

**МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ**

**ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І
СИСТЕМ**

КВАЛІФІКАЦІЙНА РОБОТА

на тему: ІНСТРУМЕНТ ДЛЯ ШВИДКОГО СТВОРЕННЯ ТЕСТІВ

Здобувача вищої освіти другого (магістерського) рівня,
II курсу, групи КМІз-24-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології в економіці»



(підпис)

Любенка Андрія Сергійовича
(прізвище, ім'я, по батькові)

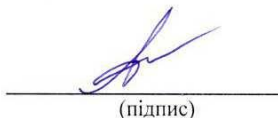
Науковий керівник



(підпис)

Ратушняк Тетяна Володимирівна,
канд. фіз.-мат. наук, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Керівник освітньо-
професійної програми



(підпис)

Омельчук Антон
Анатолійович, канд. техн. наук
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту за рішенням кафедри протокол від «09» січня 2026 № 11

Завідувач кафедри



(підпис)

Філоненко Михайло Миколайович,
канд. фіз.-мат. наук, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Ірпінь – 2026

Реєстрація № 38

02.02.2016

(дата)



(підпис відповідальної особи)

Результати перевірки

до захисту
(до захисту, на
доопрацювання)

09.01.2016

(дата)

Ін. Раєв
(підпис)

Результат захисту

4 (добре)
(оцінка)

11.02.2016

(дата)


(підпис)

Голова комісії:


(підпис)

О.С. Дюбанов

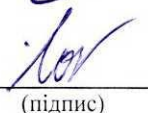
Члени комісії:


(підпис)

А.Ю. Горбовий


(підпис)

М.М. Філоненко


(підпис)

І.Д. Погореловська

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст кваліфікаційної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	22		11.02.2026 Дюбанов О.С.
2. Дотримання вимог щодо оформлення	від 0 до 10	7		Горбовий А.Ю.
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	4		Філоненко М.М. Погореловська І.Д.
4. Рецензент	від 0 до 5	4		Суховерхня Л.П.
РАЗОМ БАЛІВ	від 0 до 50	37		

Оцінювання кваліфікаційної роботи науковим керівником - 36 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підписи)
73	C	4 (добре)	11.02.2026	Дюбанов О.С. Горбовий А.Ю. Філоненко М.М. Погореловська І.Д. Суховерхня Л.П.

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	
50-59	E	3 (задовільно)
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань 12 «Інформаційні технології»
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Інформаційні управляючі системи і технології в економіці»

Затверджую

Завідувач кафедри  Філоненко М.М.

«11» грудня 2024

Завдання на кваліфікаційну роботу

здобувачу вищої освіти

Любенку Андрію Сергійовичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Інструмент для швидкого створення тестів
Затверджена наказом ректора від «10» грудня 2024 № 1852
2. Строк здачі закінченої роботи «09» січня 2026
3. Мета роботи Розробка веб-додатку для швидкого створення, редагування, управління та експорту тестів у різних форматах, який забезпечує зручний інтерфейс та гнучкість у виборі типів питань.
Об'єкт дослідження Веб-додатки для створення та управління тестами, їх архітектура, функціональність та технологічні рішення.
Предмет дослідження Методи та технології розробки веб-додатків для швидкого створення тестів з підтримкою різних форматів експорту, включаючи особливості проектування інтерфейсів користувача, організації зберігання даних та генерації експортованих матеріалів.
Орієнтовна база дослідження Дослідження зосереджується на методах інтеграції різних технологій для створення комплексного рішення, яке забезпечує ефективне створення та управління тестами.
4. Вихідні дані до роботи Традиційні методи створення тестів, які передбачають ручне форматування документів у текстових редакторах, займають значний час та не забезпечують необхідну гнучкість у виборі типів питань та форматів представлення. Розвиток дистанційної освіти та змішаних форматів навчання створив нові вимоги до способів створення та розповсюдження тестових матеріалів.
5. Перелік графічного матеріалу Інтерфейс інформаційної системи для швидкого створення тестів;
6. Зміст кваліфікаційної роботи (перелік питань за кожним розділом)
ВСТУП; РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ; РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ; РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ; РОЗДІЛ 4. ТЕСТУВАННЯ ТА ОПТИМІЗАЦІЯ; РОЗДІЛ 5. РЕЗУЛЬТАТИ ТА ЇХ ОБГОВОРЕННЯ; ВИСНОВКИ; СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ.

7. Календар план виконання роботи:

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 15.12.2024	До 15.12.2024
2	I розділ (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	26.05.2025	26.05.2025
3	II розділ (аналіз експериментальної частини про методики досліджень)	02.09.2025	02.09.2025
4	III розділ (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	24.11.2025	24.11.2025
5	Висновки	18.12.2025	18.12.2025
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	02.01.2025	02.01.2025
7	Отримання відгуку керівника	02.02.2026	02.02.2026
8	Отримання зовнішньої рецензії	05.02.2026	05.02.2026

8. Дата видачі завдання «11» грудня 2024

9. Керівник кваліфікаційної роботи


(підпис)

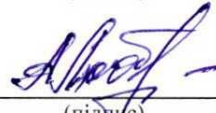
Ратушняк Т.В.
(прізвище, ініціали)

10. Керівник освітньо-професійної програми


(підпис)

Омельчук А.А.
(прізвище, ініціали)

11. Завдання до виконання отримав
здобувач вищої освіти


(підпис)

Любенко А.С.
(прізвище, ініціали)

АНОТАЦІЯ

ІНСТРУМЕНТ ДЛЯ ШВИДКОГО СТВОРЕННЯ ТЕСТІВ

В результаті проведеного дослідження та розробки створено повнофункціональний веб-додаток для швидкого створення, редагування, управління та експорту тестів у різних форматах. Система реалізована на базі сучасних веб-технологій, включаючи Flask як веб-фреймворк для серверної частини та HTML, CSS та JavaScript для клієнтської частини.

Розроблена система забезпечує всі необхідні функції для ефективного створення та управління тестами, включаючи підтримку різних типів питань, експорт у різні формати та можливість проходження тестів. Система має інтуїтивний інтерфейс, який не вимагає навчання, та забезпечує швидке створення тестів з мінімальними зусиллями. Унікальною особливістю системи є можливість експорту тестів у формат PNG з автоматичним генеруванням бланків відповідей для друку, що відсутнє в більшості існуючих рішень. Модульна архітектура системи забезпечує легке розширення функціональності та адаптацію під конкретні потреби користувачів.

Кваліфікаційна робота містить 54 сторінок, 7 рисунків, список використаних джерел 47 найменувань.

Ключові слова: інформаційна система, тестування, дистанційна освіта.

Рік захисту роботи 2026

ABSTRACT

TOOL FOR QUICK CREATION OF TESTS

As a result of the research and development, a fully functional web application was created for quick creation, editing, management and export of tests in various formats. The system is implemented on the basis of modern web technologies, including Flask as a web framework for the server side and HTML, CSS and JavaScript for the client side.

The developed system provides all the necessary functions for effective creation and management of tests, including support for various types of questions, export to various formats and the ability to pass tests. The system has an intuitive interface that does not require training and provides quick creation of tests with minimal effort. A unique feature of the system is the ability to export tests to PNG format with automatic generation of answer sheets for printing, which is absent in most existing solutions. The modular architecture of the system provides easy expansion of functionality and adaptation to specific user needs.

The qualification work contains 54 pages, 7 figures, a list of used sources of 47 items.

Keywords: information system, testing, distance education.

The year of the work is 2026

ЗМІСТ

ВСТУП.....	3
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ.....	7
РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ	15
РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ.....	24
РОЗДІЛ 4. ТЕСТУВАННЯ ТА ОПТИМІЗАЦІЯ.....	33
РОЗДІЛ 5. РЕЗУЛЬТАТИ ТА ЇХ ОБГОВОРЕННЯ	40
ВИСНОВКИ	47
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ.....	50

ВСТУП

Сучасна освітня система стикається з необхідністю швидкого та ефективного створення тестових матеріалів для перевірки знань студентів та учнів. Процес створення тестів традиційними методами є трудомістким та часозатратним, що обмежує можливості викладачів у підготовці якісних навчальних матеріалів. Розвиток дистанційної освіти та онлайн-навчання, особливо в умовах сучасних викликів, значно підвищив потребу в автоматизованих інструментах для створення, управління та розповсюдження тестів. Викладачі та освітні установи потребують універсальних рішень, які дозволяють швидко створювати структуровані тести з різними типами питань, експортувати їх у зручних форматах для різних сценаріїв використання, включаючи друк на папері, розповсюдження в електронному вигляді та інтеграцію з іншими освітніми системами. Ефективність автоматизації процесу створення тестів безпосередньо впливає на якість освітнього процесу та дозволяє викладачам зосередитися на педагогічній діяльності замість технічних аспектів підготовки матеріалів.

Актуальність теми дослідження обумовлена кількома ключовими факторами. По-перше, зростання обсягу навчальних програм та необхідність регулярної перевірки знань студентів вимагають від викладачів постійного створення нових тестових матеріалів. Традиційні методи створення тестів, які передбачають ручне форматування документів у текстових редакторах, займають значний час та не забезпечують необхідну гнучкість у виборі типів питань та форматів представлення. По-друге, розвиток дистанційної освіти та змішаних форматів навчання створив нові вимоги до способів створення та розповсюдження тестових матеріалів. Викладачі потребують інструментів, які дозволяють швидко адаптувати тести під різні формати навчання, включаючи синхронне та асинхронне онлайн-навчання, гібридні курси та традиційні аудиторні заняття. По-третє, необхідність забезпечення якості освітнього

процесу вимагає від викладачів створення різноманітних типів питань, які дозволяють комплексно оцінити знання студентів, включаючи закриті питання з множинним вибором, питання типу правда-неправда, питання з заповненням пропусків та розгорнуті питання для оцінки глибини розуміння матеріалу.

Об'єктом дослідження є веб-додатки для створення та управління тестами, їх архітектура, функціональність та технологічні рішення.

Дослідження має охоплювати аналіз існуючих веб-додатків для створення тестів, їх архітектурних рішень, використаних технологій та підходів до реалізації функціональності. Особлива увага приділяється способам організації зберігання даних, механізмам експорту тестів у різні формати та підходам до проектування інтерфейсів користувача.

Предметом дослідження виступають методи та технології розробки веб-додатків для швидкого створення тестів з підтримкою різних форматів експорту, включаючи особливості проектування інтерфейсів користувача, організації зберігання даних та генерації експортованих матеріалів.

Дослідження зосереджується на методах інтеграції різних технологій для створення комплексного рішення, яке забезпечує ефективне створення та управління тестами.

Метою дослідження є розробка веб-додатку для швидкого створення, редагування, управління та експорту тестів у різних форматах, який забезпечує зручний інтерфейс та гнучкість у виборі типів питань.

Система має дозволяти викладачам, тренерам та освітнім установам ефективно створювати професійні тести без необхідності використання складних програм або залежності від зовнішніх онлайн-сервісів. Розроблена система має забезпечувати можливість локального запуску, що дозволяє працювати без інтернет-з'єднання та забезпечує повний контроль над освітніми матеріалами. Ключовою особливістю системи є підтримка експорту тестів у різні формати, включаючи JSON для програмного використання та інтеграції з іншими системами, HTML для перегляду в браузері та електронного

розповсюдження, TXT для простого текстового формату та PNG для друку на папері з автоматичним генеруванням бланків відповідей.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

- проаналізувати існуючі інструменти для створення тестів та визначити їх недоліки, що дозволить сформулювати чітке розуміння проблем, які необхідно вирішити в новій системі;
- визначити вимоги до системи створення тестів з урахуванням потреб цільової аудиторії, включаючи функціональні та нефункціональні вимоги;
- спроектувати архітектуру веб-додатку, що забезпечує модульність та можливість розширення, використовуючи сучасні архітектурні підходи та паттерни проектування;
- реалізувати backend компоненти на базі сучасних веб-технологій, забезпечивши надійну обробку даних та валідацію вхідної інформації;
- реалізувати frontend компоненти з інтуїтивним інтерфейсом користувача, який не вимагає навчання та доступний для користувачів з різним рівнем технічної підготовки;
- реалізувати функції експорту у формати JSON, HTML, TXT та PNG для різних сценаріїв використання, забезпечивши високу якість експортованих матеріалів;
- реалізувати функцію імпорту тестів для забезпечення сумісності з іншими системами та можливості міграції даних;
- протестувати систему та оптимізувати продуктивність, переконавшись у коректній роботі всіх компонентів;
- оцінити результати та визначити напрямки подальшого розвитку системи.

Для вирішення поставлених задач використовуються такі методи дослідження:

- аналіз наукової літератури з питань розробки веб-додатків та систем управління освітнім контентом, що дозволяє використати накопичений досвід та найкращі практики в галузі;
- порівняльний аналіз існуючих рішень для виявлення їх переваг та недоліків, що формує основу для проектування нової системи;
- проектування та моделювання системи з використанням сучасних архітектурних підходів, включаючи паттерни проектування та принципи модульності; програмна реалізація на базі обраних технологій, що дозволяє перевірити практичну реалізованість запропонованих рішень;
- тестування та експериментування для перевірки функціональності та продуктивності системи, забезпечуючи надійність та якість розробленого рішення.

