МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРИВОРІЗЬКИЙ ДЕРЖАВНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ
Фізико-математичний факультет
Кафедра інформатики та прикладної математики

«Допущено до захисту»
Завідувач кафедри
_____ Моісеєнко Н. В.
«___» _____ 2024 р.

Реєстраційний № _____
«___» _____ 2024 р.

КОМПЛЕКС ЦИФРОВИХ ОСВІТНІХ РЕСУРСІВ НАВЧАННЯ МАТЕМАТИКИ В 6 КЛАСІ

Кваліфікаційна робота студента групи І-20
ступінь вищої освіти «бакалавр»
спеціальності
014.09 Середня освіта (Інформатика)
Кавецького Андрія Олександровича

Керівник
к. пед. н., доцент Шокалюк С. В.

Оцінка:
Національна шкала _____
Шкала ECTS ___ Кількість балів _____

Голова ЕК _____ _____
Члени ЕК _____ _____
_____ _____
_____ _____

Кривий Ріг – 2024

ЗАПЕВНЕННЯ

Я, *Кавецький Андрій Олександрович*, розумію і підтримую політику Криворізького державного педагогічного університету з академічної доброчесності. Запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Я не надавав і не одержував недозволену допомогу під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають покликання на відповідне джерело.

Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Криворізького державного педагогічного університету ознайомлений. Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

ЗМІСТ

ВСТУП

У сучасному світі, де технології розвиваються з неймовірною швидкістю, освіта в галузі математики стає все більш важливою. Математика є основою для багатьох наукових дисциплін, включаючи фізику, хімію, біологію, інформатику та інженерію. Вона відіграє ключову роль у розвитку критичного мислення та логічних навичок. Однак, навчання математики може бути складним процесом, особливо для молодших школярів. Саме тому розробка комплексу цифрових освітніх ресурсів навчання математики в 6 класі є актуальним та важливим дослідженням.

Метою даного дослідження є розробка інтерактивного програмного комплексу для автоматизації процесу оцінювання знань з математики учнів 6 класу, що базується на використанні Python і бібліотеки CustomTkinter. Система повинна забезпечувати зручне створення, проведення та аналіз тестів, а також підвищувати точність і ефективність оцінювання знань. Кінцевий результат полягає в наданні вчителям та учням інструменту, який спрощує процес тестування та знижує ймовірність помилок при оцінюванні.

Для досягнення сформульованої мети було визначено такі *завдання*:

1) розробити архітектуру та інтерфейс програмного комплексу;

2) створити навчальний контент та інтерактивні завдання;

3) провести тестування програмного комплексу та внести необхідні правки;

4) документувати розроблений програмний комплекс;

5) розробити зрозумілу та докладну інструкцію користувача для оптимального використання програмного комплексу.

Об’єктом дослідження є процес автоматизованого тестування знань учнів 6 класу, що включає в себе створення тестів, їх проведення, автоматичну перевірку результатів та аналіз отриманих даних. Особлива увага приділяється інтерактивним елементам системи, що роблять її зручною та ефективною для користувачів.

Предметом дослідження є програмні комплекси для автоматизованого тестування знань з математики учнів 6 класу.

У дослідженні були використані *методи* системного аналізу, проектування програмного забезпечення та тестування готового програмного комплексу. Крім того, буде проведено огляд наукової літератури для аналізу існуючих рішень та визначення кращих практик.

Практичне значення одержаних результатів полягає в розробці ефективного навчально-тренувального програмного комплексу, який враховує специфіку навчання математики учнів 6-го класу і може бути використаний в освітніх установах для підвищення якості навчання математики. Він також може допомогти учням краще зрозуміти математичні концепції та покращити їхні навички вирішення задач.

Цей кваліфікаційний проект є важливим кроком у підвищенні якості математичної освіти, і ми сподіваємося, що наша робота принесе значний внесок у цю галузь. Покладаю надії, що ця робота допоможе учням краще зрозуміти математику, покращити їхні навички вирішення задач та підвищити їхню мотивацію до навчання. Також сподіваюсь, що даний продукт допоможе вчителям ефективніше викладати математику, забезпечуючи їм інструменти для моніторингу прогресу учнів та адаптації освітнього процесу до індивідуальних потреб кожного учня.

Зрозуміло, що розробка такого продукту є великим викликом. Однак, я покладаю надії, що моя робота може зробити значний внесок у підвищення якості освіти в галузі математики. Сподіваюсь, що дана робота буде корисною для учнів, вчителів, батьків та всіх, хто зацікавлений у підвищенні якості математичної освіти.

РОЗДІЛ 1. АНАЛІЗ ПРОЕКТНОЇ ОБЛАСТІ Й ПОСТАНОВКА ЗАДАЧІ

1.1. Аналіз існуючих навчальних програмних комплексів з математики

Важливим етапом розробки комплексу цифрових освітніх ресурсів навчання математики в 6 класі “Математика-6” є аналіз існуючих програмних комплексів з математики. Це допомагає зрозуміти поточний стан ринку, визначити сильні та слабкі сторони існуючих продуктів та визначити можливі напрямки для покращення.

Далі буде оглянуто популярні платформи для вивчення математики, а деякі з них і взагалі включають в себе і інші предмети. Вони включатимуть і закордонні, і вітчизняні розробки.

Закордонні розробки

Mathletics [17] – це популярна онлайн-програма, розроблена австралійською компанією 3P Learning. Вона призначена для учнів від 5 до 16 років і пропонує персоналізоване навчання через інтерактивні завдання та захоплюючі ігри. Mathletics надає автоматичне оцінювання та звіти про прогрес учнів, що є важливими інструментами для вчителів. Програма платна, з пробним періодом, але відсутність української локалізації може ускладнити її використання для українських учнів, які не володіють англійською мовою. Проте, вона може бути адаптована до української програми завдяки своїм міжнародним освітнім стандартам.

IXL [13] – онлайн-платформа, створена американською компанією IXL Learning. Вона пропонує детальні навчальні матеріали та адаптивні вправи для учнів різного віку, включаючи 6 клас. Платформа є платною з обмеженим безкоштовним доступом. Відсутність української локалізації може бути перешкодою для її використання, але для учнів зі знанням англійської мови IXL може бути корисною. Вона відповідає широким міжнародним стандартам освіти.

Khan Academy [14] – безкоштовна освітня платформа, розроблена американською некомерційною організацією Khan Academy, Inc. Вона пропонує відеоуроки та інтерактивні завдання з різних предметів, включаючи математику. Часткова українська локалізація робить Khan Academy доступною для українських учнів. Матеріали платформи охоплюють багато тем математики і можуть бути адаптовані до української шкільної програми, що робить її дуже корисною для учнів 6 класу.

Prodigy [20] – гейміфікована освітня платформа, розроблена канадською компанією Prodigy Education. Вона пропонує інтерактивні математичні завдання у формі гри, що робить навчання цікавим для дітей. Платформа є безкоштовною з можливістю придбання додаткових платних функцій. Відсутність української локалізації може бути значною перепоною, але для учнів зі знанням англійської мови Prodigy може бути дуже корисною. Вона відповідає міжнародним стандартам освіти і може бути адаптована до української програми.

Вітчизняні розробки

МійКлас (MiyClas) [2] – українська освітня платформа, розроблена командою МійКлас. Вона пропонує інтерактивні завдання, відеоуроки та тести, які повністю відповідають українській шкільній програмі. Платформа безкоштовна з можливістю придбання додаткових платних функцій. Завдяки повній українській локалізації, МійКлас є дуже корисною для учнів 6 класу, забезпечуючи відповідність національним освітнім стандартам.

На Урок (Naurok) [3] – освітня платформа, створена ТОВ "На Урок", призначена для українських школярів. Вона містить інтерактивні уроки, завдання, тести та відеоуроки, що відповідають українській навчальній програмі. Платформа безкоштовна з можливістю придбання додаткових платних функцій. Повна локалізація робить її дуже зручною та ефективною для українських учнів 6 класу.

