НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

«На правах рукопису»
УДК 004.89

«До захисту допущено»
Завідувач кафедри
_____ Коваль О. В.
(підпис) (ініціали, прізвище)
“___”_____20__ р

Магістерська дисертація

зі спеціальності **121 Інженерія програмного забезпечення**
за спеціалізацією Інженерія програмного забезпечення розподілених систем
на тему: Система автоматизованого контролю знань на онтологічно-орієнтованому порталі з програмування
Виконав: студент 6 курсу, групи ТВ-82мп
(шифр групи)

Нероденко Владислав Вікторович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент, к.т.н., Титенко С. В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет **Теплоенергетичний**
(повна назва)

Кафедра **Автоматизації проектування енергетичних процесів і систем**
(повна назва)

Рівень вищої освіти другий (магістерський) за освітньо-професійною програмою

Спеціальність **121 Інженерія програмного забезпечення**
(код і назва)

Спеціалізація **Інженерія програмного забезпечення розподілених систем**

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Коваль О. В.
(підпис) (ініціали, прізвище)
“___”_____20__ р

ЗАВДАННЯ

на магістерську дисертацію студенту

Нероденку Владиславу Вікторовичу
(прізвище, ім’я, по батькові)

1. Тема дисертації “Система автоматизованого контролю знань на онтологічно-орієнтованому порталі з програмування
Науковий керівник Титенко Сергій Володимирович, к.т.н. , доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету № ___ від “ 4 “ листопада 2019 р.
2. Строк подання студентом дисертації _____
3. Об’єкт дослідження Система автоматизованого контролю знань
4. Предмет дослідження Методи для автоматизованої побудови тестових

завдань

5. Перелік завдань, які необхідно розробити:

1) Проаналізувати концепцію створення тестових завдань
2) Проаналізувати технологію генерації завдань на базі ПТМ
3) Проаналізувати ряд методів оптимізації ПТМ

6. Орієнтований перелік ілюстративного матеріалу:

1) Діаграма послідовності в нотації UML
2) Діаграма класів програмної реалізації
3) Схема бази даних

7. Орієнтований перелік публікацій

Generation of tests of various complexity levels in e-learning system based on educational text formalization model(Modern Aspects of Software Development: Proceedings of VI International Scientific and Practical Virtual Conference of Software Development Specialists)

8. Дата видачі завдання “ ____ ” _____ 2018 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Строки виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	28.09.19р.	
2	Опрацювання літературних джерел	01.10.18 р. – 03.02.19р.	
3	Підготовка матеріалів дисертації	04.02 – 31.05.19 р.	
4	Підготовка доповідей на конференції	01.05 – 20.05.19 р.	
5	Розробка програмного продукту	03.06 – 25.10.19 р.	
6	Переддипломна практика	2.09 – 25.10.19 р.	
7	Захист програмного продукту	25.10.19р.	
8	Розробка стартап-проекту	11.11 – 19.11.19 р.	
9	Передзахист	22.11.19 р.	
10	Оформлення дисертації	21.11- 29.11.19 р.	
11	Захист	18.12.19 р.	

Студент _____ Нероденко В. В.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Титенко С. В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дисертаційна робота складається зі вступу, 6 розділів, висновків, списку використаних джерел з 30 найменувань. Обсяг дисертації становить 92 сторінки, робота має 44 рисунки, 12 таблиць, 40 формул, 1 додаток.

Актуальність теми. Однією з ключових складових дистанційного навчання є моніторинг та перевірка знань. Дистанційне навчання дозволяє навчатись або перевіряти знання віддалено, що може бути дуже корисним у випадках неможливості очної присутності. Найпоширенішим способом перевірки знань в навчальних системах є тестування. Значна кількість досліджень в галузі тестування знань зосереджена на проблемах якості та валідності тестів. Натомість сучасний темп оновлення професійних галузей зумовлює значущість задачі автоматизованої побудови тестів та тестових завдань, що пришвидшить підготовку навчальних курсів та середовищ з сучасних напрямків навчання. На разі тестових завдань, що створюються вручну, тобто викладачем відповідної предметної дисципліни або експертом в певній галузі мають високу якість та зрозумілість. Проте, при такому підході ми отримуємо дві проблеми: трудомісткість самого процесу, недобросовісне проходження тестових завдань. Був запропонований метод автоматизованої побудови засобів тестування на основі моделі формалізації дидактичного тексту для вирішення проблем зазначених вище. В основі методу — формалізація навчальних матеріалів на базі понятійно-тезисної моделі із подальшим застосуванням алгоритмів генерації та перевірки тестових завдань. Подальші дослідження застосування ПТМ для контролю знань, її модифікацій та близьких альтернатив були зосереджені головним чином на підвищенні якості генерованих завдань. Було розглянуто методи автоматизованої побудови тестових завдань на основі модифікації ПТМ з використанням ключових слів у тезах, семантичних класів та модернізація цієї моделі шляхом усунення мовної неузгодженості. Метод генерації тестових завдань на основі декомпозиції тверджень з навчального тексту з використанням системи семантичних класів пропонує засоби керування складністю завдань. Натомість представлений метод

значним чином ускладнює попередню формалізацію дидактичного тексту у порівнянні з базовою моделлю ПТМ та вимагає значної кількості однорідних мовних конструкцій в навчальному тексті. Питання диференціації складності тестів в ПТМ потребує додаткових досліджень.

Мета й завдання дослідження. Головною метою дисертаційної роботи є створення системи автоматизованого контролю знань на онтологічно-орієнтованому порталі з програмування та удосконалення методів автоматизованої побудови тестів різної складності на базі понятійно-тезисної моделі. Ставиться задача забезпечити відповідність типів тестових завдань різним когнітивним рівням за шкалою Блума, сформувати моделі тестів, що диференціюватимуться за рівнем складності

Об'єкт дослідження. Системи автоматизованого контролю знань.

Предмет дослідження. Методи автоматизації побудови тестових завдань.

Методи дослідження. Метод об'єктно-орієнтованого аналізу для опису об'єктів предметної області, об'єктно-орієнтованого програмування для побудови модулів формування сценаріїв поведінки елементів системи, автоматизованої побудови тестових завдань.

Інноваційна новизна одержаних результатів полягає у вирішенні актуальної проблеми автоматизованої побудови тестів, а саме:

- Удосконалено методи автоматизованої побудови тестових завдань на базі ПТМ за рахунок сформованих моделей тестів, що диференціюються за рівнем складності відповідно до когнітивних цілей за шкалою Блума.;
- Удосконалено тестові шаблони за рахунок модифікації їх структури шляхом введення типів завдань, які можна застосовувати до шаблону.

Практичне значення одержаних результатів полягає в розробленні програмної системи для забезпечення ефективного та якісного контролю знань на базі ПТМ.

Апробація результатів дисертації. Результати досліджень, включених до дисертації, представлені на VI міжнародній науково-практичній віртуальній

конференції фахівців з розробки програмного забезпечення на тему «Сучасні аспекти розробки програмного забезпечення» 2019 року.

Публікації. Тези доповідей публікувалися в збірнику «Сучасні аспекти розробки програмного забезпечення» VI міжнародної науково-практичної віртуальної конференції фахівців з розробки програмного забезпечення.

Ключові слова. АВТОМАТИЗОВАНИЙ КОНТРОЛЬ ЗНАНЬ; ТЕСТОВІ ЗАВДАННЯ; ПОНЯТІЙНО-ТЕЗИСНА МОДЕЛЬ; ШКАЛА БЛУМА.

ABSTRACT

The dissertation consists of an introduction, 6 sections, conclusions, a list of used sources of 30 titles. The scope of the dissertation is 92 pages, the work has 44 drawings, 12 tables, 40 formulas, 1 application.

Actuality of theme. Monitoring and testing is one of the key components of distance learning. Distance learning allows you to study or test your knowledge remotely, which can be very useful in cases where there is no full-time presence. Testing is the most common way of testing knowledge in training systems. A considerable amount of research in the field of knowledge testing focuses on the problems of quality and validity of tests. On the other hand, the modern pace of updating of the professional branches makes the importance of the task of automated construction of tests and test tasks, which will accelerate the preparation of training courses and environments in modern areas of study. In the case of manual test tasks, ie, a teacher of the relevant subject discipline or expert in a particular field have high quality and clarity. However, with this approach we get two problems: the complexity of the process itself, the unfair passage of test tasks. An automated method of constructing testing tools based on a didactic text formalization model was proposed to solve the problems mentioned above. The method is based on the formalization of educational materials on the basis of CTM with the subsequent application of algorithms of generation and verification of test tasks. Further studies on the use of conceptual-theses model to control knowledge, its modifications, and close alternatives have focused mainly on improving the quality of the tasks generated. The methods of automated construction of test tasks based on the modification of CTM model using keywords in theses, semantic classes and modernization of this model by eliminating language mismatch were considered. The method of generating test tasks based on the decomposition of statements from a training text using a system of semantic classes offers a means of managing the complexity of the tasks. Instead, the presented method significantly complicates the prior formalization of the didactic text compared to the basic CTM and requires a considerable number of homogeneous linguistic constructs

in the educational text. The question of the differentiation of the complexity of tests in CTM requires further research.

The purpose and objectives of the study. The main purpose of the dissertation is to create a system of automated knowledge control on an ontology-oriented portal for programming and improvement of methods of automated test construction of various complexity on the basis of conceptual-thesis model. The task is to ensure that the types of test tasks correspond to different cognitive levels on the Bloom scale, to create models of tests that will differentiate by the level of difficulty.

Object of study. Automated knowledge control systems.

Subject of study. Methods of automation of construction of test tasks.

Research methods. Object-oriented analysis method for describing domain objects, object-oriented programming for building modules for generating scripts for the behavior of system elements, automated construction of test tasks.

The innovative novelty of the obtained results is to solve the urgent problem of automated test construction, namely:

- Improved methods for automated construction of CTM-based test tasks at the expense of established test models that differentiate by difficulty according to cognitive goals on the Bloom scale;
- Improved test templates by modifying their structure by introducing the types of tasks that can be applied to the template.

The practical significance of the obtained results consist in in the development of a software system to ensure effective and high quality control of knowledge based on CTM.

The practical significance of the results. The results of the research included in the dissertation are presented at the VI International Scientific and Practical Virtual Conference of Software Development Specialists on the topic "Modern Aspects of Software Development" in 2019.

Publications. The abstracts were published in the collection "Modern Aspects of Software Development" of VI International Scientific and Practical Virtual Conference of Software Engineering Specialists.

Keywords. AUTOMATED KNOWLEDGE CONTROL; TEST TASKS; CONCEPTUAL-THESES MODEL; BLOOM SCALE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ООП – об'єктно-орієнтоване програмування

БД – база даних

ПТМ – понятійно-тезисна модель

ВСТУП

Сучасні засоби навчання широко спираються на використання інтернет-простору, стимулюючи подальший розвиток компонентів дистанційних навчальних систем. Однією з ключових складових дистанційного навчання є моніторинг та перевірка знань. Дистанційне навчання дозволяє навчатись або перевіряти знання віддалено, що може бути дуже корисним у випадках неможливості очної присутності. Найпоширенішим способом перевірки знань в навчальних системах є тестування. Значна кількість досліджень в галузі тестування знань зосереджена на проблемах якості та валідності тестів [1]. Натомість сучасний темп оновлення професійних галузей зумовлює значущість задачі автоматизованої побудови тестів та тестових завдань, що пришвидшить підготовку навчальних курсів та середовищ з сучасних напрямків навчання. На разі тестових завдань, що створюються вручну, тобто викладачем відповідної предметної дисципліни або експертом в певній галузі мають високу якість та зрозумілість. Проте, при такому підході ми отримуємо дві проблеми: трудомісткість самого процесу, недобросовісне проходження тестових завдань. Був запропонований метод автоматизованої побудови засобів тестування на основі моделі формалізації дидактичного тексту [2, 3] для вирішення проблем зазначених вище. В основі методу — формалізація навчальних матеріалів на базі понятійно-тезисної моделі із подальшим застосуванням алгоритмів генерації та перевірки тестових завдань. Подальші дослідження застосування ПТМ для контролю знань, її модифікацій та близьких альтернатив були зосереджені головним чином на підвищенні якості генерованих завдань [4-6]. В даних роботах було розглянуто методи автоматизованої побудови тестових завдань на основі модифікації ПТМ з використанням ключових слів у тезах, семантичних класів та модернізація цієї моделі шляхом усунення мовної неузгодженості. У роботі [5] подано метод генерації тестових завдань на основі декомпозиції тверджень з навчального тексту з використанням системи семантичних класів. Тут пропонуються засоби керування

складністю завдань. Натомість представлений метод значним чином ускладнює попередню формалізацію дидактичного тексту у порівнянні з базовою моделлю ПТМ та вимагає значної кількості однорідних мовних конструкцій в навчальному тексті. Питання диференціації складності тестів в ПТМ потребує додаткових досліджень.

Відповідно до переглянутої таксономії Блума [7] результати навчальної діяльності диференціюються за допомогою упорядкованої шкали когнітивних цілей. На етапі формування цієї шкали вона мала три основні поділи: знаю, відчуваю, створюю. Проте, з подальшими дослідженнями вона була розширена і цілі навчання стали впорядковуватись таким чином: знати, розуміти, застосовувати, аналізувати, оцінювати та створювати [7]. Саме ця шкала дозволяє оцінювати знання за рівнем складності, тобто не є загальною усередненою. Викладач при створенні тестів в основному орієнтується на середнього учня. В разі учня нижче середнього при проходженні тесту він може отримати негативну оцінку при використанні шкали оцінювання в балах. Шкала Блума дозволить оцінювати рівень знань більш об’єктивно. Актуальною є така функціональна можливість системи тестування, що дозволить визначати рівень засвоєння навчального предмету відповідно до таксономії педагогічних цілей. Повноцінне виконання даного завдання засобами тестування, особливо на базі автоматично генерованих завдань, є важко досяжним, натомість поступ в цьому напрямку є необхідним та актуальним. Отже, генерація тестових завдань на базі ПТМ потребує доопрацювання для реалізації диференціації складності тестів, що дозволить оцінювати рівень знань відповідно до когнітивних цілей.

1. ЗАДАЧА РОЗРОБКИ СИСТЕМИ АВТОМАТИЗОВАНОГО КОНТРОЛЮ ЗНАНЬ НА ОНТОЛОГІЧНО-ОРІЄНТОВАНОМУ ПОРТАЛІ З ПРОГРАМУВАННЯ

Головною метою розробки є створення системи автоматизованого контролю знань на онтологічно-орієнтованому порталі з програмування та удосконалення методів автоматизованої побудови тестів різної складності на базі понятійно-тезисної моделі. Ставиться задача забезпечити відповідність типів тестових завдань різним когнітивним рівням за шкалою Блума, сформувати моделі тестів, що диференціюватимуться за рівнем складності, та здійснити емпіричну перевірку запропонованих рішень.

Програмний продукт повинен мати такі функції:

- можливість проходження тесту;
- перевірка тестового завдання та відображення правильних та неправильних відповідей;
- перегляд результату усього тесту;
- запис результатів тестування до бази даних;
- можливість перегляду результатів пройдених тестів;
- система повинна давати рекомендації щодо необхідності повторення певних тем після проходження тесту, якщо на якесь завдання було дано неправильну відповідь.

Структурно система повинна бути розподілена на серверну та користувацьку частини, які будуть взаємодіяти між собою.

Результуюча система має забезпечувати стабільність роботи та гнучкість архітектури, що буде надавати змогу додавати новий функціонал не порушуючи основну структуру програми.

2. АНАЛІЗ ПРОБЛЕМИ ПОБУДОВИ ТЕСТОВИХ ЗАВДАНЬ

Однією з найбільш поширених форм перевірки знань є комп'ютерне тестування. З розвитком нових технологій і підвищенням ступеня інформатизації суспільства і освіти проблема ефективного контролю знань набуває особливого значення.

Наступні підходи до генерації тестових завдань набули поширення на практиці і в дослідженнях: параметризовані тести; семантичні мережі; понятійно-тезисна модель (ПТМ) і її модифікації.

При створенні систем автоматизованого тестування потрібно вирішити наступні проблеми:

- вибір методу створення тестових завдань;
- спосіб інтерпретації результатів.

2.1. Якість та валідність тестових завдань

Загалом, тестові завдання повинні оцінювати досягнення навчальних цілей. Спочатку потрібно визначити важливі аспекти теми (поняття та концептуальні відношення). Має бути точне акцентування на важливих аспектах навчання. Необхідно виміряти відповідний рівень знань учнів. За припущенням [8] кожен тест містить в собі елемент суб’єктивності.

Тестові питання будуть зосереджені на відповідній інтелектуальній діяльності, починаючи від простого згадування фактів до вирішення проблем, критичного мислення та міркувань. Пізнавальна складність стосується різних рівнів навчання, які можна перевірити. Коли викладач в основному переймається тим, що учням потрібно запам'ятовувати факти, тест повинен перевіряти просте пригадування матеріалу. Якщо викладач намагається розвивати аналітичні

навички, тест має перевіряти не вгадування або запам’ятовування, а здатність учня до системного аналізу предметної області. Під час конвенції 1948 р. Американської психологічної асоціації група освітянських психологів вирішила, що було б корисно класифікувати різні рівні знань, яких студенти можуть досягти в курсі. У 1956 р., Після широких досліджень освітніх цілей, група науковців опублікувала свої висновки у книзі за редакцією доктора Бенджаміна С. Блума [7], професора з Гарварду. Таксономія навчальних цілей Блума перераховує шість типів знань:

- знання – запам’ятовування і відтворення учнями змісту навчальної інформації, включаючи факти, поняття, терміни й теорії;
- розуміння – здатність учнів сприймати викладене й передавати в іншій формі, встановлювати сенс інформації, прогнозувати, виходячи з раніше описаної інформації;
- застосування – вміння без зовнішньої підказки застосувати у новій ситуації знання, набуті раніше, використовувати теоретичні знання у життєвій ситуації;
- аналіз – уміння учнів розділяти матеріал на окремі складові, порівнювати частини, встановлювати їх взаємозв’язки, розуміючи модель та структуру їхньої організації;
- синтез – здатність учнів до творчого поєднання частин або елементів у єдине ціле з іншими властивостями;
- оцінка – уміння учнів робити кількісні або якісні оцінки, що основані на використанні критеріїв або стандартів та формулювати цілісні судження про ідеї, дослідження, рішення, методи.

Ці рівні знань допомагають класифікувати тестові питання. Перший крок з планування тесту [8] – окреслити фактичний зміст курсу, який охоплюватиме тест. Зручним способом досягнення цього є виділення декількох хвилин після кожного навчального курсу, щоб перерахувати на індексній картці важливі поняття, які охоплені в занятті. Ці картки потім можуть бути використані пізніше як джерело тестових завдань. Ще більш сумлінним підходом було б побудувати тестові

завдання самостійно після кожного заняття. Перевага будь-якого з цих підходів полягає в тому, що отриманий тест, ймовірно, буде кращим представленням курсу занять.

Необхідно відповісти на деякі важливі запитання щодо якості змісту тесту:

- які специфікації тесту;
- які вміння буде перевірено;
- скільки питань і скільки областей буде охоплено;
- скільки буде розділів;
- які формати будуть використані для тестування.

Якщо викладач зосередився на війні 1812 року в більшості своїх занять, цей акцент повинен бути відображений у тесті. Тест, який охоплює набагато ширший період, буде вважатися студентами несправедливим.

Щоб досягти змістовності [8] учнів не слід змушувати здогадуватися, що буде на тесті. Очевидним рішенням цієї проблеми є дати студентам конкретні навчальні запитання, а потім скласти тест із навчальних питань. Іноді це піддається критиці як викладання тесту, так, ніби наявність навчальних питань заохочує поверховий підхід. Це може бути актуальним, якщо навчальних питань дуже мало. Однак якщо вчитель пропонує запитання щодо всіх найважливіших тем у завданні, то викладання тесту – це викладання курсу.

Мовні вимоги [8] мають будуть зрозумілими студентам. Тестові завдання повинні бути викладені простою, зрозумілою мовою, без нефункціональних матеріалів та сторонніх доказів. Тестові складові також повинні бути вільними від расових, етнічних та сексуальних упереджень. Крім цих двох кваліфікацій, мовні знання студентів впливають на їхню ефективність проходження тесту. Лексика (нечасто вживання; нелітеральне вживання) та синтаксис тесту (атипові частини мови; складні структури) можуть створювати мовні бар'єри. Модифікації тесту для студентів з обмеженим рівнем володіння англійською мовою включають: оцінювання рідною мовою; текстові зміни в лексиці; модифікація мовної складності; додавання візуальних опор; використання словників рідною мовою;

використання словників англійською мовою; мовна модифікація напрямків тестування; та додаткові приклади / завдання.

Дуже часто тести створюються такими, що їх синтаксичні структури складні або є нетиповими. Щоб зменшити фрустрацію учнів, необхідно уникати даних проблем. Ці пункти прийнятні з теоретичної точки зору, але більшість підготовлених тестувальників не люблять їх. Наприклад, чим більше предмет знає студент, тим простіше робити аргументи на користь відповідей, які вчитель може вважати неправильними. Істинно-помилкові питання є найгіршими в даному випадку. Часто значення істини ізольованого твердження досить дискусійне. Все залежить від того, як його інтерпретують, визначення ключового терміна чи контексту.

Успішне виконання тесту дозволить зробити вагомі висновки [8] щодо досягнення. Презентації, сценарії, проекти додають виміру оцінки того, що традиційне тестування не може. Вчителі можуть зробити дійсні висновки щодо досягнень легше, використовуючи оцінки автентичності та ефективності. Ці узагальнення можуть включати в себе інструктивні рішення щодо розміщення, формувальні рішення щодо оцінки та діагностичні рішення. Добре побудовані тести, чи то об’єктивні, чи орієнтовані на ефективність, дозволяють вчителям зрозуміти, чому потрібно навчати далі. Викладачі також можуть відстежувати навчання учня і можуть змінювати програму навчання за потребою.

Результативність [8] учнів оцінюватиметься таким чином, що не дає переваги факторам, які не мають значення для навчання; Схеми балів будуть аналогічно справедливими. Ось кілька основних правил справедливості:

- тестові запитання повинні відображати цілі підрозділу;
- кожен елемент тесту повинен представляти чітко сформульоване завдання;
- один елемент не повинен допомагати у відповіді на інший;
- повинно бути дозволено достатньо часу на тестування;
- присвоєння балів слід визначати перед тим, як проводити тест.

Конструктивно оцінювання вимагає від викладача надання зворотнього зв’язку (письмового та / або усного), що допомагає учням оцінити те, чого вони досягли, а чого не досягли, склавши тест. Цей зворотній зв'язок може включати в себе наступне: заохочувальні коментарі до тесту чи документа, що виражають повагу до того, що студент намагався досягти; похвала за те, що студент зробив, та пропозиції щодо підвищення продуктивності

Відповіді на тестові запитання будуть постійно складатись таким чином, щоб вони представляли те, що знають студенти. Вся суть тестування полягає у заохоченні до навчання. Хороший тест [8] розроблений за схемами , про які не легко здогадатися без належного вивчення. Можна сконструювати всі типи тестових питань, які буде важко вгадувати, а тому вимагає від учня осмислення основного фактичного матеріалу.

2.2. Калібрування тестових завдань

Спочатку визначимо кількісні дані, що відповідають рівню складності завдань. Порядок складності тестових завдань визначається відсотком випробуваних, які отримали правильний результат. Нехай r – кількість тестувальників, які виконали тестове завдання, r_v – кількість тестувальників, які виконали те саме завдання правильно. Тоді порядок складності завдання можна визначити за такою формулою [9]:

$$P = \frac{r_v}{r}\ 100\%.$$

Наприклад, якщо 73% учнів виконують тестові завдання правильно, то порядок кількісної складності завдання буде 73. Слід зазначити, що чим більший порядок складності P у кількісному виразі, тим простіше тестове завдання. Тому кількісну характеристику рівня тестового завдання, тобто ступінь складності завдання, можна визначити за такою формулою:

$$B = 100 - P = 100 - \frac{r_v}{r}\ 100\%.$$

Таким чином, тестові завдання класифікуються за рівнями складності. Надалі алгоритм «Вибіркові пороги» використовується для подальшого уточнення ступеня складності питання і, таким чином, для визначення рівня складності завдання. Коефіцієнти вибіркових порогів автори були введені авторами у попередній роботі [9].

