Міністерство освіти і науки України

Луцький національний технічний університет
Факультет цифрових, освітніх та соціальних технологій
Кафедра цифрових освітніх технологій

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

РОЗРОБКА ТА ДОСЛІДЖЕННЯ ВЕБ-ДОДАТКА «CHEM TEST»

спеціальність 015.39 Професійна освіта (Цифрові технології)

освітня програма Професійна освіта (комп'ютерні технології)

Виконав: здобувач вищої освіти
групи ПОмз-21
Кульган Андрій Володимирович

_____
(підпис)

Керівник:
д.пед.н., професор
Гулай Ольга Іванівна

_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«___» _____ 2025 р.
д.пед.н., професор
гарант освітньої програми:
Гулай Ольга Іванівна

_____
(підпис)

Луцьк – 2025

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет цифрових, освітніх та соціальних технологій Кафедра цифрових освітніх технологій
Ступінь вищої освіти: магістр
Галузь знань: 01 Освіта/Педагогіка
Спеціальність: 015.39 Професійна освіта (Цифрові технології)
Освітня програма: Професійна освіта (комп'ютерні технології)

ЗАТВЕРДЖУЮ
Завідувач кафедри
цифрових освітніх технологій
_____В. Кабак
«___» _____2025 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Кульгану Андрію Володимировичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: ***Розробка та дослідження веб-додатка «Chem test»***

керівник роботи: *д.пед.н., професор Гулай Ольга Іванівна*
затверджені наказом закладу вищої освіти від «06 » лютого 2025 р. № 70/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи:

«05» грудня 2025 р.

Вихідні дані до роботи *Нормативні документи щодо якості освіти, науково-методична література, вимоги проведення педагогічного експерименту*

3. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Проведення детального аналізу літератури та мережевих інформаційних ресурсів за темою наукової роботи; аналіз цифрових інструментів створення веб-датків, постановка педагогічного експерименту; методи та способи впровадження та застосування в процесі діяльності педагога.

4. Перелік графічного матеріалу: 4 таблиці, 31 рисунок.

5. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв

6. Дата видачі завдання «06» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи магістра	Строк виконання етапів роботи	Примітка
1	*Провести огляд літературних джерел по темі кваліфікаційної роботи магістра*	до 30.08.25	
2	*Провести аналіз загальної проблеми і вибір напрямків дослідження*	до 09.09.25.	
3	*Розробити функціональну схему роботи програмного продукту*	до 17.09.25.	
4	*Описати засоби розробки об'єкта проектування*	до 29.09.25.	
5	*Описати роботу об'єкта проектування*	до 16.10.25	
6	*Розробити методику для проведення експерименту*	до 23.10.25	
7	*Провести аналіз результатів експерименту*	до 12.11.25	
8	*Оцінка отриманих даних та розробка рекомендацій впровадження гейміфікації у навчальному процесі*	до 21.11.25	
9	*Подання завершеного варіанту магістерської кваліфікаційної роботи на розгляд кафедри*	до 05.12.25	

Здобувач вищої освіти _____ Кульган А.В.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Гулай О.І.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Кульган А.В. Розробка та дослідження веб-додатка «Chem test». Рукопис

Кваліфікаційна робота магістра ОП Професійна освіта (Комп'ютерні технології) спеціальності 015.39 Професійна освіта (Цифрові технології). Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота магістра складається зі вступу, чотирьох розділів, висновків, переліку використаної літератури, додатків.

Метою дипломної роботи є розробка веб-додатка «Chem test», основною функціональністю якого є автоматична генерація контрольних робіт з хімії за заданими параметрами.

У першому розділі проведено аналіз літературних джерел і сучасних освітніх технологій, розглянуто роль автоматизації в навчальному процесі, особливості інтерактивних освітніх платформ та систем генерації тестів і завдань.

У другому розділі описано технологічні засади реалізації програмного продукту, зокрема використання фреймворку Django для серверної частини додатка, а також HTML, CSS і JavaScript для розробки користувацького інтерфейсу.

У третьому розділі наведено методику оцінювання ефективності розробленого додатка та організацію експерименту.

Четвертий розділ присвячено обробці, аналізу та інтерпретації отриманих результатів експерименту, що підтвердили ефективність застосування створеного додатка у навчальному процесі та його здатність оптимізувати підготовку дидактичних матеріалів.

Розроблений веб-додаток забезпечує автоматизацію рутинних завдань викладача, сприяє підвищенню ефективності навчального процесу, економії часу на створення контрольних матеріалів і впровадженню цифрових технологій в освіту.

Ключові слова: *веб-додаток, Django, автоматизація навчального процесу, генерація завдань, освітні технології, Python, MySQL*

ABSTRACT

Kulgan A.V. “Development and research of the web application «Chem test». Manuscript

The master's qualification work of the educational program «Vocational Education (Computer Technologies)» of the specialty 015.39 Vocational Education (Digital Technologies). Lutsk National Technical University. Lutsk, 2025.

The master's thesis consists of an introduction, four chapters, conclusions, a list of references and applications.

The purpose of this thesis is the development of a web application “Chem test”, whose main functionality is the generation of chemistry tests based on user-defined parameters.

The first chapter provides an analysis of literary sources and modern educational technologies, examining the role of automation in the learning process, the features of interactive educational platforms, virtual laboratories, and generation systems.

The second chapter describes the technological foundations of the application, including the use of the Django framework for backend development and HTML, CSS, and JavaScript for building the user interface.

The third chapter outlines the methodology for evaluating the effectiveness of the developed application and details the experimental.

The fourth chapter focuses on the processing, analysis, and interpretation of experimental results, which confirm the efficiency of the developed application in optimizing the preparation of didactic materials and enhancing the teaching process.

The web application automates routine teacher tasks, contributing to the improvement of the learning process, time efficiency, and the integration of digital technologies into modern education.

Keywords: *web application, Django, educational process automation, test generation, educational technologies, Python, MySQL.*

ЗМІСТ

ВСТУП

Сучасні тенденції цифровізації освіти зумовлюють потребу в удосконаленні засобів контролю знань, адже традиційні підходи до створення та перевірки контрольних робіт залишаються трудомісткими, ресурсозатратними та недостатньо гнучкими. Викладачі витрачають значну кількість часу на підготовку матеріалів, що стримує ефективність освітнього процесу. Порівняння існуючих ручних і частково автоматизованих способів, свідчить про необхідність розробки інструменту, який би забезпечував якісний, швидкий та об’єктивний контроль знань. Саме тому створення автоматизованої системи формування контрольних робіт є актуальним завданням сучасної педагогічної практики.

Об’єкт дослідження – процес автоматизованого формування контрольних робіт у системах підтримки освітнього процесу.

Предмет дослідження – методи, алгоритми та програмні засоби, що забезпечують автоматичне генерування контрольних робіт та сприяють підвищенню ефективності контролю знань.

Метою магістерської роботи є створення та дослідження автоматизованої системи формування контрольних робіт для підвищення ефективності процесу оцінювання знань учнів.

Завдання, передбачені для реалізації поставленої мети:

- здійснити аналіз існуючих засобів та підходів до автоматизації створення контрольних робіт;
- розробити алгоритмічні та програмні рішення для автоматичного генерування варіантів контрольних робіт;
- визначити критерії оцінювання якості згенерованих матеріалів;
- провести експериментальне дослідження роботи створеної системи та оцінити її ефективність.

Методи дослідження, які використано в роботі:

- аналіз наукових джерел – для визначення стану проблеми та існуючих рішень;
- методи проєктування програмного забезпечення – для розробки структури та функціоналу системи;
- методи алгоритмізації – для побудови механізмів генерації завдань;
- експериментальний метод – для перевірки роботи системи на практиці;
- методи порівняльного аналізу – для оцінювання отриманих результатів.

Наукова новизна отриманих результатів полягає у розробці та обґрунтуванні комплексного підходу до автоматизованого формування контрольних робіт, що передбачає адаптацію матеріалів до рівня підготовки студентів та використання алгоритмів генерації варіантів. У роботі удосконалено підхід до автоматизації контролю знань через розроблення програмного інструменту з розширеними можливостями варіативності та структурування завдань.

Практичне значення одержаних результатів полягає у можливості застосування розробленої системи в освітніх закладах для оптимізації процесу підготовки та проведення контрольних робіт, підвищення об’єктивності оцінювання та зменшення навантаження на викладачів. Система може бути впроваджена у практику викладачів хімії, викладачів фахової та вищої освіти, а також використана як основа для створення аналогічних програм з інших навчальних дисциплін.

Дослідження апробовано на Міжнародній науково-практичній конференції з проблем вищої освіти і науки «Інформаційні технології в освіті, науці і виробництві» (ІТОНВ-2025) [3].

РОЗДІЛ 1
АНАЛІЗ ЛІТЕРАТУРНИХ ДЖЕРЕЛ ЗА ТЕМОЮ КВАЛІФІКАЦІЙНОЇ РОБОТИ МАГІСТРА, ВИКЛАД ЗАГАЛЬНОЇ ПРОБЛЕМИ І ВИБІР НАПРЯМКІВ ДОСЛІДЖЕННЯ

1.1 Освітні технології та їх роль у сучасному навчанні

Освітні технології – це сукупність цифрових інструментів, ресурсів, застосунків та методик, що використовуються для полегшення процесу викладання та навчання. До них можна віднести як апаратне забезпечення, таке як планшети та інтерактивні дошки, так і програмне забезпечення, включаючи симуляції, системи управління навчанням, доповнену та віртуальну реальність, а також штучний інтелект. Основне призначення освітніх технологій полягає в розв'язанні стратегічних завдань освітньої системи, зокрема в прогнозуванні її розвитку, проектуванні та плануванні освітнього процесу, що включає визначення цілей, очікуваних результатів, етапів, методів і організаційних форм навчання. Освітні технології сприяють підвищенню ефективності та якості навчання, роблячи освітній процес більш цілеспрямованим, організованим і орієнтованим на оцінку результатів [22].

Освітні технології можна класифікувати за різними критеріями, що відображають їхню багатогранність та адаптивність до різних педагогічних потреб.

За рівнем застосування вони поділяються на загальнопедагогічні (стосуються загальних положень освітніх процесів), предметні (призначені для вдосконалення викладання окремих предметів) та локальні/модульні (передбачають часткові зміни педагогічних явищ).

За науковою концепцією засвоєння досвіду розрізняють асоціативно-рефлекторні, біхевіористські, розвивальні, сугестивні, нейролінгвістичні та гештальт технології [6].

Класифікація за орієнтацією на особистісну організацію включає

інформаційні, операційні, емоційно-художні, емоційно-моральні, технології саморозвитку, евристичні та прикладні технології.

З точки зору організації та управління пізнавальною діяльністю виділяють структурно-логічні, інтеграційні, ігрові технології (театралізовані, ділові, рольові, комп'ютерні ігри, імітаційні вправи), комп'ютерні технології, діалогові та тренінгові технології [6].

Сучасні категорії та приклади освітніх технологій здебільшого орієнтовані на інтеграцію цифрових інструментів, що дозволяють створювати більш ефективне, доступне та інтерактивне навчальне середовище. Це включає цифрову освіту, яка стала трендом номер один через глобальні зміни, такі як пандемія COVID-19, та змішане навчання внаслідок повномасштабного вторгнення, що поєднує класно-урочну систему з електронним навчанням. Конкретні інструменти охоплюють доповнену (AR) та віртуальну (VR) реальність, штучний інтелект, мобільні застосунки, інтерактивні платформи, інтерактивні дошки, інструменти гейміфікації, системи управління навчанням, хмарні інструменти, набори для кодування та робототехніки, інструменти для співпраці, адаптивне навчальне програмне забезпечення, ШІ-тьютори, допоміжні технології, а також STEM та Maker-технології.

Розмиття меж між традиційними типами технологій та зростання інтегрованих рішень є очевидним. Хоча існують чіткі категорії, такі як «комп'ютерні технології» та «ігрові технології», сучасні приклади, такі як «інструменти гейміфікації» та «інтерактивний мультимедійний контент», демонструють, що ці категорії все частіше інтегруються в єдині платформи або інструменти. Наприклад, сучасні системи управління навчанням можуть включати гейміфікацію, мобільний доступ та функціонал на основі штучного інтелекту. Ця тенденція свідчить про те, що майбутня розробка освітнього програмного забезпечення буде зосереджена на цілісних, багатофункціональних платформах, які поєднують різні педагогічні підходи та технологічні можливості для створення багатшого, більш захоплюючого та персоналізованого навчального досвіду, а не на

окремих, одноцільових інструментах. Це також передбачає збільшення складності в проєктуванні та інтеграції таких систем [2].

Впровадження освітніх технологій значно змінює педагогічні підходи, трансформуючи роль викладача, підвищуючи залученість учнів, розширюючи доступність освіти та розвиваючи ключові навички.

Роль викладачів суттєво змінюється. Вони більше не є лише трансляторами даних, а перетворюються на провідників, що спрямовують навчальний процес. Їхня роль еволюціонує з «мудреця на сцені» на «провідника поруч», що дозволяє учням брати більше відповідальності за власне навчання. Сучасні викладачі активно співпрацюють з іншими педагогами, виходячи за межі свого предмета [15].

Технології значно підвищують залученість та мотивацію учнів, роблячи навчання більш захоплюючим та інтерактивним, що покращує розуміння та запам'ятовування інформації. Інтерактивні та гейміфіковані елементи, такі як вікторини, значки та таблиці лідерів, підвищують інтерес учнів та стимулюють активну участь. Дослідження показують сильний позитивний зв'язок між технологічним навчанням та залученістю учнів, що призводить до активної участі у дискусіях, співпраці з однолітками та взаємодії з викладачами [18].

Освітні технології сприяють персоналізації та доступності освіти. Вони дозволяють диференційоване навчання, адаптоване до унікальних потреб та стилів навчання кожного учня. Наприклад, адаптивне програмне забезпечення використовує штучний інтелект для налаштування контенту, дозволяючи учням опановувати навички у власному темпі. Під час навчання на основі дослідження, студенти самі формулюють проблему, активно шукають відповіді, співставляють гіпотези та результати. Такий підхід розвиває критичне та аналітичне мислення, уміння працювати з даними, здатність планувати експеримент, творчість і дослідницькі навички [16].

Цифрові технології роблять освіту доступнішою, дозволяючи віддалене навчання та усуваючи бар'єри для різних категорій учнів, включаючи тих, хто знаходиться у віддалених районах або має особливі потреби.

Допоміжні технології, такі як програмне забезпечення для перетворення тексту на мову та зчитувачі екрана, відіграють ключову роль у забезпеченні рівного доступу до навчання [22].

Освітні технології також відіграють важливу роль у розвитку навичок 21 століття: критичного мислення, комунікаційних навичок, емоційного інтелекту, аналітичних здібностей, цифрової грамотності та технічних навичок (STEM). Крім того, технології покращують комунікацію та співпрацю. Вони полегшують спілкування між учнями, викладачами та батьками, дозволяючи учням ставити запитання, які вони можуть соромитися задати в класі, а батькам тримати зв’язок з педагогами, для відслідковування навчальних досягнень.

Інструменти для співпраці, такі як Google Meet, Microsoft Teams, Zoom та Padlet, розвивають навички командної роботи та комунікації, дозволяючи учням працювати над спільними проєктами та обмінюватися ідеями, навіть перебуваючи на відстані [1].

Разом із тим, впровадження освітніх технологій потребує балансу: надмірна цифровізація може призвести до перевантаження, відволікання уваги чи зниження концентрації, також існують занепокоєння щодо надмірного часу перед екраном.

Тому ключовим залишається усвідомлене, педагогічно обґрунтоване використання технологій [5].

Користувацький досвід (UX) описує те, що людина відчуває під час взаємодії з веб-сайтом, продуктом, послугою, веб-додатком або програмним забезпеченням. В освіті UX включає розробку освітніх продуктів, що допомагають студентам навчатися та досягати заздалегідь визначених результатів. Інтерфейс користувача (UI) є візуальною частиною взаємодії, що включає розміщення посилань, кнопок та інших інтерактивних компонентів [10].

Користувацький досвід та дизайн інтерфейсу в освітніх технологіях, базується на наступних принципах:

- інтерфейс має бути простим, чистим та зрозумілим, з чіткою та легкою навігацією. Користувачі повинні легко знаходити потрібну інформацію та

функції за мінімальну кількість кліків;

- використання візуальних елементів, інфографіки, іконок, відео, інтерактивних елементів сприяє кращому засвоєнню матеріалу, але перевантаження графікою варто уникати;
- надання користувачам можливості налаштовувати свої профілі, а також надання рекомендацій на основі їхнього прогресу та уподобань. Це змушує користувачів почуватися цінними та більш пов'язаними з продуктом;
- дизайн має враховувати потреби людей із різними можливостями та забезпечувати коректне відображення на всіх типах пристроїв. Крім того, для покращення залученості здобувачів освіти, додаток має бути адаптивним. Цього можна досягти шляхом адаптації уроків до потреб учнів, зміною складності завдань на основі попередніх відповідей [25];
- система має реагувати на дії користувача – повідомленнями, індикаторами прогресу чи підказками.

Дизайн повинен зберігати баланс між простотою та пізнавальною складністю, також він має бути інклюзивним, враховуючи запити всіх учнів, включаючи тих, хто має особливі потреби. Врахування цих принципів є критично важливим для створення ефективних освітніх додатків [10].

1.2 Огляд інтерактивних платформ та симуляторів для вивчення хімії

Серед провідних інтерактивних платформ, що використовуються для вивчення хімії, виділяється **Mozaik Education**. Цей інтернет-сервіс займає перше місце в Україні серед подібних платформ. Програмне забезпечення mozaBook надає можливість створювати та переглядати інтерактивний навчальний зміст, включає безліч готових вбудованих 3D-моделей, а також цифрові підручники, затверджені програмою для 1-11 класів.

