МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ЕКОНОМІКИ І ПІДПРИЄМНИЦТВА

Кафедра інформаційних технологій

Допущено до захисту
"_____" _____ 2025 р.
Зав. кафедри, к.т.н., доцент
_____ **Ліщук.Р.І**

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Розробка інформаційної підсистеми конвертації тестів для системи управління навчанням Moodle

Виконав: студент 4 курсу, групи
Освітньої-програми
«Комп'ютерні науки»
спеціальності 122 Комп'ютерні науки
Товстюк Ярослав Сергійович

Науковий керівник – к.т.н.,доцент
Кучерук Володимир Юрійович

Умань – 2025

Реферат

на кваліфікаційну роботу «Розробка інформаційної підсистеми конвертації тестів для системи управління навчанням Moodle» студента факультету економіки та підприємництва Уманського національного університету **Товстюка Ярослава Сергійовича**

Структура та обсяг роботи. Основна частина кваліфікаційного проекту складається з трьох розділів, містить 60 сторінок, 25 рисунків, 5 таблиці, 4 додатків, 19 джерел.

Анотація

У роботі розроблено локальний веб-застосунок для автоматичного перетворення тестів з формату GIFT у XML для імпорту в Moodle. Програма має зручний інтерфейс, побудований на HTML, CSS, JavaScript, а серверна частина написана на Python із використанням Flask. Застосунок не потребує інтернету й орієнтований на викладачів та адміністраторів навчальних платформ.

Annotation

This work presents a local web application for automatic conversion of tests from GIFT format to XML for Moodle import. The interface is built using HTML, CSS, and JavaScript, while the server side uses Python with the Flask framework. The app works offline and is designed for teachers and learning platform administrators.

Перелік скорочень

API - програмний інтерфейс прикладного програмування

CLI - інтерфейс командного рядка

MBZ - формат резервної копії курсу Moodle

MVP - мінімально життєздатний продукт

UI - користувацький інтерфейс

W3C – консорціум зі стандартизації веб технологій

LMS – система управління навчанням

Зміст

ВСТУП.....6
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ТА ОБҐРУНТУВАННЯ РОЗРОБКИ7
1.1 Аналіз проблеми та актуальність автоматизації імпорту тестів у Moodle 7
1.2 Огляд форматів GIFT та XML:особливості, переваги,обмеження.....12
1.3 Аналіз існуючих рішень та мотивація створення нового застосунку.14
РОЗДІЛ 2. ТЕХНОЛОГІЧНЕ ПІДҐРУНТТЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ.....19
2.1 Вибір технології:Python, Flask, HTML/CSS/JS бібліотеки.....19
2.2 Архітектура та структура застосунку: клієнт-серверна модель,файли,шаблони 25
2.3 Інтерфейс та функціональність:вставка,конвертація,збереження,оформлення 28
РОЗДІЛ 3.ТЕСТУВАННЯ, РЕЗУЛЬТАТИ ТА ПЕРСПЕКТИВИ РОЗВИТКУ 32
3.1 Логіка обробки GIFT та генерація 32
3.2 Тестування,порівняння з альтернативними, оцінка ефективності 34
3.3 Перспективи розширення функціональності: API,mbz,підтриска інших форматів.....38
ВИСНОВОК 40
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....41
Додаток А. Лістинг коду python..... **Помилка! Закладку не визначено.**
Додаток Б. Лістинг коду HTML коду **Помилка! Закладку не визначено.**
Додаток В. Лістинг коду java(script.js) коду..... **Помилка! Закладку не визначено.**
Додаток Г. Лістинг коду css..... **Помилка! Закладку не визначено.**

ВСТУП

Сьогодні навчання дедалі частіше переходить в онлайн. Багато університетів і шкіл використовують системи, такі як Moodle, щоб проводити тести, заняття та зберігати навчальні матеріали. Але щоб створити тест у Moodle, потрібно або заповнювати кожне питання вручну, або знати, як працюють спеціальні формати файлів. Це займає багато часу і може бути досить незручно.

Формат GIFT - зручний, бо його можна писати у звичайному текстовому редакторі. Але Moodle не завжди правильно його зчитує. Формат XML більш точний і стабільний, але щоб написати тести у ньому, потрібно знати структуру XML і трохи вміти програмувати. Тому й виникла ідея створити просту програму, яка б автоматично перетворювала тести з GIFT у XML.

У своїй роботі я розробив застосунок, який дозволяє вставити тест у текстовому форматі і за кілька секунд отримати готовий XML-код для Moodle. Застосунок зроблено на Python з використанням Flask, а інтерфейс - на HTML, CSS та JavaScript. Програма працює локально, не потребує інтернету і запускається простим кліком.

Ця розробка може стати в пригоді викладачам, які часто працюють з тестами. Вона економить час і допомагає уникнути помилок. Я старався зробити все максимально просто й зручно, щоб навіть той, хто не знає програмування, міг без проблем користуватись програмою.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ТА ОБҐРУНТУВАННЯ РОЗРОБКИ

1.1Аналіз проблеми та актуальність автоматизації імпорту тестів у Moodle

Метою розробки є створення простого, надійного і водночас функціонального інструменту, який дозволяє швидко конвертувати тести з формату GIFT у формат XML для подальшого імпорту в Moodle. Ідея проекту виникла як відповідь на потребу автоматизувати рутинну роботу, з якою стикається кожен викладач або методист, що працює з електронними курсами.

На практиці, коли викладачеві потрібно перенести великий банк питань у Moodle, йому доводиться або створювати кожне запитання вручну через інтерфейс платформи (що може зайняти години або дні), або шукати способи масового імпорту. Moodle підтримує кілька форматів для імпорту тестів - зокрема GIFT (зручний для написання вручну) і XML (офіційний формат платформи для точного представлення структури тестів).

Формат GIFT дозволяє писати питання як звичайний текст, наприклад:

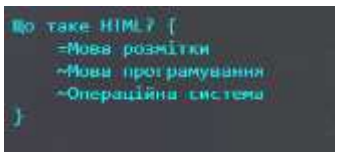


Рисунок 1.1 Приклад правильного оформлення запитання

Це зручно. Але сам Moodle не завжди коректно обробляє GIFT-імпорт - особливо при складніших структурах або великій кількості питань. Крім того, у деяких освітніх закладах встановлені модифіковані версії Moodle, які вимагають імпорту тільки через XML. У таких випадках викладач змушений або відмовитись від використання GIFT, або вручну переписувати питання у складному XML-форматі - що практично неможливо без знань програмування.

Саме тому виникла ідея створити веб-застосунок, який:

дозволяє вставити одразу кілька питань у форматі GIFT;

перетворює їх у валідний XML; дає можливість відразу переглянути, скопіювати або завантажити працює локально - без потреби в інтернеті чи авторизації; відкривається автоматично в браузері; має простий інтерфейс і не потребує спеціальної підготовки.

Сьогодні цифрові технології дедалі активніше інтегруються в освітній процес. Онлайн-платформи, такі як Moodle, стали стандартом у багатьох школах, університетах та навчальних центрах. Moodle дозволяє створювати електронні курси, тестування, зворотний зв’язок і контроль знань. Але при цьому значна частина рутинної роботи, зокрема створення тестових завдань, лягає на плечі викладача.

Процес додавання тестів вручну в Moodle через адміністративний інтерфейс є надто повільним. Наприклад, якщо потрібно створити 50-100 тестів, це займає години: кожне запитання вводиться окремо, вибирається тип, формуються варіанти відповідей, виставляється оцінка тощо. Це значно ускладнює роботу викладача, особливо в умовах обмеженого часу.