Припустимо, що n групи беруть участь у тестуванні. Нехай r_m^k - кількість випробуваних у групі G^k (k = 1,…, n), які виконали m-те завдання, $\overline{r}_m^k$ - кількість людей правильно виконавших те саме m-те завдання. Ступінь складності m -го завдання визначається за формулою:

$$B_m^k = 100 - \frac{\overline{r}_m^k}{r_m^k}\ 100.$$

Тепер формуємо матрицю ступенів складності тестового завдання:

$$\mathrm{B} = \left\| \begin{matrix} B_1^1 & B_2^1 & B_3^1 & \cdots & B_p^1 \\ B_1^2 & B_2^2 & B_3^2 & \cdots & B_p^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ B_1^n & B_2^n & B_3^n & \cdots & B_p^n \end{matrix} \right\|,$$

де p – кількість тестових завдань. Номери стовпців матриці B відповідають номерам запропонованих завдань, а рядки - номерам груп тестувань. З відповідями на один вибір, якщо всі елементи будь-якого стовпця матриці B, наприклад j-го стовпця, задовольняють балам:

$$82 < B_j^k \leq 100\ (k = 1, \dots, n),$$

де j – завдання, яке потрібно виключити з тестів.

Крім того, виключаються ті завдання одного вибору, для яких випробувані не змогли набрати нижній поріг, тобто якщо всі елементи будь-якого стовпця, наприклад, i-й стовпець, задовольняють балам:

$$B_i^k < 48\ (k = 1, \dots, n),$$

1-е завдання також буде виключено з тестів.

Для решти задач з одним вибором, які потрапили в діапазон застосовності

тесту, тобто якщо вони задовольняють нерівності

$$48 \leq B_j^k \leq 82\ (k = 1, \dots, n),$$

для випадкового результату вводиться коефіцієнт корекції вибірковості α, де: $-3 \leq \alpha \leq +3$. Якщо для розглянутих завдань існує нерівність:

$$48 \leq B_j^k \leq 74\ (k = 1, \dots, n),$$

то, використовуючи позитивний коефіцієнт селективності, ці завдання будуть називатися "LU" (таблиця 2.1) тип тестових завдань з пороговим інтервалом 50-74. Якщо вони задовольняють нерівність

$$74 \leq B_j^k \leq 82\ (k = 1, \dots, n),$$

то, використовуючи від'ємний коефіцієнт селективності, завдання буде називатися типом "SU" (таблиця 2.1) тестових завдань з пороговий інтервал 75-79. У випадку відповідей з декількома варіантами з логікою "АБО", ті завдання, які знаходяться в інтервалі непридатності, негайно виключаються з тестової бази, тобто якщо всі елементи будь-якого стовпця матриці B, наприклад, перший стовпець, задовольняють бал:

$$B_j^k \leq 77\ (k = 1, \dots, n),$$

а елементи l-го стовпця задовольняють бал

$$B_j^k > 98\ (k = 1, \dots, n),$$

то j та і завдання буде виключено з тестової бази. Для решти завдань з множинним вибором з додаванням ваги за логікою "АБО", що потрапила в діапазон застосованості тесту, тобто якщо вони задовольняють нерівності:

$$77 \leq B_j^k \leq 98\ (k = 1, \dots, n),$$

коефіцієнт корекції вибірковості α вводиться для випадкового результату, де: $-4 \leq \alpha \leq +4$. Якщо вони задовольняють нерівності:

$$77 \leq B_j^k \leq 86\ (k = 1, \dots, n),$$

то, використовуючи коефіцієнти вибірковості, ці завдання будуть включені до типу тестових завдань “SOR” (таблиця 2.1) з пороговим інтервалом 80-84. Якщо вони задовольняють нерівність:

$$86 \leq B_j^k \leq 98\ (k = 1, \dots, n),$$

то, використовуючи коефіцієнт селективності, ці завдання будуть включені до типу тестових завдань "VOR" (таблиця 2.1) з пороговим інтервалом 90-94. Аналогічно, у разі відповіді на множинні варіанти з логікою "I", ті завдання, які потрапили в діапазон інтервалу непридатності, негайно виключено з тестової бази, тобто якщо всі елементи будь-якого стовпця матриці B, наприклад, 1-го стовпця, задовольняють оцінці:

$$B_j^k < 83\ (k = 1, \dots, n),$$

та елементам l-го стовпчика задовольняє умова:

$$B_j^k > 98\ (k = 1, \dots, n),$$

тоді j-та і l-та задачі будуть виключені з тестової бази. Для решти завдань з множинним вибором з додаванням ваги за логікою "I", які потрапили в діапазон застосованості тесту, тобто якщо вони задовольняють нерівності:

$$83 \leq B_j^k \leq 98\ (k = 1, \dots, n).$$

Вибірковість поправковий коефіцієнт α вводиться для випадкового результату, де: $-3 \leq \alpha \leq +3$. Якщо вони задовольняють нерівності

$$83 \leq B_j^k \leq 92\ (k = 1, \dots, n),$$

то, використовуючи коефіцієнт вибірковості, це завдання буде віднесено до типу "SAND" (таблиця 2.1) типу тестових завдань з пороговим інтервалом 85-89. Якщо вони задовольняють нерівність:

$$92 \leq B_j^k \leq 98\ (k = 1, \dots, n),$$

то, використовуючи коефіцієнт селективності, це завдання буде називатися типом тестових завдань "VAND" (таблиця 2.1) з пороговим інтервалом 95-100. Таким чином, параметри калібрування тестових завдань та рівнів знань тестувань визначені, як показано в таблиці (таблиця 2.1). Запропонований алгоритм вибору тестових завдань з одним та множинним вибором з різними блоками реакцій та шкалою рівня знань учнів передбачає можливість співвіднесення результатів різних тестів між собою. Таким чином, розроблено метод калібрування параметрів

тестових завдань, визначено шкали підготовленості випробувачів, а також встановлено рівневі класи для тестових завдань з одним та множинним вибором.

Таблиця 2.1. Калібрування результатів тестів

Complexity levels	Easy	Medium	Medium	Medium	Medium	High	High
Choice of test tasks	Single choice answer	Single choice answer	OR	AND	AND	OR	AND
Types of test tasks	LU	SU	SOR	SAND	SAND	VOR	VAND
Scores	50-74	75-79	80-84	85-89	85-89	90-94	95-100

2.3. Методи автоматизованої генерації тестових завдань

У роботі [10] Брусиловський описує життєвий цикл тестових завдань на освітньому порталі. В цей цикл входить декілька пунктів таких як: підготовлення, подача, оцінка. Дистанційне навчання є однією з провідних форм, яке включає в себе також комп'ютерне тестування. Багато дослідників [1], які працювали у цій сфері в основному були сконцентровані на надійності на валідності тестів. Проте, такий підхід значно ускладнює роботу експертам з предметної області, по якій складаються ручні тести. Отже, задача автоматизації тестування є актуальною.

Параметризовані тести. Одним з методів побудови тестових завдань, який є доволі перспективним – це параметризовані тести [11]. Цей метод дозволяє генерувати завдання відкритого типу. Людина, яка проходить тест, найчастіше повинна ввести деяке число, яке і буде відповіддю на поставлену задачу. В цьому методі можна створювати в одному завданні відразу декілька варіантів, тобто базується на принципі фасетності. Кожному, хто проходить тестування, програма

подає лише один елемент з фасету.

Суть цього методу полягає в тому, що тестове завдання базується на певному шаблоні, який відрізняється лише параметрами, проте дає змогу отримувати кожний раз різні тести. Параметризоване питання є шаблоном питання, який створюється автором. На момент подання завдання шаблон доповнюється параметром, значення якого генерується в заздалегідь зазначених межах [12-14]. Шаблон є зразком тексту, в якому деякі елементи можна змінювати відповідно до заданого алгоритму. На виході отримуємо якісні тести, проте вони дуже технічно-орієнтовані та важко перевіряти саме теоретичні знання. В роботі [14] у якості прикладу була наведена наступна задача: У Пети было два яблока, а у Васи три. Сколько яблок было у Пети и Васи?. Для того, щоб зробити з цієї задачі шаблон, необхідно замість певних чисел вставити параметри та алгоритми, які генерують значеннях цих самих параметрів. Тоді ця задача може бути записана як: У Петі було gen (x) яблука, а у Васі gen (y). Скільки яблук було у Петі і Васі?

Тут gen (x) і gen (y) – програма, що генерує значення для змінної x і в, відповідно. До кожного шаблону потрібно докласти програму рішення задачі за згенерованими параметрами. Тоді шаблон завдання буде виглядати наступним чином: правильна відповідь: (rez = solv (x, y)), де solv (x, y) – програма обчислення правильної відповіді [12].

Недоліком цього методу є трудомісткість формування набору шаблонів завдань. Перевагою цього методу є те, що для малої кількості шаблонів можна згенерувати достатньо велику кількість завдань. Так для наведеного прикладу, коли параметр x може прийняти 10 різних значень, а параметр y – 15, то ми отримаємо можливість згенерувати 150 варіантів завдань.

Генерація питань на основі алгоритмів – окремий випадок параметризованих задач. В основі питання людині, що навчається, пропонується деякий програмний код, який реалізує певний алгоритм. Тестований повинен визначити значення деякого параметра алгоритму, тим самим демонструючи своє розуміння мови програмування. Суть методу полягає в наступному: грунтуючись на умові виходу

з циклу, можна побудувати генератор питань. У роботах [14, 15] описується метод генерації запитань на основі алгоритмів. Розглянемо приклад знаходження суми натурального ряду:

- крок 1: i=0, S=0;
- крок 2: S=S+i, i=i+2;
- крок 3: якщо i < n, то перейти до кроку 2.

- питання 1. Яке значення прийме змінна S після завершення циклу, якщо n = генерувати() ;
- питання 2. Яке значення змінної n було встановлено, якщо по завершенню циклу значення S = генерувати();
- питання 3. Скільки ітерацій було виконано, якщо по завершенню циклу значення S = генерувати() і т.д. [14].

Генерація тестових завдань на основі дерев І / АБО Метод побудови алгоритмів генерації, запропонований Кручинін В.В., заснований на використанні дерев І / АБО, дозволяє описати будь-яку комбінаторну множину у вигляді дерева І / АБО [16]. Виходячи з цього в роботі [17] за комбінаторну множину було взято тестове завдання. Представимо опис примітивного текстового завдання з математики (рисунок 2.1).

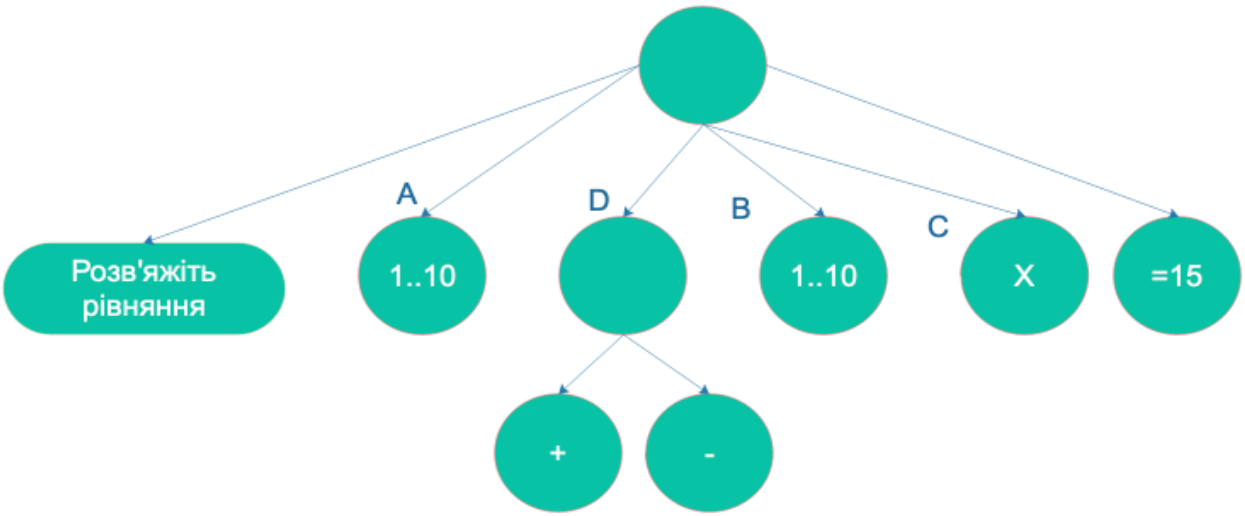

Рисунок 2.1. Опис текстового завдання у вигляді дерева І / АБО

Розглянемо гілку D. Гілка D має два АБО-варіантів представлення, перший варіант символ «+», а другий – «-» (рисунок 2.1). В даному випадку завдання буде мати 200 варіантів представлення. Однак, в разі вибору в галузі D знака «+», рішенням даного завдання буде формула:

C = (15-A) / B, при знаку «-» – C = (15-A) / - B.

Тобто рішення даного завдання також можна представити у вигляді дерева І / АБО (рисунок 2.2).

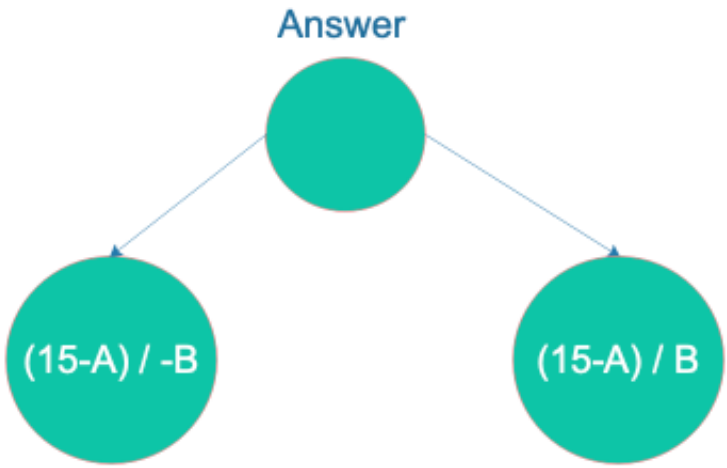

Рисунок 2.2. Гілка дерева І / АБО з варіантами відповіді

Семантичні мережі. Другим популярним методом тестування є використання семантичних мереж [18, 19]. Такий підхід базується на такому понятті, як тріада [18-20]. Наприклад, “клас” є частиною “Об’єктно-орієнтованого програмування”. Завдання формується таким чином, що, наприклад, частина тріади “є частиною” опускається, насправді вилучатись може одна з трьох частин тріади. Переваго такого методу є, те, що система може сама оперувати знаннями, недоліком є побудова семантичної мережі, яка б чітко описувала будь-яку конкретну область, яка досліджується. Також можуть виникати лінгвістичні неузгодженості, що призводять до неякісних тестів. Дуже часто генеруються

питання, які не мають цінності з точки зору педагогіки та викладання. Для класичних моделей штучного інтелекту притаманна проблема всеосвіченості, це часто спричиняє важкість на недоцільність побудови цієї самої моделі. Метою тестування все ж є навчати людину, а не систему [21].

Отже, необхідно зосередитись на формалізації навчальних текстів, з метою автоматизувати побудову тестових завдань, з такою задачею може впоратись ПТМ. Ця модель розробляється в багатьох галузях таких як: інтерпретація [22, 23], розділ лінгвістики – семантика, також дидактика, яка є частиною педагогіки.

Використання базової ПТМ. Основною структурною одиницею в ПТМ є поняття – деякий об’єкт з предметної області, який використовується у тестуванні. Для опису понять слугують спеціальні структурні елементи – тези. Вони є певною характеристикою поняття та містять інформацію про нього. Центральним носієм інформації є навчальні фрагменти, які дозволяють точно визначати, в якій області тестувальник орієнтується, а в якій ні. ПТМ дуже легко інтегрується з різними системами [24, 25], що дозволяю використовувати її як підсистему. Для автоматизації побудови завдань використовуються шаблони [2]. Якщо коротко, то в структурі шаблону є поля, які описують сутність, яка лежить в основі питання та варіантів відповідей – поняття або теза. Для забезпечення більшої якості завдань, кожне поняття та теза мають свою класифікацію [2]. В даній роботі також представлено алгоритм побудови тесту на базі ПТМ, який включає в себе наступні пункти:

1. Вибір контрольної ПТ-пари.
2. Пошук допустимих шаблонів завдань.
3. Вибір шаблону завдання.
4. Пошук альтернативних варіантів відповідей.
5. Візуалізація тестового завдання.

Перевірка тестових завдань відбувається наступним чином, порівнюються істинні варіанти відповідей та студента. В разі провалу якогось завдання, система дає рекомендації щодо повторення певних тем, а також сигналізує про необхідність

корекції сценарію викладання навчального матеріалу [26]. В роботі [2] представлено лише один тип завдань, проте ПТМ може бути модифікована для можливості побудови інших типів [26-28]. Система була протестована на порталі, який в переважній більшості орієнтується на дистанційну самопідготовку [29].

Модифікація понятійно-тезисної моделі з використанням ключових слів. В роботі [4] пропонується модифікувати ПТМ шляхом додавання до структурних елементів ще однієї складової – ключових слів.

Ключові слова – це слова або словосполучення, які використовуються для вираження деякого контекстного аспекту змісту тези про поняття. Вони несуть істотне смислове навантаження і так чи інакше характеризують поняття, про яке йдеться в тезі [4].

Під час семантичного розбору тексту викладач виділяє поняття, додає до них тези і виділяє ключові слова.

Наведемо приклад навчального матеріалу і виконаємо його семантичний розбір, згідно удосконаленої ПТМ і її базових елементів. В якості навчального матеріалу було взято фрагмент курсу лекцій «Програмування на мові Сі ++» [4]:

В С ++ з'являється видозмінена форма покажчика – посилання. Вона може розглядатися як покажчик, який є ще одним ім'ям або псевдонімом змінної.

Посилання можна ініціалізувати тільки один раз. Тут доречно порівняти покажчик і посилання. Покажчик – це змінна, яка може приймати значення адреси іншої змінної певного типу. Такі адреси можна привласнювати кілька разів. Аби це були адреси заданого типу змінних.

Посилання підвищують ефективність програми, особливо при передачі даних при виклику функції. У порівнянні з передачею даних за значенням не потрібно копіювати дані, що передаються в стек. Економляться час і пам'ять.

Виділимо наступні структурні елементи:

Поняття: Посилання

Набір тез:

- є видозміненою формою покажчика;

- може розглядатися як покажчик, який є ще одним ім'ям або псевдонімом змінної;
- можна форматувати тільки один раз;
- підвищують ефективність програми, особливо при передачі даних при виклику функції.

Ключові слова: покажчик (2), змінна (1), програма (1), функція (1).

Аналізуючи ключові слова в тезах, доцільно внести в БЗ таке поняття як «Покажчик», адже воно зустрічається досить часто в тексті тез і дидактично передує поняттю «посилання», а також є ключовим на думку автора тесту.

Поняття: Покажчик.

Набір тез:

- це змінна, яка може приймати значення адреси іншої змінної певного типу;
- адреси можна привласнювати кілька разів.

Ключові слова: адреса (2), змінна (1).

Даний підхід є перспективним з точки зору розширення варіантів тестових завдань. У той же час слід зазначити, що ПТМ передбачає автоматичний синтаксичний аналіз тексту тез на предмет входження в нього інших понять [2], що фактично замінює запропоновану в роботі [4] ручну працю.

Модифікація понятійно-тезисної моделі з використанням системи семантичних класів.

В роботі [5] для повного опису характеристик предметної області на базі ПТМ пропонується сформувати такі когнітивні абстрактні класи: визначення, проблеми, методи, ефективність методів, приклади реалізації методів. У даній роботі представлена система семантичних класів(рисунок 2.3)

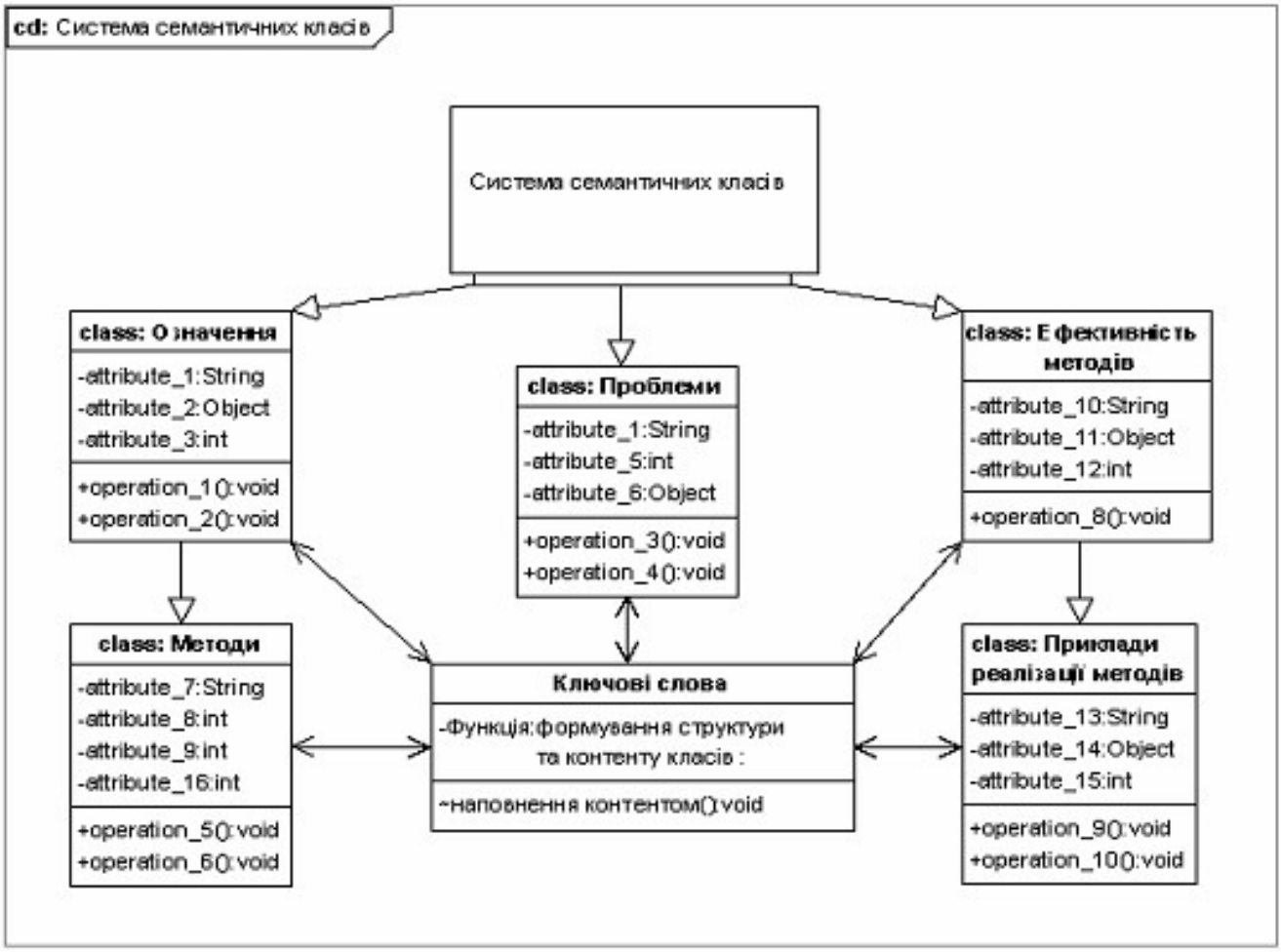

Рисунок 2.3. Система семантичних класів

Суть методу полягає в тому, що для генерації тесту деяке твердження розбивається на основну та альтернативну частини, які можуть містити одну або кілька компонент. Альтернативна частина тесту поповнюється аналогічними за лінгвістичним змістом, синтаксично узгодженими частинами інших тверджень. Синтаксичне узгодження забезпечується, коли зв'язки компонент тверджень однорідні за типами і числами.

На основі бази тверджень може бути сформований набір тестових завдань із заданими параметрами у вигляді динамічної структури.