Наукова новизна дослідження полягає в розробці комплексного підходу до створення тестів з підтримкою різних форматів експорту, включаючи унікальну можливість експорту у формат PNG з автоматичним генеруванням бланків відповідей для друку. Більшість існуючих систем не підтримують експорт у формат зображення для друку, що обмежує їх застосування в сценаріях, де необхідний друк тестів на папері. Розроблена система забезпечує автоматичне генерування високоякісних зображень у форматі A4 з роздільною здатністю 300 DPI, що забезпечує чіткий друк на стандартних принтерах. Додатково система генерує бланки відповідей з квадратами для відмітки правильних варіантів, що спрощує процес перевірки тестів після їх проходження студентами.

Практична значущість роботи визначається можливістю практичного застосування розробленої системи викладачами, освітніми установами та тренерами для швидкого створення професійних тестів без необхідності використання складних програм або залежності від зовнішніх сервісів. Модульна архітектура системи забезпечує легке розширення функціональності та адаптацію під конкретні потреби користувачів, що робить систему гнучким інструментом для різних сценаріїв використання.

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

Аналіз потреб у системах створення тестів показує, що викладачі та освітні установи потребують інструментів, які дозволяють швидко створювати тести з мінімальними зусиллями. Вимоги до швидкості створення тестів є критичними, оскільки викладачі часто мають обмежений час для підготовки навчальних матеріалів. Дослідження показують, що середній час створення тесту з десятима питаннями традиційними методами становить приблизно п'ятнадцять-двадцять хвилин, тоді як автоматизовані інструменти можуть зменшити цей час до п'яти-семи хвилин. Потреба в різноманітних форматах експорту обумовлена різними сценаріями використання: електронне розповсюдження через електронну пошту або системи управління навчанням, друк на папері для проведення тестування в аудиторіях без комп'ютерів, інтеграція з іншими системами через стандартні формати обміну даними, архівування та довгострокове зберігання навчальних матеріалів. Необхідність простоти використання є ключовою, оскільки багато викладачів не мають глибоких технічних знань та потребують інтуїтивних інтерфейсів, які не вимагають навчання. Дослідження користувацького досвіду показують, що складні інтерфейси з великою кількістю налаштувань часто відлякують користувачів та зменшують ефективність використання системи.

Огляд існуючих інструментів для створення тестів включає такі популярні рішення як Google Forms, Kahoot, Quizlet, Moodle, Typeform та SurveyMonkey.

Google Forms забезпечує простий інтерфейс для створення опитувань та тестів з базовою функціональністю, включаючи можливість додавання різних типів питань, налаштування правильних відповідей та автоматичну перевірку результатів. Однак система має обмежені можливості експорту, підтримуючи тільки експорт у формат Google Sheets, що ускладнює використання тестів поза екосистемою Google. Додатково система має залежність від інтернет-з'єднання та вимагає облікового запису Google для використання, що може бути проблемою для деяких користувачів.

Kahoot орієнтований на інтерактивні ігрові тести з реалізацією в реальному часі, що робить його ідеальним для групових занять та змагальних форматів навчання. Однак система обмежена в можливостях створення традиційних тестів для індивідуальної перевірки знань та не підтримує експорт тестів у формати, придатні для друку або архівування.

Quizlet фокусується на картках для навчання та має обмежені можливості створення структурованих тестів з різними типами питань. Система орієнтована на заучування та повторення матеріалу, а не на формальну перевірку знань, що обмежує її застосування в академічному контексті.

Moodle є повнофункціональною системою управління навчанням з потужними можливостями створення тестів, включаючи підтримку різних типів питань, налаштування параметрів тестування, автоматичну перевірку та детальну статистику результатів. Однак система вимагає складного налаштування та серверної інфраструктури, що обмежує її доступність для малих освітніх установ та індивідуальних викладачів. Встановлення та налаштування Moodle потребує технічних знань та додаткових ресурсів, що робить систему недоступною для багатьох потенційних користувачів. Typeform та SurveyMonkey орієнтовані на створення опитувань та мають обмежену функціональність для академічного тестування. Ці системи забезпечують привабливі інтерфейси та зручні інструменти для створення опитувань, але не підтримують багато функцій, необхідних для створення навчальних тестів, таких як детальна настройка правильних відповідей, різні системи оцінювання та експорт у формати, придатні для друку.

Порівняльний аналіз подібних веб-додатків показує, що більшість існуючих рішень мають значні обмеження. За показником простоти використання Google Forms та Quizlet демонструють високу оцінку, оскільки їх інтерфейси інтуїтивні та не вимагають навчання. Kahoot має середню оцінку через складність налаштування та необхідність розуміння специфіки ігрового формату тестування. Moodle вимагає значної технічної підготовки та має

складний інтерфейс з великою кількістю налаштувань, що робить його важким для початківців. Жоден з аналізованих інструментів не підтримує експорт у формат PNG для друку, що є критичним недоліком для сценаріїв, де необхідний друк тестів на папері. Всі розглянуті рішення не підтримують локальне використання без інтернет-з'єднання, що обмежує їх застосування в умовах нестабільного інтернету або для забезпечення конфіденційності даних. Підтримка різних типів питань присутня в більшості систем, але рівень гнучкості варіюється. Наприклад, Google Forms підтримує основні типи питань, але має обмеження в налаштуванні складних сценаріїв тестування. Moodle забезпечує найширшу підтримку різних типів питань, але це супроводжується складністю налаштування. Можливості імпорту та експорту у формат JSON обмежені або відсутні в більшості рішень, що ускладнює інтеграцію з іншими системами та міграцію даних між різними платформами.

Критична оцінка переваг та недоліків існуючих рішень виявляє наступні основні проблеми. Залежність від інтернет-з'єднання є критичним недоліком для багатьох онлайн-сервісів, оскільки обмежує можливості використання в умовах нестабільного інтернету або для забезпечення конфіденційності освітніх матеріалів. Багато освітніх установ мають обмежений або нестабільний доступ до інтернету, особливо в сільській місцевості або в умовах технічних обмежень. Додатково, деякі освітні матеріали містять конфіденційну інформацію, яку небажано зберігати на зовнішніх серверах, що робить локальне зберігання критично важливим. Обмежені формати експорту не дозволяють гнучко адаптувати тести під різні потреби, включаючи друк на папері, електронне розповсюдження та інтеграцію з іншими системами. Більшість систем підтримують тільки експорт у формати, специфічні для їх платформи, що ускладнює використання тестів в інших контекстах. Складність налаштування багатьох систем, особливо таких як Moodle, вимагає значної технічної підготовки та серверної інфраструктури, що обмежує доступність для малих освітніх установ. Встановлення та налаштування таких систем потребує знань

адміністрування серверів, налаштування баз даних та конфігурації веб-серверів, що недоступно для багатьох викладачів.

Відсутність локального запуску означає, що користувачі не можуть працювати з системою без інтернет-з'єднання та повністю залежать від зовнішніх сервісів. Це створює ризики втрати доступу до даних у разі проблем з сервісом, змін у політиці провайдера або технічних збоїв. Обмежена кастомізація не дозволяє адаптувати інтерфейс та функціональність під конкретні потреби користувачів. Багато систем мають фіксований набір функцій та обмежені можливості налаштування, що не дозволяє адаптувати їх під специфічні вимоги різних освітніх установ. Потреба в реєстрації для багатьох сервісів створює додаткові бар'єри для використання та може викликати проблеми з конфіденційністю даних. Процес реєстрації часто включає надання персональних даних та прийняття умов використання, що може бути проблематичним для деяких користувачів. Обмежена підтримка української мови є серйозним недоліком для українських освітніх установ, оскільки ускладнює використання системи викладачами та студентами. Багато систем мають інтерфейси тільки англійською мовою або обмежену підтримку інших мов, що створює бар'єри для користувачів, які не володіють англійською мовою на достатньому рівні.

Переваги існуючих рішень включають хмарне зберігання даних, що забезпечує доступність з будь-якого пристрою та автоматичне резервне копіювання. Це дозволяє користувачам працювати з тестами з різних пристроїв та не турбуватися про втрату даних у разі поломки локального пристрою. Спільний доступ до тестів дозволяє командам викладачів працювати разом над створенням матеріалів, що підвищує ефективність роботи та забезпечує консистентність навчальних матеріалів. Статистика та аналітика результатів проходження тестів допомагає викладачам аналізувати ефективність навчального процесу, виявляти проблемні теми та адаптувати методи навчання. Інтеграція з іншими сервісами розширює можливості використання тестів у

різних контекстах, дозволяючи вбудовувати тести в інші навчальні платформи та системи управління. Однак ці переваги часто супроводжуються значними обмеженнями, які обмежують практичне застосування систем у багатьох сценаріях.

Визначення вимог до системи створення тестів базується на аналізі потреб користувачів та виявлених недоліків існуючих рішень. Система має забезпечувати швидке створення тестів з мінімальною кількістю кроків, що дозволяє викладачам ефективно використовувати свій час. Інтерфейс має бути спроектований таким чином, щоб створення тесту з десятима питаннями займало не більше п'яти-семи хвилин, включаючи введення тексту питань, додавання варіантів відповідей та встановлення правильних відповідей. Підтримка різних типів питань, включаючи множинний вибір, правда-неправда, заповнення пропусків та розгорнуті відповіді, забезпечує гнучкість у створенні тестів для різних предметів та рівнів складності. Кожен тип питання має свої особливості та вимоги до форматування, тому система має забезпечувати зручні інструменти для створення кожного типу. Експорт у різні формати, включаючи JSON для програмного використання, HTML для перегляду в браузері, TXT для простого текстового формату та PNG для друку, забезпечує універсальність системи. Кожен формат має свої переваги та призначений для конкретних сценаріїв використання.

Простий інтуїтивний інтерфейс, який не вимагає навчання, робить систему доступною для користувачів з різним рівнем технічної підготовки. Інтерфейс має бути спроектований з урахуванням принципів користувацького досвіду, включаючи чітку навігацію, зрозумілі підказки та мінімальну кількість кроків для виконання основних операцій. Локальний запуск дозволяє використовувати систему без інтернет-з'єднання та забезпечує повний контроль над даними. Це особливо важливо для забезпечення конфіденційності освітніх матеріалів та роботи в умовах нестабільного інтернету. Відсутність необхідності реєстрації спрощує початок роботи з системою та усуває бар'єри для використання.

Користувачі можуть одразу почати створювати тести без необхідності проходження процесу реєстрації та надання персональних даних.

Аналіз цільової аудиторії показує, що система призначена для широкого кола користувачів. Викладачі та вчителі є основною цільовою аудиторією, оскільки вони регулярно створюють тести для перевірки знань студентів та учнів. Ця група користувачів потребує швидких та зручних інструментів, які дозволяють ефективно створювати тести без відволікання від основної педагогічної діяльності. Тренери та інструктори використовують тести для оцінки результатів навчання в корпоративному середовищі та на курсах підвищення кваліфікації. Ця група часто потребує можливості швидко створювати тести для різних навчальних програм та адаптувати їх під конкретні потреби організацій. Освітні установи потребують централізованих інструментів для створення та управління тестами на рівні установи. Це дозволяє забезпечити консистентність навчальних матеріалів та спростити координацію роботи викладачів. Корпоративні навчальні центри використовують тести для оцінки ефективності навчальних програм та сертифікації співробітників. Студенти, які створюють тести для самоперевірки, також можуть користуватися системою для підготовки до іспитів та перевірки власних знань.

