вимагає окремого серверу та забезпечує простий спосіб зберігання та доступу до даних.

ORM (Object-Relational Mapping) представлено SQLAlchemy. Це парадигма програмування, яка дозволяє взаємодіяти з базою даних за допомогою об'єктів програми, а не SQL-запитів. SQLAlchemy полегшує роботу з базою даних, перетворюючи дані між реляційною базою даних і об'єктами Python. Це дозволяє працювати з даними в більш об'єктно-орієнтованому стилі та спрощує управління базою даних.

Охарактеризуємо технології, які будуть використані при розробці бази даних вебзастосунку.

SQLite – це вбудована СУБД у вигляді бібліотеки, яка надає легку та ефективну можливість зберігання та отримання даних. Обрано через його простоту та легкість використання, а також можливість використання вбудованих баз даних без необхідності встановлення окремого сервера.

ORM – це парадигма програмування, що забезпечує автоматичне відображення між об'єктами програми та таблицями бази даних. SQLAlchemy використовується як ORM для полегшення взаємодії з базою даних на рівні об'єктів, що робить код більш читабельним та підтримує зручніше управління даними.

SQLAlchemy – це бібліотека для роботи з базами даних у Python, яка надає високорівневий API для взаємодії з реляційними базами даних. Обрано як потужний інструмент для використання ORM, що спрощує роботу з базою даних та дозволяє використовувати об'єктно-орієнтований підхід при роботі з даними[9].

Розробка бази даних була орієнтована на вимоги користувачів, які включають облік користувачів у ролі студентів та викладачів, а також

відображення їхніх екзаменів та тестів. Запити та потреби користувачів стали ключовими факторами при визначенні структури бази даних та виборі відповідних зв'язків між таблицями.

Особлива увага приділялася визначенню зв'язків між об'єктами в базі даних. Ключовим був зв'язок між таблицею “exams” та “questions”. Цей зв'язок визначає, що один екзамен може мати багато тестових питань, що відображає реальність управління екзаменами та їхніми складовими елементами.

Такий підхід спрямований на створення бази даних, яка відповідає потребам користувачів та дозволяє ефективно взаємодіяти з нею на рівні об'єктів програми.

Відповідно, виходячи з поставленої задачі та конкретизованого алгоритму розробки, можемо створити перший файл, в котрому, використовуючи засоби SQLAlchemy буде створено базу даних, її таблиці та зв'язки між ними. Відповідно, в подальшому процесі розробки даного вебзастосунку, будемо звертатись не безпосередньо до бази даних мовою SQL, а до класів та об'єктів класів, що будуть описані у файлі models.py (див. рисунок 3.2).

```
1 import os
2 from flask_login import UserMixin
3 from sqlalchemy import create_engine
4 from sqlalchemy.orm import sessionmaker, relationship
5 from sqlalchemy.ext.declarative import declarative_base
6 from sqlalchemy import Column, Integer, String, ForeignKey
7
8 print(os.path.abspath(os.getcwd()))
9 database = create_engine(f'sqlite:///({os.path.abspath(os.getcwd())}/app.db', connect_args={'check_same_thread': False})
10 Session = sessionmaker(bind=database)
11 session = Session()
12 Base = declarative_base()
13
14
15 class User(UserMixin, Base):
16     __tablename__ = 'users'
17
18     id = Column(Integer, primary_key=True)
19     name = Column(String)
20     username = Column(String(100), unique=True)
21     status = Column(String(10)) # teacher or student
22     password = Column(String(100))
23
24
25 class Exam(Base):
26     __tablename__ = 'exams'
27
28     id = Column(Integer, primary_key=True)
29     name = Column(String)
30     questions = relationship('Question', back_populates='exam')
31
32 class Question(Base):
33     __tablename__ = 'questions'
34
35     id = Column(Integer, primary_key=True)
36     exam_id = Column(Integer, ForeignKey('exams.id'))
37     question = Column(String(256))
38     answer1 = Column(String(256))
39     answer2 = Column(String(256))
40     answer3 = Column(String(256))
41     correct = Column(String(256))
42     exam = relationship('Exam', back_populates='questions')
43
44 if __name__ == "__main__":
45     Base.metadata.create_all(database)
```

Рисунок 3.2 – Файл models.py

Цей код використовує Flask, SQLAlchemy та SQLite для створення бази даних та моделей об'єктів для користувачів, екзаменів і питань. Розглянемо його детальніше:

До імпортованих бібліотек відносяться `os`, `usermixin`, `create_engine`, `sessionmaker`, `declarative_base`, `Column`, `Integer`, `String`, `ForeignKey`.

Кожна з цих бібліотек або модулів несуть відповідний функціонал, відповідно, важливо розкрити цей функціонал для розуміння контексту розробки.

`Os` дозволяє взаємодіяти з операційною системою, отримує шлях до поточної робочої директорії для створення файлу з базою даних у поточній директорії щоб забезпечити безпечний та вільний доступ для роботи з базою даних засобами SQLAlchemy.

`UserMixin` забезпечує базовий функціонал для користувачів Flask-Login, що в подальшому забезпечить можливість реалізації функціоналу реєстрації та обліку користувачів.

`Create_engine`, `sessionmaker`, `declarative_base` –Інструменти SQLAlchemy для налаштування та взаємодії з базою даних.

`Column`, `Integer`, `String`, `ForeignKey` –це Визначення типів даних для полів(стовпців у базі даних) та відносин в об'єктах моделі(зовнішні ключі в базі даних).

При запуску програми `__name__ == "__main__"` забезпечує виконання даного відрізка коду виключно при прямому запуску поточного файлу, тобто при імпорті даного файлу код не буде відпрацьовувати), створюється структура бази даних, яка відображає визначені моделі.

Цей код встановлює зв'язки між користувачами, екзаменами та питаннями, дозволяючи зберігати та отримувати дані про них в базі даних SQLite з використанням SQLAlchemy. Запустивши даний файл, використовуючи інтерпритатор Python, буде створено файл `app.db`, котрий і містить в собі базу даних з необхідними таблицями та зв'язками.

3.3 Розробка прототипу інтерфейсу користувача

Дизайн інтерфейсу користувача (UI) є важливою частиною розробки вебзастосунку для неформальної освіти. Наведений нижче опис включає архітектурний дизайн, структуру сторінок та розгортання, а також деталі інтерфейсу для різних типів користувачів.

Вебзастосунок має бути зручним та зрозумілим для різних категорій користувачів: студентів та викладачів. Основні аспекти архітектурного дизайну включають наступні вимоги:

- зручне меню для переходу між різними розділами: тести, екзамени, профіль користувача;
- чіткі та зрозумілі посилання для швидкого доступу до основних функцій;
- структура сторінок враховує потреби користувачів та дозволяє зручно взаємодіяти з основними функціями додатку;
- огляд доступних екзаменів та тестів;
- можливість швидкого старту тесту для студентів або створення нового екзамену для викладачів;
- відображення питань та варіантів відповідей для тесту або екзамену;
- можливість вибору відповіді, подачі тесту чи перегляду результатів;
- перегляд та редагування особистої інформації;
- доступ до пройдених тестів чи створених екзаменів;
- деталі інтерфейсу повинні забезпечувати комфортну та ефективну взаємодію користувачів з додатком;
- чіткі та лаконічні формулювання питань та варіантів відповідей;
- графічне відображення результатів для легкого розуміння прогресу;
- детальна статистика для викладачів щодо відповідей студентів.

Підсумовуючи, важливо зауважити що дизайн інтерфейсу користувача повинен надавати зручність та функціональність для ефективного використання веб-додатку для навчання та тестування. Наведені вище принципи допоможуть забезпечити позитивний користувацький досвід та оптимізувати взаємодію з

платформою для всіх типів користувачів.

Оскільки даний застосунок реалізовується засобами фреймворку Flask, котрий за замовчуванням підтримує server-side rendering [10](тобто генерацію HTML-документа на стороні сервера, з подальшою його передачею на клієнт у вигляді статичного файлу), використаємо його, для повної інтеграції front- та back-end частин застосунку. Відповідно, стандартно у Flask шаблони(templates), тобто HTML файли, розміщуються у папці templates у корені директорії. Таку поведінку програми можливо змінити, проте в даному випадку в цьому немає необхідності.

Відповідно, для стилізації додатку буде використано мову CSS. За аналогією з папкою templates для HTML файлів, у Flask має бути папка static, в якій розміщено файли статичних ресурсів, такі як CSS, зображення, або JavaScript. Flask автоматично розпізнає цю папку та дозволяє використовувати її ресурси у шаблонах (див. рисунок 3.3).

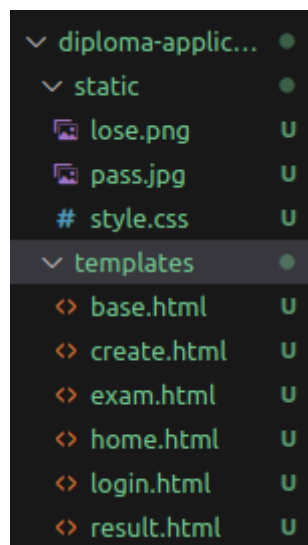


Рисунок 3.3 – Структура проєкту

Задля підключення CSS до HTML у кожному HTML-файлі необхідно додати посилання на відповідний CSS-файл, щоб використовувати його стилі.

Зараз, коли завершено етап проєктування, важливо відзначити, що основною метою розробки програмного забезпечення є забезпечення ефективної комунікації між користувачами та базою даних. Це досягається через дві ключові точки:

користувачий інтерфейс та базу даних.

Перший аспект – це користувацька частина. Ми створили декілька сторінок, таких як головна, сторінка екзамену, сторінка логізації та сторінка, на якій можна створити новий екзамен які розроблені так, щоб забезпечити зручність взаємодії. Використовуючи HTML та CSS, ми визначили вигляд та стиль цих сторінок. Jinja, як шаблонізатор, дозволяє динамічно вставляти дані у сторінки, забезпечуючи їх зв'язок з базою даних.

Другий аспект – це база даних. Було спроектовано модель з використанням SQLAlchemy та визначено зв'язки між таблицями, щоб відобразити структуру даних. SQLite була обрана як проста та легка база даних для розробки та тестування. Створений скрипт ініціалізації бази даних автоматично створює таблиці при запуску застосунку.

Усі ці елементи об'єднані в Flask, що дозволить реалізувати вебзастосунок та визначити маршрути для обробки HTTP-запитів. Важливо врахувати безпеку, таку як хешування паролів та використання Flask-Login для автентифікації. Розподілений код на різні файли сприяє чистоті та зручності управління проектом. Наразі, оскільки відповідні дії з проектуванням завершено, можемо заключити що наступним етапом розробки даного застосунку для контролю якості знань у навчальних закладах неформальної освіти є безпосередньо розробка основної логіки додатку використовуючи фреймворк Flask.

3.4 Фізична реалізація бази даних та програмна реалізація інтерфейсу користувача

Спочатку будуть розроблені методи авторизації, котрі дозволяють користувачам входити в систему та забезпечують безпеку їхніх облікових записів. Взаємодія із списками екзаменів та питань включатиме створення, редагування та видалення об'єктів.

Обробка винятків та помилок врахує можливі сценарії, що допомагає уникнути непередбачених ситуацій. Методи для тестування функціональності

включають в себе валідацію(перевірку на правильність, тобто відповідність певним вимогам) введених даних, що забезпечує коректну роботу системи навіть при різних сценаріях введення.

Загалом, цей розділ описує способи, якими користувацькі дії взаємодіють із базою даних через відповідні контролери, забезпечуючи функціональність та надійність вебзастосунку.

Даний процес має сенс розпочати з написання контролерів, що будуть відповідати за додавання нових об'єктів до бази даних (див. рисунок 3.4).

```
from flask import flash
from models import session, Question, Exam, User

def create_user(name, username, status, password):
    try:
        new_user = User(name=name, username=username, status=status, password=password)
        session.add(new_user)
        session.commit()
        flash('User created successfully', 'success')
    except Exception as e:
        session.rollback()
        flash(f'Error creating user: {str(e)}', 'error')

def create_exam(name):
    try:
        new_exam = Exam(name=name)
        session.add(new_exam)
        session.commit()
        flash('Exam created successfully', 'success')
    except Exception as e:
        session.rollback()
        flash(f'Error creating exam: {str(e)}', 'error')

def create_question(exam_id, question, answer1, answer2, answer3, correct):
    try:
        new_question = Question(exam_id=exam_id, question=question, answer1=answer1, answer2=answer2, answer3=answer3, correct=correct)
        session.add(new_question)
        session.commit()
        flash('Question created successfully', 'success')
    except Exception as e:
        session.rollback()
        flash(f'Error creating question: {str(e)}', 'error')
```

Рисунок 3.4 – Файл controllers.py

Кожна функція буде відповідати за додавання нового об'єкта відповідного класу у базу даних, забезпечуючи необхідну логіку та обробку виключень.

Ці функції використовують конструкцію try-ехсепт для обробки можливих винятків під час додавання нового об'єкта до бази даних. У випадку успішного створення об'єкта, функція викликає session.commit() для збереження змін. У випадку помилки, викликається session.rollback() для відміни будь-яких змін та виведення повідомлення про помилку за допомогою flash.