EdEra [9] – українська платформа, яка пропонує онлайн-курси та інтерактивні підручники, розроблені командою EdEra. Вони відповідають

українській шкільній програмі і включають відеоуроки, інтерактивні завдання та тести. Платформа безкоштовна з можливістю придбання додаткових платних функцій. Завдяки повній локалізації, EdEra є дуже корисною для українських учнів 6 класу.

Всеосвіта (Vseosvita) [4] – українська освітня платформа, що надає різноманітні освітні матеріали, інтерактивні завдання та відеоуроки, розроблена командою Vseosvita. Всі матеріали відповідають українській навчальній програмі. Платформа безкоштовна з можливістю придбання додаткових платних функцій. Повна локалізація робить Vseosvita дуже зручною та ефективною для учнів 6 класу.

Аналіз існуючих програмних комплексів для учнів 6 класу показує, що успішні платформи мають кілька спільних рис: адаптивне навчання, інтерактивні завдання, відео-контент, гейміфікацію та зручні інструменти для вчителів. Використовуючи ці підходи, наш навчально-тренувальний комплекс “Математика-6” може забезпечити ефективне, захоплююче та персоналізоване навчання для учнів.

Всі ці програмні комплекси мають свої сильні та слабкі сторони, і вони можуть служити важливими джерелами натхнення для розробки комплексу цифрових освітніх ресурсів навчання математики в 6 класі.

1.2. Визначення вимог до програмного комплексу “Математика-6”

Виходячи з уже існуючих рішень, при розробці програмного комплексу “Математика-6” було визначено ряд ключових вимог, які повинен виконувати кінцевий продукт. Ці вимоги були визначені на основі аналізу потреб користувачів, а також з урахуванням специфіки навчання математики для учнів 6-го класу і тенденцій в IT сьогодення.

Автоматична генерація тестових завдань

Однією з ключових вимог до програмного комплексу є можливість автоматичної генерації завдань. Раніше, в курсовому проекті, було прийнято рішення про зчитування даних з JSON/XML файлів, але через певні проблеми,

про які буде далі сказано, було прийнято рішення відмовитись від цього і створити генератор завдань.

В попередній версії, яка була в моєму курсовому проекті, було реалізовано зчитування даних з JSON та/або XML файлів (на вибір користувача цього програмного продукту). Так як ці дані не зашифровані, школярі, якщо б знайшли папку розміщення цих файлів, змогли б переглянути їх вміст через звичайний блокнот і дізнатись правильні відповіді. Це б дало їм несправедливу перевагу над іншими в момент написання самостійної/контрольної, в перспективі б це призвело до систематизації списування правильних відповідей з файлів та в кінцевому підсумку б призвело до освітніх втрат через: незасвоєння матеріалу; невідповідність результатів контрольних робіт знанням учнів.

Генерація тестів різних типів вирішує вищезазначені задачі і явно покращує вже наявний освітній процес. Головною умовою для цього є доступ до класу інформатики і наявність електроенергії.

Обов’язковий графічний інтерфейс

Виходячи з уже існуючих рішень, однозначно можна сказати, що наш програмний комплекс повинен мати графічний інтерфейс користувача. Графічний інтерфейс значно спрощує використання програми, робить її більш привабливою та зрозумілою для користувачів. Він повинен бути інтуїтивно зрозумілим, зручним для використання та має відповідати сучасним стандартам дизайну. Для цього було реалізовано 2 теми для комплексу: темна і світла. Також, для зручності, було прийнято рішення дозволити учням переходити на сторінку тесту назад/вперед у разі сумнівів з його останньої відповіді. Однак, так як віджети будуються заново, відповідь буде стертою.

Робота в одному вікні

З побаченого в схожих рішеннях, прийнято рішення, що інтерфейс не має бути перевантаженим різними елементами. Саме тому тестова система має мати головне меню, де «реєструється» користувач, потім будуються віджети відповідно до завдань тесту, і все це в цьому ж вікні.

Для зручності, як було сказано раніше, можна змінювати тему на темну або світлу, а також перемикатись між сторінками тесту вперед/назад.

Іноді трапляється так, що діти помиляються у відповіді і хочуть відповісти на питання інакше. Буде додано можливість відповісти інакше на попереднє завдання, це знижує стресовий фактор для дітей і їхні відповіді будуть краще відповідати їхньому наявному рівневі знань з математики. Іншими словами, буде реалізована навігація між завдань тесту.

Можливість запису результатів в файл

Програмний комплекс повинен мати можливість запису результатів в файл. Це дозволяє користувачам зберігати свої результати для подальшого аналізу та відстеження прогресу, ведення статистики. Файл з результатами має містити інформацію про дату проведення і найменування користувача, тему самостійної/контрольної роботи, складність (у випадку самостійної роботи) і самі результати роботи.

Вибір теми та складності

З урахуванням особливостей інших рішень, можна констатувати, що програмний комплекс обов’язково повинен надавати можливість вибору теми та складності завдань. Це, як вже можна було бачити у порівнянні уже існуючих подібних рішень, дозволяє адаптувати процес навчання до індивідуальних потреб кожного учня, забезпечуючи оптимальний рівень складності та підтримуючи мотивацію до навчання.

Ці вимоги були визначені з урахуванням потреб користувачів та специфіки навчання математики. Вони слугуватимуть основою для подальшого проектування та розробки навчально-тренувального програмного комплексу “Математика-6”.

1.3. Постановка задачі для розробки комплексу

При розробці комплексу цифрових освітніх ресурсів навчання математики в 6 класі було визначено ряд ключових вимог, які повинен виконувати кінцевий продукт. Ці вимоги були визначені на основі аналізу

існуючих рішень, потреб користувачів (вчителів та учнів під час проходження практики), а також з урахуванням специфіки навчання математики для учнів 6-го класу.

Обов’язковий графічний інтерфейс

Програмний комплекс повинен мати графічний інтерфейс користувача. Графічний інтерфейс значно спрощує використання програми, робить її більш привабливою та зрозумілою для користувачів, аніж просто консольна програма. Інтерфейс повинен бути інтуїтивно зрозумілим, зручним для використання та має відповідати сучасним стандартам дизайну.

Автоматичне генерування завдань різних типів

Однією з ключових вимог до програмного комплексу є можливість автоматичного створення неповторних тестових завдань для самостійних і контрольних робіт. Автоматична генерація завдань гарантує неможливість списати готові правильні відповіді з тестових файлів за відсутності таких.

Генератори тестів мають генерувати звичайні тести, завдання з відкритою відповіддю та завдання на встановлення відповідності.

Робота в одному вікні

Згідно поставленим вимогам, програмний комплекс повинен працювати в одному вікні. Буде реалізоване головне меню та автоматична побудова віджетів (сторінок тестів).

Можливість запису результатів в файл

Програмний комплекс повинен мати можливість запису результатів в файл. Це дозволяє користувачам зберігати свої результати для подальшого аналізу та відстеження прогресу. Файл з результатами може містити інформацію про виконані завдання, дані відповіді користувача, час виконання та інші важливі параметри.

Вибір теми та складності

Програмний комплекс повинен надавати можливість вибору теми та складності завдань. Це дозволяє адаптувати процес навчання до

індивідуальних потреб кожного учня, забезпечуючи оптимальний рівень виклику та підтримуючи мотивацію до навчання.

Ці вимоги були визначені з урахуванням потреб користувачів та специфіки навчання математики. Вони слугуватимуть основою для подальшого проектування та розробки навчально-тренувального програмного комплексу “Математика-6”.

Висновки по розділу 1

Аналіз існуючих навчальних програмних комплексів з математики дозволив визначити ключові характеристики та функціональні можливості, які вони пропонують. Це допомогло зрозуміти, які аспекти можна покращити або модифікувати для нового програмного комплексу. Вивчення існуючих програмних комплексів, таких як Mathletics, IXL, Khan Academy, Prodigy, МійКлас, На Урок, EdEra, Всеосвіта, дало можливість оцінити їх сильні та слабкі сторони. Це включає в себе розуміння того, як вони взаємодіють з користувачами, та як вони використовують графічний інтерфейс для полегшення використання цих продуктів.