Визначення функціональних вимог включає наступні основні можливості системи. Створення тестів з різними типами питань забезпечує гнучкість у підготовці навчальних матеріалів для різних предметів та рівнів складності. Система має підтримувати створення питань типу множинний вибір з можливістю додавання від двох до десяти варіантів відповідей, питань типу правда-неправда для швидкої перевірки базових знань, питань з заповненням пропусків для перевірки знання конкретних термінів або фактів, та розгорнутих питань для оцінки глибини розуміння матеріалу. Редагування та видалення тестів дозволяє користувачам оновлювати матеріали відповідно до змін у навчальних програмах та виправляти помилки. Система має забезпечувати можливість редагування всіх аспектів тесту, включаючи назву, опис, типи питань та

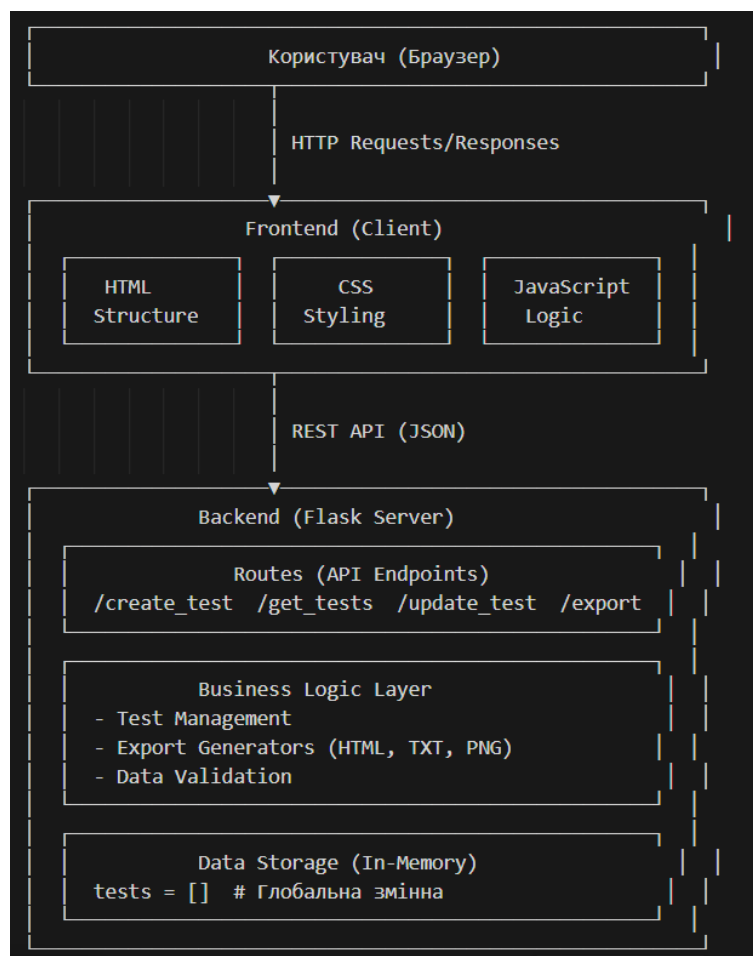
відповіді. Експорт у формати JSON, HTML, TXT та PNG забезпечує універсальність використання тестів у різних сценаріях. JSON формат дозволяє програмно обробляти тести та інтегрувати їх з іншими системами, HTML формат забезпечує зручний перегляд у браузері, TXT формат забезпечує універсальну сумісність, а PNG формат дозволяє друкувати тести на папері. Імпорт з JSON дозволяє мігрувати тести з інших систем та обмінюватися матеріалами між користувачами. Попередній перегляд тестів дозволяє користувачам перевірити зовнішній вигляд та структуру перед збереженням. Проходження тестів забезпечує можливість використання системи не тільки для створення, але й для безпосереднього тестування. Збереження результатів дозволяє відстежувати успішність проходження тестів та аналізувати ефективність навчального процесу.

Визначення нефункціональних вимог включає вимоги до продуктивності, зручності використання та технічних характеристик системи. Швидкість відгуку системи має бути менше однієї секунди для більшості операцій, що забезпечує комфортну роботу користувачів без затримок. Операції створення та редагування тестів мають виконуватися миттєво, а операції експорту можуть займати більше часу залежно від розміру тесту та формату експорту, але не повинні перевищувати кілька секунд для стандартних тестів. Простота інтерфейсу є критичною вимогою, оскільки система призначена для користувачів з різним рівнем технічної підготовки. Інтерфейс має бути інтуїтивним та не вимагати навчання для виконання основних операцій. Підтримка української мови забезпечує доступність системи для українських освітніх установ та сприяє зручності використання. Кросплатформенність дозволяє використовувати систему на різних операційних системах та пристроях, включаючи Windows, Linux та macOS. Легкість встановлення спрощує початок роботи з системою та зменшує технічні бар'єри для користувачів. Система має встановлюватися за допомогою простих команд та не вимагати складного налаштування.

В результаті проведеного аналізу визначено потреби та вимоги до системи створення тестів, виявлено недоліки існуючих рішень, які обмежують їх практичне застосування, сформульовано вимоги до нової системи, які враховують потреби цільової аудиторії та усувають виявлені недоліки існуючих рішень. Проведений аналіз обґрунтовує необхідність розробки нової системи, яка поєднує простоту використання існуючих онлайн-сервісів з можливістю локального запуску та розширеними можливостями експорту, включаючи унікальну підтримку формату PNG для друку. Визначені вимоги до системи формують основу для подальшого проектування та реалізації веб-додатку, який забезпечить ефективне рішення проблеми швидкого створення тестів для освітніх потреб. Розроблена система має забезпечити викладачам, тренерам та освітнім установам зручний та ефективний інструмент для створення професійних тестів, який усуває виявлені недоліки існуючих рішень та забезпечує необхідну функціональність для різних сценаріїв використання.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ

Проектування системи для швидкого створення тестів починається з визначення архітектурних рішень, які забезпечать ефективну роботу системи та можливість її подальшого розширення. Архітектура системи базується на класичній моделі клієнт-сервер, де клієнтська частина відповідає за відображення інтерфейсу користувача та взаємодію з користувачем, а серверна частина обробляє бізнес-логіку та управління даними. Така архітектура забезпечує розділення відповідальностей між компонентами системи та дозволяє незалежно розвивати клієнтську та серверну частини.



Загальна архітектура веб-додатку включає три основні рівні: рівень представлення, який реалізований за допомогою HTML, CSS та JavaScript у

браузері користувача, рівень бізнес-логіки, який реалізований на сервері за допомогою Flask фреймворку, та рівень зберігання даних, який на початковому етапі реалізований через зберігання в пам'яті сервера.

Архітектура клієнт-сервер забезпечує ефективну взаємодію між компонентами системи. Клієнтська частина, реалізована у браузері користувача, відповідає за відображення інтерфейсу, обробку дій користувача та відправку запитів на сервер. Використання сучасних веб-технологій HTML5, CSS3 та JavaScript ES6+ забезпечує швидку роботу інтерфейсу та зручну взаємодію з користувачем. Серверна частина, реалізована на Flask веб-сервері, обробляє запити від клієнта, виконує бізнес-логіку, валідує дані та повертає відповіді у форматі JSON. Використання протоколу HTTP/HTTPS забезпечує стандартну та надійну комунікацію між клієнтом та сервером. Формат обміну даними JSON є універсальним та легко обробляється як на сервері, так і на клієнті, що спрощує реалізацію та забезпечує сумісність з різними системами.

Паттерни проектування, використані в системі, забезпечують модульність, підтримуваність та можливість розширення. Паттерн MVC (Model-View-Controller) розділяє систему на три компоненти: Model відповідає за структуру даних та бізнес-логіку, View відповідає за відображення даних користувачу, а Controller відповідає за обробку вхідних даних та координацію між Model та View. У контексті веб-додатку Model представлена структурою даних тестів, яка зберігається в пам'яті сервера, View представлена HTML шаблонами та CSS стилями, які визначають зовнішній вигляд інтерфейсу, а Controller представлена Flask routes та JavaScript кодом, який обробляє запити користувача та координує взаємодію між компонентами. Використання RESTful API забезпечує стандартизований підхід до проектування API, де кожен ресурс має унікальний URL, а операції над ресурсами виконуються через стандартні HTTP методи. Це робить API зрозумілим та легким у використанні, а також забезпечує сумісність з різними клієнтськими додатками. Модульність системи досягається через розділення коду на окремі файли та компоненти, кожен з яких відповідає за

конкретну функціональність. Це спрощує підтримку коду, дозволяє незалежно тестувати окремі компоненти та полегшує додавання нової функціональності.

Модульна структура системи включає кілька основних компонентів. Файл `app.py` містить всю backend логіку, включаючи визначення маршрутів, обробку запитів, валідацію даних та генерацію експортованих файлів. Це центральний компонент серверної частини, який координує роботу всіх backend функцій. Файл `templates/index.html` містить HTML структуру інтерфейсу користувача, включаючи всі вкладки, форми, модальні вікна та інші елементи інтерфейсу. Використання шаблонізатора Jinja2 дозволяє динамічно генерувати HTML контент на основі даних з сервера. Файл `static/css/style.css` містить всі стилі для оформлення інтерфейсу, включаючи градієнтні фони, анімації, адаптивний дизайн та стилі для різних компонентів інтерфейсу. Файл `static/js/app.js` містить всю клієнтську логіку, включаючи обробку подій, взаємодію з API, динамічне оновлення інтерфейсу та реалізацію системи уведомлень та модальних вікон.

Вибір технологій та інструментів розробки є критично важливим для успішної реалізації системи. Обґрунтування вибору мови програмування Python базується на кількох ключових факторах. По-перше, простота синтаксису Python робить код легким для читання та підтримки, що особливо важливо для швидкої розробки та подальшого розвитку системи. По-друге, велика кількість бібліотек та модулів Python забезпечує готові рішення для багатьох завдань, включаючи обробку зображень, роботу з JSON, веб-розробку та багато іншого. По-третє, швидка розробка дозволяє ефективно реалізувати функціональність системи без необхідності написання великої кількості допоміжного коду. По-четверте, підтримка веб-розробки через різні фреймворки робить Python ідеальним вибором для створення веб-додатків. По-п'яте, кросплатформенність Python дозволяє запускати систему на різних операційних системах без необхідності змін у коді.

Вибір веб-фреймворку Flask обумовлений його перевагами для розробки невеликих та середніх веб-додатків. Легкість та мінімалізм Flask означають, що

фреймворк не нав'язує жорстку структуру проекту та дозволяє розробнику самому визначати архітектуру додатку. Це робить Flask ідеальним вибором для проєктів, де потрібна гнучкість та контроль над структурою коду. Гнучкість Flask дозволяє легко додавати нову функціональність та інтегрувати різні бібліотеки та модулі. Добра документація Flask забезпечує швидке навчання та ефективну розробку, а активна спільнота надає підтримку та готові рішення для багатьох завдань. Підтримка RESTful API через Flask дозволяє легко створювати API endpoints для різних операцій, що є критично важливим для забезпечення взаємодії між клієнтською та серверною частинами.

Технології frontend обрані з урахуванням сучасних стандартів веб-розробки та потреб користувачів. HTML5 забезпечує семантичну розмітку, яка робить структуру сторінки зрозумілою та доступною для різних пристроїв та програм. Використання семантичних елементів HTML5 покращує доступність інтерфейсу та спрощує його підтримку. CSS3 забезпечує сучасну стилізацію з градієнтами та анімаціями, що робить інтерфейс візуально привабливим та сучасним. Використання CSS3 дозволяє створювати складні візуальні ефекти без необхідності використання зображень, що покращує продуктивність та швидкість завантаження сторінки. JavaScript ES6+ забезпечує сучасну логіку клієнта з підтримкою нових можливостей мови, таких як стрілкові функції, деструктуризація, проміси та `async/await`, що робить код більш читабельним та підтримуваним.

Бібліотеки та залежності обрані для забезпечення необхідної функціональності системи. Flask 2.3.3 є основним веб-фреймворком, який забезпечує всю серверну функціональність. Pillow 10.0.1 використовується для обробки зображень та генерації PNG експорту, що є критично важливим для реалізації унікальної функції експорту тестів у формат зображення для друку. Werkzeug 2.3.7 надає WSGI утиліти, які необхідні для роботи Flask сервера. Jinja2 3.1.2 є шаблонізатором, який дозволяє динамічно генерувати HTML контент на

основі даних з сервера. Font Awesome використовується для додавання іконок до інтерфейсу, що покращує візуальну привабливість та зрозумілість інтерфейсу.

Інструменти розробки та середовище виконання включають Python 3.7 або вищу версію, яка забезпечує необхідну функціональність та підтримку сучасних можливостей мови. `pip` є менеджером пакетів Python, який дозволяє легко встановлювати та керувати залежностями проєкту. Веб-браузер з підтримкою JavaScript необхідний для роботи клієнтської частини системи. Текстовий редактор або IDE дозволяє ефективно редагувати код та забезпечує підтримку розробки, включаючи підсвітку синтаксису, автодоповнення та інші корисні функції.

Проектування структури даних є критично важливим для забезпечення ефективної роботи системи. Модель даних тесту включає такі поля як унікальний ідентифікатор, назва тесту, опис тесту, тип тесту, дата створення, дата оновлення та масив питань. Кожне поле має своє призначення та обмеження. Ідентифікатор забезпечує унікальність кожного тесту та дозволяє ефективно знаходити та обробляти конкретні тести. Назва тесту є обов'язковим полем, яке ідентифікує тест для користувача. Опис тесту є опціональним полем, яке дозволяє додати додаткову інформацію про тест. Тип тесту визначає загальний тип питань у тесті, хоча окремі питання можуть мати інші типи. Дати створення та оновлення дозволяють відстежувати історію змін тесту. Масив питань містить всі питання тесту разом з їх варіантами відповідей та правильними відповідями.

Модель даних питання включає текст питання, тип питання, масив варіантів відповідей та індекс правильної відповіді. Текст питання є обов'язковим полем, яке містить саме питання, яке буде показано користувачу. Тип питання визначає формат питання та спосіб його відображення та обробки. Масив варіантів відповідей містить всі можливі варіанти відповіді для питань типу множинний вибір або правда-неправда. Для питань типу заповнення пропусків масив містить правильні варіанти відповіді. Для розгорнутих питань масив може містити приклад правильної відповіді. Індекс правильної відповіді

визначає, який варіант є правильним для питань типу множинний вибір або правда-неправда.