MEL Chemistry є одним з найкращих мобільних застосунків, призначених

для візуалізації молекул. Платформа пропонує набори для практичних експериментів та доступ до цифрових матеріалів, включаючи відеогіди, віртуальну та доповнену реальність. Проте, для користування потрібно купити підписку на сервіс [7].

MolView – це інтуїтивно зрозумілий веб-додаток для візуалізації наукових даних. Він дозволяє користувачам малювати молекули в 2D-редакторі та перетворювати їх на 3D-моделі, що є особливо цінним для розуміння складних структур. Платформа також дозволяє шукати структури у великих наукових базах даних, таких як PubChem, RCSB, Crystallography Open Database та NIST Chemistry WebBook, а також переглядати молекулярні спектри. Цільова аудиторія MolView включає учнів, викладачів та студентів, яким потрібні інструменти для візуалізації структур та пояснення хімічних реакцій.

Платформи «**Наурок**» та «**Всеосвіта**» є значними базами презентацій, тестів та інших навчальних матеріалів, що активно використовуються українськими викладачами. Вони пропонують широкий спектр ресурсів для всіх шкільних предметів, включаючи хімію [4].

Вивчення хімії потребує практичних дослідів, проте в умовах сьогодення проведення реальних експериментів часто є складним або навіть неможливим. У таких обставинах віртуальні лабораторії стають ефективною та безпечною альтернативою, забезпечуючи неперервність освітнього процесу.

PhET Interactive Simulations пропонує широкий спектр інтерактивних симуляцій не тільки з хімії, а й з фізики, математики, біології та наук про Землю.

Для вивчення хімії доступні десятки симуляцій, зокрема «Будова атома», «Молекулярні зв’язки», «Розчини», «Кислотність», «Хімічна рівновага». Користувач може змінювати параметри, спостерігати результати в реальному часі й порівнювати їх із теоретичними. Такий підхід сприяє формуванню наукового мислення, розумінню закономірностей і розвитку експериментальних навичок.

Платформа надає матеріали різними мовами, включаючи українську, що значно розширює їхню доступність. PhET дозволяє організовувати дослідницьку

діяльність учнів, наприклад, для формування поняття про шкалу pH за допомогою онлайн-експериментів [7].

ChemCollective є віртуальною лабораторією, що надає онлайн-симуляції хімічних експериментів, допомагаючи учням пов'язати хімічні розрахунки з реальними лабораторними дослідами. Платформа дозволяє користувачеві працювати з наборами реактивів, приладів і розчинів, змінювати їхні концентрації, об’єми та температури, аналізувати результати реакцій. Такі вправи сприяють кращому розумінню концепцій стехіометрії, кислотно-основних реакцій, окисно-відновних процесів тощо. Цільова аудиторія ChemCollective – це викладачі та учні шкіл та закладів вищої освіти.

LabXchange – це безкоштовна освітня платформа, розроблена в Гарвардському університеті, вона забезпечує доступ до інтерактивних симуляцій та навчальних матеріалів для вивчення природничих наук. Для хімії на платформі доступні віртуальні експерименти, що охоплюють як базові поняття (атом, молекула, хімічний зв’язок), так і складніші теми – синтез, хімічну рівновагу, аналітичні методи. Усі експерименти супроводжуються мультимедійними поясненнями, що підвищує залученість здобувачів освіти. Платформа орієнтована на викладачів, учнів та науковців.

Labster – онлайн-платформа для симуляції лабораторних експериментів, яка дозволяє студентам і учням вивчати науку через віртуальні лабораторії. Через 3D-середовище та інтерактивні симуляції, платформа надає можливість проводити дослідження з різних наукових дисциплін, а також тестувати складні наукові концепції, розвивати практичні навички роботи з лабораторним обладнанням, не потребуючи фізичного доступу до реальних лабораторій [17].

Corinth 3D є інтерактивним освітнім інструментом, що має велику бібліотеку 3D-моделей для STEM-освіти, включаючи хімію, біологію, фізику та інші предмети. Контент на платформі науково перевірений у співпраці з провідними університетами, що забезпечує його точність та достовірність. Цей інструмент спрямований на підвищення мотивації учнів та полегшення підготовки уроків для

вчителів, дозволяючи візуалізувати абстрактні та складні теми.

Акцент на 3D-моделюванні, як це реалізовано в Corinth 3D, MolView та MEL Chemistry, дозволяє учням «бачити» та маніпулювати об'єктами, які є невидимими або абстрактними в реальному світі. Це покращує просторове мислення, допомагає у розумінні хімічних зв'язків, реакцій та властивостей речовин на молекулярному рівні, а також може сприяти розвитку інженерного мислення.

Всеукраїнська школа онлайн (ВШО) є державною платформою, створеною за ініціативи Міністерства освіти і науки України. На сайті можна знайти матеріали для учнів 5-11 класів, включаючи відеоуроки, тести та контрольні роботи, додаткові матеріали. Доступ до ресурів – безкоштовний, є мобільні застосунки для різних операційних систем. Платформа охоплює всі шкільні предмети, і для хімії доступні курси для 7-11 класів, а також первинні та вторинні діагностичні тестування для 7-9 класів. Цільова аудиторія ВШО – це школярі та вчителі.

«МійКлас» є українською освітньою онлайн-платформою, яка спрямована на організацію ефективного дистанційного. На сервісі доступна велика база інтерактивних завдань з різних шкільних предметів, автоматичне тестування, домашні завдання та електронні щоденники. Платформа активно використовується вчителями для перевірки знань здобувачів освіти, а також батьками та школярами для самостійного повторення та закріплення матеріалу.

iLearn – це українська платформа, створена громадською спілкою «Освіторія» для підготовки учнів 10–11 класів до Національного мультипредметного тесту (НМТ) та зовнішнього незалежного оцінювання (ЗНО). Платформа надає матеріали з основних шкільних предметів, дозволяє перевірити свої знання, за допомогою тестування, переглянути вебінари та курси для підвищення рівня знань[4].

Сучасні онлайн-сервіси значно спрощують процес створення та проведення оцінювання знань учнів, пропонуючи широкий спектр функціональних можливостей, від платформи для створення контрольної роботи за власними

матеріалами, до повністю готових тестів та робіт.

Google Forms – один з найпопулярніших та найзручніших інструментів для створення тестів завдяки своїй простоті використання та безкоштовному доступу. Платформа дозволяє створювати тести з автоматичним оцінюванням, інтегрувати їх з Google Classroom, використовувати різноманітні форми запитань (відкриті, закриті, шкали), аналізувати результати у вигляді таблиць і графіків, а також додавати зображення, відео та аудіофайли для урізноманітнення завдань [8].

Наурок є однією з провідних українських освітніх платформ, яка пропонує велику бібліотеку готових тестових завдань з усіх шкільних предметів, а також надає вчителям можливість створювати власні тести. Використання готових тестів дозволяє значно економити час вчителя та не потребує додаткової підготовки.

JustClass – це безкоштовна українська платформа, розроблена спеціально для вчителів, яка допомагає швидко організовувати навчальний процес та створювати тести. Платформа також пропонує готові шкільні завдання [8].

Kahoot! є дуже популярною платформою для створення тестів у форматі вікторин з інтерактивними елементами, що активно використовує гейміфікацію для підвищення мотивації та залученості учнів. Вона дозволяє проводити тести в живому режимі, має кольоровий та динамічний інтерфейс, рейтингову систему та можливість створювати тести, адаптовані для шкільної програми. Вчителі отримують звіти про те, як учні пройшли завдання.

Wayground– ще одна платформа для створення тестів, що дозволяє проводити оцінювання у різних форматах. Доступне автоматичне оцінювання відповідей, можливість встановлення таймерів та елементи гейміфікації для підвищення залученості учнів. Платформа має велику базу готових тестів та підтримує українську мову, що робить її зручною для шкільного використання. Вчителі мають можливість встановити захист від списування, переглядати звіти про виконані тести та коригувати навчальний процес. Проте, без платної підписки, платформа має обмежену функціональність [8].

LearningApps є сервісом для створення інтерактивних вправ та ігор. Він

дозволяє створити нестандартні форми командної роботи, такі як Web-квести. [9]

Quizlet – це платформа для навчання через картки, яка генерує тести на їхній основі. Вона пропонує режими тренування та гри, голосове відтворення тексту та можливість обміну картками між учнями та вчителями [8].

Classtime – це інтерактивна платформа для швидкого тестування, яка забезпечує миттєвий зворотний зв'язок про рівень розуміння учнів. Вона пропонує стандартизовані банки питань, поглиблений аналіз даних та персоналізовані рекомендації. Платформа також має українськомовні відеоінструкції, що спрощує її використання для українських педагогів.

Kwiga – платформа для створення тестів онлайн, що пропонує безкоштовний конструктор тестів та можливість миттєвої публікації опитувань. Перші 20 проходжень кожного місяця безкоштовні.

Серед інших релевантних сервісів, що використовуються для створення тестів та вікторин, можна назвати Online Test Pad, ClassMarker, PurposeGames, ClassTools.net, Testmoz, Quizalize, Mentimeter, Nearpod та Learning.ua [9].

Поява та швидке поширення інструментів на основі штучного інтелекту (ШІ) спричинили значний зсув у методах підготовки навчальних матеріалів, відкриваючи нові можливості для педагогів. Інструменти ШІ, можуть значно прискорити процес створення різноманітних навчальних матеріалів та завдань, адаптованих до різних рівнів складності, що може звільнити час вчителя для більш індивідуальної роботи з учнями.

ChatGPT, є однією із систем штучного інтелекту, яка пропонує безкоштовну та преміум-версії. Його функціонал включає генерацію тексту, аналітичні та візуальні інструменти, можливість завантажувати документи та витягувати з них ключові висновки та інформацію, а також роботу зі звуком. Цей інструмент може використовуватися для створення різноманітних навчальних матеріалів, узагальнення інформації та допомоги в критичному аналізі. В безкоштовній версії існує обмеження на кількість запитів та модель ШІ, яку можна використовувати.

Claude є ще однією великою мовною моделлю, яка дуже корисна для

дослідницьких та академічних завдань. Він значно пришвидшує узагальнення статей, створення тез та допомогає в аналізі текстів. Claude пропонує різні версії, включаючи безкоштовний доступ до однієї з останніх моделей з обмеженим використанням.

Gemini – чат-бот компанії Google, який надає можливість використовувати функції великих мовних моделей, зокрема підсумовування тексту, аналіз даних, створення зображень і переклад. Gemini може обробляти аудіо, наприклад, транскрибувати мову, аналізувати медіавміст або допомагати створювати тести.

Системи для генерації тестів та завдань пропонують численні переваги, які оптимізують процес оцінювання та підвищують його ефективність. Однією з головних переваг цих систем є автоматизація перевірки знань та економія часу викладача. Платформи, такі як Google форми, Quizizz, Kahoot та Наурок, автоматично оцінюють відповіді учнів, значно скорочуючи час, який вчитель витрачає на ручну перевірку.

Миттєвий зворотний зв'язок для учнів є ще однією важливою перевагою. Учні отримують негайну інформацію про правильність своїх відповідей, що сприяє швидшому виправленню помилок та закріпленню матеріалу. Це підтримує активне навчання та самоконтроль.

Також такі сервіси пропонують широкий спектр форматів завдань та можливість їх адаптації. Вони включають різноманітні типи питань, а також дозволяють додавати мультимедійні елементи, що урізноманітнює тестування та робить його більш інтерактивним [8].

Збір та аналіз даних про успішність учнів є потужною функцією цих систем. Платформи надають статистику успішності, звіти про виконані тести та аналіз результатів у вигляді таблиць і графіків. Це дозволяє викладачам виявляти прогалини у знаннях, відстежувати прогрес учнів та коригувати навчальний процес на основі даних.

Гейміфікація та підвищення залученості є значною перевагою, яку пропонують такі інструменти, як Kahoot! та Wayground. Вони перетворюють

тестування на інтерактивну гру, підвищуючи мотивацію та залученість учнів через змагальний елемент, що робить процес оцінювання менш стресовим та більш захоплюючим [8].

Поряд з перевагами, системи для генерації тестів та завдань мають і певні недоліки, які вимагають уваги та стратегічного підходу.

Одним із найсерйозніших викликів є ризики академічної недоброчесності, а саме, списування. В умовах онлайн-тестування існує підвищений ризик списування, а поява інструментів штучного інтелекту може посилити цю проблему, оскільки учні можуть використовувати їх для генерації відповідей на завдання. Це вимагає розробки нових методів оцінювання та стратегій для забезпечення чесності.

Більшість автоматизованих систем тестування краще підходять для перевірки фактологічних знань та розуміння базових концепцій, ніж для оцінки творчого мислення та критичного аналізу.

Залежність від стабільності інтернет-з'єднання є практичним недоліком. Нестабільний інтернет або відсутність доступу до пристроїв може перешкоджати проведенню онлайн-тестування, створюючи нерівні умови для учнів, особливо в регіонах з недостатньо розвиненою інфраструктурою.

Наявність платних функцій, що обмежують повний доступ, також є проблемою. Хоча багато платформ пропонують безкоштовні версії з базовим функціоналом, розширені можливості, такі як більша кількість завдань або детальна аналітика часто доступні лише за платною підпискою [9].

Використання штучного інтелекту для генерації контенту створює необхідність верифікації згенерованого. Хоча ШІ може швидко створювати завдання, викладач повинен ретельно перевіряти їх на точність та відповідність навчальній програмі. Це вимагає додаткового часу та експертизи з боку педагога, щоб забезпечити якість та достовірність навчальних матеріалів.

РОЗДІЛ 2
ОПИС РІШЕННЯ ЗАГАЛЬНОЇ ПРОБЛЕМИ ТА ОСНОВНИХ МЕТОДІВ ДОСЛІДЖЕННЯ

2.1 Огляд сучасних веб-технологій

Еволюція веб-розробки за останні десятиліття перетворила статичні сторінки на складні, динамічні та інтерактивні додатки. Сучасний веб вимагає забезпечення високої інтерактивності, надійної безпеки, здатності обробляти значні обсяги даних та обслуговувати мільйони користувачів. Цей перехід – від простих інформаційних ресурсів до повноцінних програмних платформ значно підвищив вимоги до технологій, які використовуються для їх створення.

Вибір правильного набору технологій є критично важливим для успіху будь-якого веб-проєкту. Він безпосередньо впливає на швидкість розробки, ефективність функціонування, довгострокову підтримку, масштабованість та здатність адаптуватися до майбутніх змін. Необхідно враховувати не лише поточні потреби, а й потенційний ріст та еволюцію додатка.

Python є однією з найбільш популярних мов програмування для розробки серверної частини веб-додатків, він відповідає за логіку, що залишається невидимою для кінцевого користувача, але виконує операції, які забезпечують основну функціональність веб-сайту. Функції Python у веб-розробці охоплюють широкий спектр завдань: від надсилання та отримання даних з серверів, обробки інформації та взаємодії з базами даних до маршрутизації URL-адрес та забезпечення безпеки даних. Фактично Python виступає як «клей», що ефективно пов'язує різні компоненти веб-додатка [24].

Роль Python як серверної мови програмування підкреслює фундаментальний архітектурний поділ у веб-розробці на клієнтську та серверну частини. Це означає, що складні обчислення, доступ до баз даних, обробка конфіденційних даних та реалізація логіки додатка ефективніше виконуються на сервері. Такий підхід

забезпечує підвищену безпеку, оскільки внутрішня логіка та облікові дані бази даних не виставляються на клієнтську сторону, а також покращує продуктивність, використовуючи потужніші серверні ресурси. Цей поділ є ключовим для побудови масштабованих, безпечних та підтримуваних веб-додатків, оскільки він дозволяє спеціалізувати команди розробників та оптимізувати ресурси. Позиція Python як домінуючої бекенд-мови є прямим наслідком цієї архітектурної парадигми, оскільки він надає надійні інструменти та фреймворки, такі як Django, для ефективного управління на стороні сервера.

HTML, або Hypertext Markup Language, є стандартом, що використовується для створення та дизайну веб-сторінок, формуючи їхню базову організацію та вміст. Він слугує фундаментом, на якому будується весь візуальний інтерфейс користувача. HTML надає структуру та макет веб-сторінки за допомогою системи тегів та атрибутів, визначаючи такі елементи, як заголовки, абзаци, зображення, посилання та багато іншого. Це дозволяє дизайнерам створювати візуально привабливі та добре організовані сторінки, розміщуючи елементи в ієрархічній структурі [14].

Часто використовується аналогія, що HTML є «кістяком» веб-сторінки, тоді як CSS надає «шкіру» (візуальне оформлення), а JavaScript – «мозок» (інтерактивність). Ця метафора ефективно підкреслює їхні взаємодоповнюючі ролі у створенні повноцінного веб-досвіду.

Серед ключових переваг HTML виділяють: надзвичайну легкість у використанні, що робить його легким в освоєнні, навіть для людей, без попереднього досвіду програмування; широку підтримку у всіх основних веб-браузерах, що забезпечує універсальне та послідовне відображення веб-сайтів для всіх відвідувачів, а також покращення доступності для людей з обмеженими можливостями за допомогою відповідних HTML-тегів та атрибутів, що робить веб-сайт більш зручним для користувачів екранних читачів або інших допоміжних технологій [14].

Простота та універсальність HTML є дуже важливими для створення

адаптивних та ефективних веб сторінок. Основне призначення HTML полягає в семантичній розмітці вмісту (наприклад, використання <h1> для основного заголовка або <p> для абзацу). Це семантичне значення потрібне не лише людям, а й комп’ютерам: веб-браузери використовують його для відображення контенту, пошукові системи – для індексації та ранжування сторінок, а допоміжні технології (наприклад, програми для читання з екрана для користувачів із вадами зору) – для інтерпретації та передачі вмісту. Оскільки HTML надає універсальну, машиночитану структуру, це забезпечує доступність контенту на широкому спектрі пристроїв та браузерів.