Визначення вимог до комплексу цифрових освітніх ресурсів навчання математики в 6 класі було важливим етапом, який допоміг визначити специфікації та очікування від нового продукту. Це включає в себе розуміння цільової аудиторії, її потреб та того, як новий комплекс може задовольнити ці потреби. Визначення вимог до програмного комплексу включає в себе визначення ключових функцій, які має виконувати програмний комплекс, та того, як він повинен взаємодіяти з користувачами. Це включає в себе визначення вимог до інтерфейсу користувача, вимог до даних, вимог до запису результатів в файл, а також вимог до вибору теми та складності.

Постановка задачі для розробки комплексу була критично важливою для визначення основних цілей та завдань, які має виконати новий програмний комплекс. Це включає в себе визначення ключових функцій, які має виконувати програмний комплекс, та того, як він повинен взаємодіяти з

користувачами. Постановка задачі включає в себе визначення основних цілей та завдань, які має виконати програмний комплекс, включаючи визначення ключових функцій, які має виконувати програмний комплекс, та того, як він повинен взаємодіяти з користувачами.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО КОМПЛЕКСУ "МАТЕМАТИКА-6"

2.1. Архітектура програмного комплексу

Проект складається з двох основних частин: основної програми і модуля-генератора тестів. Основна програма використовує бібліотеку CustomTkinter для створення графічного інтерфейсу користувача, тоді як модуль-генератор тестів відповідає за генерацію тестових запитань відповідно до обраної складності та режиму роботи.

Структура програмного комплексу. Основний модуль

Основна програма містить клас TestApp, який відповідає за управління всіма аспектами взаємодії з користувачем. Клас TestApp містить конструктор __init__(), який ініціалізує основні змінні та створює початкові віджети інтерфейсу.

Метод create_widgets() створює початкові елементи інтерфейсу, такі як поля введення для імені та прізвища, опції для вибору режиму та складності, а також кнопку для початку тестування.

Метод update_mode() відповідає за оновлення доступності меню вибору складності залежно від обраного режиму роботи.

Метод start_test() перевіряє введені дані користувача та ініціює генерацію тестових запитань за допомогою функції generate_tests() з модуля-генератора тестів. Він також починає відображати перше запитання.

Метод show_question() визначає, яке саме запитання необхідно показати користувачу, та викликає відповідний метод для відображення цього запитання.

Для відображення тестових питань використовується метод display_test_question(), який створює віджети для питань з вибором варіантів відповідей.

Метод display_open_question() відображає запитання з відкритою відповіддю, а метод display_matching_question() відповідає за відображення питань на встановлення відповідності.

Методи prev_question() та next_question() дозволяють користувачам переходити між запитаннями.

Метод next_question() також зберігає відповіді користувачів та викликає метод finish_test(), коли всі запитання завершені.

Метод finish_test() підраховує результати тесту, записує їх у файл та відображає результат користувачеві. Після цього викликається метод reset_app(), який скидає всі змінні та віджети до початкового стану для нового тесту.

Всі методи класу **TestApp** взаємодіють між собою, забезпечуючи функціональність програмного комплексу. Інтерфейс користувача, модуль обробки даних, модуль обробки завдань та модуль управління об'єднані в один клас.

Перемикання теми реалізоване через глобальну функцію toggle_theme(), яка змінює зовнішній вигляд додатку з світлої теми на темну та навпаки. Це досягається за допомогою бібліотеки CustomTkinter, яка дозволяє змінювати режим відображення програми.

Модуль-генератор тестів

Модуль-генератор тестів (функція generate_tests()) відповідає за створення тестових питань відповідно до обраної складності та режиму роботи. Ця функція повертає три списки: test_questions(), open_questions() та matching_question(), які містять відповідні типи запитань. test_questions() містить запитання з варіантами відповідей, open_questions() містить запитання з відкритими відповідями, а matching_question() містить запитання на встановлення відповідності.

Перспективи удосконалення програмного комплексу

В перспективі можна зробити конструктор тестових завдань. Це дозволить легко додавати інші теми для тестувань і легко робити нові модулі

для тестів. Також можна в перспективі зробити модуль-фідбекер. Принцип роботи такий: якщо в роботі дитини є помилки, він робить звіт для вчителя і учня про те, які були допущені помилки. Учневі надається інформація про те як більше не допускати подібних помилок. Це покращить збір статистики про можливості дітей і допоможе вчителям витрачати менше часу на відгуки по самостійним.

2.2. Функціональні вимоги до програмного продукту

Функціональні вимоги визначають, що саме повинен робити програмний продукт. Вони описують послідовність дій, які виконує система, щоб виконати конкретну задачу або досягти конкретного результату. Функціональні вимоги включають такі аспекти, як взаємодія користувача з системою, обробка даних, інтеграція з іншими системами та інше.

Програма повинна мати можливість введення імені та прізвища користувача перед початком тестування. Введені дані повинні бути перевірені на заповнення, обидва поля мають бути не порожніми. Користувач повинен мати можливість обрати один з двох режимів: "Самостійна робота" та "Контрольна робота". В залежності від обраного режиму, доступність меню вибору складності повинна змінюватись: у режимі "Контрольна робота" меню складності повинно бути недоступне.

Для режиму "Самостійна робота" користувач повинен мати можливість обрати складність тесту з трьох варіантів: "низька", "середня", "висока". Програма повинна генерувати тестові запитання відповідно до обраної складності та режиму. Генерація тесту повинна включати п'ять запитань з варіантами відповідей, два запитання з відкритими відповідями та одне запитання на встановлення відповідності. Програма повинна відображати поточне запитання з можливістю вибору відповіді у випадку тестових запитань або введення відкритої відповіді. Також програма повинна відображати прогрес тестування, показуючи номер поточного запитання з загальної кількості запитань.

Користувач повинен мати можливість переходити між запитаннями за допомогою кнопок "Попереднє" та "Наступне". Відповіді користувачів повинні зберігатися при переході між запитаннями. Програма повинна перевіряти правильність відповідей після завершення тесту. Для тестових запитань правильність визначається за обраним варіантом. Для запитань з відкритими відповідями правильність визначається за збігом введеного значення з правильною відповіддю з допустимою похибкою 0.01. Для запитань на встановлення відповідності правильність визначається за збігом обраних варіантів з правильними відповідями.

Після завершення тесту програма повинна відображати кількість правильних відповідей та загальну кількість набраних балів. Програма повинна зберігати результати тестування у файл з зазначенням дати, режиму, складності, імені та прізвища користувача, а також кількості набраних балів. Крім цього, програма повинна дозволяти користувачу перемикати тему інтерфейсу між світлою та темною. Після завершення тесту програма повинна мати можливість скидання стану для проведення нового тесту, включаючи очищення полів введення та віджети інтерфейсу.

2.3. Проектування інтерфейсу користувача

Проектування інтерфейсу користувача є ключовим етапом розробки програмного продукту. Інтерфейс користувача є основним каналом взаємодії з програмою, тому його створення має бути обдуманим і спрямованим на досягнення максимальної зручності та ефективності для користувачів.

Структура інтерфейсу користувача

Інтерфейс користувача програмного комплексу "Математика-6" спроектований для легкого доступу до всіх його функцій. Основні елементи інтерфейсу включають:

1. Меню реєстрації:

– Користувач вводить своє ім’я та прізвище (лейбли з підказками і поля для вводу).

– Обирає режим зі списку, самостійна або контрольна робота.

– Обирає складність зі списку (низька, середня і висока).

– Є кнопка з темою самостійної або контрольної роботи.

– Є кнопка для перемикання теми оформлення

2. Тест:

– Написи де дано тестове завдання.

– Радіокнопки з варіантами відповідей, якщо це тестове завдання.

– Поле для вводу, якщо це питання з відкритою відповіддю.

– Варіанти і списки відповідей, якщо це встановлення відповідності.

– Кнопки «Попереднє» і «Наступне» для навігації по завданням.

– Кнопка зміни теми оформлення.

3. Вікно результатів:

– Користувачу оголошуються результати тесту.

– Радіокнопки з варіантами відповідей, якщо це тестове завдання.

Технології розробки інтерфейсу користувача

Для створення графічного інтерфейсу користувача використано бібліотеку tkinter, та customtkinter. Tkinter дозволяє створювати різноманітні елементи інтерфейсу, обробляти події та надає широкі можливості для взаємодії з користувачем, а customtkinter надає більше можливостей в редагуванні інтерфейсу і тим самим дає можливість зробити його зручнішим.