Відповідно, на момент, коли буде необхідно реалізувати функції користувача(views), вони викликатимуть дані функції, котрі описані вище,

оскільки в них ізольована логіка роботи з базою даних (див. рисунок 3.5).

```
def get_all_exam_names():
    try:
        exam_names = [exam.name for exam in session.query(Exam).all()]
        return exam_names
    except Exception as e:
        flash(f'Error retrieving exam names: {str(e)}', 'error')
        return []

def get_questions_for_exam(exam_id):
    try:
        questions = session.query(Question).filter_by(exam_id=exam_id).all()
        return questions
    except Exception as e:
        flash(f'Error retrieving questions for exam: {str(e)}', 'error')
        return []

def get_user_by_id(user_id):
    try:
        user = session.query(User).get(user_id)
        return user
    except Exception as e:
        flash(f'Error retrieving user: {str(e)}', 'error')
        return None
```

Рисунок 3.5 – Функції, що ізолюють в собі необхідний функціонал

Важливим для розуміння контексту розробки даного додатку обґрунтувати необхідність даних функцій та розібрати процес їхньої роботи.

Отримання всіх назв екзаменів (`get_all_exam_names`). Дана функція важлива для відображення доступних екзаменів користувачам та надає їм можливість вибору конкретного екзамену для проходження. Використовуючи SQLAlchemy, функція отримує всі об'єкти класу `Exam` та сортуємо тільки назви, тобто фактично отримуємо назви всіх екзаменів. Це дозволяє динамічно побудовувати список екзаменів, що доступних для вибору.

Отримання всіх питань для конкретного екзамена за його `id` (`get_questions_for_exam`). Даний фрагмент коду дозволяє отримати список питань для певного екзамену та відобразити їх користувачеві під час проходження тесту. Запитуючи об'єкти класу `Question`, які мають вказаний `exam_id`, функція отримує

пов'язані питання для конкретного екзамену. Це важливо для підготовки та відображення питань під час тестування.

Отримання користувача за його `id` (`get_user_by_id`). Дана функція використовується для отримання об'єкта користувача за його ідентифікатором, що може бути корисним для виведення інформації про користувача та редагування профілю. Використовуючи ідентифікатор користувача, ми запитуємо об'єкт класу `User` за допомогою `SQLAlchemy`, щоб отримати повну інформацію про користувача. Це важливо для персоналізації та взаємодії з профілем користувача.

Наведені вище функції роблять роботу з базою даних більш абстрактною та ефективною, забезпечуючи зручний доступ до необхідних даних для взаємодії з користувачем.

Підсумовуючи, важливо зауважити що у цьому розділі було розглянуто ключові функції, які відповідають за взаємодію з базою даних у вебзастосунку. Контролери є важливим елементом проєкту, оскільки вони визначають логіку додавання, отримання та оновлення даних у базі. Наведені функції спрощують роботу з базою даних та забезпечують стабільну та ефективну взаємодію із збереженням даних.

Створені контролери для додавання користувачів, екзаменів та питань в базу даних дозволяють системі швидко реагувати на потреби користувачів та адміністраторів. Функції для отримання списків екзаменів та питань розширюють можливості відображення важливої інформації для подальшого використання в застосунку.

У роботі з контролерами враховані норми безпеки та можливі виключення для забезпечення стійкості системи. Вони є ключовим елементом для забезпечення коректності та ефективності взаємодії із збереженням даних у вебзастосунку, що розробляється.

Враховуючи масштаби проєкту, важливим вважаємо зауважити що було прийняте рішення не наводити зображення всіх контролерів, проте, вони можуть зустрічатись у наступних розділах.

Наступний етап розробки відіграє ключову роль у створенні інтерфейсу

користувача та забезпеченні взаємодії між користувачем і системою. У веб-розробці термін “views” вказує на частину додатку, яка відповідає за відображення інформації та обробку запитів користувачів.

Views є ключовим елементом в архітектурі веб-додатків. Вони визначають, як дані повинні бути представлені користувачеві і взаємодіяти з логікою додатку. Views обробляють HTTP-запити, визначаючи, які дані потрібно вивести та які дії повинні бути виконані при взаємодії з користувачем.

У веб-додатку, що розробляється у рамках даної роботи views грають важливу роль у наданні користувачам можливості взаємодії з екзаменами та тестуванням. Вони відповідають за відображення списків екзаменів, питань, а також за обробку відповідей користувачів під час тестування.

Реалізація користувацьких методів views є важливим етапом, оскільки вона визначає, як користувачі взаємодіють з системою. Ефективні та інтуїтивно зрозумілі views допомагають створити позитивний досвід використання додатку та забезпечити його легку навігацію та використання.

Початковий етап створення функцій користувача передбачає налаштування додатку на базі Flask (див. рисунок 3.6). Для цього використовується об'єкт Flask, який дозволяє легко розробляти вебзастосунки на мові програмування Python. Під час конфігурації Flask, було визначено кілька ключових аспектів:

SECRET_KEY – Це унікальний ключ, який використовується для підпису cookies і інших елементів безпеки. Важливо використовувати стійкий ключ для забезпечення безпеки сесій та інших областей, де важлива конфіденційність даних.

Login Manager – Для роботи з системою автентифікації користувачів використовується Flask-Login. В даному проєкті буде використано LoginManager для управління процесом автентифікації та визначив сторінку, на яку буде перенаправлено користувача у випадку невдалої автентифікації.

Наведені вище конфігураційні елементи сприяють створенню безпечного та ефективного вебзастосунку з можливістю аутентифікації користувачів та збереження їхнього стану між запитами. Правильна конфігурація є важливим етапом для забезпечення стабільної та безпечної роботи застосунку.

```

1  from flask import Flask
2  from flask_login import LoginManager
3
4
5  app = Flask(__name__)
6
7  login_manager = LoginManager()
8  login_manager.init_app(app)
9  login_manager.login_view = "login"
10
11  💡
12  import views
13
14  @login_manager.user_loader
15  def load_user(user_id):
16      from controllers import get_user_by_id
17      return get_user_by_id(user_id)
18
19  app.secret_key = "sldgw;aru8tuj20349haiufQ7F"

```

Рисунок 3.6 – Конфігурація додатку на Flask(файл app.py)

Наразі, на базі сконфігурованого застосунку, можемо визначати paths(шляхи, за якими додаток буде приймати HTTP-запити) (див. рисунок 3.7). Кожному з таких шляхів буде відповідати одна view-function(користувацька функція представлення), котра має реагувати на дані, що прийшли від користувача, певним чином їх обробити та повернути відповідний HTTP-response, тобто відповісти на запит, що був отриманий від користувача. У випадку такого додатку, який наразі розробляється, більшість таких функцій будуть повертати користувачу(тобто на клієнтську частину додатку) сформований HTML файл.

```

10 @app.route('/')
11 def home():
12     return render_template('home.html', exams=get_all_exam_names())
13
14
15 @app.route('/exam/<int:exam_id>')
16 def exam(exam_id):
17     exam_questions = get_questions_for_exam(exam_id)
18     if exam_questions:
19         [test.insert(randint(1, 3), test.pop(3)) for test in exam_questions]
20         return render_template('exam.html', exam_questions=exam_questions)
21     return redirect(url_for('home'))
22
23
24 @app.route('/login', methods=['GET', 'POST'])
25 def login():
26     if request.method == 'POST':
27         username = request.form['username']
28         password = request.form['password']
29         user = session.query(User).filter_by(username=username).first()
30         if user and check_password_hash(user.password, password):
31             login_user(user)
32             return redirect(url_for('home'))
33     return render_template('login.html')
34
35
36 @app.route('/logout')
37 @login_required
38 def logout():
39     logout_user()
40     return redirect(url_for('home'))
41
42
43 @app.route('/result')
44 @login_required
45 def result():
46     is_passed, correct_answers = count_correct_answers(request.form)
47     return render_template("see_result.html", correct_answers=correct_answers, passed=is_passed)
48
49
50 @app.route('/create_exam', methods=['GET', 'POST'])
51 @login_required
52 def create_exam():
53     if request.method == 'POST':
54         create_exam(request.form['name'])
55         create_question(exam_id=request.form['exam_id'], question=request.form['question'], answer1=request.form['answer1'],
56                        answer2=request.form['answer2'], answer3=request.form['answer3'], correct=request.form['correct'])
57     return redirect(url_for('home'))

```

Рисунок 3.7 – Файл views.py, в котрому прописана логіка користувача

Є необхідним детально розібрати дані функції, так як на них будується весь функціонал додатку. Дані функції, що агрегують в собі і HTTP-requests (тобто запити від користувача) і логіку backend, тобто логіку взаємодії з базою даних. Також дані функції визначають собою маршрутизацію сервера, що буде запущений за допомогою Python.

Головна сторінка, яка доступна за маршрутом `/` відповідає за відображення головної сторінки додатку. Функція `get_all_exam_names()` викликається для отримання списку всіх екзаменів, які потім передаються у шаблон `home.html`. Доступна за маршрутом `/`.

Сторінка екзамену відображає сторінку конкретного екзамену. Функція `get_questions_for_exam(exam_id)` викликається для отримання питань для конкретного екзамену, після чого порядок відповідей перемішується перед

відображенням. Доступна за маршрутом `/exam/<int:exam_id>`.

Сторінка входу (логіну) відповідає за відображення сторінки входу. При введенні користувачем логіну та паролю у відповідну форму та їхньому поданні, виконується перевірка за допомогою `check_password_hash`, і якщо дані вірні, користувач автентифікується і перенаправляється на головну сторінку. Доступна за маршрутом : `/login`.

Вихід (логаут). При виклику цього маршруту користувач виходить із системи, використовуючи `logout_user()`, та перенаправляється на головну сторінку. Доступно за адресою `/logout`

Сторінка результатів відображає сторінку результатів після проходження тесту. Використовує функцію `count_correct_answers()` для підрахунку правильних відповідей. Знаходиться за адресою `/result`.

Сторінка створення екзамену відображає сторінку для створення нового екзамену та питань для нього. Обробляє POST-запит, додає новий екзамен та питання до бази даних. Доступна за адресою `/create_exam`.

Кожен маршрут взаємодіє з відповідними функціями контролерів, які виконують логіку додатку. Всі шаблони відображення використовуються для представлення даних користувачу, а логіка зберігання та обробки даних реалізована в контролерах та моделях бази даних.

Важливо зауважити що в даному розділі також висвітлені тільки основні функції користувацької взаємодії, тобто такі, на яких будується логіка всього додатку. По завершенню розробки даного додатку, з його повним кодом можна буде ознайомитись за прикріпленим у висновку посиланням.

Наступна частина розробки буде орієнтована на висвітленні процесу реалізації елементів, які безпосередньо взаємодіють із користувачем. Цей аспект є важливим, оскільки саме через інтерфейс користувач взаємодіє з усім функціоналом додатку. Відповідно, можна заключити що саме ця частина додатку є найважливішою для користувача у будь-якому додатку, адже якщо користувач, наприклад, не знайде потрібну кнопку – він не захоче повертатись на сайт. Структуру сторінок визначено наступною:

- головна сторінка (home.html). На цій сторінці користувач отримує перелік усіх доступних екзаменів. Це дозволяє зручно обирати бажаний тест для проходження;
- сторінка екзамену (exam.html). Після вибору конкретного екзамену користувач переходить на сторінку з питаннями для тестування. Питання відображаються у випадковому порядку, щоб зробити тест більш варіативним;
- сторінка логіну (login.html). Для того щоб мати можливість проходження тестів та відстеження результатів, користувач повинен увійти в систему. На цій сторінці відбувається процес аутентифікації;
- сторінка результатів (result.html). Після завершення тестування користувач переходить на сторінку результатів, де відображається інформація про кількість правильних відповідей та результат тесту;
- сторінка створення нового екзамену (create.html). Для викладачів чи адміністраторів передбачена можливість створення нових екзаменів та додавання до них питань для подальшого використання в системі.

Створення зручного та ефективного користувацького інтерфейсу є важливою частиною процесу розробки. Обрані функції враховують потреби різних користувачів, забезпечуючи їм зручність при взаємодії з додатком. Головна сторінка пропонує перегляд доступних екзаменів, а система логіну дозволяє користувачам взаємодіяти із персональними тестами та відстежувати свій прогрес.

Отже, у цьому розділі буде висвітлено деталі реалізації кожної з вищеписаних функцій, використовуючи мову програмування Python та фреймворк Flask. Також буде розглянуто використання шаблонізатора Jinja для побудови динамічних та інтерактивних сторінок.

Шаблонізатор Jinja — це інструмент для генерації текстового вмісту, який широко використовується в Flask. Він дозволяє вбудовувати код на Python в HTML-файли для створення динамічного контенту. Такий підхід дозволяє використовувати змінні, умовні конструкції та цикли прямо в HTML, що полегшує генерацію та модифікацію вмісту в залежності від обставин.

Для того, щоб уніфікувати всі сторінки додатку, зробити їх легко

змінюваними, впізнаваними та стабільними, необхідно скористатись функцією наслідування шаблонів, принципово схожим на наслідування класів у ООП.

