МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему:

Система підтримки прийняття рішень при тестуванні зі складними закритими запитаннями

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма **«Аналітика даних»**

Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи АнД- 41


Морозов Р. О.
(прізвище та ініціали)

Керівник Гнатієнко Г. М.
(прізвище та ініціали)


к.т.н.
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол № 11 від 06.06.2022 р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ - 2022

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій
Кафедра інтелектуальних технологій
Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
інтелектуальних технологій
Іларіонов О.Є.
_____
"___"_____2021 р.

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

_____Морозову Ростиславу
Олеговичу_____
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)
Система підтримки прийняття рішень при тестуванні зі складними закритими запитаннями
_____
___

затверджена протоколом засідання кафедри від « 23 » грудня 2021 р. №4

2. Термін здачі студентом закінченого проекту (роботи) 29 травня 2022 року

3. Вихідні дані до проекту (роботи)
СППР тестування зі складними закритими запитаннями, створюється для визначення рівня підготовки осіб, які пройшли навчання за деякою дисципліною. Вхідними даними для СППР є тестові запитання, які користувач має можливість ввести в систему. На основі відповідей особи, яка проходить тестування, визначається оцінка результативності навчання._____
_____
_____
___

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
Вибір та аналіз предметної області, постановка задачі тестування зі складними закритими запитаннями, огляд систем підтримки задач тестування, проєктування архітектури системи, розробка системи, вибір тестових задач, тестування роботи системи_____
_____
___

5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій)
Огляд систем підтримки задач тестування, ілюстрація розв'язання задачі прийняття рішення при тестуванні, архітектура системи, програмна реалізація системи, презентація Power Point_____
_____
___

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Вибір та аналіз предметної області		
1	Постановка задачі тестування зі складними закритими запитаннями		
1	Огляд систем підтримки задач тестування: - дослідження по задачах тестування (хто цим займається і т.д.) - які є інструменти для тестування (інформаційні системи, їх особливості...) - які важливі елементи системи тестування слід врахувати, щоб наша система була досконалою (перевірка особистості студента, можливість введення тестів екзаменатором, справедливе визначення оцінки...) - яке математичне забезпечення застосовується в системах тестування (навести приклади...)		
2	Проектування архітектури системи		
3	Програмна реалізація системи		
3	Вибір тестових задач, тестування роботи системи		

7. Дата видачі завдання 15 лютого 2022 року

Керівник_____ _____ / Гнатієнко Г. М. /
(підпис) (ПІБ)

Завдання прийняв до виконання _____ / Морозов Р. О. /
(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1.	Вибір та аналіз предметної області	15.01-22.02	
2.	Постановка задачі тестування зі складними закритими запитаннями	23.02-25.02	
3.	Огляд систем підтримки задач тестування	26.02-03.03	
4.	Проектування архітектури системи	04.03-27.03	
5.	Програмна реалізація системи	28.03-14.04	
6.	Вибір тестових задач, тестування роботи системи	15.04-26.04	

Студент-дипломник __ _ / Морозов Ростислав Олегович /
(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи_ __ / Гнатієнко Григорій Миколайович/
(підпис) (ПІБ)

Анотація

Морозов Ростислав Олегович виконав випускну кваліфікаційну роботу на тему «Система підтримки прийняття рішень при тестуванні зі складними закритими запитаннями» за спеціальністю 122 – «Комп'ютерні науки».

У випускній кваліфікаційній роботі проведено огляд систем підтримки задач тестування, спроектовано архітектуру системи, програмно реалізовано систему підтримки прийняття рішень при тестуванні зі складними закритими запитаннями, що визначає рівень підготовки осіб, які пройшли навчання за деякою дисципліною.

Ключові слова: СППР, тестування, запитання, оцінка.

Summary

The degree project: «Decision support system for testing with complex closed questions» has completed by **Morozov Rostyslav** specialty 122 – «Computer Scienses».

In this final qualifying work, a review of the support system for testing tasks was conducted, an architectural system was developed, a system for supporting decision support in testing with complex closed-ended questions was determined, which determines the level of training of trained students in the discipline.

Keywords: DSS, testing, questions, evaluation.

ЗМІСТ

ВСТУП

Вже декілька років в Україні є дуже популярним онлайн-навчання у школах та вищих закладах освіти. Тож питання якісного дистанційного оцінювання знань учнів та студентів є як ніколи актуальним. За допомогою розробленої системи тестувань вчитель має змогу прийняти рішення щодо оцінки учня спираючись не лише на власні міркування, а на конкретні результати, що отримає учень після складання тесту. Таке оцінювання буде найбільш об'єктивним та обґрунтованим і тому не викличе запитань чи заперечень збоку учня.

Опрацювання і розробка тестування упродовж багатьох років перебуває у центрі уваги науковців усіх країн світу. Різні аспекти і проблеми контролю знань опрацьовуються багатьма українськими та європейськими дослідниками. На сьогоднішній день велике число публікацій присвячено дослідженню потуг і перспектив комп’ютерних технологій тестування рівня освіти учнів. Деталізоване вивчення та удосконалення процесів, пов’язаних із тестуванням, допомагатиме розвитку та прогресу рівню системи освіти загалом.

Мета роботи: створення системи підтримки прийняття рішень при тестуванні зі складними закритими запитаннями, що визначає рівень підготовки осіб, які пройшли навчання за деякою дисципліною.

У розділі 1 було проведено аналітичний огляд систем підтримки прийняття рішень при тестуванні з закритими запитаннями та сформовано постановку задачі.

У розділі 2 даної випускної кваліфікаційної роботи було спроектовано архітектуру системи.

У розділі 3 було детально описано код програми, інтерфейс системи і проведено тестування розробленої системи.

1 РОЗДІЛ. АНАЛІТИЧНИЙ ОГЛЯД

1.1. Вибір та аналіз предметної області

Предметною областю в даній дипломній роботі постає система підтримки прийняття рішень зі складними закритими запитаннями для тестування освітнього процесу.

Тестування – це інструмент, що розрахований для замірювання рівня навченості учня, який формується із системи тестових завдань, стандартизованого процесу проведення, обробки та дослідження результатів.

Тестування освітнього процесу має велику кількість плюсів:

– це всеохоплюючий інструмент, що дає змогу окреслити рівень знань учня по всій навчальній дисципліні в цілому і за окремих її частинах;

– воно є якісним та об'єктивним способом оцінювання тому, що стандартизовано процес проведення та перевірки покажчиків якості тестових завдань для всіх протестованих у класі;

– тестування є доволі точним інструментом, оскільки шкала оцінювання тесту залежить від кількості запитань, що до нього внесено, і може великою мірою змінюватися;

– це якісний інструмент, що ставить всіх учнів в рівні умови, використовуючи загальну процедуру та загальні критерії оцінювання;

– є економічно потужним оскільки головні витрати при цьому способі оцінювання мають одноразовий характер та є сильно меншими, аніж при письмовому чи усному контролі;

– є справедливим методом оцінювання рівня знань, що ставить всіх учнів в рівні умови під час контролю і під час оцінювання знань, значно усуваючи суб'єктивізм вчителя: за статистикою тестування дає змогу зменшити кількість апеляцій більше ніж в тричі.

Закрите запитання - це звичайне запитання, яке вимагає короткої відповіді. Ця коротка відповідь може бути одним словом або коротким реченням. Відповіді на закриті запитання, як правило, мають одну відповідь або обмежені варіанти, такі як Так / Ні або Правда / Неправда або виберіть одну з відповідей.

Система підтримки прийняття рішень (СППР) – комп'ютерна автоматизована система, що потрібна для допомоги і сприянню різних типів діяльності людини під час прийняття рішення згідно розв'язання структурованих чи неструктурованих проблем. Використатння СППР встановлює виконання ґрунтовного і об'єктивного дослідження предметної області під час прийняття рішень в непростих умовах [10].

Рішенням визнається умотивований набір дій зі сторони особи, яка обирає рішення (ОПР), спрямованих на об'єкт або систему керування, що дає перспективу привести даний об'єкт або систему до очікуваного стану чи здобути наміченої мети. Рішення є одним з видів інтелектуального процесу та демонстрацією волі людини. Особливими рисами рішення є: – можливість вибору із переліку альтернативних варіантів: за недостачі альтернатив, відсутній також вибір, тому, немає і рішення; – присутність цілі: безцільний вибір не вважається як рішення; – необхідність вольового акту ОПР під час вибору рішення, оскільки вона створює рішення під час боротьбі мотивів та думок.

Прийняття рішення – це процес обрання максимального преференційного рішення із множини допустимих рішень чи впорядкування множини рішень. Прийняття рішень можливе на підставі знань про об'єкт управління, процеси, які у ньому існують та можуть чинитися із перебігом часу, а також за наявності множини показників, що характеризують ефективність і якість прийнятого рішення.

1.2. Постановка задачі

Нехай задано множину варіантів відповідей $a_i \in A, \quad i \in I = \{1,...,n\}$, кількість яких дорівнює $n, \ n = |A|.$ Частина відповідей $n_1, \ n_1 < n,$ є вірними та вони складають підмножину $A^1, \ A^1 \subset A,$ а інша частина відповідей $n_0, \ n_0 < n,$ є помилковими і вони складають підмножину $A^0, \ A^0 \subset A,$ причому $A^1 \cup A^0 = A.$ Крім того, будемо вважати, що усі запропоновані тестовим завданням відповіді $a_i \in A, \ i \in I,$ є рівноцінними.

Для великої кількості тестових завдань така постановка є природньою і звичною. Наприклад, обрати серед заданих варіантів чисел ті, що є дільниками заданого числа. Можна навести багато варіантів такого роду завдань. Тобто, такий підхід має місце у звичайному житті та задача його формалізації під час тестування є актуальною. Особливість таких задач полягає в тому, що вони віддзеркалюють популярну думку: «Скільки людей – стільки думок». Тому розв'язок повинен бути обґрунтованим такою мірою, яку підказує логіка його побудови, регламент оцінювання, сформована організаторами тестування, здоровий глузд тощо.

Для визначеної постановки задачі тестування треба застосовувати алгебраїчний підхід до визначення результатів оцінювання, який ефективно використовується в теорії прийняття рішень і під час застосування технологій експертного оцінювання. При алгебраїчному підході формалізація завбачає розрахунок і обґрунтування всіх можливих варіантів відповідей. Найбільша кількість балів за правильно вибрану підмножину варіантів дорівнює B. Кількість балів за правильно обраний елемент підмножини правильних відповідей $b = B / n_1$.

1.3. Огляд систем підтримки задач тестування

1.3.1. Дослідження по задачах тестування

Загальне і об'єктивне оцінювання знань через надання ефективного зворотного зв'язку є запорукою якісного навчання. Різноманітність систем тестування вражає. Вони відрізняються за великою кількістю критеріїв: формами тестових запитань, видами тестів, взаємодією із користувачем, методами оцінювання ефективності результатів і параметрами інтерфейсу [12].

Розроблена для певного навчального курсу чи конкретної предметної області окрема підсистема тестування має тільки специфічний набір функцій та параметрів, що допомагає вирішити конкретну проблему. Це призвело до розробки у світі величезного різноманіття вузькоспеціалізованих систем тестування, несумісних між собою, із мінімальною специфічною функціональністю. Майже кожна система автоматизованого навчання включає у себе власний варіант підсистеми тестування. Такі розробки сильно схожі між собою, також більшість із них реалізують тільки найлегші сценарії тестування, залишаючи комплексний математичний підхід до оцінки знань на майбутнє [12].

Важливою задачею є вибір технології і архітектурного підходу, який би дозволив створити якісну платформу побудови систем тестування та легкого їх налаштування на будь-який навчальний курс будь-якої навчальної платформи без обмеження на види тестування чи на типи тестових питань, формат тестів. Така платформа повинна мати чітко визначений опис інтерфейсів та основних компонентів системи, для подальшої тісної інтеграції з будь-якими системами навчання без необхідності вносити зміни в вихідні коди системи тестування [12].

Сучасний етап розвитку і функціонування тестового контролю характеризується застосуванням до вирішення психолого-педагогічних задач методології латентно-структурного аналізу (LSA). Одним з напрямків LSA є

Item Response Theory (IRT) – математична теорія параметричної оцінки тестових завдань і тих, хто проходить тестування. Відповідно до цієї теорії встановлено, що між результатом виконання, що спостерігається, і латентним параметром учасників тестування є деяка залежність, яку можна виразити за допомогою функції. Для IRT характерне прагнення до фундаментального теоретичного підходу і разом з цим до коректного розв'язання низки практичних задач [19].

1.3.2. Інструменти для тестування

З настанням XXI століття, в один момент у різних країнах, значні за розмірами колективи вчителів, науковців, викладачів, використовували багато матеріальних ресурсів, створюючи цифрові інструменти, що цілеспрямовані на покращенні якості навчання головних предметів, проведення тестування, опитування або відео спілкування. Велика кількість вчителів організовують свою роботу із учнями або студентами на базі запропонованих закладами освіти онлайн платформ чи популярних месенджерів WhatsApp або Viber, Facebook або Telegram. В даній ситуації ці додатки-месенджери є якісними і ефективними для організації консультативного навчання, надання порад, повідомлення результатів академічних досягнень; надають змогу дистанційного навчання.

Проте часто для організації якісної роботи під час уроку вчителю необхідні інші засоби, цифрові інструменти, що будуть корисними для реалізації певних завдань, враховуватимуть особливості предмету і урізноманітнять активність учнів чи студентів під час заняття. Таких освітніх інструментів на сьогоднішній день існує велика кількість, частина з яких є гарно адаптовані для українського користувача (багато з них є на англійській мові); низка ресурсів є абсолютно безкоштовними чи частково платними.

1. Kahoot - це безкоштовний навчальний інструмент, за допомогою якого можна реалізовувати інтерактивні заняття і перевірку знань студентів чи учнів використовуючи онлайн-тестування. Інструмент Kahoot об’єднує гру і навчальний процес. Сервіс Kahoot пасує для вивчення будь-якої навчальної дисципліни студентів університетів або предмету учнів усіх вікових категорій закладів загальної середньої освіти. У цього інструменту багато плюсів – він якісний, соціальний, доволі зрозумілий і привабливий для користувача. Сервіс Kahoot був розроблений в 2013 році як інструмент, що призначений для якісної реалізації усього інтерактивного: тестів, опитувань та розглядів, тощо. Софт, створений на платформі онлайн сервісу, отримав цікаву назву «кахути», що позначають утворений навчальний матеріал. У розроблені самостійно тести можна додавати навчальні відео та зображення, а процес реалізації нового завдання займає пару хвилин. Завдячуючи цьому, великою мірою зменшується час на підготовку до заняття [8].

Одним з плюсів цього інструменту є система заготовлених тестів, завантажених іншими користувачами сервісу. Тому є можливість використовувати тести, що створені вами самостійно чи взяті у завдання інших розробників.

Суттєво, що результати тестувань завантажуються в вигляді переліку у таблиці MS Excel та вчитель має можливість реалізовувати детальний аналіз результатів досягнень учнів. Онлайн сервіс Kahoot! якісний інструмент для створення навчальних тестів, вікторин, обговорень та опитувань. Інтерактивні навчальні тести складаються із низки запитань з декількома варіантами відповідей. Ці тестові форми роботи можуть бути застосовані в навчальному процесі із метою випробування навичок учнів чи студентів (ґрунтовне оцінювання), чи також для підготовки класу до підсумкової контрольної чи для реалізації складової командної роботи під час заняття. Kahoot! дає можливість якісно і гарно звірити знання учнів із теми та суттєво зменшує час, витрачений вчителем на підготовку до тестування. Окрім цього, інструмент може бути

небезкорисним керуючому і педагогічному колективу навчального закладу для усяких форм методичної, наукової і організаційної роботи. Kahoot доречно застосовувати для проведення контрольних та самостійних робіт, невеликих тестувань, дебатів і командних обговорень, із метою звичайного або формувального оцінювання та аналізу. Для застосування Kahoot у навчальній діяльності потрібно розуміти, яке завдання вчитель ставить перед класом реалізуючи тестування, та, виходячи із цього, необхідно складати навчальні запитання [8].