Типи питань та їх структура визначають різні способи перевірки знань студентів. Тип `multiple_choice` дозволяє створити питання з кількома варіантами відповіді, з яких тільки один є правильним. Це найпоширеніший тип питання, який дозволяє швидко перевірити знання студентів. Тип `true_false` дозволяє створити питання з двома варіантами відповіді: правда або неправда. Це найпростіший тип питання, який дозволяє швидко перевірити базові знання. Тип `fill_blank` дозволяє створити питання, де студент має вписати відповідь у поле. Це дозволяє перевірити знання конкретних термінів або фактів. Тип `essay` дозволяє створити розгорнуте питання, де студент має надати детальну відповідь. Це дозволяє перевірити глибину розуміння матеріалу та здатність студента формулювати думки.

Формати зберігання та обміну даними визначають способи збереження та передачі даних між компонентами системи. JSON є основним форматом обміну даними між клієнтом та сервером, оскільки він є легким, читабельним та легко обробляється як на сервері, так і на клієнті. Зберігання в пам'яті використовується під час роботи системи для швидкого доступу до даних. Експорт у формати JSON, HTML, TXT та PNG забезпечує різні способи використання тестів у різних сценаріях.

Проектування інтерфейсу користувача базується на принципах UX/UI дизайну, які забезпечують зручність та ефективність використання системи. Простота та інтуїтивність інтерфейсу означають, що користувач має зрозуміти, як використовувати систему, без необхідності навчання або читання документації. Консистентність інтерфейсу забезпечує, що подібні елементи виглядають та поведуться однаково в різних частинах системи. Візуальна привабливість інтерфейсу робить роботу з системою приємною та мотивує користувачів використовувати систему. Зручна навігація забезпечує легкий доступ до всіх функцій системи.

Структура навігації та вкладок організує функціональність системи в логічні групи. Вкладка "Створити тест" містить форму для створення нового тесту, включаючи поля для назви, опису, типу тесту та динамічне додавання питань. Вкладка "Мої тести" відображає список всіх створених тестів з можливістю перегляду, редагування, видалення та експорту. Вкладка "Пройти тест" дозволяє вибрати тест та пройти його, зберігаючи результати. Вкладка "Імпорт" дозволяє завантажити тест з JSON файлу.

Компоненти інтерфейсу включають форми створення та редагування тестів, які забезпечують введення та зміну даних тестів. Модальні вікна використовуються для попереднього перегляду тестів, налаштування параметрів експорту та підтвердження критичних дій, таких як видалення тесту. Система уведомлень інформує користувача про результати операцій, помилки та важливі події. Картки тестів відображають інформацію про кожен тест у зручному форматі з кнопками для різних дій.

Адаптивність та доступність інтерфейсу забезпечують, що система працює на різних пристроях та доступна для різних користувачів. Підтримка різних розмірів екранів забезпечує коректне відображення інтерфейсу на десктопах, планшетах та мобільних пристроях. Зручна навігація з клавіатури забезпечує доступність для користувачів, які не можуть використовувати мишу. Чіткі підказки та повідомлення допомагають користувачам зрозуміти, як використовувати систему та що відбувається під час виконання операцій.

Проектування API та маршрутизації визначає способи взаємодії між клієнтом та сервером. RESTful API endpoints забезпечують стандартизований спосіб доступу до функціональності системи. Endpoint GET / надає головну сторінку системи з інтерфейсом користувача. Endpoint POST /create_test дозволяє створити новий тест, приймаючи дані тесту у форматі JSON та повертаючи ідентифікатор створеного тесту. Endpoint GET /get_tests повертає список всіх тестів у системі. Endpoint GET /get_test/<id> повертає дані конкретного тесту за його ідентифікатором. Endpoint PUT /update_test/<id> дозволяє оновити

існуючий тест, приймаючи оновлені дані у форматі JSON. Endpoint DELETE /delete_test/<id> видаляє тест з системи. Endpoint POST /export_test/<id> генерує експортований файл тесту у вказаному форматі. Endpoint POST /import_test дозволяє імпортувати тест з JSON файлу.

Методи HTTP та їх використання визначають семантику різних операцій. Метод GET використовується для отримання даних без зміни стану системи. Метод POST використовується для створення нових ресурсів, експорту та імпорту, оскільки ці операції можуть мати побічні ефекти. Метод PUT використовується для оновлення існуючих ресурсів, оскільки він є ідемпотентним та передбачає повну заміну ресурсу. Метод DELETE використовується для видалення ресурсів з системи.

Обробка запитів та відповідей включає валідацію вхідних даних для забезпечення коректності даних перед їх обробкою, обробку помилок для надання зрозумілих повідомлень користувачу у разі проблем, та використання JSON формату для відповідей, що забезпечує стандартизований спосіб обміну даними. Валідація вхідних даних включає перевірку обов'язкових полів, перевірку типів даних, перевірку діапазонів значень та перевірку формату даних для забезпечення їх коректності та безпеки.

Проектування функцій експорту визначає способи перетворення тестів у різні формати. Експорт у формат JSON забезпечує повну структуру тесту з можливістю приховування правильних відповідей та метаданих для різних сценаріїв використання. Експорт у формат HTML генерує стилізовану веб-сторінку з підсвіткою правильних відповідей та готову до друку. Експорт у формат TXT забезпечує простий текстовий формат, який легко читається та має універсальну сумісність. Експорт у формат PNG генерує зображення у форматі A4 з роздільною здатністю 300 DPI, квадратики для відповідей та бланком відповідей, що робить його ідеальним для друку на папері.

План створення системи включає п'ять основних етапів. Етап планування та проектування включає визначення вимог до системи, проектування структури даних, вибір технологій та проектування інтерфейсу користувача. Етап backend розробки включає налаштування Flask додатку, створення API endpoints, реалізацію CRUD операцій, розробку функцій експорту та реалізацію імпорту. Етап frontend розробки включає створення HTML структури, розробку CSS стилізації, реалізацію JavaScript логіки, систему уведомлень та модальні вікна. Етап тестування та оптимізації включає тестування створення тестів, тестування редагування та видалення, перевірку експорту, тестування імпорту та оптимізацію продуктивності. Етап документації включає створення документації, опис проєкту та приклади використання. Хронологічний план виконання передбачає два тижні на планування та проектування, три тижні на backend розробку, три тижні на frontend розробку, два тижні на тестування та оптимізацію та один тиждень на документацію.

Висновки до розділу показують, що в результаті проектування визначено архітектуру системи, яка забезпечує модульність та можливість розширення, обрані технології та інструменти, які забезпечують ефективну реалізацію системи, визначено структуру даних, яка забезпечує гнучкість у роботі з тестами та питаннями, та сформовано план реалізації, який визначає послідовність робіт та необхідні ресурси для створення системи.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ

Реалізація системи почалася з налаштування Flask додатку, який є основою серверної частини системи. Flask додаток ініціалізується через створення екземпляра класу Flask з назвою проєкту. Використання модуля `render_template` дозволяє генерувати HTML сторінки на основі шаблонів, модуль `request` забезпечує доступ до даних HTTP запитів, а модуль `jsonify` дозволяє легко повертати JSON відповіді. Глобальна змінна `tests` зберігає всі тести в пам'яті під час роботи сервера, а змінна `next_test_id` забезпечує унікальність ідентифікаторів тестів. Така організація зберігання даних забезпечує швидкий доступ до тестів, але має обмеження у вигляді втрати даних після перезапуску сервера, що є прийнятним для початкової версії системи.

Реалізація маршрутів та обробників забезпечує обробку всіх HTTP запитів від клієнта. Кожен маршрут визначений через декоратор `@app.route()`, який вказує URL та дозволені HTTP методи. Маршрути для CRUD операцій реалізовані відповідно до принципів RESTful API, де кожна операція має свій унікальний URL та HTTP метод. Обробка HTTP запитів включає отримання даних з запиту, їх валідацію, виконання необхідних операцій та повернення відповіді у форматі JSON. Використання JSON формату для відповідей забезпечує стандартизований спосіб обміну даними між клієнтом та сервером та спрощує обробку відповідей на клієнті.

Реалізація CRUD операцій з тестами забезпечує повний функціонал управління тестами. Операція Create реалізована через маршрут `POST /create_test`, який приймає дані тесту у форматі JSON, валідує їх, створює новий тест з унікальним ідентифікатором та додає його до списку тестів. Операція Read реалізована через маршрути `GET /get_tests` для отримання списку всіх тестів та `GET /get_test/<id>` для отримання конкретного тесту за ідентифікатором. Операція Update реалізована через маршрут `PUT /update_test/<id>`, який знаходить тест за ідентифікатором, валідує оновлені дані та оновлює тест у

списку. Операція Delete реалізована через маршрут DELETE /delete_test/<id>, який видаляє тест зі списку за ідентифікатором. Всі операції включають обробку помилок та повернення зрозумілих повідомлень у разі проблем.

Система валідації даних реалізована через функцію `validate_test_data()`, яка перевіряє коректність даних тесту перед їх обробкою. Функція перевіряє наявність обов'язкових полів, таких як назва тесту та питання, перевіряє діапазони значень, такі як максимальна довжина назви тесту, опису та тексту питань, перевіряє типи даних, такі як наявність масиву питань та коректність структури кожного питання, та перевіряє логічну коректність даних, таку як наявність правильної відповіді для питань типу множинний вибір. Валідація питань включає перевірку наявності тексту питання, коректності типу питання, наявності варіантів відповідей для питань типу множинний вибір та наявності правильної відповіді для таких питань. Обробка помилок та винятків забезпечує, що будь-які помилки, які виникають під час обробки запитів, коректно обробляються та повертаються користувачу у зрозумілому форматі. Використання try-ехсепт блоків дозволяє перехопити винятки та повернути відповідні HTTP статус коди та повідомлення про помилки.

Реалізація функцій експорту забезпечує перетворення тестів у різні формати для різних сценаріїв використання. Генерація JSON експорту використовує стандартну бібліотеку Python `json` для серіалізації даних тесту у JSON формат. Функція експорту приймає параметри, які дозволяють приховати правильні відповіді та метадані, що робить експортований файл придатним для розповсюдження серед студентів без розкриття правильних відповідей. Генерація HTML експорту реалізована через функцію `generate_html_export()`, яка створює повноцінну HTML сторінку з вбудованими CSS стилями. HTML шаблон включає заголовок тесту, опис, метадані та всі питання з варіантами відповідей. Підсвітка правильних відповідей реалізована через додавання спеціальних CSS класів до правильних варіантів, що робить їх візуально відмінними. HTML сторінка оптимізована для друку через використання відповідних CSS стилів.

Генерація TXT експорту реалізована через функцію `generate_txt_export()`, яка створює простий текстовий файл з структурованою інформацією про тест. Текстовий формат забезпечує універсальну сумісність та легке читання, але має обмеження у вигляді відсутності форматування та можливості відображення складних структур. Функція обмежує довжину текстів для забезпечення читабельності та уникнення проблем з дуже довгими рядками. Генерація PNG експорту реалізована через функцію `generate_png_export()` з використанням бібліотеки Pillow для створення зображень. Функція створює зображення у форматі A4 з роздільною здатністю 300 DPI, що забезпечує високу якість друку. Зображення включає заголовок тесту, опис, метадані та всі питання з квадратами для відповідей. Для питань типу множинний вибір генеруються квадратики з літерами варіантів відповідей, для питань типу заповнення пропусків генеруються лінії для написання відповіді, а для розгорнутих питань генеруються кілька ліній для детальної відповіді. Опціонально функція може додати бланк відповідей з таблицею для швидкої перевірки тестів після їх проходження.

Опції експорту дозволяють користувачу налаштувати параметри експортованого файлу. Параметр `show_answers` визначає, чи показувати правильні відповіді в експортованому файлі, що дозволяє створювати версії тестів для студентів без правильних відповідей та версії для викладачів з правильними відповідями. Параметр `include_metadata` визначає, чи включати метадані, такі як дата створення тесту, що може бути корисним для архівування або необхідним для ідентифікації тесту. Параметр `include_answer_sheet` визначає, чи додавати бланк відповідей до PNG експорту, що спрощує процес перевірки тестів після їх проходження студентами.

Реалізація frontend компонентів почалася зі створення HTML структури інтерфейсу. HTML структура використовує семантичну розмітку для забезпечення доступності та коректного відображення на різних пристроях. Основна структура включає заголовок сторінки, навігаційні вкладки для різних

функцій системи, контентні області для кожної вкладки та модальні вікна для різних операцій. Використання семантичних елементів HTML5, таких як `header`, `nav`, `main` та `section`, покращує доступність інтерфейсу та спрощує його підтримку. Вкладки навігації реалізовані через кнопки, які перемикають видимість різних контентних областей. Форми створення та редагування тестів використовують стандартні HTML елементи `input`, `textarea` та `select` для введення даних. Модальні вікна реалізовані через `div` елементи з абсолютним позиціонуванням та напівпрозорим фоном.