Принципи проектування інтерфейсу користувача

Проектування інтерфейсу користувача враховує наступні принципи:

1. Простота:

інтерфейс повинен бути простим та зрозумілим для користувача, з усіма елементами, які є логічними та очевидними.

2. Зручність використання:

всі елементи повинні бути легко доступні, а процес використання програми – інтуїтивно зрозумілим.

3. Естетика:

застосування приємних кольорів, шрифтів та елементів дизайну для створення привабливого інтерфейсу.

4. Зміна тем оформлення:

інтерфейс повинен надавати можливість зміни теми оформлення застосунку для більшої зручності.

Висновки до розділу 2

У другому розділі визначено концептуальний і технічний фундамент майбутнього продукту. Зокрема, були розглянуті основні вимоги до даного програмного продукту, його функціональні характеристики та характеристики інтерфейсу, а також принципи та технології, які будуть використовуватися при розробці. В цьому розділі було висвітлено важливі аспекти, такі як обробка даних, взаємодія з користувачем.

На основі визначених вимог можна зробити важливий висновок, що програмний продукт "Математика-6" має бути багатофункціональним, зручним у використанні та дбайливо розробленим з огляду на освітні потреби учнів 6-го класу. Заснований на технологіях, які забезпечують якість та ефективність, продукт має за мету полегшити процес вивчення математики та надати вчителям та учням інструмент для ефективного освітнього процесу.

Архітектура проекту побудована таким чином, щоб забезпечити гнучкість, зручність використання та надійність у роботі. Клас TestApp відповідає за основну логіку додатку, управління інтерфейсом та взаємодію з користувачем. Функції генерації тестів реалізовані в окремому модулі test_generator, що забезпечує модульність та можливість розширення функціоналу в майбутньому.

Функціонал програми включає введення імені та прізвища користувача, вибір режиму та складності тесту, динамічне генерування тестових запитань, перевірку правильності відповідей та відображення результатів тестування. Крім того, програма зберігає результати тестування у файл для подальшого аналізу.

Інтерфейс користувача розроблений з урахуванням вікових особливостей учнів 6 класу. Він простий та зрозумілий, з чітко позначеними елементами вводу та навігації. Додаток забезпечує можливість перемикання між темною та світлою темами, що підвищує комфорт використання для різних користувачів.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО КОМПЛЕКСУ "МАТЕМАТИКА-6"

3.1. Опис розроблених модулів та компонентів. Реалізація

Розробка програмного продукту "Математика-6" базується на структурованому підході, де модулі та компоненти взаємодіють між собою, надаючи програмі високу ефективність та гнучкість. Опис розроблених модулів та компонентів дозволить зрозуміти архітектурний підхід до створення програмного продукту, а також окреслить їхню внутрішню логіку та функціональність.

В даній версії програмного комплексу "Математика-6" існує два модулі та файл результатів. На етапі розробки даного продукту було прийнято рішення створити генератор тестів. Це має ряд переваг перед роботою з файлами xml або json: через незашифровані файли можна було дізнатись правильні відповіді, а через майбутнє додавання тем, розміри цих файлів би зросли до надвеликих і їх опрацьовувати було б складно. Всі ці недоліки нівелює генератор тестів. На кожну тему можна зробити окремий генератор тестів, що робить можливості.

Далі буде надано код проекту. Спочатку буде надано код генератора тестів різних типів на тему “Відсотки”.

Генератор тестів. Імпорт і функція-генератор тесту:

```
import random

def generate_percentage_test_question(difficulty):
    if difficulty == 'низька':
        a = random.randint(1, 50) * 10
        b = random.randint(1, 10) * 10
    elif difficulty == 'середня':
        a = random.randint(1, 20) * 25
        b = random.randint(1, 20) * 5
    else:  # 'висока'
        a = random.randint(1, 1000)
        b = random.randint(1, 200)

    question = f"Скільки буде {b}% від {a}?"
    correct_answer = round(a * b / 100, 2)
```

```
        answers = [
            correct_answer,
            round(correct_answer * random.uniform(0.5, 1.5), 2),
            round(correct_answer * random.uniform(0.5, 1.5), 2),
            round(correct_answer * random.uniform(0.5, 1.5), 2)
        ]
        random.shuffle(answers)
        answers = [str(answer) + ('t' if answer == correct_answer else
'') for answer in answers]
        return question, answers
```

Ця функція призначена для створення звичайного тесту. Параметром функції generate_percentage_test_question(difficulty) виступає складність. Чим більша складність, тим більший розкид чисел і їх відсотків. Наприклад, на низькій складності це числа кратні 10 в діапазоні від 10 до 500 і відсотки кратні 10 в діапазоні від 10 до 100. Правильна відповідь вирахувана заздалегідь, а хибні просто мають випадкову похибку.

Генератор тестів. Функція-генератор завдання з відкритою відповіддю:

```
    def generate_percentage_open_question(difficulty):
        if difficulty == 'низька':
            a = random.randint(1, 50) * 10
            b = random.randint(1, 10) * 10
        elif difficulty == 'середня':
            a = random.randint(1, 20) * 25
            b = random.randint(1, 20) * 5
        else:  # 'висока'
            a = random.randint(1, 1000)
            b = random.randint(1, 200)

        question = f"Скільки буде {b}% від {a}?"
        correct_answer = round(a * b / 100, 2)
        return question, correct_answer
```

Ця функція призначена для створення тесту з відкритою відповіддю. Функція generate_percentage_open_question(difficulty) має відповідний параметр складності. Складність працює аналогічним чином до попередньої функції. Тут є тільки правильна відповідь. При тестуванні буде порівнюватись відповідь користувача з правильною відповіддю.

Генератор тестів. Функція-генератор завдання на встановлення відповідності:

```
    def generate_percentage_matching_question(difficulty):
        pairs = []
        if difficulty == 'низька':
            for _ in range(3):
                a = random.randint(1, 50) * 10
                b = random.randint(1, 10) * 10
                correct_answer = round(a * b / 100, 2)
                pairs.append((f"{b}% від {a}", correct_answer))
            for _ in range(2):  # Додаткові хибні варіанти
                a = random.randint(1, 50) * 10
                b = random.randint(1, 10) * 10
                incorrect_answer = round(a * b / 100, 2)
                pairs.append((f"{b}% від {a}", incorrect_answer))
        elif difficulty == 'середня':
            for _ in range(3):
                a = random.randint(1, 20) * 25
                b = random.randint(1, 20) * 5
                correct_answer = round(a * b / 100, 2)
                pairs.append((f"{b}% від {a}", correct_answer))
            for _ in range(2):  # Додаткові хибні варіанти
                a = random.randint(1, 20) * 25
                b = random.randint(1, 20) * 5
                incorrect_answer = round(a * b / 100, 2)
                pairs.append((f"{b}% від {a}", incorrect_answer))
        else:  # 'висока'
            for _ in range(3):
                a = random.randint(1, 1000)
                b = random.randint(1, 200)
                correct_answer = round(a * b / 100, 2)
                pairs.append((f"{b}% від {a}", correct_answer))
            for _ in range(2):  # Додаткові хибні варіанти
                a = random.randint(1, 1000)
                b = random.randint(1, 200)
                incorrect_answer = round(a * b / 100, 2)
                pairs.append((f"{b}% від {a}", incorrect_answer))

        random.shuffle(pairs)
        questions = [pair[0] for pair in pairs[:3]]  # Використовуємо
тільки перші 3 питання
        correct_answers = [str(pair[1]) for pair in pairs[:3]]
        options = [str(pair[1]) for pair in pairs]  # Всі варіанти
        random.shuffle(options)
        return questions, correct_answers, options
```

Ця функція створює завдання на відповідність. Так само, як і попередні, вона має параметр складності, який працює аналогічним чином. В цьому

завдання є 2 зайвих відповіді, що дещо підвищує рівень складності цього завдання.