2. Форми Google – це онлайн сервіс для реалізації тестувань, опитувань, форм авторизації для подій та отримання зворотнього зв'язку. Наприклад, форма Гугл водночас дає змогу реалізовувати звіти. Усю інформацію, що заповнюють користувачі, можливо автоматично змінити в Google Таблиці. Завдяки такій можливості результати Гугл Форми можливо якісно проаналізувати [7].

Користувачу не треба нічого видумувати, інструмент продуманий до деталей. Гугл Формс тести містять усі необхідні функції для опрацювання питань за опитуваннями із невеликими затратами часу та сил. Для користування Гугл Форм необхідно зробити вхід в обліковий аккаунт Google.

Інструмент має простий та зрозумілий дизайн. Для користувачів наявні різноманітні приклади та шаблони Google Forms, на підґрунті яких можливо реалізовувати власні варіанти тестів, форм для реєстрації на заходи і багато іншого. Сервіс має великий функціонал, даючий змогу розробки тесту в Гугл Формі чи будь-якого іншого опитувальника із застосуванням різноманітних типів питань, стилізованих під проекти або унікальний стиль компанії.

Основні переваги інструменту:

- доступність – будучи у різних місцях та у будь-який час, користувач має змогу користуватись інструментом. Створені тести для опитування завантажуються на Google Диск;

- адаптивність – реалізація, зміна та перегляд тестів відкрито на усіх типах девайсів;
- простота користування – полягає у комфортності користування під час всіх етапів, від початку до кінця створення тесту;
- комфортабельність аналізу – після створення тестування інформація за замовчуванням обробляється інструментом, та користувач має змогу здобути підготовлену статистику результатів.
- унікальний дизайн – для оформлення можливо вибрати готові теми або приєднати власні ілюстрації;

3. Proprofs створює тести на всякий смак – можливо реалізувати на вибір один чи декілька варіантів, попрохати вставити пропущене слово чи написати відкриту відповідь. Інструмент дає змогу завантажувати в завдання текстові файли і презентації, файли PDF, і також зображення, аудіо та відео файли. Виконавши роботу над тестуванням, можливо покинути його у відкритому доступі на сайті Proprofs чи завантажити на власний аккаунт.

При тому, що інструмент безкоштовний, можливості Proprofs збільшуються у платних послугах. Вчителям потрібно звернути увагу на послуги Basic і Professional. Перша дає доступ до усіх головних можливостей інструменту та дає змогу реалізовувати велику кількість тестувань за $20 в місяць; друга дає змогу поєднувати учнів у приватні групи та вартуватиме $40. Нові користувачі мають змогу задарма користуватися усіма функціями Proprofs на протязі 15 днів після авторизації вперше.

1.3.3. Важливі елементи системи тестування слід врахувати, щоб наша система була досконалою

Завдання в тестовому форматі повинні відповідати таким вимогам:

- однаковість правил оцінки відповідей;

- стисліcть;

- технологічність;

- точність форми;

- коректність наповнення;

- простота інструкції для усіх учнів;

- зрозуміла форма висловлювання;

- коректність розташування питань завдання;

- наявність необхідного місця для відповідей;

- наявність інструкцій форми та змісту завдання.

Інноваційними в представленому визначенні постають вимоги якості змісту і технологічності завдань. Перше із них - умова предметної правильності сформульованого змісту завдань. Звершення цієї умови підпорядковане від компетентності розробника завдань і від експертів, що перевіряють змістовну правильність інформації, закладених у основу завдання.

1.3.4. Математичне забезпечення, що застосовується в системах тестування

Для оптимізації контролю знань потрібно використовувати дані процедури:

– стандартизація балів;

– однозначність тлумачення питань;

– гарантування інформаційної одностайності оцінювання;

– гарантування системності оцінювання;

– однозначність генерування оцінки спираючись на реультатів.

Використання даних процедур зв'язана із формалізацією видів питань, процесами їх оцінювання і критеріями формування загальної оцінки.

Нехай загальна кількість запитань, що визначають предметну область, згідно із онтологією, та створюють логічну схему тестування, дорівнює n [11].

Всі вони розподілені на k типів, в залежності від того, який тип відповіді повинен відповідати запитанню [11].

Множину запитань опишемо наступним чином:

$$Q = (Q_1, Q_2, ..., Q_n) = (Q_1, Q_2, ..., Q_{n_1}, Q_{n_1+1}, ..., Q_{n_{k-1}}, Q_{n_{k-1}+1}, ..., Q_{n_k})$$

Дихотомічні запитання з вибором альтернативної відповіді формально представляються таким чином:

$$Q^1 = \langle Q_i, A_i(\{0,1\}), 1 \rangle,$$

де $Q^1 -$ підмножина дихотомічних запитань із вибором альтернативної відповіді, $Q_i^1 -$ запитання першого типу, $Q_i^1 \in Q^1$, $i = \{1, ..., n_1\}$, $A_i^1 -$ множина можливих відповідей на $i -$ е запитання, 1 – кількість відповідей, які необхідно вибрати з множини значень A_i^1, $i = \{1, ..., n_1\}$.

Запитання із вибором однієї вірної відповіді формально представляються у вигляді:

$$Q^2 = \langle Q_i, A_i(a_1, a_2, ..., a_{p_1}), 1 \rangle,$$

де $i = \{n_1 + 1, ..., n_2\}$, $a_j -$ можливі варіанти відповідей на запитання з вибором однієї правильної відповіді, $j = \{1, ..., p_i\}$, $p_i -$ кількість варіантів відповідей.

Запитання з вибором декількох правильних відповідей визначаються таким формалізмом:

$$Q^3 = \langle Q_i, A_i(a_1, a_2, ..., a_{p_i}), d_i \rangle,$$

де $i = \{n_2 + 1, ..., n_3\}$, $a_j -$ можливі варіанти відповідей на запитання із вибором декількох правильних відповідей, $d_i -$ можлива кількість відповідей, причому $d_i = const \in \{2, p_i\}$, що визначається апріорно, або $d_i -$ змінна величина, $d_i \in \{1, p_i\}$.

Запитання на встановлення відповідності, де частини однієї множини треба виставити у відповідність до частин іншої множини, визначають завданнями на встановлення відповідності, та формально описуються таким чином:

$$Q^4 = \langle Q_i, A_i(A_i^1, A_i^2), f(D_i^1, D_i^2) \rangle,$$

де, $i = \{n_3 + 1, ..., n_4\}$ $A_i^1,\ A_i^2 -$ множини елементів, між якими слід встановити відповідність, $f : D_i^1 \to D_i^2 -$ відповідь у вигляді встановленої відповідності між множинами $A_i^1,\ A_i^2$.

Запитання на встановлення правильної послідовності, де треба установити правильну послідовність дій чи слів (варіантів відповідей тощо), використовуються в тестових завданнях на встановлення правильної послідовності, та описуються таким формалізмом:

$$Q^5 = \langle Q_i, A_i(a_1, a_2, ..., a_{p_i}), R_i \rangle,$$

де $i = \{n_4 + 1, ..., n_5\}$, $a_j -$ варіанти відповідей на запитання на встановлення правильної послідовності множини елементів, для яких слід

встановити правильну послідовність, тобто побудувати ранжування $R_i = a_{i_1} \succ a_{i_2} \succ ... \succ a_{i_{n_5}}$.

Для оцінювання правильності встановленої послідовності використовувалася – метрика Кука неспівпадання рангів (місць, позицій) елементів списку

$$d^K(R^0, R^*) = \sum_{i \in I} \left| r_i^0 - r_i^* \right|,$$

де $r_i^0 -$ ранг $i-$го елемента списку у еталонному ранжуванні елементів списку R^0, $r_i^* -$ ранг $i-$го елемента списку у ранжуванні елементів R^*, заданому опитуваним;

Приклад:

Нехай розглядається ситуація побудови тестового завдання з такими параметрами: $n = 5,\ n_1 = 3,\ n_0 = 2.$ Без зменшення загальності, будемо вважати, що у варіантах відповідей на тестове запитання перші три відповіді є вірними, а останні дві – помилковими. Тобто запитання поставлено так, що вірні та помилкові варіанти відповідей можна представити у вигляді вектора: (1,1,1,0,0).

У векторах, які відповідають відповідям респондента, елементи будемо позначати таким чином: респондентом вибрано вірну відповідь – «1», вибрано невірну відповідь – «0», відсутність відповіді – «*» [11].

При побудові цього ілюстративного тестового оцінювання застосовуються евристики Е1.1, Е2.1 і Е3. Таким чином, евристики Е1, Е2, Е3 трансформуються у такі варіанти відповідей та обчислюються такі оцінки за них [11].

– Е1.1 – деякому обґрунтованому коефіцієнту $k,$ який відображує суб'єктивне уявлення організаторів тестування про «ціну помилки», наприклад, $k = 2$;

часткове пропорційне приписування балів:

– Е2.1 – відповідно до співвідношення одержаних вірних відповідей $\nu_1 \le \nu,$ до загальної кількості вірних відповідей n_1;

Евристика Е3. Для ситуацій, коли респондентом не вибрано жодної відповіді $(\nu = 0)$ або усі відповіді позначені, як вірні, тобто $\nu = n,$ то штрафом є нульова оцінка результату – за відсутність вибірковості: $B = 0$ [11].

Відповідь1: (1,*,*,*,*)v(*,1,*,*,*)v(*,*,1,*,*) = 1/3;

Відповідь2: (1,1,*,*,*)v(*,1,1,*,*)v(1,*,1,*,*) =2/3;

Відповідь3: (1,1,1,*,*) =1;

Відповідь4:
(1,*,*,0,*)v(*,1,*,0,*)v(*,*,1,0,*)v(1,*,*,*,0)v(*,1,*,*,0)v(*,*,1,*,0) =1/3/2=1/6;

Відповідь5: (1,*,*,0,0)v(*,1,*,0,0)v(*,*,1,0,0)=1/3/4=1/12;

Відповідь6:
(1,1,*,0,*)v(*,1,1,0,*)v(1,*,1,0,*)v(1,1,*,*,0)v(*,1,1,*,0)v(1,*,1,*,0) =2/3/2=1/3;

Відповідь7: (1,1,*,0,0)v(*,1,1,0,0)v(1,*,1,0,0)=2/3/4=1/6;

Відповідь8: (1,1,1,0,*)v(1,1,1,*,0) =1/2;

Відповідь9: (1,1,1,0,0)=0.

При побудові тестового контенту можна використати евристики, що будуть мати інші закономірності і відповідати іншим конфігураціям відповідей. Залежно від вибору евристик, змінюється чутливість функції, що визначає значення результуючої оцінки [11].

2 РОЗДІЛ. ПРОЕКТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ

2.1. Побудова SADT та IDEF0 – діаграм

Під час виділення системи із середовища, аналізу головної цілі системи, її властивостей і проектування архітектури системи потрібно використовувати принципи системного підходу.

Для аналізу та побудови моделей системи можна використовувати різні методології. Максимально розлогий вибір моделей пропонують CASE технології . CASE (Computer Aided Software Engineering) завбачає різні типи моделей.

Побудова цих моделей завбачає використання принципу декомпозиції. Таким чином, можливо відокремити глобальну функцію системи на підфункції, окреслити, які можуть бути виділені підсистеми, та деталізувати саме ті функції/підфункції, блоки, що треба для поставленої задачі.

SADT - одна із найпопулярніших та масово застосовуваних систем проектування. SADT - Structured Analysis & Design Technique (методика структурного аналізу і проектування) - це графічні відмітки і підхід до опису систем, що моделюють вхідні дані, що отримує система, опис глобальної функції системи, та якими будуть вихідні дані із урахуванням керуючих впливів.

SADT відображає сукупність методів, процедур та правил, що призначені для побудови функціональної моделі об’єкта або системи. Головні елементи даної методології мають відповідати наступним положенням:

1. Небагатослівність та строгість.

2. Лімітування рівнів деталізації, які мають бути для досягнення мети рішення проблеми або завдання.

3. Графічне подання блокового моделювання.

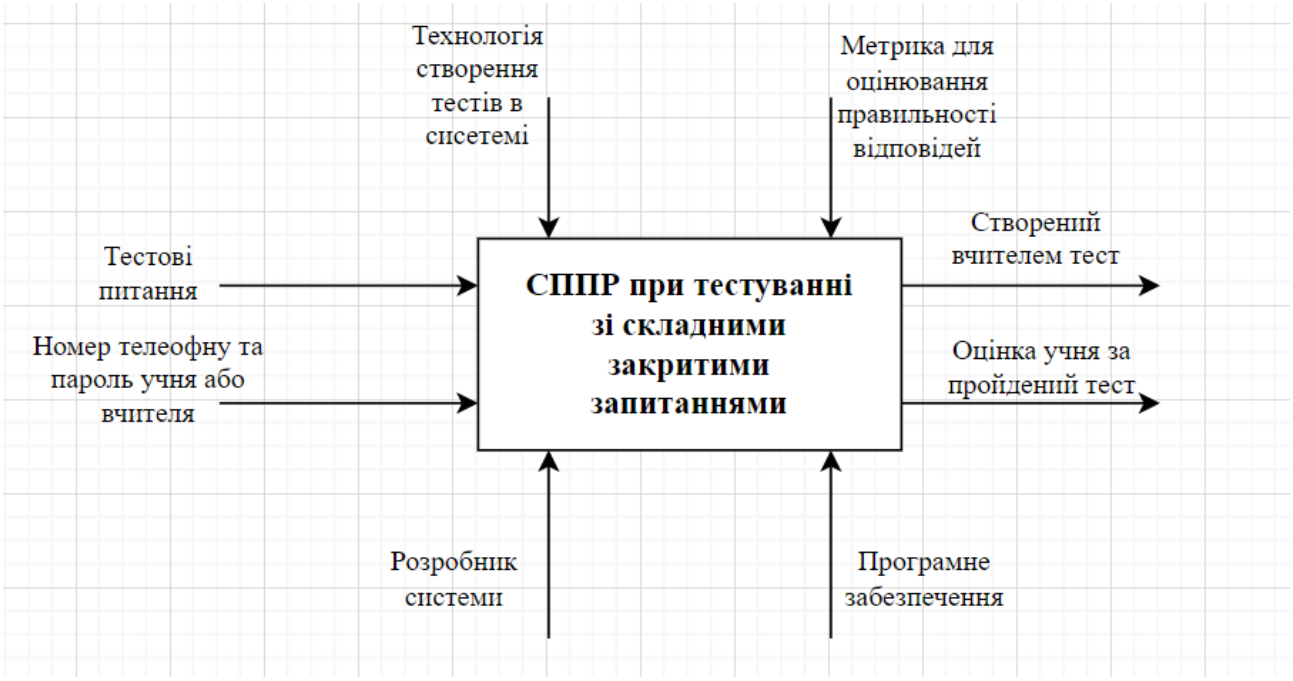

Рис.2.1 – SADT діаграма для СППР при тестуванні зі складними закритими запитаннями

Опис профілів зацікавлених сторін

1. Внутрішні зацікавлені сторони:

- Розробник програми;

- Вчителі;

- Учні.

2. Зовнішні зацікавлені сторони:

- Контролюючі органи;

- Конкурентні продукти.

Більш детальну інформацію про систему можна відобразити побудувавши деталізовану діаграму IDEF0.

IDEF0 – методологія функціонального моделювання і графічна нотація, що використовується для формалізації і характеристики бізнес-процесів. Основною відмінністю IDEF0 є її притиск на підрядність об'єктів. В IDEF0 розбираються логічні зносини між процесами, а не їх тимчасова послідовність.

Стандарт IDEF0 був розроблений у 1981 році у США департаментом Військово-повітряних сил для автоматизації промислових підприємств. У процесі розробки програмного забезпечення розробники наштовхнулися на необхідність розробки нових методів аналізу бізнес-процесів.

Внаслідок цього з'явилася методологія функціонального моделювання IDEF0, у якій для аналізу застосовуються спеціальні нотації IDEF0.