CSS стилізація та дизайн забезпечують візуально привабливий та сучасний інтерфейс. Градієнтний фон створює привабливий візуальний ефект та робить інтерфейс більш сучасним. Картки для тестів використовують тіні та закруглені кути для створення об'ємного ефекту та візуального розділення різних тестів. Анімації та переходи роблять інтерфейс більш динамічним та приємним у використанні. Адаптивний дизайн забезпечує коректне відображення інтерфейсу на різних розмірах екранів через використання медіа-запитів та гнучких розмірів елементів. Стили для модальних вікон включають напівпрозорий фон, центрування вікна та плавні анімації появи та зникнення. Стили для уведомлень включають різні кольори для різних типів повідомлень та анімації появи та зникнення.

JavaScript логіка клієнта забезпечує всю інтерактивність інтерфейсу та взаємодію з сервером. Обробка подій включає обробку кліків на кнопки, зміни значень у формах, відправку форм та інші дії користувача. Взаємодія з API реалізована через функцію `fetch`, яка відправляє HTTP запити на сервер та обробляє відповіді. Використання `async/await` забезпечує асинхронну обробку запитів без блокування інтерфейсу. Динамічне оновлення інтерфейсу включає додавання та видалення елементів, зміну вмісту елементів та оновлення відображення даних на основі відповідей від сервера.

Система уведомлень реалізована через клас `NotificationSystem`, який забезпечує централізоване управління уведомленнями. Клас має методи для

відображення різних типів уведомлень: `success` для успішних операцій, `error` для помилок, `warning` для попереджень та `info` для інформаційних повідомлень. Кожне уведомлення має іконку, заголовок, текст повідомлення та кнопку закриття. Уведомлення автоматично зникають через вказаний час, але можуть бути закриті вручну користувачем. Система модальних вікон реалізована через клас `ModalSystem`, який забезпечує управління модальними вікнами. Клас має методи для відкриття та закриття модальних вікон, встановлення контенту та обробки підтверджень. Модальні вікна використовуються для попереднього перегляду тестів, налаштування параметрів експорту, підтвердження критичних дій та відображення результатів проходження тестів.

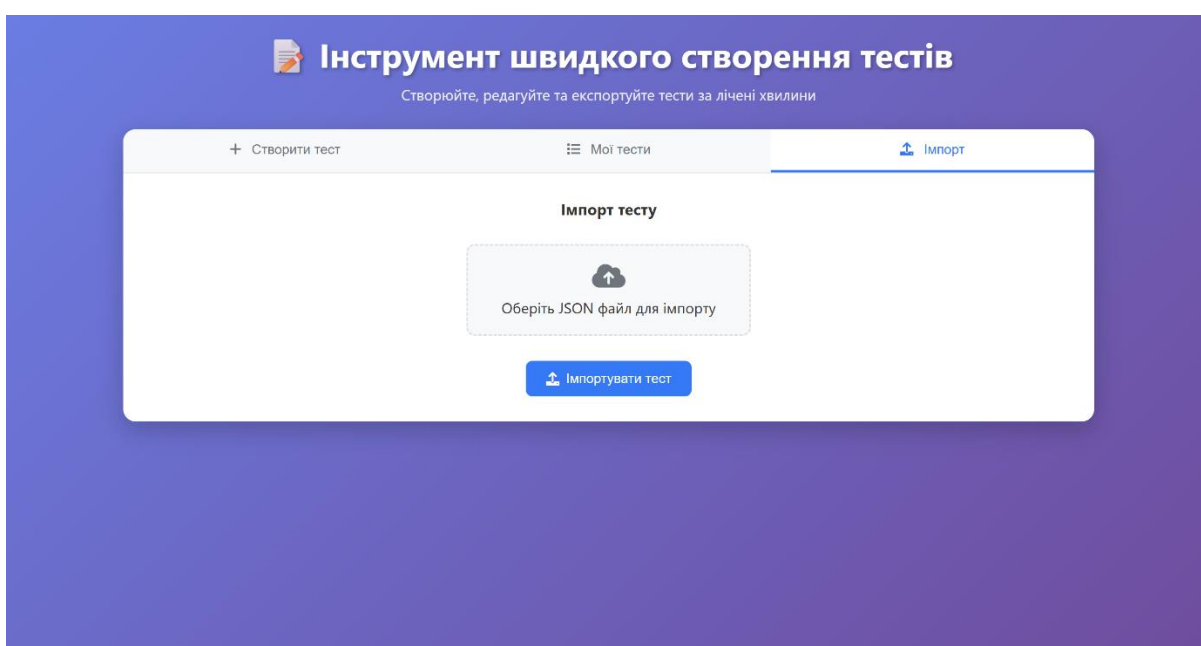
Реалізація функціональності створення тестів включає форму створення тесту з полями для назви, опису та типу тесту. Форма має валідацію на стороні клієнта, яка перевіряє заповненість обов'язкових полів перед відправкою на сервер. Динамічне додавання питань реалізовано через функцію `addQuestion()`, яка створює HTML елементи для нового питання та додає їх до контейнера питань. Функція підтримує різні типи питань та створює відповідні елементи інтерфейсу для кожного типу. Для питань типу множинний вибір створюються поля для введення варіантів відповідей та радіо-кнопки для вибору правильної відповіді. Для питань типу правда-неправда автоматично створюються два варіанти відповіді. Для питань типу заповнення пропусків створюються поля для введення правильних варіантів відповіді. Для розгорнутих питань створюється поле для введення прикладу правильної відповіді. Видалення питань реалізовано через кнопки видалення біля кожного питання, які видаляють відповідні HTML елементи.

Підтримка різних типів питань забезпечує гнучкість у створенні тестів для різних предметів та рівнів складності. Тип `multiple_choice` дозволяє створити питання з кількома варіантами відповіді, де тільки один варіант є правильним. Тип `true_false` дозволяє створити просте питання з двома варіантами відповіді. Тип `fill_blank` дозволяє створити питання, де студент має вписати відповідь,

причому може бути кілька правильних варіантів відповіді. Тип *essay* дозволяє створити розгорнуте питання для детальної оцінки знань студента. Валідація на стороні клієнта перевіряє заповненість всіх необхідних полів та коректність даних перед відправкою на сервер, що забезпечує швидкий зворотний зв'язок користувачу та зменшує навантаження на сервер.

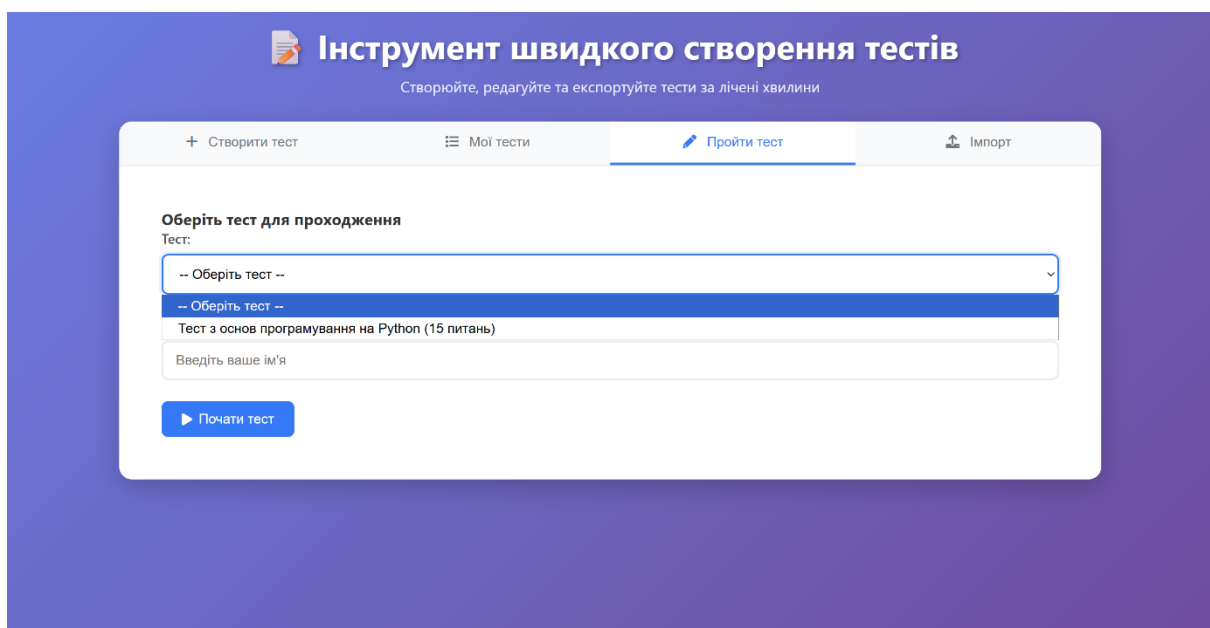
Реалізація функціональності управління тестами включає перегляд списку тестів, який завантажує дані з сервера та відображає їх у вигляді карток. Кожна картка містить назву тесту, опис, дату створення, кількість питань, тип тесту та кнопки для різних дій. Редагування тестів реалізовано через завантаження даних тесту з сервера, заповнення форми редагування цими даними та збереження змін через API. Форма редагування підтримує всі типи питань та дозволяє додавати та видаляти питання. Видалення тестів реалізовано через підтвердження видалення в модальному вікні, після чого відправляється запит на видалення через API та оновлюється список тестів. Попередній перегляд тестів реалізовано через модальне вікно, яке відображає тест у форматі, подібному до формату для студентів, з можливістю показу правильних відповідей.

Реалізація імпорту та експорту забезпечує можливість обміну тестами між різними системами та користувачами.



Імпорт тестів з JSON файлів реалізовано через завантаження файлу через HTML input елемент типу file, читання вмісту файлу, парсинг JSON та валідацію даних перед додаванням тесту до системи. Валідація імпортованих даних використовує ту саму функцію `validate_test_data()`, що й при створенні тесту, що забезпечує консистентність валідації. Обробка файлів та безпека включає перевірку розміру файлу, щоб уникнути перевантаження сервера, перевірку розширення файлу для забезпечення коректного формату, та обробку помилок парсингу JSON для надання зрозумілих повідомлень користувачу.

Реалізація функції проходження тестів включає вибір тесту для проходження через випадаючий список доступних тестів. Відображення питань без правильних відповідей через спеціальний endpoint, який видає правильні відповіді з тесту перед відправкою клієнту, збір відповідей користувача через форму з питаннями, де кожне питання має відповідні елементи для введення відповіді, перевірку відповідей та підрахунок результатів через endpoint, який порівнює відповіді користувача з правильними відповідями та обчислює кількість правильних відповідей та відсоток, та збереження результатів у пам'яті сервера для подальшого аналізу.



Відображення питань без правильних відповідей через спеціальний endpoint, який видає правильні відповіді з тесту перед відправкою клієнту, збір відповідей користувача через форму з питаннями, де кожне питання має відповідні елементи для введення відповіді, перевірку відповідей та підрахунок результатів через endpoint, який порівнює відповіді користувача з правильними відповідями та обчислює кількість правильних відповідей та відсоток, та збереження результатів у пам'яті сервера для подальшого аналізу. Перевірка відповідей реалізована окремо для кожного типу питання: для питань типу множинний вибір порівнюється індекс вибраної відповіді з індексом правильної відповіді, для питань типу заповнення пропусків порівнюється введений текст з правильними варіантами з урахуванням регістру, для розгорнутих питань відповідь завжди вважається правильною, оскільки потребує ручної перевірки.

Захист та безпека системи забезпечуються через кілька механізмів. Захист від XSS атак реалізовано через функцію `sanitize_html()`, яка екранує HTML символи у вхідних даних, використовуючи стандартну бібліотеку Python `html.escape()`. Це запобігає виконанню зловмисного JavaScript коду, який може бути введений у тексти тестів або питань. Екранування використовується при генерації HTML експорту та відображенні даних у інтерфейсі. Валідація вхідних даних на сервері забезпечує перевірку всіх даних перед їх обробкою, включаючи обмеження довжини текстів для запобігання перевантаження системи та перевірку типів даних для забезпечення коректності обробки. Обмеження розміру файлів для імпорту встановлено на 5 мегабайт, що запобігає перевантаженню сервера при завантаженні великих файлів. Безпечна обробка файлів включає перевірку розширення файлу для забезпечення коректного формату, валідацію JSON для перевірки коректності структури даних та обробку помилок для надання зрозумілих повідомлень користувачу у разі проблем.

Висновки до розділу показують, що в результаті реалізації створено повнофункціональну систему, яка забезпечує всі необхідні можливості для створення, редагування, управління та експорту тестів. Реалізовано всі основні

компоненти системи, включаючи backend логіку, frontend інтерфейс, функції експорту та імпорту, систему проходження тестів та механізми захисту. Забезпечено функціональність системи через реалізацію всіх запланованих функцій та можливостей. Реалізовано захист від основних загроз через валідацію даних, екранування HTML та обмеження розміру файлів. Система готова до тестування та подальшого використання.