Файл `base.html` (див. рисунок 3.8) можна розглядати як заготовку (шаблон) для усіх інших сторінок в додатку. У цьому файлі можна визначити загальні елементи сторінки, такі як хедер, футер, стилі, скрипти тощо. Його сенс полягає в тому, щоб визначити базову структуру, яка буде використовуватися на всіх сторінках додатку.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Екзамен</title>
6      <link rel="stylesheet" href="../static/styles.css">
7  </head>
8  <body>
9      <ul class="nav">
10         <li><a href="/">Головна сторінка</a></li>
11         <li><a href="/create_exam">Додати новий екзамен</a></li>
12         {% if current_user.is_authenticated %}
13         |     <li><a href="/logout">Вийти</a></li>
14         |     {% else %}
15         |     <li><a href="/login">Увійти</a></li>
16         |     {% endif %}
17         </ul>
18         {% block content %}
19         {% endblock %}
20     </body>
21 </html>
22
23
```

Рисунок 3.8 – Базовий шаблон `base.html`

У Flask і Jinja наслідування шаблонів реалізується через ключове слово `{% extends "base.html" %}` в початковому рядку нового шаблону. Коли шаблон успадковує інший, він отримує всі його блоки та можливість використовувати їх або перезаписувати за необхідності (див. рисунок 3.9).

```

1  {% extends "base.html" %}
2
3  {% block content %}
4      <form action = "{{ url_for('create_exam') }}" method = "POST">
5          <h3>Add new test</h3>
6
7          <select name="name">
8              {% for exam in exams %}
9                  <option value="{{ exam.id }}">{{ exam.name }}</option>
10             {% endfor %}
11         </select>
12
13         <label for="question">Question</label>
14         <input type = "text" id="question" name = "question" /><br>
15
16         <label for="ans1">First answer</label>
17         <input type = "text" id="ans1" name = "ans1" /><br>
18
19         <label for="ans2">Second answer</label>
20         <input type = "text" id="ans2" name = "ans2" /><br>
21
22         <label for="correct">Correct answer</label>
23         <input type = "text" id="correct" name = "correct" /><br>
24
25
26         <input type = "submit" value = "Submit" /><br>
27     </form>
28 {% endblock %}

```

Рисунок 3.9 – Сторінка для створення нових екзаменаційних питань

Як було згадано вище, даний шаблон буде основою для всіх сторінок у вебзастосунку. Відповідно, ті елементи, котрі реалізовані тут, будуть наслідуватись на всі сторінки, що будуть розширювати собою даний шаблон. Також на 12 рядку даного коду важливо помітити що завдяки шаблонізатору, є можливість відображати певні кнопки за певних умов. Коли користувач вже зайшов в свій аккаунт – він бачить кнопку, яка дозволяє йому вийти, а коли ще не зайшов, або вже вийшов – має можливість натиснути на кнопку “Увійти”, котра перенаправить його на сторінку входу та зайти там у свій аккаунт.

Ця сторінка, представлена у шаблоні `create_exam.html`, використовує концепцію наслідування шаблонів від `base.html`.

`{% extends "base.html" %}` – це вказівка, що цей шаблон успадковує структуру та стилі з `base.html`.

{% block content %}...{% endblock %} – це блок, який визначає частину вмісту сторінки, яка буде вставлена в блок content в base.html. Все, що знаходиться між цими тегамі, стане частиною блока content.

<form action="{ { url_for('create_exam') } }" method="POST"> – форма, яка відправляє дані на URL, звідки обробляється створення нового екзамену.

<select name="name">...</select> – випадаючий список для вибору існуючого екзамену, до якого додається нове питання.

Елементи <label> та <input> використовуються для введення питання та варіантів відповідей, а також для визначення правильної відповіді.

{% for exam in exams %}...{% endfor %} – цикл, який проходиться по всіх існуючих екзаменах та додає їхні імена до випадаючого списку.

<input type="submit" value="Submit" /> – кнопка для відправки форми з новим питанням.

Ця сторінка (див. рисунок 3.10) дозволяє користувачеві додавати нові питання до існуючих екзаменів. Вибравши екзамен зі списку, користувач вводить питання та варіанти відповідей, вказує правильну відповідь та відправляє дані форми. Це спрощує процес редагування та додавання тестового матеріалу до системи.

```
1  {% extends "base.html" %}
2
3  {% block content %}
4
5  <form action="{ { url_for('check-test') } }" method="post">
6      {% for single_test in all_data %}
7          <br><span>{ { single_test[0] } }</span><br>
8          {% for j in single_test[1:] %}
9              <input type="radio" name="{ { single_test[0] } }" id="{ { j } }" value="{ { j } }" >
10             <label for="{ { j } }" name="{ { j } }">{ { j } }</label><br>
11             {% endfor %}
12         {% endfor %}
13         <input type="submit" value="Submit">
14     </form>
15
16 {% endblock %}
```

Рисунок 3.10 – Шаблон, відповідальний за відображення сторінки для складання екзамену

Також цікавим та важливим для розуміння логіки роботи програми

елементом є сторінка, яка відповідає за проходження екзамену. В ній реалізовано значно більше можливостей шаблонізатора, ніж на даній сторінці, проте важливим є висвітлити обидва приклади, адже перший(сторінка для створення нових питань) послуговується гібридним шаблонуванням, тобто має доволі багато власного HTML-коду. Сторінка ж, яка буде розглянута нижче, має значно менше власного коду, завдяки можливості генерації нових тегів використовуючи цикли з мови програмування Python.

Наведений вище шаблон, представлений у файлі з розширенням .html, використовується для відображення тесту студентам. Розглянемо детальніше структуру та основні елементи цього шаблону.

`{% extends "base.html" %}` – цей тег вказує, що цей шаблон успадковує структуру та стилі з іншого шаблону, зазначеного як `"base.html"`.

`{% block content %}...{% endblock %}` – це блок, який визначає частину контенту, яка буде вставлена в блок `content` в розширеному шаблоні `"base.html"`. Весь контент між цими тегами буде вставлений у відповідне місце в базовому шаблоні та, після такої коомпіляції відправлений користувачу у якості звичайного HTML-файла.

`<form action="{% url_for('check-test') %}" method="post">` – форма для відправки результатів тестування на сервер. Дані форми будуть відправлені на URL, який вказаний в `url_for('check-test')`, метод відправки – POST.

`{% for single_test in all_data %}` – цикл, який проходиться по всіх питаннях тесту. `all_data` містить дані для всіх питань, включаючи текст питання та варіанти відповідей.

`<span>{{ single_test[0] }}</span>` – виведення тексту питання.

`{% for j in single_test[1:] %}...{% endfor %}` – внутрішній цикл, який проходиться по варіантах відповідей для кожного питання.

`<input type="radio" name="{% single_test[0] %}" id="{% j %}" value="{% j %}" >` – елемент для вибору відповіді з використанням радіокнопок. Кожна радіокнопка має унікальний ідентифікатор та значення, яке відповідає варіанту відповіді.

`<label for="{% j %}" name="{% j %}">{% j %}</label><br>` – мітка для кожної

радіокнопки, яка виводить текст варіанту відповіді.

`<input type="submit" value="Submit">` – кнопка для відправки результатів тестування.

Цей шаблон створений так, щоб зручно відображати та обробляти тестові завдання для користувачів, які проходять тест. Завдяки реалізації через цикл, даний шаблон зможе відобразити як екзамен на 2 питання, так і екзамен на сотні питань.

Перед тим як розглядати концепції докеризації та управління контейнерами за допомогою Kubernetes і Helm в контексті проєкту, що є на стадії розробки, важливим є розглянути кожен з цих технологій у відриві, щоб зрозуміти для чого вони існують в даній роботі, їхню важливість та необхідність вивчення таких технологій всіма.

Докеризація є стратегією розробки та розгортання програмного забезпечення. Вона використовує технологію контейнеризації для упакування додатків та їх залежностей в стандартизовані контейнери. Кожен контейнер включає усе необхідне для виконання програми, включаючи код, середовище виконання, бібліотеки та інші налаштування. Це створює ізольоване та середовище, яке можна дублювати, переносити та змінювати, що дозволяє запускати додатки безпечно та незалежно від конкретної операційної системи чи конфігурації [11].

Докер дозволяє розробникам уникнути проблем, пов'язаних зі сумісністю середовищ, і забезпечує консистентність між розробкою та виробництвом. Це полегшує розгортання, відлагодження та масштабування додатків.

Helm є інструментом для керування пакетами Kubernetes, який допомагає у встановленні, оновленні та керуванні додатками в середовищі Kubernetes. Helm використовує концепцію “charts” – це пакети, які містять у собі всі необхідні компоненти та конфігурації для розгортання додатку.

Charts визначаються файлами конфігурації, які описують різні аспекти додатку, такі як сервіси, ресурси, налаштування та залежності. Helm спрощує керування складними додатками, роблячи їх легко відтворюваними та масштабованими.

Kubernetes – це платформа для оркестрації контейнерів, яка надає

автоматизоване управління, масштабування та деплоймент додатків в контейнерах. Он вирішує проблеми управління додатками в середовищах, де запущено тисячі контейнерів [12].

Kubernetes дозволяє визначати декларативно, як має працювати додаток, і автоматизує процеси розгортання, масштабування та відновлення після збоїв. Він надає рівень абстракції для інфраструктури, що спрощує життя розробників та адміністраторів систем.

Інтеграція цих технологій дозволяє автоматизувати і уніфікувати розгортання та керування додатками, що забезпечує високу стабільність та продуктивність інфраструктури.

Почати розгляд даного розділу варто з упакування у Docker контейнер готового додатку на Flask. Для запуску такого контейнера потрібно мати встановлений Docker daemon(в багатьох десктопних дистрибутивах Linux він встановлений за замовчуванням, наприклад, в Ubuntu). Контейнеризація додатку відбувається завдяки спеціальному файлу у корні директорії – Dockerfile(дивись рисунок 3.11).

```
1 FROM python:3.8
2
3 COPY requirements.txt /app/requirements.txt
4 WORKDIR /app
5 RUN pip install -r requirements.txt
6
7 COPY . /app
8
9 ENV FLASK_APP=app.py
10 ENV FLASK_RUN_HOST=0.0.0.0
11
12 CMD ["flask", "run"]
```

Рисунок 3.11 – Dockerfile

Цей Docker-файл призначений для збірки та запуску Flask-додатку з SQLite базою даних у Docker-контейнері. Для розуміння контексту, є важливим

розглянути кожен рядок, адже рівень абстракції Докера не дозволяє розглядати блоки коду.

FROM Python:3.8 визначає базовий образ, з якого буде створений контейнер. У цьому випадку використовується офіційний образ Python версії 3.8.

COPY requirements.txt /app/requirements.txt копіює файл requirements.txt з локальної машини до робочої директорії /app в контейнері.

WORKDIR /app встановлює робочу директорію /app для подальших команд.

RUN pip install -r requirements.txt виконує команду для встановлення залежностей, оголошених у файлі requirements.txt(сам файл можна побачити в GitHub репозиторії.

COPY . /app копіює всі файли з поточної директорії на локальній машині до робочої директорії /app в контейнері.

ENV FLASK_APP=app.py встановлює змінну середовища, вказуючи файл, який використовується як основний скрипт Flask додатку.

ENV FLASK_RUN_HOST=0.0.0.0 встановлює змінну середовища, яка вказує Flask слухати всі доступні мережові інтерфейси.

CMD ["flask", "init-db"] ініціалізує базу даних перед запуском додатку. Це може бути ваша власна команда для створення або оновлення бази даних.

CMD ["flask", "run"] запускає Flask додаток.

Цей Docker-файл дозволяє збудувати контейнер, який містить Flask-додаток та його залежності, і запустити його у середовищі Docker. Відповідно, маючи контейнер, його можна запустити в будь-якому середовищі, де є процесор, RAM та материнська плата. Докер забезпечує стабільність роботи на будь-якому залізі та на будь-якій системі, адже він ізолює в своїх контейнерах процеси, подібно до того, як це роблять віртуальні машини.

Наступним кроком підготовки додатку до потенційної публікації, тобто викладення коду на віддаленому(або фізичному) сервері є написання Helm-chart. Це потрібно для того, щоб створити динамічний маніфест, написаний на yaml(yet another markup language), котрий буде визначати як саме Kubernetes має сприймати даний додаток всередині контейнера. Оскільки Kubernetes будує свою абстракцію

над контейнером – под, то необхідно визначити всі характеристики цього пода [13].

Для створення Helm-chart, в корньовій директорії проєкту необхідно виконати команду `helm create НАЗВА ЧАРТУ`. В якості назви автором був обраний `flask`. Це створює відповідну структуру (див. рисунок 3.12).

```
my-app/  
|-- charts/  
|   |-- my-app/  
|       |-- templates/  
|           |-- deployment.yaml  
|           |-- service.yaml  
|           |-- Chart.yaml  
|-- templates/  
|   |-- _helpers.tpl  
|   |-- NOTES.txt  
|   |-- deployment-configmap.yaml  
|   |-- deployment-secrets.yaml  
|   |-- service-ingress.yaml  
|   |-- service-loadbalancer.yaml  
|-- values.yaml
```

Рисунок 3.12 – Стандартна структура Helm chart

Важливим є розібрати структуру та файли, що її утворюють.

`charts/my-app/Chart.yaml`: -це основний файл, де вказуються метадані чарту.

`charts/my-app/templates/deployment.yaml` (див. рисунок 3.13) – це файл для розгортання (Deployment) додатку. Включає конфігурацію контейнера, ресурси та інші параметри.