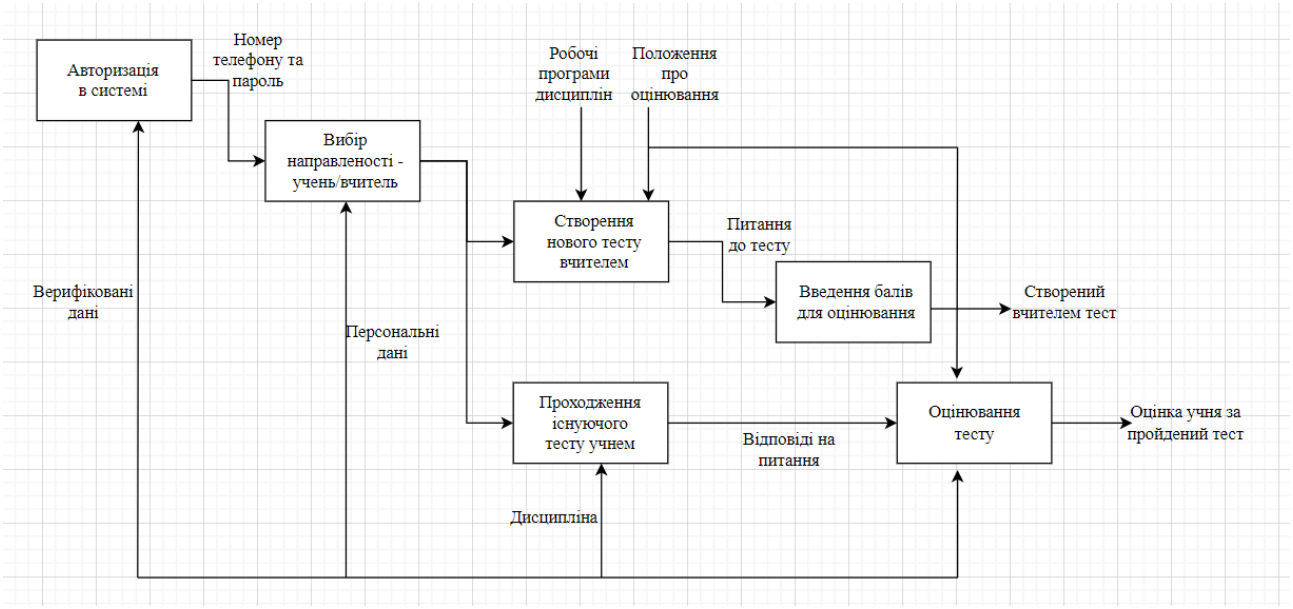

Рис.2.2. Деталізована IDEF0 діаграма для СППР при тестуванні зі складними закритими запитаннями

Згідно цієї діаграми користувач спочатку авторизується в системі за допомогою номеру телефону та паролю. Далі він обирає направленість – учень або вчитель. Вчитель за допомогою даної СППР може створювати тести та

вводити бали для оцінювання кожного запитання, в свою чергу, учень має можливість проходити створені заздалегідь тести та отримувати за них оцінки.

Також було розроблено діаграму прецедентів, що відображає відношення між акторами та прецедентами у системі.

Актори – застосовують систему для досягнення цілі. Актори позначаються на діаграмі прецедентів схематичними людськими фігурками. Актори поділяються на основних та другорядних. Основні актори – ініціюють взаємодію з системою, другорядні – більшою мірою реакційні.

Прецедент – це опис декотрого аспекта системи. Варто зазначити, що прецедент не вказує, як саме досягається певний результат, а тільки що дійсно відбувається. Прецеденти позначаються на діаграмі прецедентів овалами.

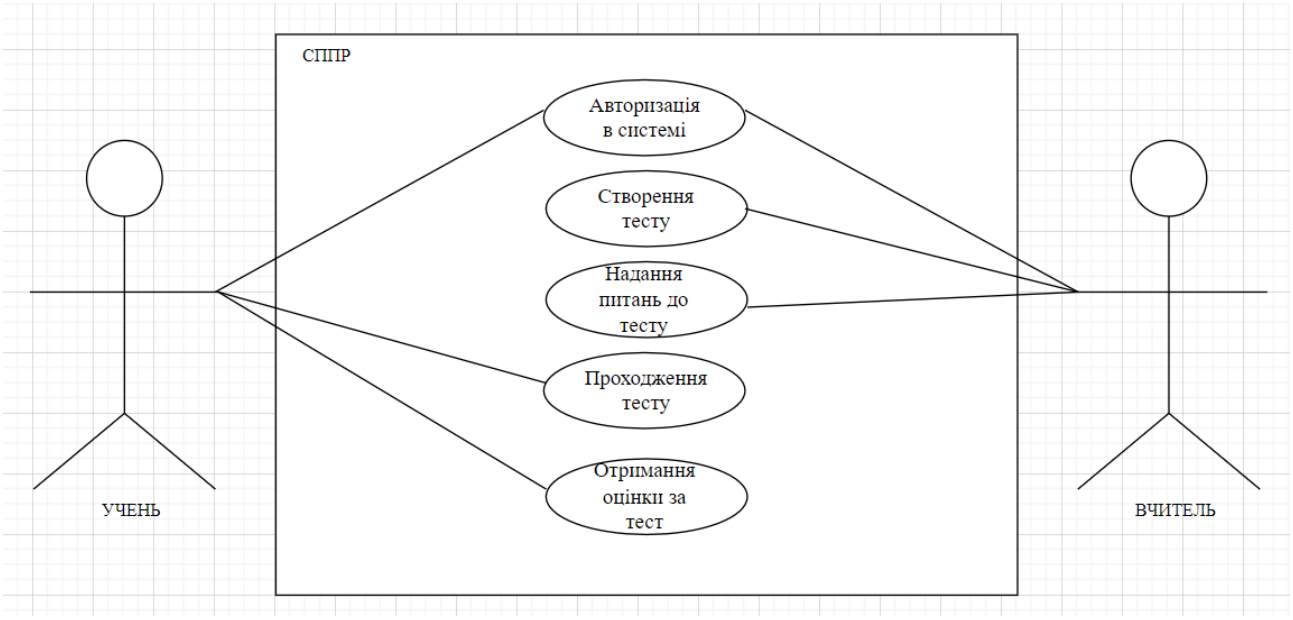

Рис.2.3. Діаграма прецедентів для СППР при тестуванні зі складними закритими запитаннями

Акторами в даній СППР виступають:

- Учень;
- Вчитель.

Згідно даної діаграми учень може виконати авторизацію в системі, проходження тесту та отримання оцінки, вчитель також має можливість авторизуватися в системі, а також може виконувати створення тесту та надання питань до тесту.

2.2. Засоби розробки

Базовим середовищем розробки було середовище Project Jupyter — це сфера розробки, де можна побачити результат запуску коду і його окремих фрагментів. На відміну від традиційного середовища де, що код не можна розбити на шматки та виконувати їх у довільному порядку.

В такому середовищі розробки можливо, скажімо, написати функцію і одразу оцінити її роботу, без запуску програми цілком. Також можливо перемінити порядок виконання коду. Можна порізно завантажити файл на диск, порізно перевірити вміст, порізно обробити вміст.

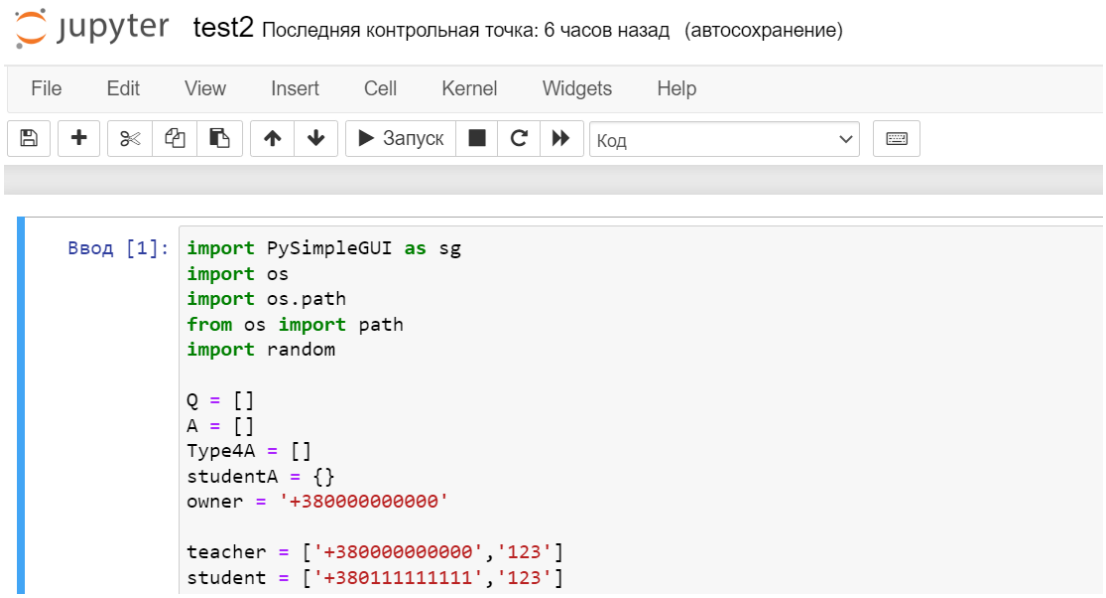

Рис.2.4. приклад програмного коду Project Jupyter

Для розробки безпосередньо програми використовувалась мова програмування Python версією 3.8.2.

Python - це високорівнева мова програмування суспільного призначення, що застосовується у тому числі і для розробки систем. Мова адаптована на покращення продуктивності користувача та читання коду.

Python підтримує декілька парадигм програмування: структурне, об'єктноорієнтоване, функціональне, аспектно-орієнтоване. У мові присутня динамічна типізація, автоматичне управління пам'яттю, загальна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень та підходящі високорівневі структури даних.

Код на Python об’єднуються у функції і класи, що можуть об'єднуватися в модулі, а вони у свою чергу можуть бути об'єднаніу пакети.

Мова програмування Python використовується в таких сферах як:

- розробка програмного забезпечення;
- математика;
- системний сценарій.

За допомогою мови програмування Python можна виконувати наступне:

- зчитування файлів;
- зміна та редагування файлів;
- впровадження веб-додатків на сервері;
- реалізація робочих процесів;
- підключення до баз даних;

Python реально використовувати на різноманітних операційних системах (Windows, Mac, Linux, Raspberry Pi тощо). Python має доволі зрозумілий синтаксис, схожий до англійської мови. У Python є синтаксис, який дає змогу розробникам писати програми із меншою кількістю рядків, ніж багато інших мов програмування. Код може бути одразу запущений, оскільки Python працює в системі інтерпретатора. Це сідчить, що прототипування може бути доволі швидким. Найсвіжішою основною версією мови програмування Python є Python 3, що використовується при написанні програми. Проте, Python 2, хоча не змінюється нічим іншим, крім покращення безпеки, все ще є досить популярним.

Синтаксис Python подібний на інші мови програмування, проте все одно має низку переваг в порівнянні з іншими мовами програмування:

- Python був створений для якісного розуміння і має багато спільностей з англійською мовою під впливом математики;
- Python застосовує нові рядки для виконання команди, на відміну від інших мов програмування, що постійно застосовують крапку із комою або круглі дужки;
- для формулювання області застосування Python полягається на відступ, використовуючи пробіли, наприклад, область циклів, класи і функції. Решта мов програмування постійно використовують для цієї цілі фігурні дужки.

Один із найголовніших якісних плюсів Python є те, що стандартна бібліотека та інтерпретатор існують у безкоштовному доступі, в двійковій та вихідній формах. Проблем з доступом також не існує, тому, що Python і усі потрібні інструменти відкриті на усіх головних платформах. Завдяки цьому мова програмування Python є дуже популярною серед розробників, які не мають можливості витрачати кошти на розробку.

Python доволі нескладний, цю мову програмування легко зрозуміти, бо вона зобов’язує використання унікального синтаксису, який легко читати. Розробник може розбирати та аналізувати код Python значно легше, ніж інші мови програмування. Це, насамперед, понижує витрати на підтримку та створення програми, тому, що це дозволяє розробникам працювати разом без суттєвих перепон в мові або досвіді.

Також, мова програмуваня Python підтримує застосування пакетів та модулів, а це свідчить про те, що програма може бути розроблена в модульному стилі, код можливо застосовувати ще безліч разів у багатьох різних розробках. Створивши необхідний розробнику пакет чи модуль, його можливо реалізувати альтернативних проектах та просто імпортувати або експортувати цей модуль.

Проте, як і будь яка мова програмування, Python має певні недоліки, а саме:

Продуктивність. Більшість розробників, та й сам автор мови, сходяться на думці, що Python не настільки спритний, наскільки хотілося б. Це пов'язано з тим, що Python інтерпретується. Але навіть у порівнянні з іншими мовами, що інтерпретуються, помітно, що Python програє у продуктивності. Але це легко можна змінити за допомогою реалізацій тієї чи іншої проблемної ділянки коду. В умовах сьогоднішніх потужностей це несильно помітно.

Синтаксис - так, синтаксис це і мінус теж, тому що якщо розробник переходить із іншої мови програмування, синтаксис буде незвичним та трошки дивним для нього, проте це лише діло звички.

Динамічна типізація – через динамічну типізацію Python бере більше ресурсів, ніж хотілося б, проте це часто відшкодовується внутрішнім кешуванням.

Global Interpriter Lock. На сьогоднійшній день саме це є головною проблемою продуктивності Python, а також саме цим зумовлена ненайкраща реалізація багатопоточності. Код GIL не змінювався з першої версії мови. Це

говорить про те, що він застарів. Лишається вірити, що розробники приділять цьому увагу найближчими релізами.

Для розробки системи підтримки прийняття рішень при було використано бібліотеки мови програмування Python, такі як:

- Pysimplegui
- Os

PySimpleGUI — це бібліотека Python, що дає змогу користувачу Python всіх рівнів реалізовувати графічні інтерфейси. Розробник створює своє вікно графічного інтерфейсу, використовуючи «макет», який містить віджети (вони називаються «елементами» в PySimpleGUI) [1].

Макет використовується для створення вікна із допомогою однієї із 4 підтримуваних рамок для відображення і взаємодії із вікном користувача. Підтримувані фреймворки включають tkinter, Qt, WxPython або Remi [1].

Код з PySimpleGUI простіший і менший, ніж написання безпосередньо за допомогою базової платформи, тому що PySimpleGUI реалізує більшу частину "шаблонного коду" для користувача. Крім того, інтерфейси спрощені, щоб отримати якомога менше коду для отримання бажаного результату. Залежно від використовуваної програми та фреймворку, програма PySimpleGUI може вимагати від 1/2 до 1/10 обсягу коду для створення ідентичного вікна за допомогою безпосередньо одного з фреймворків [2].

Хоча мета полягає в тому, щоб інкапсулювати/приховати конкретні об’єкти і код, які використовуються в структурі графічного інтерфейсу, на якій ви працюєте, у разі потреби користувач може отримати прямий доступ до віджетів і вікон, що залежать від фреймворків. Якщо налаштування або функція ще не відкриті або доступні за допомогою API PySimpleGUI, користувач має доступ до фреймворку. Також він має змогу розширити можливості, не змінюючи безпосередньо сам пакет PySimpleGUI [3].

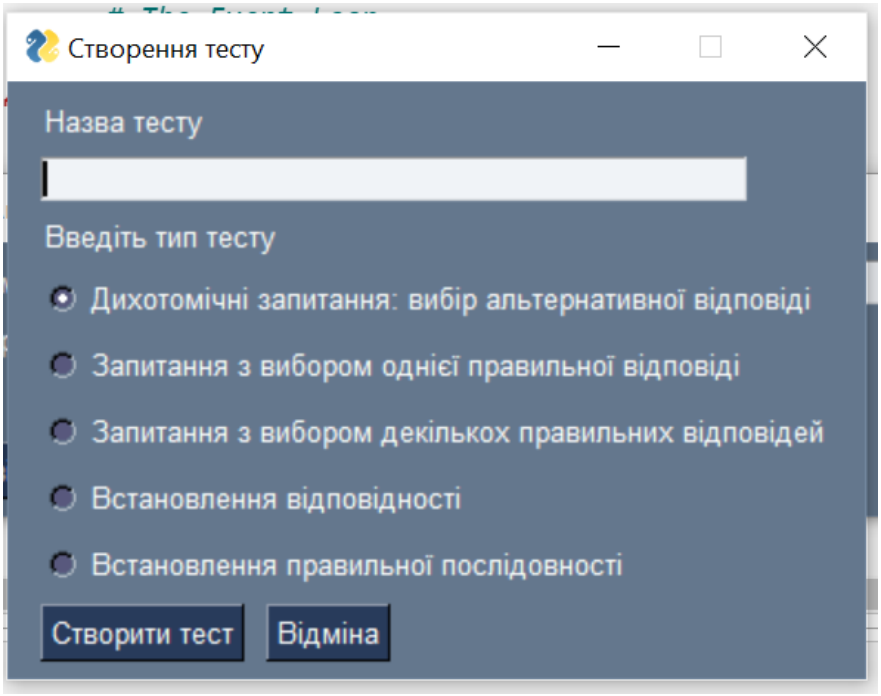

Рис.2.5. приклад інтерфейсу створеного за допомогою бібліотеки PySimpleGUI

Os – це бібліотека функцій для роботи з операційною системою. Методи, включені до неї дають змогу зазначити тип операційної системи, отримувати доступ до змінних оточення, керувати директоріями та файлами [4].