РОЗДІЛ 4. ТЕСТУВАННЯ ТА ОПТИМІЗАЦІЯ

Тестування системи є критично важливим етапом розробки, який забезпечує якість та надійність реалізованого рішення. Стратегія тестування включає кілька видів тестування, кожен з яких перевіряє різні аспекти системи. Функціональне тестування перевіряє, чи всі функції системи працюють відповідно до вимог та очікувань користувачів. Це включає перевірку створення тестів, редагування, видалення, експорту, імпорту та проходження тестів. Інтеграційне тестування перевіряє взаємодію між різними компонентами системи, такими як взаємодія між frontend та backend, обробка даних між різними функціями та коректність передачі даних між клієнтом та сервером. Інтерфейсне тестування перевіряє коректність відображення інтерфейсу на різних пристроях, зручність використання та відповідність інтерфейсу принципам UX/UI дизайну. Тестування продуктивності перевіряє швидкість роботи системи, час відгуку на різні операції та обробку великих обсягів даних.

Планування тестових сценаріїв включає визначення послідовності дій для перевірки кожної функції системи. Сценарій створення тесту включає відкриття форми створення тесту, введення назви та опису, вибір типу тесту, додавання кількох питань різних типів, встановлення правильних відповідей та збереження тесту. Сценарій редагування тесту включає вибір тесту зі списку, відкриття форми редагування, зміну назви або опису, додавання або видалення питань, зміну правильних відповідей та збереження змін. Сценарій видалення тесту включає вибір тесту зі списку, натискання кнопки видалення, підтвердження видалення в модальному вікні та перевірку, що тест видалено зі списку. Сценарій експорту включає вибір тесту, відкриття модального вікна експорту, вибір формату експорту, налаштування параметрів експорту та завантаження файлу. Сценарій імпорту включає відкриття вкладки імпорту, вибір JSON файлу, завантаження файлу та перевірку, що тест додано до списку. Сценарій проходження тесту включає вибір тесту для проходження, відповіді на всі питання, відправку відповідей та перегляд результатів.

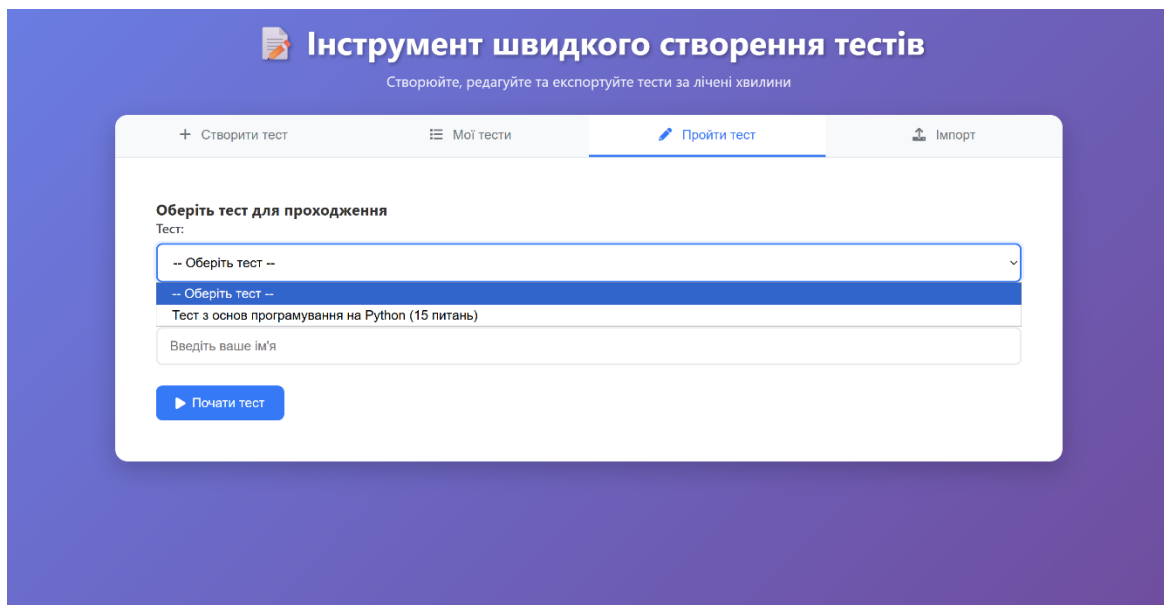
Тестові дані та приклади включають тести з різними типами питань для перевірки підтримки всіх типів, тести різного розміру від малих тестів з п'ятьма питаннями до великих тестів з п'ятдесятьма питаннями для перевірки обробки різних обсягів даних, та граничні випадки, такі як тести з мінімальною кількістю питань, тести з максимальною довжиною текстів, тести з особливими символами у текстах та тести з некоректними даними для перевірки обробки помилок.

Тестування функціональності включає детальну перевірку всіх функцій системи. Тестування створення тестів перевіряє створення тесту з різними типами питань, включаючи множинний вибір, правда-неправда, заповнення пропусків та розгорнуті питання. Перевіряється валідація обов'язкових полів, включаючи перевірку, що система не дозволяє створити тест без назви, без питань або з некоректними даними. Перевіряється обробка помилок, включаючи перевірку, що система коректно обробляє помилки валідації та повертає зрозумілі повідомлення користувачу. Тестування редагування та видалення перевіряє оновлення тесту, включаючи зміну назви, опису, питань та правильних відповідей. Перевіряється видалення тесту, включаючи підтвердження

видалення та перевірку, що тест видалено зі списку. Перевіряється оновлення списку після редагування або видалення тесту.

Тестування експорту у всі формати перевіряє коректність генерації кожного формату. JSON експорт перевіряється на коректність структури даних, наявність всіх необхідних полів та коректність приховування правильних відповідей та метаданих за потреби. HTML експорт перевіряється на коректність HTML структури, наявність CSS стилів, коректність підсвітки правильних відповідей та придатність для друку. TXT експорт перевіряється на читабельність тексту, коректність форматування та наявність всієї необхідної інформації. PNG експорт перевіряється на коректність розміру зображення, наявність всіх питань, коректність квадратиків для відповідей та наявність бланку відповідей за потреби. Тестування імпорту перевіряє імпорт валідного JSON файлу, включаючи перевірку, що всі дані коректно імпортовані та тест додано до списку. Перевіряється обробка невалідних файлів, включаючи файли з некоректною структурою, файли з некоректними даними та файли занадто великого розміру. Перевіряється валідація імпортованих даних, включаючи перевірку, що система відхиляє некоректні дані та повертає зрозумілі повідомлення про помилки.

Тестування проходження тестів перевіряє вибір тесту для проходження, включаючи перевірку, що список тестів коректно відображається та тест можна вибрати. Перевіряється відповіді на питання, включаючи перевірку, що всі типи питань коректно відображаються та можна ввести відповіді. Перевіряється перевірка результатів, включаючи перевірку, що правильні та неправильні відповіді коректно визначені, кількість правильних відповідей коректно підрахована та відсоток коректно обчислений.



Тестування інтерфейсу користувача включає перевірку відображення на різних пристроях. Перевірка на десктопі включає перевірку коректного відображення інтерфейсу на різних розмірах екранів, коректності розташування елементів та зручності використання з клавіатурою та мишею. Перевірка на планшеті включає перевірку адаптивності інтерфейсу до середніх розмірів екранів, коректності відображення форм та зручності використання з сенсорним екраном. Перевірка на мобільному пристрої включає перевірку адаптивності інтерфейсу до малих розмірів екранів, коректності відображення всіх елементів та зручності використання на мобільних пристроях.

Тестування інтерактивності перевіряє роботу всіх інтерактивних елементів інтерфейсу. Перевірка кнопок та форм включає перевірку, що всі кнопки коректно працюють, форми коректно відправляють дані та валідація форм працює коректно. Перевірка навігації між вкладками включає перевірку, що перемикання між вкладками працює коректно, контент кожної вкладки коректно відображається та стан інтерфейсу коректно зберігається при перемиканні. Перевірка модальних вікон включає перевірку, що модальні вікна коректно відкриваються та закриваються, контент модальних вікон коректно відображається та взаємодія з модальними вікнами працює коректно.

Перевірка системи уведомлень включає перевірку різних типів уведомлень, включаючи перевірку, що успішні операції показують успішні уведомлення, помилки показують повідомлення про помилки, попередження показують попереджувальні повідомлення та інформаційні повідомлення коректно відображаються. Перевіряється автоматичне зникнення уведомлень через вказаний час та можливість закриття уведомлень вручну. Тестування модальних вікон включає перевірку відкриття та закриття модальних вікон, включаючи перевірку, що модальні вікна коректно відкриваються при натисканні відповідних кнопок, коректно закриваються при натисканні кнопки закриття або кліку на фон та анімації відкриття та закриття працюють коректно. Перевіряється взаємодія з контентом модальних вікон, включаючи перевірку, що форми в модальних вікнах коректно працюють, кнопки в модальних вікнах коректно виконують дії та контент модальних вікон коректно оновлюється.

Тестування продуктивності включає перевірку часу відгуку системи на різні операції. Створення тесту має виконуватися менше ніж за одну секунду для стандартних тестів з десятима питаннями. Завантаження списку тестів має виконуватися менше ніж за півсекунди для списку з до ста тестів. Експорт тестів залежить від формату та розміру тесту: JSON експорт виконується миттєво, HTML експорт займає менше секунди, TXT експорт займає менше секунди, а PNG експорт може займати кілька секунд для великих тестів через складність генерації зображення. Обробка великих тестів перевіряється на тестах з п'ятдесятьма та більше питаннями для перевірки, що система коректно обробляє великі обсяги даних без сповільнення роботи. Тести з довгими текстами перевіряються для перевірки, що система коректно обробляє питання та відповіді з великою кількістю тексту. Оптимізація обробки включає перевірку, що система ефективно обробляє дані та не витрачає зайві ресурси.

Оптимізація генерації PNG зображень включає оптимізацію розміру зображення для забезпечення швидкої генерації та завантаження, оптимізацію швидкості генерації через ефективне використання бібліотеки Pillow та

оптимізацію якості зображення для забезпечення чіткого друку на стандартних принтерах.

Виявлені проблеми та їх вирішення включають кілька ключових моментів. Проблема зберігання даних у пам'яті означає, що всі дані втрачаються після перезапуску сервера, що є серйозним обмеженням для практичного використання. Рішення цієї проблеми включає документування цього обмеження як відомого недоліку початкової версії системи та планування інтеграції з базою даних для майбутніх версій. Проблеми з PNG експортом виникають через необхідність встановлення бібліотеки Pillow, яка не завжди встановлена за замовчуванням. Рішення включає додавання перевірки наявності бібліотеки при спробі експорту у PNG формат та надання зрозумілих повідомлень про помилки з інструкціями щодо встановлення бібліотеки. Проблеми валідації виникали через недостатню валідацію на стороні клієнта, що призводило до відправки некоректних даних на сервер та необхідності їх обробки на сервері. Рішення включає додавання валідації на обох сторонах: на клієнті для швидкого зворотного зв'язку користувачу та на сервері для забезпечення безпеки та коректності даних. Виправлення помилок включає виправлення обробки помилок для забезпечення, що всі помилки коректно обробляються та повертаються користувачу у зрозумілому форматі, покращення повідомлень для забезпечення, що користувач розуміє, що сталося та що потрібно зробити, та оптимізацію коду для покращення продуктивності та підтримованості.

Оптимізація коду включає рефакторинг backend коду для покращення структури та підтримованості. Винесення функцій валідації в окремі функції дозволяє повторно використовувати код та спрощує тестування. Оптимізація маршрутів включає об'єднання подібних операцій та спрощення логіки обробки запитів. Покращення обробки помилок включає створення централізованої системи обробки помилок та стандартизацію форматів повідомлень про помилки. Оптимізація frontend коду включає модульність JavaScript для розділення коду на логічні модулі, що спрощує підтримку та тестування.

Оптимізація CSS включає використання спільних стилів, мінімізацію дублювання коду та ефективне використання селекторів. Мінімізація запитів до сервера включає кешування даних на клієнті, об'єднання запитів де можливо та використання асинхронних операцій для покращення продуктивності.

Покращення структури даних включає оптимізацію моделей даних для забезпечення ефективного зберігання та обробки, ефективне зберігання через використання оптимальних структур даних та мінімізацію дублювання даних. Зменшення навантаження на сервер включає кешування де можливо для зменшення обчислень, оптимізацію запитів для мінімізації обробки даних та ефективне використання ресурсів сервера.

Обмеження системи та відомі проблеми включають зберігання даних у пам'яті, що означає втрату всіх даних після перезапуску сервера та непридатність для продакшн використання без додаткових заходів. Відсутність багатокористувацької підтримки означає, що всі користувачі бачать одні й ті ж тести та немає розділення даних між користувачами. Відсутність автентифікації означає, що будь-хто може редагувати тести та немає захисту даних. Обмеження розміру тестів включає максимум сто питань на тест та максимум десять варіантів на питання для забезпечення продуктивності системи.