Генератор тестів. Функція-генератор самостійної або контрольної роботи:

```
    def generate_tests(difficulty, mode):
        test_questions = []
        open_questions = []
        matching_question = []

        if mode == "самостійна робота":
            for _ in range(5):

test_questions.append(generate_percentage_test_question(difficulty))
            for _ in range(2):

open_questions.append(generate_percentage_open_question(difficulty))

matching_question.append(generate_percentage_matching_question(difficu
lty))
        else:  # "контрольна робота"

test_questions.extend([generate_percentage_test_question('низька')  for
_ in range(3)])

test_questions.append(generate_percentage_test_question('середня'))

test_questions.append(generate_percentage_test_question('висока'))

open_questions.extend([generate_percentage_open_question('середня') for
_ in range(2)])

matching_question.append(generate_percentage_matching_question('висока
')
```

Функція generate_tests(difficulty, mode) створює тести до самостійної або контрольної роботи. Вона має параметри difficulty і mode. Перший, параметр складності, приймає складність, але спрацює тільки у випадку якщо режим «Самостійна робота». У випадку з контрольною роботою, складність для завдань обирається автоматично: перші 3 тестові завдання мають низьку складність, четверте має середню і 5 має високу, 6 і 7 завдання мають середню складність і 8 має високу. Це оптимальний загальний рівень складності для контрольної роботи.

Програмний комплекс "Математика-6". Імпорти і оформлення:

```
import customtkinter as ctk
from tkinter import messagebox
import test_generator
import datetime

answers = {}
current_mode = "light"

def toggle_theme():
    global current_mode
    if current_mode == "light":
        ctk.set_appearance_mode("dark")
        current_mode = "dark"
        theme_button.configure(text="☀")
    else:
        ctk.set_appearance_mode("light")
        current_mode = "light"
        theme_button.configure(text="🌙")
```

В цьому коді, якщо ми імпортуємо необхідні нам модулі і визначаємо глобальні змінні, що відповідають за відповіді користувача і тему оформлення за замовчуванням. Далі бачимо функцію, що відповідає за оформлення нашого комплексу цифрових освітніх ресурсів навчання математики в 6 класі.

Програмний комплекс "Математика-6". Клас TestApp:

```
class TestApp:
    def __init__(self, master):
        self.master = master
        self.frame = ctk.CTkFrame(self.master)
        self.frame.pack(pady=20, padx=20, fill="both", expand=True)
        self.test_var = ctk.StringVar()
        self.difficulty_var = ctk.StringVar(value="низька")
        self.name_var = ctk.StringVar()
        self.surname_var = ctk.StringVar()
        self.current_test = None
        self.current_difficulty = None
        self.current_question = None
        self.score = 0
        self.questions_answered = 0
        self.mode = ctk.StringVar(value="самостійна робота")
        self.test_questions = []
        self.open_questions = []
        self.matching_question = []
        self.create_widgets()
```

В цьому коді ми створюємо клас TestApp і необхідний для нього метод-конструктор __init__(). Конструктор визначає всі необхідні поля цього класу.

Програмний комплекс "Математика-6". Клас TestApp. Метод створення віджетів

```
        def create_widgets(self):
            self.name_label = ctk.CTkLabel(self.frame, text="Ім'я:")
            self.name_label.pack(pady=5)
            self.name_entry         =         ctk.CTkEntry(self.frame,
textvariable=self.name_var)
            self.name_entry.pack(pady=5)
            self.surname_label        =        ctk.CTkLabel(self.frame,
text="Прізвище:")
            self.surname_label.pack(pady=5)
            self.surname_entry        =        ctk.CTkEntry(self.frame,
textvariable=self.surname_var)
            self.surname_entry.pack(pady=5)

            self.mode_label = ctk.CTkLabel(self.frame, text="Режим:")
            self.mode_label.pack(pady=5)
            self.mode_rb1       =       ctk.CTkRadioButton(self.frame,
text="Самостійна    робота",    variable=self.mode,    value="самостійна
робота", command=self.update_mode)
            self.mode_rb1.pack(pady=5)
            self.mode_rb2       =       ctk.CTkRadioButton(self.frame,
text="Контрольна    робота",    variable=self.mode,    value="контрольна
робота", command=self.update_mode)
            self.mode_rb2.pack(pady=5)

            self.difficulty_label      =      ctk.CTkLabel(self.frame,
text="Складність:")
            self.difficulty_label.pack(pady=5)
            self.difficulty_optionmenu = ctk.CTkOptionMenu(self.frame,
variable=self.difficulty_var, values=["низька", "середня", "висока"])
            self.difficulty_optionmenu.pack(pady=5)

            self.test_button        =        ctk.CTkButton(self.frame,
text="Відсотки", command=self.start_test)
            self.test_button.pack(pady=20)
```

Метод create_widgets() необхідний для нашого інтерфейсу. Він і створює всі графічні елементи в нашому вікні.

Програмний комплекс "Математика-6". Клас TestApp. Методи оновлення режиму і початку тесту:

```
        def update_mode(self):
            if self.mode.get() == "контрольна робота":
                self.difficulty_optionmenu.configure(state="disabled")
            else:
                self.difficulty_optionmenu.configure(state="normal")

        def start_test(self):
            name = self.name_var.get().strip()
            surname = self.surname_var.get().strip()

            if name == "" or surname == "":
                messagebox.showerror("Помилка", "Будь ласка, введіть
ім'я та прізвище")
                return

            self.current_difficulty = self.difficulty_var.get()
            self.test_questions,                    self.open_questions,
self.matching_question                                                =
test_generator.generate_tests(self.current_difficulty, self.mode.get())
            self.current_question = 0
            self.score = 0
            self.questions_answered = 0
            self.show_question()
```

Метод update_mode() працює наступним чином: при виборі контрольної роботи блокується вибір складності, інакше розблоковується. Метод start_test() починає тест на обрану тему та складність. Також тут є перевірки, бо обов’язково мають бути введені Ім’я та Прізвище.

Програмний комплекс "Математика-6". Клас TestApp. Метод показу питання:

```
        def show_question(self):
            if self.current_question < 8:
                for widget in self.frame.winfo_children():
                    widget.pack_forget()     #   тимчасово   приховуємо
віджети замість їх видалення

                # Відображаємо прогрес
                progress_label        =        ctk.CTkLabel(self.frame,
text=f"Завдання {self.current_question + 1} / 8")
                progress_label.pack(pady=10)

            if self.current_question < 5:
                question,                   options                   =
self.test_questions[self.current_question]
```

```
                self.display_test_question(question, options)
            elif self.current_question < 7:
                question,                correct_answer                =
self.open_questions[self.current_question - 5]
                self.display_open_question(question, correct_answer)
            else:
                questions,       correct_answers,       options       =
self.matching_question[0]
                self.display_matching_question(questions,
correct_answers, options)
```

Цей метод очищує (приховує) елементи GUI від попередніх віджетів, питань та варіантів відповідей. Виводить нове питання залежно від прогресу користувача. Відображає прогрес користувача.

Програмний комплекс "Математика-6". Клас TestApp. Метод показу тесту:

```
        def display_test_question(self, question, options):
            self.question_label       =       ctk.CTkLabel(self.frame,
text=question)
            self.question_label.pack(pady=10)
            self.answer_var = ctk.StringVar(value="None")
            for option in options:
                rb           =           ctk.CTkRadioButton(self.frame,
text=option.rstrip('t'), variable=self.answer_var, value=option)
                rb.pack(pady=5)

            self.nav_frame = ctk.CTkFrame(self.frame)
            self.nav_frame.pack(pady=10)
            if self.current_question > 0:
                self.prev_button    =    ctk.CTkButton(self.nav_frame,
text="Попереднє", command=self.prev_question)
                self.prev_button.pack(side=ctk.LEFT, padx=10)
            self.next_button      =      ctk.CTkButton(self.nav_frame,
text="Наступне", command=self.next_question)
            self.next_button.pack(side=ctk.RIGHT, padx=10)
```

Цей метод показує звичайне тестове питання. Він відображає саме питання, варіанти відповідей, кнопки для навігації вперед і назад по питанням.