Методи з бібліотеки os можуть використовуватись розробником для різних цілей. Нижче показані найбільш популярні з них, що дозволяють отримувати дані про операційну систему. Також отримувати відомості про файли та папки, які зберігаються в пам'яті на жорсткому диску ПК [5].

- Отримання інформації про ОС

Щоб дізнатися про ім'я поточної ОС, достатньо скористатися методом name. Залежно від встановленої платформи, він поверне її коротке найменування у рядковому поданні.

- Зміна робочої директорії

За замовчуванням робочою директорією програми є каталог, де міститься документ із її вихідним кодом. Завдяки цьому можна не вказувати абсолютний

шлях до файлу, якщо той знаходиться саме в цій папці. За бажанням, робочу директорію можна налаштувати на свій розсуд, застосувавши метод chdir із бібліотеки os. Для цього необхідно передати йому як параметр абсолютну адресу до нового каталогу. Якщо вказаного шляху насправді не існує, програма буде завершена в аварійному режимі через викинутий виняток.

- Перевірка існування шляху

Щоб уникнути помилок, пов'язаних із відсутністю певного файлу або директорії, які мають бути оброблені програмою, слід попередньо перевіряти їх наявність за допомогою методу exists. Передавши йому як аргумент шлях до потрібного файлу або папці, можна розраховувати на лаконічний відповідь у вигляді булевого значення true/false, що повідомляє про наявність/відсутність зазначеного об'єкта в пам'яті комп'ютера.

- Створення директорій

Можливості модуля os дозволяють не тільки відображати інформацію про об'єкти, що вже існують у пам'яті, але і генерувати абсолютно нові. Наприклад, за допомогою методу mkdir досить легко створити папку, просто вказавши їй бажаний шлях.

- Видалення файлів та директорій

Позбутися непотрібного в подальшій роботі файлу можна за допомогою методу remove, віддавши йому як аргумент абсолютний або відносний шлях до об'єкта.

- Запуск на виконання

Вбудовані функції бібліотеки os дозволяють запускати окремі файли та папки прямо з програми. З цим завданням чудово справляється метод startfile, якому варто передати адресу необхідного об'єкта.

- Обчислення розміру

Щоб визначити розмір документа або папки, варто скористатися функцією getsize, як це показано в прикладі для файлу test.txt. Функція print виводить розмір документа в байтах. Скористатися getsize можна й у виміру обсягу директорій [6].

- Відображення списку записів у каталозі

За допомогою методу listdir можливо реалізувати повернення списоку, що містить імена записів у каталозі, заданому шляхом. Список відобразиться у довільному порядку.

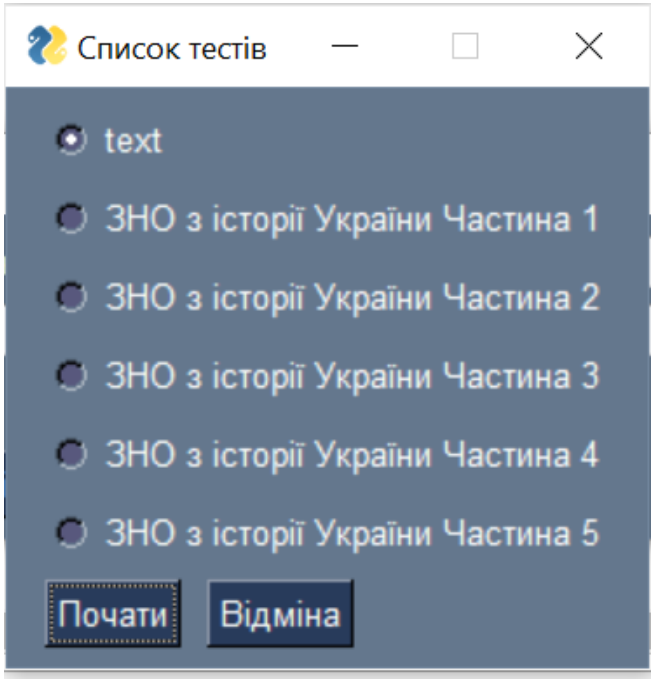

Рис.2.6. приклад інтерфейсу створеного за допомогою бібліотеки OS

3 РОЗДІЛ. ОПИС ТА ТЕСТУВАННЯ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ПРИ ТЕСТУВАННІ ЗІ СКЛАДНИМИ ЗАКРИИМИ ЗАПИТАННЯМИ

3.1. Програмна реалізація системи

Для реалізації системи підтримки рішень при тестуванні зі складними закритими запитаннями було створено наступні функції мови програмування Python:

1) Функція для створення тесту

2) Функція для створення питання

3) Функція для додання відповідей до питань типу 1, 2, 3

4) Функція для додання відповідей до питань типу 4

5) Функція для додання відповідей до питань типу 5. Також унеможливлено використання недопустимих символів

6) Функція для відображення питання тесту в залежності від типу тесту

7) Функція для читання даних тесту з файлу

8) Функція для відображення списку тестів

Також для оцінювання правильності відповідей була використана Метрика Кука.

3.2. Вибір тестових задач, тестування роботи системи

Для тестування системи підтримки прийняття рішень було обрано використати тести ЗНО з історії України. Дані тести є найбільш підходящими, оскільки мають усі п’ять видів необхідних тестових запитань закритої форми, а саме:

- Вибір альтернативної відповіді;
- Вибір однієї правильної відповіді;

- Вибір декількох правильних відповідей;
- Встановлення відповідності;
- Встановлення правильної послідовності.

На першому кроці вчитель авторизується в системі.

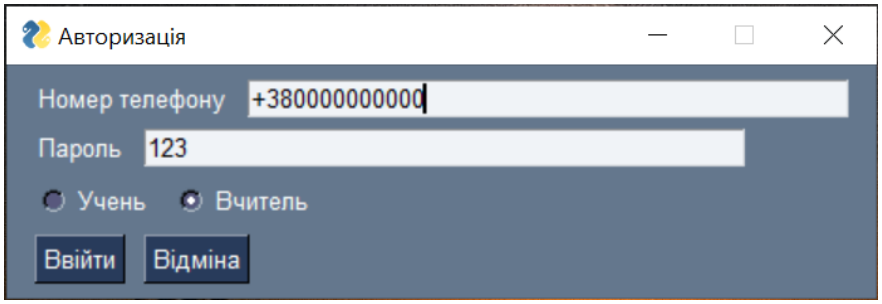

Рис.3.1 – Авторизація в системі вчителя

Далі вчитель задає назву тесту та вибирає тип запитань для тестування. В нашому випадку – це вибір альтернативної відповіді.

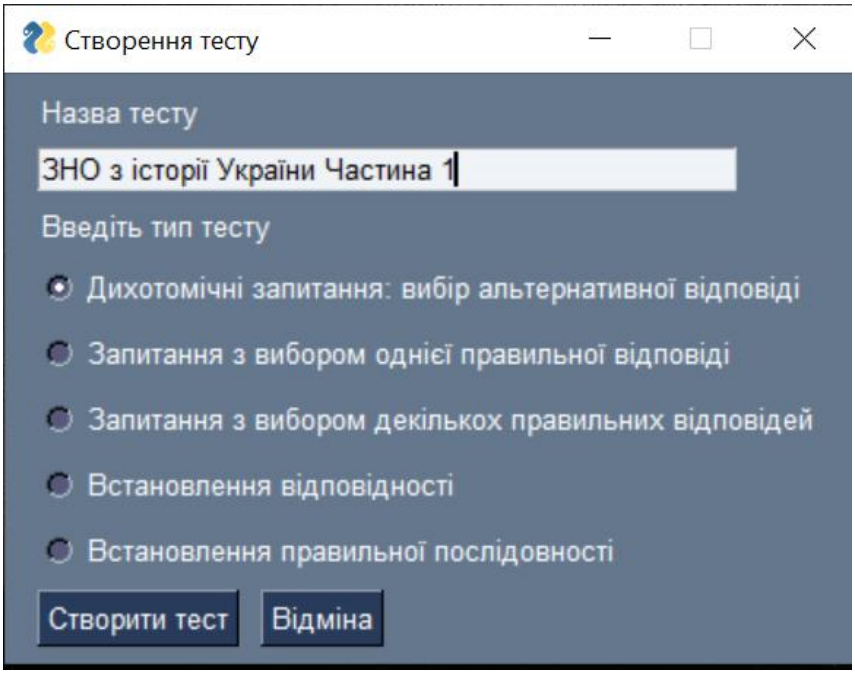

Рис.3.2 – Створення тесту (вибір альтернативної відповіді)

Наступним кроком вчителю необхідно ввести запитання, варіанти відповідей та кількість балів за правильну відповідь.

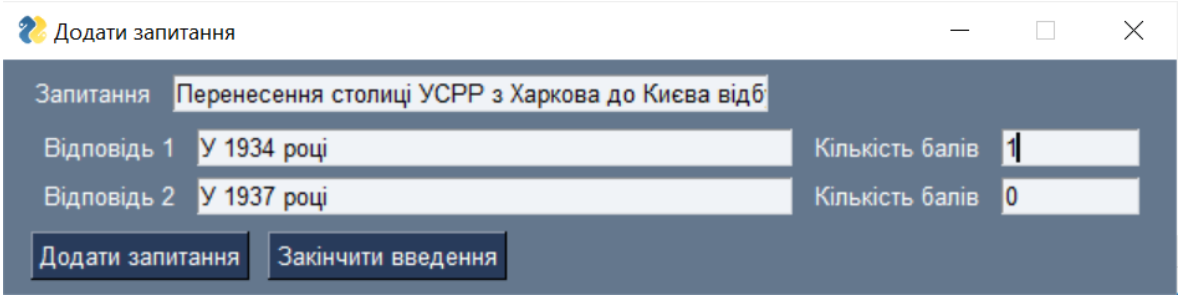

Рис.3.3. Додання запитання (вибір альтернативної відповіді)

Інші запитання можна також ввести в текстовому файлі

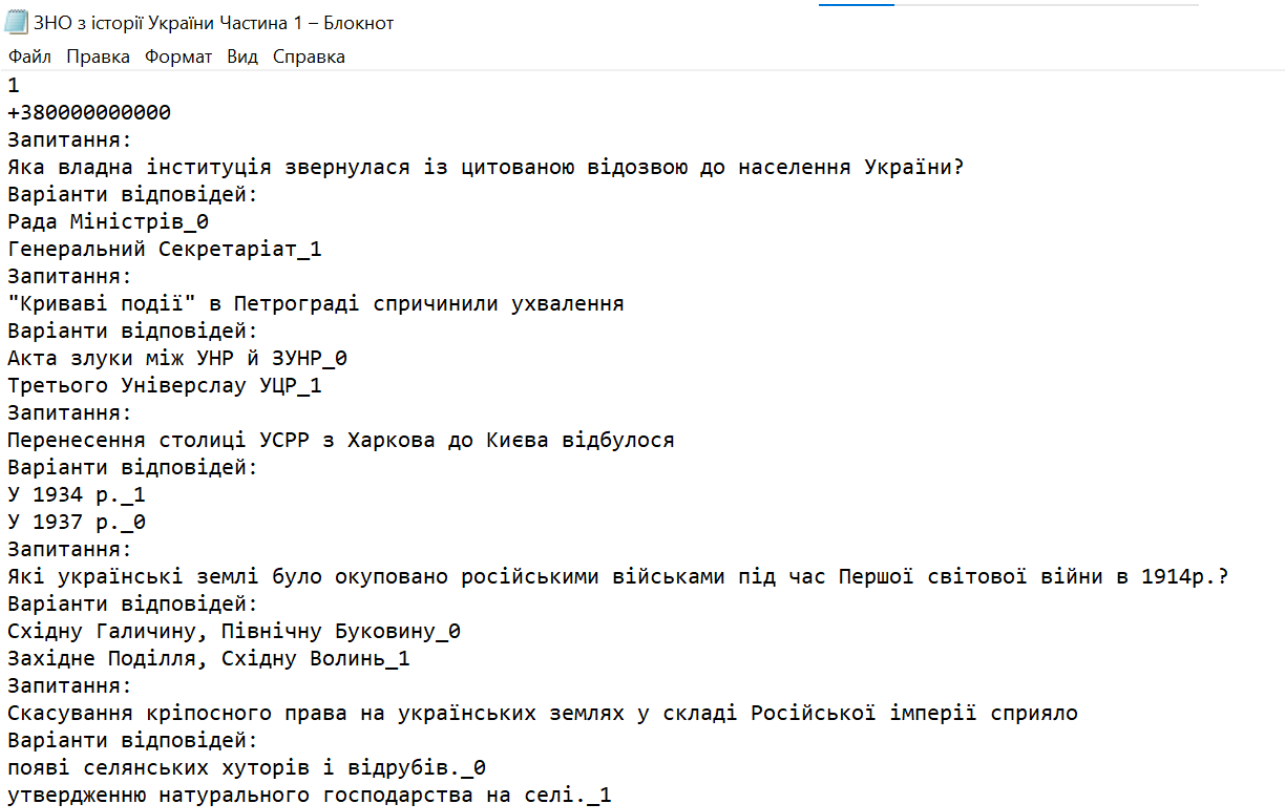

Рис.3.4. Додання запитання через файл (вибір альтернативної відповіді)

За таким самим алгоритмом створюємо всі інші частини тесту:

В другій частині тесту використовується тип запитань – запитання з вибором однієї правильної відповіді, також для цієї частини є можливість додавати необмежену кількість варіантів відповідей.

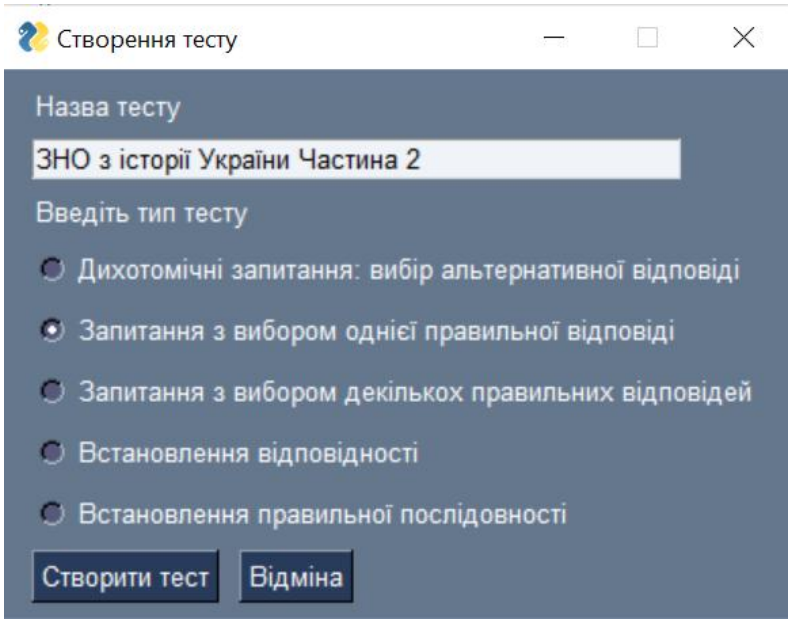

Рис.3.5. Створення тесту (вибір однієї правильної відповіді)

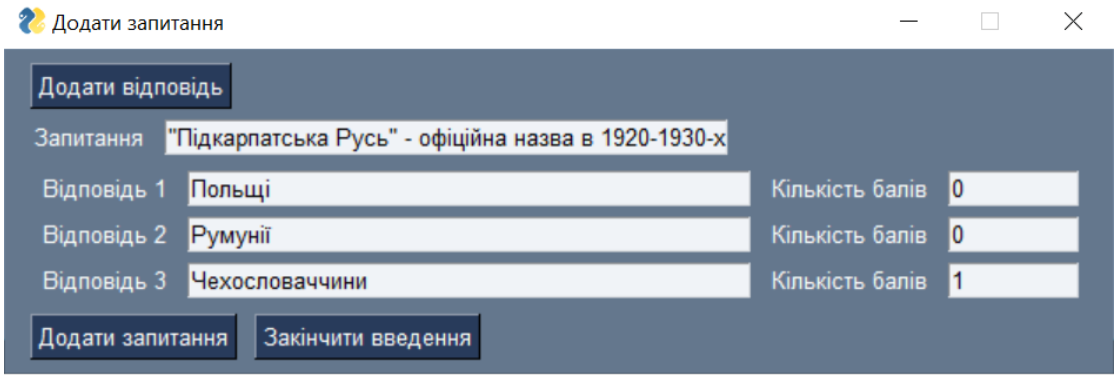

Рис.3.6. Додання запитання (вибір однієї правильної відповіді)

В третій частині тесту використовується тип запитань – запитання з вибором декількох правильних відповідей, також для цієї частини є можливість додавати необмежену кількість правильних відповідей.