Таким чином, простота та структурна роль HTML є фундаментальними для глобального охоплення та інклюзивності вебу, роблячи його не просто інструментом макетування.

Для оформлення веб-сторінок (кольорів, шрифтів, відступів, анімацій, позиціонування) використовується CSS (Cascading Style Sheets) [23].

Завдяки централізованому управлінню стилями, CSS дозволяє застосовувати їх на всіх сторінках веб-сайту, що значно спрощує підтримку та оновлення дизайну, та усуває необхідність змінювати кожен елемент окремо.

Переваги використання CSS у веб-розробці є численними: він значно покращує зовнішній вигляд веб-сайтів, пропонуючи гнучкість та можливості налаштування, які неможливо досягти лише за допомогою HTML; спрощує розробку та оновлення, оскільки зміни дизайну можна вносити без впливу на вміст, це зменшує зусилля, пов'язані з розробкою та оновленнями, сприяє економії часу та грошей, усуваючи складнощі елементів дизайну та зменшуючи витрати на працю; покращує користувацький досвід завдяки привабливому дизайну, легшій навігації та підвищенню швидкості завантаження сторінок; а також забезпечує можливість адаптації до різних розмірів екранів та пристроїв, що є критично важливим для сучасних мобільних веб сайтів. Крім того, CSS може покращити доступність, дозволяючи розробникам створювати дизайни, сумісні з програмами для читання з екрана та іншими допоміжними технологіями [23].

Для додавання інтерактивності використовують JavaScript. Це скриптова мова програмування, яка дозволяє реалізовувати складні функції на веб-сторінках. Його часто називають «мозком» веб-сторінки, оскільки він оживляє її, дозволяючи виконувати дії у відповідь на взаємодію користувача.

JavaScript працює, взаємодіючи та змінюючи один або кілька елементів на сторінці через Document Object Model, виконуючи операції з ними (наприклад, додавання чисел, зміна тексту, динамічне застосування нових стилів) та реагуючи на певні події, що відбуваються на веб-сторінці (наприклад, кліки користувача або введення даних). Його код виконується у «середовищі виконання» браузера, при цьому кожна вкладка браузера має власне ізольоване середовище, що забезпечує безпеку та запобігає прямому впливу коду однієї вкладки на іншу, захищаючи від потенційних зловмисних дій.

JavaScript перетворив Інтернет зі статичного середовища на динамічну та інтерактивну платформу. До нього, будь-яка взаємодія користувача, що вимагала змін на сторінці (наприклад, надсилання форми або натискання кнопки для відображення нового вмісту), зазвичай передбачала повторне завантаження даних з сервера. Здатність JavaScript динамічно змінювати частини сайту означає, що ці зміни можуть відбуватися миттєво в браузері користувача без необхідності звернення до сервера. Це є наріжним каменем односторінкових додатків та багатьох інтернет-додатків, де інтерфейс користувача більше схожий на настільну програму, ніж на традиційний веб-сайт.

JavaScript є рушієм сучасної веб-розробки, який дозволяє створювати складні користувацькі інтерфейси та комплексну логіку і фундаментально змінює спосіб взаємодії користувачів з онлайн-контентом та послугами [26].

Бази даних є невід’ємною складовою будь-якого веб-додатку, адже саме в них зберігається основна інформація. Розрізняють реляційні (MySQL, PostgreSQL, SQLite) та нереляційні (MongoDB, Redis, Firebase) системи.

MySQL – це одна з найпопулярніших у світі реляційних систем керування базами даних з відкритим вихідним кодом. Для роботи з цією системою

використовується SQL (Structured Query Language), яка дозволяє виконувати різні операції над даними: створення (CREATE), читання (READ), оновлення (UPDATE) та видалення (DELETE). Цей набір операцій відомий як CRUD-операції.

Системи керування базами даних, або СКБД, є програмним забезпеченням, що забезпечує взаємодію користувачів та програм з базами даних. Вони виконують функцію посередника між користувачем і даними, що зберігаються на диску.

До основних їх функцій відносять:

- СКБД дозволяє структуровано зберігати великі обсяги інформації, запобігаючи її втраті та нецільовому використанню;
- забезпечення цілісності даних, шляхом використання різних обмежень, таких як унікальні ключі та зовнішні ключі гарантує, що дані є точними та послідовними;
- можливість встановлювати рівні доступу та права користувачів, що забезпечує безпеку конфіденційної інформації від несанкціонованого доступу;
- СКБД дозволяє одночасний доступ до даних кільком користувачам або процесам без конфліктів;
- відновлення після збоїв за допомогою вбудованих механізмів для резервного копіювання та відновлення даних.

Таким чином, СКБД є не просто сховищем, а складним інструментом для управління даними, що забезпечує їх надійність, доступність та цілісність. Без них сучасні додатки, корпоративні системи та дослідження були б неможливими [19].

Всі описані технології є блоками, на яких побудовані більшість веб-сайтів. Вони працюють у тісній взаємодії, щоб створити повноцінний, функціональний та візуально привабливий веб-сайт (таблиця 2.1). HTML надає базову структуру та вміст веб-сторінки. CSS контролює зовнішній вигляд та представлення цього вмісту, що надає стиль та естетику. JavaScript, у свою чергу, програмує функціональність та інтерактивність, що дозволяє сторінці реагувати на дії користувача та динамічно змінюватися.

Уявімо кнопку на веб-сторінці. HTML створює саму кнопку з текстом, CSS робить її красивою з певним кольором та округлими краями, а JavaScript додає функціональність: коли користувач натискає кнопку, JavaScript може змінити текст на сторінці, показати повідомлення або відправити дані на сервер.

Ця архітектура дозволяє розділити відповідальність: дизайнери можуть працювати з CSS, контент-менеджери з HTML, а розробники з JavaScript, при цьому всі три технології гармонійно працюють разом для створення повноцінного веб-досвіду.

Важливим доповненням до цих технологій є системи управління базами даних. Вони забезпечують надійне зберігання даних і можливість їх швидкого доступу, що дозволяє веб-сайтам ефективно працювати з великими обсягами інформації та підтримувати динамічний контент [20].

Таблиця 2.1 – Основні ролі HTML, CSS, JavaScript та СУБД у веб-розробці

Технологія	**Основна роль**	**Ключові функції**	**Переваги**
1	2	3	4
HTML	Структура та вміст	Визначення елементів, організація ієрархії сторінки. Надання семантичного значення контенту.	Легкість використання, широка підтримка браузерами, покращення доступності.
CSS	Стилізація та візуальне оформлення	Визначення кольорів, шрифтів, макетів, анімацій, адаптивний дизайн. Відокремлення представлення від структури.	Покращення зовнішнього вигляду, підвищення швидкості завантаження сторінок, покращення UX.

Продовження таблиці 2.1

1	2	3	4
JavaScript	Інтерактивність та динамічний контент	Маніпуляція DOM, обробка подій, валідація форм, асинхронні запити, створення інтерактивних елементів.	Створення динамічних та чутливих користувацьких інтерфейсів, швидкий відгук без перезавантаження сторінки.
СУБД	Зберігання та управління даними	Організація, зберігання, доступ та маніпулювання даними через запити	Швидкий доступ до даних, масштабованість, забезпечення надійності та цілісності даних, підтримка складних запитів.

2.2 Обґрунтування вибору Django для розробки веб-додатка

Django – це фреймворк для розробки веб-додатків мовою Python, який дозволяє швидке створення, тестування та розгортання проєктів та орієнтований на створення структурованих, безпечних і масштабованих рішень.

Django розроблений для того, щоб допомогти розробникам якомога швидше перетворити ідею додатка у готовий продукт. Він бере на себе більшу частину рутини веб-розробки, дозволяючи зосередитися на написанні унікальної бізнес-логіки додатка без необхідності «винаходити колесо»

Ключовим аспектом, що сприяє цій швидкості, є філософія «Batteries Included» (Все включено). Це означає, що Django постачається з більшістю бібліотек та інструментів, необхідних для поширених випадків використання, «з коробки» До них належать: потужний об'єктно-реляційне відображення (Object-Relational Mapper), проміжне програмне забезпечення (middlewares), надійна

система автентифікації та дозволів, HTTP-бібліотеки, підтримка багатосайтовості, інтернаціоналізація, автоматична адміністративна панель та гнучка система шаблонів. Така повнота значно зменшує потребу в інтеграції сторонніх інструментів та їх конфігурації, що є поширеною проблемою в інших фреймворках.

Іншим фундаментальним принципом, що прискорює розробку, є DRY (Don't Repeat Yourself – Не повторюй себе). Django активно заохочує та навіть змушує дотримуватися цього принципу, мінімізуючи дублювання коду. Фреймворк спроєктований таким чином, що розробнику доведеться докласти зусиль, щоб порушити цей принцип.

Філософія «Batteries Included» у поєднанні з примусовим дотриманням принципу DRY є основним рушієм швидкості розробки в Django. Це робить його особливо привабливим для проєктів з жорсткими термінами або обмеженим бюджетом, а також для швидкої ітерації мінімально життєздатних продуктів та прототипів [11].

Django допомагає розробникам уникнути багатьох поширених помилок у сфері кібербезпеки. Це один із його ключових принципів, що забезпечує високий рівень захисту для веб-додатків.

Фреймворк має вбудований захист від більшості поширених типів вразливостей, що значно знижує ризики для додатків:

- SQL-ін'єкції: запити, що генеруються Django, захищені від SQL-ін'єкцій, оскільки код запиту визначається окремо від його параметрів, які безпечно екрануються, запобігаючи виконанню довільного SQL-коду зловмисником;
- cross-site scripting: шаблони Django за замовчуванням екранують специфічні символи, які є особливо небезпечними для HTML, захищаючи від більшості подібних атак. Це запобігає впровадженню зловмисних клієнтських скриптів у браузери інших користувачів;
- CSRF (Cross-Site Request Forgery): Django має вбудований захист від більшості типів CSRF-атак, який перевіряє наявність секретного токена в

кожному POST-запиті. Цей механізм не дозволяє зловмисникам виконувати дії, використовуючи облікові дані іншого користувача без їхнього відома чи згоди [21].

Крім того, Django включає вбудовану, надійну систему автентифікації та дозволів, яка автоматично шифрує паролі та інші конфіденційні дані, під час передачі.

Комплексні, вбудовані функції безпеки Django значно зменшують зусилля, необхідні для забезпечення безпеки. Це дозволяє розробникам зосередитися на логіці додатка, а не на низькорівневих механізмах захисту. Такий підхід неявно сприяє формуванню більш безпечної культури розробки, оскільки безпечні практики стають типовими. Це має вирішальне значення для захисту конфіденційних даних та підтримки довіри користувачів, що є незамінним для будь-якого успішного веб-додатка.

Django вирізняється своєю гнучкістю та розширюваністю, що робить його адаптивним рішенням для широкого спектру веб-проєктів. Фреймворк має модульну архітектуру, яка дозволяє розробникам використовувати лише ті компоненти, які їм потрібні, уникаючи зайвого коду та залежностей [11].

Фреймворк також дуже гнучкий у дизайні URL-адрес. Система маршрутизації Django (URLconf) дозволяє розробникам створювати адреси сторінок, які не пов'язані з базовим кодом Python, таким як назви функцій. Це дозволяє одній і тій же програмі мати різні URL-адреси в різних контекстах (наприклад, /stories/ проти /news/), що є важливим для зручності використання, а також, роблячи створення «красивих» URL-адрес легшим, ніж «некрасивих».

Система шаблонів Django також демонструє високу розширюваність. Вона дозволяє авторам шаблонів розширювати її функціональність за допомогою власних тегів та фільтрів. Хоча система шаблонів фокусується на відокремленні логіки від представлення та запобіганні виконанню довільного коду Python для безпеки, вона також надає механізми для додавання спеціалізованої логіки [12].

2.3 Розробка функціональної схеми об’єкта програмування

Основною цільовою аудиторією додатка є вчителі та викладачі хімії закладів освіти. Для задоволення їх потреб необхідно якомога чіткіше визначити вміст сайту, відповідно до чого створюємо структуру сайту. Це важливо, оскільки користувачі повинні досить швидко та без труднощів знаходити потрібні сторінки. Тому проаналізовано можливий шлях користувачів по сторінках сайту та розроблено навігацію так, щоб вона була інтуїтивно зрозуміла кожному відвідувачу [13].

Загальну структуру сайту можна зобразити схемою (рис. 2.1)

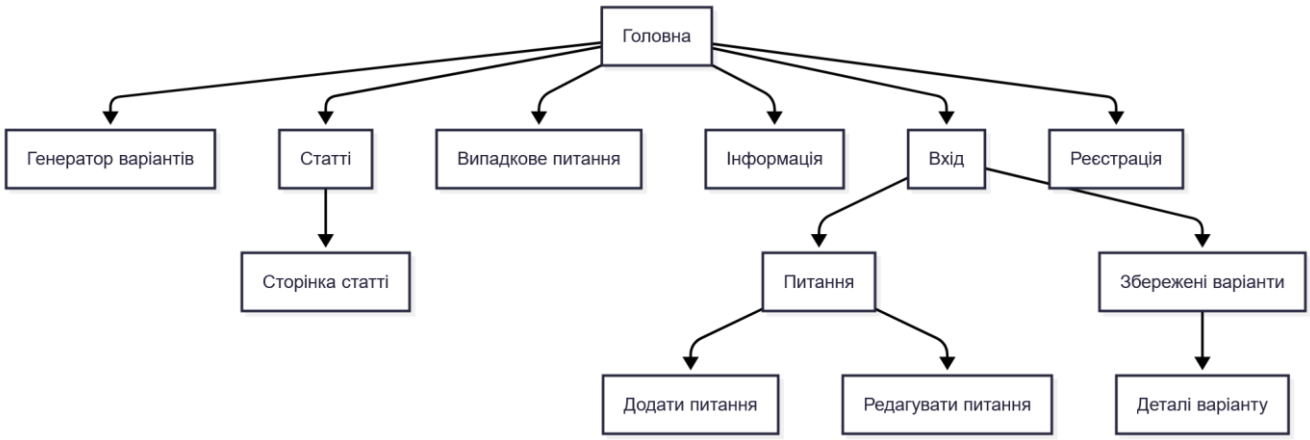

Рисунок 2.1 – Схема сайту

Головна сторінка (рис. 2.2), знайомить користувача з можливостями системи. Тут відображена коротка інформація про додаток, опис його переваг при використанні вчителями та викладачами, інструкція з використання додатка

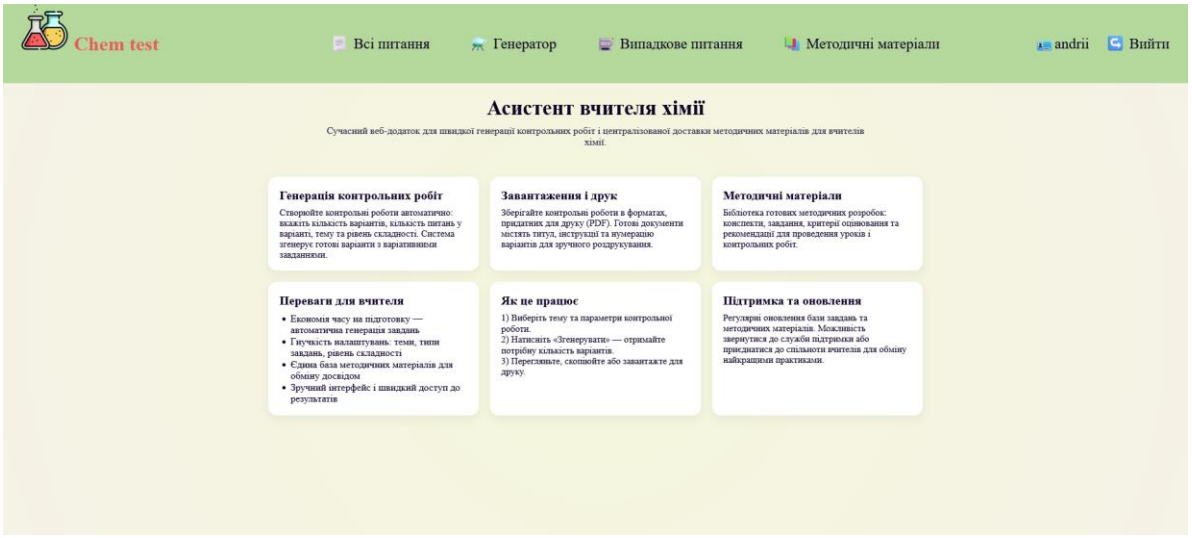

Рисунок 2.2 – Головна сторінка сайту

Сторінка перегляду питань (рис. 2.3), призначена для ознайомлення з базою питань веб-додатка. Є можливість відфільтрувати питання за певною темою. Ця сторінка доступна тільки зареєстрованим користувачам, це зроблено, для захисту бази питань від копіювання. Також на цій сторінці можна додати до бази даних власні завдання та відредагувати існуючі, якщо знайдена помилка.

Рисунок 2.3 – Сторінка перегляду питань

Сторінка додавання та модифікації питання (рис. 2.4), призначена для додавання нових питань до бази або модифікації існуючого питання. Доступна тільки зареєстрованим користувачам. Питання можна додати після заповнення форми і введення тексту питання, категорії, рівня складності та відповіді на завдання.