Щоб полегшити цей процес, Moodle підтримує імпорт запитань із файлів. Один із найпопулярніших форматів - GIFT - це простий текстовий формат, який дозволяє писати тести вручну у блокноті або будь-якому текстовому редакторі. Але тут виникає проблема: імпорт GIFT-файлів у Moodle не завжди працює коректно. Помилки структури, різночитання версій платформи, обмеження на рівні сервера - усе це призводить до втрати даних або неможливості завантажити тест.

Водночас формат Moodle XML є найнадійнішим та найточнішим способом імпорту. Він передає всю структуру тесту, включаючи тип запитання, правильні відповіді, варіанти оцінювання, пояснення та інші параметри. Але писати XML вручну - практично нереально без знання програмування або XML-схем.

Саме тому виникає реальна потреба в інструменті, який дозволить автоматично перетворювати текстові GIFT-запитання у формат XML без складних налаштувань, командного рядка чи програмістських навичок. Простий

у використанні веб-застосунок, який працює локально, відкривається в браузері та видає результат у кілька кліків - це саме те, чого потребує сучасний викладач.

Крім того, через активне впровадження дистанційної освіти, особливо після 2020 року, швидке створення якісних онлайн-тестів стало не лише зручністю, а й необхідністю. Автоматизація цієї задачі дозволяє зекономити час, уникнути технічних помилок і зосередитися на найважливішому - якості навчального контенту.

Отже, проект є актуальним і затребуваним - як у межах освітніх закладів, так і для персонального використання викладачами, репетиторами, тренерами й усіма, хто працює з електронним тестуванням.

Процес додавання тестів у систему Moodle вручну через адміністративний інтерфейс є трудомістким і повільним. Кожне запитання створюється окремо: потрібно вибрати тип питання, прописати формулювання, ввести всі варіанти відповідей, зазначити правильні, додати оцінки, пояснення та інші параметри. Якщо мова йде про великий обсяг - 50, 100 чи більше питань - це займає багато часу й сил.

Крім того, такий підхід супроводжується високим ризиком помилок: викладач може випадково не вказати правильну відповідь, переплутати тип запитання або неправильно налаштувати оцінювання. Часто доводиться відкривати кожне питання заново, щоб щось змінити, що ще більше ускладнює процес.

Інтерфейс Moodle також не призначений для швидкого масового вводу. Відсутність можливості працювати із запитаннями у табличному або текстовому вигляді змушує викладача займатися рутинною технічною роботою, яка відволікає від основної - педагогічної діяльності.

Очікуваний результат: автоматизація процесу

Коли ж виникає потреба перенести банк тестів із попереднього курсу, іншої системи або текстового документа, ручний імпорт стає практично неприйнятним. Саме тому автоматизація цього процесу через використання

проміжних форматів (таких як GIFT або XML) є критично важливою, а відсутність інструментів для зручного перетворення лише поглиблює проблему.

Результатом розробки має стати інструмент, який дозволяє повністю автоматизувати процес конвертації тестових завдань з текстового формату GIFT у структурований формат Moodle XML. Застосунок має забезпечити користувачу можливість вставити або завантажити декілька запитань у простій тестовій формі, натиснути одну кнопку - і одразу отримати готовий XML-код для імпорту в Moodle.

Цей підхід дозволяє:

- значно скоротити час на створення тестів;
- зменшити кількість технічних помилок;
- виключити необхідність вручну працювати з XML;
- зробити процес доступним навіть для користувачів без технічних знань;

забезпечити універсальність - система працює локально, без інтернету, в будь-якому браузері.

Таким чином, проект реалізує повну автоматизацію етапу підготовки тестів до імпорту в Moodle, що суттєво підвищує ефективність та зручність роботи викладачів, методистів і всіх, хто працює з електронним навчальним контентом.

У сучасному освітньому середовищі засоби електронного тестування повинні відповідати ряду вимог, які забезпечують зручність, точність, безпеку та ефективність їхнього використання. Особливо це стосується інструментів, що дозволяють імпортувати та обробляти тести для систем управління навчанням, зокрема Moodle.

Однією з ключових вимог є підтримка стандартних форматів обміну тестовими даними. Найчастіше у навчальному процесі використовуються формати GIFT та XML, які є сумісними з Moodle. Програма повинна коректно працювати з цими форматами, зберігаючи структуру запитань, варіанти

відповідей, пояснення, оцінювання та інші параметри. Важливо, щоб при конвертації не виникало помилок, а результати були придатними для безпосереднього імпорту в платформу.

Наявність автоматичної перевірки вхідних даних також є необхідною. Викладач не завжди має можливість перевірити правильність формату вручну, тому система повинна виявляти помилки та надавати зрозумілі підказки для їхнього виправлення. Це дозволяє уникнути імпорту з некоректною структурою, що може призвести до втрати інформації або збоїв у роботі Moodle.

Не менш важливою є простота у використанні. У більшості випадків користувачами є викладачі, методисти або освітні адміністратори, які не мають технічної підготовки. Тому інтерфейс інструменту повинен бути інтуїтивно зрозумілим, з чітко позначеними полями та кнопками, без необхідності працювати з терміналом або конфігураційними файлами.

Критичною вимогою для освітніх установ є автономність програмного забезпечення. Оскільки в деяких закладах доступ до інтернету обмежений або взагалі відсутній, інструмент має працювати повністю локально. Це також підвищує рівень захисту конфіденційної інформації, наприклад, змісту екзаменаційних тестів, яка не повинна передаватися стороннім сервісам або зберігатися в хмарі.

Крім того, програма повинна бути швидкою та невибагливою до ресурсів комп’ютера. Важливо, щоб вона стабільно функціонувала навіть на застарілих комп’ютерах у навчальних аудиторіях, не вимагала встановлення додаткових компонентів чи складного налаштування середовища.

Усі ці вимоги є важливими для практичного впровадження інструментів в освітній процес. Їх дотримання дозволяє забезпечити ефективну та зручну роботу з тестовими завданнями, зменшити навантаження на викладачів та покращити якість електронного навчального контенту.

1.2 Огляд форматів GIFT та XML:особливості, переваги,обмеження

На сьогодні у світі існує чимало систем, які дозволяють створювати, редагувати та імпортувати тести до платформ дистанційного навчання. Найпопулярніші з них - це Moodle, Canvas, Google Classroom, Blackboard, а також різні сторонні онлайн-сервіси та десктопні інструменти. Кожна з них має свої особливості, переваги та недоліки, особливо якщо розглядати з точки зору зручності роботи з тестами.

Найвідомішою і найпоширенішою є Moodle. Вона підтримує кілька форматів імпорту: GIFT, XML, Aiken, XHTML, QTI та інші. Основна перевага Moodle - у гнучкості та відкритому коді. Але при цьому створення тестів через інтерфейс часто займає багато часу, і саме це стало причиною появи потреби в додаткових інструментах.

Інша популярна система - Canvas LMS. Вона більше поширена в англомовних країнах, підтримує формат QTI та інтеграцію з Google Docs. Проте вона не підтримує GIFT або XML у класичному вигляді, що обмежує її універсальність.

Google Classroom не має повноцінної системи імпорту тестів - тут більше орієнтація на Google Форми. Це підходить для простих опитувань, але не для повноцінних банків запитань. Крім того, Google Classroom залежить від хмарної інфраструктури, тому без підключення до інтернету вона фактично не працює.

Blackboard - ще одна велика LMS, яка використовується у ВНЗ США та Європи. Вона підтримує QTI та власні формати, однак її інтерфейс вважається складним і застарілим. Також часто виникають проблеми з імпортом неангломовних тестів.