```

1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: {{ include "my-app.fullname" . }}
5    labels:
6      {{- include "my-app.labels" . | nindent 4 }}
7  spec:
8    replicas: {{ .Values.replicaCount }}
9    selector:
10     matchLabels:
11       {{- include "my-app.selectorLabels" . | nindent 6 }}
12   template:
13     metadata:
14       labels:
15         {{- include "my-app.selectorLabels" . | nindent 8 }}
16     spec:
17       containers:
18         - name: {{ .Chart.Name }}
19           image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
20           ports:
21             - name: http
22               containerPort: {{ .Values.service.port }}
23           env:
24             - name: FLASK_APP
25               value: "{{ .Values.flaskApp }}"
26             # Add more environment variables as needed
27             {{- with .Values.resources }}
28               resources:
29                 {{ toYaml . | nindent 12 }}
30             {{- end }}

```

Рисунок 3.13 – Абстрактний об'єкт Kubernetes – Deployment

charts/my-app/templates/service.yaml – представляє собою файл для створення Kubernetes сервісу. Визначає, як служба буде доступна в межах кластера.

templates/deployment-configmap.yaml – configMap для конфігурації Flask-додатку.

templates/deployment-secrets.yaml – описує шаблон секретів для конфігурації чутливих даних, таких як пароль для бази даних або іншої конфіденційної інформації.

templates/service-ingress.yaml містить в собі шаблон ресурса, необхідного для публікації вебзастосунку.

templates/service-loadbalancer.yaml містить конфігурацію для використання LoadBalancer.

templates/_helpers.tpl – це службовий файл, який може містити функції та шаблони, що використовуються в інших файлах чарту.

templates/NOTES.txt є файлом з нотатками, які виводяться після успішного розгортання чарту.

values.yaml – це файл, в якому можна визначити значення за замовчуванням для параметрів чарту. Ці значення можна перезаписати при встановленні чарту.

Наразі додаток готовий до публікації за допомогою хмарних провайдерів. Оскільки будь-яка хмара надає можливості створювати свій Kubernetes cluster, для прикладу візьмемо один з трьох найпопулярніших сервісів, а саме – GCP (Google Cloud Platform).

За допомогою GCP Console або інтерфейсу командного рядка необхідно створити Kubernetes кластер. На цьому етапі потрібно визначити необхідні ресурси(процесорні ядра та кількість оперативної пам'яті) для кластера, також кількість його вузлів і розташування.

Для публікації додатку необхідно створити VPC(virtual private cloud), котрий ізолює в собі відповідні мережі та гарантує безпеку додатків всередині себе. Необхідно також перевірити всі служби, розгалуження та інші параметри, необхідні для стабільної роботи даного додатку.

Використовуючи Helm з командного рядка локального ПК, потрібно запустити процес публікації додатку.

Важливою складовою будь-якого додатку є моніторинг його роботи, відповідно, Kubernetes пропонує велику кількість інструментів для цього. Найпопулярнішими є інструменти моніторингу, такі як Prometheus, і системи логування (наприклад, ELK стек), для відстеження роботи додатку в Kubernetes [14].

Після налаштування безпеки, такі як TLS-шифрування для служб, обмеження доступу, та інших параметрів, щоб забезпечити безпеку додатку, необхідно відстежувати стан додатку та кластера в цілому за допомогою тих моніторингових інструментів, що були згадані раніше. Важливим також буде організація використання SSL сертифікатів для безпечної передачі даних за протоколом

Цей процес дозволить публікувати та управляти додатком на Google Cloud Platform, забезпечуючи його доступність, масштабованість та стійкість у розподіленому середовищі.

Тестування програмного забезпечення є невід'ємною частиною процесу розробки, спрямованою на забезпечення якості та надійності програми. Існує ряд методологій тестування, кожна з яких має свої переваги та особливості. В даному огляді буде розглянуто кілька ключових методологій тестування, а також обґрунтовано вибір автоматизованого unit-тестування для конкретного проєкту. Для розгляду обрано наступні методології:

- методологія водоспаду передбачає послідовний процес розробки, де кожна фаза завершується перед початком наступної. У водоспадній моделі тестування відбувається після завершення розробки. Основна перевага – простота у відстеженні та управлінні, але не ефективна для проєктів з частими змінами вимог.

- Scrum базується на ітераційному та інкрементальному підході до розробки. Тестування відбувається на кожному етапі розробки, і результати використовуються для вдосконалення продукту. Scrum сприяє гнучкості та відкритому спілкуванню.

- DevOps об'єднує розробку та операції, забезпечуючи автоматизацію та здатність швидко реагувати на зміни. Тестування стає автоматизованим, і весь цикл розробки стає більш прозорим та ефективним [16].

- методологія тестування моделей. Ця методологія фокусується на визначенні та тестуванні моделей, які визначають поведінку системи. Це включає тестування функціональних та нефункціональних вимог.

- автоматизоване unit-тестування є методологією, де окремі частини (одиниці) коду перевіряються на відповідність специфікаціям. Це ефективний підхід для виявлення та виправлення помилок на ранніх етапах розробки.

Враховуючи особливості даного проєкту, де важлива точність та надійність програмного забезпечення, вибір автоматизованого unit-тестування є виправданим. Це дозволить ефективно перевіряти окремі функції та компоненти системи на

відповідність вимогам та уникнути потенційних проблем на ранніх етапах розробки.

Наступна частина розробки застосунку, відіграє важливу роль у розробці надійних та ефективних додатків. Тестування є необхідною складовою кожного проєкту, спрямованого на створення високоякісного програмного продукту. У даному розділі буде детально описано реалізацію unit-тестування для ключових функцій нашого проєкту.

Однією з найефективніших методологій тестування є unit-тестування, яке дозволяє перевірити коректність роботи окремих фрагментів коду, ізолювати помилки та підтримувати стабільність програмного забезпечення. Маємо необхідність розглянути реалізацію unit-тестування для важливих функцій додатку за допомогою бібліотеки ``unittest`` та інших інструментів, що сприяють надійності та якості програмного продукту [17].

Описані в цьому розділі тести допомагатимуть впевнитися в правильності функцій, їхній відповідності вимогам та забезпечать можливість швидко знаходити помилки та потенційні слабкі місця коду в разі змін у коді. Детальний розгляд призначення та реалізації тестів дозволить краще зрозуміти переваги unit-тестування в контексті розробки нашого додатку.

Оскільки для того щоб протестувати одну функцію, необхідно написати декілька, буде висвітлено не весь функціонал тестування даного додатку, проте, оскільки даний додаток буде опубліковано, то за посиланням, що вказано у джерелах, можна знайти повний код застосунку та повну версію файлу `tests.py` (див. рисунок 3.14).