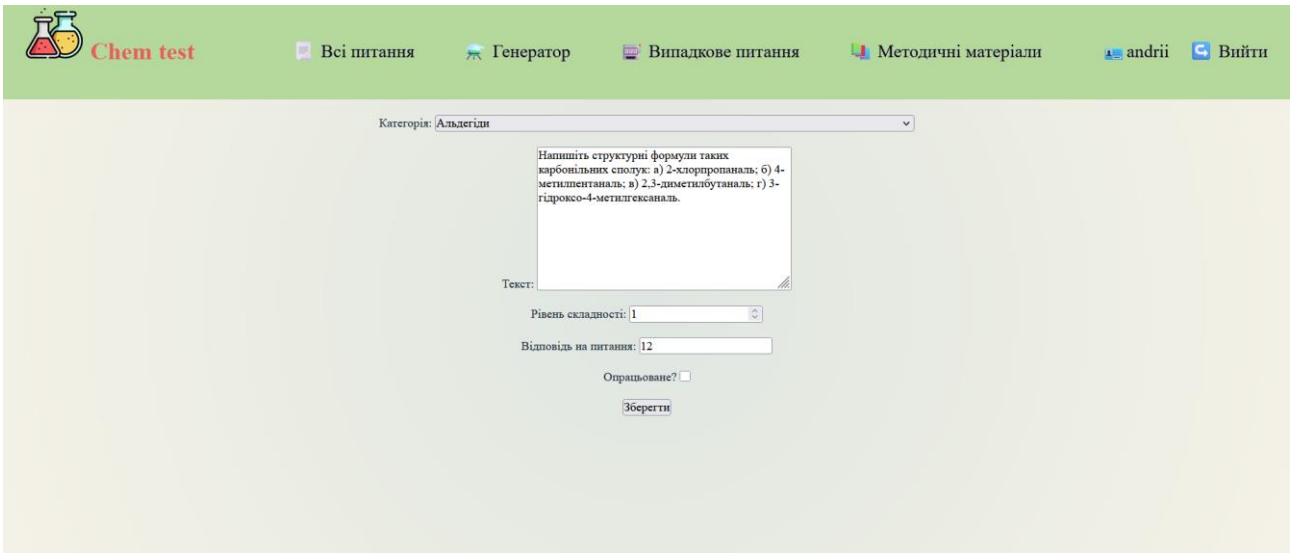

Рисунок 2.4 – Сторінка додавання та редагування питання

Генератор (рис. 2.5) – сторінка, на якій реалізовано основний функціонал сайту. Вона призначена для напівавтоматичного формування варіантів контрольних робіт. Це реалізується шляхом вибору параметрів генерації (кількість варіантів, питань, категорії), після цього формуються варіанти із випадковим набором питань.

Після відображення результату генерації в користувачів з’являється можливість копіювання тексту варіантів до буфера обміну, експорт контрольної роботи у форматі PDF. Для авторизованих користувачів також доступна функція збереження згенерованої роботи для подальшого використання (в майбутньому планується реалізація функції заміни питань в згенерованих варіантах).

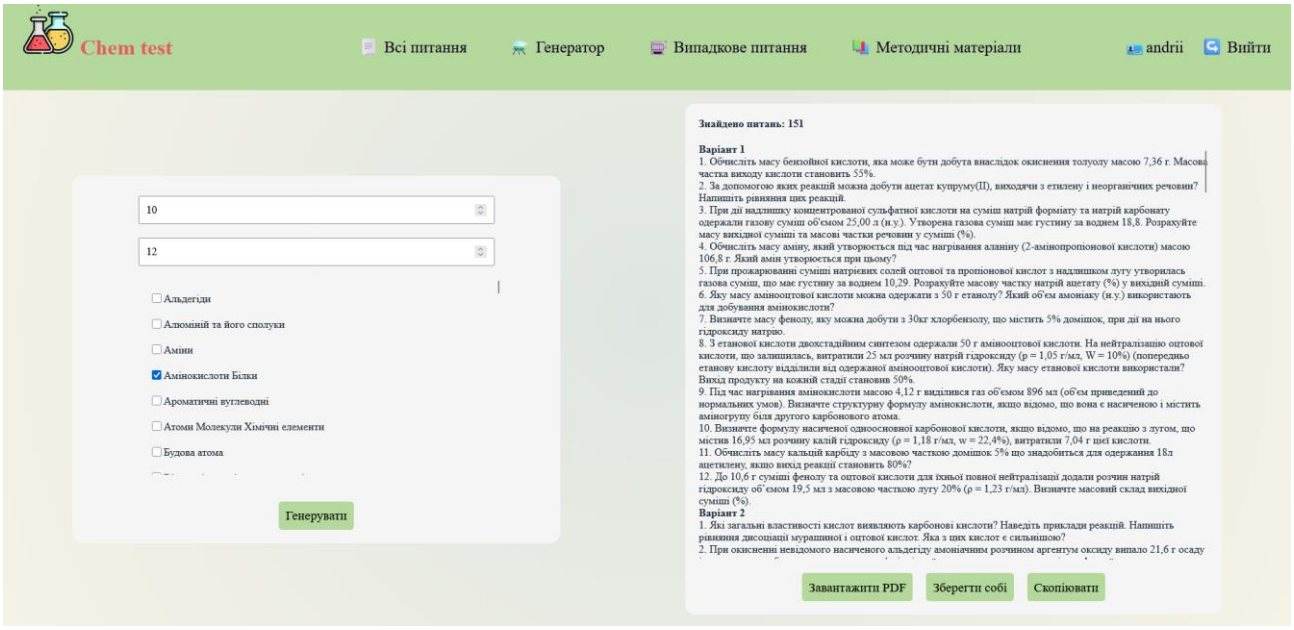

Рисунок 2.5 – Сторінка генератора

На Сторінці збережених контрольних робіт (рис. 2.6) можна переглянути збережені варіанти згенерованих робіт. Ця сторінка доступна тільки авторизованим користувачам. Для кожної збереженої роботи відображається її назва та дата створення. Після натискання на назву контрольної користувач переходить до детального перегляду конкретного варіанту (рис. 2.7), на цій сторінці також доступні кнопки експорту до PDF та копіювання до буфера обміну.

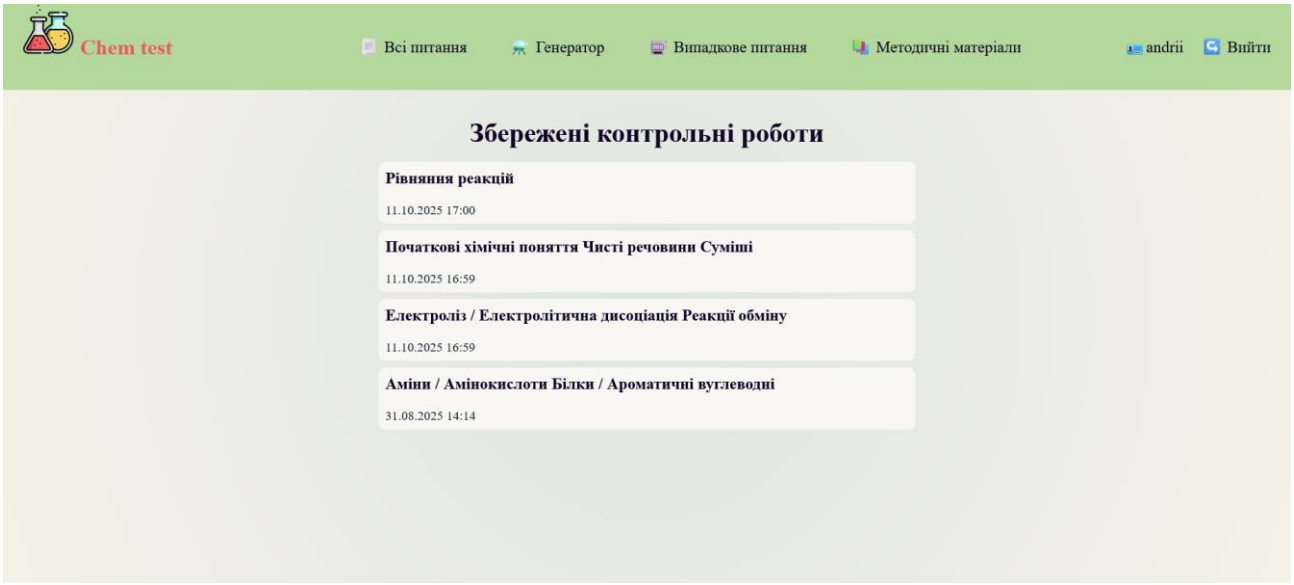

Рисунок 2.6 – Сторінка збережених контрольних робіт користувача

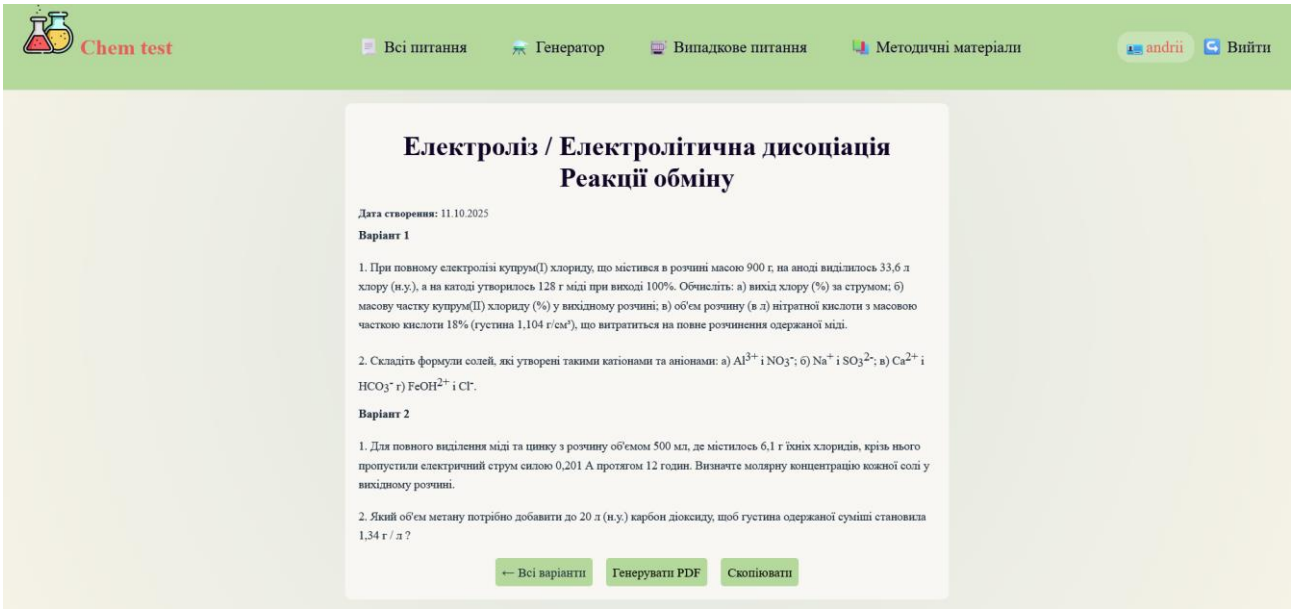

Рисунок 2.7 – Деталі збереженої контрольної роботи

Сторінку випадкового питання можна використовувати для самоперевірки знань користувача. При переході на сторінку відображається випадкове питання і виводиться форма для введення відповіді. Після натискання на кнопку «відправити», система перевіряє чи дана відповідь правильна і виводить на екран користувача.

Сторінка методичних матеріалів (рис. 2.8, рис.2.9) це список навчальних, модельних програм, календарних планів та інших матеріалів, які стосуються методики викладання хімії. Всі статті відсортовані за датою додавання.

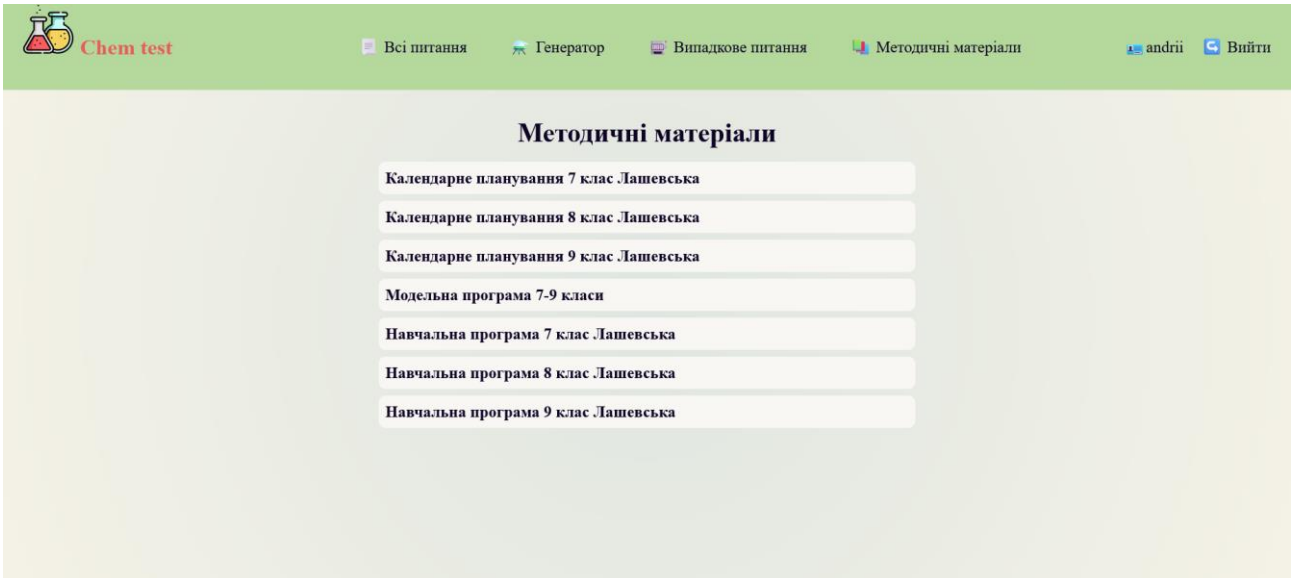

Рисунок 2.8 – Сторінка методичних матеріалів

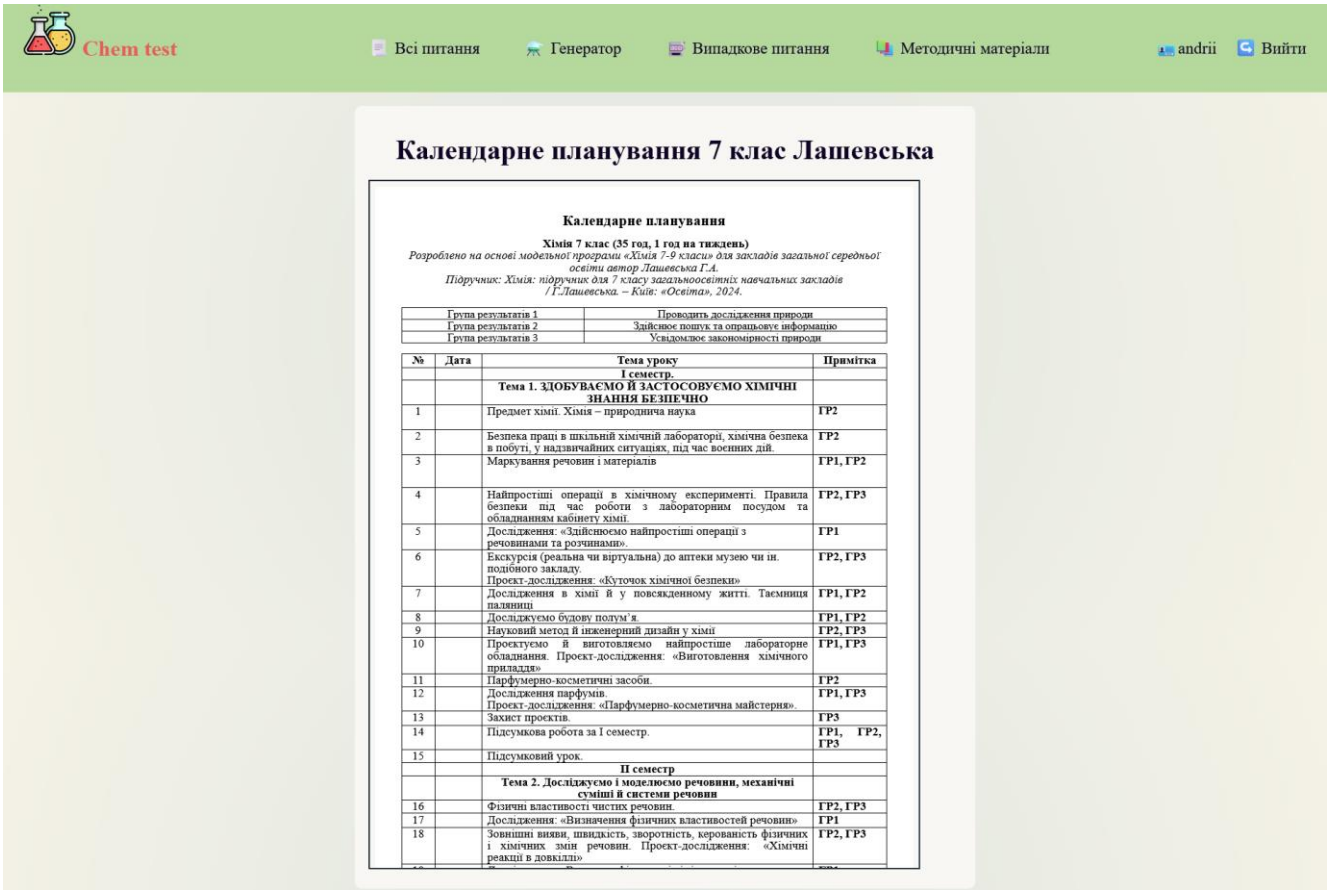

Рисунок 2.9 – Деталі методичного матеріалу

Сторінка реєстрації та входу це ключовий елемент повноцінного сайту. Система аутентифікації користувачів управляє акаунтами користувачів, групами, правами і призначеними для користувача сесіями. Для авторизації необхідно перевірити, хто є цей користувач (зазвичай, за допомогою перевірки імені та пароля по базі даних користувачів) та перевірити, що користувач авторизований для виконання певних дій. Обов'язкові поля: логін, пошта, пароль двічі

Для зберігання та впорядкування інформації, що використовуватиметься в процесі функціонування веб-ресурсу потрібна база даних.