У зазначеній роботі [5] запропоновано розв'язання мовної неузгодженості, характерною для деяких тестових завдань, що генеруються на основі ПТМ. Водночас даний підхід передбачає значні трудові витрати на формалізацію додаткових характеристик і сутностей, що доповнюють базову версію ПТМ.

2.4. Висновки

У даному підрозділі було розглянуто методи автоматизованої побудови тестових завдань на базі ПТМ та її модифікацій: з використанням ключових слів, системи семантичних класів. Також описано використання семантичних мереж та параметризованих тестів. Розглянуто переваги та недоліки кожного підходу. Представлено один з варіантів калібрування тестових завдань та розглянуто основні концепції, які необхідні для забезпечення якості, валідності тестів.

3. РОЗРОБКА СИСТЕМИ АВТОМАТИЗОВАНОГО КОНТРОЛЮ ЗНАНЬ

Дана система була розроблена на базі ПТМ з модифікацією, яка полягаю в доповнені структури шаблону та введенні керованої складності. Перевагами ПТМ є:

- невеликі часові витрати на генерацію тестових завдань;
- процес формування бази знань легкий для сприйняття, його може здійснювати викладач відповідної дисципліни;
- повна автоматизація побудови безпосередньо тестових завдань;
- достатня якість, хоча і поступається тестам створених вручну;
- легке переструктурування, оскільки зв'язок семантичних даних та навчального матеріалу закладено в самій моделі;
- висока точність визначення ділянки навчального матеріалу необхідного для повторення у разі провалу тестового завдання.

Проте є ряд недоліків існуючих систем на базі ПТМ (рисунок 3.1), що зумовлює необхідність модернізації ПТМ.

Система на базі ПТМ	Дослідник	Недолік
Використання семантичних класів	Мельник А.М	Ускладнює формалізацію дидактичного тексту у порівнянні з базовою моделлю ПТМ
Використання ключових слів у тезах	Петрова Л. Г.	Погана якість завдань закритого типу
Базова ПТМ	Титенко С.В	Відсутня керована складність

Рисунок 3.1. Недоліки існуючих систем на базі ПТМ

3.1. Понятійно-тезисна модель формалізації дидактичного тексту

До ПТ-елементів належать поняття і тези [2]. Поняття описує деяку сутність з предметної області. Множина понять:

$$C = \{c_1, c_2, \dots, c_n\}.$$

Тези є природомовним вираженням знань з предметної області у формі фрагментів навчального тексту та медіа-вмісту. Теза – це деяка відомість або твердження про поняття. Множина тез:

$$T = \{t_1, t_2, \dots, t_n\}.$$

Кожна теза стосується одного поняття, і цей зв'язок задається відношенням:

$$CT: T \rightarrow C.$$

У свою чергу кожне поняття може мати довільну кількість тез, що описується відношенням:

$$TC: C \rightarrow 2^T.$$

Також є такі тези, які пояснюють інші тези, позначимо їх:

$$TExplanation = \{te_1, te_2, \dots, te_n\}.$$

Тоді зв’язок між головною тезою та її поясненням відображається так:

$$TExplanationT: TExplanation \rightarrow T.$$

В роботі [8] декомпозиція понять реалізується за допомогою відношень part-of, is-a, instance-of, що описуються відображеннями:

$$PartOf: C \rightarrow C, IsA: C \rightarrow C, InstanceOf: C \rightarrow C.$$

Реалізація даних відношень в ПТМ відбувається на базі спеціальних тез-відношень, які є підмножиною тез в системі:

$$Trel \subset T.$$

Тези-відношення посилаються на поняття, з яким у поняття-власника тези встановлюється відношення:

$$RelCT: Trel \rightarrow C.$$

При цьому тип відношення вказується через клас тези:

$$TRealClasses \subset TClassses,$$

$$TRelClasses = \{tRelPartOf, tRelIsA, tRelInstanceOf\}.$$

Окрім цих трьох відношень є такі поняття, які є аспектами інших і не є самостійними. Такі поняття [8] позначаються наступним чином:

$$Aspects = \{c \in C: isAspect(c) = true\},$$

$$isAspect: C \rightarrow is\_aspect,$$

де is_aspect={true,false}.

Поняття, яке не має самостійного значення і використовується для опису деякої особливості головного поняття, буде виступати аспектом головного поняття:

$$AspectOf: Aspects \rightarrow C.$$

Центральним носієм знань вважається навчальний матеріал. позначимо множину фрагментів або сторінок навчального контенту такою множиною:

$$V = \{v_1, \dots, v_n\}.$$

Також зв'язок понять та тез з навчальним матеріалом можна відобразити наступною схемою (рисунок 3.2):

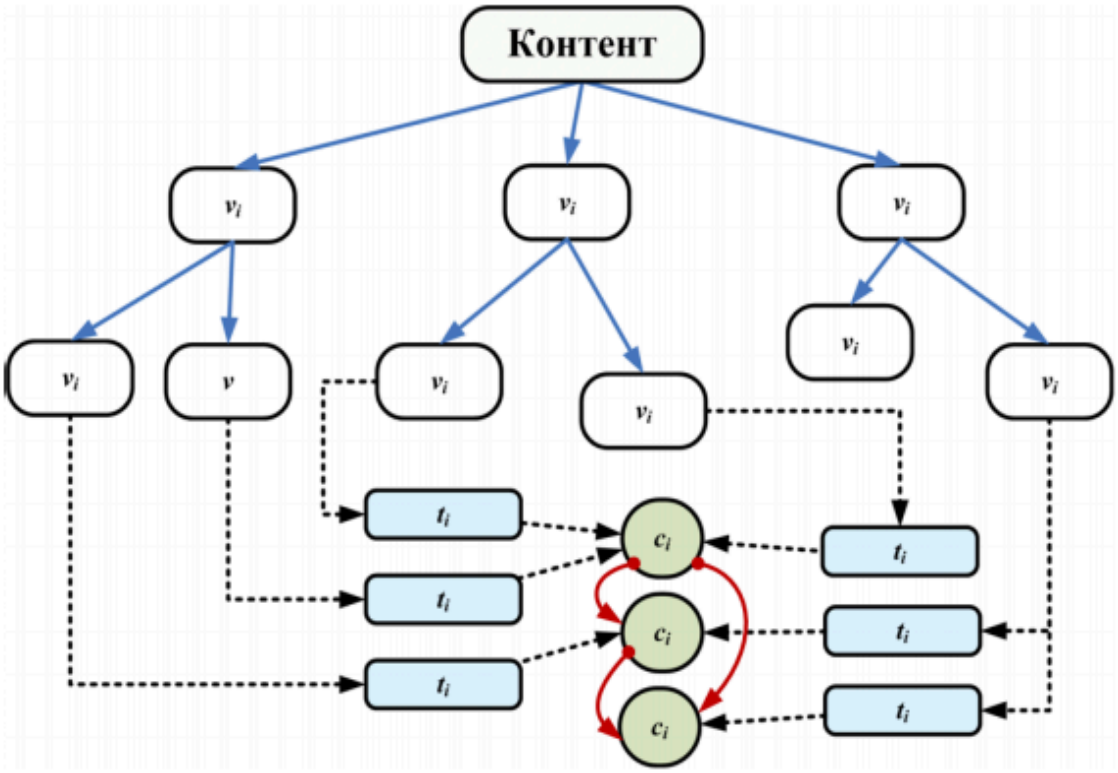

Рисунок 3.2. Зв'язок понять та тез з навчальним матеріалом

Навчальні фрагменти беруться з основного порталу, приклад одного з них (рисунок 3.3):

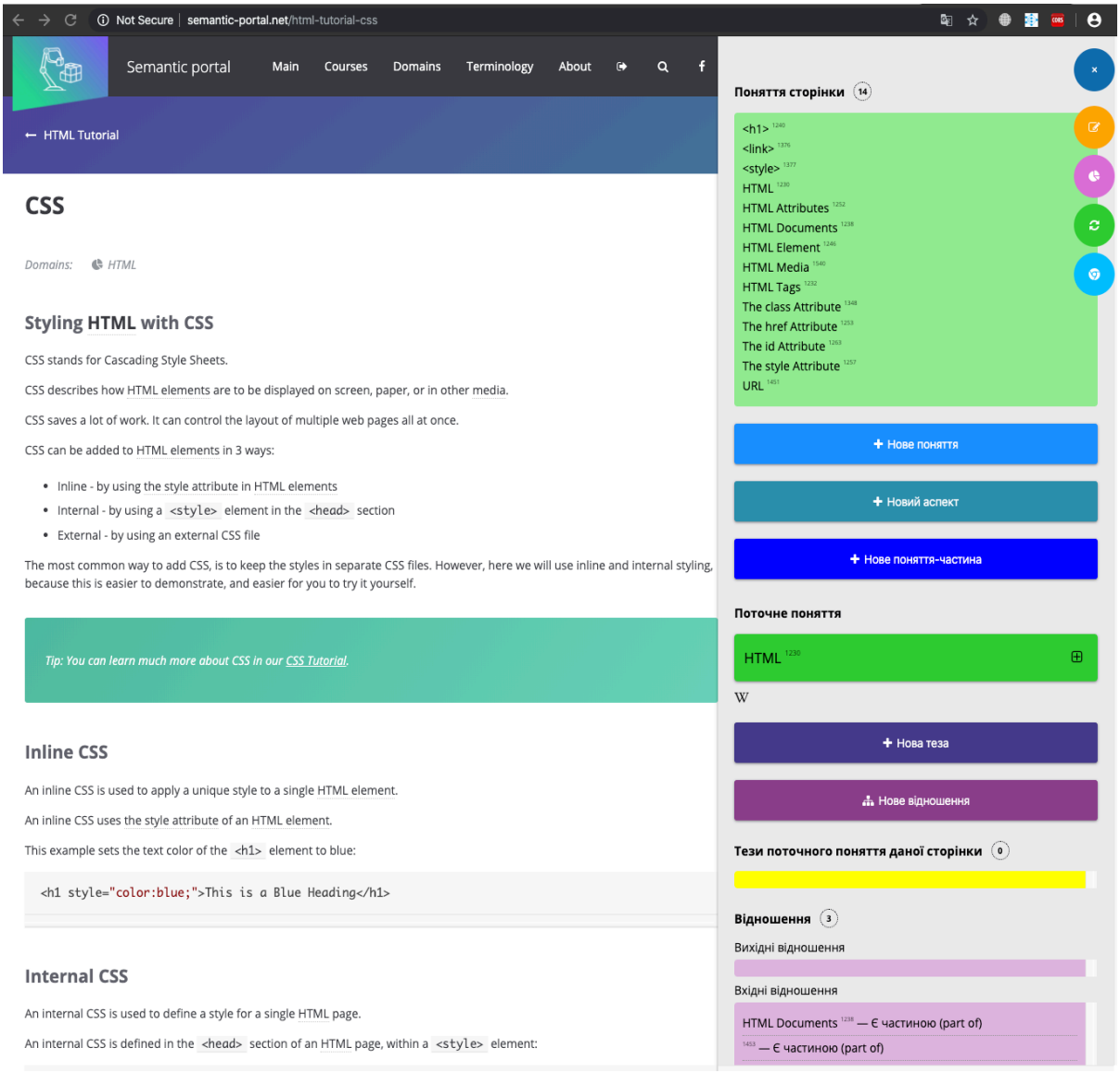

Рисунок 3.3. Приклад навчального фрагменту

3.2. Шаблони та типи тестових завдань

Для автоматизації створення тестових завдань будемо використовувати шаблони, які опишемо множиною:

$$TTempl = \{tt_1, tt_2, \dots, tt_n\}.$$

У раніше запропонованій реалізації тестування на базі ПТМ шаблони поділялись на два основних типи: 1) в основі питання — теза, в основі варіантів відповідей — поняття; 2) в основі питання — поняття, варіанти відповідей є тезами [2]. Шаблони [2] описують сутність запитального елементу (поняття чи теза), допустимі класи для запитального елементу та елементів-відповідей. Пропонується диверсифікувати використання подібної структури шаблону для генерації різних типів тестових завдань.

Усі тестові завдання належали до такого типу, що передбачає вибір однієї правильної відповіді, чого недостатньо для покриття шкали Блума (рисунок 3.4):

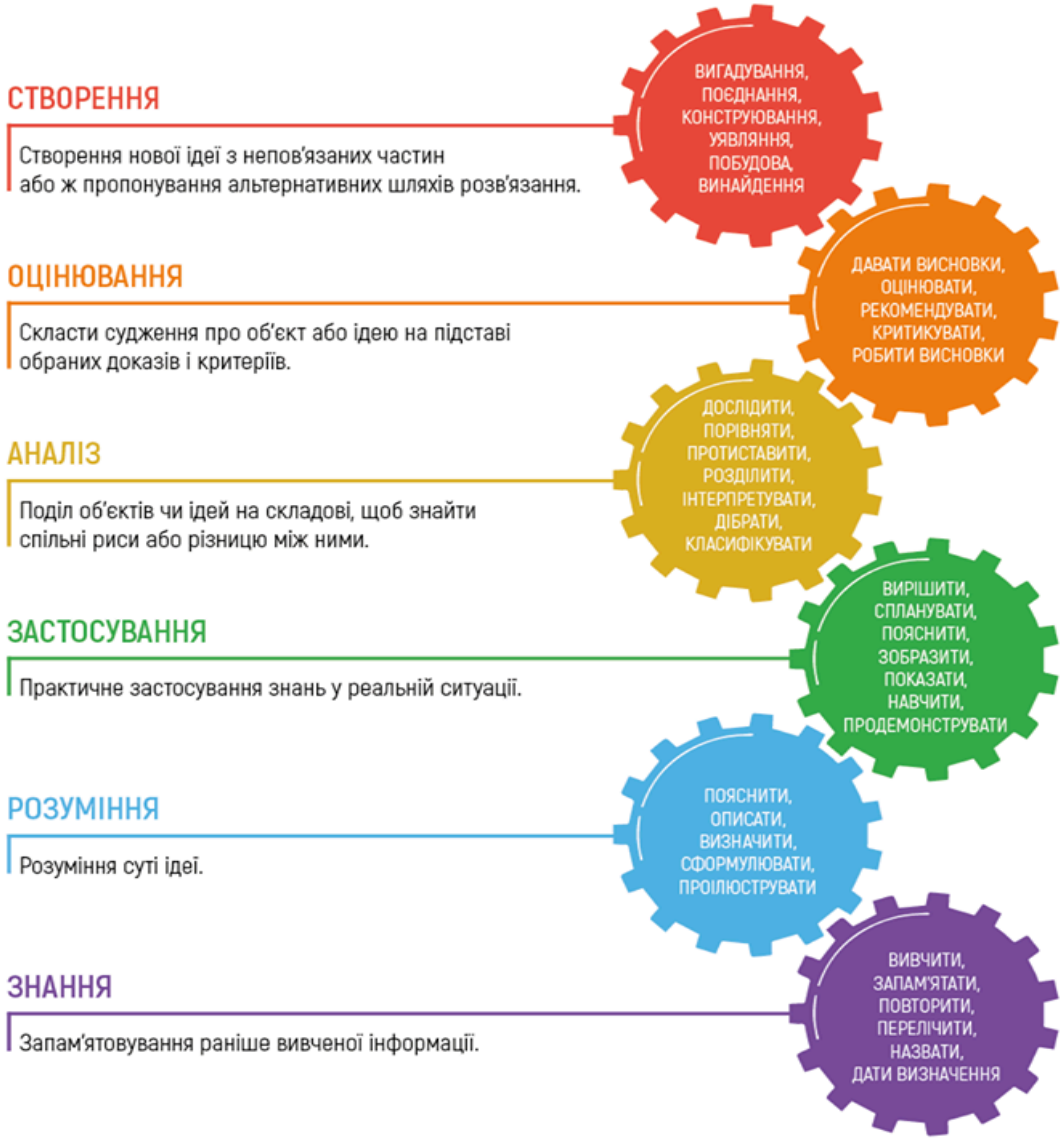

Рисунок 3.4. Шкала Блума

Множину завдань позначимо наступним чином: $Task = \{Task_i\}$, а множину їх типів: $TType = \{CO, CS, WA, MI, GF, RW\}$.Дамо опис для кожного типу завдання:

- *CO* — завдання множинного вибору з вимогою вибору однієї вірної відповіді;
- *CS* — завдання множинного вибору з вимогою вибору декількох вірних відповідей;
- *WA* — завдання відкритого типу, в якому потрібно вписати поняття-відповідь на тезу-визначення, що зазначена в питанні;
- *MI* — завдання на зіставлення наборів відповідних ПТ-елементів;
- *GF* — завдання відкритого типу, де необхідно заповнити прогалину в твердженні;
- *RW* — завдання множинного вибору з вимогою вказати некоректні варіанти відповідей.

Множину типів завдань, наявних у системі, та їх структуру можна зобразити схематично (рисунок 3.5)

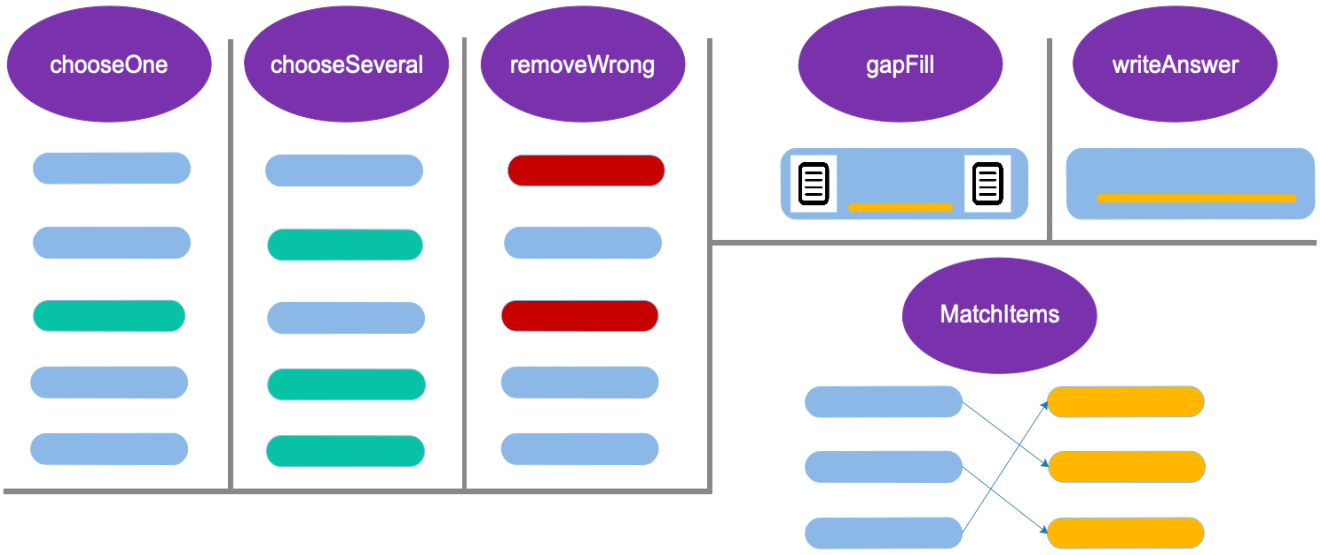

Рисунок 3.5. Структура тестових завдань

Типи завдань, доступні для генерації на базі даного шаблону, опишемо наступним зв’язком:

$$TTemplTypes: TTempl \rightarrow 2^{TType}.$$

Таким чином, атрибути шаблону tt_i:

- *TQEntity(tt_i)* — ПТ-елемент, який лежить в основі питання;
- *TQStr(tt_i)* — шаблонний текст, який відповідає запитанню тестового завдання;
- *TQClasses(tt_i)* — допустимі класи ПТ-елементу, що лежать в основі питання, який можна використовувати для шаблону;
- *TQNotClasses(tt_i)* — недопустимі класи ПТ-елементу, що лежать в основі питання;
- *TAClasses(tt_i)* — допустимі класи ПТ-елементу, що лежать в основі відповіді;
- *TANotClasses(tt_i)* — класи ПТ-елементу, що лежить в основі відповіді, який не можна використовувати для шаблону;
- *TTemplTypes(tt_i)* — нове поле в структурі шаблону, в якому вказується множина типів завдань, які можуть бути згенеровані на базі даного шаблону.

В кожному шаблоні (рисунок 3.6) в полі *TQStr* також зберігається маркер для заповнення ПТ-елементом. Якщо такого маркеру нема, то ПТ-елемент вставляється в кінець рядка *TQStr(tt_i)*. Зазначимо, що для завдань *{MI, GF, RW}* рядок запитання не буде містити ПТ-елемент.

Рисунок 3.6. Структура шаблону

3.3. Генерація тестового завдання з диференціацією за рівнем складності

Опишемо різні когнітивні цілі за модифікованою шкалою Блума [7] множиною:

$$KT = \{Remember, Understand, Apply, Analyze, Evaluate, Create\}.$$

Рівень складності тестів поділимо на низький, середній та складний, позначимо їх множиною:

$$Complexity = \{Low, Middle, Hard\}.$$

Такий поділ дозволить розподілити завдання за рівнем складності, а потім провести диференційоване тестування. Позначимо зв'язок рівня складності тесту в залежності від типу знань за шкалою Блума [7]:

$$ComplexityKT: Complexity \rightarrow 2^{KT}.$$

Також цю залежність можна описати у вигляді системи:

$$ComplexityKT(cl) = \begin{cases} \{Remember, Understand\}, cl = Low \\ \{Remember, Understand, Apply\}, cl = Middle \\ \{Remember, Understand, Apply, Analyze\}, cl = Hard \end{cases}$$

Типи завдань, які генеруються, в залежності від складності тесту описуються наступним чином:

$$TypesByComplexity(cl) = \begin{cases} \{CO, CS, WA\}, cl = Low \\ \{CO, CS, WA, MI\}, cl = Midlle \\ \{CO, CS, WA, MI, GF, RW\}, cl = Hard \end{cases}$$

Формулу зазначену вище можна представити графічно (рисунок 3.7), зазначимо що на цьому рисунку зображено лише ті типи знань за шкалою Блума, на основі яких реалізовані тестові завдання. Типи, що перевіряють знання створення, оцінювання, в даній модифікації алгоритму поки що не вдалося впровадити.

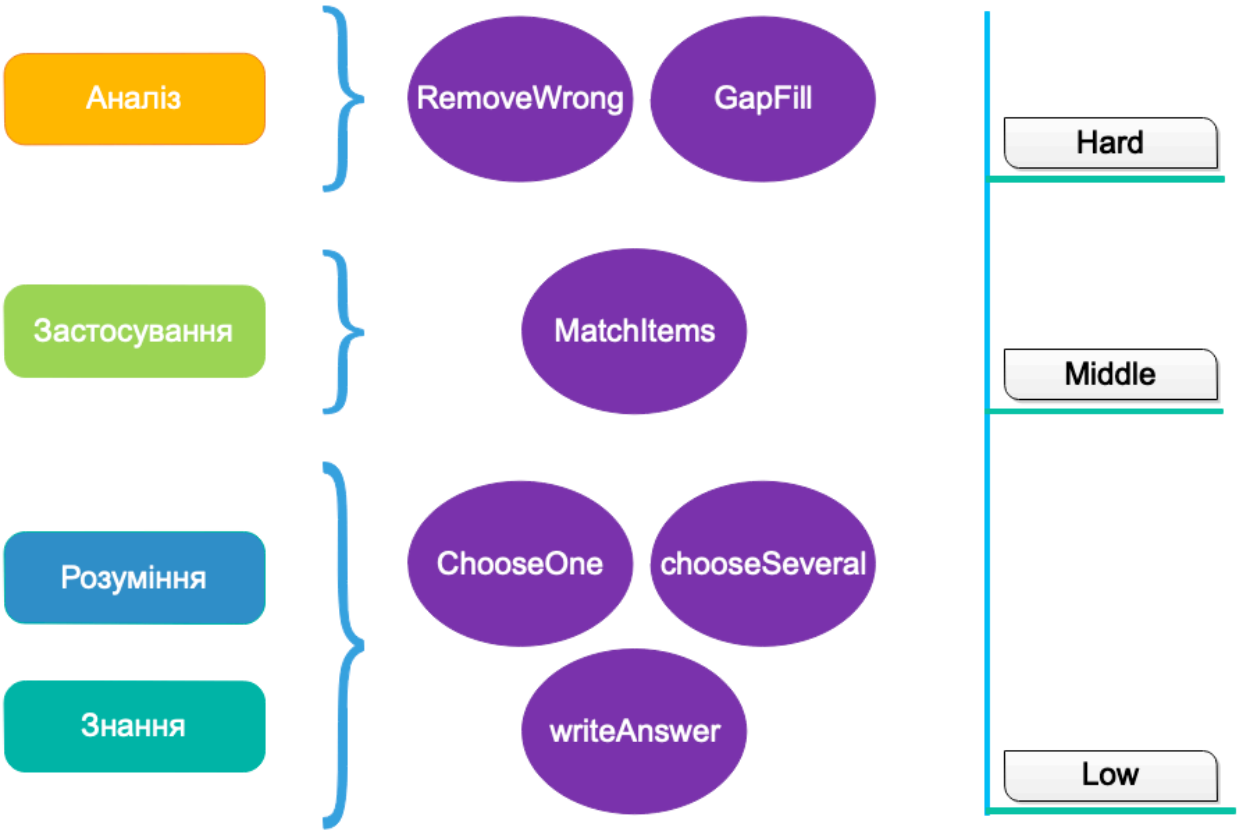

Рисунок 3.7. Відповідність типів тестових завдань рівню складності та знанням, що вони перевіряють

Для забезпечення валідності тестових завдань, було виявлено, що аспекти понять можна використовувати, тільки коли вони або їх тези виступають запитальним ПТ-елементом. В першому випадку тези, що відносяться до інших аспектів-відповідей або відносяться до головного поняття аспекту-питання, не мають бути дистракторами. В другому, поняття-дистрактори не мають бути аспектами, а також не мають бути головними поняттям аспекту, до якого ця теза-запитання відноситься.