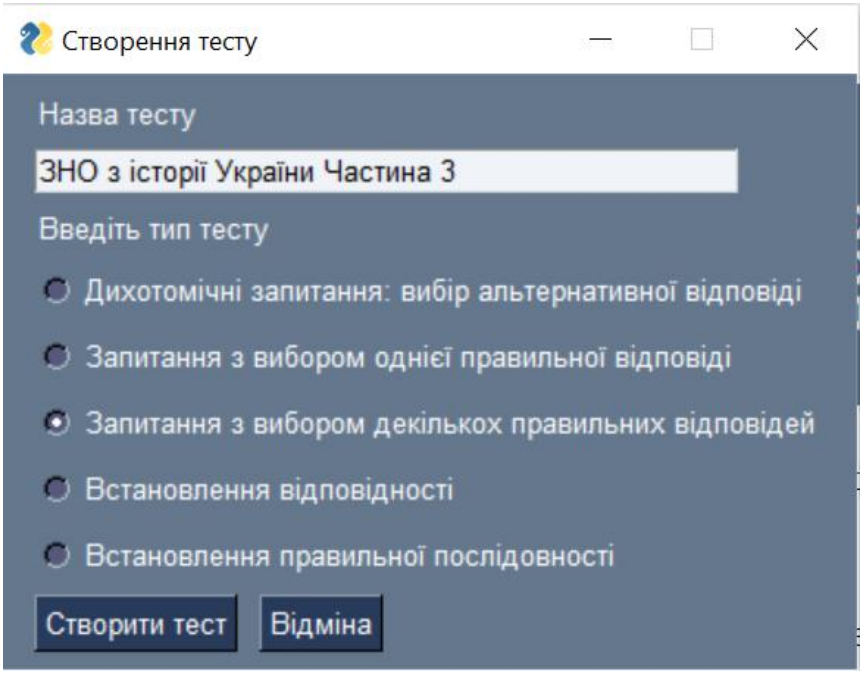

Рис.3.7. Створення тесту (вибір декількох правильних відповідей)

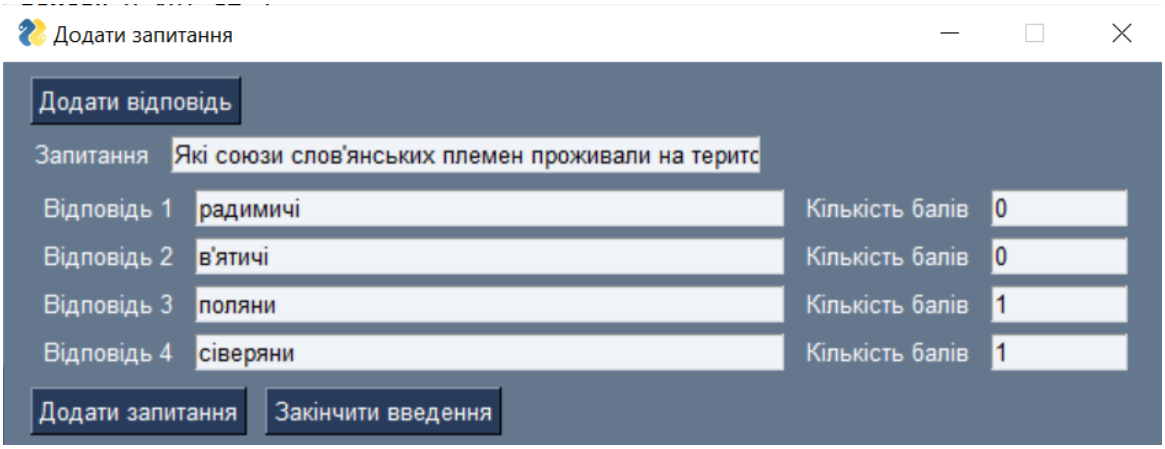

Рис.3.8. Додання запитання (вибір декількох правильних відповідей)

В четвертій частині тесту використовується тип запитань – встановлення відповідності.

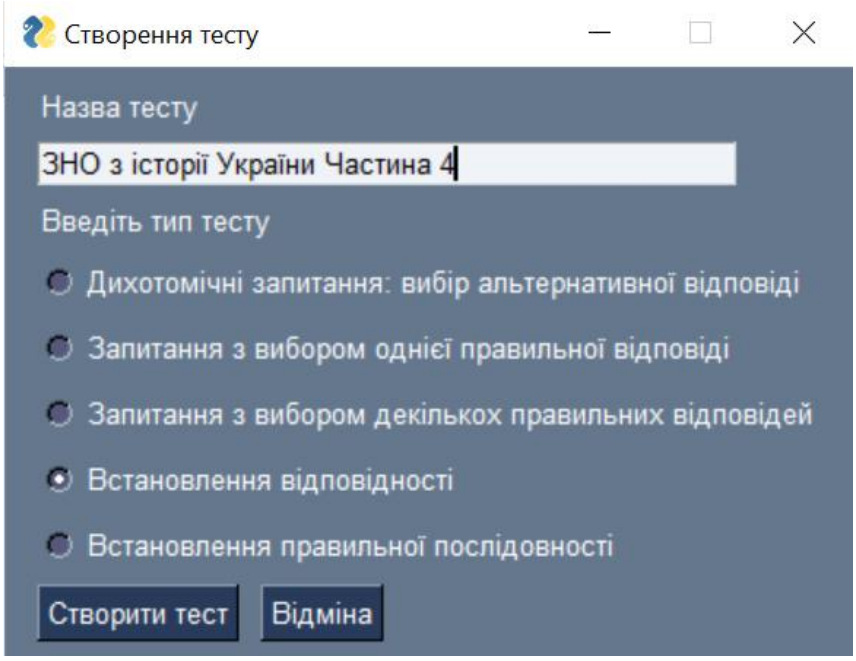

Рис.3.9. Створення тесту (встановлення відповідності)

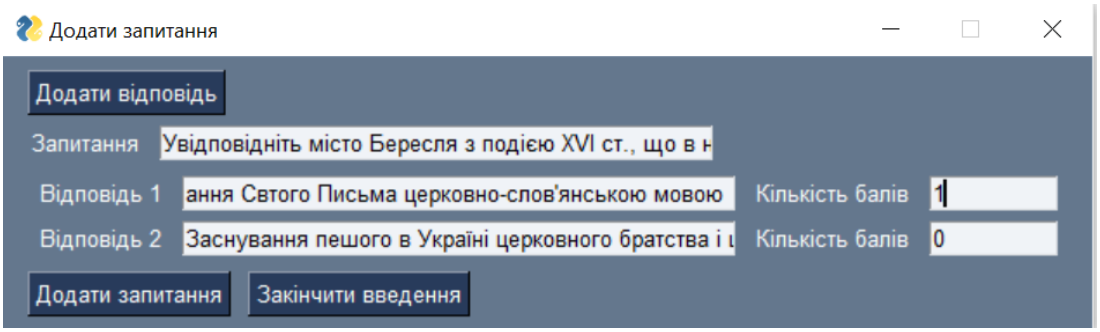

Рис.3.10. Додання запитання (встановлення відповідності)

В п’ятій частині тесту використовується тип запитань – встановлення правильної послідовності. В цій частині вчителю необхідно записати послідовність в правильному порядку та оцінити правильність виконання балами. Також, для цієї частини тесту унеможливлено введення інших символів окрім цифр.

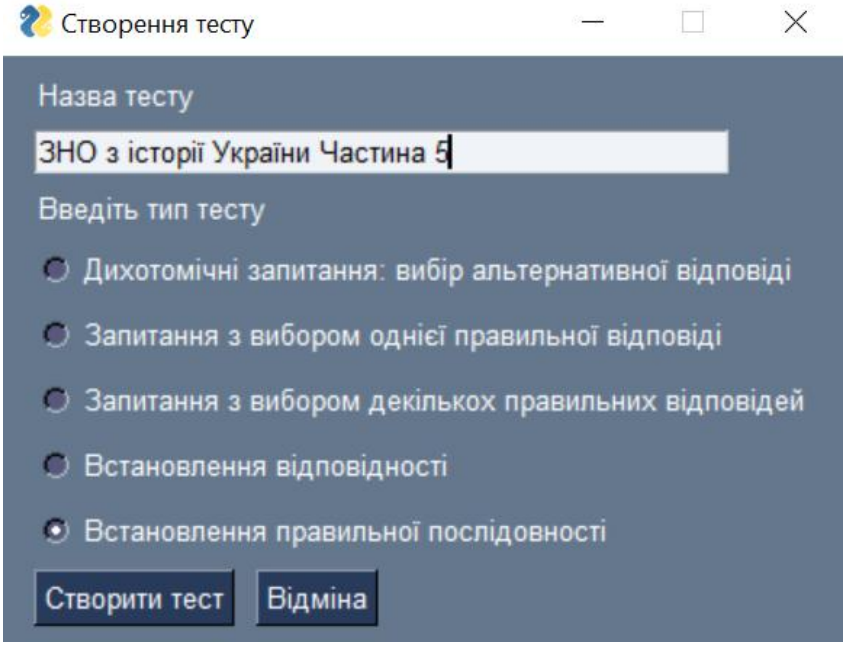

Рис.3.11. Створення тесту (встановлення правильної послідовності)

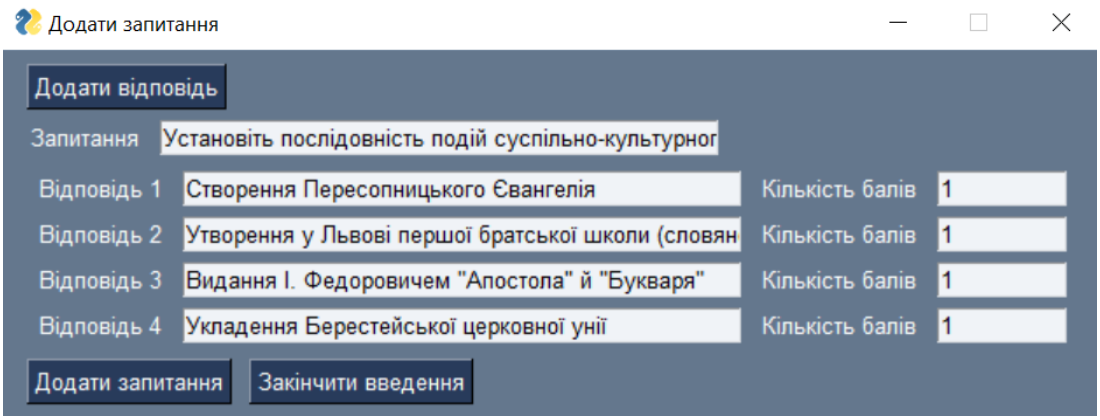

Рис.3.12. Додання запитання (встановлення правильної послідовності)

Тепер, коли ми маємо всі п’ять типів тестів в системі, учень має можливість авторизуватися в системі, пройти кожен із тестів та отримати результат – відсоток правильних відповідей, оцінку за 200-бальною і 5-бальною шкалою.

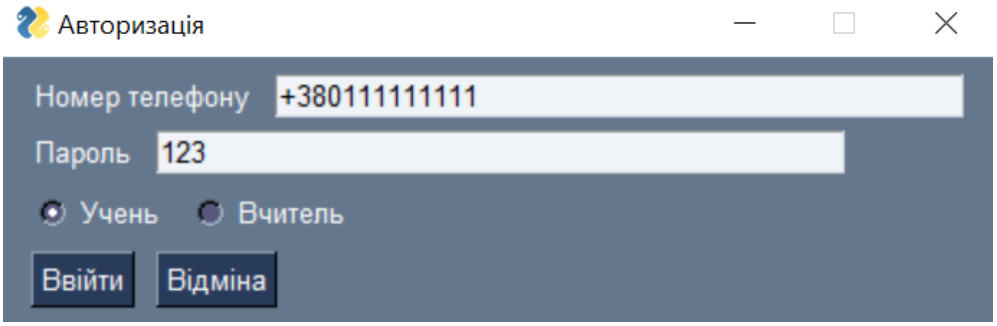

Рис.3.13. Авторизація в системі учня

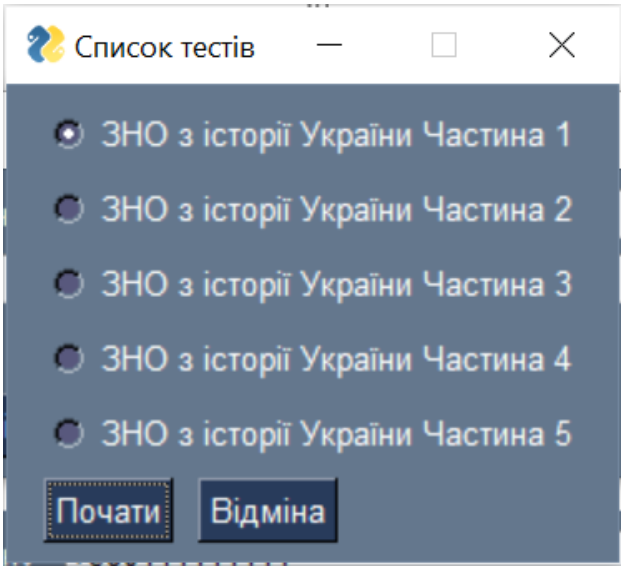

Рис.3.14. Вибір тесту для проходження

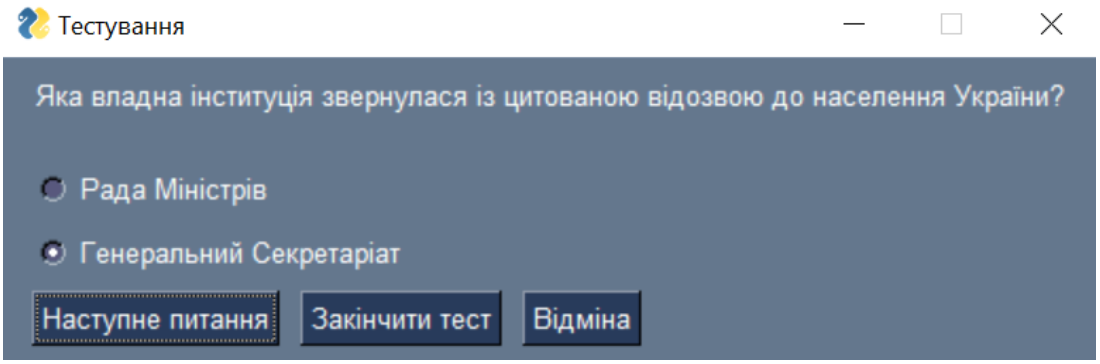

Рис.3.15. Проходження тесту (вибір альтернативної відповіді)