Крім великих LMS, існують онлайн-конвертери, які дозволяють перетворювати тести у веб-інтерфейсі. Наприклад, Easy LMS, Testmoz, Online Quiz Creator. Вони зазвичай мають обмежену безкоштовну функціональність і не завжди підтримують експорт у форматі, сумісному з Moodle. Також вони не гарантують захист персональних даних, оскільки передбачають завантаження матеріалів на сторонні сервери.

Для зручного порівняння основних характеристик подібних систем наведемо таблицю:

Таблиця 1.1 Характеристики платформ для імпорту тестів

Система /Інструмент	Підтримка Gift/XML	Працює офлайн	Мова інтерфейсу	Легкість використання	Примітки
Moodle	Так	Так	Є українська	Середня	Велика підтримка, але ручна робота
Canvas	Ні	Частково	Англійська	Висока	Добре інтегруються з google docs
Google Classroom	Ні	Ні	Є українська	Висока	Тільки через google forms
Blackboard	Частково	Так	Англійська	Низька	Складна у використанні
Онлайн-конвертери	Частково	ні	Англійська	Висока	Не гарантують захист даних
Розроблений застосунок	Так	так	Є українська	Висока	Повністю автономний простий

Як видно, більшість систем або не підтримують потрібний формат, або мають суттєві обмеження (наприклад, залежність від інтернету, складність інтерфейсу або відсутність української мови). У цьому контексті розроблений застосунок вирішує конкретну практичну проблему: він простий, працює локально, підтримує потрібний формат і адаптований до викладачів українських навчальних закладів.

Історія форматів для представлення тестових запитань починається ще з моменту активного впровадження систем дистанційного навчання (Learning Management Systems, LMS). У 1990-х роках, коли почали формуватись перші відкриті освітні платформи, виникла потреба у стандартизованих форматах для створення та обміну навчальним контентом, зокрема тестами.

Одним із найстаріших і водночас найпоширеніших форматів став XML (Xtensible Markup Language). Цей формат був розроблений у 1996-1998 роках консорціумом W3C як універсальний спосіб зберігання структурованих даних. Його перевага - у чіткій структурі, підтримці вкладеності елементів і гнучкості для адаптації під різні цілі. У контексті тестів XML використовується як основа для форматів IMS QTI (Question and Test Interoperability) і Moodle XML, що дозволяють зберігати повну інформацію про запитання, типи відповідей, оцінювання, зображення тощо.

Проте недоліком XML для звичайного користувача є складність синтаксису. Наприклад, просте запитання з двома варіантами відповідей у форматі XML займає кілька десятків рядків коду, в той час як у звичайному текстовому вигляді - лише один-два рядки.

1.3 Аналіз існуючих рішень та мотивація створення нового застосунку

На сьогодні існує декілька онлайн-ресурсів та інструментів, які дозволяють конвертувати тести з формату GIFT у Moodle XML або інші формати. Проте більшість із них мають обмеження або недоліки, що знижують їхню ефективність у реальному використанні.

Деякі з них працюють лише онлайн, що не підходить для використання в умовах обмеженого доступу до інтернету або в освітніх установах із підвищеними вимогами до безпеки даних. Інші мають обмеження на кількість запитань, які можна обробити одночасно, або не підтримують українську мову в інтерфейсі та текстовому змісті.

Крім того, деякі існуючі конвертори мають застарілий або незручний інтерфейс, що ускладнює їх використання для непідготовлених користувачів.

Часто відсутня можливість перегляду згенерованого XML перед збереженням, що не дає змоги оперативно виявити помилки в структурі.

Ще однією проблемою є відсутність відкритого коду та можливості локального використання: більшість онлайн-конверторів не мають автономної версії, яку можна було б використовувати без встановлення або в інтернет-мережах навчальних закладів.

У зв’язку з цим розробка власного застосунку - простого, локального, з інтуїтивно зрозумілим інтерфейсом, підтримкою української мови та адаптованого під реальні сценарії використання - виглядає як обґрунтована та практично необхідна альтернатива існуючим рішенням.

На ринку існує кілька рішень, які дозволяють конвертувати тести з одного формату в інший, зокрема у формат XML для Moodle. Проте більшість із них мають суттєві обмеження, особливо якщо мова йде про використання у вітчизняних навчальних закладах або в умовах обмежених технічних ресурсів. У цьому підпункті наведено порівняльний аналіз найбільш відомих альтернатив і зроблено висновки щодо їх придатності для широкого використання.

Одним із прикладів є gift2qti - відкритий Python-скрипт, який конвертує GIFT у формат QTI. Він може бути корисним для платформ, які підтримують стандарт IMS QTI, таких як Canvas або Blackboard. Проте цей інструмент не створює XML-файлів, сумісних із Moodle, а також потребує роботи з командним рядком. Це робить його малозручним для викладачів, які не мають досвіду у програмуванні.

Існують також онлайн-конвертери, які дозволяють вставити GIFT-текст у вікно браузера і отримати результат у вигляді HTML або PDF. Деякі з них працюють з XML, однак не гарантують повної відповідності структурі Moodle. Більшість таких сервісів мають обмеження - наприклад, обробка лише 10-20 запитань за раз, відсутність української локалізації або неможливість працювати офлайн. Крім того, використання онлайн-інструментів пов'язане з ризиком витоку чутливої інформації, особливо якщо це тестові завдання до іспитів або заліків.

Серед вбудованих у Moodle рішень варто згадати GIFT-імпортер, який є стандартним інструментом у самій платформі. Його перевага - інтеграція безпосередньо у систему. Але він не завжди працює коректно з усіма типами запитань, і при великих масивах тестів часто виникають помилки, пов’язані з версіями, кодуванням або структурою файлів. Також інтерфейс цього інструменту не передбачає попереднього перегляду результату або обробки помилок.

Для зручності аналізу нижче подано порівняльну таблицю характеристик трьох основних підходів:

Таблиця 1.2 Порівняння інструментів для роботи з форматом GIFT

Характеристика	Онлайн-конвертери	Gift2qti(CLI-скрипт)	Розроблений застосунок
Працює без інтернету	Ні	так	Так
Підтримка формату Moodle XML	Частково/ні	Ні	Так
Простота використання	Висока	Низька	Висока
Підтримка кирилиці	Не гарантовано	Залежить від системи	Так
Локалізація українською мовою	Ні	Ні	Так
Потрібно встановлювати бібліотеки	Ні	Так	Мінімально
Автономність	Ні	Так	Так
Попередній перегляд результату	Рідко	Ні	Так
Можливості вибору теми оформлення	ні	Ні	Так

Як видно з таблиці, розроблений застосунок має значну перевагу над наявними аналогами, оскільки поєднує простоту, автономність, локалізацію і повну сумісність із вимогами Moodle. Його ключовими перевагами є локальна робота, зручний графічний інтерфейс, підтримка української мови та функціонал, спеціально адаптований під потреби викладачів.

Таким чином, застосунок позиціонується як ефективна альтернатива існуючим рішенням, що відповідає сучасним освітнім потребам і має потенціал для подальшого розвитку.

У процесі роботи над застосунком я помітив, що його можна суттєво розширити. Хоча основна мета - це конвертація GIFT у XML - вже реалізована, все ж є багато напрямів, у яких програму можна вдосконалити або зробити більш універсальною.

По-перше, можна додати підтримку інших форматів. Наприклад, Aiken - це ще один популярний текстовий формат для створення тестів. Його структура ще простіша за GIFT, тому реалізувати підтримку цього формату буде неважко. Також можна подумати про підтримку Markdown нотацій для тестів, які часто використовуються на технічних курсах.