База веб-додатка «Chem test» включає наступні таблиці (таблиця 2.2):

- Users - зареєстровані користувачі;
- Category - категорії питань;
- Question - питання;
- SavedVariant - збережені варіанти;
- Article - статті (методичні матеріали)

Таблиця 2.2 – Структура бази даних

Модель	Поле	Тип даних	Опис
User	id	Integer (PK)	Первинний ключ
	password	varchar	Пароль
	username	CharField	Ім’я користувача
	email	CharField	Пошта
	is_staff	BooleanField	Чи адміністратор
	date_joined	DateTimeField	Дата реєстрації
Category	id	Integer (PK)	Первинний ключ
	category_name	CharField	Назва категорії
Question	id	Integer (PK)	Первинний ключ
	category	ForeignKey	Категорія питання
	question_text	TextField	Текст питання
	answer	TextField	Відповідь
	difficulty	IntegerField	Рівень складності
	is_checked	BooleanField	Статус
	source	CharField	Джерело
Article	id	Integer (PK)	Первинний ключ
	title	CharField	Заголовок статті
	type	CharField	Тип статті
	content	TextField	Зміст статті
SavedVariant	id	Integer (PK)	Первинний ключ
	user	ForeignKey	Користувач
	name	CharField	Назва варіанту
	created_at	DateTimeField	Дата створення
	variant_data	JSONField	Дані варіанту

2.4 Опис процесу розробки додатка

Початкова генерація файлів проекту виконується командою з лістингу 2.1.

Лістинг 2.1 – Створення нового проекту Django

```
django-admin.py startproject chem_test
```

Кінець лістингу 2.1

В результаті виконання в поточному каталозі буде створено теку chem_test, в якому буде розміщено manage.py – скрипт для управління Django-проектом і каталог chem_test.

Проект – це набір додатків, які повністю реалізують роботу сайту. Для початку роботи з Django потрібно створити додаток для кожного модуля, який буде працювати на сайті.

Перебуваючи в тому ж каталозі, що і файл manage.py створимо додаток, який буде відповідати за роботу основної логіки сайту з назвою chemistry. Для цього введемо команду, наведену в лістингу 2.2.

Лістинг 2.2 – Створення нового додатка Django

```
python manage.py startapp chemistry
```

Кінець лістингу 2.2

В даному проекті ми використовуємо СУБД MySQL, тому необхідно здійснити підключення Django до сервера бази даних. Щоб це зробити, потрібно відредагувати блок «DATABASES» у файлі chemistry/settings.py, вказавши параметри і шлях до файлу бази даних. Налаштування бази даних додатка «Chem test» наведені у лістингу 2.3.

Лістинг 2.3 – Налаштування параметрів бази даних

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': os.getenv("DB_NAME"),
        'USER': os.getenv("DB_USER"),
        'PASSWORD': os.getenv("DB_PASSWORD"),
        'HOST': os.getenv("DB_HOST"),
        'PORT': os.getenv("DB_PORT"),
        'OPTIONS': {
            'init_command': "SET
sql_mode='STRICT_TRANS_TABLES'",
        }
    }
}
```

Кінець лістингу 2.3

Можна вписати облікові дані бази даних, безпосередньо в файл settings.py, але така практика створює ризики безпеці сайту, а також, в подальшому, може створити труднощі в адмініструванні. Тому, всі параметри додатка будуть зберігатися у файлі .env(змінні середовища виконання), який призначений для зручного керування паролями і конфіденційними даними додатка.

Після налаштування бази даних, потрібно зареєструвати додаток в налаштуваннях проекту. Для цього відкрити chem_test/settings.py і в блоці «INSTALLED_APPS» додати 'chemistry'.

Далі створюються моделі, які визначають структуру бази даних та додаткові метадані.

Модель – це основний елемент, що описує дані, які зберігаються у додатку. Вона містить набір полів і логіку поведінки цих даних. Django дотримується принципу DRY (Don’t Repeat Yourself), тобто кожен елемент інформації повинен бути визначений лише один раз.

Кожну модель ми представляємо у вигляді класу, який успадковує від стандартної моделі django. Приклад коду моделі категорії наведено у лістингу 2.4.

Лістинг 2.4 – Модель категорії

```
class Category(models.Model):
    category_name = models.CharField(max_length=100)
        def __str__(self):
        return self.category_name
```

Кінець лістингу 2.4

Кожне поле моделі задається окремим екземпляром класу Field, який визначає тип даних, що зберігатиметься у базі. Наприклад, CharField використовується для текстових значень, а DateTimeField – для зберігання дати та часу.

Назви полів (наприклад, name) подаються у зручному для машин форматі (machine-friendly) й використовуються у програмному коді, тоді як у базі даних вони виступають назвами відповідних колонок.

Зі зразками інших моделей можна ознайомитись в додатку А.

Після створення всіх потрібних моделей, для створення відповідних таблиць в базі даних, потрібно створити та застосувати міграції та створити супер користувача, який зможе керувати всіма даними через веб-інтерфейс вбудованої панелі адміністратора (лістинг 2.5).

Лістинг 2.5 – Створення та застосування міграцій

```
python manage.py makemigrations
python manage.py migrate
python manage.py createsuperuser
```

Кінець лістингу 2.5

Наступний крок – додати посилання на «chemistry.urls» у головній конфігурації url-адрес. Для цього у файлі «chem_test/urls.py» потрібно імпортувати

include з django.conf.urls і додати include() до списку urlpatterns.

Отримаємо такий код, який наведено у лістингу 2.6.

Лістинг 2.6 – Додавання маршрутів

```
from django.conf.urls import patterns, include, url
from django.contrib import admin
urlpatterns = patterns('',
    url('admin/', include(admin.site.urls)),
    url('', include('chemistry.urls')),
)
```

Кінець лістингу 2.6

Таким чином реалізується зв’язок між головною конфігурацією проєкту і маршрутами конкретного додатка.

Далі потрібно створити логіку додатка, за допомогою файлу views.py. Він виконує ключову роль – саме він визначає, що відображатиме користувачеві сервер, коли той відкриває певну сторінку.

Простіше кажучи, view – виконує функцію «мозку» веб-додатку, який:

- отримує запит (request) від користувача,
- обробляє дані (може звертатися до бази даних, моделей, форм тощо),
- повертає відповідь (response) – зазвичай HTML-сторінку, JSON-дані або перенаправлення.

Для кожної сторінки сайту потрібно створити відповідну функцію у views та додати маршрут до неї у urls.py додатка.

Приклад функції відображення головної сторінки, наведено у лістингу 2.7. На головній сторінці не відбувається обробка даних, тому ця функція лише повертає шаблон з розміткою та стилями.

Лістинг 2.7 – Відображення головної сторінки

```
def index(request):
    #головна сторінка
    return render(request, 'chemistry/index.html')
```

Кінець лістингу 2.7

Після отримання запиту від користувача django звертається до urls.py та визначає, яка фунція відповідає за обробку даного запиту і передає їй об’єкт запиту request, view обробляє дані, готує контекст для шаблону та формує відповідь у вигляді HTML-сторінки.

Для кожної задачі, яку виконує додаток також має бути створена відповідна функція, яка буде обробляти вхідні дані(request) та відправляти їх відповідному view для відображення користувачеві(response).

Функція генерації контрольних (лістинг 2.8), відповідає за головну функцію додатка - генерацію контрольної роботи. Разом із запитом функція отримує кількість варіантів, кількість питань та id категорій для генерації. Після цього, вибирає з бази даних, всі питання з потрібних категорій та перевіряє, чи вистачить питань для створення роботи. Всі питання перемішуються, і для кожного варіанту вибирається потрібна кількість завдань, при цьому, вже обрані завдання видаляються із загальної вибірки.

Готова контрольна робота, у вигляді кортежу, передається з відповіддю для відображення на сторінці.

Лістинг 2.8 – Функція генерації контрольної роботи

```
def generate(numOfVariants, numOfQuestions, multicategory):
    all_questions = []  # Список всіх питань у категоріях
    for category_id in multicategory:
        try:
                # Отримуємо об'єкт Category за id
            category = Category.objects.get(id=category_id)
            category_questions = list(Question.objects.
                        filter(category=category).values_list(
                    'text', flat=True))
            all_questions.extend(category_questions)
        except Category.DoesNotExist:
            continue  # Пропускаємо, якщо категорія не знайдена

    if len(all_questions) < numOfQuestions * numOfVariants:
        return {1: ['<span class="material-symbols-rounded">
        warning </span>','Недостатня кількість запитань з даної
                         теми в базі, оберіть менше питань або
                         варіантів']}
```

```
    # Перемішуємо всі питання для випадкового вибору
    random.shuffle(all_questions)
    generated_q = {}

    for variant_num in range(1, numOfVariants + 1):
        # Вибираємо перші N питань
        variant_questions = all_questions[:numOfQuestions]
        # Видаляємо вже вибрані питання
        all_questions = all_questions[numOfQuestions:]
        generated_q[variant_num] = [f'{i + 1}. {q}' for i, q
                                in enumerate(variant_questions)]

    return generated_q
```

Кінець лістингу 2.8

Функція відображення сторінки генератора наведена в лістингу 2.9.

Лістинг 2.9 – Відображення сторінки генератора

```
def generator(request):
    total_questions = 0
    # Якщо потрібно створити PDF
    if request.method == "POST" and 'generate_pdf'
                          in request.POST:
        variants = request.session.get('generated_variants',{})
        if variants:
            pdf_buffer = make_pdf(variants)
            return FileResponse(pdf_buffer, as_attachment=True,
                                filename="variants.pdf")
    generated_q = []
    if request.method == "POST":
        form = VariantCreationForm(request.POST) #or None
        if form.is_valid():
           numOfVariants = int(request.POST.get('numOfV',''))
           numOfQuestions = int(request.POST.get('numOfQ',''))
           multicategory = request.POST.getlist('category')
           request.session['last_categories'] = multicategory
           generated_q = generate(numOfVariants,
                                  numOfQuestions,multicategory)
            total_questions = num_of_questions(multicategory)
           # Зберігаємо згенеровані варіанти в сесії
            request.session['generated_variants'] = generated_q
    else:
        form = VariantCreationForm()
    context = {'generated_q': generated_q,
               'form': form,
               'total_questions': total_questions}
    return render(request, 'chemistry/generator.html', context)
```

Кінець лістингу 2.9

З прикладами інших функцій сайту можна ознайомитись в додатку Б.

У фреймворку Django шаблони (templates) використовуються для формування інтерфейсу користувача. Вони відповідають за візуальне представлення даних, які надходять із рівня контролера (views.py), та дозволяють відокремити логіку від представлення – реалізуючи архітектурний принцип Model–View–Template (MVT).

Для їх створення, в додатку chemistry необхідно додати папку templates, а в ній – chemistry. В цій директорії Django буде шукати заготовки сторінок, які потрібно відобразити і наповнюватиме їх отриманими від view даними.

Шаблони використовують Django Template Language (DTL), вона дозволяє:

- вставляти динамічні дані за допомогою подвійних дужок {{ змінна}};
- виконувати логічні операції за допомогою тегів {% ... %};
- розширювати інші шаблони через механізм наслідування.

Для забезпечення єдиного стилю інтерфейсу застосовано механізм наслідування шаблонів, що є ключовою особливістю Django Template Language.

Базовий шаблон base.html містить загальні елементи сторінки – заголовок, підключення стилів та основні блоки, які використовуються у всіх шаблонах.

Приклад базового шаблону наведено у лістингу 2.10.

Лістинг 2.10 – Базовий шаблон сторінок

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>{% block title %}
               Асистент викладача хімії
           {% endblock %}
     </title>
    <link rel="stylesheet" href="{% static 'css/style.css' %}">
</head>
<body>
    {% include 'navbar.html' %}
    <main>
        {% block content %}{% endblock %}
```

```
    </main>
</body>
</html>
```

Кінець лістингу 2.10

2.5 Публікація веб-додатка

Для того, щоб додатком могли користуватись не тільки локальні користувачі, а й користувачі зі всього світу, потрібно його опублікувати, тобто розмістити на зовнішньому сервері. Публікація веб-додатків, створених на фреймворку Django, може бути реалізована кількома способами залежно від цілей, бюджету, масштабів проєкту та вимог до продуктивності (таблиця 2.3).

Класичним методом публікації, який передбачає ручне налаштування серверного середовища, є розгортання на традиційному VPS або виділеному сервері. На віртуальному сервері встановлюються необхідні компоненти: веб сервер (Apache або Nginx), середовище Python, модуль Gunicorn або uWSGI, система керування базами даних (MySQL, PostgreSQL тощо). Основна перевага цього методу – повний контроль над конфігурацією сервера, безпекою та масштабуванням. Недоліком є складність налаштування та необхідність адміністрування.

Хмарні сервіси (PaaS) типу Heroku, PythonAnywhere, Render, Railway або Google Cloud Run дозволяють автоматизувати процес розгортання. Розробник завантажує код застосунку у репозиторій (наприклад, GitHub), після чого платформа автоматично виконує побудову середовища, встановлення залежностей та запуск додатка. Цей метод є найбільш зручним для освітніх, навчальних і невеликих проєктів, оскільки не потребує глибоких знань у сфері системного адміністрування.

Використання контейнеризації, зокрема Docker, дозволяє створити універсальне середовище для запуску Django-додатка незалежно від операційної системи. Застосунок, база даних та інші служби розміщуються у контейнерах.

Такий підхід спрощує розгортання, забезпечує відтворюваність середовища та легке масштабування, але недоліком є необхідність придбання VPS хостингу та його адміністрування.

Таблиця 2.3 – Порівняльна характеристика методів розгортання

Метод	**Переваги**	**Недоліки**	**Рекомендоване застосування**
VPS/виділений сервер	Повний контроль, гнучкість налаштувань	Потребує адміністрування	Великі проекти
Хмарні платформи (Heroku, PythonAnywhere)	Простота, швидке налаштування	Обмеження безкоштовних планів	Навчальні та малі проєкти
Docker-контейнери	Універсальність, масштабованість	Потребує знань Docker	Середні та великі проєкти

Для навчальних або невеликих проєктів доцільним є використання сервісу PythonAnywhere, який надає готове середовище для розгортання Python-додатків. Тому, саме цей сервіс було обрано для публікації додатка ChemTest.

Щоб опублікувати свій додаток, потрібно виконати таку послідовність дій:

1. Створити обліковий запис на платформі, придумавши логін, пароль та вказавши електронну пошту. Після реєстрації та входу потрапляємо до панелі керування, зразок інтерфейсу наведено на рисунку 2.10

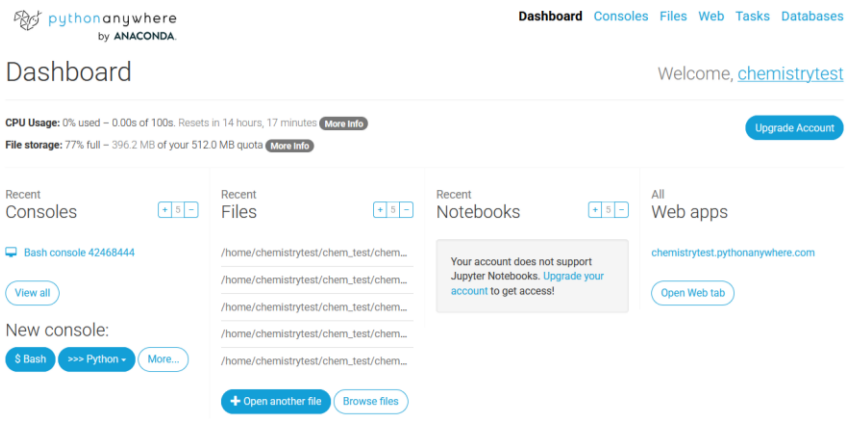

Рисунок 2.10 – Панель керування Pythonanywhere

Переходимо на вкладку «Files» та починаємо завантаження всіх файлів локально створеного проєкту.

2. Ключем до стабільної роботи є ізоляція проекту у власному середовищі, для цього у Bash Console (рис. 2.11) створюємо віртуальне середовище. Після створення воно активується автоматично. Переходимо до папки проєкту і встановлюємо усі бібліотеки з файлу requirements.txt (лістинг 2.11).

Лістинг 2.11 – Створення середовища і встановлення залежностей

```
mkvirtualenv --python=/usr/bin/python3.10 virtualenv
pip install -r requirements.txt
```

Кінець лістингу 2.11

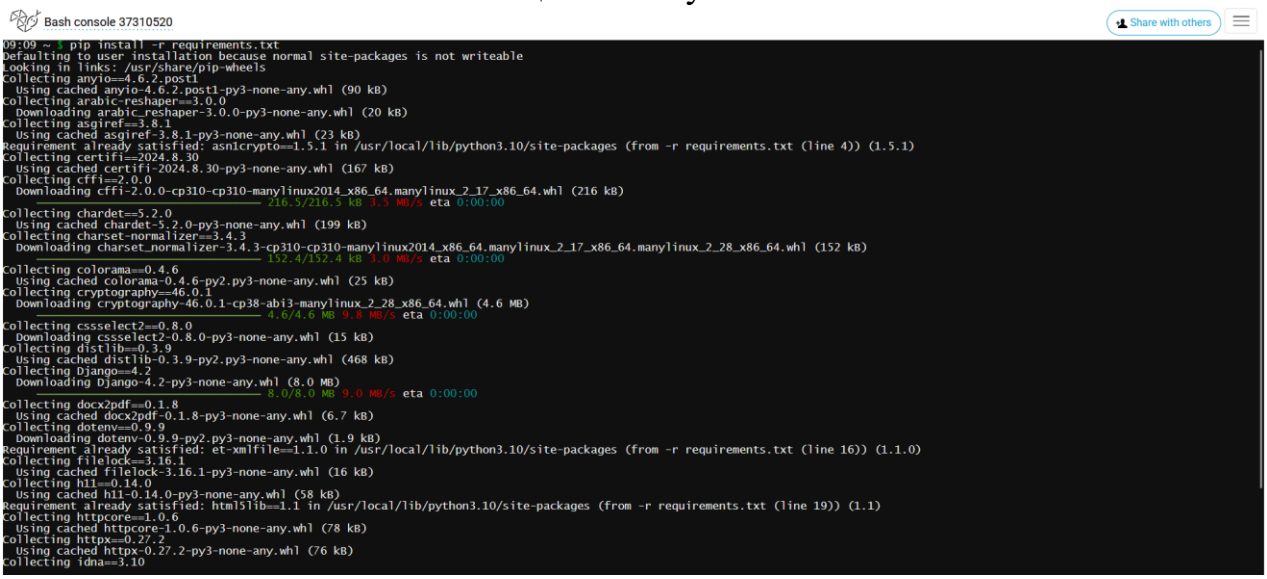

Рисунок 2.11 – Консоль

3. Налаштування веб-додатка через панель керування - це головний етап, де ми активуємо сайт.