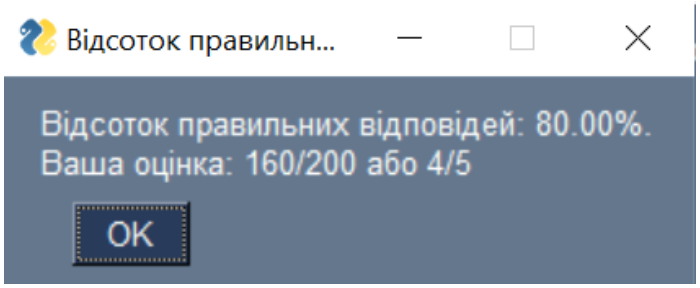

Рис.3.16. Оцінка за тест (вибір альтернативної відповіді)

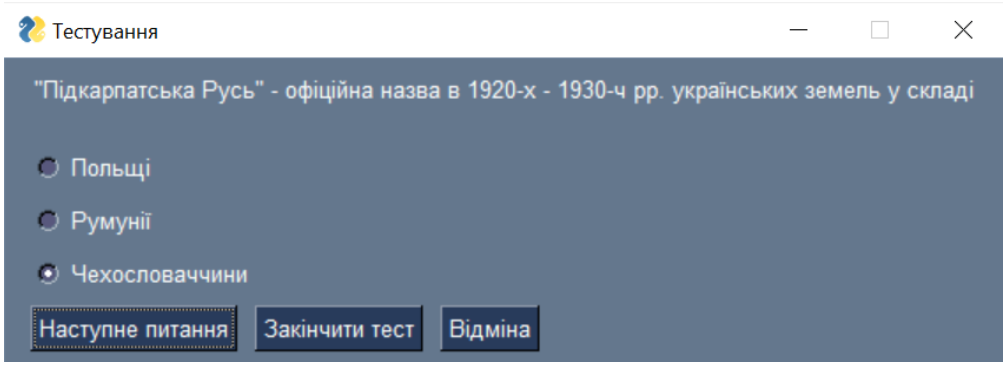

Рис.3.17. Проходження тесту (вибір однієї правильної відповіді)

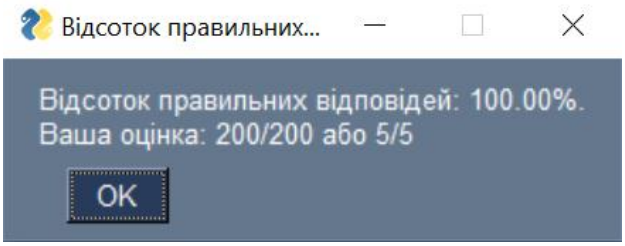

Рис.3.18. Оцінка за тест (вибір однієї правильної відповіді)

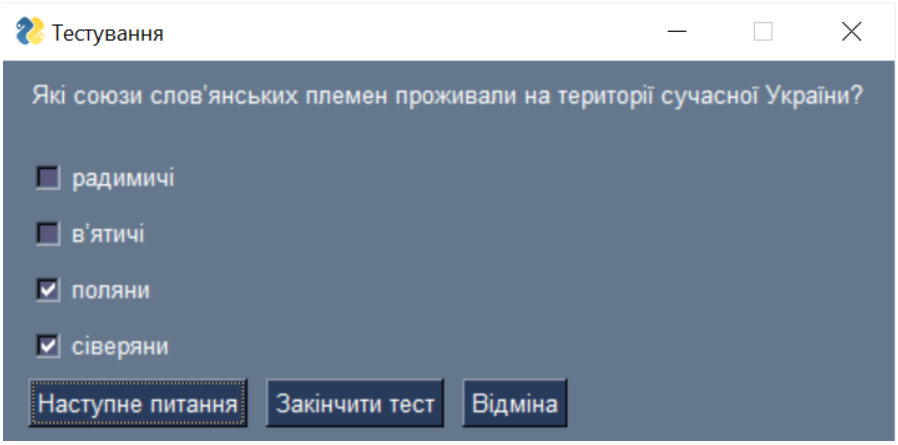

Рис.3.19. Проходження тесту (вибір декількох правильних відповідей)

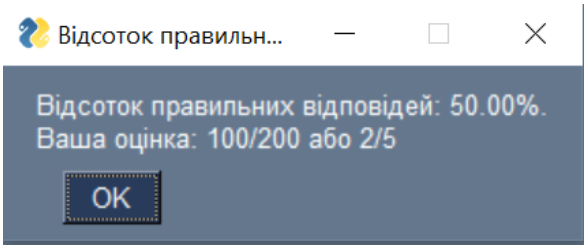

Рис.3.20. Оцінка за тест (вибір декількох правильних відповідей)

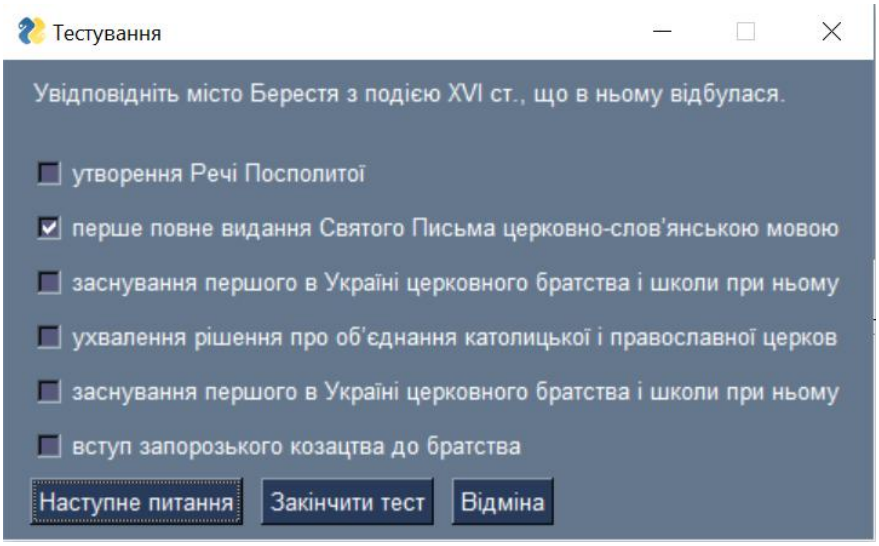

Рис.3.21. Проходження тесту (встановлення відповідності)

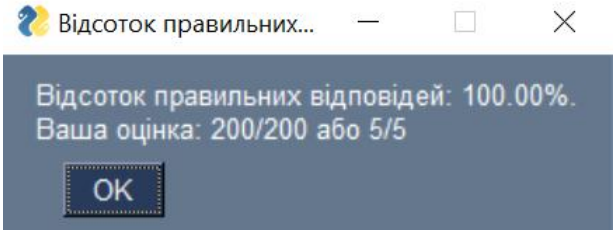

Рис.3.22. Оцінка за тест (встановлення відповідності)

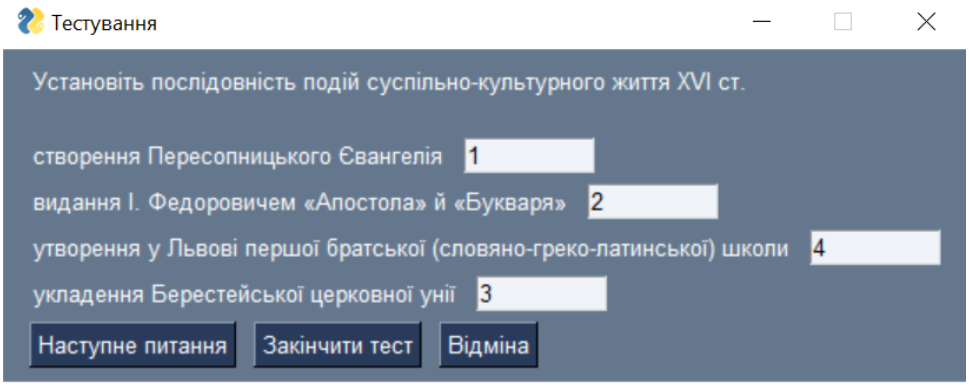

Рис.3.23. Проходження тесту (встановлення відповідності)

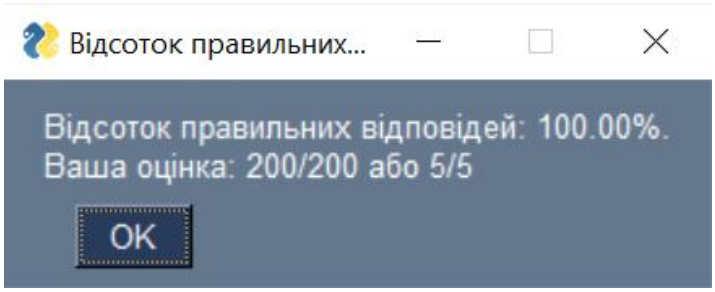

Рис.3.24. Оцінка за тест (встановлення правильної послідовності)

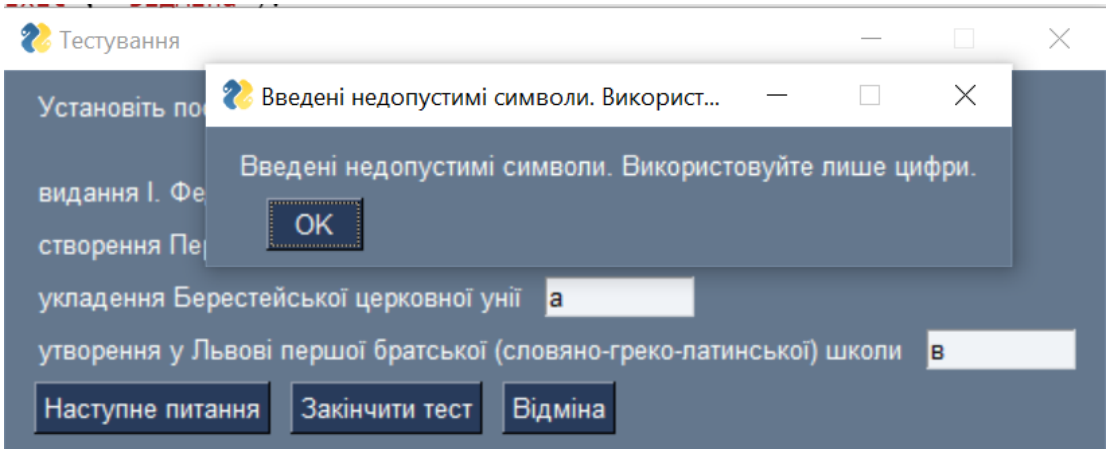

Рис.3.25. унеможливлення введення інших символів окрім цифр

Усі створені тести зберігаються у текстових файлах.

ЗНО з історії України Частина 1 – Блокнот

Файл Изменить Просмотр

```
1
+380000000000
Запитання:
Яка владна інституція звернулася із цитованою відозвою до населення України?
Варіанти відповідей:
Рада Міністрів_0
Генеральний Секретаріат_1
Запитання:
"Криваві події" в Петрограді спричинили ухвалення
Варіанти відповідей:
Акта злуки між УНР й ЗУНР_0
Третього Універслау УЦР_1
Запитання:
Перенесення столиці УСРР з Харкова до Києва відбулося
Варіанти відповідей:
У 1934 р._1
У 1937 р._0
Запитання:
Які українські землі було окуповано російськими військами під час Першої світової війни в 1914р.?
Варіанти відповідей:
Східну Галичину, Північну Буковину_0
Західне Поділля, Східну Волинь_1
Запитання:
Скасування кріпосного права на українських землях у складі Російської імперії сприяло
Варіанти відповідей:
появі селянських хуторів і відрубів._0
утвердженню натурального господарства на селі._1
```