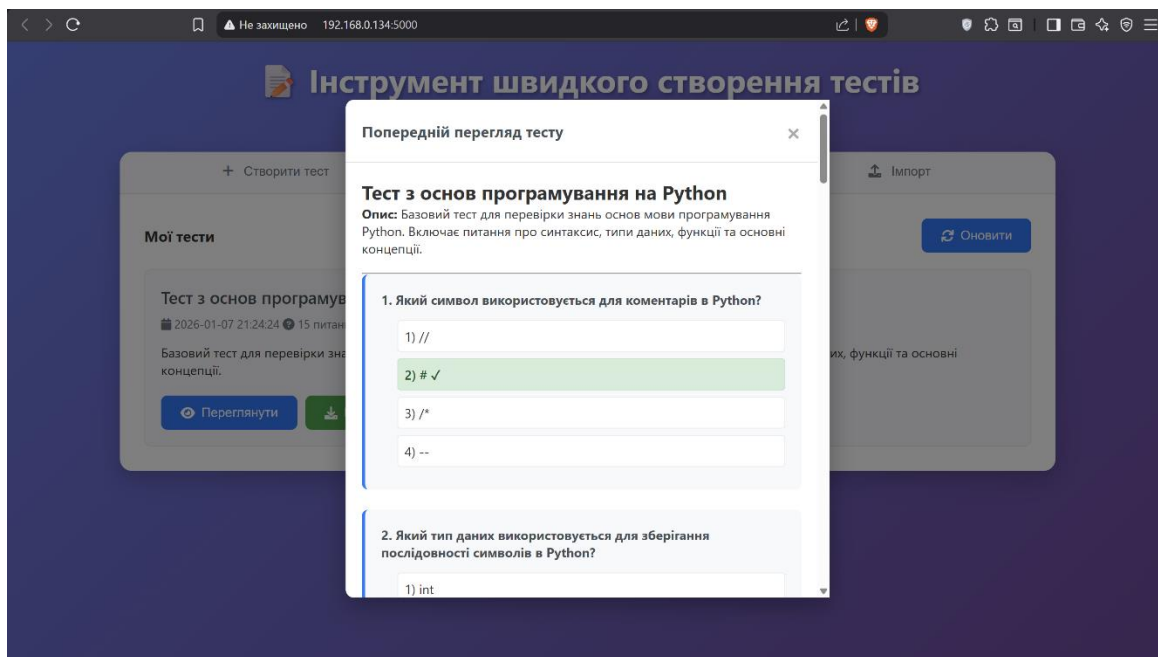
Висновки до розділу показують, що система протестована та працює коректно для всіх основних функцій. Виявлено та виправлено основні проблеми, включаючи проблеми з валідацією, обробкою помилок та продуктивністю. Визначено обмеження та напрямки покращення, включаючи необхідність інтеграції з базою даних, додавання системи автентифікації та покращення продуктивності для обробки великих обсягів даних. Система готова до використання в освітніх цілях з урахуванням виявлених обмежень.

РОЗДІЛ 5. РЕЗУЛЬТАТИ ТА ЇХ ОБГОВОРЕННЯ

Огляд реалізованої системи показує, що в результаті розробки створено повнофункціональний веб-додаток для швидкого створення, редагування, управління та експорту тестів у різних форматах. Основні функції та можливості системи включають створення тестів з чотирма типами питань: множинний вибір, правда-неправда, заповнення пропусків та розгорнуті питання. Кожен тип питання має свої особливості та призначений для різних сценаріїв перевірки знань.

Редагування та видалення тестів забезпечують повний контроль над створеними матеріалами, дозволяючи користувачам оновлювати тести відповідно до змін у навчальних програмах та видаляти застарілі матеріали. Експорт у чотири формати забезпечує універсальність системи: JSON для програмного використання та інтеграції з іншими системами, HTML для перегляду в браузері та електронного розповсюдження, TXT для простого текстового формату та універсальної сумісності, PNG для друку на папері з автоматичним генеруванням бланків відповідей.

Імпорт з JSON дозволяє мігрувати тести з інших систем та обмінюватися матеріалами між користувачами. Проходження тестів забезпечує можливість використання системи не тільки для створення, але й для безпосереднього тестування студентів. Збереження результатів дозволяє відстежувати успішність проходження тестів та аналізувати ефективність навчального процесу.



Технічні характеристики системи включають використання Python 3.7 або вищої версії як основної мови програмування, що забезпечує сучасні можливості та широку підтримку бібліотек. Flask 2.3.3 використовується як веб-фреймворк, що забезпечує легкість, гнучкість та ефективність розробки. Підтримка української мови забезпечує доступність системи для українських освітніх установ та сприяє зручності використання викладачами та студентами. Кросплатформенність дозволяє запускати систему на різних операційних системах, включаючи Windows, Linux та macOS, що розширює можливості використання системи. Локальний запуск дозволяє використовувати систему без інтернет-з'єднання та забезпечує повний контроль над даними, що є критично важливим для забезпечення конфіденційності освітніх матеріалів.

Архітектурні рішення системи базуються на принципах MVC архітектури, що забезпечує чітке розділення відповідальностей між компонентами системи. Model відповідає за структуру даних та бізнес-логіку, View відповідає за відображення даних користувачу, а Controller відповідає за обробку вхідних даних та координацію між Model та View. Використання RESTful API забезпечує стандартизований підхід до проектування API, де кожен ресурс має унікальний

URL, а операції над ресурсами виконуються через стандартні HTTP методи. Модульна структура системи забезпечує легке розширення функціональності та адаптацію під конкретні потреби користувачів. Розділення frontend та backend забезпечує незалежну розробку та тестування компонентів системи.

Порівняння з аналогічними системами показує переваги реалізованої системи. Локальний запуск є унікальною особливістю системи, оскільки більшість існуючих рішень вимагають інтернет-з'єднання та залежать від зовнішніх сервісів. Це дозволяє використовувати систему в умовах нестабільного інтернету або для забезпечення конфіденційності даних. Експорт у PNG для друку є унікальною функцією, яка відсутня в більшості існуючих систем. Це дозволяє швидко створювати тести для друку на папері без необхідності використання додаткових інструментів. Простий інтерфейс робить систему доступною для користувачів з різним рівнем технічної підготовки, на відміну від складних систем типу Moodle, які вимагають значної технічної підготовки. Відсутність необхідності реєстрації спрощує початок роботи з системою та усуває бар'єри для використання. Підтримка української мови забезпечує доступність системи для українських освітніх установ, на відміну від багатьох існуючих систем, які мають обмежену підтримку української мови. Гнучкі опції експорту дозволяють адаптувати тести під різні потреби, включаючи можливість приховування правильних відповідей та метаданих.

Недоліки та обмеження системи включають зберігання даних у пам'яті, що означає втрату всіх даних після перезапуску сервера. Це є серйозним обмеженням для практичного використання, оскільки вимагає постійної роботи сервера та не забезпечує довгострокове зберігання даних. Відсутність багатокористувацької підтримки означає, що всі користувачі бачать одні й ті ж тести, що обмежує можливості використання системи в багатокористувацькому середовищі. Відсутність автентифікації означає, що будь-хто може редагувати тести, що є проблемою для забезпечення безпеки даних. Немає хмарного зберігання, що означає відсутність автоматичного резервного копіювання та

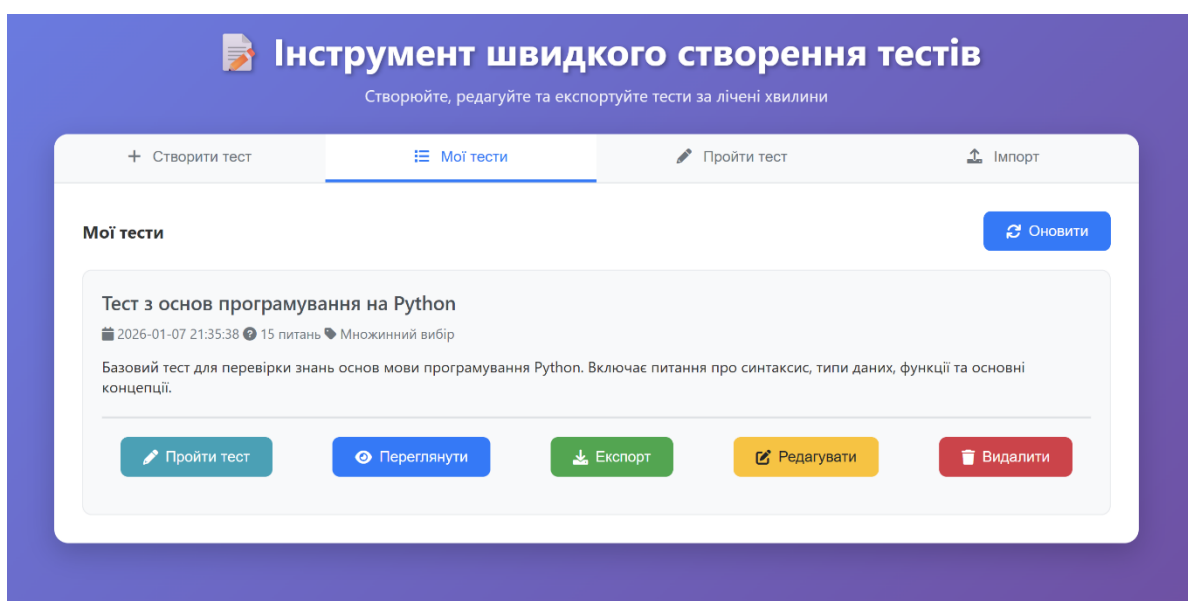
доступу з різних пристроїв. Немає статистики та аналітики, що обмежує можливості аналізу ефективності навчального процесу.

Унікальні особливості системи включають комбінацію локального запуску та веб-інтерфейсу, що поєднує переваги локальних додатків та веб-сервісів. PNG експорт з бланком відповідей є унікальною функцією, яка дозволяє швидко створювати тести для друку з автоматичним генеруванням бланків для швидкої перевірки. Підтримка проходження тестів зі збереженням результатів забезпечує повний цикл роботи з тестами від створення до аналізу результатів.

Практичне застосування системи включає різні сценарії використання. Створення тестів для уроків дозволяє викладачам швидко підготувати матеріали для перевірки знань студентів під час занять. Підготовка матеріалів для друку дозволяє створити тести для проведення в аудиторіях без комп'ютерів або для роздачі студентам для самостійної роботи. Швидке створення тестів для перевірки знань дозволяє ефективно використовувати час викладачів, зосередившись на педагогічній діяльності замість технічних аспектів підготовки матеріалів. Імпорт та експорт тестів між системами дозволяє обмінюватися матеріалами між викладачами та інтегрувати тести з іншими освітніми системами. Цільова аудиторія системи включає викладачів та вчителів, які регулярно створюють тести для перевірки знань студентів та учнів. Тренери використовують систему для оцінки результатів навчання в корпоративному середовищі та на курсах підвищення кваліфікації. Освітні установи використовують систему для централізованого створення та управління тестами на рівні установи. Студенти використовують систему для створення тестів для самоперевірки та підготовки до іспитів.

Приклади створених тестів включають тести з математики для перевірки знання формул та рішення задач, тести з історії для перевірки знання дат, подій та історичних фактів, тести з мови для перевірки знання граматики, лексики та правопису, та тести для корпоративного навчання для оцінки знань співробітників після навчальних програм.

Оцінка ефективності системи показує, що швидкість створення тестів значно перевищує традиційні методи. Створення тесту з десятима питаннями займає приблизно п'ять хвилин, що в три-чотири рази швидше, ніж створення тесту традиційними методами. Експорт тесту виконується менше ніж за дві секунди для більшості форматів, що забезпечує швидку підготовку матеріалів. Імпорт тесту виконується менше ніж за одну секунду, що дозволяє швидко додавати тести з інших джерел.



Зручність використання системи забезпечується інтуїтивним інтерфейсом, який не вимагає навчання для виконання основних операцій. Мінімальна кількість кроків для створення тесту робить процес швидким та ефективним. Зрозумілі повідомлення допомагають користувачам зрозуміти, що відбувається та що потрібно зробити у разі проблем.

Якість експортованих матеріалів забезпечується високою якістю кожного формату. JSON формат забезпечує повну структуру тесту з можливістю програмної обробки та сумісність з іншими системами. HTML формат забезпечує стилізовану веб-сторінку, готову до друку та перегляду в браузері. TXT формат забезпечує читабельний текстовий формат з універсальною

сумісністю. PNG формат забезпечує високу якість зображення, готового до друку на стандартних принтерах.

Надійність системи забезпечується стабільною роботою всіх компонентів, обробкою помилок для забезпечення коректної роботи навіть у разі проблем та валідацією даних для забезпечення коректності обробки.

Критична оцінка результатів показує, що досягнуті цілі включають розробку веб-додатку для створення тестів з усіма необхідними функціями, реалізацію експорту у різні формати, включаючи унікальний PNG експорт, створення зручного інтерфейсу, який не вимагає навчання, та забезпечення функціональності управління тестами, включаючи створення, редагування, видалення та проходження тестів.

Недоліки реалізації включають зберігання даних у пам'яті, що обмежує практичне використання системи, відсутність бази даних, що ускладнює довгострокове зберігання даних, відсутність автентифікації, що обмежує безпеку системи, та обмежену масштабованість через зберігання даних у пам'яті.

Проблеми та обмеження включають втрату даних після перезапуску сервера, відсутність розділення даних між користувачами, обмеження на розмір тестів для забезпечення продуктивності та залежність від Pillow для PNG експорту, що вимагає додаткового встановлення.