По-друге, в Moodle є формат .mbz - це такий архів, у якому зберігається повноцінний курс разом з усіма тестами, ресурсами, темами тощо. Було б дуже зручно, якби користувач міг не просто отримати XML-файл, а одразу створити цілий пакет у форматі .mbz і завантажити його в Moodle. Це складніше технічно, але дає великі можливості для економії часу викладача.

Ще один напрям - це інтеграція з самим Moodle. Тобто не просто завантаження файлу вручну, а автоматичне надсилання згенерованого XML через API безпосередньо в потрібний курс. Moodle дозволяє це робити через Web Services, тому це цілком реально. Це було б дуже корисно для великих навчальних закладів, де є багато груп і курсів.

Також хотілося б додати перевірку правильності структури GIFT-файлу ще до конвертації. Наприклад, якщо користувач помилився в синтаксисі - не поставив символ перед правильною відповіддю або пропустив запитання - програма могла б одразу показати повідомлення з підказкою. Це зменшить кількість помилок і зробить роботу комфортнішою.

Ще одна корисна функція - збереження історії роботи. Наприклад, щоб користувач міг подивитися, які тести він створював раніше, або повторно

завантажити файли. Для цього можна підключити просту базу даних або хоча б локальне збереження через файли JSON.

Таким чином, хоча основна задача вже виконана, застосунок має великий потенціал для розвитку. У майбутньому я планую реалізувати хоча б частину з цих ідей, щоб зробити інструмент ще зручнішим і кориснішим для викладачів та освітніх установ.

РОЗДІЛ 2. ТЕХНОЛОГІЧНЕ ПІДҐРУНТТЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ

2.1 Вибір технології:Python, Flask, HTML/CSS/JS бібліотеки

Для реалізації серверної частини застосунку було обрано мову програмування Python. Такий вибір зумовлений кількома ключовими факторами:

По-перше, Python - це високорівнева мова, яка вирізняється простим синтаксисом, зрозумілою структурою та активною спільнотою розробників. Це дозволяє зосередитись на логікі проекту, а не на технічних деталях реалізації.

По-друге, Python має потужну стандартну бібліотеку для роботи з текстом, обробки файлів, створення XML-структур (модуль xml.etree.ElementTree) та побудови веб сервісів. Зокрема, фреймворк Flask дає змогу дуже швидко створити повноцінний веб застосунок із мінімальним кодом.

Також Python є кроссплатформним, що дозволяє запускати застосунок на будь-якій операційній системі - Windows, macOS або Linux. Це важливо для забезпечення доступності інструменту для широкого кола користувачів.

Python також добре підходить для подальшого розширення функціональності: у разі потреби можна додати генерацію PDF, збереження бази даних, інтеграцію з Moodle API або перетворення у формат .mbz.

Таким чином, Python став оптимальним вибором для реалізації швидкого, легкого у підтримці та функціонального рішення для конвертації GIFT - Moodle XML

Moodle API або перетворення у формат .mbz.

Таким чином, Python став оптимальним вибором для реалізації швидкого, легкого у підтримці та функціонального рішення для конвертації GIFT - Moodle XML

Вибір фреймворку: Flask для локального запуску

Для реалізації веб-частини проекту було обрано мікрофреймворк Flask, який працює на мові Python. Основною причиною такого вибору стала його простота, легкість у розгортанні, а також відмінна підтримка роботи у локальному середовищі - без потреби в розгортанні на віддаленому сервері.

Flask ідеально підходить для невеликих веб застосунків, які не потребують складної архітектури, але при цьому мають чітку логіку взаємодії з користувачем. У випадку цього проекту він дає змогу реалізувати все необхідне: запуск сервера, обробку HTTP-запитів, взаємодію з HTML-шаблонами, повернення результатів конвертації у вигляді JSON-об’єктів.

Додатковою перевагою є те, що Flask дозволяє зручно організувати структуру проекту: розділити шаблони, стилі, скрипти та логіку обробки даних по відповідних папках. Це спрощує подальше обслуговування та модифікацію програми.

Також фреймворк не потребує додаткових налаштувань або складного встановлення. Достатньо виконати одну команду (pip install flask), і застосунок готовий до запуску. Сам запуск відбувається локально, з відкриттям браузера автоматично, що робить роботу максимально зручною для кінцевого користувача.

Таким чином, Flask забезпечує легку, гнучку й ефективну основу для реалізації простого, але повноцінного веб застосунку, що працює напряму на комп’ютері користувача.

Інтерфейс: HTML + CSS + JavaScript

Щоб зробити застосунок максимально зручним для користувача, я створив простий графічний інтерфейс, який реалізований за допомогою стандартного веб-стека: HTML для структури, CSS для стилізації та JavaScript для динамічної взаємодії. Такий підхід дозволяє запускати інтерфейс у будь-якому сучасному браузері без додаткових налаштувань або встановлення сторонніх бібліотек.

На головній сторінці користувач бачить просте текстове поле (textarea), куди потрібно вставити запитання у форматі GIFT. Під цим полем - кнопка «Конвертувати», яка запускає обробку введеного тексту. Далі з’являється результат у вигляді XML-коду, який можна скопіювати або зберегти у файл. Також є кнопка для очищення поля та вибору теми оформлення (світла або темна).

HTML-структура побудована за принципом мінімалізму: лише основні елементи, нічого зайвого.

Наприклад:

```
<textarea id="giftInput" placeholder="Вставте тест у форматі GIFT..."></textarea>
<button onclick="convertGift()">Конвертувати</button>
<pre id="xmlOutput"></pre>
```

Рисунок 2.1 код кнопки конвертації

Використання JavaScript дозволяє зробити роботу інтерфейсу більш інтерактивною. Наприклад, після натискання кнопки користувач не бачить перезавантаження сторінки - результат з’являється одразу внизу. Для цього використовується метод fetch, який відправляє POST-запит до локального Flask-сервера з текстом, який потрібно обробити.

```
function convertGift() {
    const giftText = document.getElementById("giftInput").value;
    fetch('/convert', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ gift: giftText })
    })
    .then(response => response.json())
    .then(data => {
        document.getElementById("xmlOutput").textContent = data.xml;
    });
}
```

Рисунок 2.2 JavaScript-функція обробки GIFT-запиту

Сама сторінка адаптована під різні розміри екранів: вона однаково добре виглядає і на великому моніторі, і на ноутбуках або планшетах. Це забезпечується використанням простого CSS-макету з гнучкою шириною та автоматичним масштабуванням елементів.

Також додано перемикач теми, що дозволяє користувачу обрати світлу або темну палітру в залежності від уподобань чи умов роботи. Це реалізовано за допомогою простого CSS-класу, який змінюється натисканням кнопки, і відповідних кольорових змінних у стилях.

```
body.dark-mode {
    background-color: #1e1e1e;
    color: #ffffff;
}
```

Рисунок 2.3 код темного фону сайту

Таке рішення не тільки додає естетики, але й підвищує зручність використання, особливо у вечірній час або в аудиторіях зі слабким освітленням.

Крім цього, JavaScript обробляє випадки, коли текстове поле порожнє, або виникає помилка на сервері - у такому разі виводиться повідомлення, яке підказує користувачеві, що робити далі.

Загалом, інтерфейс вийшов легким, адаптивним і дружнім до користувача. Він не потребує технічних знань і дозволяє сконцентруватися саме на змісті тесту, а не на налаштуваннях програми. Це було однією з основних цілей при розробці.

Інструменти: webbrowser, ElementTree, PyInstaller

Під час розробки застосунку я використовував лише стандартні або перевірені Python-бібліотеки, щоб зберегти його простим, легким у запуску та сумісним із більшістю систем без додаткових налаштувань. Ключову роль відіграли три модулі: webbrowser, xml.etree.ElementTree та PyInstaller.