Програмний комплекс "Математика-6". Клас TestApp. Метод показу питання з відкритою відповіддю:

```
        def display_open_question(self, question, correct_answer):
            self.question_label       =       ctk.CTkLabel(self.frame,
text=question)
```

```
            self.question_label.pack(pady=10)
            self.answer_entry = ctk.CTkEntry(self.frame)
            self.answer_entry.pack(pady=10)
            self.correct_answer = correct_answer
            example_label   =   ctk.CTkLabel(self.frame,   text="Приклад
відповіді: 50.75")
            example_label.pack(pady=10)

            self.nav_frame = ctk.CTkFrame(self.frame)
            self.nav_frame.pack(pady=10)
            if self.current_question > 0:
                self.prev_button    =    ctk.CTkButton(self.nav_frame,
text="Попереднє", command=self.prev_question)
                self.prev_button.pack(side=ctk.LEFT, padx=10)
            self.next_button      =      ctk.CTkButton(self.nav_frame,
text="Наступне", command=self.next_question)
            self.next_button.pack(side=ctk.RIGHT, padx=10)
```

Цей метод показує питання з відкритою відповіддю. Він відображає саме питання, поле для вводу, кнопки для навігації вперед і назад по питанням.

Програмний комплекс "Математика-6". Клас TestApp. Метод показу завдання на встановлення відповідності:

```
        def display_matching_question(self, questions, correct_answers,
options):
            self.matching_vars = []
            self.correct_answers = correct_answers

            for i, question in enumerate(questions):
                q_frame = ctk.CTkFrame(self.frame)
                q_frame.pack(fill=ctk.X, pady=5)
                q_label = ctk.CTkLabel(q_frame, text=question)
                q_label.pack(side=ctk.LEFT, padx=10)
                var = ctk.StringVar(value="Відповідь")
                self.matching_vars.append(var)
                dropdown  =  ctk.CTkOptionMenu(q_frame,  variable=var,
values=options)
                dropdown.pack(side=ctk.RIGHT, padx=10)

            self.nav_frame = ctk.CTkFrame(self.frame)
            self.nav_frame.pack(pady=10)
            if self.current_question > 0:
                self.prev_button    =    ctk.CTkButton(self.nav_frame,
text="Попереднє", command=self.prev_question)
                self.prev_button.pack(side=ctk.LEFT, padx=10)
            self.next_button      =      ctk.CTkButton(self.nav_frame,
text="Наступне", command=self.next_question)
            self.next_button.pack(side=ctk.RIGHT, padx=10)
```

Цей метод показує питання знаходження відповідності. Він відображає самі питання, списки з варіантами відповідей, кнопки для навігації вперед і назад по питанням.

Програмний комплекс "Математика-6". Клас TestApp. Методи навігації по завданням тесту:

```
    def prev_question(self):
            self.current_question -= 1
            self.show_question()

        def next_question(self):
            if self.current_question < 5:  # Тестові питання
                answers[(self.current_question,       "test")]       =
self.answer_var.get()

            elif self.current_question < 7:  # Відкриті питання
                user_answer = self.answer_entry.get().strip()
                answers[(self.current_question, "open")] = user_answer

            else:  # Встановлення відповідності
                user_answers = [var.get() for var in self.matching_vars]
                answers[(self.current_question,     "matching")]     =
user_answers
            self.current_question += 1
            if self.current_question < 8:
                self.show_question()
            else:
                self.finish_test()
```

Ці методи дозволяють нам переходити на попереднє або наступне питання.

Програмний комплекс "Математика-6". Клас TestApp. Метод завершення тесту:

```
        def finish_test(self):
            self.score = 0
            for q_num in range(8):
                if q_num < 5:
                    if answers.get((q_num, "test"), "").endswith('t'):
                        self.score += 1
                elif q_num < 7:
                    user_answer    =    answers.get((q_num,    "open"),
"").replace(",", ".")
                    try:
                        user_value = float(user_answer)
```

```
                        correct_answer                                  =
float(self.open_questions[q_num - 5][1])
                        if abs(user_value - correct_answer) <= 0.01:
                            self.score += 2
                    except ValueError:
                        pass
                else:
                    user_answers = answers.get((q_num, "matching"), [])
                    correct_answers = self.correct_answers
                    for ua, ca in zip(user_answers, correct_answers):
                        if ua == ca:
                            self.score += 1

            self.questions_answered = 8
            result_text = f"Ваша оцінка {self.score} із 12 балів."

            # Записуємо результат у файл
            with open("results.txt", "a") as f:
                f.write(f"{datetime.datetime.now()},
{self.mode.get()}, {self.current_difficulty}, {self.surname_var.get()},
{self.name_var.get()}, {self.score}/{self.questions_answered}\n")

            messagebox.showinfo("Результат", result_text)
            self.reset_app()
```

Метод завершення тесту можна описати таким чином:

1. Збереження відповіді: отримується відповідь на поточне питання та зберігається в словникі answers.

2. Розрахунок балів: проходиться по всіх питаннях у поточному тесті та складності, рахує кількість правильних відповідей.

3. Запис у файл: записує результати тесту у файл "results.txt", включаючи час, ім'я, прізвище, результат.

4. Вивід результатів: показує вікно з результатами тесту.

5. Скидання даних: скидає дані про поточний тест, бали, кількість відповідей.

Програмний комплекс "Математика-6". Клас TestApp. Метод завершення тесту:

```
        def reset_app(self):
            answers.clear()
            self.name_var.set("")
```

```
            self.surname_var.set("")
            self.difficulty_var.set("низька")
            self.current_question = None
            self.test_questions = []
            self.open_questions = []
            self.matching_question = []
            for widget in self.frame.winfo_children():
                widget.pack_forget()
            self.create_widgets()
```

Метод скидання інтерфейсу видаляє попередні питання та варіанти відповідей, показує початковий екран.

Після класу програми ми все це запускаємо за допомогою:

```
    if __name__ == "__main__":
        ctk.set_appearance_mode("light")
        ctk.set_default_color_theme("blue")

        root = ctk.CTk()
        root.geometry("500x700")
        root.title("Testing System")

        app = TestApp(root)
        theme_button = ctk.CTkButton(root, text="🌙",
command=toggle_theme)
        theme_button.pack(side="top", pady=10)

        root.mainloop()
```

3.2. Тестування та валідація програмного комплексу

Тестування та валідація програмного комплексу "Математика-6" важливі для перевірки його функціональності, взаємодії з користувачем, та забезпечення високої якості роботи. З огляду на невеликий розмір програми, було прийнято рішення використовувати мануальне тестування.

Мануальне тестування

Мануальне тестування використовується для перевірки різних аспектів програмного комплексу шляхом активного втручання тестувальника. Основні етапи мануального тестування включають:

1. Взаємодія з інтерфейсом користувача:

– перевірка зручності та інтуїтивного розуміння інтерфейсу;

– перевірка коректності відображення елементів інтерфейсу на різних роздільностях екрану.

2. Тестування функціональності:

– перевірка коректності створення та збереження тестів;

– введення різноманітних відповідей та перевірка правильності обробки;

– тестування вибору теми та складності завдань.

3. Валідація виведення результатів:

– перевірка правильності розрахунку та відображення результатів тесту;

– перевірка коректності виведення повідомлень про помилки.

4. Тестування продуктивності:

– здійснення тестування на пристроях з різними характеристиками для оцінки продуктивності;

– перевірка швидкості завантаження та відгуку програмного комплексу.

5. Тестування безпеки:

– перевірка наявності та ефективності заходів безпеки;

– тестування введення відповідей для виявлення можливостей обхіду системи.

Висновки з тестування

Мануальне тестування дозволило виявити та виправити деякі недоліки в роботі програмного комплексу. Взаємодія з інтерфейсом була зручною, і система ефективно обробляла введені дані користувачів. Під час тестування вдалося коректно створювати тести, отримувати та обробляти відповіді. Виведення результатів тесту також пройшло успішно, а повідомлення про помилки були відображені чітко та зрозуміло.

Загалом, програмний комплекс "Математика-6" відповідає функціональним вимогам та ефективно виконує свої завдання. Мануальне тестування допомогло виявити та виправити невеликі недоліки, що підвищило загальну якість продукту.