Модифікований алгоритм побудови тестового завдання містить наступні етапи:

1. **Вибір контрольної області контенту**. Обрана сукупність елементів контенту з множини контенту V формують контрольну область даного тесту

$$test: SuffControlA \subseteq V.$$

Тут же здійснюється перевірка достатності даних для побудови тесту:

$$(|\{t: t \in T \wedge VT(t) \in SuffControlA\}| \geq rta) \rightarrow content\_valid(SuffControlA),$$

де rta —мінімальна необхідна кількість тез для забезпечення генерації тестових завдань, встановлюється евристично.

2. **Вибір складності тесту** $cl_{test} \in Complexity$.

3. **Генерація необхідної кількості тестових завдань відповідно до складності тесту**. З множини допустимих типів завдань для даного рівня складності, формуємо необхідну кількість тестових завдань в циклі.

Генерація тестового завдання.

3.1. **Вибір одного з допустимих типів тестових завдань** *tType:*

$$tType \in TypesByComplexity(cl_{test}).$$

3.2.**Вибір шаблону, що підходить для генерації завдання з типом** *tType*. Шукаємо шаблон tt_i, що задовольняє умові $tType \in TTemplTypes(tt_i)$.

3.3.**Пошук допустимих ПТ-елементів для даного шаблону** tt_i**, що знаходяться в контрольній області** *SuffControlA*. Множину понять та тез області *SuffControlA*будемо визначати так:

$$SCControlA = \{c: VC(c) \cap SuffControlA \neq \emptyset\},$$

$$STControlA = \{t: VT(t) \in SuffControlA\}.$$

Для множини завдань всіх типів, крім *MI*, оператор вибору допустимих запитальних ПТ-елементів для даного шаблону tt_i виглядає так:

$$STQEntities(tt_i) = \begin{cases} \left\{ \begin{array}{l} t: t \in T \wedge t \in STCControlA \wedge \\ \left(\begin{array}{l} TClass(t) \in TQClasses(tt_i) \\ \vee\, TQClasses(tt_i) = 0 \end{array} \right) \wedge \\ TClass(t) \notin TQNotClasses(tt_i) \end{array} \right\}, TQEntity(tt_i) = Thesis \\ \left\{ \begin{array}{l} c: c \in C \wedge t \in SCControlA \wedge \\ \left(\begin{array}{l} CClass(t) \in TQClasses(tt_i) \\ \vee\, TQClasses(tt_i) = 0 \end{array} \right) \wedge \\ CClass(c) \notin TQNotClasses(tt_i) \end{array} \right\}, TQEntity(tt_i) = Concept \end{cases}$$

Якщо множина *STQEntities(tt_i)* непуста, з множини *STQEntities(tt_i)* обираємо випадковий елемент, позначимо його *STQEntity*.

Для типу завдання на зіставлення *MI TQEntity(tt_i)* може бути лише поняттям, і в межах одного завдання необхідно отримати набір різних понять, для яких далі буде обрано відповідний набір тез для співставлення аналогічно наступним крокам алгоритму.

Якщо множина *STQEntities(tt_i)* виявилась порожньою, то tt_i шаблон виявився недопустимим або в навчальному фрагменті, що відноситься до множини *SuffControlA,* ПТ-елементи не підійшли під умови відбору. У такому випадку здійснюється перехід до п.3.2. Якщо не вдалося знайти жодного шаблону для завдання *tType,* то спочатку відбувається перехід до пункту п.3.2 зі зміною CTEntity, якщо і в такому разі не знайдено жодного шаблону здійснюється перехід до п.3.1. для вибору іншого типу завдання.

3.4. **Пошук правильних варіантів відповідей**. ПТ-сутність для варіантів відповідей умовно позначимо як *Entity*:

$$Entity = \left(ent: ent \in \left(CTEntity \backslash TQEntity(tt_i)\right)\right).$$

Множина відповідних ПТ-елементів :

$$E = \begin{cases} C, Entity = Concept \\ T, Entity = Thesis \end{cases}$$

Умовно позначимо відповідні множини класів ПТ-елементів як *EClass*.

Правильність варіантів забезпечується їх відношенням до запитального ПТ-елемента. Предикат *correcponds(e,a)* формалізує твердження про те, що аргумент-теза дійсно відноситься до аргументу-поняття:

$$(e,a): \begin{pmatrix} (e \in C \wedge a \in T \wedge CT(a) = e) \vee \\ (e \in T \wedge a \in C \wedge CT(e) = a) \end{pmatrix} \rightarrow corresponds(e,a).$$

Оператор вибору множини можливих вірних варіантів відповідей для всіх типів завдань, крім *tType=GF*:

$$RTAEntities(tt_i) = \left\{ \begin{matrix} e: e \in E \ \wedge corresponds(e, STQEntity) \wedge \\ \left(\begin{matrix} EClass(e) \in TAClasses(tt_i) \vee \\ TAClasses(tt_i) = \emptyset \end{matrix} \right) \wedge \\ EClass(e) \notin TANotClasses(tt_i) \end{matrix} \right\}.$$

Для типу завдання на заповнення прогалини в твердженні, де $tType = GF, TQEntity(tt_i) = Concept$, узагальнений оператор пошуку допустимих тез-тверджень:

$$RTAEntities(tt_i) = \{e: e \in E \ \wedge corresponds(e, STQEntity) \wedge Entity = Thesis \wedge isSubstring(STQEntity, e) \wedge (EClass(e) \in TAClasses(tt_i) \vee TAClasses(tt_i) = \emptyset) \wedge EClass(e) \notin TANotClasses(tt_i)\}.$$

3.5. **Пошук дистракторів**. Дистрактори шукаються для типів завдань *CO,CS,RW*. Оператор пошуку враховуючи, що тези, які відносяться до аспектів понять використовувати не можна, а також випадків, коли SQTEntity є аспектом або тезою, що відноситься до аспекту, оператор пошуку, має вигляд:

$$WTAEntities(tt_i) = \{e: e \in E \neg corresponds(e, STQEntity) \wedge \left(corresponds(e, C(c_i)) \wedge C(c_i) \notin Aspects\right) \wedge (EClass(e) \in TAClasses(tt_i) \vee TAClasses(tt_i) = \emptyset \wedge EClass(e) \notin TANotClasses(tt_i)\}.$$

Для випадку, коли STQEntity є аспектом додається ще одна умова: тези, що відносяться до головних понять цього аспекту не повинні попадати в тестове завдання. Позначимо головні поняття аспекту STQEntity як MCofSTQEntity , тоді оператор пошуку дистракторів виглядає так :

$$WTAEntities(tt_i) = \{e: e \in E \neg corresponds(e, STQEntity) \wedge (corresponds(C(c_i), e) \wedge C(c_i) \notin Aspects \wedge C(C_i) \notin MCofSTQEntity) \wedge (EClass(e) \in TAClasses(tt_i) \vee TAClasses(tt_i) = \emptyset \wedge EClass(e) \notin TANotClasses(tt_i)\}.$$

Для випадку, коли SQTEntity є тезою, що відноситься до аспекту поняття, позначимо головні поняття цього аспекту як MCofRTAEntities(tt_i):

$$WTAEntities(tt_i) = \{e: e \in E \neg corresponds(STQEntity, e) \wedge (corresponds(T(t_i), e) \wedge e \notin Aspects \wedge e \notin MCofRTAEntities(tt_i)) \wedge (EClass(e) \in TAClasses(tt_i) \vee TAClasses(tt_i) = \emptyset \wedge EClass(e) \notin TANotClasses(tt_i)\}.$$

Тестове завдання побудоване. Перехід до побудови наступного тестового завдання (п.3). Якщо необхідна кількість завдань тесту побудована, побудову тесту завершено. Алгоритм побудови тестового завдання у нотації UML діаграма діяльності (рисунок 3.8)