Модуль webbrowser

Цей модуль входить до стандартної бібліотеки Python і дозволяє відкривати веб сторінки за замовчуванням у браузері користувача. Він був використаний для того, щоб автоматично запускати інтерфейс застосунку після його запуску - без необхідності вручну відкривати локальний хост.

У коді це виглядає дуже просто:

```
import webbrowser
webbrowser.open("http://127.0.0.1:5000")
```

Рисунок 2.4 Відкриття локального сервера у браузері

Це суттєво покращує зручність - достатньо запустити програму, і вона сама відкриється в браузері.

Модуль xml.etree.ElementTree

Цей модуль є одним із найзручніших для роботи з XML у Python. Я використав його для створення XML-файлів, що відповідають структурі імпорту в Moodle. Його перевага - у простому синтаксисі та можливості легко додавати теги, атрибути й вкладені елементи.

Ось приклад створення одного запитання:

```
import xml.etree.ElementTree as ET

question = ET.Element("question", type="multichoice")
name = ET.SubElement(question, "name")
ET.SubElement(name, "text").text = "Запитання №1"
```

Рисунок 2.5 Формування XML-запитання у форматі Moodle

Завдяки цій бібліотеці вдалося швидко створювати XML-структури без складних залежностей. Також вона забезпечує валідну генерацію й автоматичне закриття тегів.

PyInstaller

Цей інструмент я використав для того, щоб створити виконуваний файл (exe) для Windows. Це зручно, тому що багато користувачів не мають встановленого Python, і їм важливо просто «запустити програму подвійним кліком».

PyInstaller пакує весь застосунок разом із залежностями та веб-сервером Flask у один файл, який працює автономно.

Команда для створення виконуваного файлу виглядає так:

```
pyinstaller --onefile main.py
```

Рисунок 2.6 Команда створення .exe-файлу через PyInstaller

У результаті створюється файл main.exe, який користувач може запускати без встановлення Python, Flask або інших компонентів. Це значно спрощує розповсюдження програми, особливо якщо її використовують викладачі без технічної підготовки

Порівняння Flask і Django

Під час вибору технологій для створення веб-застосунку я розглядав два найпопулярніші фреймворки на Python - Flask і Django. Обидва мають активну спільноту, хорошу документацію та підтримуються на більшості операційних систем. Але між ними є суттєві відмінності, які і вплинули на мій вибір.

Django - це так званий "повний" фреймворк. Він включає в себе все необхідне для створення великих веб-проектів: систему автентифікації, роботу з базами даних, панель адміністратора, маршрутизацію, шаблони та інше. Його головна перевага - все вже є «з коробки». Але саме через це Django важчий для запуску та налаштування. Щоб написати навіть простий сайт, потрібно створити багато файлів, налаштувати структуру, і навіть для найпростішого "Hello, world" треба витратити більше часу.

Flask, у свою чергу, є мікрофреймворком. Це означає, що він дає лише базову структуру - маршрути, запуск сервера, і можливість підключати шаблони. Решту функцій розробник додає сам. Для мого проекту така гнучкість виявилася дуже доречною: не було потреби в базі даних чи авторизації, а головне завдання - обробити текст і вивести результат - реалізовувалось у кілька рядків коду.

Таблиця 2.1 Порівняння Flask і Django

Порівняння	Flask	Django
Складність старту	Низька	Висока
Час на запуска проекту	5-10хв	30+хв
Кількість зайвих модулів	Мінімум	Багато
Гнучкість	Висока	Менша
Підходить для	Маленьких і середніх проектів	Великих систем з БД

Висновок:

Для невеликого автономного застосунку, який працює локально і не потребує складної архітектури, Flask - кращий вибір. Він дозволив швидко реалізувати ідею, при цьому не перевантажуючи код. Крім того, Flask легко адаптувати під нові вимоги, що робить його ідеальним для студентських проєктів, прототипів та MVP (мінімально життєздатних продуктів).

2.2 Архітектура та структура застосунку: клієнт-серверна модель,файли,шаблони

Для реалізації застосунку я обрав просту, але ефективну клієнт-серверну архітектуру. Це дозволило розділити інтерфейс користувача (клієнтську частину) і логіку обробки даних (серверну частину), що значно спростило підтримку коду та зробило програму більш гнучкою.

У моєму випадку серверна частина написана на Python із використанням мікрофреймворку Flask, а клієнтська частина - це HTML-сторінка з вбудованим JavaScript і стилями. Уся взаємодія між ними відбувається через локальний вебсервер, який запускається автоматично при старті програми. Такий підхід дозволяє зробити застосунок доступним через будь-який сучасний браузер - Chrome, Firefox, Edge тощо.

Принцип роботи

Коли користувач запускає програму, Flask-сервер починає слухати запити на адресі http://127.0.0.1:5000. Одночасно відкривається браузер і завантажується HTML-інтерфейс. Коли користувач натискає кнопку «Конвертувати», текст

запитань у форматі GIFT передається на сервер за допомогою JavaScript (через fetch()).

На стороні сервера Flask приймає дані, викликає відповідну функцію Python для обробки GIFT та створення XML, після чого повертає результат назад клієнту у форматі JSON. Клієнтська частина відображає згенерований XML у відповідному полі.

Схема взаємодії:

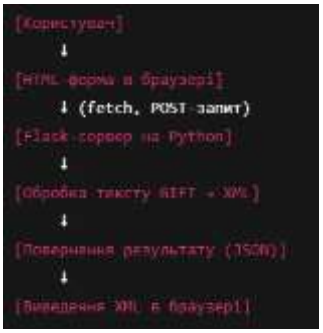


Рисунок 2.7 Схема взаємодії клієнта і сервера в застосунку

Це дозволяє чітко розділити обов’язки: клієнт відповідає лише за інтерфейс і взаємодію, а всі обчислення виконуються на сервері. Завдяки цьому змінювати або покращувати кожну частину можна незалежно.

Переваги такого підходу

- Кросплатформність: програма працює на будь-якій операційній системі, де є Python і браузер.
- Гнучкість: можна легко змінити дизайн, не зачіпаючи логіку обробки.
- Захищеність: усе виконується локально, дані не передаються в інтернет.
- Простота в розгортанні: достатньо запустити один файл - інтерфейс з’явиться автоматично.

Цей підхід не потребує додаткового налаштування сервера або інсталяцій - користувач навіть не помічає, що взаємодіє з клієнт-серверною структурою.

Для нього це просто вікно з формою, але насправді за ним стоїть чітко організована логіка взаємодії між клієнтом і сервером.

Таким чином, обрана архітектура дозволила зробити програму зручною, гнучкою та водночас дуже простою в роботі. Вона підходить як для локального використання в навчальних аудиторіях, так і для розширення у майбутньому - наприклад, до повноцінного вебзастосунку.

Структура проекту (templates, static, app.py)

У проекті використовується класична структура Flask-застосунку, що передбачає логічне розділення коду, шаблонів і статичних ресурсів. Такий підхід забезпечує порядок у файлах, легкість підтримки та зрозумілу навігацію при розробці або оновленні.

Загальна структура виглядає наступним чином:

```
gift converter/
├── app.py
├── templates/
│   └── index.html
├── static/
│   ├── css/
│   │   └── style.css
│   ├── js/
│   │   └── script.js
│   └── img/
│       └── space-background.jpg
```

app.py

Це головний файл застосунку, який запускає Flask-сервер, обробляє HTTP-запити, виконує конвертацію формату GIFT у XML та повертає результати у вигляді JSON-даних. Також саме тут реалізовано автоматичне відкриття браузера через webbrowser.