3.3. Інструкція користувача

Інструкція користувача надає детальний опис процесу використання комплексу цифрових освітніх ресурсів навчання математики в 6 класі "Математика-6". Нижче наведено інструкцію для користувача з використання основних функцій програми.

1. Запуск програми

1.1. Запустіть програму "Математика-6", подвійно натиснувши на відповідний ярлик на робочому столі чи в меню програм вашої операційної системи.

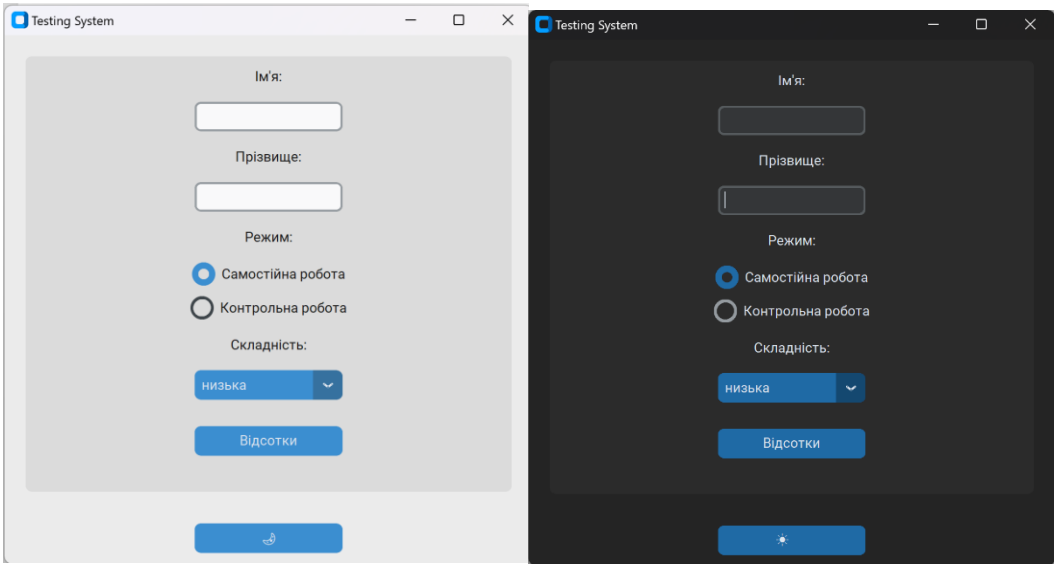

Рис. 3.1. Запущена програма (світла або темна тема)

1.2. У відкритому вікні введіть ваше прізвище та ім’я

1.3. Можна за бажанням змінити тему оформлення на темну

2. Обрання режиму та складності

2.1. Оберіть режим самостійної або контрольної роботи.

2.2. Оберіть рівень складності, якщо обрали самостійну роботу: "низька", "середня", або "висока".

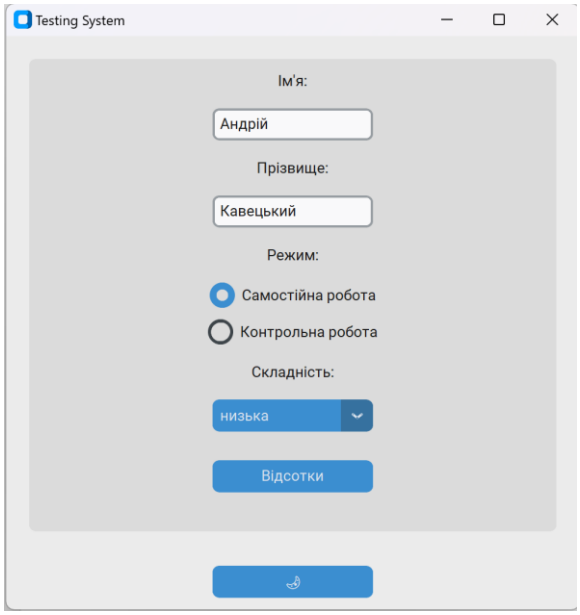

Рис. 3.2. Заповнена форма

3. Проходження тесту

3.1. Після обрання теми та складності, натисніть кнопку "Відсотки" для проходження роботи з цієї теми.

3.2. Вам буде представлено питання різних типів: тести, питання з відкритою відповіддю, завдання на встановлення відповідності.

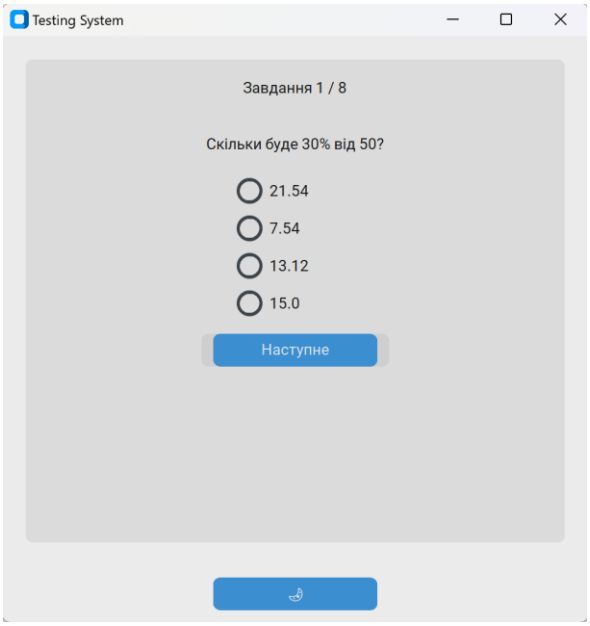

Рис. 3.3. Приклад першого питання

3.3. Оберіть одну з відповідей, натисніть “Наступне”, а якщо уже перейшли далі та маєте сумніви у попередній відповіді, ви можете натиснути кнопку “Попереднє” щоб повернутись.

3.4. Повторюйте цей процес для всіх тестових питань у тесті.

3.5. Далі вас чекають 2 питання з відкритою відповіддю. Вам буде надано приклад відповіді, вам треба в поле відповіді ввести відповідь за зразком.

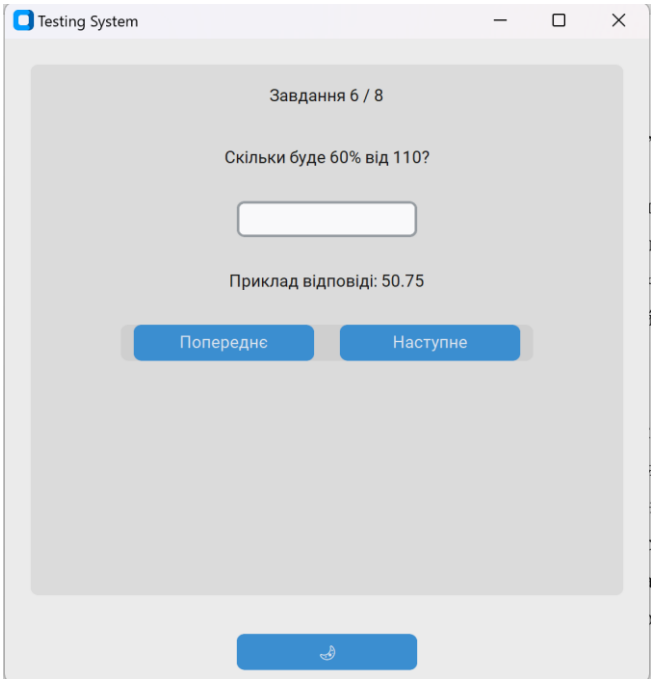

Рис. 3.4. Приклад питання з відкритою відповіддю

3.6. Останнє питання буде на встановлення відповідності (див. рис. 3.5). Напроти варіантів відповідей є списки з відповідями. Обирайте відповіді обережно так як є 2 зайвих.

4. Завершення тесту

4.1. Після відповіді на всі питання, натисніть кнопку "Наступне".

4.2. Результати вашого тесту будуть виведені на екрані, демонструючи кількість набраних балів (див. рис. 3.6).