Перейшовши на вкладку «Web» (рис. 2.12) у головному меню, потрібно натиснути «Add a new web app». Після вибору доменного імені (chemistrytest.pythonanywhere.com), обираємо фреймворк Django.

У розділі «Code» треба вказати шлях до папки з проєктом (/home/chemistrytest/chem_test). Там же потрібно відредагувати рядок «WSGI configuration file», вказавши правильний шлях до проєкту.

У розділі «Virtualenv» вказуємо шлях до віртуального середовища (/home/chemistrytest/.virtualenvs/chem_test-virtualenv)

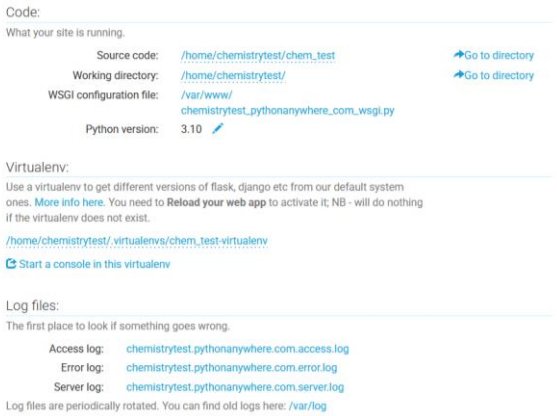

Рисунок 2.12 – Конфігурація додатка

Переходимо на вкладку «Databases» (рис. 2.13) у головному меню. PythonAnywhere автоматично надає ім'я хоста, ім'я користувача та порти. Створюємо нову базу даних, ввівши назву та натиснувши кнопку «Create» у вкладці «Create database» Після цього у вкладці «MySQL password» створюємо надійний пароль бази даних. Ім'я хоста, користувача та пароль потрібно внести у файл конфігурації django (settings.py).

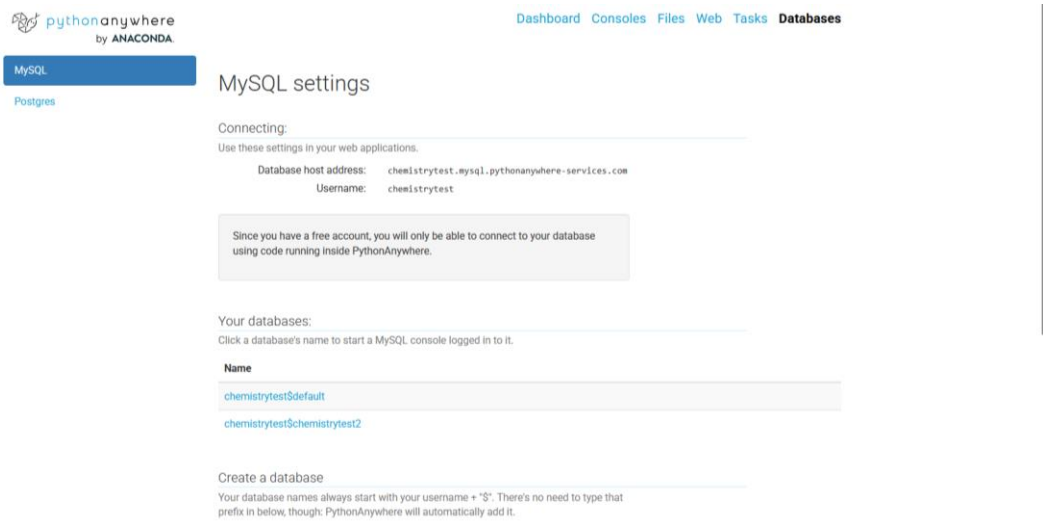

Рисунок 2.13 – Налаштування бази даних

Оскільки ми використовуємо змінні середовища, у головній папці проєкту (home/chemistrytest) потрібно створити файл «.env» і записати ці дані до нього (рис. 2.14).

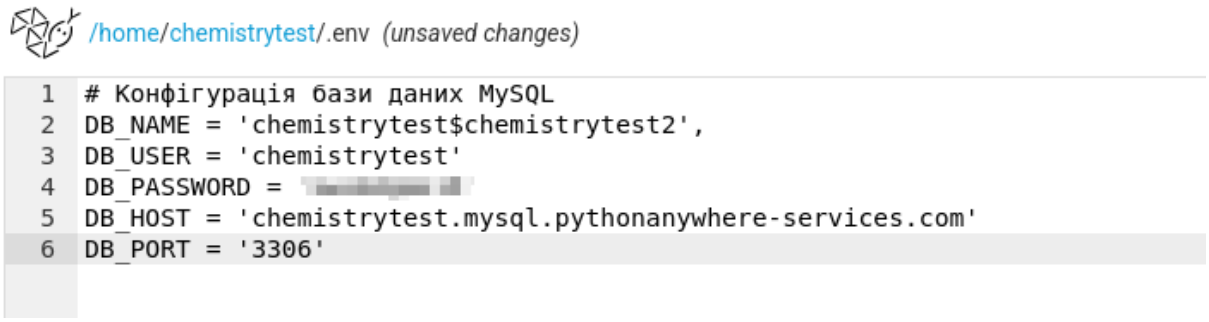

Рисунок 2.14 – Файл .env

А тоді, у файлі settings.py, в розділі «Databases» вносимо посилання на змінні середовища (рис. 2.15).

Рисунок 2.15 – Файл settings.py

Коли оновлення налаштувань проєкту завершено, переходимо до Bash Console, активуємо віртуальне середовище та виконуємо команди міграції, для створення таблиць у базі даних MySQL і для збору статичних файлів (лістинг 2.12). Статичні файли використовуються для створення вигляду сайту, але не змінюються динамічно, наприклад шрифти, файли стилів. Кожного разу при заміні будь якого з цих файлів, потрібно виконати команду збору статики.

Лістинг 2.12 – Команди міграції та збору статичних файлів

```
python manage.py makemigrations
python manage.py migrate
python manage.py collectstatic
```

Кінець лістингу 2.12

4. Після того, як все налаштовано і виконані міграції. Потрібно повернутись до вкладки «Web» і натиснути кнопку «Reload», щоб сервер перезавантажився з новими налаштуваннями.

Після виконання всіх наведених дій, можна відкрити посилання на додаток у браузері та почати наповнювати базу даних питаннями.

РОЗДІЛ 3
МЕТОДИКА ПРОВЕДЕННЯ ЕКСПЕРИМЕНТУ

3.1 Методика оцінювання ефективності

Оцінити ефективність веб-додатка можна за допомогою кількісних та якісних методів, також важливо враховувати як технічні, так і користувацькі критерії. Основними елементами, на які варто звернути увагу при оцінці можна виділити:

1. Кількісні метрики (об'єктивні дані):
   - час на створення контрольної роботи – порівняти, скільки часу потрібно на складання тесту вручну та за допомогою додатка;
   - кількість використань – скільки викладачів/студентів користуються додатком за певний період;
   - точність згенерованих завдань – порівняти кількість виправлених помилок у контрольних, створених вручну та автоматично;
   - рівень успішності учнів – чи впливає використання тестів із додатка на результати учнів (можна порівняти середній бал до і після впровадження);
   - частота повторного використання – якщо користувачі повертаються до додатка, це свідчить про його корисність.
2. Якісні метрики (відгуки та сприйняття):
   - анкетування викладачів – опитати, наскільки вони задоволені функціоналом, дизайном та зручністю;
   - якість сформованих контрольних – оцінити, наскільки завдання відповідають навчальним програмам;
   - частота технічних проблем – наскільки стабільно працює додаток.
3. Порівняння з альтернативами. Оцінити, як додаток співвідноситься з аналогами (Google Forms, Moodle, інші генератори тестів).

Врахування оцінок за цими параметрами дасть можливість зрозуміти, що варто зробити для поліпшення досвіду користувачів і досягнення цілей (наприклад, збільшення кількості нових користувачів або скорочення витрат). Постійне відслідковування і вдосконалення цих аспектів дозволяє адаптувати веб-додаток до змінних умов, покращуючи його цінність для кінцевих користувачів.

3.2 Методика проведення експерименту з користувачами «Chem Test»

Експериментальна робота полягала у оцінюванні ефективності додатка та у виявленні основних напрямів подальшого розвитку.

Під час експерименту відбулись 3 етапи роботи, констатувальний, впроваджувальний та контрольно-узагальнювальний.

Констатувальний етап полягав у визначенні початкового стану володіння педагогами ІКТ-компетентностями та наявних способів підготовки контрольних робіт. Було проведено анкетування під час якого з’ясували, які засоби викладачі використовують для створення завдань (Word, Excel, ручна підготовка тощо), з якими труднощами стикаються у процесі створення контрольних робіт, оцінено середню кількість часу, необхідного для створення однієї контрольної роботи та сформовано групу педагогів, які виявили бажання протестувати додаток.

Під час впроваджувального етапу, відбулась презентація функціональності додатка. Учасникам був продемонстрований алгоритм створення тестів, базу готових завдань та можливості експорту матеріалів. Після цього, надано можливість створити власні контрольні роботи за допомогою додатка.

Контрольно-узагальнювальний етап полягав у оцінці впливу використання веб-додатка «Chem test» на професійну діяльність педагогів, аналізі ефективності, зручності та доцільності його впровадження у робоче середовище викладачів.

Дані для проведення експерименту збирались шляхом анкетування педагогів в Google формах за критеріями наведеними в таблиці 3.1

Таблиця 3.1 – Критерії оцінювання додатків

Критерій	**Показник**
Ефективність	Порівняння часу, витраченого на створення контрольної роботи в «Chem Test» проти звичних методів.
Зручність	Оцінка інтуїтивності інтерфейсу. Чи легко викладачам було знайти потрібні функції без підказок?
Функціональність	Наскільки коректно відображаються хімічні формули? Чи достатньо наявних шаблонів завдань?
Задоволеність	Суб'єктивна оцінка користувачів – готовність рекомендувати додаток колегам.

Анкети містили три типи запитань:

- закриті запитання (з вибором варіантів) – для статистичного аналізу;
- шкали Лайкерта (від 1 до 5) – для оцінки ступеня згоди з твердженнями про зручність додатка;
- відкриті запитання – для отримання розгорнутих відгуків, пропозицій щодо покращення функціоналу та опису знайдених багів.

Очікувані результати експерименту:

- зростання рівня цифрової компетентності педагогів;
- скорочення часу на підготовку контрольних робіт;
- підвищення якості та різноманітності контрольних робіт;
- формування позитивного ставлення педагогів до використання ІТ-засобів у професійній діяльності;
- виявлення напрямів для подальшого удосконалення додатка.

РОЗДІЛ 4
ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА, ДОСЛІДЖЕННЯ ТА ОБРОБКА, АНАЛІЗ І СПІВСТАВЛЕННЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1 Зміст та організація експериментального дослідження

З метою дослідження ефективності веб-додатку «Chem test» та визначення напрямів його подальшого вдосконалення було проведено анкетування серед педагогів і викладачів хімії закладів загальної середньої та фахової передвищої освіти Горохівської територіальної громади.

Анкетування є одним із найбільш поширених методів збору емпіричних даних, що дозволяє отримати структуровану інформацію про ставлення, досвід і пропозиції респондентів щодо використання цифрових освітніх ресурсів. Основна мета анкетування полягала у виявленні рівня зацікавленості педагогів у застосуванні веб-технологій у навчанні хімії, а також у оцінці зручності, функціональності та практичної користі створеного додатка.

В рамках організації дослідження було передбачено кілька етапів:

1. Розроблення структури анкети. Анкета складалася з чотирьох частин:

Ознайомлювальна: виявлення загального рівня обізнаності з цифровими освітніми технологіями та веб-додатками для підтримки викладання природничих дисциплін.

Аналітична: оцінка інтерфейсу, функцій, дизайну та зручності використання веб-додатку «Chem test».

Оцінювально-рефлексивна: визначення рівня практичної користі додатку, готовності викладачів використовувати його у своїй роботі, а також збирання пропозицій щодо подальшого удосконалення.

Відгуки учнів: педагогам було запропоновано опитати учнів стосовно зрозумілості завдань, їх складності та справедливості розподілу завдань між варіантами.

2. Визначення вибірки. До анкетування було залучено викладачів хімії закладів середньої освіти, коледжів і ліцеїв, які мають досвід використання інформаційно-комунікаційних технологій у навчальному процесі. Вибіркова сукупність становила 15 респондентів, що забезпечило достатній рівень достовірності отриманих результатів.

3. Розповсюдження анкети. Опитування проводилося в електронній формі за допомогою Google Форм, що дало змогу охопити ширше коло педагогів, незалежно від їхнього місця роботи чи проживання. Анкету було розповсюджено серед педагогів Горохівської міської територіальної громади, крім того, після генерації контрольної роботи, кожному користувачу надавалось посилання для проходження опитування.

Усі відповіді були анонімними, що дало можливість педагогам вільно висловлювати власні думки, зауваження та пропозиції. Респондентам було гарантовано конфіденційність отриманих даних, що підвищило достовірність результатів.

4. Аналіз результатів. Після завершення опитування результати було зведено у вигляді таблиць і діаграм для подальшої обробки та аналізу. Особлива увага приділялася питанням, пов’язаним із зручністю користувацького інтерфейсу, наявністю корисних інструментів для планування уроків, а також рівнем зацікавленості педагогів у використанні веб-додатку у повсякденній діяльності.

5. Інтерпретація та узагальнення. Отримані результати дозволили визначити позитивні сторони створеного додатку та окреслити напрями його вдосконалення – зокрема, щодо розширення бази дидактичних матеріалів, інтеграції з хмарними сервісами та створення додаткових аналітичних інструментів для викладача.

4.2 Обробка результатів

З метою оцінки ефективності, зручності та практичної цінності веб-додатка «Chem test» було проведено анкетування серед педагогів, які використовували розроблену систему у своїй діяльності. Опитування дало змогу визначити рівень задоволеності користувачів, окреслити сильні сторони додатка та виявити напрями для подальшого вдосконалення.

Було проаналізовано 14 відповідей викладачів щодо їхнього досвіду використання веб-додатку «Chem test». Респонденти позитивно оцінили свій досвід з додатком. Проте, результати опитування показали, що досвід користувачів можна зробити ще кращим. Серед основних напрямків подальшого розвитку та покращення додатку, користувачі відзначили функцію онлайн тестування, генерацію правильних відповідей на завдання, адаптивність до мобільних пристроїв та подальше розширення бази питань.

На запитання «Якими засобами зазвичай користуєтесь для оцінювання?» (рис. 4.1), більше 70% респондентів зазначили у відповіді, що використовують системи онлайн тестування (наурок, всеосвіта та ін.) близько 43% учителів відповіли, що використовують Google форми та PowerPoint, також 21% респондентів використовують для підготовки матеріалів MS Word.

Рисунок 4.1 – Цифрові інструменти

Більшість респондентів (57%) відзначили, що витрачають на створення контрольної роботи традиційними методами близько 1-2 годин. 29% потрібно від 30 хвилин до 1 години, 14% педагогів витрачають менше 30 хвилин (рис. 4.2).

Рисунок 4.2 – Час на створення роботи

Труднощі при створенні контрольних робіт виникають у 86% педагогів. (рис. 4.3) Зі складністю добору завдань стикаються 64% учасників. 43% викладачів визнають, що мають недостатньо часу, а 36% - важко створювати кілька варіантів завдань для контрольних робіт, ще 14% опитаних важко підбирати завдання, через їх однотипність.

Рисунок 4.3 – Труднощі при створенні робіт

На запитання «Чи вважаєте Ви доцільним використання спеціалізованого веб-додатка для автоматичної генерації контрольних робіт з хімії?»(рис. 4.4) ствердно відповіли 57% опитаних, ще 43% більше схиляються до ствердної відповіді ніж до негативної.

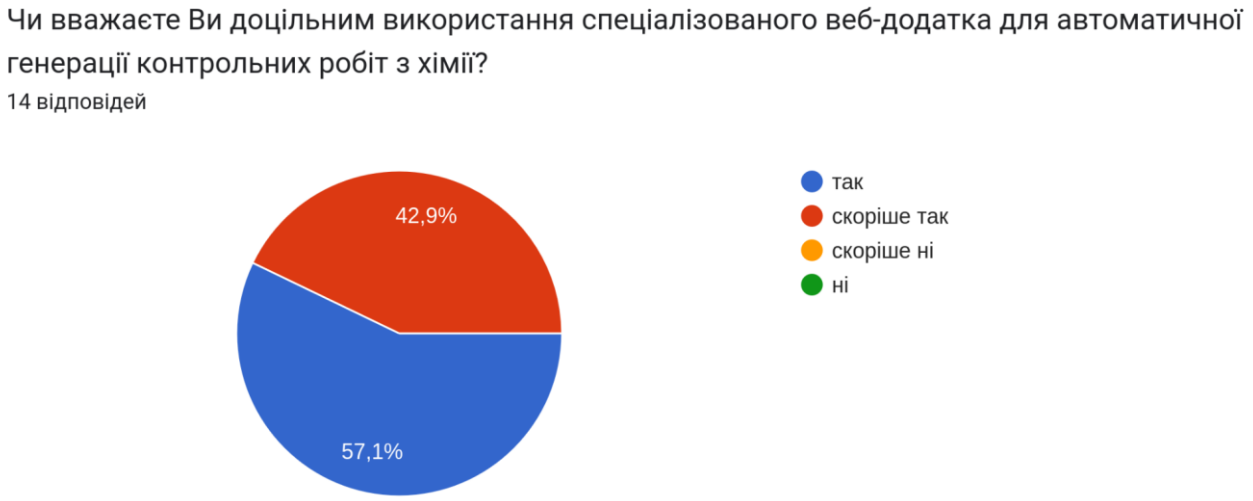

Рисунок 4.4 – Доцільність автоматизованих генераторів контрольних робіт

Більшість опитаних (92%) відзначили, що інтерфейс програми зручний, а процес роботи не викликає труднощів (рис. 4.5). Окремі користувачі вказали на незначні незручності, пов’язані з інтерфейсом або індивідуальними технічними особливостями пристроїв. Середня оцінка – 4.9.