Рис.3.36. Файл з тестом для першого типу запитань

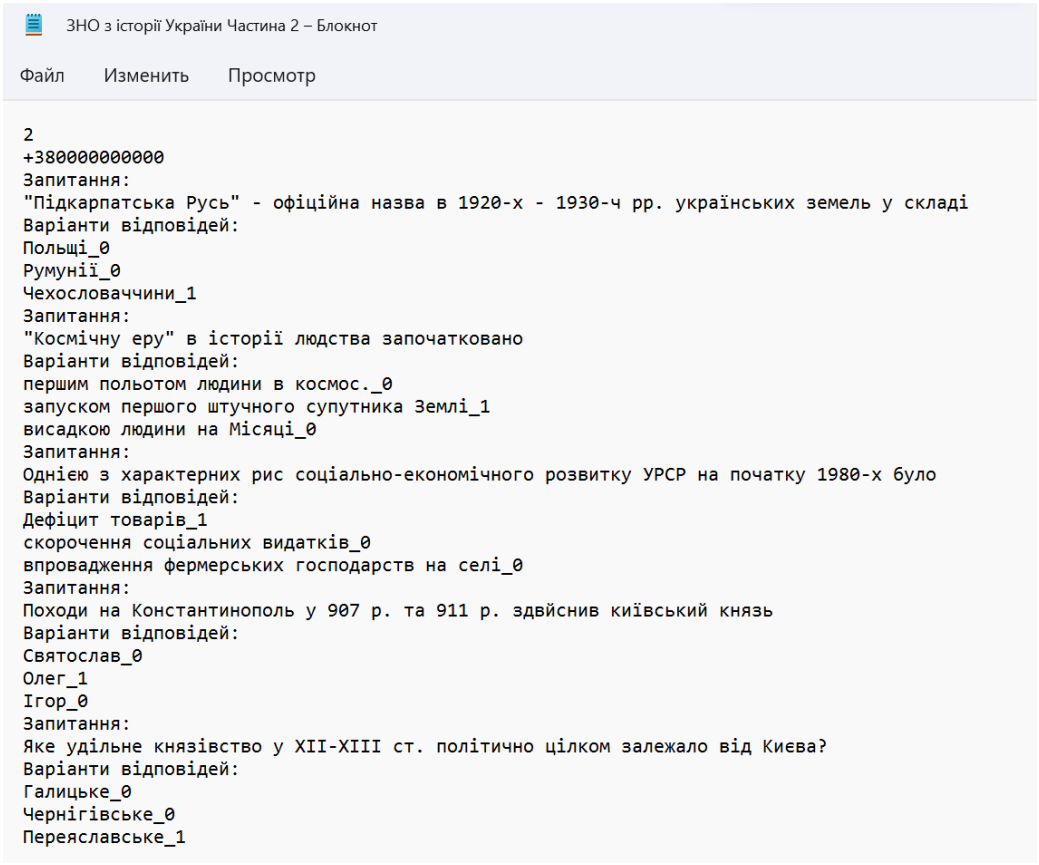

Рис.3.37. Файл з тестом для другого типу запитань

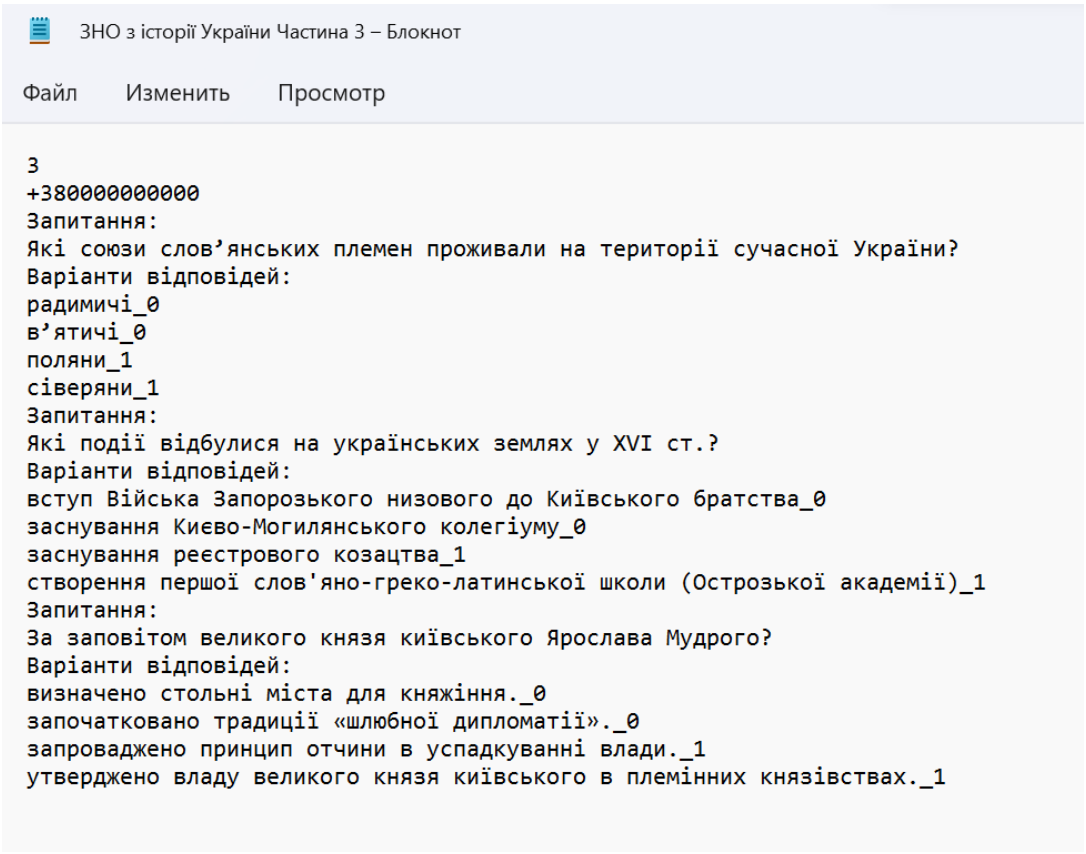

Рис.3.38. Файл з тестом для третього типу запитань

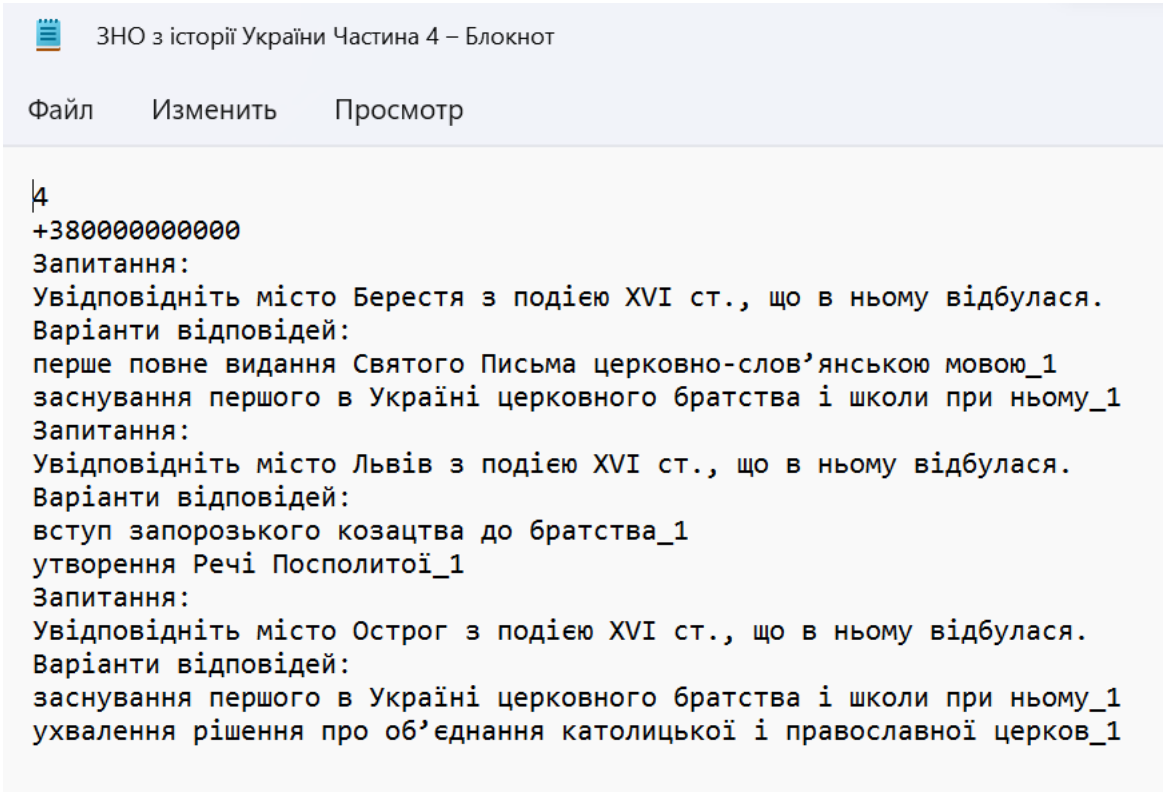

Рис.3.39. Файл з тестом для четвертого типу запитань

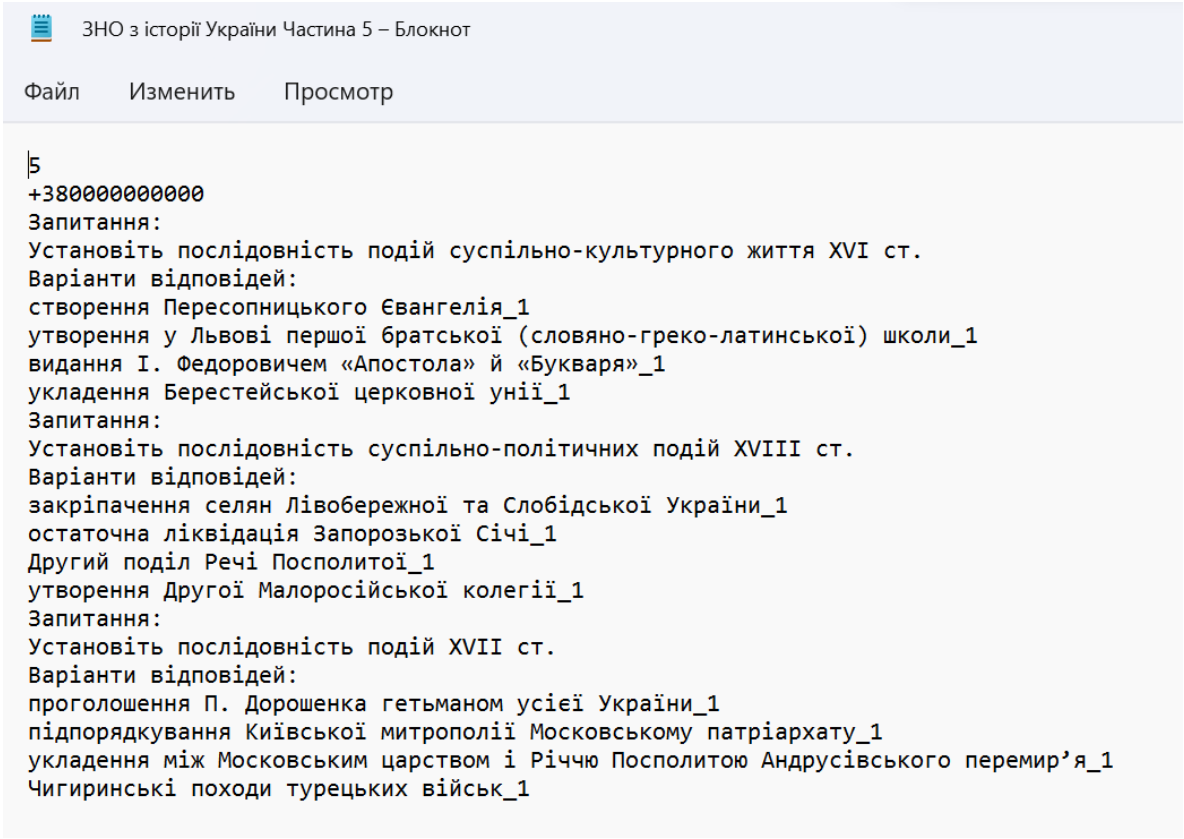

Рис.3.40. Файл з тестом для першого типу запитань

ВИСНОВКИ

У результаті виконання дипломної роботи було розроблено систему підтримки прийняття рішень при тестуванні зі складними закритими запитаннями. Розроблена система дозволяє вчителю створювати тести з різними типами запитань закритої форми, а саме: запитання з вибором альтернативної відповіді, з вибором однієї правильної відповіді, з вибором декількох правильних відповідей, запитання на встановлення відповідності та встановлення правильної послідовності. Розроблена система має режим ручного створення тесту та зчитування його з текстового документу.

Також розроблена система підтримки прийняття рішень дає змогу учню проходити різноманітні тести та отримувати за них оцінку в різних шкалах оцінювання.

Під час роботи над дипломною роботу було оглянуто системи підтримки задач тестування, досліджено існуючі на даний момент інструменти для тестування. Також було спроектовано архітектуру системи та програмно реалізовано її на мові програмування Python.

Результати роботи системи було продемонстровано на прикладі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Інформація про бібліотеки Python:
1. https://all-python.ru/osnovy/os.html
2. https://pythonru.com/osnovy/rabota-s-fajlami-v-python-s-pomoshhju-modulja-os
3. https://www.tutorialspoint.com/python/os_listdir.htm
4. https://russianblogs.com/article/13121807542/
5. https://pysimplegui.readthedocs.io/
6. https://pypi.org/project/PySimpleGUI/
Інформація про системи тестування:
7. https://ciot.pnu.edu.ua/wp-content/uploads/sites/144/2021/05/4-%D0%BD%D0%B0-%D0%B4%D1%80%D1%83%D0%BA-%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D1%96-%D1%96%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B8-%D0%BF%D0%BE%D1%81%D1%96%D0%B1%D0%BD%D0%B8%D0%BA-_edited-ISBN_%D0%905.pdf
8. https://osvita.ua/vnz/high_school/73080/
9. https://web-promo.ua/ua/blog/kratkij-gajd-vse-vozmozhnosti-google-forms/
10. https://studfile.net/preview/5725402/page:31/
11. Гнатієнко Г.М., Снитюк В.Є., Тмєнова Н.П., Волошин О.Ф. Удосконалення задач тестування з використанням експертних технологій прийняття рішень / Інформаційні технології і безпека. Матеріали XX Міжнародної науково-практичної конференції ІТБ-2020. – Київ: Інжиніринг. – 174 с. – С. 1522.
http://ceur-ws.org/Vol-2859/paper5.pdf
12.http://ekmair.ukma.edu.ua/bitstream/handle/123456789/1605/Glybovets_Ostapenko_Tests%20analysis.pdf?sequence=1&isAllowed=y
13. http://mdcs.knuba.edu.ua/article/view/219873
14. Буйницька О.П. Інформаційні технології та технічні засоби навчання. Навч. посіб. – К.: Центр учбової літератури, 2012. – 240 с.
https://chtyvo.org.ua/authors/Buinytska_Oksana/Informatsiini_tekhnolohii_ta_tekhnichni_zasoby_navchannia/
15. Кінцель Д. А., Кузнєцов А. В. (2007). Нечисловий підхід до тестової моделі та оцінка параметрів тесту. Освітні технології та суспільство, 10, 276-281.
. https://cyberleninka.ru/article/n/nechislovoy-podhod-k-modelyam-testirovaniya-i-otsenivaniyu-parametrov-testov

16. Зіньковський Ю.Ф., Мірських Г.О. (2010). Методика оцінювання рівнів складності навчальних тестів. Вісник Національного технічного університету України «КПІ», серія – Радіотехніка. Радіотехнічна, 163, 41 https://cyberleninka.ru/article/n/metodika-otsinyuvannya-rivniv-skladnosti-navchalnih-testiv
17. Федорук, П.І. (2007). Адаптивні тести: статистичні методи аналізу результатів тестового контролю знань. Математичні машини та системи, 3.4, 122-138. http://www.immsp.kiev.ua/publications/articles/2007/2007_3,4/Fedoruk_034_2007.pdf