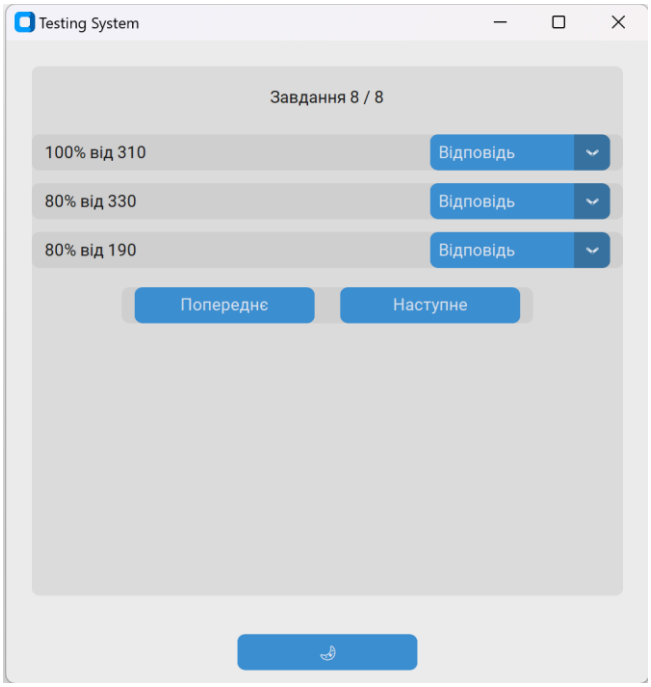

Рис. 3.5. Завдання на встановлення відповідності

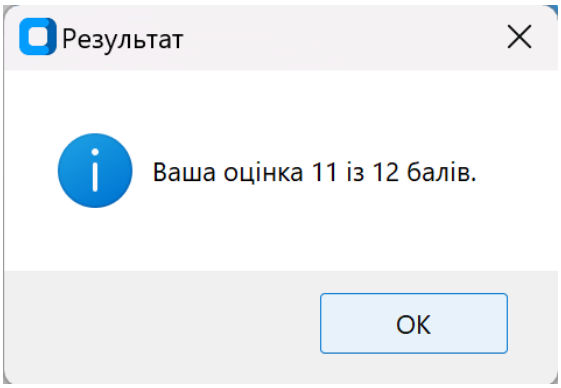

Рис. 3.6. Результати

5. Виведення результатів

Оцінка та результат тесту будуть автоматично записані у файл "results.txt" для подальшого перегляду.

```
2024-06-12 17:45:40.313811, самостійна робота, низька, Кавецький, Андрій, 11/12
```

Рис. 3.7. Приклад запису результату у файл

6. Вихід з програми

Після завершення використання програмного комплексу, натисніть кнопку "Вийти" для закриття програми.

Висновки до розділу 3

У цьому розділі ми детально розглянули архітектуру нашого програмного комплексу, реалізацію алгоритмів, процеси тестування та валідації, а також сформували інструкцію для користувача. Архітектура проекту була побудована з акцентом на модульність і зрозумілість, що дозволило розподілити обов'язки між компонентами та забезпечити легкість у підтримці і розширенні системи. Основний компонент, клас TestApp, відповідає за графічний інтерфейс користувача, створення віджетів, управління тестовими питаннями, обробку відповідей і зберігання результатів тестування.

Реалізація алгоритмів включала розробку функцій для генерації тестів різних типів та складності, що було досягнуто завдяки модулю test_generator. Ми забезпечили можливість створення питань з вибором варіантів відповіді, відкритих питань та питань на встановлення відповідності, що дозволяє проводити різноманітні типи оцінювання. Інтерфейс користувача був спроектований з врахуванням зручності та інтуїтивності, що полегшує взаємодію користувачів з системою.

Процеси тестування і валідації нашого продукту були спрямовані на забезпечення його стабільності та надійності. Ми тестували різні сценарії використання системи, включаючи введення даних користувачем, перехід між питаннями, обробку результатів та перемикання теми інтерфейсу. Всі помилки та неточності були своєчасно виявлені та виправлені, що дозволило забезпечити високий рівень якості кінцевого продукту.

На завершення, ми сформували детальну інструкцію користувача, яка описує всі основні функції та можливості системи. Інструкція включає кроки для початку роботи з програмою, вибору параметрів тестування, проходження тестів, отримання та збереження результатів, а також перемикання теми

інтерфейсу. Це забезпечує легкість у освоєнні системи навіть для нових користувачів, гарантуючи, що кожен зможе швидко зрозуміти і використовувати всі можливості програми.

ВИСНОВКИ

На початку були розглянуті існуючі навчальні програмні комплекси з математики, такі як Mathletics, IXL, Khan Academy, Prodigy, МійКлас, На Урок, EdEra та Vseosvita. Цей аналіз допоміг зрозуміти, які функції та можливості є важливими для користувачів, а також визначити сильні та слабкі сторони існуючих рішень. На основі цього аналізу ми визначили вимоги до програмного комплексу "Математика-6", включаючи необхідність інтерактивності, гнучкості у налаштуваннях, підтримки різних рівнів складності та можливості збереження результатів тестування. Постановка задачі для розробки комплексу включала створення інтуїтивно зрозумілого інтерфейсу, розробку різноманітних типів тестів і забезпечення надійності та стабільності роботи системи.

На етапі проектування було розроблено архітектуру програмного комплексу, яка забезпечує модульність та розширюваність системи. Були визначені функціональні вимоги до програмного продукту, що включають можливості створення та управління тестами, обробку відповідей користувачів та генерацію звітів про результати. Особливу увагу було приділено проектуванню інтерфейсу користувача, який мав бути простим у використанні та водночас функціональним. Інтерфейс включає можливість вибору режиму тестування, рівня складності та перегляду результатів.

Ми реалізували програмний комплекс, створивши різні модулі та компоненти, включаючи генератор тестів, обробник відповідей та інтерфейс користувача. Було проведено тестування та валідацію системи, що дозволило виявити та виправити помилки, забезпечити стабільність та надійність роботи комплексу. На завершенні була підготована інструкція користувача, яка детально описує всі основні функції системи та допомагає користувачам швидко освоїти роботу з програмою. Інструкція включає кроки для початку роботи, вибору параметрів тестування, проходження тестів та отримання результатів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Істер О. С. Математика. 6 клас: Підручник для загальноосвітніх навчальних закладів. Київ: Генеза, 2023. С. 15-70.
2. МійКлас. URL: https://www.miyklas.com.ua/p/matematika-nush-serednya-shkola/6-klas
3. На Урок. URL: https://naurok.com.ua/test/matematika/klas-6
4. Всеосвіта. URL: https://vseosvita.ua/test?s=&type=&cat=20&is_pay=&page=&sort=&class=6&title_only=&=
5. Al Sweigart. Automate the Boring Stuff with Python. 2022. P. 319-334.
6. Brett Slatkin. Effective Python: 90 Specific Ways to Write Better Python. 2015. C. 30-55.
7. David Beazley, Brian K. Jones. Python Cookbook. 2022. P. 141-171.
8. Doug Hellmann. Python 3 Module of the Week. URL: https://pymotw.com/3/xml.etree.ElementTree/index.html
9. EdEra. URL: https://courses.ed-era.com/courses/course-v1:EDERA-OSVITORIA+Math101+2019/about
10. Eric Matthes. Python Crash Course. 2019. P. 91-127.
11. Fredrik Lundh. The Python Standard Library. 2022. P. 70-172.
12. H. Bhasin. Python Basics: A Practical Introduction to Python 3. 2022. P. 164-180.
13. IXL: Мультипредметна онлайн платформа для дітей різного віку. URL: https://www.ixl.com/
14. Khan Academy. URL: https://www.khanacademy.org/math
15. Luciano Ramalho. Fluent Python. 2015. P. 63-93.
16. Mark Lutz. Learning Python. 2013. P. 835-846.
17. Mathletics: Математична онлайн платформа для дітей різного віку. URL: https://www.mathletics.com/uk/
18. Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser. Data Structures and Algorithms in Python. 2022. P. 31-50.

19. Python Software Foundation. Python Documentation. URL: https://docs.python.org/3/
20. Prodigy Math. URL: https://www.prodigygame.com/main-en/
21. Swaroop C.H. A Byte Of Python. 2021. P. 109-124.