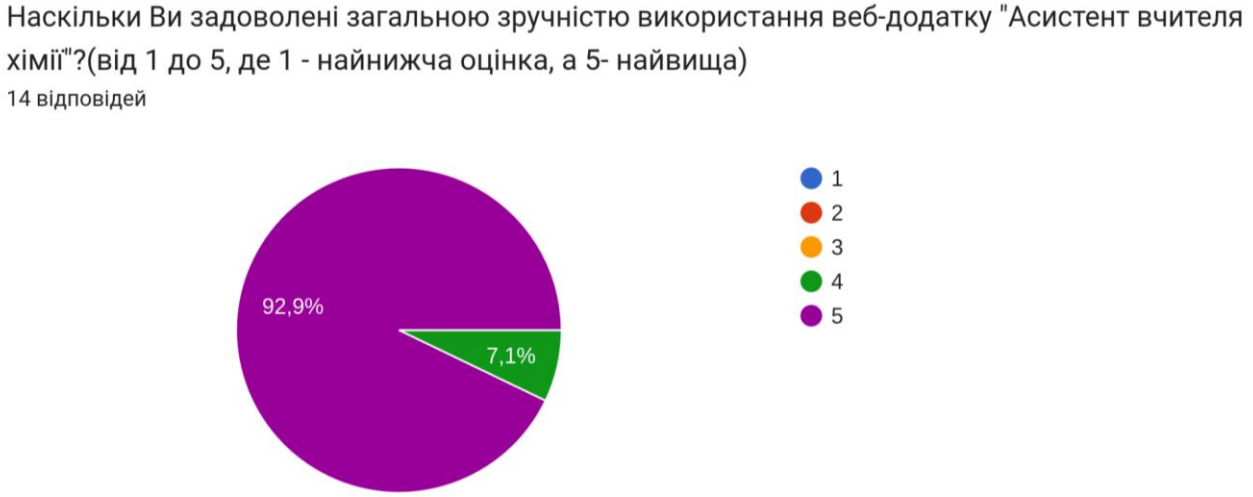

Рисунок 4.5 – Зручність використання

Педагоги відзначили легкість створення нових завдань, особливо можливість швидко генерувати їх на основі теми (рис. 4.6). Деякі респонденти вважають, що процес редагування міг би бути більш інтуїтивним,.

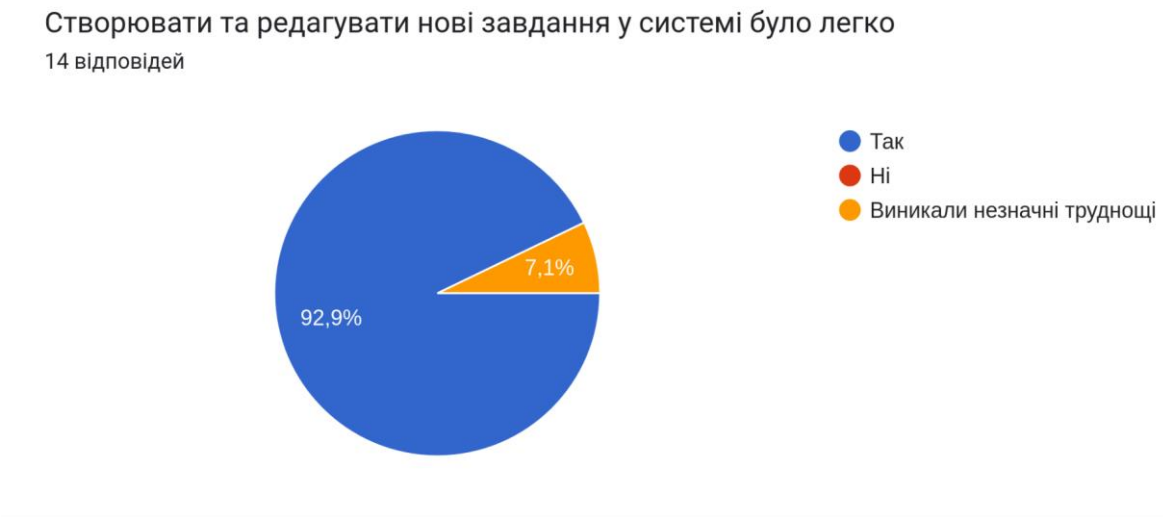

Рисунок 4.6 – Інтуїтивність створення завдань

Більшість опитаних (93%), оцінила зручність категоризації позитивно. Проте, один з респондентів зазначив, що система категоризації потребує доопрацювання (рис. 4.7).

Рисунок 4.7 – Система категоризації завдань

Респонденти схвально відгукнулися про можливість обирати кількість варіантів, складність завдань і кількість запитань. Такі налаштування допомагають індивідуалізувати перевірку знань і запобігти списуванню (рис. 4.8).

Рисунок 4.8 – Зручність налаштування параметрів генерації

Усі опитані відзначили, що використання додатка скорочує час підготовки контрольних робіт, оскільки більшість дій автоматизована, а збереження шаблонів дозволяє повторно використовувати матеріали (рис. 4.9).

Рисунок 4.9 – Час на підготовку контрольної роботи

71% опитаних в середньому витрачають від 10 до 30 хвилин на створення однієї контрольної роботи з додатком, 29% витрачають менше 10 хвилин, жоден з педагогів не витрачає більш як 30 хвилин (рис. 4.10). При порівнянні цих даних з отриманими раніше, можна зробити висновок, що додаток дозволяє витрачати суттєво менше часу, в порівнянні зі створенням робіт звичними способами.

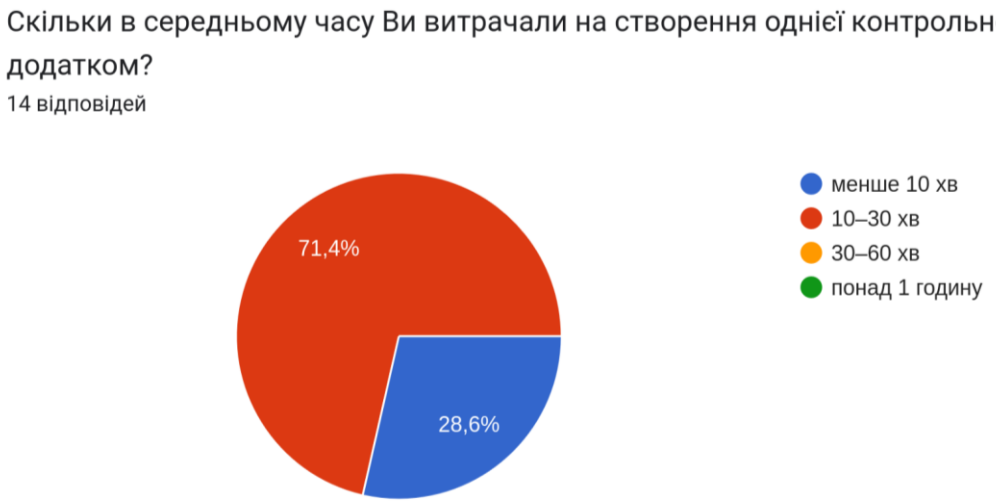

Рисунок 4.10 – Час на створення контрольної роботи з додатком

Учителі зазначили, що додаток генерує достатньо різноманітні варіанти тестів, що допомагає запобігати списуванню. 79% опитаних поставили додатку найвищу оцінку - 5, тоді як 21% - оцінку 4 (рис. 4.11). Середня оцінка – 4.8.

Рисунок 4.11 – Унікальність згенерованих варіантів

Користувачі позитивно оцінили простоту навігації (рис. 4.12), зрозумілі меню та логічну послідовність етапів роботи. Інтерфейс сприймається як зручний навіть для користувачів без технічної підготовки. Середня оцінка – 4,9.

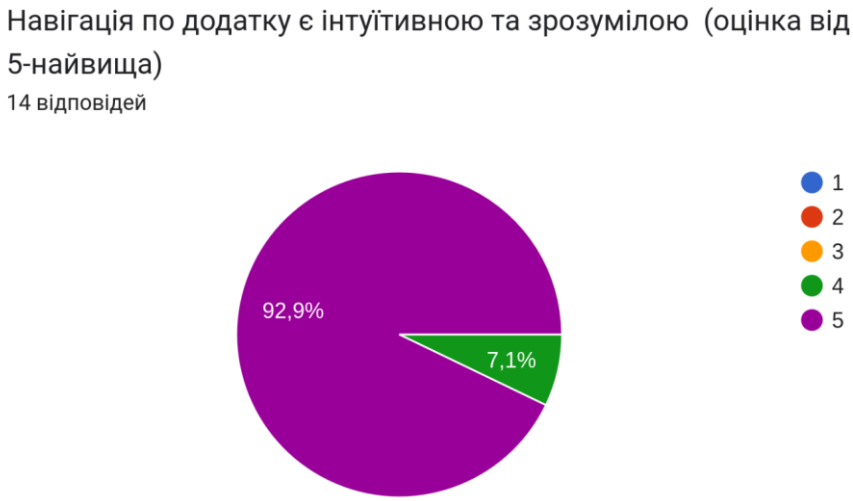

Рисунок 4.12 – Оцінка навігації по додатку

Переважна більшість викладачів задоволені швидкістю роботи, проте один респондент повідомив про певні затримки під час переходу між сторінками або створення контрольних (рис. 4.13). Це може бути пов’язано з характеристиками пристрою або швидкістю Інтернету. Середня оцінка – 4,9.

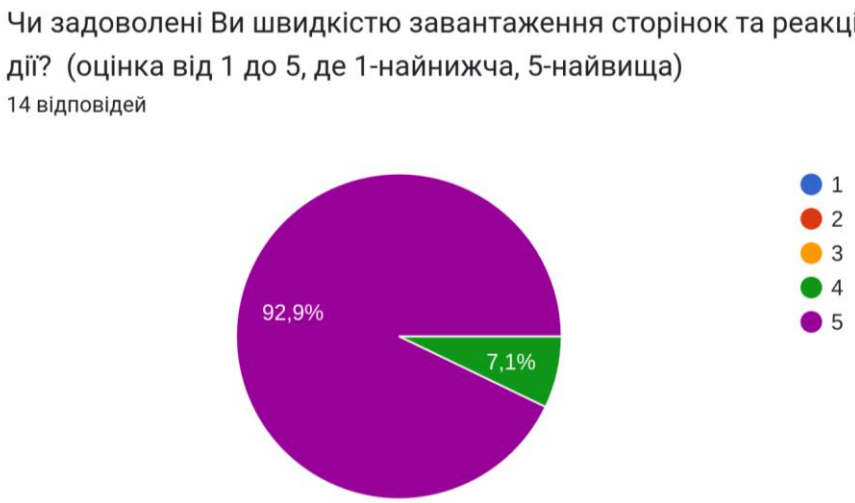

Рисунок 4.13 – Швидкодія додатка

Користувачі зазначили, що додаток коректно працює на всіх пристроях (рис. 4.14). Проте в деяких користувачів виникли певні труднощі, що вказує на потребу доопрацювання мобільної версії інтерфейсу для забезпечення ще більшої зручності під час роботи. Середня оцінка – 4,8.

Рисунок 4.14 – Адаптивність інтерфейсу

Половина учителів заявили, що готові рекомендувати додаток колегам, ще половина зазначили, що будуть рекомендувати додаток, проте мають певні зауваження (рис. 4.15).

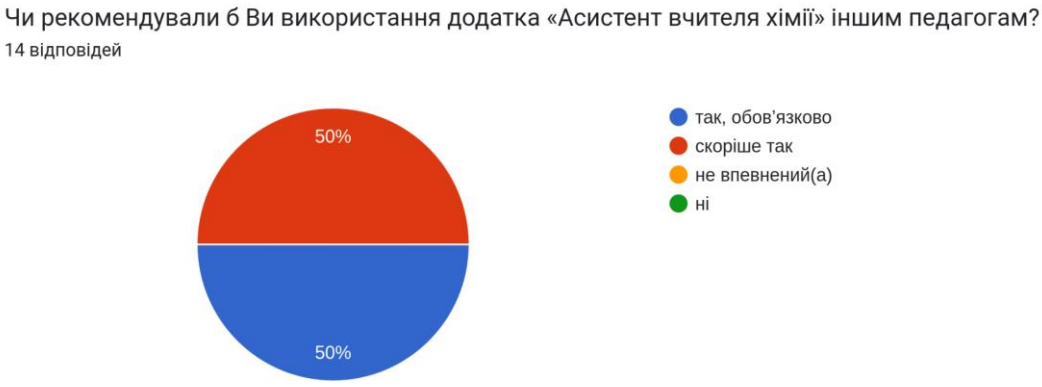

Рисунок 4.15 – Ймовірність рекомендації додатка

За відгуками педагогів, учні позитивно сприймають створені контрольні роботи. Вони відзначають, що завдання зрозумілі, мають чітке формулювання та

адекватну складність. Близько 62% студентів повідомили, що вважають варіанти справедливими за рівнем складності (рис. 4.16). Це підтверджує якість алгоритму генерації завдань.

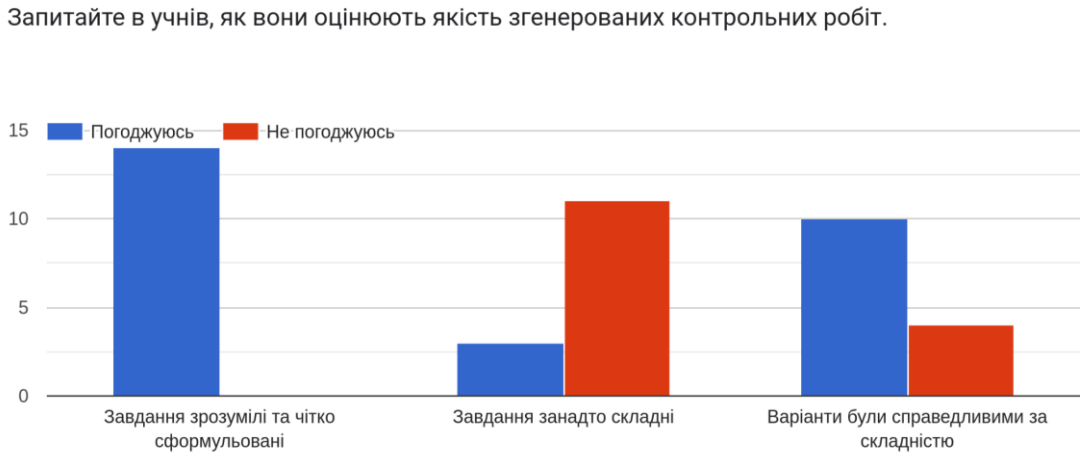

Рисунок 4.16 – Опитування учнів

У відповідях на відкрите запитання «Які аспекти використання додатка Вам сподобалися найбільше?», респонденти виділили такі переваги системи:

- швидке створення контрольних робіт;
- зручність у підготовці кількох варіантів;
- унікальність запитань;
- економія часу;
- можливість систематизації матеріалів за темами.

Серед основних недоліків учасники опитування найчастіше вказували відсутність функціоналу онлайн-тестування. Окремі респонденти також зазначили потребу в додаткових категоріях, розширенні бази запитань та створенні відповідей на питання згенерованої контрольної.

У відповідях, на відкрите запитання щодо пропозицій удосконалення додатка, викладачі запропонували такі напрями подальшого розвитку системи:

- впровадження онлайн-тестування з перевіркою відповідей;
- розширення бази запитань за темами;
- введення рівнів складності для різних категорій учнів;
- оптимізація для мобільних пристроїв

ВИСНОВКИ

Сучасний освітній ландшафт переживає цифрову трансформацію, зумовлену швидким прогресом інформаційно-комунікаційних технологій. Така трансформація вимагає нових підходів до викладання та навчання. Традиційне навчання часто призводить до труднощів у засвоєнні знань та застосуванні їх для вирішення задач. Розробка цифрових інструментів та додатків пропонує перспективний шлях для подолання цих труднощів, шляхом підвищення інтерактивності, персоналізації та доступності освітнього контенту.

У ході аналізу встановлено, що традиційні методи підготовки контрольних матеріалів є надмірно трудомісткими, потребують значних часових ресурсів і не завжди забезпечують об’єктивність оцінювання знань учнів, а наявні програмні аналоги часто мають високу вартість або обмежений функціонал. Виявлено потребу у створенні гнучкого веб-інструменту, який би враховував специфіку предметної галузі не вимагав складного встановлення, дозволяв швидко формувати варіанти завдань різного рівня складності та був доступний з будь-якого пристрою, що й обумовило вибір технологій веб-розробки.