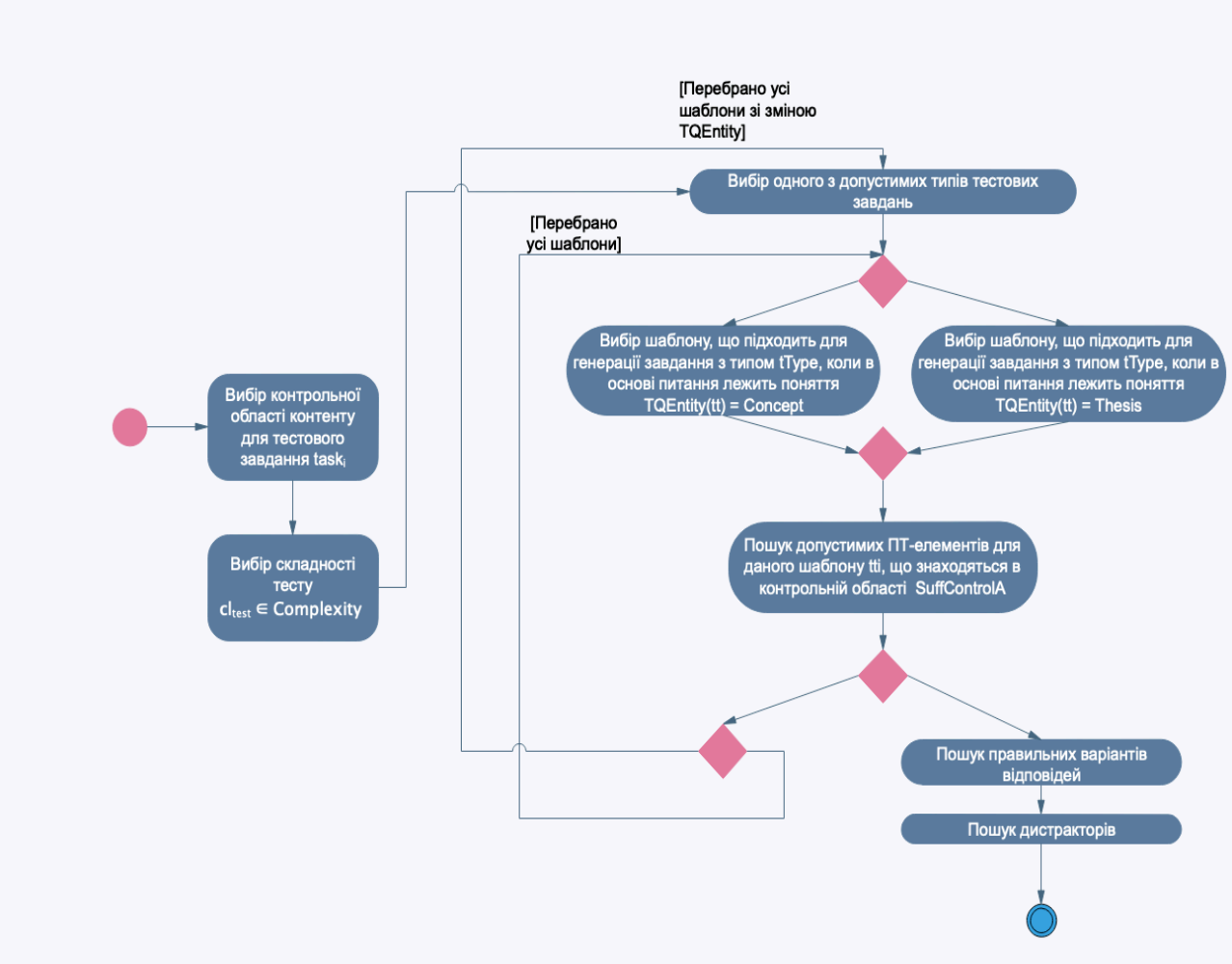

Рисунок 3.8. Алгоритм побудови тестового завдання

Приклади завдань які генеруються зображені на рисунках 3.9 – 3.14

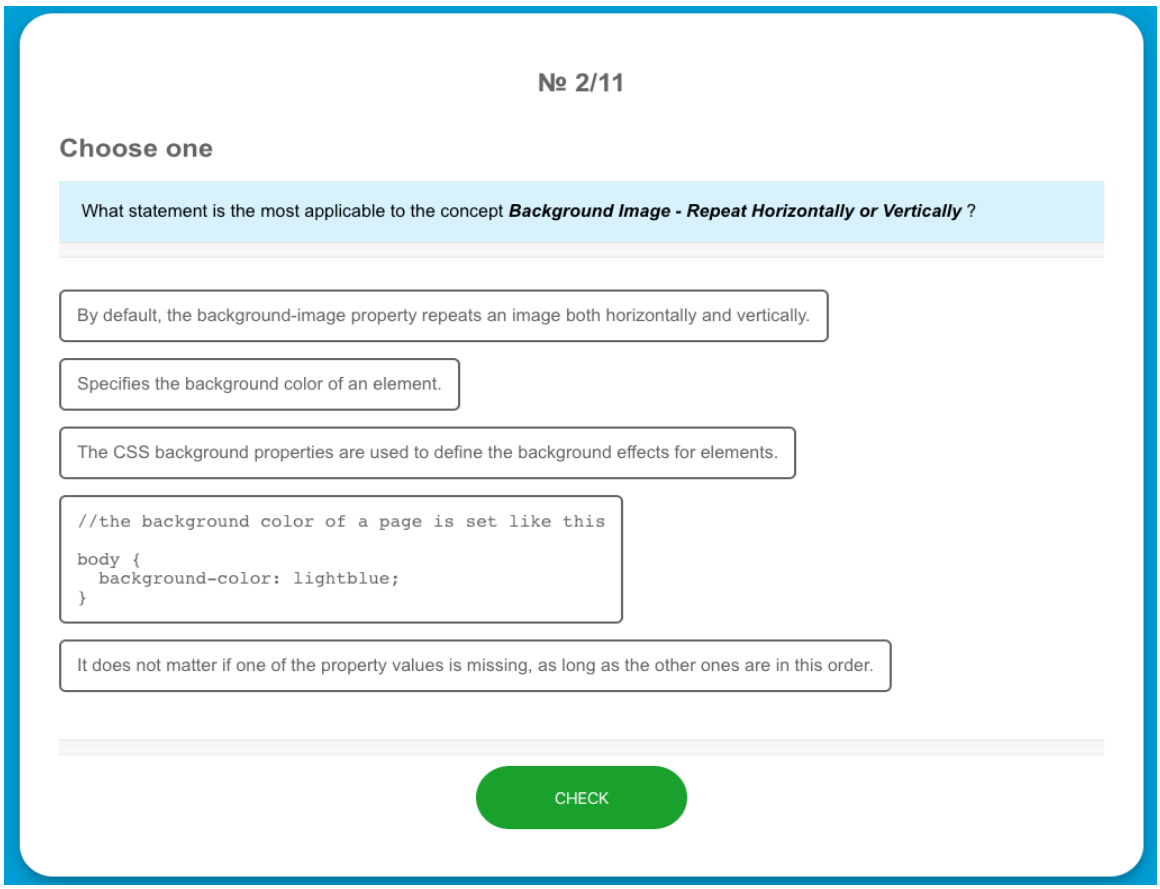

Рисунок 3.9. Приклад завдання на вибір однієї правильної відповіді

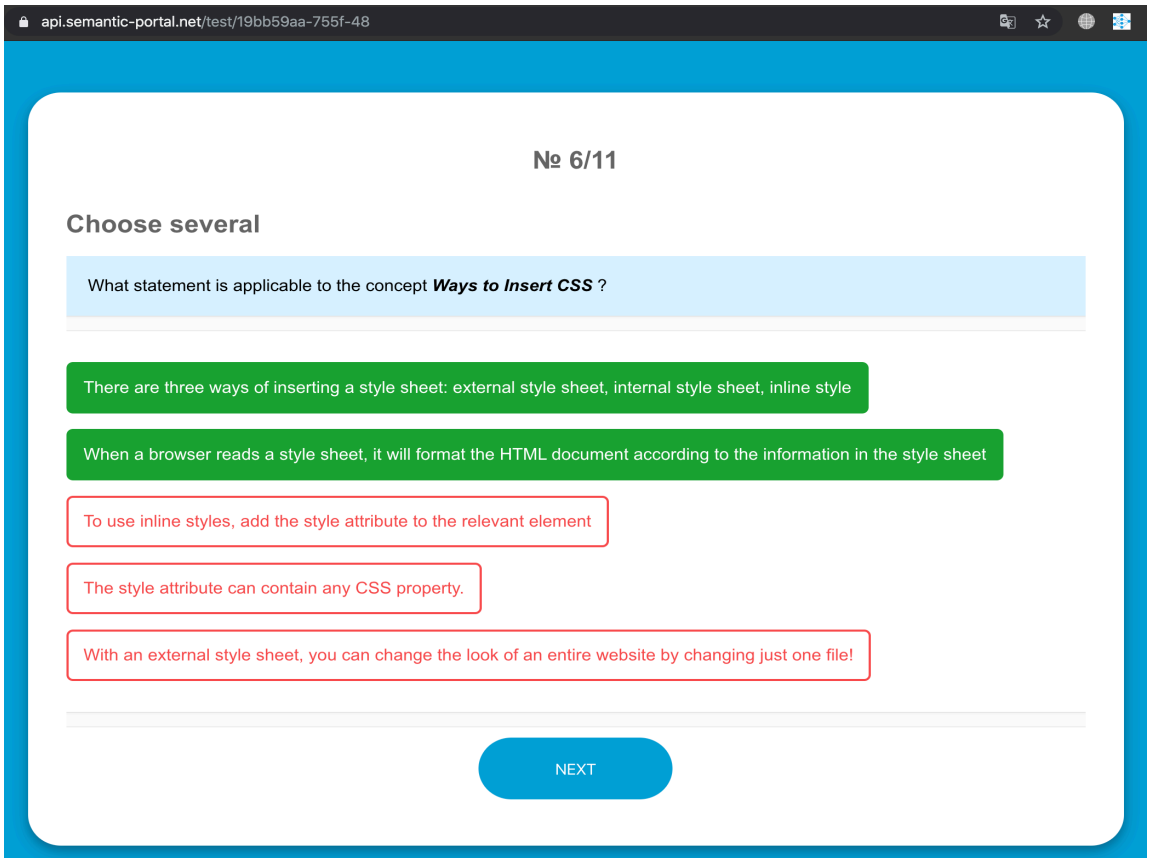

Рисунок 3.10. Приклад завдання на вибір декількох правильних відповідей

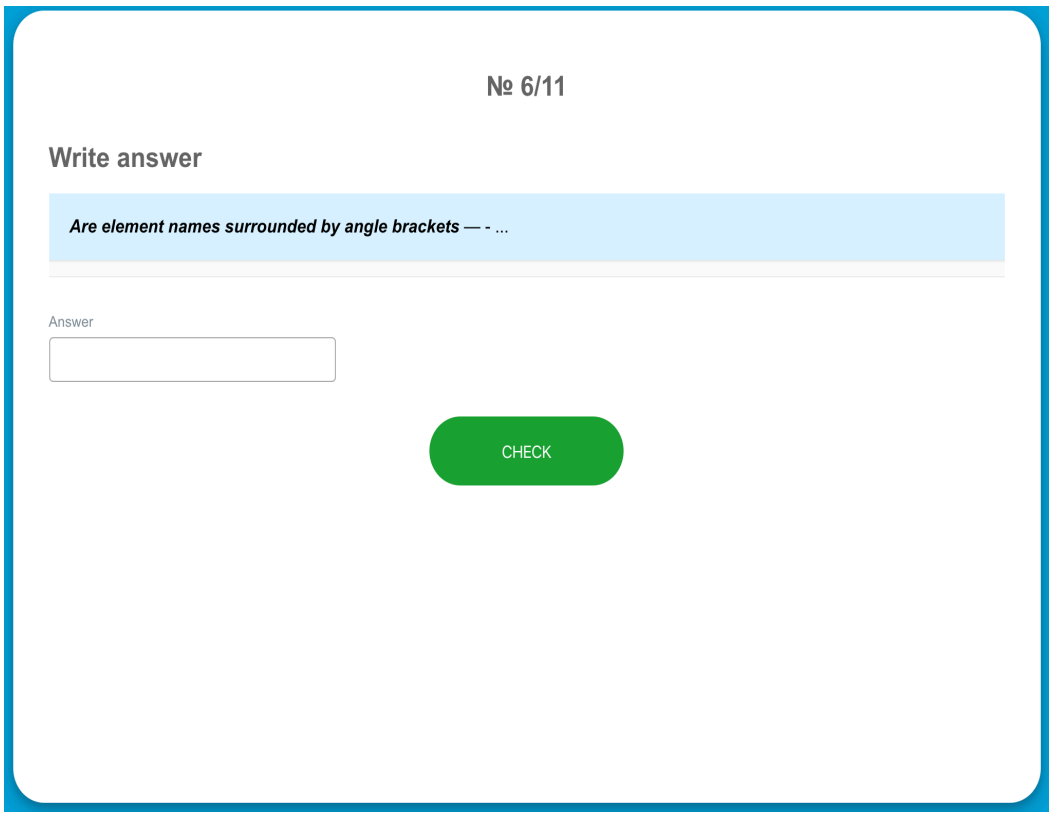

Рисунок 3.11. Приклад завдання відкритого типу

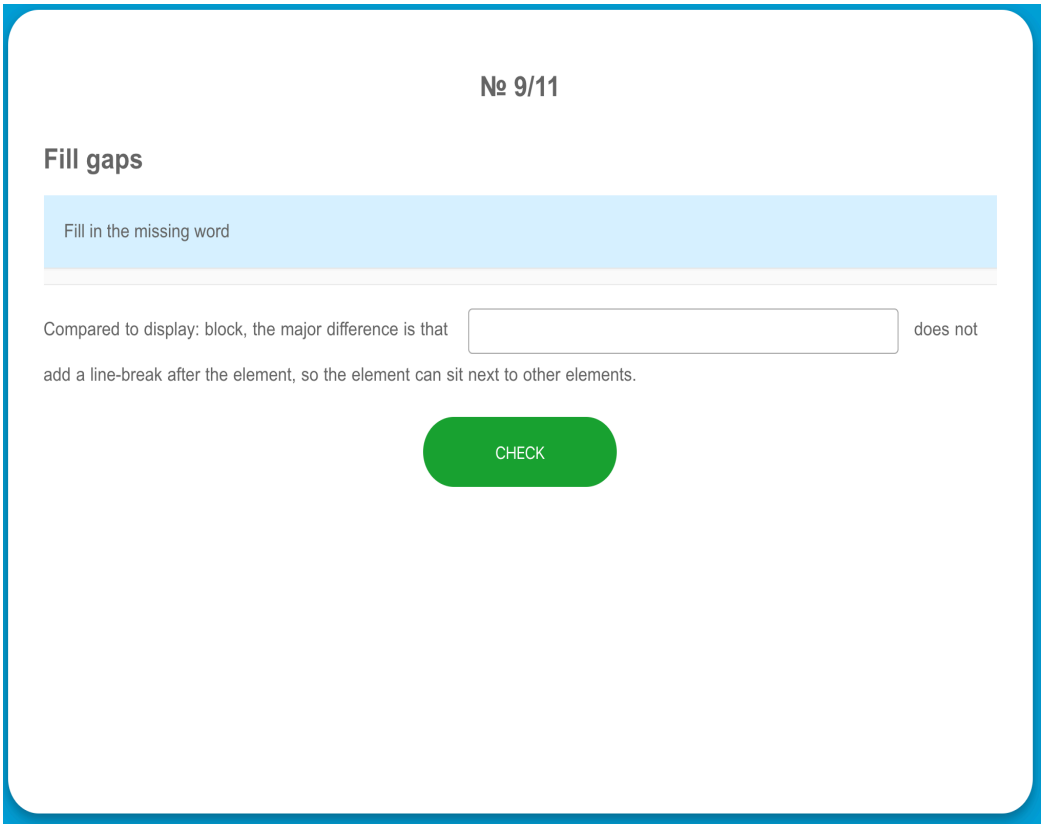

Рисунок 3.12. Приклад завдання на заповнення прогалини в твердженні

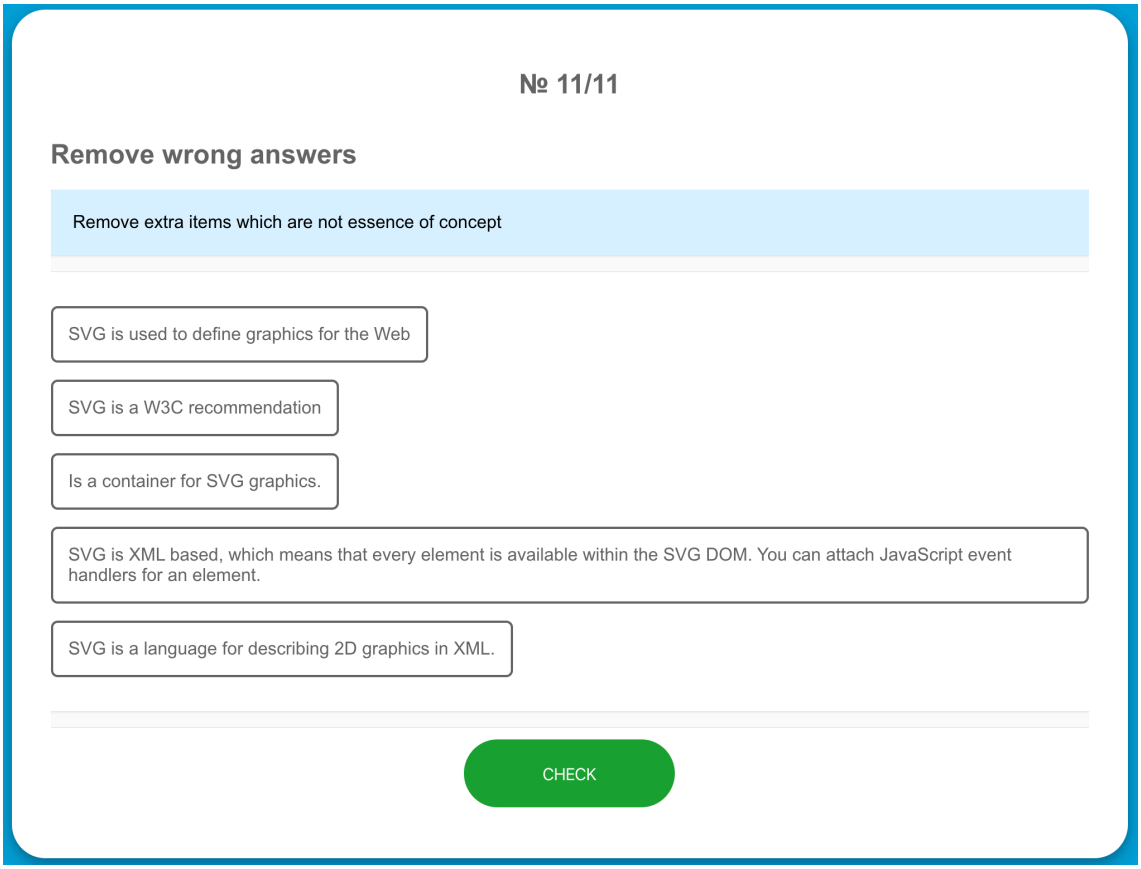

Рисунок 3.13. Приклад завдання вибору неправильних відповідей

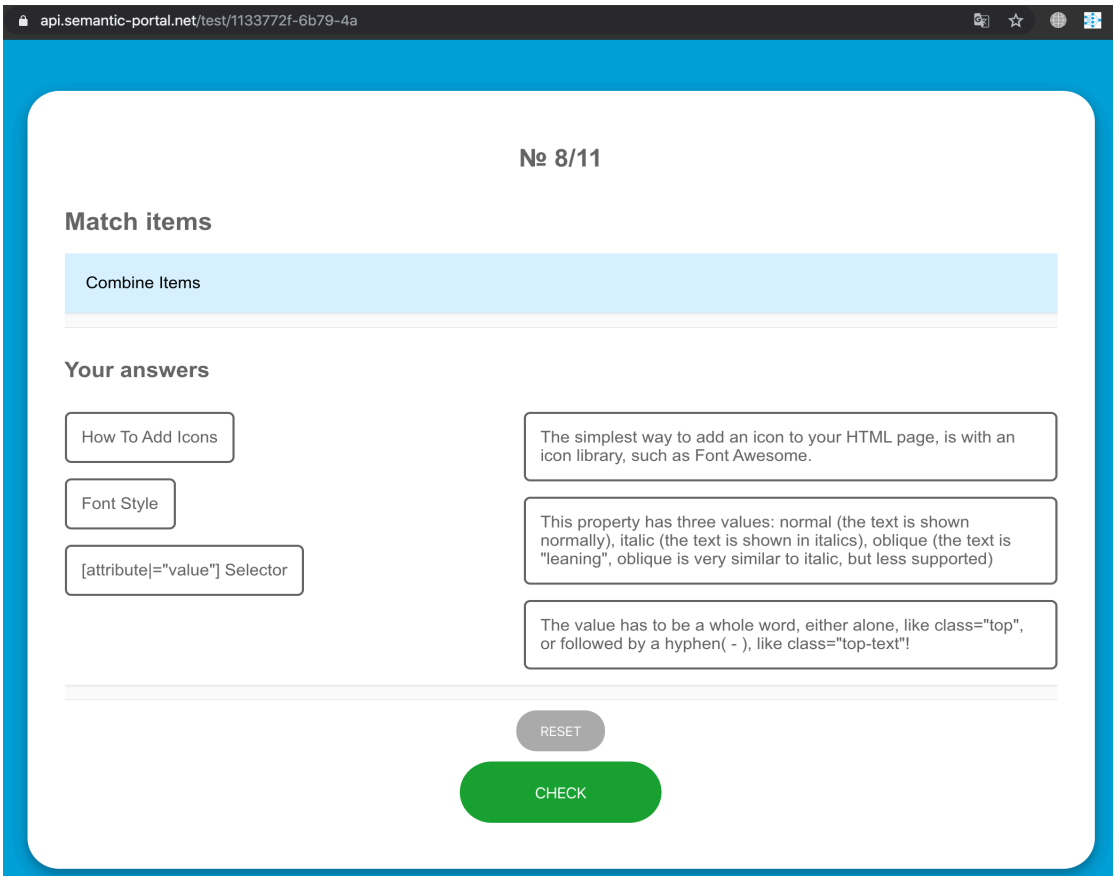

Рисунок 3.14. Приклад завдання на співставлення наборів

3.6. Формування рекомендацій по результатам тестування

Після проходження тесту система складає список тем необхідних для повторення на основі тих завдань, на які було дано неправильні відповіді. Цей список являє собою посилання на фрагмент навчального матеріалу основного порталу, з коротким описом цієї теми (рисунок 3.15)

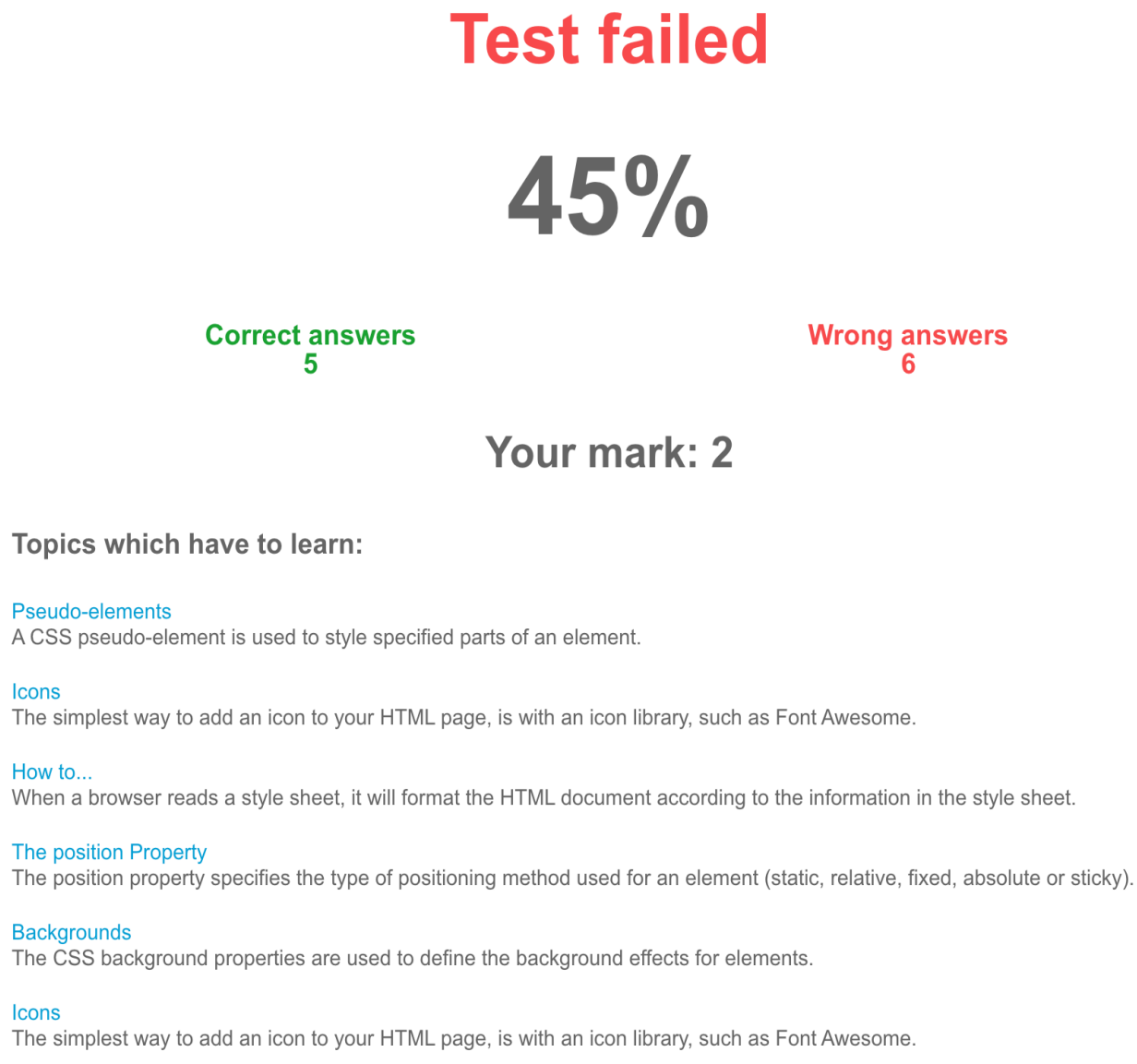

Рисунок 3.15. Рекомендаційна система

Список тем необхідних для повторення генерується в не залежності від того пройдено тест на позитивну оцінку чи ні. Якщо на хоча б одне х тестових завдань

було дано неправильну відповідь, ця тема з’явиться у списку.

3.5. Експеримент з контролю знань на основі розробленої системи

Тестування відбувалося для 29 студентів другого курсу по дисциплінам з інженерії програмного забезпечення. Теми, винесені на тестування в більшості стосувалися предметів, пройдених даною групою на попередньому році навчання, деякі теми стосувалися поточного семестру. Результати тестування (рисунок 3.16) оцінювалися у відсотковому відношенні вірних та хибних відповідей.

При проходженні тесту, що базується на моделі складності *Low*, результати тестування в переважній більшості відповідали оцінці 90%. На рівні *Middle* – 75%, в той час як на *Hard* – 65%.

Рівень *Low* перевіряє знання типу *Understand*, *Remember*. У цьому рівні тесту [30] рідко зустрічались важкі завдання, в основному вони потребували навичок запам’ятовування та орієнтування в контрольній області.

Рівень *Middle* завдяки завданню на співставлення *MI* перевіряє тип знань *Apply.* Тут витрачалося більше часу на осмислення, а також були необхідні навички застосовувати знання. Це можна аргументувати тим, що зустрічались поняття з різних навчальних фрагментів контрольної множини контенту, що вимагало від учня глибокого розуміння предметної області.

Рівень *Hard* завдяки завданням *GF* та, особливо, *RW* перевіряє тип знань *Analyze.* Користувачу було необхідно зрозуміти принцип, за яким потрібно прибрати зайві варіанти відповідей, що вимагало здатності проаналізувати твердження та володіти аналітичною здатністю в даній предметній області.

Слід зазначити, що студенти, що показали вищий рівень семестрового навчання, загалом мали вищий відсоток в тестуванні рівня *Hard*. Отже, можна казати про успіх диференційованого тестування на базі запропонованої моделі та необхідність його подальшого дослідження для повного покриття модифікованої

шкали Блума [7].

Рисунок 3.16. Результати тестування

3.6. Висновки

В даному розділі було запропоновано модернізацію побудови тестових завдань на базі ПТМ. Подано формальний апарат генерації тестів різного рівня складності з використанням модифікованої шкали Блума. Представлено нові моделі тестових завдань, що відповідають когнітивним цілям, видозмінено структуру шаблону. Розглянуто проблему використання аспектів понять та тез, що відносяться до них. Наведено графічні приклади тестових завдань та алгоритму їх побудови. Також запропоновано та здійснено програмну реалізацію рекомендаційної системи. Було проведено тестування, за результатами якого система показала свою надійність та швидкість. Описано у відсотковому відношенні кількість правильних відповідей студентів в залежності від складності тесту, що дало змогу оцінити відповідність завдань шкалі Блума. Загалом можна казати про успіх даної модифікації ПТМ з диференціацією за когнітивними цілями.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Реалізація будь-якої системи потребує перш за все аналізу та пошуку оптимального шляху вирішення цього завдання. Після аналізу постановки задачі було виділено основні критерії, яким має задовольняти система з точки зору програмної реалізації:

- незалежна розробка серверної та користувацької частин програмного продукту для забезпечення безпеки даних та ясності коду;
- гнучкість розробленої системи.

Після аналізу цих вимог було вирішено, що найбільш повно цим критеріям задовольняє об’єктно-орієнтований підхід побудови архітектури системи.

Об'єктно-орієнтоване програмування базується на таких принципах:

- інкапсуляція – реалізація та стан кожного об'єкта є приватними всередині визначеної межі або класу;
- абстракція – об'єкти розкривають лише внутрішні механізми, що мають відношення до використання інших об'єктів;
- наслідування – можуть бути призначені відносини та підкласи між об'єктами, що дозволяє розробникам повторно використовувати загальну логіку, зберігаючи унікальну ієрархію;
- поліморфізм – об'єкти можуть приймати більше однієї форми залежно від контексту.

4.1. Архітектура програмної системи

Основні функції програмної системи можна описати за допомогою use-case діаграми (рисунок 4.1)

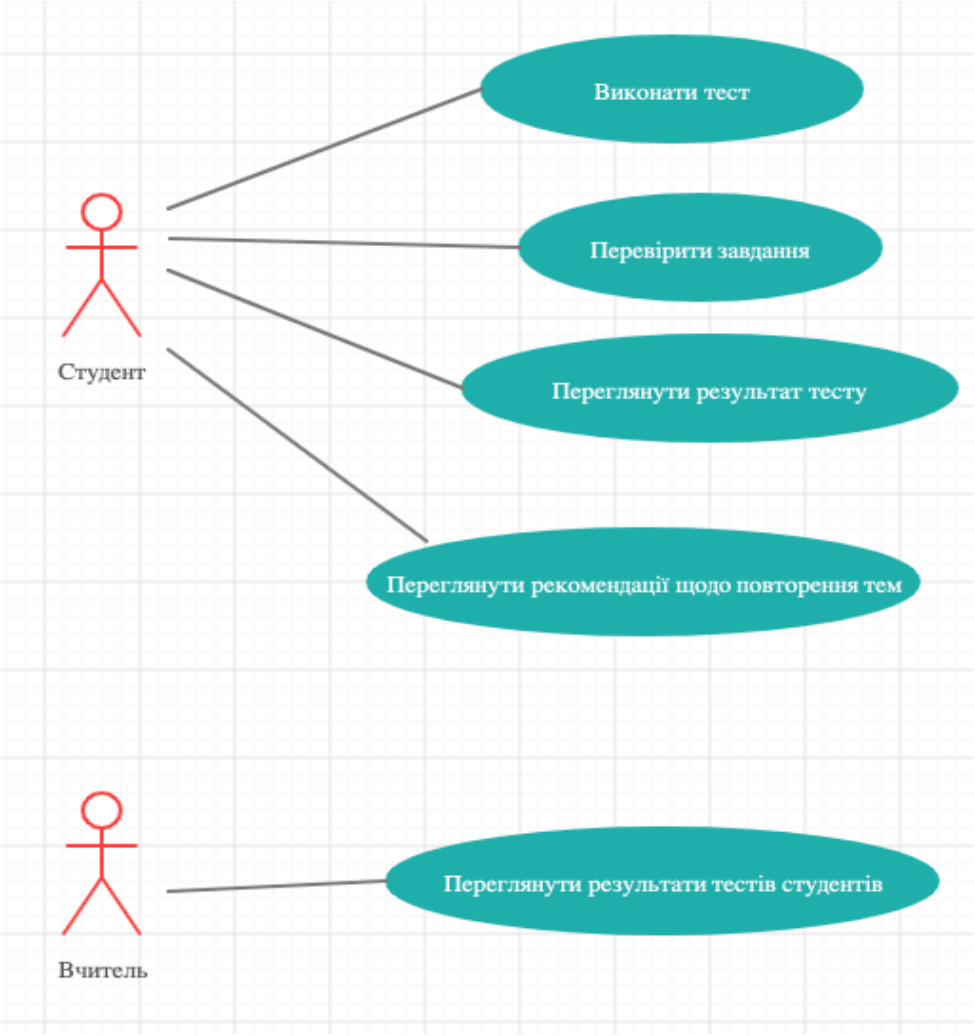

Рисунок 4.1. Use-case діаграма користувачів у системі

Програму було розбито на декілька модулів (рисунок 4.2), основним модулем є модуль тесту. Основним модулем є AppMod, в якому зберігаються основні класи програми. EnvMod відповідає за віртуальне середовище та налаштування програми, а також в ньому зберігаються усі версії станів бази даних. Модуль Cache допомагає програмі кешувати деякі дані для більш швидкої генерації тестового завдання та зменшує навантаження на БД. TestMod відповідає безпосередньо за генерацію тестових завдань, та отримання даних з бази даних.Модуль користувача дозволяє зберігати результати тестових завдань та виводити усі необхідні дані. Модуль View отримує актуальні дані щодо навчальних фрагментів, які можна використовувати в тесті та проводить відповідну фільтрацію, відбираючи фрагменти в яких достатньо даних для побудови тестового завдання

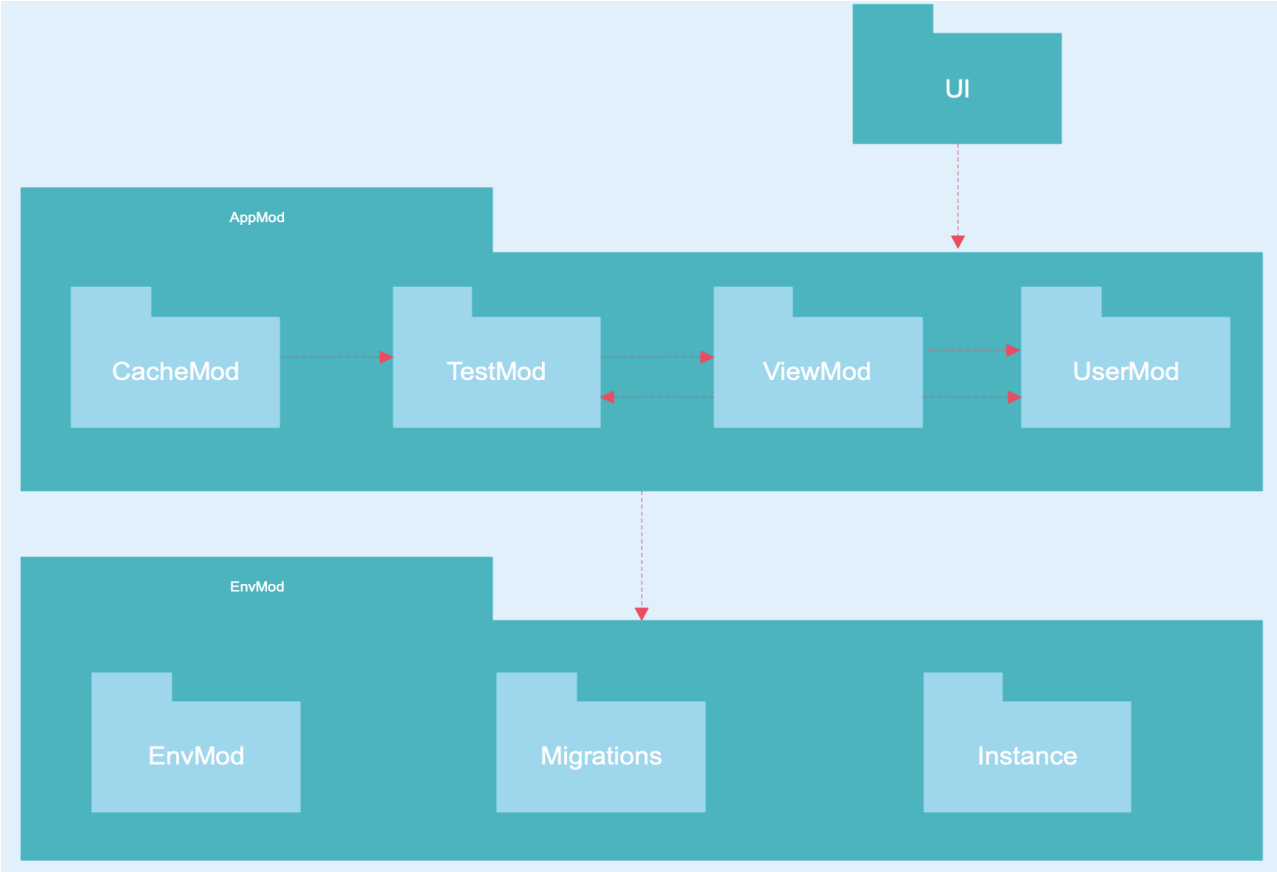

Рисунок 4.2. Діаграма пакетів

Точний опис кожного класу зобразимо діаграмою класів аналізу (рисунок 4.3)