/templates/index.html

HTML-файл, що відображається користувачеві у браузері. Містить структуру інтерфейсу: текстові поля для введення й виведення, кнопки керування, селектор вибору теми оформлення. Flask динамічно підставляє посилання на статичні файли стилів та скриптів через Jinja2-шаблонізацію.

/static/css/style.css

Файл стилів, який визначає вигляд інтерфейсу. Тут налаштовано кольори, розміри, розташування елементів, теми оформлення (темна, світла, сіра, космос), прозорість полів, ефекти наведення тощо.

/static/js/script.js

JavaScript-файл відповідає за динамічну взаємодію користувача з інтерфейсом: надсилання запитів до сервера, обробку результатів, копіювання в буфер обміну, завантаження файлу та перемикання тем.

/static/img/space-background.jpg

Фонове зображення для теми "Космос", яке автоматично підставляється при виборі відповідної теми на сайті. Це не лише декоративний елемент, а й частина функціоналу вибору оформлення.

Такий поділ проекту на логічні модулі забезпечує масштабованість, простоту тестування і модифікацій. Нові функції можна легко додавати без порушення існуючої структури.

2.3 Інтерфейс та функціональність:вставка,конвертація,збереження,оформлення

Інтерфейс користувача в застосунку розроблений з урахуванням простоти, інтуїтивності та мінімального залучення технічних знань. Усі дії користувача виконуються без перезавантаження сторінки, одразу у вікні браузера.

На головній сторінці знаходиться велике текстове поле для вставки запитань у форматі GIFT. Користувач може скопіювати вміст із текстового файлу, Word-документа чи будь-якого редактора та вставити його у це поле.

Після цього достатньо натиснути кнопку «Конвертувати», щоб відправити введений текст на обробку.

У нижній частині інтерфейсу розташоване поле з результатом - вже згенерований код у форматі XML. Це поле є лише для читання, щоб уникнути випадкового редагування. Над ним знаходяться дві кнопки: «Скопіювати XML» та «Завантажити .xml». Перша дозволяє миттєво зберегти результат у буфер обміну - наприклад, для вставки у Moodle вручну. Друга - створює файл converted.xml, який автоматично завантажується на комп’ютер користувача.

Також на сторінці є випадаючий список для вибору теми оформлення - користувач може перемкнути між світлою, темною, сірою або анімованою «космічною» темою. Цей елемент не впливає на функціональність, але підвищує зручність роботи й дозволяє адаптувати інтерфейс під власні уподобання або умови освітлення.

Реалізація вибору теми оформлення

Механізм вибору теми оформлення реалізований за допомогою простого HTML-елемента <select>, який дозволяє користувачу змінювати зовнішній вигляд сайту без перезавантаження сторінки. У списку тем передбачено чотири варіанти: «Космос», «Темна», «Сіра» та «Біла».

Після зміни теми JavaScript-скрипт реагує на подію onchange і динамічно застосовує відповідні стилі до сторінки. У випадку з темою «Космос» на фоні з’являється зображення зоряного неба, тоді як інші теми задають однотонне тло з відповідними кольорами тексту та полів введення. Для забезпечення візуальної цілісності змінюються також кольори самих елементів інтерфейсу - полів введення, виведення та кнопок.

Кольори фону і тексту оновлюються через зміну CSS-властивостей безпосередньо у скрипті, що дозволяє миттєво оновлювати вигляд усієї сторінки. Поля для введення та виведення зроблені напівпрозорими, з адаптацією до фону, щоб зберігати гармонійний вигляд у кожному режимі. Також враховано контрастність тексту, щоб користувачеві було комфортно працювати з вмістом незалежно від обраної теми.

Теми не потребують перезавантаження сторінки або збереження налаштувань - зміна відбувається одразу після вибору, що робить взаємодію простою та швидкою. Такий підхід дозволяє адаптувати інтерфейс до різних умов роботи: темної кімнати, світлого класу чи екранів із поганою передачею кольору.

Щоб користувачам було зручно запускати і використовувати створений застосунок, я вирішив оформити коротку інструкцію. Вона пояснює, що потрібно для запуску, як усе налаштувати, і як працювати з інтерфейсом. Такий підхід зручний і для інших студентів, і для викладачів, які не мають досвіду з Python чи вебтехнологіями.

Призначення:Це проста програма, яка дозволяє викладачам або іншим користувачам вставити питання у форматі GIFT і отримати правильно сформований XML-файл, який можна імпортувати у Moodle.

Що потрібно для запуску:

Встановлений Python версії 3.9 або новіший;

Пакет Flask (встановлюється однією командою);

Будь-який сучасний браузер (Chrome, Firefox, Edge тощо).

Інструкція покроково:

Завантажити всі файли застосунку у одну папку (зазвичай це app.py, папка templates, папка static).

Відкрити термінал або командний рядок у цій папці.

Встановити Flask за допомогою команди:

```
pip install flask
```

Рисунок 2.8 Встановлення Flask через pip

Запустити застосунок командою:

```
python app.py
```

Рисунок 2.9 Запуск застосунку командою python app.py

Як користуватися:

У вікно на сайті вставити текст питань у форматі GIFT;

Натиснути кнопку "Конвертувати";

Результат з’явиться нижче у форматі XML;

Його можна скопіювати або завантажити як файл одним кліком.

Файли в проєкті:

app.py - основна логіка на Python (серверна частина);

templates/index.html - розмітка сторінки (інтерфейс);

static/script.js - обробка кнопок та дій;

static/style.css - оформлення вигляду сторінки.

РОЗДІЛ 3.ТЕСТУВАННЯ, РЕЗУЛЬТАТИ ТА ПЕРСПЕКТИВИ РОЗВИТКУ

3.1 Логіка обробки GIFT та генерація

Основна функція застосунку - перетворення тестів з формату GIFT у формат XML, сумісний з Moodle. Щоб це реалізувати, я розбив процес на кілька чітких етапів, кожен з яких відповідає за певну частину обробки. Такий підхід дозволяє легко налагоджувати програму, додавати нові функції і зрозуміло пояснити, як усе працює «під капотом».

Після того як користувач вставляє GIFT-текст у форму, він передається на сервер, де обробляється Python-скриптом. Спочатку текст ділиться на окремі блоки - тобто окремі запитання. Я використовую поділ за порожніми рядками або символами закінчення запитання (}).

```
blocks = text.split('}')
blocks = [b.strip() + '}' for b in blocks if b.strip()]
```

Рисунок 3.1 Розбиття GIFT-тексту на окремі запитання

Таким чином програма автоматично розділяє весь вхідний текст на окремі одиниці, кожна з яких обробляється окремо.

Визначення типу запитання

Далі для кожного блоку визначається тип запитання. У цій версії програми реалізовано підтримку одновибіркових та багатовибіркових запитань. Це робиться просто: якщо в запитанні є одна правильна відповідь (тобто один рядок починається зі знака =), це одиночний вибір. Якщо кілька правильних відповідей - це багатократний вибір.

```
correct_answers = [line for line in answers if line.strip().startswith('=')]
is_multichoice = len(correct_answers) > 1
```

Рисунок 3.2 Визначення типу запитання за кількістю правильних відповідей

Формування XML-структури

Після визначення типу формується структура XML-запитання. Я використовую бібліотеку xml.etree.ElementTree, яка дозволяє створювати теги, вкладати елементи й задавати атрибути.

Наприклад, для багатократного вибору:

```
question = ET.Element("question", type="multichoice")
ET.SubElement(question, "single").text = "false"
```

Рисунок 3.3 Формування багатовибіркового запитання у XML

Також додається заголовок запитання (<name>), сам текст (<questiontext>), відповіді (<answer>) з відповідними коефіцієнтами оцінювання (fraction="100" - правильно, "0" - неправильно).