```

1  import unittest
2  from unittest.mock import patch, MagicMock
3  from flask import flash
4
5  # Імпортуємо функції, які будемо тестувати
6  from controllers import create_user, create_exam, create_question, \
7  get_all_exam_names, get_questions_for_exam, get_user_by_id, \
8  count_correct_answers, get_all_exams
9
10 class TestYourApp(unittest.TestCase):
11
12     # Тестуємо функцію create user
13     def test_create_user(self):
14         with patch('your_module.session') as mock_session:
15             # Створюємо об'єкт користувача
16             create_user('John Doe', 'john_doe', 'student', 'password123')
17
18             # Перевіряємо, чи викликали методи session правильно
19             mock_session.add.assert_called_once()
20             mock_session.commit.assert_called_once()
21
22             # Перевіряємо, чи виведено правильне повідомлення flash
23             flash.assert_called_once_with('User created successfully', 'success')
24
25     # Тестуємо функцію create_exam
26     def test_create_exam(self):
27         with patch('your_module.session') as mock_session:
28             # Створюємо об'єкт екзамену
29             create_exam('Math Exam')
30
31             # Перевіряємо, чи викликали методи session правильно
32             mock_session.add.assert_called_once()
33             mock_session.commit.assert_called_once()
34
35             # Перевіряємо, чи виведено правильне повідомлення flash
36             flash.assert_called_once_with('Exam created successfully', 'success')
37
38     # Тестуємо функцію create_question
39     def test_create_question(self):
40         with patch('your_module.session') as mock_session:
41             # Створюємо об'єкт питання
42             create_question(1, 'What is 2 + 2?', '4', '5', '6', '4')
43
44             # Перевіряємо, чи викликали методи session правильно
45             mock_session.add.assert_called_once()
46             mock_session.commit.assert_called_once()
47
48             # Перевіряємо, чи виведено правильне повідомлення flash

```

Рисунок 3.14 – Частина файлу tests.py

Unit-тестування є практикою в програмуванні, яка полягає в тестуванні окремих компонентів (функцій, методів, класів) програмного коду для визначення, чи працюють вони вірно та відповідають специфікації [18].

Основні принципи що реалізовані у unit-тестуванні даного додатку:

- ізоляція компонентів, позаяк unit-тести повинні тестувати окремі компоненти коду без залежностей від інших частин програми, що дозволяє виявляти помилки в конкретних функціях чи методах.

- тести мають бути автоматизовані, тобто вони повинні виконуватися автоматично, без необхідності ручного втручання, що забезпечує швидке виявлення помилок та зручність в управлінні тестами.

- незалежність, відповідно результати тестування одного компонента не повинні впливати на інші тести, з чого випливає те, що кожен тест повинен виконуватися в ізольованому середовищі.

- тести повинні бути простими та ефективними у написанні. Це сприяє легшому та швидшому розширенню тестового покриття.

- тести слід створювати так, щоб вони перевіряли коректність роботи конкретного компонента.

- тестовий код також повинен бути відстежуваним системою контролю версій, так само, як і основний код програми.

Буде розглянуто дві перші функції висвітленого вище файлу `tests.py`: `test_create_user` та `test_create_exam`. Тести, що реалізовані через ці функції мають на меті перевірку правильності роботи функцій `create_user` та `create_exam`.

Тест `test_create_user`:

- `mocking session` означає що використовуючи `patch`, ми “підробляємо” об’єкт `session`, щоб перевірити, чи правильно викликаються його методи;

- виклик функції створить об’єкт користувача, викликаючи функцію `create_user` зі зазначеними параметрами;

- перевірка методів `session` перевірить чи правильно викликали методи `session.add` та `session.commit`;

- перевірка повідомлення `flash` перевірить, чи виведено правильне повідомлення `flash`.

Тест `test_create_exam`:

- аналогічно до першого тесту, об’єкт `session` підробляється для відтворення реальних умов;

- створюється об’єкт екзамену, викликаючи функцію `create_exam` зі зазначеною назвою.

- перевіряються методи `session`, відповідно чи правильно викликано методи `session.add` та `session.commit`.

- перевірка чи виведено правильне повідомлення `flash`.

Ці тести допомагають переконатися, що функції створення користувача та

екзамену працюють коректно та взаємодіють із сесією та іншими компонентами програми. Вони допомагають переконатися, що виправлення чи зміни в цих функціях не вплинуть на їхню коректність.

Інші функції у даному файлі реалізовані з прямим слідуванням тим принципам, що були викладені вище. Таким за прикладом даних функцій, розробник додатку має можливість розширювати і функціонал тестування, притримуючись того ж самого стилю написання коду та тестів. Відповідно, у вищезгаданому репозиторії на GitHub, посилання на котрий буде залишено у джерелах, даний код буде прокоментований та пояснений.

Для визначення ефективності розробленого вебзастосунку проводиться комплексна оцінка, що включає аналіз різноманітних аспектів, таких як продуктивність, зручність використання, стабільність та інші. Дана оцінка спрямована на забезпечення оптимального функціонування та задоволення потреб користувачів [19].

Першим критерієм оцінки є продуктивність вебзастосунку. Використовуючи інструменти для вимірювання швидкодії та завантаження сторінок, аналізуються час відповіді сервера, завантаження ресурсів та інші параметри для забезпечення ефективності та зручності використання [20].

Оцінка зручності використання заснована на аналізі інтерфейсу користувача. Здійснюється перевірка логічності та інтуїтивності елементів керування, забезпечення навігації, адаптивності до різних пристроїв, а також вивчення відгуків користувачів.

Аспекти стабільності та безпеки грають ключову роль. Проводиться аналіз стійкості до помилок, виявлення та виправлення можливих вразливостей, перевірка забезпечення конфіденційності даних користувачів.

Важливим етапом є перевірка відповідності функціональності розробленого вебзастосунку визначеним вимогам. Аналізується виконання основних завдань та можливість розширення функціоналу.

Здійснюється порівняння отриманих результатів із початковими вимогами та цілями проєкту для визначення ступеня виконання задуманого.

Виходячи з вищенаведених критеріїв оцінки програмного забезпечення, є необхідним провести оцінку даного додатку на відповідність висвітленим критеріям та визначити ступінь відповідності вимогам, що визначені метою дипломної роботи. Даний аналіз буде проведено для кожної вимоги:

Розроблений вебзастосунок демонструє високий рівень продуктивності. Під час тестування встановлено, що час відповіді сервера невеликий, що сприяє швидкому завантаженню сторінок. Крім того, оптимізація ресурсів дозволяє уникнути зайвого навантаження та підтримує стабільну роботу, навіть при великій кількості одночасних запитів.

Інтерфейс користувача розробленого вебзастосунку є інтуїтивно зрозумілим та логічно побудованим. Всі елементи керування розташовані логічно, забезпечуючи зручну навігацію для користувачів будь-якого рівня. Адаптивний дизайн дозволяє ефективно використовувати застосунок на різних пристроях, що покращує його загальну зручність використання.

Розроблений вебзастосунок виявляє високий рівень стабільності та безпеки. Під час тестування були виявлені та виправлені потенційні вразливості. Система стійка до помилок та здатна ефективно вирішувати можливі проблеми, що підвищує загальний рівень безпеки.

Функціональність розробленого вебзастосунку повністю відповідає визначеним вимогам та завданням. Усі основні функції працюють правильно, і користувач може легко виконувати необхідні завдання. Під час тестування було перевірено взаємодію з базою даних, обробку форм, а також інші ключові аспекти функціональності.

Розроблений вебзастосунок успішно відповідає початковим вимогам та цілям проєкту. Всі визначені функції та можливості реалізовані, що дозволяє вважати проєкт успішним у досягненні своїх основних цілей.

Загалом, оцінка ефективності розробленого вебзастосунку є дуже позитивною. Високий рівень продуктивності, зручність використання та стабільність роботи роблять його привабливим для користувачів, а відповідність функціональності вимогам дозволяє вважати проєкт успішним.

На основі проведеної комплексної оцінки можна зробити висновки щодо ефективності розробленого вебзастосунку та визначити можливі напрями для подальшого вдосконалення.

У ході роботи над проєктом було визначено кілька ключових цілей, які успішно досягнуті.

Розробка вебзастосунку для онлайн-школи “GoITeens” була виконана, забезпечуючи відповідність основним потребам навчального закладу. Застосунок отримав позитивний відгук від користувачів та адміністраторів онлайн-школи.

У процесі створення вебзастосунку акцент був зроблений на гнучкість, інтерактивність та персоналізацію навчання. Інтеграція інтерактивних елементів та можливостей персоналізації допомогла зробити навчальний процес більш цікавим та ефективним.

Створений вебзастосунок також покликаний підвищити мотивацію та результативність навчання. Впровадження інтерактивних завдань та персоналізованих підходів позитивно вплинуло на активність учнів та їхні досягнення.

Особливу увагу приділено доступності вебзастосунку для учнів з різними потребами. Застосування принципів універсального дизайну сприяє включенню різних груп користувачів.

Потенційний вплив проєкту на розвиток онлайн-школи та неформальної освіти розглядається у контексті можливостей використання вебзастосунку для організації дистанційних курсів та поліпшення існуючих платформ дистанційного навчання.

Можна визначити, що розроблений вебзастосунок відповідає визначеним цілям та створює перспективи для подальшого росту та вдосконалення системи онлайн-освіти.

ВИСНОВКИ

Для визначення ефективності розробленого вебзастосунку проводиться комплексна оцінка, що включає аналіз різноманітних аспектів, таких як продуктивність, зручність використання, стабільність та інші. Дана оцінка спрямована на забезпечення оптимального функціонування та задоволення потреб користувачів.

Продуктивність вебзастосунку є першим критерієм оцінки. Для цього використовуються інструменти для вимірювання швидкодії та завантаження сторінок, що дозволяє аналізувати час відповіді сервера, завантаження ресурсів та інші параметри, важливі для забезпечення ефективності та зручності використання.

Оцінка зручності використання базується на аналізі інтерфейсу користувача. Перевіряється логічність та інтуїтивність елементів керування, забезпечення навігації, адаптивність до різних пристроїв, а також враховуються відгуки користувачів.

Стабільність та безпека відіграють ключову роль. Проводиться аналіз стійкості до помилок, виявлення та виправлення можливих вразливостей, а також перевіряється забезпечення конфіденційності даних користувачів.

Важливим етапом є перевірка відповідності функціональності розробленого вебзастосунку визначеним вимогам. Аналізується виконання основних завдань та можливість розширення функціоналу.

Здійснюється порівняння отриманих результатів із початковими вимогами та цілями проєкту для визначення ступеня виконання задуманого. Виходячи з вищенаведених критеріїв оцінки програмного забезпечення, необхідно провести оцінку даного додатку на відповідність висвітленим критеріям та визначити ступінь відповідності вимогам, що визначені метою дипломної роботи.

Розроблений вебзастосунок демонструє високий рівень продуктивності. Під час тестування встановлено, що час відповіді сервера невеликий, що сприяє швидкому завантаженню сторінок. Крім того, оптимізація ресурсів дозволяє уникнути зайвого навантаження та підтримує стабільну роботу, навіть при великій

кількості одночасних запитів.

Інтерфейс користувача розробленого вебзастосунок є інтуїтивно зрозумілим та логічно побудованим. Всі елементи керування розташовані логічно, забезпечуючи зручну навігацію для користувачів будь-якого рівня. Адаптивний дизайн дозволяє ефективно використовувати застосунок на різних пристроях, що покращує його загальну зручність використання.

Розроблений вебзастосунок виявляє високий рівень стабільності та безпеки. Під час тестування були виявлені та виправлені потенційні вразливості. Система стійка до помилок та здатна ефективно вирішувати можливі проблеми, що підвищує загальний рівень безпеки.

Функціональність розробленого вебзастосунок повністю відповідає визначеним вимогам та завданням. Усі основні функції працюють правильно, і користувач може легко виконувати необхідні завдання. Під час тестування було перевірено взаємодію з базою даних, обробку форм, а також інші ключові аспекти функціональності.

Розроблений вебзастосунок успішно відповідає початковим вимогам та цілям проєкту. Всі визначені функції та можливості реалізовані, що дозволяє вважати проєкт успішним у досягненні своїх основних цілей. Загалом, оцінка ефективності розробленого вебзастосунок є дуже позитивною. Високий рівень продуктивності, зручність використання та стабільність роботи роблять його привабливим для користувачів, а відповідність функціональності вимогам дозволяє вважати проєкт успішним.

На основі проведеної комплексної оцінки можна зробити висновки щодо ефективності розробленого вебзастосунок та визначити можливі напрями для подальшого вдосконалення.

Закінчивши розробку вебзастосунок для онлайн-школи “GoITeens”, визначено кілька напрямків для подальших вдосконалень та розширень.

Однією з можливостей є розширення функціоналу інтерактивних завдань. Додавання різноманітних типів завдань, таких як віртуальні лабораторії, відеоуроки чи інтерактивні тести, може розширити можливості навчання та

підвищити привабливість для учнів.

Для збільшення персоналізації можна розглядати можливість впровадження системи індивідуального відстеження прогресу. Це дозволить студентам відстежувати свій власний успіх та вчителям – надавати більш точне та персоналізоване навчання.