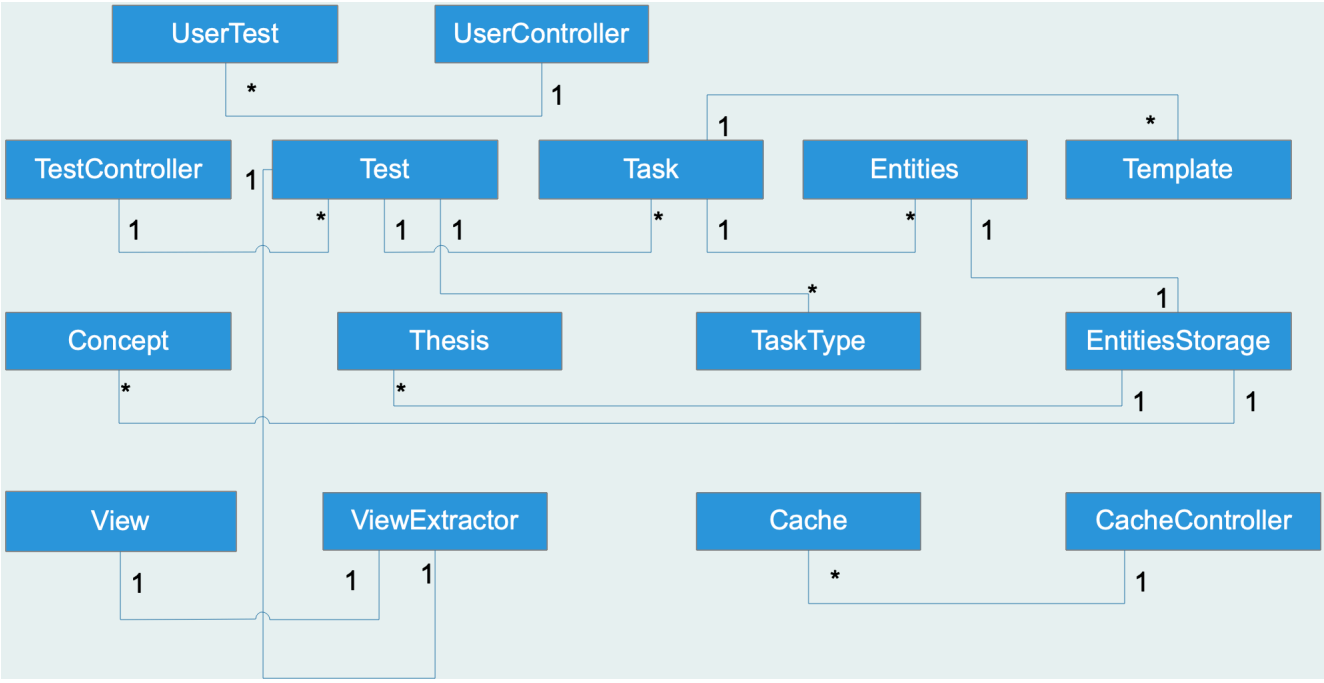

Рис 4.3. Діаграма класів аналізу серверної частини програми

Клас UserTest виконує роботу по створенню, оновленню даних по тесту, який проходить користувач. Test являє собою основний клас, який відповідає за керування іншими класами. Спочатку за допомогою класу View та ViewExtraxtor дістаються назви навчальних фрагментів , які підходять для генерації тестових завдань. Потім клас Template дістає усі необхідні шаблони та фільтрує їх. Далі дістаються усі поняття та тези, що знаходяться у відібраних навчальних фрагментах за допомогою класів Concept та Thesis. Оскільки в програмній реалізації використовується ORM, то необхідно усі поля понять та тез, які є об’єктами класів Concept, Thesis інкапсулювати, для цього слугує клас EntitiesStorage, який являється масивом з копіями цих об’єктів, проте не за посиланням, а за значенням. Якщо не створити такого контейнеру, то усі зміни внесені до полів об’єктів призведуть і до зміни полів у базі даних. Клас TaskType є набором полів з типами тестових завдань, так званий enum. Далі необхідно обрати випадковий шаблон для генерації завдання та передати контроль класу Task, який спочатку знайде усі можливі варіанти ПТ-елементу запитання, а потім усі правильні варіанти відповідей та дистрактори відповідно до обраного ПТ-елемету, що лежить в основі питання.

Основним алгоритмом є побудова тесту (рисунок 4.4), де приймається рішення щодо необхідних типів завдань, які треба згенерувати, в залежності від вибраної користувачем складності.

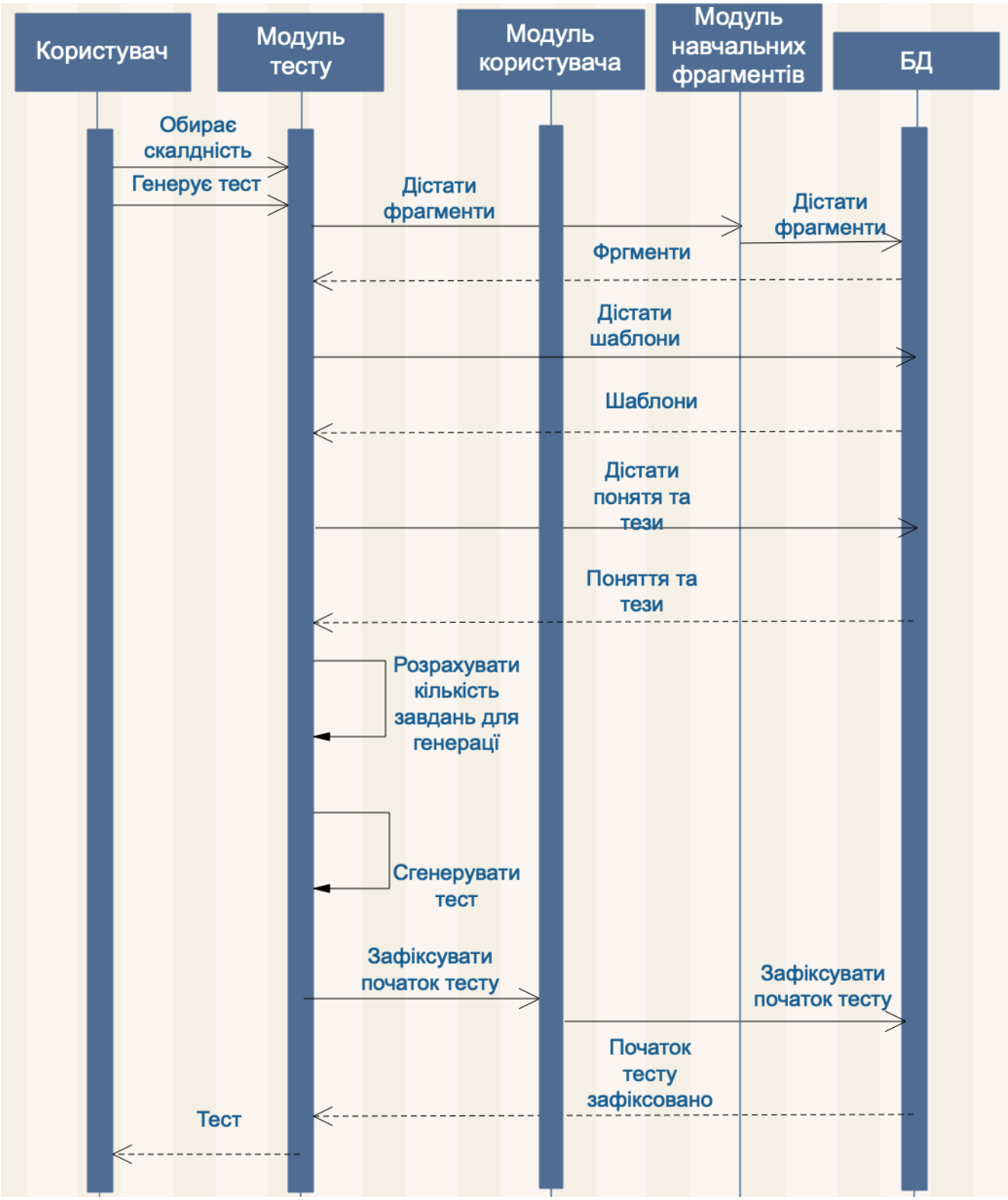

Рисунок 4.4. Діаграма послідовності

Кількість завдань та їх типи, що створюються в залежності від складності розраховується у відсотковому відношенні і не задається вручну, а генерується автоматично, це дозволяє керувати складністю ще більше. Також це впливає на

часові витрати проходження тестових завдань. Наприклад, для тестування якості так процесу генерації тесту можна вставити велику кількість тестових завдань, що будуть генеруватись, замість того, щоб викликати цей процес декілька разів.

Фіксація початку тесту необхідна для часового контролю, оскільки якщо викладач поставить умови проходження тесту в певний період, то без цього параметру неможливо буде нічого визначити. Якщо для деякого типу тестового завдання недостатньо даних або їх нема, то алгоритм замінює його на інший. Усі фрагменти, які не можна використовувати, видаляються зі списку для генерації, тому майже не виникає випадків коли тест не генеруються або генерується дуже довго. Завдяки модулю кешування описаному вище процес створення тестових завдань пришвидшено у 2 рази.

Опишемо процес генерації тестового завдання з точки зору процесів та потоків даних (рисунок 4.5 – 4.7)

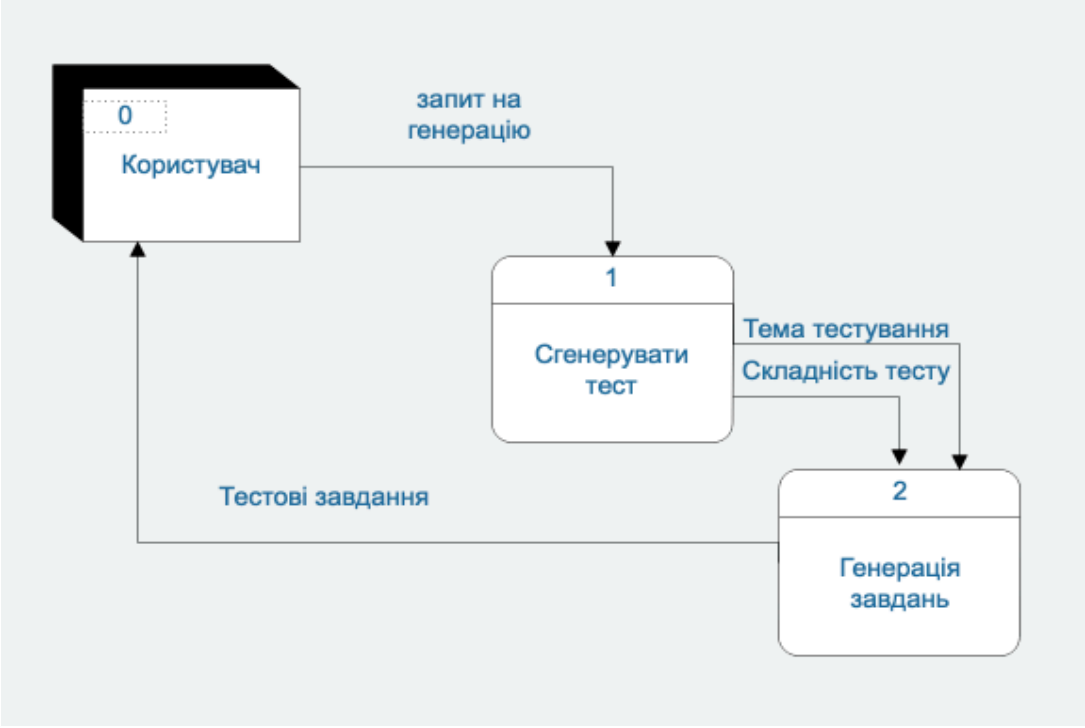

Рисунок 4.5. Загальна діаграма DFD

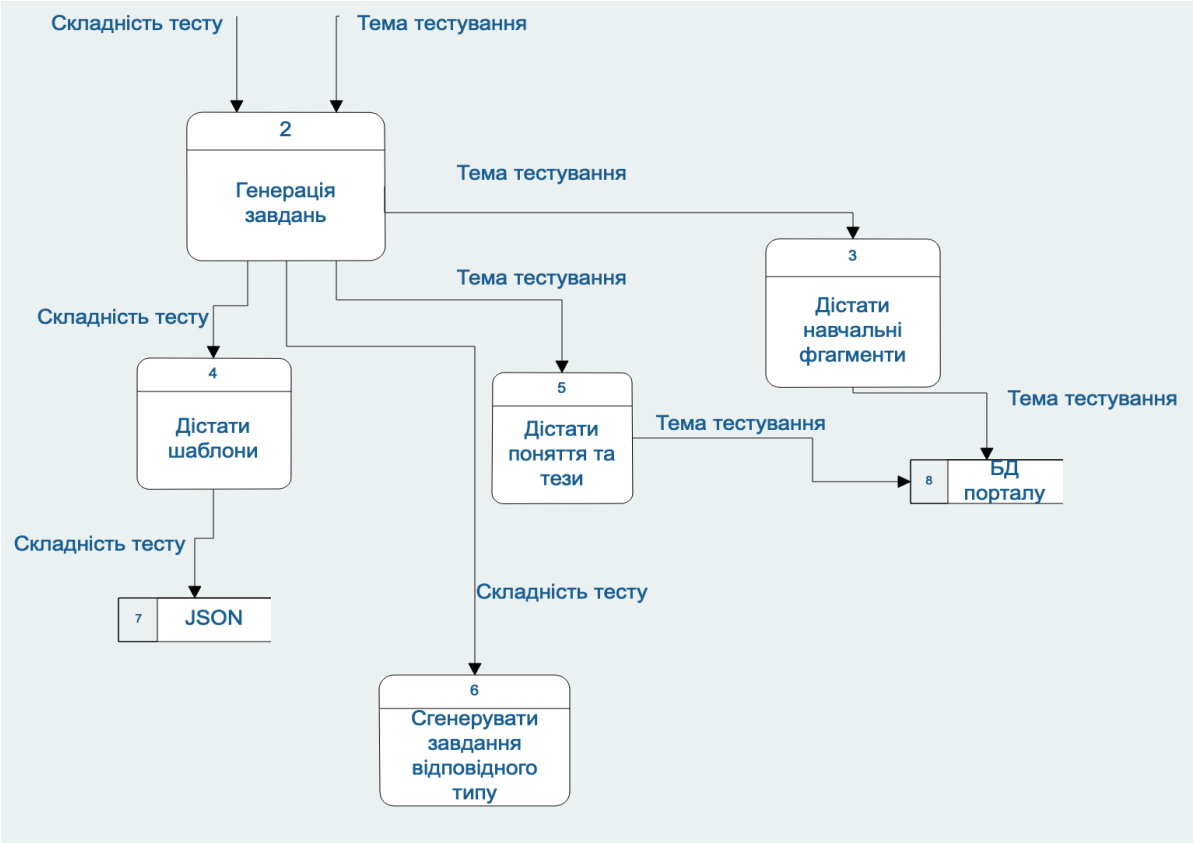

Рисунок 4.6. Декомпозиція процесу генерації тесту DFD діаграма

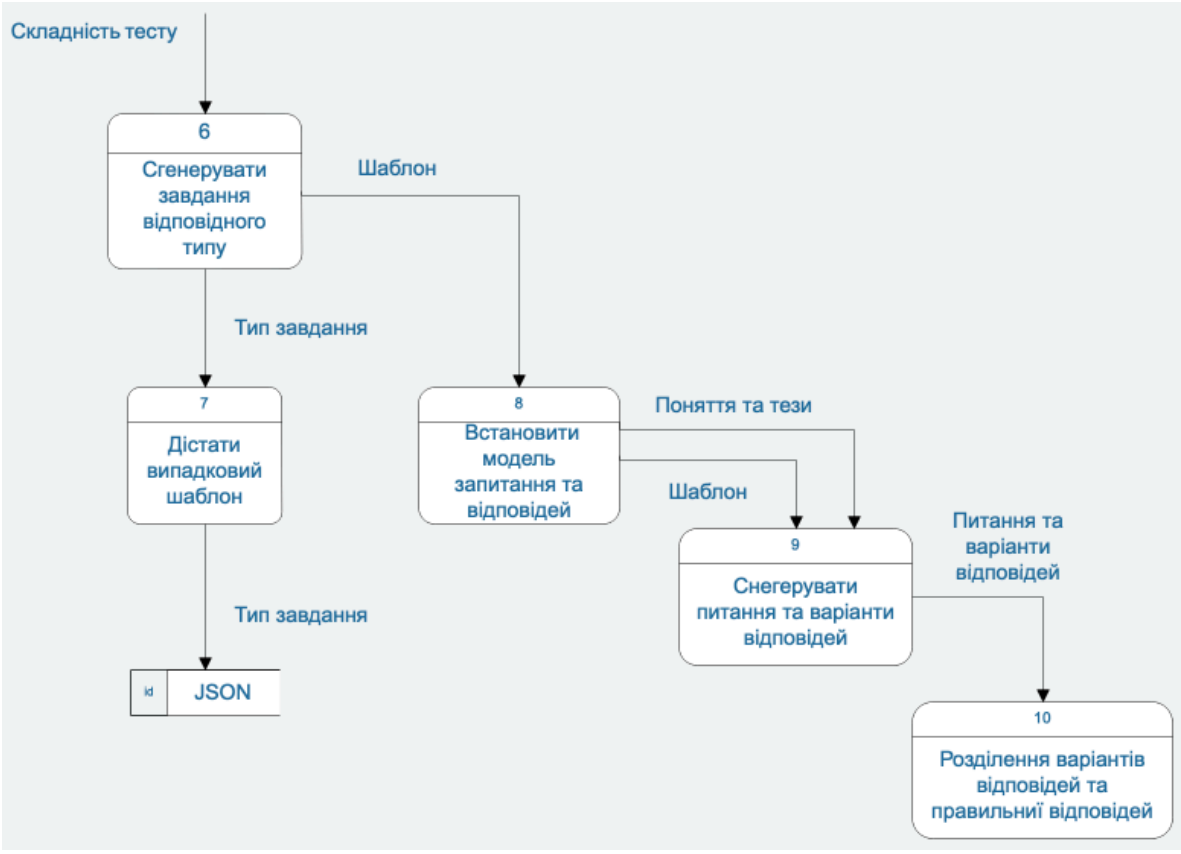

Рисунок 4.7. Декомпозиція процесу генерації завдання DFD діаграма

4.2. Опис бази даних

База даних в системі порталу є дуже великою, зобразимо концептуальну схему (рисунок 4.8) лише частини, яка безпосередньо використовується для тестування.

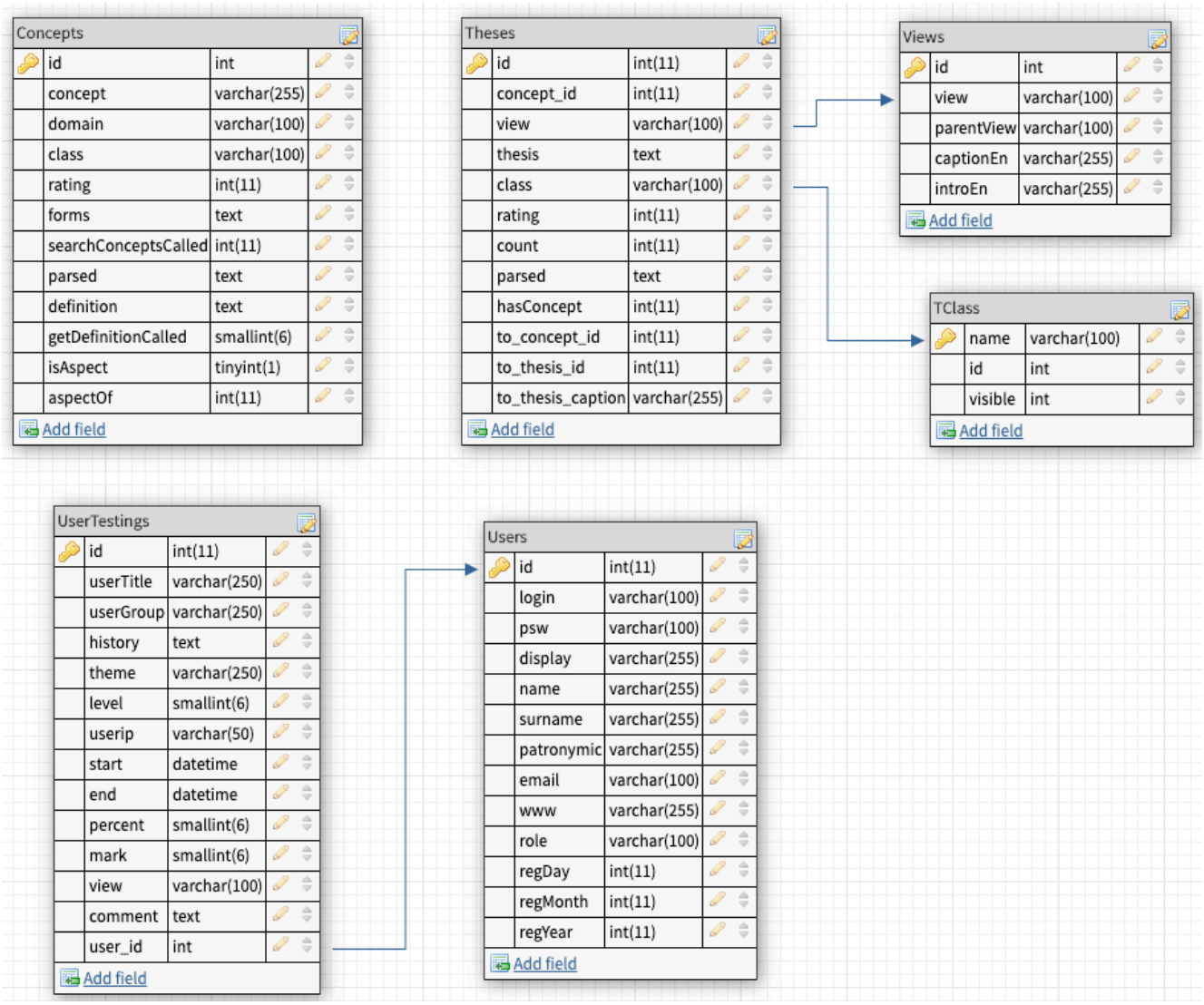

Рисунок 4.8. Концептуальна схема частини БД, що використовується в системі

Основними полями в таблиці Concepts, що являє собою набір понять, є такі поля:

- назва поняття(concept);
- домен(domain), тобто до якої предметної області відноситься поняття;

- клас поняття(class);
- словоформи поняття(forms);
- чи є поняття аспектом(isAspect);
- головне поняття аспекту(aspectOf).

Таблиця Theses, містить с собі набір тез понять, також має багато полів, відзначимо основні:

- назва тези(thesis);
- поняття, до якого відноситься теза(concept_id);
- до якого фрагменту навчального матеріалу відноситься теза(view);
- клас тези(class);
- чи містить теза поняття(hasConcept), використовується для полегшення програмної реалізації, визначає чи містить теза поняття, оскільки використання таких тез потребує окремої обробки;
- теза, яка є поясненням до іншої тези(to_thesis_id).

Оскільки центральним носієм інформації є навчальні фрагменти, необхідно також описати ключові поля таблиці Views, яка й відповідає їм:

- назва навчального фрагменту(view);
- навчальний фрагмент, який є батьківським для даного фрагменту(parentView), оскільки матеріал подається структуровано і ділиться на розділи та підрозділи;
- назва теми до якої відноситься навчальний фрагмент(captionEn), англійською мовою, оскільки портал функціонує саме на ній;
- короткий опис теми(introEn), також англійською.

В системі наявні службові класи тез, які допомагають інженерам формувати базу знань, для цього існує таблиця TClass. В ній зберігаються усі класи тез, з поміткою які можна використовувати для тестування, оскільки саме службові не можна. Опишемо ключові поля цієї таблиці:

- назва класу тези(name);
- чи можна використовувати клас тези для тестування(visible).

Оскільки тестування відбувається для людей, то також наявна таблиця Users, що є списком користувачів системи і містить наступні головні поля:

- логін користувача(login);
- пароль(password);
- ім’я користувача(name);
- прізвище(surname);
- ім’я по батькові(patronymic);
- електронна пошта(email);
- роль, що розділяє права доступу на порталі(role);
- дата реєстрації, regDay, regMonth, regYear(день, місяць, рік реєстрації).