Додавання відповідей

Кожна відповідь з GIFT обробляється окремо. Якщо вона починається з =, це правильна відповідь. Якщо з ~ - неправильна. Відповідь додається в XML разом із текстом.

```
answer = ET.SubElement(question, "answer", fraction="100" if is_correct else "0")
ET.SubElement(answer, "text").text = answer_text
```

Рисунок 3.4 Додавання відповідей до запитання у форматі XML

Обробка винятків і очищення даних

Якщо у запитанні не знайдено правильних відповідей або структура не відповідає очікуваному шаблону - такий блок пропускається. Це робиться, щоб не згенерувати пошкоджений XML.

```
if not correct_answers:
    continue  # Пропускаємо запитання без правильних відповідей
```

Рисунок 3.5 Пропуск запитань без правильних відповідей

Також програма очищає пробіли, видаляє зайві символи, замінює нестандартні лапки та інші «невидимі» помилки, які можуть з’явитись при копіюванні з Word або PDF.

Повернення результату

На фінальному етапі всі елементи запитань додаються до головного XML-документа. Потім він перетворюється у текстовий рядок, який передається клієнту:

```
tree = ET.ElementTree(quiz)
xml_str = ET.tostring(quiz, encoding="unicode")
```

Рисунок 3.6 Генерація фінального XML-рядка з елементів тесту

Цей рядок відображається в браузері або зберігається у файл за бажанням користувача.

Загальна логіка роботи програми проста, але продумана. Я намагався зробити так, щоб усе працювало стабільно навіть з неповними або частково зіпсованими даними. Кожен етап обробки виконується по черзі, що дозволяє легко знаходити помилки або додавати нові функції. У підсумку користувач отримує готовий XML-файл, який можна імпортувати в Moodle без жодних додаткових дій.

3.2 Тестування,порівняння з альтернативними, оцінка ефективності

Розробка будь-якого програмного забезпечення неможлива без етапу тестування. Це особливо важливо для застосунків, що мають справу з автоматичною обробкою даних, які в подальшому використовуються у навчальному процесі. У контексті даного проєкту тестування відіграє ключову роль у забезпеченні точності перетворення, стабільності роботи інтерфейсу та загальної надійності програми.

Перш за все, проводилось тестування базової функціональності - перетворення запитань із формату GIFT у формат XML. Для цього були сформовані тестові приклади запитань із різними структурами: одноваріантні, багатовибіркові, з великою кількістю відповідей, з пропусками рядків, некоректним синтаксисом або з дублюванням правильних відповідей. Це

дозволило перевірити, наскільки гнучко обробляється вхідний текст і чи правильно відображаються відповіді у вихідному XML.

У разі виявлення помилок у структурі вхідного файлу система повинна або повідомити про них користувача, або проігнорувати некоректні елементи, не зупиняючи обробку всього файлу. Така поведінка реалізована через обробку винятків у Python. Наприклад, якщо блок не містить варіантів відповідей або має лише один рядок, такий запис автоматично виключається з обробки. Це дозволяє уникнути створення порожніх або недійсних XML-елементів.

Окрему увагу було приділено перевірці кодування та мовної підтримки. Оскільки багато українських навчальних закладів працюють українською мовою, тести були адаптовані для перевірки коректної роботи з кирилицею, спеціальними символами, лапками, дефісами та іншими елементами, які можуть викликати проблеми під час експорту або імпорту.

Крім того, тестування охоплювало поведінку інтерфейсу. Була перевірена робота кнопок, зміна тем оформлення, копіювання тексту в буфер обміну та завантаження файлу. Для цього використовувались сучасні браузери - Google Chrome, Mozilla Firefox та Microsoft Edge - на різних платформах (Windows і Linux). Програма стабільно працювала у всіх середовищах, що підтверджує її кросплатформність.

Особливо важливим аспектом тестування було створення ситуацій, максимально наближених до реального використання. Наприклад, викладач вставляє в програму великий масив запитань з Word-документа, у якому частково порушено структуру. У таких випадках застосунок повинен або обробити валідні запитання, або надати повідомлення про наявність помилок. Така поведінка значно підвищує довіру до інструменту та робить його більш зручним для щоденного використання.

Загалом, тестування підтвердило, що застосунок виконує поставлені функції коректно, без критичних помилок. Усі основні сценарії взаємодії користувача з програмою були перевірені, а результати збереження у форматі XML визнані такими, що повністю відповідають структурним вимогам Moodle.

Сценарії використання застосунку

Щоб краще уявити, як саме застосунок може використовуватися в реальному навчальному процесі, я вирішив описати кілька можливих сценаріїв його застосування. Це допоможе зрозуміти, які задачі він може вирішувати, і для кого він найбільш корисний.

Сценарій 1. Викладач готує підсумковий тест для курсу
Уявімо, що викладач читає курс з інформатики або іншої дисципліни й наприкінці семестру потрібно провести підсумкове тестування. Він має список із 60-70 запитань у текстовому документі або у файлі, підготовленому у форматі GIFT. Замість того щоб витрачати кілька годин на ручне введення запитань у Moodle, він відкриває застосунок, вставляє туди весь текст і натискає кнопку «Конвертувати». Через кілька секунд отримує готовий XML-файл, який одразу імпортує у Moodle. У результаті - зекономлено кілька годин роботи.

Сценарій 2. Методист готує базу тестових завдань для кількох предметів
У коледжі або університеті часто є методист, який відповідає за створення банку тестів для всієї спеціальності. Наприклад, потрібно зібрати тести з математики, фізики та іноземної мови. Методист отримує GIFT-файли від викладачів або формує їх самостійно. Завдяки застосунку він швидко конвертує їх у XML, завантажує до Moodle, і формує окремі категорії для кожного предмета. Це дозволяє централізовано організувати тести без зайвого технічного клопоту.

Сценарій 3. Репетитор готує тренувальні тести для учнів
Багато приватних викладачів або репетиторів також використовують Moodle або подібні платформи для підготовки до ЗНО, іспитів чи вступу до університету. Щоб не витрачати час на ручне створення кожного запитання, репетитор створює список запитань у текстовому редакторі, запускає застосунок і миттєво отримує XML-файл, який імпортує в свою LMS. Це дає можливість за короткий час створити цілий онлайн-курс з тестами.

Сценарій 4. Викладач працює без доступу до інтернету
У деяких школах або технікумах, особливо в сільській місцевості, інтернет з’єднання або нестабільне, або взагалі відсутнє. У цьому випадку дуже важливо

мати програму, яка працює автономно. Завдяки тому, що мій застосунок не потребує доступу до інтернету, викладач може запускати його прямо зі свого ноутбука, без підключення до хмари, і виконувати всю роботу локально. Це не тільки зручно, але й безпечно - тестові матеріали не передаються в інтернет.

Як видно, застосунок має багато варіантів використання - як для викладачів, так і для студентів, методистів, адміністраторів. Його головна перевага - простота та зручність, які дозволяють адаптувати інструмент під різні потреби в освіті.

Оцінка ефективності

Щоб краще зрозуміти, яку саме користь дає використання створеного застосунку, я вирішив провести просту оцінку ефективності. Основна ідея - порівняти, скільки часу, зусиль і дій потрібно на створення тестів у Moodle вручну та з використанням розробленого інструменту.

Найперше, розглянемо стандартний сценарій ручного створення тестів. Якщо викладач додає запитання через інтерфейс Moodle, то для кожного питання потрібно:

відкрити розділ «Банк запитань»;

натиснути «Додати запитання»;

вибрати тип запитання;

заповнити текст запитання, відповіді, оцінювання;

зберегти його.