Уроки, винесені з розробки, включають важливість планування архітектури на ранніх етапах для забезпечення масштабованості та підтримуваності, необхідність раннього тестування для виявлення проблем на ранніх етапах, важливість документації для забезпечення підтримуваності та розвитку системи та необхідність врахування майбутніх потреб при проектуванні для забезпечення можливості розширення.

Напрямки подальшого розвитку включають інтеграцію з базою даних для забезпечення довгострокового зберігання даних, систему автентифікації для забезпечення безпеки та розділення даних між користувачами, додаткові формати експорту, такі як PDF, Word та Excel, категорії та теги для організації

тестів, статистику та аналітику для аналізу ефективності навчального процесу та багатокористувацьку підтримку для спільної роботи над тестами.

Висновки до розділу показують, що система успішно реалізована та працює для всіх основних функцій. Виявлено обмеження та напрямки покращення, які формують основу для подальшого розвитку системи. Визначено практичну цінність системи для викладачів, освітніх установ та тренерів, які потребують швидкого та зручного інструменту для створення тестів.

ВИСНОВКИ

В результаті проведеного дослідження та розробки створено повнофункціональний веб-додаток для швидкого створення, редагування, управління та експорту тестів у різних форматах. Система реалізована на базі сучасних веб-технологій, включаючи Flask як веб-фреймворк для серверної частини та HTML, CSS та JavaScript для клієнтської частини.

Розроблена система забезпечує всі необхідні функції для ефективного створення та управління тестами, включаючи підтримку різних типів питань, експорт у різні формати та можливість проходження тестів. Система має інтуїтивний інтерфейс, який не вимагає навчання, та забезпечує швидке створення тестів з мінімальними зусиллями. Унікальною особливістю системи є можливість експорту тестів у формат PNG з автоматичним генеруванням бланків відповідей для друку, що відсутнє в більшості існуючих рішень. Модульна архітектура системи забезпечує легке розширення функціональності та адаптацію під конкретні потреби користувачів.

Досягнення мети та виконання задач показує, що всі поставлені завдання успішно виконані. Проаналізовано існуючі інструменти для створення тестів та визначено їх недоліки, що дозволило сформулювати чітке розуміння проблем, які необхідно вирішити в новій системі. Визначено вимоги до системи створення тестів з урахуванням потреб цільової аудиторії, включаючи функціональні та нефункціональні вимоги. Спроековано архітектуру веб-додатку, що забезпечує модульність та можливість розширення, використовуючи сучасні архітектурні підходи та паттерни проектування. Реалізовано backend компоненти на базі Flask, забезпечивши надійну обробку даних, валідацію вхідної інформації та генерацію експортованих файлів. Реалізовано frontend компоненти з інтуїтивним інтерфейсом користувача, який не вимагає навчання та доступний для користувачів з різним рівнем технічної підготовки. Реалізовано функції експорту у формати JSON, HTML, TXT та PNG для різних сценаріїв використання,

забезпечивши високу якість експортованих матеріалів. Реалізовано функцію імпорту тестів для забезпечення сумісності з іншими системами та можливості міграції даних. Протестовано систему та оптимізовано продуктивність, переконавшись у коректній роботі всіх компонентів. Оцінено результати та визначено напрямки подальшого розвитку системи, включаючи інтеграцію з базою даних, систему автентифікації та додаткові формати експорту.

Перспективи подальших досліджень включають інтеграцію з базами даних для забезпечення довгострокового зберігання даних та підвищення надійності системи. Система автентифікації дозволить забезпечити безпеку даних та розділення даних між користувачами, що розширить можливості використання системи в багатокористувацькому середовищі. Додаткові формати експорту, такі як PDF, Word та Excel, розширять можливості використання тестів у різних контекстах та забезпечать кращу сумісність з іншими системами. Статистика та аналітика дозволять аналізувати ефективність навчального процесу, виявляти проблемні теми та адаптувати методи навчання. Багатокористувацька підтримка дозволить командам викладачів працювати разом над створенням тестів та спільно використовувати матеріали. Категорії та теги дозволять організувати тести за предметами, темами або рівнями складності, що спростить управління великою кількістю тестів. Пошук та фільтрація дозволять швидко знаходити потрібні тести серед великої кількості матеріалів.

Рекомендації щодо використання системи включають використання системи в освітніх цілях для локального створення тестів та їх експорту у різні формати. Система особливо корисна для викладачів, які потребують швидкого створення тестів без необхідності використання складних програм або залежності від зовнішніх сервісів. Для продакшн використання необхідна інтеграція з базою даних для забезпечення довгострокового зберігання даних та система автентифікації для забезпечення безпеки та розділення даних між користувачами.

Система може бути використана як основа для розробки більш складних систем управління навчанням або інтегрована з існуючими освітніми платформами. Модульна архітектура системи дозволяє легко додавати нову функціональність та адаптувати систему під конкретні потреби користувачів.

Розроблена система успішно вирішує проблему швидкого створення тестів для освітніх потреб, поєднуючи простоту використання з потужною функціональністю. Система усуває виявлені недоліки існуючих рішень, включаючи залежність від інтернет-з'єднання, обмежені формати експорту та складність налаштування. Модульна архітектура системи забезпечує можливість подальшого розвитку та розширення функціональності відповідно до потреб користувачів. Система готова до використання в освітніх цілях та може служити основою для розробки більш складних систем управління навчанням.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Python Software Foundation. Python 3.7+ Documentation. URL: <https://docs.python.org/3/>
2. Pallets Projects. Flask Documentation (Version 2.3.x). URL: <https://flask.palletsprojects.com/en/2.3.x/>
3. Pallets Projects. Werkzeug Documentation (Version 2.3.x). URL: <https://werkzeug.palletsprojects.com/en/2.3.x>
4. Pallets Projects. Jinja2 Documentation (Version 3.1.x). URL: <https://jinja.palletsprojects.com/en/3.1.x/>.
5. Python Pillow Team. Pillow (PIL Fork) Documentation (Version 10.0.x). URL: <https://pillow.readthedocs.io/en/stable/> .
6. WHATWG. HTML Living Standard. URL: <https://html.spec.whatwg.org/> .
7. W3C. CSS Specifications. URL: <https://www.w3.org/Style/CSS/> .
8. Ecma International. ECMAScript 2015 (ES6) Language Specification. URL: <https://www.ecma-international.org/ecma-262/6.0/> .
9. Mozilla Developer Network (MDN). Web APIs Documentation. URL: <https://developer.mozilla.org/en-US/docs/Web/API> .
10. Mozilla Developer Network (MDN). Fetch API Specification. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API .
11. IETF. RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. URL: <https://tools.ietf.org/html/rfc7231> .
12. IETF. RFC 8259: The JavaScript Object Notation (JSON) Data Interchange Format. URL: <https://tools.ietf.org/html/rfc8259> .
13. W3C. Web Content Accessibility Guidelines (WCAG) 2.1. URL: <https://www.w3.org/WAI/WCAG21/quickref/> .
14. W3C. PNG (Portable Network Graphics) Specification. URL: <https://www.w3.org/TR/PNG/> .

15. Unicode Consortium. The Unicode Standard. URL: <https://unicode.org/standard/standard.html> .
16. Python Package Index (PyPI). Flask Package (Version 2.3.3). URL: <https://pypi.org/project/Flask/2.3.3/> .
17. Python Package Index (PyPI). Werkzeug Package (Version 2.3.7). URL: <https://pypi.org/project/Werkzeug/2.3.7/> .
18. Python Package Index (PyPI). Jinja2 Package (Version 3.1.2). URL: <https://pypi.org/project/Jinja2/3.1.2/> .
19. Python Package Index (PyPI). Pillow Package (Version 10.0.1). URL: <https://pypi.org/project/Pillow/10.0.1/> .
20. Python Package Index (PyPI). MarkupSafe Package (Version 2.1.3). URL: <https://pypi.org/project/MarkupSafe/2.1.3/> .
21. Python Package Index (PyPI). itsdangerous Package (Version 2.1.2). URL: <https://pypi.org/project/itsdangerous/2.1.2/> .
22. Python Package Index (PyPI). click Package (Version 8.1.7). URL: <https://pypi.org/project/click/8.1.7/> .
23. Python Package Index (PyPI). blinker Package (Version 1.6.2). URL: <https://pypi.org/project/blinker/1.6.2/> .
24. Grinberg, M. Flask Web Development: Developing Web Applications with Python. 2nd Edition. O'Reilly Media, 2018. URL: <https://www.oreilly.com/library/view/flask-web-development/9781491991725/>
25. Ramalho, L. Fluent Python: Clear, Concise, and Effective Programming. 2nd Edition. O'Reilly Media, 2022. URL: <https://www.oreilly.com/library/view/fluent-python-2nd/9781492056348/>
26. Lutz, M. Learning Python: Powerful Object-Oriented Programming. 5th Edition. O'Reilly Media, 2013. URL: <https://www.oreilly.com/library/view/learning-python-5th/9781449355722/>
27. Percival, H., Gregory, B. Architecture Patterns with Python: Enabling Test-Driven Development, Domain-Driven Design, and Event-Driven Microservices.

O'Reilly Media, 2020. URL: <https://www.oreilly.com/library/view/architecture-patterns-with/9781492052197/>

28. Flanagan, D. JavaScript: The Definitive Guide. 7th Edition. O'Reilly Media, 2020. URL: <https://www.oreilly.com/library/view/javascript-the-definitive/9781491952016/>

29. Duckett, J. HTML & CSS: Design and Build Websites. Wiley, 2011. URL: <https://www.wiley.com/en-us/HTML+and+CSS%3A+Design+and+Build+Websites-p-9781118008188>

30. Robbins, J. N. Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics. 5th Edition. O'Reilly Media, 2018. URL: <https://www.oreilly.com/library/view/learning-web-design/9781491960196/>

31. Nielsen, J. Usability Engineering. Morgan Kaufmann, 1993. URL: <https://www.nngroup.com/books/usability-engineering/>

32. Krug, S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability. 3rd Edition. New Riders, 2014. URL: <https://www.pearson.com/us/higher-education/program/Krug-Don-t-Make-Me-Think-Revisited-A-Common-Sense-Approach-to-Web-Usability-3rd-Edition/PGM249854.html>

33. Norman, D. The Design of Everyday Things: Revised and Expanded Edition. Basic Books, 2013. URL: <https://www.basicbooks.com/titles/don-norman/the-design-of-everyday-things/9780465050659/>

34. Gamma, E., Helm, R., Johnson, R., Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, 1994. URL: <https://www.pearson.com/us/higher-education/program/Gamma-Design-Patterns-Elements-of-Reusable-Object-Oriented-Software/PGM14333.html>

35. Fielding, R. T. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000. URL: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

36. Martin, R. C. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall, 2008. URL: <https://www.pearson.com/us/higher-education/program/Martin-Clean-Code-A-Handbook-of-Agile-Software-Craftsmanship/PGM63937.html>
37. Hunt, A., Thomas, D. The Pragmatic Programmer: Your Journey to Mastery. 20th Anniversary Edition. Addison-Wesley Professional, 2019. URL: <https://pragprog.com/titles/tpp20/the-pragmatic-programmer-20th-anniversary-edition/>
38. Newman, S. Building Microservices: Designing Fine-Grained Systems. 2nd Edition. O'Reilly Media, 2021. URL: <https://www.oreilly.com/library/view/building-microservices-2nd/9781492034018/>
39. Anderson, L. W., Krathwohl, D. R. (Eds.). A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives. Pearson, 2001. URL: <https://www.pearson.com/us/higher-education/program/Anderson-Taxonomy-for-Learning-Teaching-and-Assessing-A-A-Revision-of-Bloom-s-Taxonomy-of-Educational-Objectives-Abridged-Edition/PGM211309.html>
40. Black, P., Wiliam, D. Assessment and Classroom Learning. Assessment in Education: Principles, Policy & Practice, Vol. 5, No. 1, 1998, pp. 7-74. URL: <https://www.tandfonline.com/doi/abs/10.1080/0969595980050102>
41. Haladyna, T. M. Developing and Validating Multiple-Choice Test Items. 3rd Edition. Routledge, 2004. URL: <https://www.routledge.com/Developing-and-Validating-Multiple-Choice-Test-Items/Haladyna/p/book/9780805854886>
42. Osterlind, S. J. Constructing Test Items: Multiple-Choice, Constructed-Response, Performance, and Other Formats. 2nd Edition. Springer, 1998. URL: <https://www.springer.com/gp/book/9780792381792>
43. Python Software Foundation. pip - The Python Package Installer. URL: <https://pip.pypa.io/en/stable/> .

44. Python Software Foundation. venv - Creation of virtual environments. URL: <https://docs.python.org/3/library/venv.html> .
45. Git. Git Documentation. URL: <https://git-scm.com/doc> .
46. OWASP Foundation. OWASP Top Ten Web Application Security Risks. URL: <https://owasp.org/www-project-top-ten/> .
47. OWASP Foundation. Cross-Site Scripting (XSS) Prevention Cheat Sheet. https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html