Окрім того, слід розглядати можливість інтеграції засобів відгуків. Введення системи зворотного зв'язку від учнів та вчителів може визначити слабкі та сильні сторони застосунку, що дозволить активніше вдосконалювати його.

Також, буде розглянуто підтримку різних типів користувачів. Покращення доступності для людей з особливими потребами або вдосконалення інтерфейсу для різних вікових груп може розширити аудиторію та забезпечити більший вплив.

У цілому, проєкт відкриває двері до численних можливостей для подальшого розвитку та покращення в сфері онлайн-освіти, створюючи потенціал для покращення якості навчання та розширення аудиторії користувачів.

У результаті проведеного аналізу, розробки та дослідження вебзастосунку для онлайн-школи “GoITeens”, маємо змогу висунути ряд важливих висновків. Розроблений вебзастосунок дотримується основних принципів сучасної онлайн-освіти та враховує актуальні потреби користувачів.

Важливо відзначити необхідність розширення можливостей для дистанційного навчання, яке стає особливо актуальним в умовах сучасного світу. Врахування різних стилів навчання та індивідуальних потреб користувачів зробить застосунок вкрай гнучким та привабливим для широкого кола учнів.

В процесі розробки даного застосунку для перевірки знань учнів були враховані ключові вимоги та потреби освітнього середовища. Застосунок успішно втілює ідею надання гнучкого та інтерактивного інструменту для навчання, сприяючи підвищенню мотивації та результативності учнів.

Однією з ключових особливостей є можливість персоналізації навчання, що дозволяє адаптувати матеріали та завдання під індивідуальні потреби користувачів. Майбутня інтеграція системи оцінювання та надання зворотного зв'язку сприятиме якісному моніторингу прогресу.

Важливо відзначити, що розроблений застосунок може стати важливим інструментом для покращення дистанційного навчання, особливо в умовах сучасних реалій. Забезпечуючи доступ до навчання з будь-якого місця та за будь-яких умов, він відповідає сучасним викликам освіти.

Проект не лише відповідає завданням, але й має потенціал для подальшого розвитку. Рекомендації щодо розширення функціоналу та вдосконалень у напрямку автоматизованого тестування можуть сприяти подальшій еволюції цього важливого інструменту для освіти.

Використані технології у процесі розробки вебзастосунку забезпечили ефективну та сучасну архітектуру системи. Використання Flask та SQLAlchemy у бекенд частині застосунку дозволило створити потужну та гнучку систему для обробки запитів та взаємодії з базою даних.

Використання Jinja2 для шаблонізації HTML створило можливість ефективного відображення динамічного контенту та простоти у розробці фронтенду. Технології Docker, Helm, та Kubernetes надали зручний інструментарій для контейнеризації, публікації та управління застосунком, що відповідає найсучаснішим стандартам розробки та розгортання.

Такий технологічний набір дозволив ефективно реалізувати всі поставлені завдання та створити продукт, який відповідає вимогам сучасної інноваційної освіти. Застосування сучасних технологій допомагає не тільки створити продукт високої якості, але і підготувати його до швидкого та ефективного впровадження в реальні умови використання.

Підсумовуючи, проект відповідає поставленим завданням, а також має потенціал для подальших розширень та вдосконалень. З урахуванням сучасних тенденцій у сфері освіти, розроблений вебзастосунок може відкрити нові можливості для розвитку онлайн-навчання та стати важливим інструментом для покращення якості освіти в цифрову епоху.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Coursera official web site. URL: <https://www.coursera.org/> (date of access: 30.05.2024).
2. MDN Web Docs HTML. MDN Web Docs. URL: <http://surl.li/renhq> (date of access: 30.03.2024).
3. MDN Web Docs CSS. MDN Web Docs. URL: <http://surl.li/ubhsp> (date of access: 07.04.2024).
4. Javascript online – URL: <https://www.javascript.com/> (date of access: 08.06.2024).
5. Google Cloud Platform Tutorial. Google. URL: <https://cloud.google.com/> (date of access: 15.05.2024)
6. Understanding the Role of Digital Technologies in Education: A review / P. A. Haleem et al. Sustainable Operations and Computers. 2022. Vol. 3. Pp. 275-285. doi: 10.1016/j.susoc.2022.05.004.
7. Martynenko S. Ensuring the quality of university education in conditions of blended and distance learning. Continuing Professional Education: Theory and Practice. 2020. No. 4. P. 7 – 13. URL: <https://doi.org/10.28925/1609-8595.2020.4.1> (date of access: 10.06.2024).
8. Welcome to Flask – Flask Documentation (2.1.x). Welcome to Flask – Flask Documentation (3.0.x). URL: <http://surl.li/renfe> (date of access: 04.03.2024).
9. Verification of ORM-based controllers by summary inference / G. Chawla et al. ICSE '22: 44th International Conference on Software Engineering, Pittsburgh Pennsylvania. New York, NY, USA, 2022. URL: <https://doi.org/10.1145/3510003.3510148> (date of access: 10.06.2024).
10. Abbas A. M. Server-Side Rendering / Ashraf Mughal Abbas. 2023. URL: <http://surl.li/ujwqk> (date of access: 07.06.2024).
11. Pierre, Guillaume, and Arif Ahmed. "Docker-pi: Docker container deployment in fog computing infrastructures." International Journal of Cloud Computing 9, URL: <http://dx.doi.org/10.1504/ijcc.2020.10027559> (date of access: 07.06.2024).
12. Carrión C. Kubernetes Scheduling: Taxonomy, ongoing issues and challenges. ACM

Computing Surveys. 2022. URL: <https://doi.org/10.1145/3539606> (date of access: 10.06.2024).

13. Burns B., Hightower K., Beda J. Kubernetes: Up and Running: Dive into the Future of Infrastructure; Reilly Media, 2017. 202 p. URL: <http://surl.li/rfcjr>.

14. Khushalani P. Security and Monitoring. Kubernetes Application Developer. Berkeley, CA, 2022. P. 109 – 134. URL: https://doi.org/10.1007/978-1-4842-8032-4_5 (date of access: 02.06.2024).

15. Kubernetes Security: A Quick Reference / S. Shaikh et al. International Journal of Advanced Research in Science, Communication and Technology. 2021. P. 386 – 393. URL: <https://doi.org/10.48175/ijarsct-1152> (date of access: 10.06.2024).

16. DEVOPS. International Research Journal of Modernization in Engineering Technology and Science. 2022. URL: <https://doi.org/10.56726/irjmets30906> (date of access: 03.06.2024).

17. Pajankar A. Python Unit Test Automation. Berkeley, CA : Apress, 2022. URL: <https://doi.org/10.1007/978-1-4842-7854-3> (date of access: 10.06.2024).

18. Percival H.J. Test-Driven Development with Python. O’Reilly Media, Incorporated, 2017. URL: <http://surl.li/rfcew>.

19. Python for web development / U. Patkar et al. International Journal of Computer Science and Mobile Computing. 2022. Vol. 11, no. 4. P. 36 – 48. URL: <https://doi.org/10.47760/ijcsmc.2022.v11i04.006> (date of access: 10.06.2024).

20. Mendes E., Kitchenham B. Web Productivity Measurement and Benchmarking. Web Engineering. Berlin/Heidelberg. P. 75 – 106. URL: https://doi.org/10.1007/3-540-28218-1_3 (date of access: 10.06.2024).

ДОДАТКИ

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

**На розробку вебзастосунку організації дистанційних курсів
неформальної освіти**

Погоджено:

Кандидат економічних наук, доцент кафедри
кібернетики та інформатики

Студент групи ICT 2001 с.т.

Лапченко Іван Володимирович

СУМИ 2024

1. Призначення й мета створення вебзастосунку

1.1 Призначення вебзастосунку

Вебзастосунок призначений для підвищення якості надання послуг неформальної освіти онлайн ІТ-академією GoITeens.

1.2 Мета створення вебзастосунку

Розробка вебзастосунку організації дистанційних курсів неформальної освіти, що буде використаний у навчальних цілях ІТ-академією GoITeens.

1.3 Цільова аудиторія

У цільовій аудиторії вебзастосунку можна виділити такі групи:

1. Викладачі
2. Учні

2 Вимоги до вебзастосунку

2.1 Вимоги до вебзастосунку в цілому

2.1.1 Вимоги до структури й функціонування вебзастосунку

Вебзастосунок повинен бути реалізований у вигляді сайту, доступного в мережі Інтернет. Сайт повинен складатися із взаємозалежних розділів із чітко розділеними функціями. Архітектура вебзастосунку має реалізовувати клієнт-серверний підхід, при використанні server-side rendering, відповідно вебзастосунок має працювати як у клієнта в браузері так і на сервері.

2.1.2 Вимоги до персоналу

Від персоналу не має вимагатися особливих технічних навичок для підтримки й експлуатації вебзастосунку, окрім загальних навичок роботи з персональним комп'ютером і стандартним веб-браузером (наприклад, Google Chrome).

2.1.3 Вимоги до збереженні вебзастосунку

Уся інформація надана у вебзастосунку буде зберігатися у базі даних реалізованій засобами системи управління базами даних MySQL.

2.1.4 Вимоги до розмежування доступу

- Розроблюваний вебзастосунок не має бути приватним.
- Відповідно до прав доступу у вебзастосунок, користувачів можна поділити на робітників, учнів та адміністратора.
- Робочі аккаунти можуть переглядати усі сторінки вебзастосунку, після авторизації за логіном та паролем.
- Адміністратор може редагувати зовнішній вигляд та наповнення вебзастосунку.

2.2 Структура вебзастосунку

2.2.1 Навігація

- Відповідно до бажаного дизайну вебзастосунку, для навігації, у шапці буде створена система контент меню. Меню необхідне для швидкого переміщення користувача по усім доступним сторінкам. Меню буде відображатися на всіх сторінках, щоб відвідувач міг в будь-який момент часу перейти на будь-яку

сторінку вебзастосунку.

2.2.2 Наповнення вебзастосунку (контент)

—

— Сторінки всіх розділів сайту мають бути автоматично створені на основі даних з бази даних на сервері. Редагування вмісту розділів повинно бути можливим через адміністраторський web-інтерфейс, який дозволяє редагувати інформаційний вміст сторінок сайту без необхідності програмування. Наповнення сторінок інформацією повинне бути здійснюватися за допомогою шаблонів сайту.

2.3 Вимоги до функціонування системи

2.3.1 Потреби користувача

Система керування контентом (адміністративна частина) повинна надавати можливість додавання, редагування й видалення вмісту сторінок. Також повинна бути передбачена можливість додавання інформації без відображення на сайті для користувачів.

2.3.2 Функціональні вимоги

На головній сторінці будуть представлені такі елементи:

- Меню навігації;
- Список доступних для проходження тестів;

2.3.3 Системні вимоги

Системні вимоги відповідно до потреб користувачів:

- наявність реєстрації та авторизації.
- редагування викладачами тестів;

- створення, редагування, видалення, даних в тестах викладачами;
- можливість проходження учнями тестів;

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Реалізація вебзастосунку відбувається з використанням:

- JS ES14
- Python 3.12
- CSS3
- HTML5
- SQLite 3.46.0
- Kubernetes 1.29
- Helm 3.14

2.4.2 Вимоги до лінгвістичного забезпечення

Вебзастосунок має бути виконаний українською мовою.

2.4.3 Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

- Веб-браузер: Internet Explorer 7.0 і вище, або Firefox 4.5 і вище, або Safari 3.3.1 і вище, або Chrome 3 і вище.

3 Склад і зміст робіт зі створення вебзастосунку

Докладний опис етапів роботи зі створення вебзастосунку наведено в таблиці

А.1.

Таблиця А.1 – Етапи створення вебзастосунку

№	Склад і зміст робіт	Строк розробки
1	Аналіз та планування	4 дні
2	Вивчення аналогів	1 днів
3	Вибір технологічного стеку	5 днів
5	Розробка функціональності	5 днів
6	Розробка інфраструктури	5 днів
7	Розробка бази даних	3 днів
8	Веб-дизайн та UX/UI	4 днів
10	Комплексне тестування	1 день
11	Виправлення помилок та вдосконалення	1 дні
13	Впровадження та підтримка	2 день
	Загальна тривалість робіт	31 днів

4. Вимоги до складу й змісту робіт із введення вебзастосунку в експлуатацію

Для того, щоб вебзастосунком могли користуватися робітники селищної ради необхідно розмістити його у мережі Інтернет, тому необхідно придбати доменне ім'я та місце на хостингу. На хостинг переноситься вебзастосунок і наповнення бази даних з подальшою їх доробкою. Для коректного переносу вебзастосунку на хостинг необхідно, щоб параметри хостинга відповідали вимогам, зазначеним у ТЗ.

ДОДАТОК Б

ПЛАНУВАННЯ РОБІТ

Сьогоднішній світ динамічний, мінливий і сповнений нових викликів. У цій епосі безперервного розвитку освіта стає не просто привілеєм, а й життєвою необхідністю, ключем до самореалізації та успішної кар'єри. Традиційні формати навчання, хоча й залишаються важливими, вже не можуть задовольнити всі зростаючі потреби суспільства.

Важливу роль у цій трансформації відіграє неформальна освіта, яка виходить за рамки жорстких програм та стандартів, пропонуючи гнучкий та індивідуальний підхід до набуття знань та навичок. Вона дає можливість людям будь-якого віку, з різним досвідом та потребами розширювати свій кругозір, вдосконалювати особисті компетенції та йти в ногу з часом.

Визначення початку і завершення завдань, розподіл робочих навантажень та встановлення контрольних точок, моніторинг їх виконання у порівнянні з початковим планом. Корекція графіку за потреби, оптимізація використання людських і матеріальних ресурсів. Інформування команди про статус проєкту, заплановані завдання та строки їх виконання все це потрібно для успішного роботи і з усім цим допомагає Діаграма Ганта (див. рисунок Б.1).

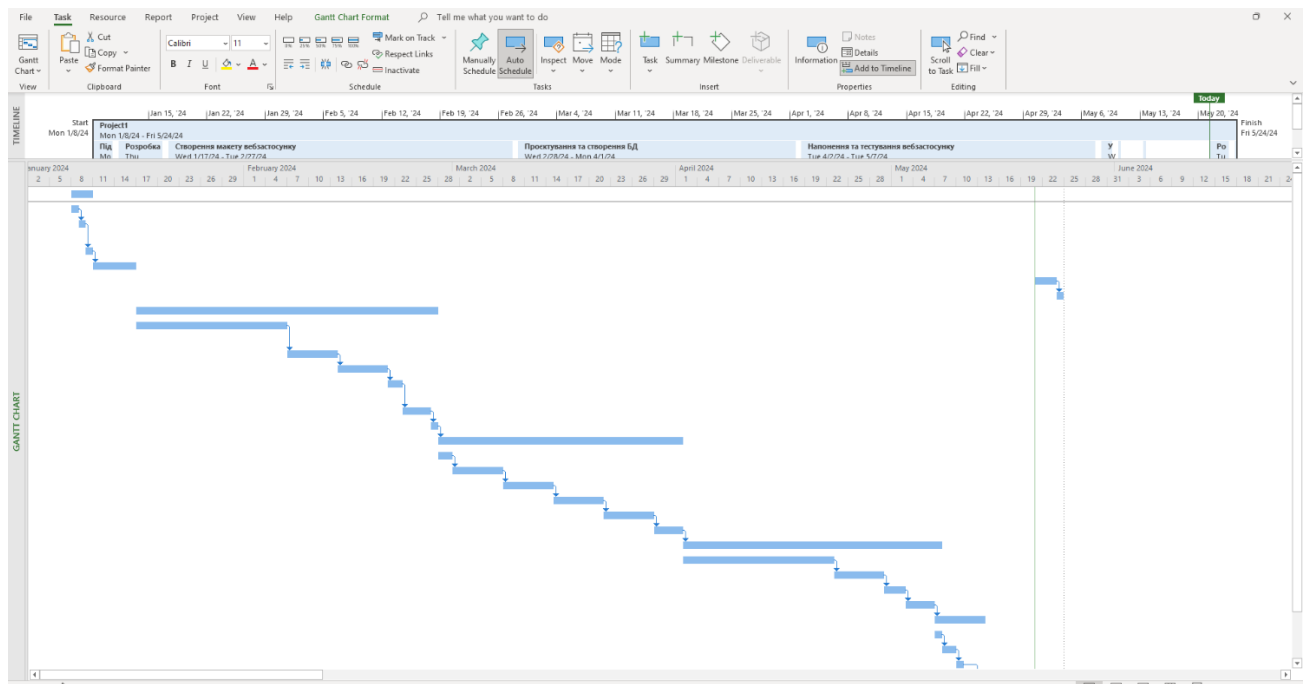


Рисунок Б.1 – Діаграма Ганта

Оцінка ризиків в ІТ-проєкті створенню вебзастосунку організації курсів дистанційної освіти відіграє ключову роль у забезпеченні успішного виконання проєкту. Вона включає в себе ідентифікацію, аналіз і управління ризиками, що можуть вплинути на досягнення цілей проєкту. Основними компонентами оцінки ризиків є опис ризику, імовірність виникнення, вплив на проєкт, стратегії управління та відповідальні особи.

Застосування цих компонентів дозволяє команді проєкту вчасно виявляти потенційні загрози, оцінювати їх серйозність і розробляти заходи для мінімізації або усунення негативних наслідків. Процес оцінки ризиків починається з ідентифікації можливих ризиків, таких як технічні проблеми, затримки у розробці, бюджетні обмеження або зміни у вимогах замовника.