Усі результати тестування для контролю знань пишуться у таблицю UserTestings, яка зберігає данні пройдених тестів, має наступний набір полів:

- ПІБ(UserTitle);
- групу студента(userGroup);
- тему, за якою відбулось тестування(theme);
- обраний рівень складності(level);
- айпі адреса користувача, для зменшення шансу проходження тесту іншим студентом(userip);
- дата початку тесту(start);
- дата закінчення тесту(end);
- відсотковий результат проходження тесту(percent);
- оцінка за шкалою ECTS, тобто 100 бальна система, проте яка потім переводиться у 5 бальну(mark).

4.3. Засоби програмної реалізації

В даній роботі основною мовою програмування на сервері є Python, в клієнтській частині була використана мова Javascript. Для доступу та зберігання

даних було використано такі бази даних як Redis та MySQL. Програма була розгорнута на сервері hetzner.

4.3.1 Мова програмування Python

Python – інтерпретована мова програмування високого рівня, загального призначення. Створена Гвідо ван Россумом та вперше випущена в 1991 році, філософія дизайну Python підкреслює читабельність коду завдяки помітному використанню значного відступу. Його мовні конструкції та об'єктно-орієнтований підхід мають на меті допомогти програмістам написати чіткий логічний код для малих та масштабних проектів.

Python динамічно збирає сміття. Він підтримує декілька парадигм програмування, включаючи процедурне, об'єктно-орієнтоване та функціональне програмування. Python часто описується мовою "включені батареї" завдяки своїй всебічній бібліотеці стандартів.

Пітон був задуманий в кінці 1980-х років як наступник мови ABC. Python 2.0, випущений у 2000 році, представив такі функції, як розуміння списків та систему збору сміття, здатну збирати еталонні цикли. Python 3.0, випущений у 2008 році, був основною редакцією мови, яка не є повністю зворотньо сумісною, і багато коду Python 2 не працює на модифікованому Python 3.

Мова Python 2, тобто Python 2.7.x, припинює підтримуватись менше ніж через місяць 1 січня 2020 року (після продовження підтримки; вперше заплановано на 2015 рік), і команда добровольців Python не виправить проблеми безпеки та не покращить її іншими засобами після цієї дати. Із закінченням циклу буде підтримуватися лише Python 3.5.x та пізніші версії.

Інтерпретатори Python доступні для багатьох операційних систем. Глобальне співтовариство програмістів розробляє та підтримує CPython, відкритий вихідний код. Некомерційна організація, програмний фонд Python, спрямовує ресурси для розробки Python та CPython.

4.3.2 Мова програмування Javascript

JavaScript, що часто скорочується як JS, – це об'єктно-орієнтована, сучасна об'єктно-орієнтована мова програмування, яка відповідає специфікації ECMAScript. У JavaScript є синтаксис фігурної дужки, динамічне введення тексту, орієнтована на прототип об'єктна орієнтація та функції першого класу.

Поряд із HTML та CSS, JavaScript є однією з основних технологій всесвітньої павутини. JavaScript включає інтерактивні веб-сторінки і є невід'ємною частиною веб-додатків. Переважна більшість веб-сайтів використовують його , а основні веб-браузери мають спеціальний механізм JavaScript для його виконання.

Як мова з багатьма парадигмами, JavaScript підтримує керовані подіями, функціональні та імперативні (включаючи об'єктно-орієнтовані та засновані на прототипі) стилі програмування. Він має API для роботи з текстом, масивами, датами, регулярними виразами та DOM, але сама мова не включає жодних потоків вводу-виводу, таких як мережеві засоби, сховища чи графічні засоби. Він покладається на головне середовище, в яке він вбудований, щоб забезпечити ці функції.

Спочатку впроваджений лише клієнтською стороною у веб-браузерах, двигун JavaScript тепер вбудований у багато інших типів хост-програмного забезпечення, включаючи сторону сервера у веб-серверах та базах даних, а також у не веб-програмах, таких як текстові процесори та програмне забезпечення PDF, а також під час виконання середовища, які роблять JavaScript доступним для написання мобільних та настільних програм, включаючи віджети для настільних ПК.

Терміни Vanilla JavaScript і Vanilla JS відносяться до JavaScript, не поширюються жодними рамками та додатковими бібліотеками. Сценарії, написані у Vanilla JS, є простим кодом JavaScript.

Хоча між JavaScript та Java є схожість, включаючи назву мови, синтаксис та відповідні стандартні бібліотеки, обидві мови відрізняються та сильно

відрізняються за дизайном. На JavaScript впливали мови програмування, такі як Self та Scheme.Формат серіалізації JSON, який використовується для зберігання структур даних у файлах або для передачі їх по мережах, заснований на JavaScript.

4.3.3 База даних MySQL

MySQL – це система управління реляційними базами даних з відкритим кодом (RDBMS). Її назва – це поєднання "My", ім'я дочки співзасновника Майкла Віденуса та "SQL", абревіатура для Structured Query Language.

MySQL – це безкоштовне програмне забезпечення з відкритим кодом за умовами загальної публічної ліцензії GNU, а також доступне під різноманітними власними ліцензіями. MySQL належить та спонсується шведською компанією MySQL AB, яку придбала компанія Sun Microsystems (зараз корпорація Oracle). У 2010 році, коли Oracle придбав Sun, Widenius відправив проект MySQL з відкритим кодом для створення MariaDB.

MySQL є компонентом стеку програмного забезпечення для веб-додатків LAMP (та інших), що є абревіатурою для Linux, Apache, MySQL, Perl / PHP / Python. MySQL використовується багатьма веб-програмами, керованими базами даних, включаючи Drupal, Joomla, phpBB та WordPress. MySQL також використовується на багатьох популярних веб-сайтах, включаючи Facebook, Flickr, MediaWiki, Twitter, та YouTube.

4.3.4 База даних Redis

Redis (Сервер віддаленого словника) – це проект структури даних в пам'яті, що реалізує розподілену базу даних зі значеннями ключів у пам'яті з необов'язковою довговічністю. Redis підтримує різні види абстрактних структур даних, такі як рядки, списки, хеш, множини, відсортовані множини, HyperLogLogs,

бітовий хеш, потоки та просторові індекси. Проект в основному розробляється Сальваторе Санфіліппо, а станом на 2019 рік фінансується Лабораторіями Redis. Це програмне забезпечення з відкритим кодом, випущене за ліцензією BSD на 3 статті.

4.5. Методика роботи з системою

Першою сторінкою порталу (рисунок 4.9) є авторизація, де необхідно ввести прізвище та ім’я, а також групу, в якій навчається студент.

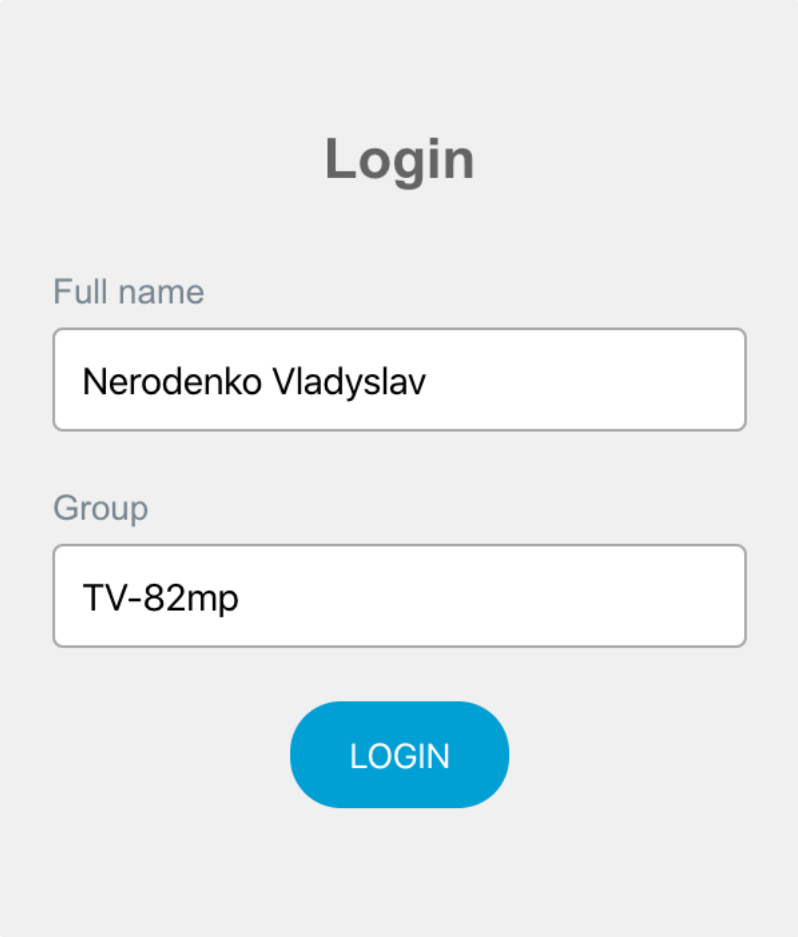

Рисунок 4.9. Авторизація користувача

Після чого користувач опиняється на головній сторінці системи тестування (рисунок 4.10). У лівому верхньому куті знаходиться посилання на головний

портал. На цій сторінці відображаються усі доступі теми для тестування, в кожному блоці (рисунок 4.11). Спочатку необхідно обрати складність тестування: Low, Middle або Hard, потім натиснути на кнопку generate test. Якщо не обрати складність, то буде генеруватись тестове завдання рівня Low, яке встановлене за замовчуванням.

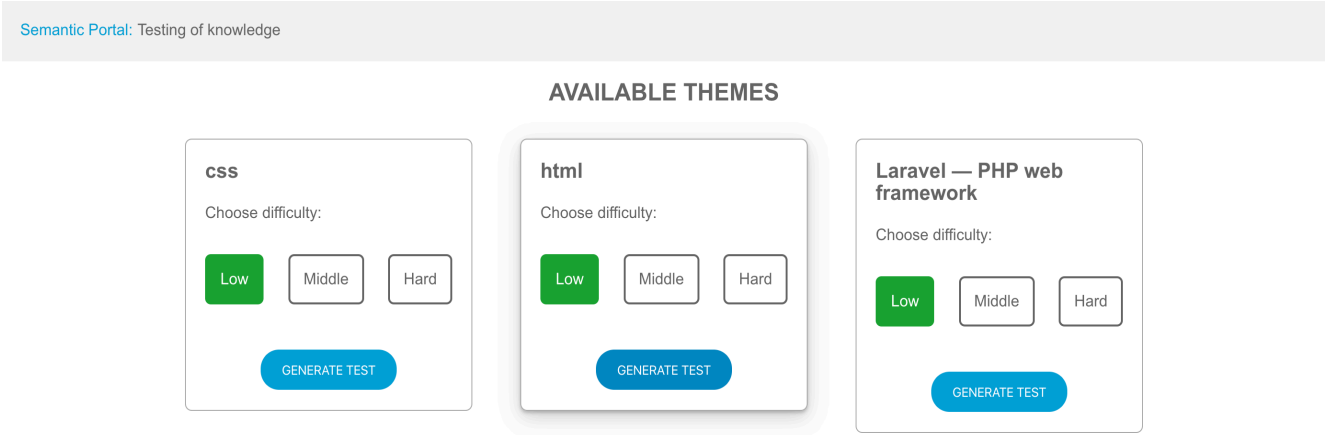

Рисунок 4.10. Головна сторінка системи тестування

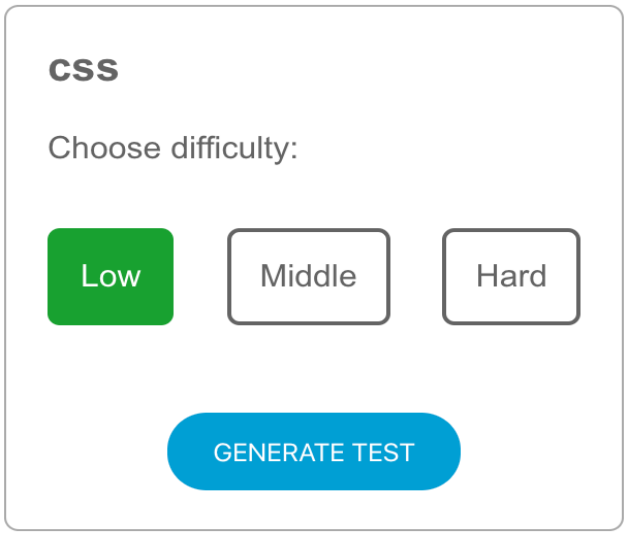

Рисунок 4.11. Блок вибору складності

Після цього буде згенеровано тест і відображено перше завдання для проходження (рисунок 4.12). При наведенні та натиснені на один з блоків (рисунок 4.13). Зазначимо, що для типів завдань множинного вибору, робота з інтерфейсом буде аналогічною.

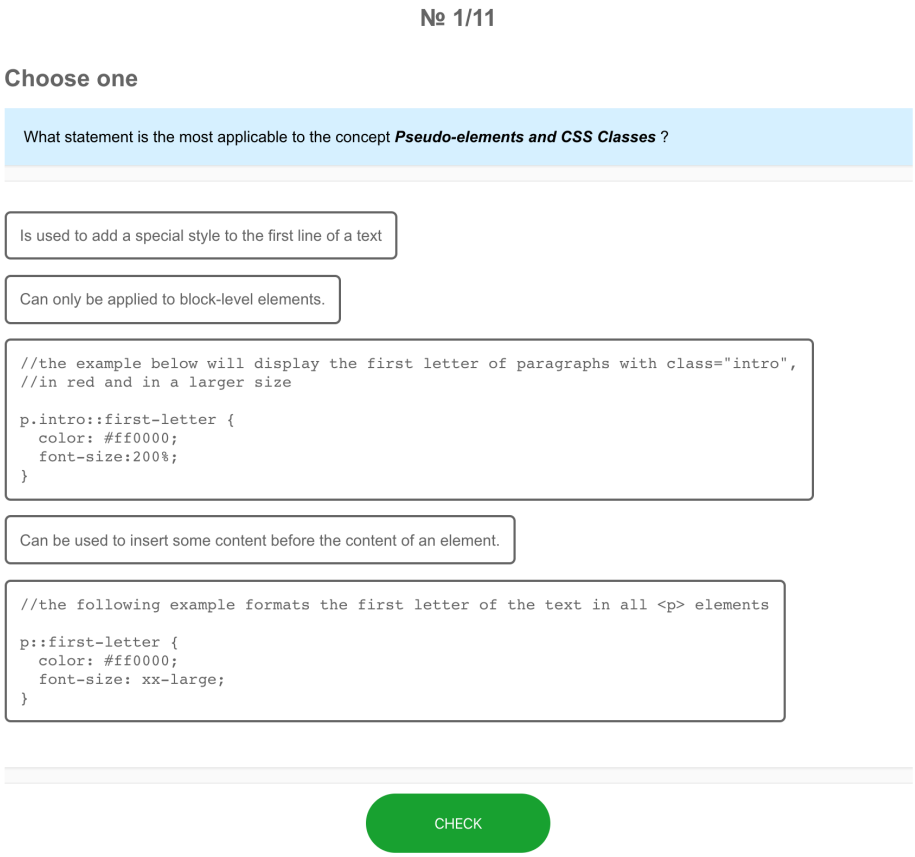

Рисунок 4.12. Перше завдання

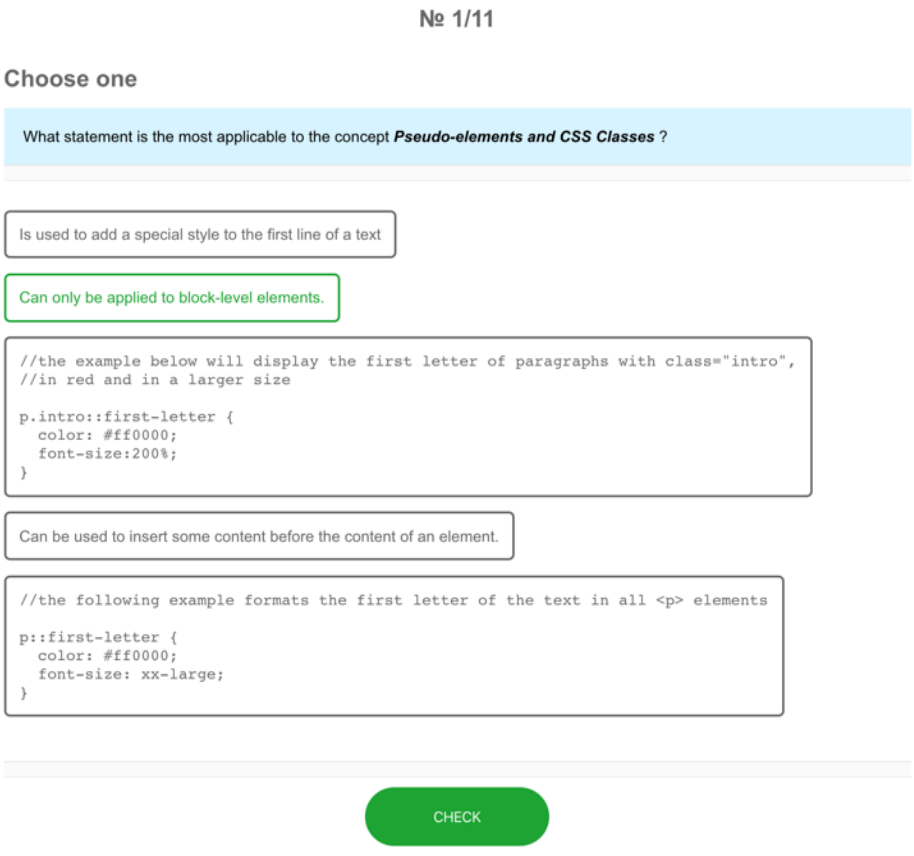

Рисунок 4.13. Підсвітка вибраного варіанту відповіді

Коли вибрано варіант відповіді необхідно натиснути кнопку check в нижній частині тесту (рисунок 4.14) для перевірки завдання.

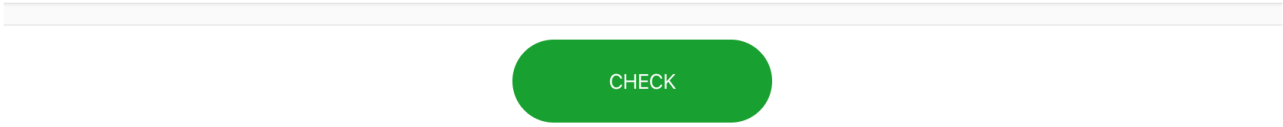

Рисунок 4.14. Кнопка перевірки тестового завдання

Далі відображається наступний інтерфейс (рисунок 4.15). Червоним обідком позначаються неправильні варіанти відповідей завдання. Червоною заливкою з хрестиком відображається неправильний варіант відповіді, обраний користувачем.

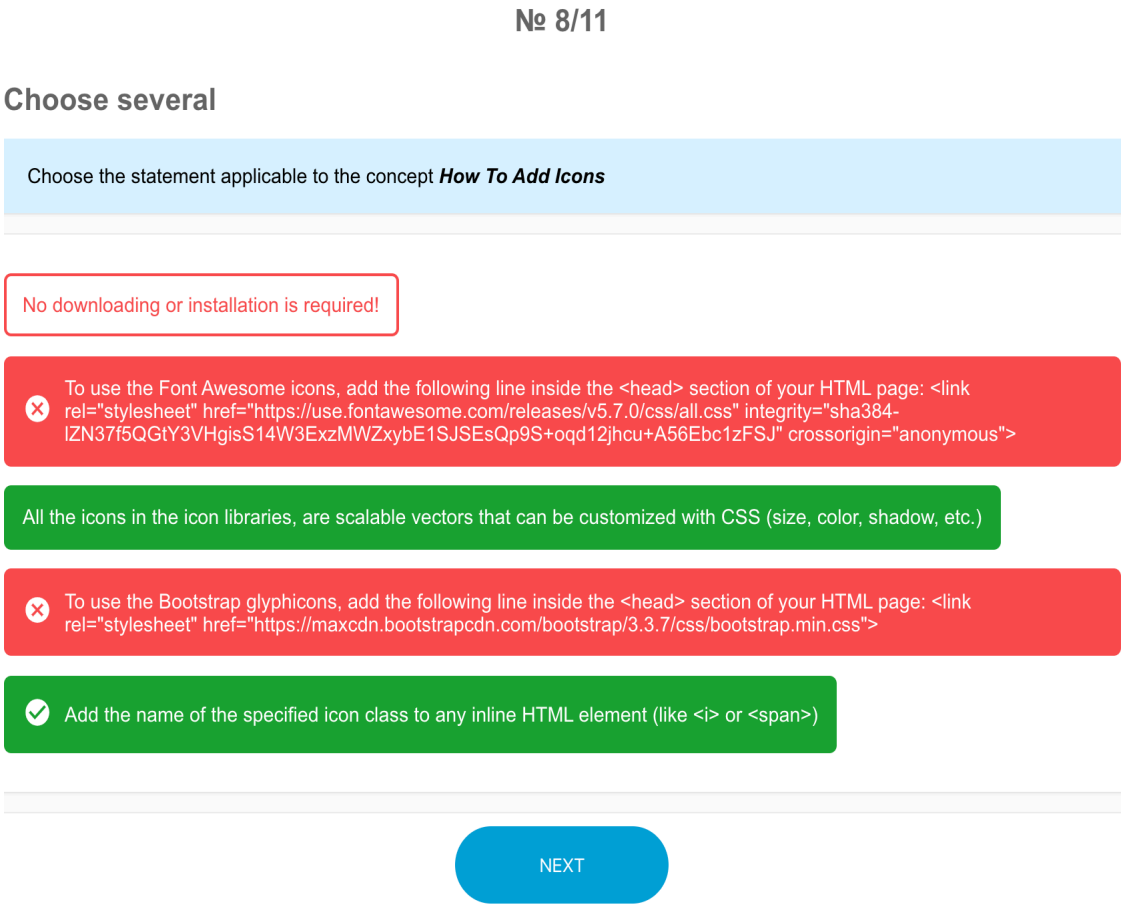

Рисунок 4.15. Приклад інтерфейсу після натиснення кнопки check

Зеленою заливкою відображаються правильні варіанти відповідей, які користувач не обрав. І нарешті правильні варіанти відповідей відображаються зеленою заливкою з пташкою. Після перевірки завдання необхідно натиснути кнопку next. Для завдань відкритого типу, наприклад, заповнення прогалини в твердженні необхідно ввести відповідь у біле поле для вводу (рисунок 4.16).

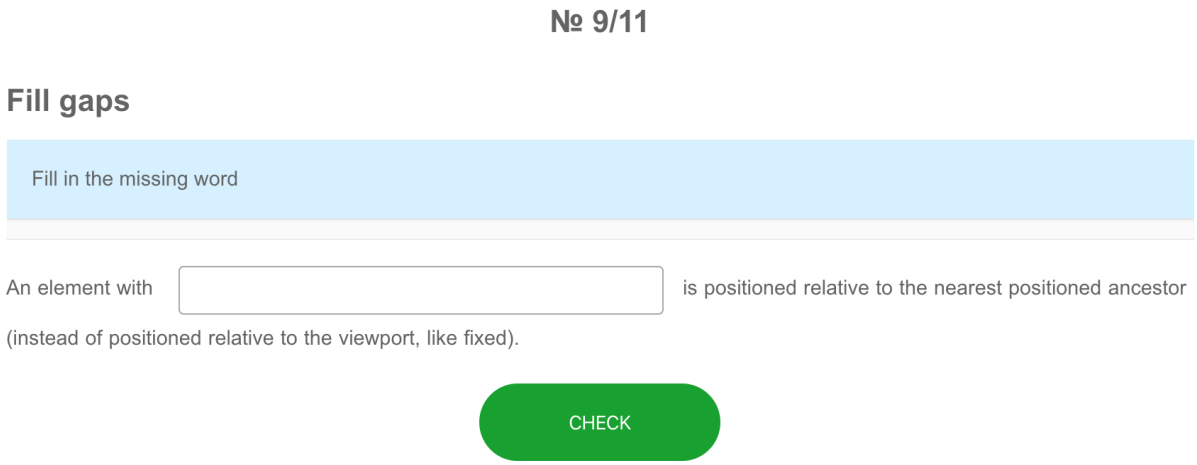

Рисунок 4.16. Приклад вводу відповіді на завдання відкритого типу

У разі правильної відповіді поле заповниться зеленим кольором (рисунок 4.17).