ДОДАТКИ

Код програми:

```
import os
import random  # модуль для генерації випадкових чисел
from os import path  # функціонал для взаємодії із файловою системою

import PySimpleGUI as sg  # модуль для створення графічного інтерфейсу

Q = []  # список запитань
A = []  # список відповідей
Type4A = []  # список відповідей на тести 4 типу
studentA = {}  # словник відповідей студента
owner = '+380000000000'  # номер телефону (логін) адміна системи
teacher = ['+380000000000', '123']  # приклад логіну та паролю вчителя
student = ['+380111111111', '123']  # приклад логіну та паролю студента


# Створення тесту
def create():
    test_type = 0  # тип тесту

    # створення та розташування елементів інтерфейсу
    layout = [
        [sg.Text('Назва тесту')],
        [sg.InputText('', key='testName')],
        [sg.Text('Введіть тип тесту')],
        [sg.Radio('Дихотомічні запитання: вибір альтернативної відповіді',
"RADIO1", key='1', default=True)],
        [sg.Radio('Запитання з вибором однієї правильної відповіді', "RADIO1",
key='2')],
        [sg.Radio('Запитання з вибором декількох правильних відповідей',
"RADIO1", key='3')],
        [sg.Radio('Встановлення відповідності', "RADIO1", key='4')],
        [sg.Radio('Встановлення правильної послідовності', "RADIO1", key='5')],
        [sg.Submit('Створити тест'), sg.Cancel('Відміна')]
    ]
    # виклик нового вікна зі створеними елементами
    window = sg.Window('Створення тесту', layout)

    while True:  # основний цикл
        event, values = window.read()  # считуємо дані з полей вікна
        if event in (None, 'Exit', 'Відміна'):
            break  # якщо користувач вирішив вийти, виходимо з циклу
```

```
        # якщо користувач вибрав створення тесту
        if event == 'Створити тест':
            #  дізнаємося тип тесту
            if values['1']:
                test_type = 1
            elif values['2']:
                test_type = 2
            elif values['3']:
                test_type = 3
            elif values['4']:
                test_type = 4
            elif values['5']:
                test_type = 5
            # якщо назва тесту не пуста
            if values['testName'] != "":
                #  перевіряємо, чи існує файл із такою назвою
                file_exists = path.exists(values['testName'] + ".txt")
                #  якщо не існує, закриваємо вікно та визиваємо функцію створення
питання
                if file_exists == False:
                    window.close()
                    createQuestion(values['testName'], test_type)
                else:  # інакше, видаємо повідомлення про помилку
                    sg.popup('Тест з такою назвою вже існує')
            else:
                # якщо назва тексту пуста, видаємо повідомлення про помилку
                sg.popup('Поле назва тесту не може бути пустим')
    # закриваємо вікно
    window.close()


# Створення питання
def createQuestion(testName, testType):
    n = 2  # кількість відповідей
    #  створення та розташування елементів інтерфейсу
    layout = [
        [sg.Text('Запитання'), sg.InputText('12345', key='question')],
        [sg.Col([
            [sg.Text('Відповідь 1'), sg.InputText(key=f'answer1'), sg.Text('Кількість
балів'),
             sg.InputText('0', key=f'point1', size=(10, 20))],
            [sg.Text('Відповідь 2'), sg.InputText(key=f'answer2'), sg.Text('Кількість
балів'),
             sg.InputText('0', key=f'point2', size=(10, 20))]
        ], key='COL')],
```

```
        [sg.Submit('Додати запитання'), sg.Cancel('Закінчити введення')]
    ]
    #  якщо тип тесту - 1, тоді додаємо нову кнопку додавання відповіді
    if testType != 1:
        layout.insert(0, [sg.Button('Додати відповідь')])

    #  створюємо нове вікно із зазначеними елементами
    window = sg.Window('Додати запитання', layout)

    while True:  # основний цикл
        # читаємо дані, які ввів користувач
        event, values = window.read()

        # якщо користувач вирішив вийти з програми, закінчуємо цикл
        if event in (None, 'Exit'):
            break
        # якщо користувач вирішив додати відповідь
        if event == 'Додати відповідь':
            n += 1  # збільшуємо кількість відповідей на 1
            # додаємо нові елементи до вікна
            window.extend_layout(window['COL'], [
                [sg.Text(f'Відповідь {n}'), sg.InputText(key=f'answer{n}'),
sg.Text('Кількість балів'),
                 sg.InputText('0', key=f'point{n}', size=(10, 20))]
            ])
        # якщо користувач вирішив закінчити введення
        if event == 'Закінчити введення':
            f = open(testName + ".txt", 'w', encoding='utf-8')  # відкриваємо новий файл
з назвою, яка співпадає з назвою тесту
            f.write(f'{str(testType)}\n')  # записуємо в файл тип тесту
            f.write(f'{owner}\n')  # записуємо в файл логін адміну
            # записуємо в файл усі запитання та варіанти відповідей
            for i in range(len(Q)):
                f.write('Запитання:\n')
                f.write(f'{Q[i]}\n')
                f.write('Варіанти відповідей:\n')
                for j in range(len(A[i])):
                    f.write(f'{A[i][j][0]}_{A[i][j][1]}\n')
            f.close()  # закриваємо файл
            window.close()  # закриваємо вікно
            create()  # повертаємося до вікна створення тесту

        # якщо користувач вирішив додати запитання
        if event == 'Додати запитання':
            window.close()  # закриваємо вікно
```

```
        Q.append(values['question'])  # додаємо запитання

        # записуємо відповіді у список
        answers = []
        for i in range(n):
            if values[f'answer{i + 1}'] != "":
                answers.append([values[f'answer{i + 1}'], values[f'point{i + 1}']])
        A.append(answers)  # додаємо відповіді до головного списку відповідей
        createQuestion(testName, testType)  # викликаємо функцію створення питання
    # закриваємо вікно
    window.close()


# Додати відповіді для тестів типу 1, 2, 3
def appendAnswers(values, nQ):
    answers = []  # сюди будемо записувати відповіді
    for i in range(len(A[nQ])):
        if values[f'{i}']:  # якщо у словнику values існують значення за ключем i
            answers.append(A[nQ][i][0])  # додаємо до списку відповідей значення з основного списку відповідей
    studentA[nQ] = answers  # записуємо у список відповідей студента список відповідей


# Додати відповіді для тестів типу 4
def appendAnswers4(values, nQ):
    answers = []  # сюди будемо записувати відповіді
    for i in range(len(Type4A)):
        if values[f'{i}']:  # якщо у словнику values існують значення за ключем i
            answers.append(Type4A[i])  # додаємо до списку відповідей значення зі списку відповідей тесту 4 типу

    # видаляємо значення зі списку відповідей тесту 4 типу
    for i in range(len(answers)):
        Type4A.remove(answers[i])

    studentA[nQ] = answers  # записуємо у список відповідей студента список відповідей


# Додати відповіді для тестів типу 5
def appendAnswers5(values, nQ, answers):
    ans = []  # сюди будемо записувати відповіді
    # намагаємося додати відповіді
```

```
    try:
        # зчитуємо дані, які ввів користувач
        for i in range(len(answers)):
            for j in range(len(answers)):
                # якщо значеня не пусте та дорівнює (i + 1)
                if values[f'{j}'] != "" and int(values[f'{j}']) == (i + 1):
                    ans.append(answers[j][0])  # додаємо дані до списку відповідей
                    break  # виходимо з циклу
        studentA[nQ] = ans  # додаємо відповіді до відповідей студента
        return True  # повертаємо True, що означає, що функції вдалося зчитати
дані
    except ValueError:
        # якщо користувач ввів не тільки цифри, виникне ValueError та користувач
побачить помилку
        sg.popup('Введені недопустимі символи. Використовуйте лише цифри.')
        return False  # повертаємо False, що означає, що функції не вдалося зчитати
дані


# Відобразити питання тесту в залежності від типу тесту
def showTest(testType, nQ):
    # якщо тип тесту 1 або 2
    if testType == 1 or testType == 2:
        # створюємо елементи
        layout = [
            [sg.Text(f'{Q[nQ]}')],
            [sg.Submit('Наступне питання'), sg.Submit('Закінчити тест'),
sg.Cancel('Відміна')]
        ]
        # для кожної відповіді додаємо до інтерфейсу Radio елемент
        for i in range(len(A[nQ])):
            if nQ > 0 and nQ < len(studentA) and studentA[nQ][0] == A[nQ][i][0]:
                layout.insert(i + 1,
                              [sg.Radio(f'{A[nQ][i][0]}', "RADIO1", key=f'{i}', default=True)]
                              )
            else:
                layout.insert(i + 1,
                              [sg.Radio(f'{A[nQ][i][0]}', "RADIO1", key=f'{i}', default=False)]
                              )
    # якщо тип тесту 3
    elif testType == 3:
        # створюємо елементи
        layout = [
            [sg.Text(f'{Q[nQ]}')],
```

```
            [sg.Submit('Наступне питання'), sg.Submit('Закінчити тест'),
sg.Cancel('Відміна')]
        ]
        # для кожної відповіді додаємо до інтерфейсу Checkbox елемент
        for i in range(len(A[nQ])):
            layout.insert(i + 1,
                          [sg.Checkbox(f'{A[nQ][i][0]}', key=f'{i}', default=False)]
                          )
    # якщо тип тесту 4
    elif testType == 4:
        # створюємо елементи
        layout = [
            [sg.Text(f'{Q[nQ]}')],
            [sg.Submit('Наступне питання'), sg.Submit('Закінчити тест'),
sg.Cancel('Відміна')]
        ]
        # для кожної відповіді додаємо до інтерфейсу Checkbox елемент
        for i in range(len(Type4A)):
            layout.insert(i + 1,
                          [sg.Checkbox(f'{Type4A[i]}', key=f'{i}', default=False)]
                          )
    # якщо тип тесту 5
    elif testType == 5:
        # створюємо елементи
        layout = [
            [sg.Text(f'{Q[nQ]}')],
            [sg.Submit('Наступне питання'), sg.Submit('Закінчити тест'),
sg.Cancel('Відміна')]
        ]
        # випадково перемішуємо відповіді
        answers = A[nQ][:]
        random.shuffle(answers)
        # для кожної відповіді додаємо до інтерфейсу поле вводу
        for i in range(len(answers)):
            layout.insert(i + 1,
                          [sg.Text(f'{answers[i][0]}'), sg.InputText('', key=f'{i}', size=(10,
20))]
                          )
    # відкриваємо нове вікно із створеними елементами
    window = sg.Window('Тестування', layout)
    while True:  # основний цикл
        event, values = window.read()  # зчитуємо дані, які ввів користувач

        if event in (None, 'Exit', 'Відміна'):
            break  # якщо користувач вирішив вийти з програми, закінчуємо цикл
```

```
    # якщо користувач вирішив переейти до наступного питання
    if event == 'Наступне питання':
        if nQ < (len(Q) - 1):  # якщо питання не останнє
            # считуємо дані, які ввів користувач та додаємо їх до списку
            if testType == 1 or testType == 2 or testType == 3:  # якщо тип питання
1, 2 або 3
                appendAnswers(values, nQ)  # викликаємо функцію для типу 1, 2 або
3
            elif testType == 4:  # якщо тип питання 4
                appendAnswers4(values, nQ)  # викликаємо фукнцію для типу 4
            elif testType == 5:  # якщо тип питання 5
                is_correct = appendAnswers5(values, nQ, answers)  # викликаємо
функцію для типу 5
                if not is_correct:  # якщо функція повернула False, користувач
помилився при вводі даних
                    continue  # переходимо до наступної ітерації циклу без
інкременту, тобто, зашилаємося на тому ж самому питанні
            nQ += 1  # збільшуємо номер питання на 1
            window.close()  # закриваємо вікно із поточним питанням
            showTest(testType, nQ)  # викликаємо функцію для показу наступного
питання
        else:
            sg.popup('Питань більше немає')  # виводимо користувачу
повідомлення, що питань більше немає
    # якщо користувач вирішив повернутися до наступного питання
    if event == 'Попереднє питання':
        # считуємо дані, які ввів користувач та додаємо їх до списку
        if testType == 1 or testType == 2 or testType == 3:
            appendAnswers(values, nQ)  # викликаємо функцію для типу 1, 2 або 3
        elif testType == 4:  # якщо тип питання 4
            appendAnswers4(values, nQ)  # викликаємо фукнцію для типу 4
        elif testType == 5:  # якщо тип питання 5
            is_correct = appendAnswers5(values, nQ, answers)  # викликаємо
функцію для типу 5
            if not is_correct:  # якщо функція повернула False, користувач
помилився при вводі даних
                continue  # переходимо до наступної ітерації циклу без інкременту,
тобто, зашилаємося на тому ж самому питанні
        if nQ > 0:  # якщо номер питання быльше 0
            nQ -= 1  # зменшуємо номер питання на 1
            window.close()  # закриваємо поточне вікно
            showTest(testType, nQ)  # викликаємо функцію для показу
попереднього питання
    # якщо користувач вирішив закінчити тест
    if event == 'Закінчити тест':
```

```
        sum = 0  # сума правильних відповідей
        max = 0  # максимальний бал
        if len(studentA) > 0:  # якщо кількість відповідей більше 0
            if testType == 1 or testType == 2 or testType == 3:
                appendAnswers(values, nQ)  # для тестів 1, 2 та 3 типу викликаємо
відповідну функцію
                # для кожного запитання розраховуємо суму та максимальний бал
                for i in range(len(Q)):
                    maxPoint = 0
                    point = 0
                    n = 0
                    wrong = 0
                    for j in range(len(A[i])):
                        maxPoint += float(A[i][j][1])
                        if float(A[i][j][1]) > 0:
                            n += 1
                        if len(studentA) > i:
                            for k in range(len(studentA[i])):
                                if studentA[i][k] == A[i][j][0]:
                                    point += float(A[i][j][1])
                                    if point == 0:
                                        wrong += 1
                    max += maxPoint / n
                    if len(studentA) > i and len(studentA[i]) != 0:
                        sum += point / maxPoint - wrong * (point / maxPoint) /
len(studentA[i])
            # якщо тип тесту 4
            elif testType == 4:
                appendAnswers4(values, nQ)  # викликаємо функцію для типу тесту 4
                # для кожного запитання розраховуємо суму та максимальний бал
                for i in range(len(Q)):
                    maxPoint = 0
                    n = 0
                    point = 0
                    A0 = set()
                    for k in range(len(A[i])):
                        A0.add(A[i][k][0])
                    if len(studentA) > i:
                        Al = set(studentA[i])
                    else:
                        Al = set()
                    for j in range(len(A[i])):
                        maxPoint += float(A[i][j][1])
                        if float(A[i][j][1]) > 0:
                            n += 1
```

```
                    point = 2 * len(A0 & Al) / (len(A0) + len(Al))
                    sum += point
                    if n > 0:
                        max += maxPoint / n
            # якщо тип тесту 5
            elif testType == 5:
                is_correct = appendAnswers5(values, nQ, answers)  # викликаємо функцію для тесту 5
                if not is_correct:  # якщо функція повернула False, користувач помилився при вводі даних
                    continue  # переходимо до наступної ітерації циклу без інкременту, тобто, зашилаємося на тому ж самому питанні
                # для кожного запитання розраховуємо суму та максимальний бал
                for i in range(len(Q)):
                    maxPoint = 0
                    n = 0
                    point = 0
                    # Метріка Кука
                    dK = 0
                    dM = len(A[i])
                    for j in range(len(A[i])):
                        maxPoint += float(A[i][j][1])
                        if float(A[i][j][1]) > 0:
                            n += 1
                        r1 = 0
                        if A[i][j][0] in studentA[i]:
                            r1 = studentA[i].index(A[i][j][0])
                        dK += abs(j - r1)
                    point = maxPoint / n * (1 - float(dK) / dM)
                    if point < 0: point = 0
                    sum += point
                    max += maxPoint / n
            #  фінальна оцінка (у відсотках) = сума правильних відповідей / максимальній бал * 100
            grade = sum / max * 100
            # виводимо на екран фінальну оцінку у двох шкалах та відсотках
            sg.popup(
                f'Відсоток правильних відповідей: {grade:.2f}%.\nВаша оцінка: {round(grade * 2)}/200 або {round(grade * 0.05)}/5')
            window.close()  # закриваємо вікно
            showTestList()  # повертаємося до вікна показу списку тестів
    window.close()  # закриваємо вікно


# Считування даних тесту з файлу
```

```
def readTest(testName):
    fQ = 0
    fA = 0
    f = open(testName + ".txt", 'r', encoding='utf-8')  # відкриваємо файл
    i = 0
    j = 0
    # видаляємо зі списків дані попереднього тесту
    Q.clear()
    A.clear()
    answers = []
    testType = 0
    testOwner = ""
    # считуємо дані
    for line in f:
        if testType == 0:
            testType = int(line)  # тип тесту
        elif testType != 0 and testOwner == "":
            testOwner = line.replace("\n", "")  # володар тесту
        elif line == 'Запитання:\n':
            fQ = 1
            if fA == 1:
                A.append(answers)
            answers = []  # запитання
        elif line == 'Варіанти відповідей:\n':
            fA = 1
        elif fQ == 1:
            Q.append(line)
            fQ = 0
            fA = 0
        elif fA == 1 and fQ == 0:
            answer, point = line.split("_", 1)  # відповіді
            answers.append([answer, point.replace("\n", "")])
            fQ = 0
    if fA == 1 and fQ == 0:
        A.append(answers)  # додаємо відповіді до основного списку
    f.close()  # закриваємо файл

    return testType  # повертаємо тип тесту


# Відобразити список тестів
def showTestList():
    layout = []  # сюди будемо додавати елементи інтерфейсу
    testList = []  # список тестів
    i = 0
```

```
    # зчитуємо усі текстові файли у директорії
    for x in os.listdir():
        if x.endswith(".txt"):
            name = x.replace(".txt", "")
            if i == 0:
                layout.append([sg.Radio(f'{name}', "RADIO1", default=True, key=f'{i}')])
# додаємо елемент
            else:
                layout.append([sg.Radio(f'{name}', "RADIO1", default=False, key=f'{i}')])
# додаємо елемент
            testList.append(name)  # додаємо ім'я тесту
            i += 1

    layout.append([sg.Submit('Почати'), sg.Cancel('Відміна')])  # додаємо кнопку
старту
    window = sg.Window('Список тестів', layout)  # відкриваємо вікно зі списком
тестів
    while True:  # основний цикл
        event, values = window.read()  # зчитуємо дані користувача
        if event in (None, 'Exit', 'Відміна'):
            break  # якщо користувач вирішив вийти з прогрмаи, виходимо з циклу
        # якщо користувач вирішив почати
        if event == 'Почати':
            test = ""
            # визначаємо, який тест вибрав користувач
            for j in range(i):
                if values[f'{j}']:
                    test = testList[j]
                    break
            studentA.clear()  # видаляємо дані попереднього тесту
            testType = readTest(test)  # зчитуємо тип тесту
            if testType == 4:  # якщо тип тесту 4
                Type4A.clear()  # видаляємо дані попереднього тесту типу 4
                #  додаємо відповіді до списку типу 4
                for i in range(len(A)):
                    for j in range(len(A[i])):
                        Type4A.append(A[i][j][0])
                random.shuffle(Type4A)  # перемішуємо іх

            window.close()  # закриваємо вікно
            showTest(testType, 0)  # виводимо тест на екран
    window.close()  # закриваємо вікно


# елементи вікна логіну
```

```
layout = [
    [sg.Text('Номер телефону'), sg.InputText('+380111111111', key='owner'),
     ],
    [sg.Text('Пароль'), sg.InputText('123', key='password')
     ],
    [sg.Radio('Учень', "RADIO1", default=True, key='student'),
     sg.Radio('Вчитель', "RADIO1", default=False, key='teacher')],
    [sg.Submit('Ввійти'), sg.Cancel('Відміна')]
]
window = sg.Window('Авторизація', layout)  # визиваємо перше вікно
while True:  # основний цикл
    event, values = window.read()
    if event in (None, 'Exit', 'Відміна'):
        break  # якщо користувач вирішив вийти, виходимо з циклу
    # якщо користувач вирішив ввійти, зчитуємо введені дані та шукаємо серед студентів або викладачів таку пару логін-пароль
    if event == 'Ввійти':
        if values['student'] == True and values['owner'] == '+380111111111' and values['password'] == '123':
            showTestList()  # якщо знайшли пару логін-пароль студента, викликаємо інтерфейс студента
        elif values['teacher'] == True and values['owner'] == '+380000000000' and values['password'] == '123':
            create()  # якщо знайшли пару логін-пароль вчителя, викликаємо інтерфейс вчителя
        else:
            sg.popup('Номер телефону або пароль введено невірно')  # якщо не знайшли, видаємо помилку

window.close()  # закриваємо вікно
```