У ході роботи було розроблено та обґрунтовано алгоритмічні та програмні рішення, які забезпечують процес автоматичного генерування варіантів контрольних робіт. Спроектовано та реалізовано інформаційну систему, що базується на архітектурі «клієнт-сервер». Програмна реалізація виконана з використанням сучасного стеку веб-технологій, що забезпечує швидкість веб-додатка, сприяє його масштабованості, стабільності роботи та можливості подальшого розвитку й інтеграції нових функціональних модулів.

Логіка додатка розроблена на базі високорівневого фреймворку Django. Це забезпечило надійність роботи системи, безпеку даних та швидку реалізацію алгоритмів генерації варіантів завдяки потужним вбудованим бібліотекам. Крім того, при потребі розширити додаток на інші предметні галузі, використання Django значно спростить масштабування проєкту.

Для збереження та структурування бази тестових завдань, профілів користувачів та згенерованих варіантів використано реляційну систему управління базами даних MySQL, що дозволяє ефективно обробляти великі обсяги інформації, є надійною та детально задокументованою.

Інтерфейс користувача створено за допомогою HTML та CSS, що забезпечило чітку структуру та приємний візуальний стиль. Інтерактивність системи (динамічне додавання питань, миттєва валідація форм) реалізовано мовою JavaScript, що зробило взаємодію викладача з системою інтуїтивно зрозумілою та зручною.

Реалізовано алгоритми генерації та критерії якості, це гарантує унікальність та мінімізацію повторів у варіантах, повну відповідність згенерованих робіт навчальній програмі, можливість швидкого редагування контенту та додавання нових запитань через адміністративну панель веб-сайту.

Розроблена автоматизована система генерування контрольних робіт забезпечує можливість швидкого створення індивідуальних варіантів завдань, адаптованих до рівня підготовленості студентів, що сприяє підвищенню ефективності навчального процесу та об’єктивності оцінювання.

Апробація розробленого веб-ресурсу довела його практичну цінність. Використання веб-технологій дозволило викладачам отримувати доступ до системи дистанційно, без прив’язки до конкретного робочого місця.

Проведене експериментальне дослідження підтвердило, що веб-додаток «Chem test» має високу практичну цінність і може бути ефективним інструментом у роботі педагогів. Дослідження показало суттєве скорочення часу на підготовку контрольних робіт, підвищення об’єктивності оцінювання та забезпечення однакових умов для всіх здобувачів освіти, завдяки стандартизації структури завдань та генерації унікальних варіантів. Користувачі позитивно оцінили зручність, швидкість роботи, велику кількість завдань в базі та гнучкість налаштувань. Основним напрямом подальшого розвитку є інтеграція онлайн-тестування та покращення адаптивності інтерфейсу. Загалом рівень задоволеності

користувачів свідчить про високу готовність додатка до впровадження у навчальний процес.

Створений програмний продукт є кросплатформним рішенням, готовим до впровадження в освітніх закладах різного рівня. Завдяки використанню Django, система має потенціал до масштабування — наприклад, інтеграції з іншими освітніми платформами або розширення функціоналу модулем автоматичної перевірки відповідей.

Застосування «Chem test» сприяє підвищенню якості контролю знань, зменшенню методичного навантаження на викладачів та покращенню організації освітнього процесу.

Результати дослідження, представлені на Міжнародній конференції «ІТОНВ-2025», підтверджують актуальність обраного підходу до цифровізації контролю знань, практичну значущість отриманих результатів та перспективність подальших досліджень у напрямі автоматизації освітніх процесів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гулай О., Мороз І., Шемет В., Шевчук М. Інтерактивні методи навчання як засіб активізації пізнавальної діяльності здобувачів освіти. 2024, 1(4), С. 256–264.
2. Денисенко С. М. Педагогічний дизайн в сучасному освітньому процесі. Вісник ЖДУ ім. І. Франка. 2015. 3(81). С. 79-83.
3. Кульган А.В., Гулай О.І. Веб-додаток «Асистент вчителя хімії». Тези доповідей X Міжнародної науково-практичної конференції з проблем вищої освіти і науки «Інформаційні технології в освіті, науці і виробництві (ІТОНВ-2025) (23-24 травня 2025 року). Луцьк: відділ іміджу та промоції ЛНТУ, 2025. С. 126-129.
4. Освітні онлайн-платформи для дистанційного навчання школярів. Znayshov.com, URL: https://znayshov.com/News/Details/osvitni_onlain_platformy_dlia_dystantsiinoho_navchannia_shkoliariv (Дата звернення: 23.07.2025).
5. Приймак В.М., Кондрашова О.В. Вплив технологічного прогресу на освіту в умовах швидких змін. Педагогічна академія. 2025, (22), URL: https://doi.org/10.5281/zenodo.17099869 (Дата звернення: 08.10.2025)
6. Розтока О. В. Класифікація технологій навчання учнів, eVNUIR, URL:https://evnuir.vnu.edu.ua/bitstream/123456789/3674/1/Roztoka.pdf (Дата звернення: 05.07.2025).
7. Старченко В. С., Макєєв С. Ю. Організація дослідницької діяльності на уроках хімії за допомогою інтерактивного симулятора PhET. Природничі науки та освіта: сучасний стан і перспективи розвитку: IV Міжнар. наук.-практ. конф., (7–8 листоп, 2024 р.), Харків: нац. пед. ун-т ім. Г. С. Сковороди, 2024. С. 394–396.

8. 6 платформ для створення тестів на урок. JustClass. URL: https://justclass.com.ua/blog/6-platform-dlya-stvorennya-testiv-na-urok/ (Дата звернення: 23.07.2025).
9. 7 сервісів для створення навчальних тестів та завдань онлайн. BUKI. URL: https://buki.com.ua/news/7-servisiv-dlya-stvorennya-navchalnykh-testiv-ta-zavdan-onlayn/ (Дата звернення: 23.07.2025).
10. Britez A. What does UX in education look like?. UX Collective. URL:https://uxdesign.cc/what-does-ux-in-education-look-like-ae1fda4497a8 (Дата звернення: 11.07.2025).
11. Django introduction. Learn web development. MDN. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Django/Introduction (Дата звернення: 27.07.2025).
12. Duisebekova K., Khabirov R., Django as secure web-framework in practice, Vestnik KaZATK, 116(1), P. 275-281
13. Functional vs. Non Functional Requirements. GeeksforGeeks. URL: https://www.geeksforgeeks.org/software-engineering/functional-vs-non-functional-requirements/ (Дата звернення: 25.07.2025).
14. HTML for Web Design. GeeksforGeeks. URL: https://www.geeksforgeeks.org/html-for-web-design/ (Дата звернення: 25.07.2025).
15. How Has Technology Changed Education?. Purdue University College of Education. URL: https://education.purdue.edu/2024/01/how-has-technology-changed-education/ (Дата звернення: 05.07.2025).
16. Hulai, O., Moroz, I. and Shemet, V. Inquiry-Based Learning in the Study of Chemical Disciplines by Food Technologies Students. In Proceedings of the 4th International Conference on History, Theory and Methodology of Learning, 2024, SciTePress, P. 107-113.

17. Hulai, O., Shemet, V., Moroz, I., Savaryn, P., & Kabak, V. Using The Labster Virtual Laboratory to Study Chemistry at a Technical University. Information Technologies and Learning Tools, 102(4), 2024, P. 95-107.
18. Kusumo B., Sutrisman H. The Impact of Technology-Based Learning on Student Engagement and Achievement in the Digital Era, International Journal of Educational Evaluation and Policy Analysis, 2024 1(4), P. 41-53.
19. MySQL: Understanding What It Is and How It's Used. Oracle. URL: https://www.oracle.com/mysql/what-is-mysql/ (Дата звернення: 27.07.2025).
20. Paliwal P. Web Development via HTML, CSS and Javascript, International Journal of Research Publication and Reviews, 2024, 5(4), P. 3326-3333.
21. Security in Django. Django documentation. URL: https://docs.djangoproject.com/en/5.2/topics/security/ (Дата звернення: 27.07.2025).
22. Todino, M. D. Educational Technologies, Encyclopedia of Social Sciences, 2025, 5(1), 23, URL: https://doi.org/10.3390/encyclopedia5010023
23. Unleashing the Importance of CSS in Web Development. Antino Labs. URL: https://www.antino.com/blog/importance-css-web-development (Дата звернення: 25.07.2025).
24. What Is Python Used For? A Beginner's Guide. Coursera. URL: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python (Дата звернення: 27.07.2025).
25. What is Adaptive Learning?. Smart Sparrow. URL: https://www.smartsparrow.com/what-is-adaptive-learning/ (Дата звернення: 12.07.2025).
26. What is JavaScript?. Learn web development. MDN. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Scripting/What_is_JavaScript (Дата звернення: 25.07.2025).

ДОДАТКИ

Додаток А. Зразки моделей додатка

Лістинг А.1 – Моделі додатка (models.py)

```
#Модель питання
class Question(models.Model):
    subject = models.ForeignKey(Subject, on_delete=models.CASCADE)
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    text = models.TextField(default='0') #текст питання
    level = models.IntegerField(default=1) #рівень складності
    answer = models.CharField(max_length=50, default='Без відповіді')
    processed = models.BooleanField(default=False) #чи модероване
    source = models.CharField(max_length=250) #джерело питання

#Модель статті
class Article(models.Model):
    title = models.CharField(max_length=200)
    type = models.CharField(max_length=100)
    content = models.TextField()
    def __str__(self):
        return self.title


#Модель збереженого варіанта
class SavedVariant(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    name = models.CharField(max_length=200, default="Варіант")
    created_at = models.DateTimeField(auto_now_add=True)
    data = models.JSONField()  # Зберігаємо питання як JSON

    def __str__(self):
        return f"{self.name} ({self.user.username})"
```

Кінець лістингу А.1

Додаток Б. Зразки функцій додатка

Лістинг Б.1 – Функції додатка (views.py)

```
#Створення PDF зі згенерованого
def make_pdf(variants):
     # Імена шрифтів, які будуть використовуватися в PDF
    fonts = ['Times', 'NotoSerif', 'RobotoSerif', 'Rubik']
    font = fonts[0]  # Ім'я шрифту, що буде використовуватися в PDF
    font_bold = f'{font}-Bold'  # Жирний шрифт
# Шлях до файлів шрифтів
     font_path = os.path.join(settings.BASE_DIR,
                              'chemistry', 'static', 'chemistry',
                              'font', f'{font}', f'{font}.ttf')
    font_bold_path = os.path.join(settings.BASE_DIR,
                              'chemistry', 'static', 'chemistry',
                              'font', f'{font}', f'{font_bold}.ttf')
# Реєструємо шрифти
    pdfmetrics.registerFont(TTFont(f'{font}', font_path))
    pdfmetrics.registerFont(TTFont(f'{font}-Bold', font_bold_path))
    buffer = io.BytesIO()
    pdf_canvas = canvas.Canvas(buffer, pagesize=A4)
    pdf_canvas.setFont(f'{font}', 14)
    page_width, page_height = A4  # Розміри сторінки
    left_margin = 50
    right_margin = 50
     # Максимальна ширина тексту
     max_width = page_width - left_margin - right_margin
     # Початкова позиція для тексту (50 відступ зверху)
    y_position = page_height - 50
def render_text_with_tags(canvas, text, x, y):
      #Рендерить текст із підтримкою тегів html.
     parts=re.split(r'(<sub>.*?</sub>|<sup>.*?</sup>|<br>|<span.*?>.
     *?</span>|<p>.*?</p>)', text)
     for part in parts:
```

```
        # Витягуємо текст між <sub> і </sub>
        if part.startswith('<sub>') and part.endswith('</sub>'):
            sub_text = part[5:-6]
            canvas.setFont(f'{font}', 10)
            canvas.drawString(x, y - 5, sub_text)
            x += pdfmetrics.stringWidth(sub_text, f'{font}', 10)
            canvas.setFont(f'{font}', 14)
        elif part.startswith('<sup>') and part.endswith('</sup>'):
            sup_text = part[5:-6]
                canvas.setFont(f'{font}', 10)
            canvas.drawString(x, y + 5, sup_text)
            x += pdfmetrics.stringWidth(sup_text, f'{font}', 10)
            canvas.setFont(f'{font}', 14)
        elif part == '<br>':
            y -= 15  # Додаємо новий рядок
            x = left_margin  # Повертаємося до лівого краю
        elif part.startswith('<span') and part.endswith('</span>'):
            span_text = re.sub(r'<.*?>', '', part)  #Видаляємо теги
            canvas.drawString(x, y, span_text)
            x += pdfmetrics.stringWidth(span_text, f'{font}', 14)
        elif part.startswith('<p>') and part.endswith('</p>'):
            p_text = part[3:-4]
            canvas.drawString(x, y, p_text)
            y -= 20  # Додаємо відступ після параграфа
            x = left_margin
        else:
            canvas.drawString(x, y, part)
            x += pdfmetrics.stringWidth(part, f'{font}', 14)
    return x, y
for variant_num, questions in variants.items():
    if int(variant_num) > 1:
        y_position -= 50  # Відступ перед номером варіанту


    # Додаємо номер варіанту
```

```
        pdf_canvas.setFont(f'{font}-Bold', 16)
        pdf_canvas.drawString(left_margin, y_position,
                              f"Варіант {variant_num}")
        y_position -= 30  # Відступ після номера варіанту
        pdf_canvas.setFont(f'{font}', 14)
    for question in questions:
            # Розбиваємо текст, який перевищує ширину сторінки
      wrapped_text = simpleSplit(question, f'{font}', 14, max_width)
            for line in wrapped_text:
                    x, y_position = render_text_with_tags(
                         pdf_canvas, line, left_margin, y_position
                     )
                y_position -= 15
      # Якщо текст виходить за нижній край сторінки, додаємо нову
                if y_position < 50:
                    pdf_canvas.showPage()
                    pdf_canvas.setFont(f'{font}', 14)
                    y_position = page_height - 50
    pdf_canvas.save()
    buffer.seek(0)
    return buffer

#Створення PDF зі збереженого
def generate_pdf_from_saved(request, pk):
    variant = get_object_or_404(SavedVariant, pk, user=request.user)
    buffer = make_pdf(variant.data)
    filename = f"{variant.name}.pdf"
    response = FileResponse(buffer, as_attachment=True, filename)
    return response

#Генерація контрольної
def generate(numOfVariants, numOfQuestions, multicategory):
    all_questions = []  # Список всіх питань у вибраних категоріях
    for category_id in multicategory:
        try:
```

```
            category = Category.objects.get(id=category_id)
            category_questions =
             list(Question.objects.filter(category=category).values_l
             ist('text', flat=True))
            all_questions.extend(category_questions)
        except Category.DoesNotExist:
            continue  # Пропускаємо, якщо категорія не знайдена
    if len(all_questions) < numOfQuestions * numOfVariants:
         return {1: ['<span class="material-symbols-rounded"> warning
         </span>','Недостатня кількість запитань з даної теми в базі,
         оберіть менше питань або варіантів']}
    random.shuffle(all_questions)  # Перемішуємо всі питання
    generated_q = {}
    for variant_num in range(1, numOfVariants + 1):
          # Вибираємо перші N питань
          variant_questions = all_questions[:numOfQuestions]
          # Видаляємо вибрані питання з вибірки
          all_questions = all_questions[numOfQuestions:]
          generated_q[variant_num] = [f'{i + 1}. {q}' for i, q
                                in enumerate(variant_questions)]
    return generated_q

#Підрахунок кількості питань у вибраних категоріях:
def num_of_questions(multicategory):
    return Question.objects.filter(category__id__in=multicategory)

#Сторінка відображення питань
def all_questions_page(request):
    questions = Question.objects.filter(processed=False).values()
    questions_json = json.dumps(list(questions))
    categories = Category.objects.all()
    context = {
        'questions': questions,
        'categories': categories,
        'questions_json': questions_json,
```

```
    }
    return render(request, 'chemistry/questions.html', context)

#Сторінка редагування питання
def edit_question(request, question_id):
    question = Question.objects.get(id=question_id)
    if request.method != 'POST':
        form = QuestionForm(instance=question)
    else:
        form = QuestionForm(instance=question, data=request.POST)
        if form.is_valid():
            form.save()
    context = {'question': question, 'form': form}
    return render(request, 'chemistry/edit_question.html', context)

#Збереження згенерованого варіанта
def save_generated_variant(request):
    if request.method == "POST":
        variants = request.session.get('generated_variants', {})
        category_ids = request.session.get('last_categories', [])
        categories = Category.objects.filter(id__in=category_ids)
        name = " / ".join([cat.category_name for cat in categories])
        if variants:
               SavedVariant.objects.create(user=request.user,
                                           name,data=variants)
               return redirect('chemistry:saved_variants')
    return redirect('chemistry:generator')

#Додавання питання
def add_question(request):
    if request.method != 'POST':
        form = QuestionForm(initial={'processed':True})
    else:
        form = QuestionForm(data=request.POST)
        if form.is_valid():
```

```
            form.save()
            return redirect('chemistry:add_question')
    context = {'form': form}
    return render(request, 'chemistry/add_question.html', context)


def random_question(request):
    #вибір випадкового питання
    question=Question.objects.filter(processed=False)
              .exclude(answer=0).order_by('?').first()
    context = {'question': question}
   return render(request, 'chemistry/random_question.html', context)

#Відображення збережених контрольних
def saved_variants(request):
     variants=SavedVariant.objects.filter(user=request.user)
                                  .order_by('-created_at')
    context = {'variants': variants}
    return render(request, 'chemistry/saved_variants.html', context)
def saved_variant_detail(request, pk):
    variant = get_object_or_404(SavedVariant,
              pk=pk, user=request.user)
    return render(request, 'chemistry/saved_variant_detail.html',
              {'variant': variant})
#Сторінка методичних матеріалів
def articles_page(request):
    articles = Article.objects.all().order_by('title')
    return render(request, 'chemistry/articles.html' ,
                  {'articles': articles})
def article_page(request, article_id):
    article = get_object_or_404(Article, id=article_id)
    return render(request, 'chemistry/article_page.html',
                  {'article': article})
```

Кінець лістингу Б.1