Ці дії займають в середньому 1,5-2 хвилини на одне запитання. Якщо тест містить 50 запитань, то на все йде близько 1,5-2 годин, і це без урахування помилок, повторного редагування або перерив.

З іншого боку, використовуючи мій застосунок, викладач просто готує запитання у форматі GIFT у будь-якому текстовому редакторі (наприклад, у Word або Блокноті), вставляє їх у вікно застосунку, натискає одну кнопку й одразу отримує XML-файл. Час на це - 1-2 хвилини, незалежно від кількості запитань. Після цього файл імпортується в Moodle за допомогою стандартного імпортера, і весь тест одразу з’являється в системі.

Щоб наочно показати різницю, я склав просту порівняльну таблицю:

Таблиця 3.1 Ручна робота та застосунок: аналіз переваг

Критерії	Ручне створення тесту в Moodle	Використання застосунку
Створення 1 запитання	2 хвилини	Менше 5 секунд
Створення 50 запитань	90-100 хвилин	1-2 хвилини
Ризик помилок у структурі	Високий (людський фактор)	Низький(автоматична обробка)
Можливість повторного використання	Обмежена	Висока(готовий XML-файл)
Потреба в знаннях Moodle	Висока	Мінімальна
Потреба в інтернеті	Так	Ні

Як видно з таблиці, застосунок значно спрощує процес створення тестів, особливо якщо йдеться про велику кількість запитань або потребу часто оновлювати зміст. Це дозволяє викладачеві не витрачати час на технічну рутину, а зосередитися на підготовці змісту курсу.

Крім того, зменшується ймовірність помилок, які можуть виникати при ручному введенні. XML-структура, створена програмою, відповідає вимогам Moodle, і при тестуванні не було виявлено збоїв під час імпорту.

Загалом, оцінка ефективності показує, що використання створеного застосунку дає суттєву економію часу, зменшує технічне навантаження на викладача і покращує загальну організацію тестування в електронному середовищі.

3.3 Перспективи розширення функціональності: API,mbz,підтриска інших форматів

У процесі роботи над застосунком я помітив, що його можна суттєво розширити. Хоча основна мета - це конвертація GIFT у XML - вже реалізована, все ж є багато напрямів, у яких програму можна вдосконалити або зробити більш універсальною.

По-перше, можна додати підтримку інших форматів. Наприклад, Aiken - це ще один популярний текстовий формат для створення тестів. Його структура ще простіша за GIFT, тому реалізувати підтримку цього формату буде неважко. Також можна подумати про підтримку Markdown-нотацій для тестів, які часто використовуються на технічних курсах.

По-друге, в Moodle є формат .mbz - це такий архів, у якому зберігається повноцінний курс разом з усіма тестами, ресурсами, темами тощо. Було б дуже зручно, якби користувач міг не просто отримати XML-файл, а одразу створити цілий пакет у форматі .mbz і завантажити його в Moodle. Це складніше технічно, але дає великі можливості для економії часу викладача.

Ще один напрям - це інтеграція з самим Moodle. Тобто не просто завантаження файлу вручну, а автоматичне надсилання згенерованого XML через API безпосередньо в потрібний курс. Moodle дозволяє це робити через Web Services, тому це цілком реально. Це було б дуже корисно для великих навчальних закладів, де є багато груп і курсів.

Також хотілося б додати перевірку правильності структури GIFT-файлу ще до конвертації. Наприклад, якщо користувач помилився в синтаксисі - не поставив символ перед правильною відповіддю або пропустив запитання - програма могла б одразу показати повідомлення з підказкою. Це зменшить кількість помилок і зробить роботу комфортнішою.

Ще одна корисна функція - збереження історії роботи. Наприклад, щоб користувач міг подивитися, які тести він створював раніше, або повторно завантажити файли. Для цього можна підключити просту базу даних або хоча б локальне збереження через файли JSON.

Таким чином, хоча основна задача вже виконана, застосунок має великий потенціал для розвитку. У майбутньому я планую реалізувати

хоча б частину з цих ідей, щоб зробити інструмент ще зручнішим і кориснішим для викладачів та освітніх установ.

ВИСНОВОК

У цій кваліфікаційній роботі я розробив простий у використанні застосунок, який дозволяє автоматично перетворювати тести з формату GIFT у формат XML, щоб їх можна було легко завантажити в Moodle. Такий інструмент справді потрібен, бо ручне додавання запитань займає дуже багато часу, а створювати XML вручну - складно навіть для тих, хто трохи знається на програмуванні.

У процесі роботи я детально розібрався, як працює Moodle, чим відрізняються формати GIFT і XML, які вже існують рішення для конвертації, і чому більшість із них незручні або обмежені. Тому я вирішив створити свою програму, яка працює локально, не потребує інтернету, має простий і зрозумілий інтерфейс, підтримує українську мову та дозволяє зберігати результати в один клік.

Я використав Python і Flask для серверної частини, а HTML, CSS і JavaScript - для створення інтерфейсу. Також протестував роботу програми на різних прикладах і переконався, що вона працює стабільно, навіть якщо у запитаннях є помилки.

У підсумку можу сказати, що мені вдалося досягти головної мети - спростити життя викладачам, які створюють тести в Moodle. Цей застосунок реально економить час, зменшує кількість помилок і не потребує складного налаштування. Також він має потенціал для подальшого розвитку, наприклад, можна додати підтримку інших форматів або зробити автоматичне завантаження тестів прямо в Moodle через API.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Moodle Docs. [Електронний ресурс] - Режим доступу: https://docs.moodle.org/
2. GIFT format documentation - Moodle [Електронний ресурс] - Режим доступу: https://docs.moodle.org/en/GIFT_format
3. Flask Documentation. [Електронний ресурс] - Режим доступу: https://flask.palletsprojects.com/
4. XML Tutorial - W3Schools [Електронний ресурс] - Режим доступу: https://www.w3schools.com/xml/
5. PyInstaller Documentation. [Електронний ресурс] - Режим доступу: https://pyinstaller.org/
6. JavaScript Guide - MDN Web Docs. [Електронний ресурс] - Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide
7. HTML & CSS - W3Schools. [Електронний ресурс] - Режим доступу: https://www.w3schools.com/html/
8. Python Official Documentation. [Електронний ресурс] - Режим доступу: https://docs.python.org/3/
9. Григор'єв В. Ю. Python 3: навчальний посібник - Харків: Фоліо, 2021. - 220 с.
10. Мещеряков В. П. Веб-програмування. HTML, CSS, JavaScript - Київ: Кондор, 2020. - 304 с.
11. Блінов В.І., Лещенко М.П. Інформаційні технології в освіті - К.: Центр учбової літератури, 2016. - 256 с.
12. Романовський О. Г. Системи дистанційного навчання - Харків: ХНУРЕ, 2019. - 180 с.
13. Кучерук В. Ю. Методичні вказівки до виконання кваліфікаційної роботи бакалавра - Умань: УНУС, 2024. - 35 с.
14. Кінг Д. Python. Програмування для початківців - Львів: Видавництво Старого Лева, 2022. - 276 с.
15. Хенсон Р. Мова розмітки XML. Короткий довідник - Київ: Діалектика, 2018. - 160 с.
16. Юрченко М. С. Проектування інтерфейсів користувача - Київ: Каравела, 2017. - 184 с.
17. Чекмарьов С. В. Архітектура веб-застосунків - Харків: Ранок, 2021. - 212 с.
18. Онлайн-курс: "Розробка веб-додатків на Flask". [Prometheus] - https://courses.prometheus.org.ua/
19. Стаття: "GIFT to XML: швидке створення тестів у Moodle", журнал Цифрова освіта, №4, 2023, с. 12-16.