Визначення відповідальних осіб за управління кожним ризиком є важливим аспектом, оскільки це забезпечує чітку відповідальність та своєчасну реалізацію заходів щодо управління ризиками. Регулярний моніторинг та перегляд ризиків протягом усього життєвого циклу проєкту допомагають вчасно виявляти нові ризики та коригувати стратегії управління.

Отже, систематична оцінка ризиків (див. таблицю Б.1) є невід'ємною частиною управління ІТ-проектами, яка забезпечує їх успішне виконання та досягнення поставлених цілей.

Таблиця Б.1 – Оцінка ризиків ІТ-проекту.

Ризики	Імовірність виникнення	Вплив на проєкт	Стратегії управління	Відповідальні особи
Втрата даних	Низька	Висока	Регулярна перевірка резервне копіювання даних, застосування шифрування	ІТ Фахівець
Відмова обладнання	Низька	Середня	Планування та впровадження запасних/резервних пристроїв, регулярне технічне обслуговування	ІТ Фахівець
Відмова сервера через перевищення навантаження	Середня	Середній	Масштабування серверної інфраструктури, впровадження механізмів автоматичного моніторингу	ІТ Фахівець
Затримка в розробці через зміни вимог	Середня	Середній	Узгодження докладних вимог до функціоналу, впровадження гнучких методологій розробки	Еколог
Проблеми з безпекою	Низька	Високий	Регулярні аудити безпеки, впровадження стандартів безпеки, навчання персоналу з питань безпеки	ІТ-фахівець
Зміна нормативних вимог	Низька	Високий	Моніторинг законодавства, залучення юридичних консультантів, адаптація до нових вимог	Еколог
Збої в ПЗ	Низька	Середній	Регулярна перевірка ПЗ, тестування, планування альтернативних варіантів	ІТ-фахівець

ДОДАТОК В

КОПІЇ ПУБЛІКАЦІЙ



Сумський національний аграрний університет



Національний технічний університет «ХПІ»



Політехніка Свентокржинська в Кельцах (Польща)



ТОВ «ТРІЗ»



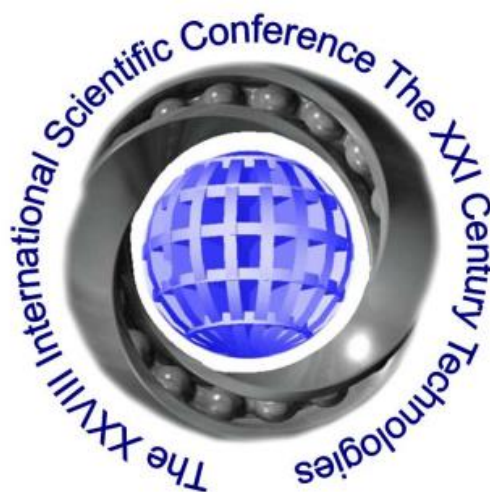
Сумський державний університет



Державний біотехнологічний університет



Українська технологічна академія



ТЕХНОЛОГІЇ ХХІ СТОРІЧЧЯ

Збірник тез за матеріалами 29^{ої} міжнародної науково-практичної конференції
(20-22 листопада 2023 р.)

Частина 2

Секції: «Прогресивні технології в економіці»,
«Прогресивні технології в екології та енергоефективність»,
«Прогресивні технології у педагогіці»

Суми – 2023

ДЕЯКІ АСПЕКТИ ЗАСТОСУВАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ НАВЧАННЯ

Постановка проблеми. В теперішній час інформаційні технології мають значний вплив на систему освіти. Пандемія COVID-19, воєнний стан в Україні стали чинниками впровадження дистанційної освіти, що ще більше ініціювало розвиток цифрових технологій в освіті. Інформаційні технології суттєво вплинули на зміну якості всієї системи освіти. Інформаційні технології навчання є постачальником знань, співавтором інформації, наставником, оцінювачем та тренажером. Отже, дослідження теоретичних та практичних аспектів застосування інформаційних технологій в навчанні є актуальним.

Викладення основного матеріалу. Аналіз наукових робіт щодо застосування інформаційних технологій (далі – ІТ) навчання надає можливість краще розуміти позитивні та негативні наслідки використання ІТ в освіті, з'ясувати практичні аспекти впровадження цих технологій у навчальний процес. В роботі [1] проводиться дослідження щодо необхідності застосування цифрових технологій в освіті, охарактеризовано основні застосунки для навчання та обговорюються можливі виклики в освіті. Автори визначили наступні напрямки застосування ІТ навчання, що висвітлюються у наукових дослідженнях: підвищення продук-

тивності навчання, розвиток онлайн-бібліотек, сприяння дистанційному навчанню, навчання студентів з особливими потребами, створення віртуального класу, формування знань та навичок розуміння, створення інклюзивного навчального середовища, розвиток командної роботи та навичок спілкування, вирішення навчальних завдань, розширений доступ до освітніх ресурсів, допомога учням з низьким базовим рівнем в завершенні навчального плану, інноваційний спосіб навчання, заняття з арифметики, зручне викладання та навчання, цифрова освіта, створення незвичайних занять, розробка навчальних програм та допоміжних матеріалів, поліпшення успішності учнів, гнучка освіта, набуття здібностей до самонавчання, розширення знань, усунення прогалин у навчанні, прискорення отримання інформації, електронні книги, проведення онлайн-занять через платформу МООС, навчання на основі відео, збільшення освітніх можливостей, доступ до сучасного навчального матеріалу, подолання освітніх бар'єрів, екологічні переваги, динамічне навчання, зменшення навантаження на вчителя, оцінювання учнів у режимі реального часу, перехід до гібридного викладання та навчання. Надано характеристику зазначених напрямків та наведені посилання на наукові джерела, що присвячені дослідженням з вказаних напрямків. У дослідженні [2] обговорюються останні досягнення розвитку ІТ у секторі освіти, такі як гейміфікація, доповнена реальність, нові застосунки для навчання, Інтернет речей (IoT). В дослідженні [3] наведено огляд літератури з використання мобільних технологій та навчальних ігор у вищій освіті. Описуються переваги зазначеного альтернативного методу викладання та навчання.

Наведені джерела надають важливу інформацію, яка допомагає визначити позитивні аспекти використання ІТ в освіті та розробити стратегії подальшого впровадження цих технологій.

ЛІТЕРАТУРА

1. Understanding the Role of Digital Technologies in Education: A review / P. A. Haleem et al. *Sustainable Operations and Computers*. 2022. Vol. 3. Pp. 275-285. doi:10.1016/j.susoc.2022.05.004.
2. Zain S. Digital transformation trends in education. *Future Directions in Digital Information*. 2021. P. 223-234. doi:10.1016/b978-0-12-822144-0.00036-7.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**СУМСЬКИЙ НАЦІОНАЛЬНИЙ
АГРАРНИЙ УНІВЕРСИТЕТ**

МАТЕРІАЛИ

**науково-практичної конференції
викладачів, аспірантів та студентів
Сумського НАУ**

(14-16 травня 2024 р.)

Luo Hao. A STUDY ON CONSUMER BEHAVIOUR MANAGEMENT THROUGH ADVANCED MARKETING TOOLS	414
Wang Baichao. MANAGEMENT CRISIS IN AGRICULTURAL COMPANIES IN CURRENT SITUATIONS	415
Wang Ruijie. HUMAN RESOURCES MANAGEMENT OPTIMIZATION AND INNOVATION BASED ON ARTIFICIAL INTELLIGENCE	416
Yu Hang. RESEARCH ON THE INTRODUCTION OF MODERN MANAGEMENT TECHNOLOGY INTO ENTERPRISES	417
Zhou Kunming. PECULIARITIES OF MANAGING HIGH-TECH ENTERPRISES IN A TRANSPARENT ECONOMY	418
Бусенко І.В. АНАЛІЗ ОСНОВНИХ РІЗНОВИДІВ SQL-ІН'ЄКЦІЙ, АНАЛІТИКА ЗАСТОСУВАННЯ ТА СПОСОБИ ЗАХИСТУ ВІД НИХ	419
Белоконь П.А. РОЗРОБКА ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ ПІДТРИМКИ РОБОТИ HR-МЕНЕДЖЕРА	420
Стеблянюк В. В. РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ОБЛІКУ ВИТРАТ НА ВИРОБНИЦТВО ПРОДУКЦІЇ. РЕАЛІЗАЦІЯ ПРОЄКТУ	421
Фідірко Т.Р. РОЗРОБКА ВЕБЗАСТОСУНКУ ОБЛІКУ ТЕХНІЧНОГО СТАНУ ЛЕГКОВИХ АВТОМОБІЛІВ. ПОСТАНОВКА ЗАДАЧІ	422
Фідірко Т.Р. РОЗРОБКА ВЕБЗАСТОСУНКУ ОБЛІКУ ТЕХНІЧНОГО СТАНУ ЛЕГКОВИХ АВТОМОБІЛІВ. РЕАЛІЗАЦІЯ ПРОЄКТУ	423
Биченко Є.В. РОЗРОБКА ПРОТОТИПУ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА ВЕБОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПІДТРИМКИ УПРАВЛІННЯ ДІЯЛЬНІСТЮ ПІДПРИЄМСТВА	424
Корецький В.А. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ МОНІТОРИНГУ ІНТЕРНЕТ-ТРАФІКУ	425
Лапченко І. В. ТА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ОРГАНІЗАЦІЇ ДИСТАНЦІЙНИХ КУРСІВ НЕФОРМАЛЬНОЇ ОСВІТИ	426
Хоменко Р.М. ПРОЄКТУВАННЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА ЧАТ-БОТА ДЛЯ ЗРУЧНИХ ТА ЕФЕКТИВНИХ ОНЛАЙН-ЗАМОВЛЕНЬ ЕЛЕКТРОНІКИ	427
Блаженко Д. О. ПРОБЛЕМИ ПРОЄКТУВАННЯ БАЗИ ДАНИХ ПІДСИСТЕМИ ВЕДЕННЯ ЗАМОВЛЕНЬ ДЛЯ ВИГОТОВЛЕННЯ СІЛЬСЬКОГОСПОДАРСЬКОЇ ПРОДУКЦІЇ	428
Ковальов Р. С. ПРАКТИЧНЕ ПРОГРАМУВАННЯ ГЕНЕРАТОРІВ 3D КОНТЕНТУ НА ПРИКЛАДІ ГЕНЕРАТОРУ МЕША РЕЛЬЄФУ МІСЦЕВОСТІ	429
Ковальов Р. С. ПРАКТИЧНЕ ПРОГРАМУВАННЯ ГЕНЕРАТОРІВ 3D КОНТЕНТУ НА ПРИКЛАДІ ГЕНЕРАТОРУ МЕША РЕЛЬЄФУ МІСЦЕВОСТІ	430
Руденко О. А. МІНІМІЗАЦІЯ ЗАТРИМОК У ВЕБДОДАТКАХ ВИКОРИСТОВУЮЧИ ГРАНИЧНІ ТЕХНОЛОГІЇ	431
Сіденко А. С. ПРОБЛЕМИ РЕАЛІЗАЦІЇ ПІДСИСТЕМИ ОБЛІКУ ПРИРОДООХОРОННИХ ЗАХОДІВ	432
Терещенко С.С. АВТОМАТИЗАЦІЯ ОБЛІКОВИХ ПРОЦЕСІВ: КЛЮЧ ДО ЕФЕКТИВНОГО УПРАВЛІННЯ В АТЕЛЬЄ З ВИГОТОВЛЕННЯ ОДЯГУ	433
Михайлюк О.С. ОГРУНТУВАННЯ НЕОБХІДНОСТІ РОЗРОБКИ WEB-ОРІЄНТОВАНОЇ ПІДСИСТЕМИ ПІДТРИМКИ ОБЛІКОВИХ ПРОЦЕСІВ РЕГІОНАЛЬНОГО ІНТЕРНЕТ-ПРОВАЙДЕРА	434
Олійник А.В. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПІДСИСТЕМИ ОБЛІКУ ПАЛИВНО-МАСТИЛЬНИХ МАТЕРІАЛІВ В ТЕРИТОРІАЛЬНІЙ ГРОМАДІ	435
Чубун М.П. АНАЛІЗ ТА ВИЗНАЧЕННЯ ВИМОГ ДО СУЧАСНОГО ВЕБ-САЙТУ	436
Дубінчин В. С. ВИЗНАЧЕННЯ ФУНКЦІОНАЛЬНИХ ВИМОГ ДО ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ ОЦІНЮВАННЯ ПРАЦЕЗДАТНОСТІ КОМП'ЮТЕРИЗОВАНИХ РОБОЧИХ МІСЦЬ	437
Санжаков А. А. НОВИЙ ПІДХІД РОЗРОБКИ ДИЗАЙНУ ВЕБ-ЗАСТОСУНКІВ – MATERIAL 3	438
Федірко Д.Д. ПРОБЛЕМИ ПРОЄКТУВАННЯ ВЕБ-ЗАСТОСУНКІВ ОЦІНКИ ТАРИФНИХ ПРОПОЗИЦІЙ МОБІЛЬНИХ ОПЕРАТОРІВ УКРАЇНИ	439
Федірко Д.Д. РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ АНАЛІЗУ МОБІЛЬНИХ ОПЕРАТОРІВ УКРАЇНИ	440
Маслак О. М. ДЕРЖАВНЕ РЕГУЛЮВАННЯ ПРОЦЕСІВ АГРОПРОМИСЛОВОГО ВИРОБНИЦТВА НА РЕГІОНАЛЬНОМУ РІВНІ (НА ПРИКЛАДІ СУМСЬКОЇ ОБЛАСТІ)	441
Горбань С.В. ІМІДЖ ПІДПРИЄМСТВА: ВПЛИВ, ФОРМУВАННЯ ТА ОЦІНКА	442
Заболотна А.І. НОРМАТИВНО-ПРАВОВЕ ЗАБЕЗПЕЧЕННЯ ЕКОНОМІЧНОГО РОЗВИТКУ ДРУЖІВСЬКОЇ ТЕРИТОРІАЛЬНОЇ ГРОМАДИ	443
Гулініна О.Ю. ШЛЯХИ ОПТИМІЗАЦІЇ ОРГАНІЗАЦІЙНОЇ СТРУКТУРИ УПРАВЛІННЯ ЗАКЛАДУ ОХОРОНИ ЗДОРОВ'Я	444
Сущенко В. О. ПРОЦЕС РОЗВИТКУ ПЕРСОНАЛУ, ЯК ЧИННИК КОНКУРЕНТОЗДАТНОСТІ ПІДПРИЄМСТВА В РИНКОВИХ УМОВАХ	445
Каплун В.М. УПРАВЛІННЯ СІЛЬСЬКОГОСПОДАРСЬКИХ ПІДПРИЄМСТВ ІЗ ЗАСТОСУВАННЯ ПРОГРАМНОГО КОНТРОЛЮ І МОНІТОРИНГУ	446
Бондар А.В. ТЕОРЕТИКО-МЕТОДИЧНІ ПІДХОДИ ДО ВПРОВАДЖЕННЯ ТА ОЦІНКИ СТРАТЕГІЧНОГО УПРАВЛІННЯ РОЗВИТКОМ СІЛЬСЬКОГОСПОДАРСЬКИХ ПІДПРИЄМСТВ	447
Романенко Т.А. СУЧАСНІ НАПРЯМИ ІНТЕНСИФІКАЦІЇ ВИРОБНИЦТВА ПРОДУКЦІЇ РОСЛИННИЦТВА В	

ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ОРГАНІЗАЦІЇ ДИСТАНЦІЙНИХ КУРСІВ НЕФОРМАЛЬНОЇ ОСВІТИ

Лапченко І. В., студ. 4 курсу ФЕІМ, спец. «Інформаційні системи та технології»
Науковий керівник: к.е.н., доц. Я.В. Долгіх
Сумський НАУ

Проектування та реалізація вебзастосунку організації дистанційних курсів неформальної освіти представляє собою складне та багатогранне завдання, яке вимагає комплексного підходу та системної роботи з різними аспектами. У світлі сучасних тенденцій у розвитку освіти, де дистанційне навчання здобуває все більшу популярність, створення ефективного вебзастосунку для надання курсів неформальної освіти стає важливим завданням для багатьох освітніх установ та організацій.

Для того, щоб успішно реалізувати проект, необхідно врахувати ряд ключових аспектів. Перш за все, необхідно провести детальний аналіз цільової аудиторії та їхніх потреб у неформальній освіті. Це допоможе визначити ключові теми, формати навчання та інструменти, які найбільше цікавлять майбутніх учасників курсів. Важливо також звернути увагу на технічні можливості та особливості цільової аудиторії, щоб забезпечити їм зручний та доступний інтерфейс.

Далі, на основі проведеного аналізу, необхідно розробити вебзастосунок, який забезпечить зручний та ефективний доступ до навчальних матеріалів. При цьому важливо врахувати простоту інтерфейсу та можливість інтерактивного навчання, що зробить процес навчання цікавим та зрозумілим для користувачів.

Окрему увагу слід приділити створенню якісного навчального контенту. Це включає в себе підготовку відеоуроків, текстових матеріалів, тестів та завдань для студентів. Важливо, щоб контент був цікавим, актуальним та зрозумілим для аудиторії.

Після запуску платформи важливо забезпечити постійну технічну підтримку для користувачів та здійснювати постійне оновлення та модернізацію функціоналу з урахуванням змін у потребах аудиторії та розвитку технологій. Крім того, ефективне просування платформи є важливим етапом у залученні аудиторії та популяризації курсів.

Не менш важливим етапом є систематична оцінка ефективності платформи шляхом опитування користувачів та аналізу даних про успішність навчання. Це допоможе вчасно виявляти проблеми та вдосконалювати процес навчання для досягнення найкращих результатів.

У загальному, успішність проекту залежить від здатності команди розробників врахувати потреби аудиторії та забезпечити високу якість навчального контенту та технічного забезпечення платформи. Тільки завдяки комплексному підходу та системній роботі з різними аспектами проекту можна досягти успіху та забезпечити якісну та доступну освіту для всіх бажаючих.

Крім того, з урахуванням різноманітності форматів навчання та індивідуальних особливостей студентів, важливо забезпечити гнучкість та можливість персоналізації навчального процесу. Це може включати в себе можливість вибору тем, темпу та послідовності вивчення матеріалу, а також можливість взаємодії з викладачами та іншими студентами через форуми, чати та онлайн-конференції.

Необхідно також звернути увагу на забезпечення високої якості та достовірності навчального контенту. Це включає в себе перевірку інформації та джерел, які використовуються в курсах, а також забезпечення актуальності та відповідності матеріалів сучасним стандартам та тенденціям у відповідній галузі.

Окрім того, важливо розглядати можливості інтеграції з іншими освітніми платформами та сервісами, що може сприяти підвищенню ефективності та доступності освітніх ресурсів для студентів. Також важливо створити механізми для оцінювання успішності та досягнень студентів, які б дозволили якісно визначати їхній прогрес та допомагали виявляти слабкі місця для подальшого покращення.

Загалом, успішне проектування та реалізація веб-застосунку для організації дистанційних курсів неформальної освіти вимагає комплексного підходу, системної роботи з різними аспектами та постійного вдосконалення на основі зворотного зв'язку від користувачів та аналізу даних. Тільки таким чином можна забезпечити високу якість навчання та задоволення потреб аудиторії в навчальних ресурсах та можливостях саморозвитку.

ДОДАТОК Г

ЛІСТІНГИ ПРОГРАМ

Лістинг основних файлів керування роботою додатку:

app.py:

```
from flask import Flask
from flask_login import LoginManager
```

```
app = Flask(__name__)
```

```
login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = "login"
```

```
import views
```

```
@login_manager.user_loader
def load_user(user_id):
    from controllers import get_user_by_id
    return get_user_by_id(user_id)
```

```
app.secret_key = "sldgw;aru8tuj20349haiufQ7F"
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

models.py:

```

import os
from flask_login import UserMixin
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker, relationship
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy import Column, Integer, String, ForeignKey

print(os.path.abspath(os.getcwd()))
database = create_engine(f'sqlite:/// {os.path.abspath(os.getcwd())}/app.db',
connect_args={'check_same_thread': False})
Session = sessionmaker(bind=database)
session = Session()
Base = declarative_base()

class User(UserMixin, Base):
    __tablename__ = 'users'

    id = Column(Integer, primary_key=True)
    name = Column(String)
    username = Column(String(100), unique=True)
    status = Column(String(10)) # teacher or student
    password = Column(String(100))

class Exam(Base):
    __tablename__ = 'exams'

    id = Column(Integer, primary_key=True)
    name = Column(String)

```

```
questions = relationship('Question', back_populates='exam')
```

```
class Question(Base):
```

```
    __tablename__ = 'questions'
```

```
    id = Column(Integer, primary_key=True)
```

```
    exam_id = Column(Integer, ForeignKey('exams.id'))
```

```
    question = Column(String(256))
```

```
    answer1 = Column(String(256))
```

```
    answer2 = Column(String(256))
```

```
    answer3 = Column(String(256))
```

```
    correct = Column(String(256))
```

```
    exam = relationship('Exam', back_populates='questions')
```

```
if __name__ == "__main__":
```

```
    Base.metadata.create_all(database)
```

controllers.py:

```
from flask import flash
```

```
from models import session, Question, Exam, User
```

```
def create_user(name, username, status, password):
```

```
    try:
```

```
        new_user = User(name=name, username=username, status=status,  
password=password)
```

```
        session.add(new_user)
```

```
        session.commit()
```

```
        flash('User created successfully', 'success')
```

```
    except Exception as e:
```

```
        session.rollback()
```

```
flash(f'Error creating user: {str(e)}', 'error')
```

```
def create_exam(name):
```

```
    try:
```

```
        new_exam = Exam(name=name)
```

```
        session.add(new_exam)
```

```
        session.commit()
```

```
        flash('Exam created successfully', 'success')
```

```
    except Exception as e:
```

```
        session.rollback()
```

```
        flash(f'Error creating exam: {str(e)}', 'error')
```

```
def create_question(exam_id, question, answer1, answer2, answer3, correct):
```

```
    try:
```

```
        new_question = Question(exam_id=exam_id, question=question,  
answer1=answer1, answer2=answer2, answer3=answer3, correct=correct)
```

```
        session.add(new_question)
```

```
        session.commit()
```

```
        flash('Question created successfully', 'success')
```

```
    except Exception as e:
```

```
        session.rollback()
```

```
        flash(f'Error creating question: {str(e)}', 'error')
```

```
def get_all_exam_names():
```

```
    try:
```

```
        exam_names = [exam.name for exam in session.query(Exam).all()]
```

```
        return exam_names
```

```
    except Exception as e:
```

```
        flash(f'Error retrieving exam names: {str(e)}', 'error')
```

```
        return []
```

```

def get_questions_for_exam(exam_id):
    try:
        questions = session.query(Question).filter_by(exam_id=exam_id).all()
        questions_dict = {}

        for question_obj in questions:
            question_text = question_obj.question
            answers = {
                'answer1': question_obj.answer1,
                'answer2': question_obj.answer2,
                'answer3': question_obj.answer3,
            }
            correct_answer = question_obj.correct

            questions_dict[question_text] = {'answers': answers, 'correct_answer':
correct_answer}

        return questions_dict
    except Exception as e:
        flash(f'Error retrieving questions for exam: {str(e)}', 'error')
        return []

def get_user_by_id(user_id):
    try:
        user = session.query(User).get(user_id)
        return user
    except Exception as e:
        flash(f'Error retrieving user: {str(e)}', 'error')
        return None

```

```

def count_correct_answers(user_answers: dict) -> tuple:
    correct_answers = session.query(Question.correct).filter_by(exam_id=user_answers["exam_id"]).all()
    number_of_correct_answers = 0
    if len(user_answers) < len(correct_answers):
        return False, 0
    for i in correct_answers:
        if i in user_answers.values:
            number_of_correct_answers += 1

    if number_of_correct_answers / len(correct_answers) > 0.75:
        return True, number_of_correct_answers
    return False, number_of_correct_answers

def get_all_exams():
    return session.query(Exam).all()

```

views.py:

```

from app import app
from random import randint
from models import User, session
from werkzeug.security import check_password_hash
from flask import render_template, request, redirect, url_for
from flask_login import login_user, login_required, logout_user
from controllers import count_correct_answers, create_question, get_all_exam_names,
get_all_exams, get_questions_for_exam

```

```

@app.route('/')

```

```
def home():
    return render_template('home.html', exams=get_all_exam_names())

@app.route('/exam/<int:exam_id>')
def exam(exam_id):
    exam_questions = get_questions_for_exam(exam_id)
    if exam_questions:
        [test.insert(randint(1, 3), test.pop(3)) for test in exam_questions]
        return render_template('exam.html', exam_questions=exam_questions)
    return redirect(url_for('home'))
```

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        user = session.query(User).filter_by(username=username).first()
        if user and check_password_hash(user.password, password):
            login_user(user)
            return redirect(url_for('home'))
    return render_template('login.html')
```

```
@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('home'))
```

```
@app.route('/result')
@login_required
def result():
    is_passed, correct_answers = count_correct_answers(request.form)
    return render_template("result.html", correct_answers=correct_answers,
passed=is_passed)
```

```
@app.route('/create_exam', methods=['GET', 'POST'])
@login_required
def create_exam():
    if request.method == 'POST':
        create_exam(request.form['name'])
        create_question(exam_id=request.form['exam_id'],
question=request.form['question'], answer1=request.form['answer1'],
                answer2=request.form['answer2'], answer3=request.form['answer3'],
correct=request.form['correct'])
        return redirect(url_for('home'))
    return render_template('create.html', exams=get_all_exams())
```

base.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Екзамен</title>
  <link rel="stylesheet" href="../../static/styles.css">
</head>
<body>
<ul class="nav">
  <li><a href="/">Головна сторінка</a></li>
  <li><a href="/create_exam">Додати новий екзамен</a></li>
  {% if current_user.is_authenticated %}
    <li><a href="/logout">Вийти</a></li>
  {% else %}
    <li><a href="/login">Увійти</a></li>
  {% endif %}

</ul>
{% block content %}
{% endblock %}
</body>
</html>
```

login.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Екзамен</title>
  <link rel="stylesheet" href="../../static/styles.css">
</head>
<body>
<ul class="nav">
  <li><a href="/">Головна сторінка</a></li>
  <li><a href="/create_exam">Додати новий екзамен</a></li>
  {% if current_user.is_authenticated %}
    <li><a href="/logout">Вийти</a></li>
  {% else %}
    <li><a href="/login">Увійти</a></li>
  {% endif %}

</ul>
```

```
{% block content %}
{% endblock %}
</body>
</html>
```

home.html:

```
{% extends "base.html" %}

{% block content %}

<section>
  <h2>Доступні Екзамени</h2>
  {% for exam in exams %}
    <div class="exam-card">
      <h3>{{ exam }}</h3>
      <a href="{{ url_for('exam', exam_id=exam.id) }}">Почати тест</a>
    </div>
  {% endfor %}
</section>

{% endblock %}
```

exam.html:

```
{% extends "base.html" %}

{% block content %}

<form action="{{ url_for('check-test') }}" method="post">
  {% for single_test in all_data %}
    <br><span>{{ single_test[0] }}</span><br>
    {% for j in single_test[1:] %}
      <input type="radio" name="{{ single_test[0] }}" id="{{ j }}" value="{{ j }}" >
      <label for="{{ j }}" name="{{ j }}">{{ j }}</label><br>
    {% endfor %}
  {% endfor %}
  <input type="submit" value="Submit">
</form>

{% endblock %}
```

values.yaml:

```
replicaCount: 1
```

service:
 type: ClusterIP
 port: 80

livenessProbe:
 httpGet:
 path: /
 port: http

readinessProbe:
 httpGet:
 path: /
 port: http