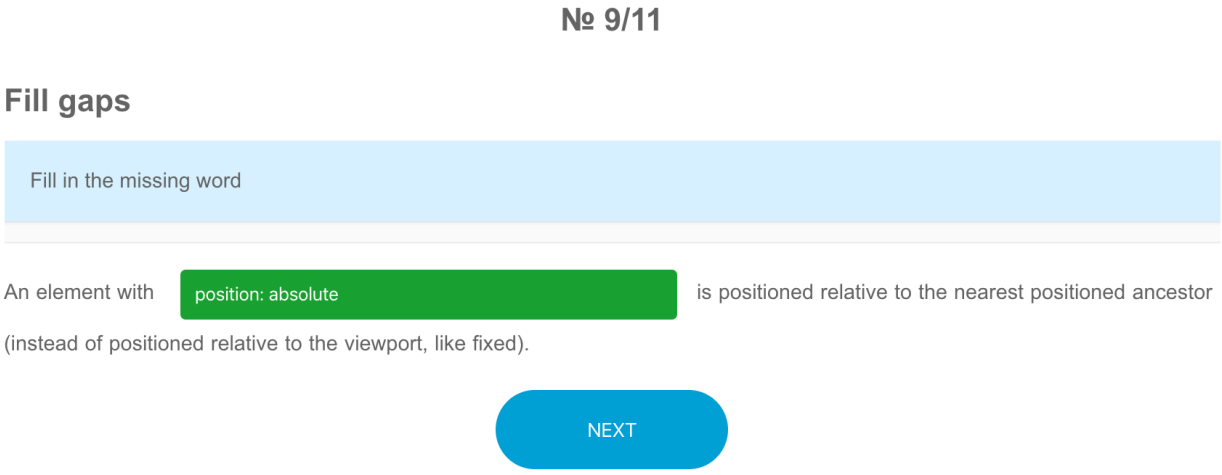

Рисунок 4.17. Приклад успішного проходження завдання відкритого типу

Якщо завдання провалено, поле для вводу заповниться червоним кольором (рисунок 4.18). А також буде виведено список усіх правильних відповідей. Для типу завдання “впишіть відповідь” процедура заповнення та перевірки аналогічна.

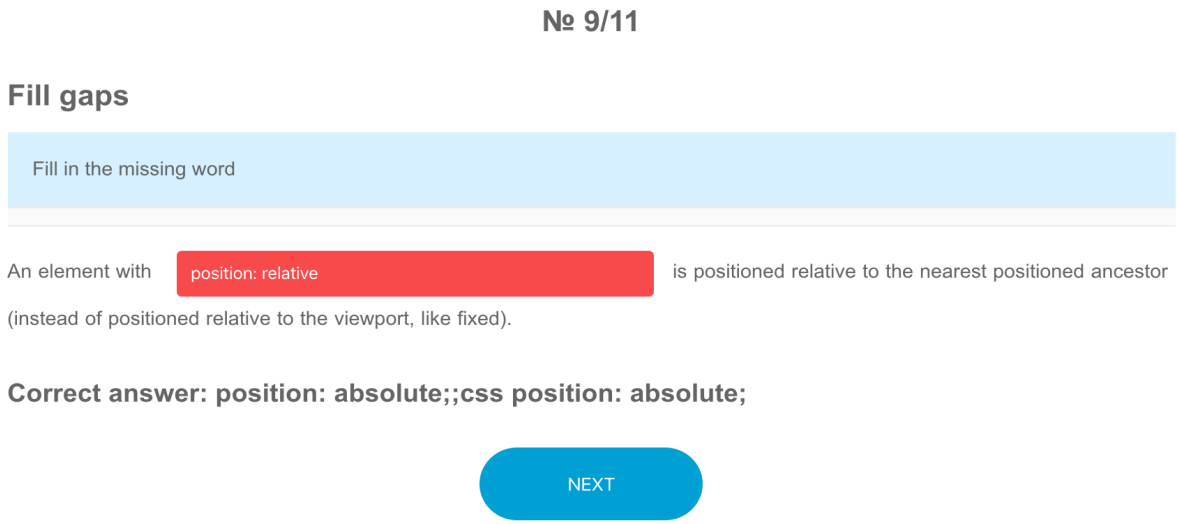

Рисунок 4.18. Приклад провалу завдання відкритого типу

Для типу завдання на співставлення (рисунок 4.19), зліва знаходяться поняття, справа тези.

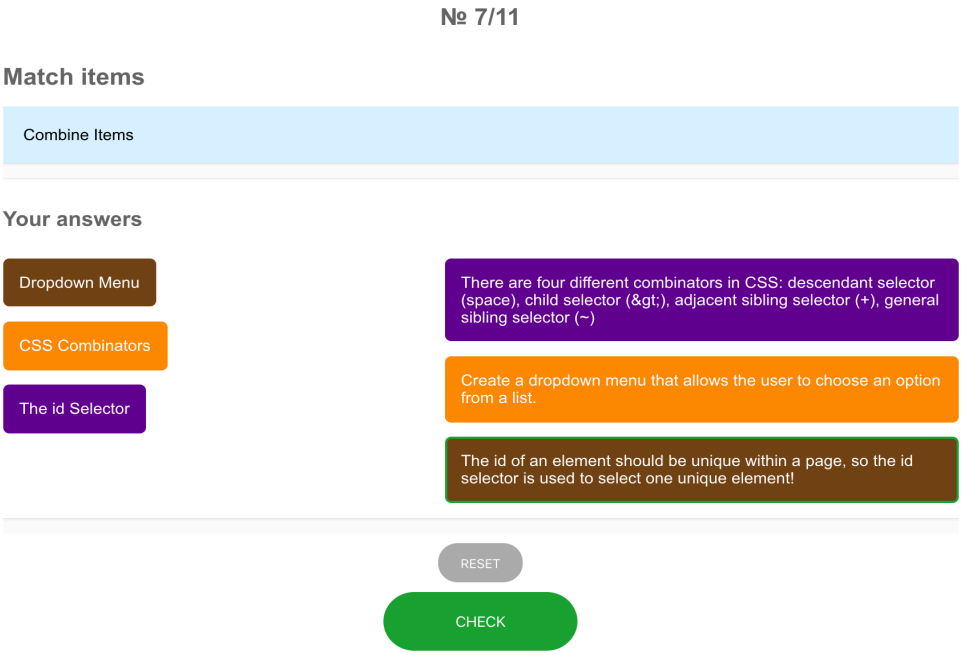

Рисунок 4.19. Приклад роботи із завданням на співставлення

Необхідно зіставити правильні пари, які помічаються відповідним кольором. Наприклад, якщо поняття зліва було обрано і заповнилось помаранчевим кольором, то відповідна теза, яка буде обиратись до цього поняття, заповниться тим самим кольором. Після перевірки цього завдання (рисунок 4.20) відповідні правильні пари варіантів відповідей(не користувача) відображаються знизу над завданням і помічаються певним кольором з пташками. Як можемо бачити в даному тесті на усі завдання було дано неправильні відповіді, які помічаються парами кольорів з хрестиками. На разі правильних відповідей обрані пари помічались би пташками (рисунок 4.21).

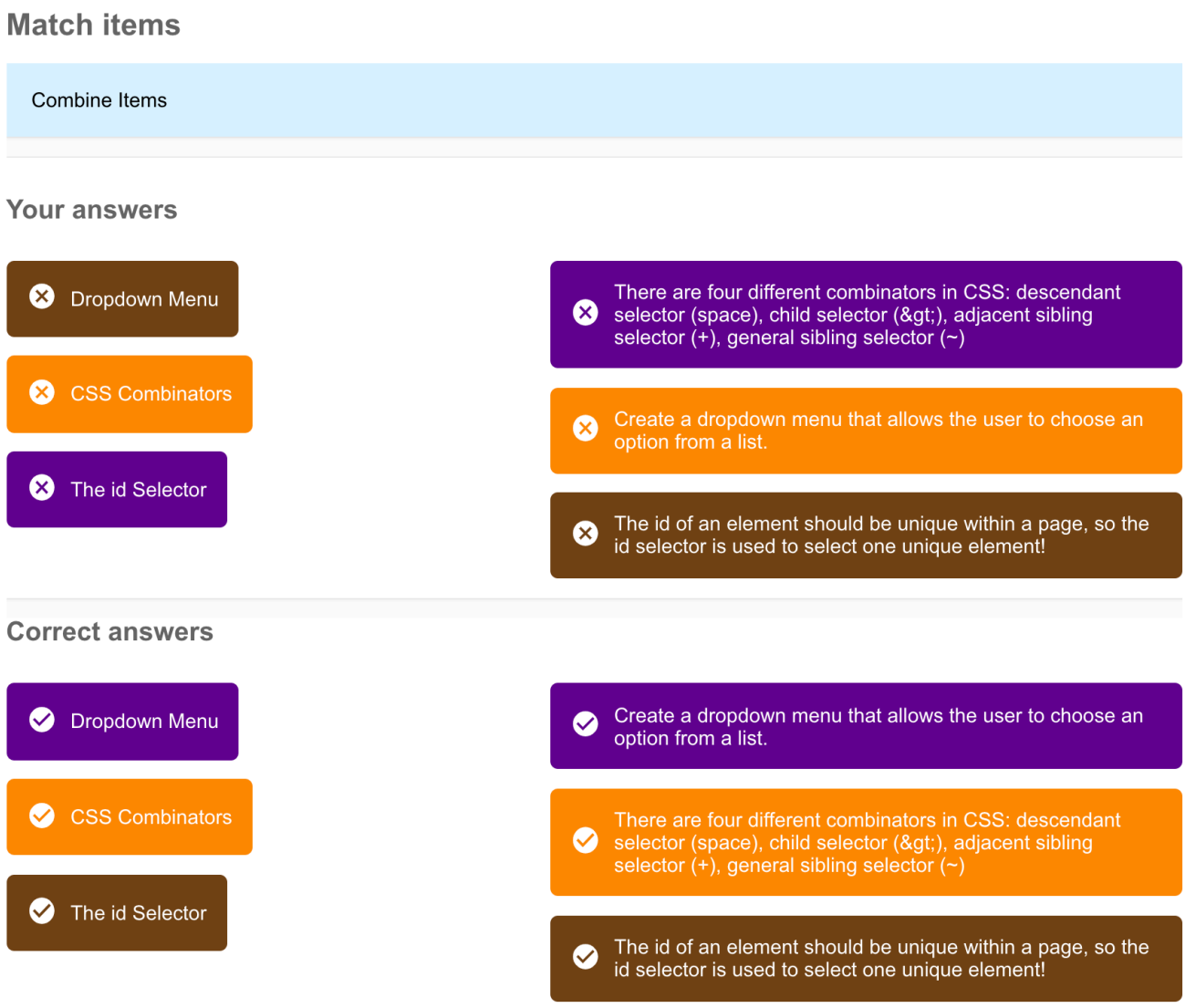

Рисунок 4.20. Перевірка завдання на співставлення, провал завдання

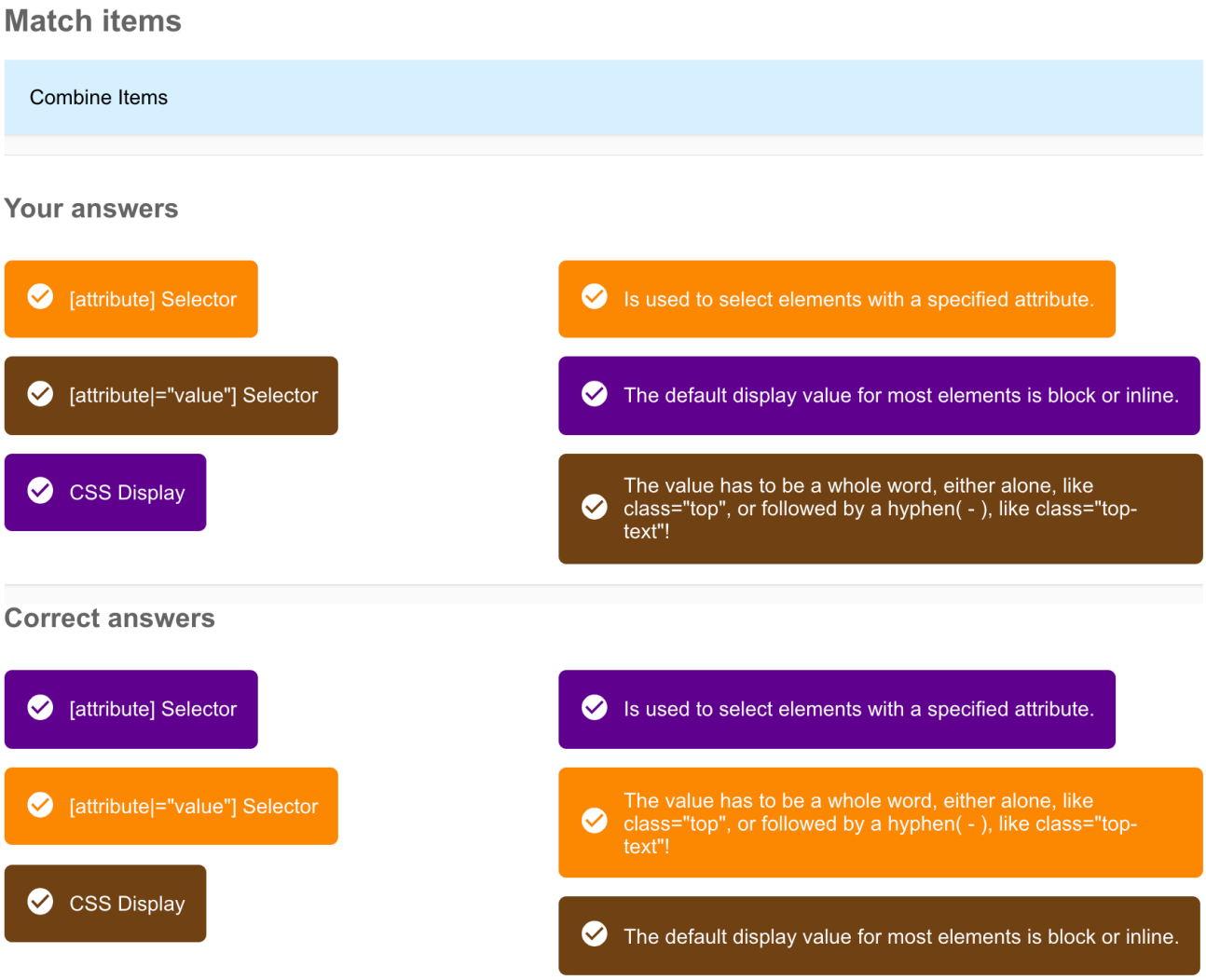

Рисунок 4.21. Успішне проходження завдання на співставлення

Завершальним етапом є результати тестування є результати тестування (рисунок 4.22). Оцінка ставиться за 5-ти бальною шкалою, також видається кількість правильних, включаючи відсоткове їх відношення, та неправильних відповідей. Список, який зображений знизу, відображає список тем, необхідних для повторення, зіставлений на основі провалених завдань. Кожен елемент цього списку є посиланням на основний портал, з коротким описом теми, де і зберігаються навчальні фрагменти. Кнопка back to themes, повертає користувача на головну сторінку порталу (рисунок 4.10).

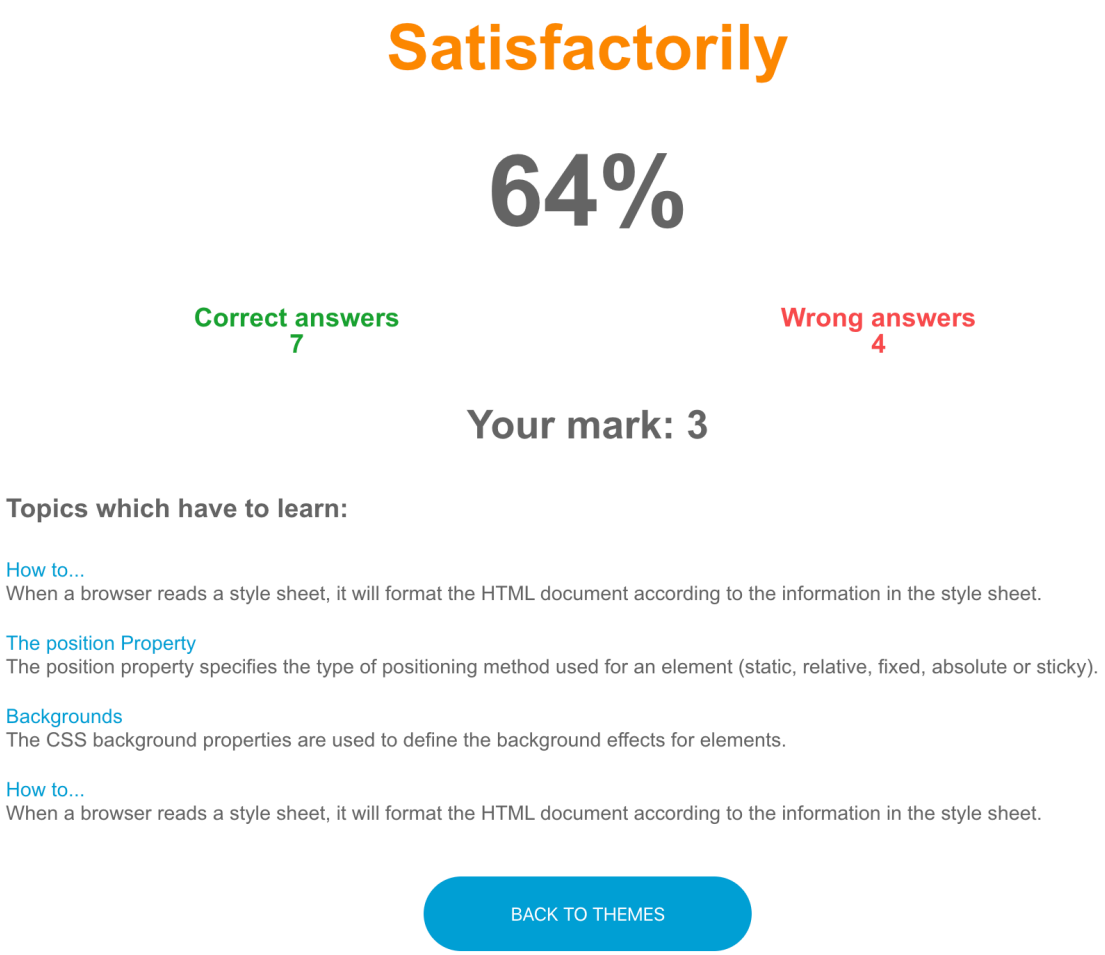

Рисунок 4.22. Результати тестування

4.6. Висновки

В даному розділі було описано програмну реалізацію системи з різних точок зору, подано діаграму компонентів, що відображає загальну архітектуру системи. Було описано зв’язки між різними сутностями програми, що більш детально описує структуру. Розроблено гнучку архітектуру, що базується на принципах ООП. Кожний компонент є незалежним, що сильно вплинуло на забезпечення адаптивності, а також дозволило провести ефективне модульне тестування програми. Завдяки модульності реалізується висока надійність системи. Архітектура побудована таким чином, що програма може бути інтегрована в іншу систему. Розроблено швидкий та зручний інтерфейс програми.

5. Стартап

5.1. Опис ідеї проекту

Постала потреба створити автоматичну систему генерування тестових завдань, на меті якої є перевірка знань спеціалістів IT сфери без участі викладачів, менторів.

Таблиця 5.1. Опис ідеї проекту та його застосування

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Перевірка знань спеціалістів IT сфери без участі викладачів, Менторів	Перевірка знань студентів	Викладачу не потрібно створювати самому створювати тести та перевіряти їх
	Перевірка знань IT спеціалістів	HR може покласти частину перевірки знань на тести
	Перевірка компетенції IT спеціалістів	Рекомендації сервісу щодо підвищення рівня на кар’єрних сходинках робітника

Проведемо аналіз потенційних техніко-економічних переваг ідеї (чим відрізняється від існуючих аналогів та замінників) порівняно з пропозиціями конкурентів передбачає:

- визначення переліку техніко-економічних властивостей та характеристик ідеї;
- визначення попереднього кола конкурентів (проектів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та

проводиться збір інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку;

- проводиться порівняльний аналіз показників: для власної ідеї визначаються показники, що мають а) гірші значення (W, слабкі); б) аналогічні (N, нейтральні) значення; в) кращі значення (S, сильні) (таблиця 5.2).

Таблиця 5.2. Характеристики проекту

No п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/ концепції конкурентів			W	N	S
		Мій проект	Конкурент1	Конкурент2			
1	Економічні	-Вартість обслуговування 1000$ -Знижки 15$ -Вартість ліцензії 50$	-Вартість обслуговуванння 1200$ -Знижки 10$ -Вартість ліцензії 40$	-Вартість обслуговуванння 1500$ -Знижки 20$ -Вартість ліцензії 35$	3	2	1
2	Надійності	Висока	Низька	Середня			1
3	Технологічні	Середня	Висока	Низька		1	
4	Естетичні	Висока	Середня	Висока			1

5.2. Технологічний аудит ідеї проекту

В межах даного підрозділу необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару). Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (таблиця 5.3):

- за якою технологією буде виготовлено товар згідно ідеї проекту;
- чи існують такі технології, чи їх потрібно розробити/доробити;

- чи доступні такі технології авторам проекту.

Таблиця 5.3. Технологічний аудит

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Автоматичне створення тестів	Використання ПТМ	наявна	доступна
2	Можливість використання ШІ	Мова програмування Python	наявна	доступна
3	Створення рекомендаційної системи	Мова програмування Python	наявна	доступна
4	Розгортка серверу	Hetzner	наявна	доступна

5.3. Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту з урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Проведемо аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (таблиця 5.4). Фахівці стверджують, що нехтувати аналізом ринку на етапі започаткування бізнесу ні в якому разі не можна. Тому якщо ви хочете залучити інвестора, на етапі складання бізнес-плану основну увагу приділяйте вивченню ринку. Дані по ринку повинні бути актуальними, інформативними і точними.

Таблиця 5.4. Аналіз попиту

№ п/п	Показник стану ринку(найменування)	Характеристика
1	Кількість головних гравців, од	5
2	Загальний обсяг продаж, грн/ум.од	100 000$
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	40%

Визначимо потенційні групи клієнтів, їх характеристики, та сформуємо орієнтований перелік вимог до товару для кожної групи (таблиця 5.5).

Таблиця 5.5. Аналіз аудиторії

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Потреба в автоматизованій перевірці знань	Студенти, навчальні заклади, ІТ-компанії	Ціна для студентів та ІТ-компаній буде різною	• Базова перевірка знань • Рекомендації щодо кваліфікації/рівня знань

Поверхневе розуміння ринку, на якому планується робота, або ж відсутність

інформації про конкурентні переваги, яка з'являється тільки після вивчення потреб ринку і конкурентів, експерти відносять до однієї з найпоширеніших помилок в презентаціях українських стартапів перед інвесторами. Проведемо аналіз ринкового середовища: складемо таблиці факторів, що сприяють ринковому впровадженню проекту (таблиця 5.6), та факторів, що йому перешкоджають (таблиця 5.7).

Чим більш інноваційним є проект, тим більше питань виникає у потенційних інвесторів, так і самому автору стартапу важливо тверезо оцінювати його перспективи. Давайте подивимось, як оцінити інвестиційну привабливість стартапу.

Таблиця 5.6. Аналіз факторів сприяння

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Зробити тести автоматичними	Дозволить скоротити час на підготовку навчальних матеріалів	Необхідно виділити більше ресурсів для розробки
2	Модифікації тестів на інші сфери діяльності	Дозволить збільшити цільову аудиторію	Вихід на більш крупний ринок та можливо розгортання окремого стартапу
3	Якість	Дозволить залучити деяку кількість аудиторії конкурентів	Дозволить перетворити конкуренцію на монополістичну
4	Сертифікації системи	Зробити систему стандартом перевірки знань в деяких країнах	Необхідність отримання патенту

Існують традиційні методи оцінки компанії або бізнесу. Як правило, застосовується один з цих підходів:

- ринковий підхід – вартість оцінюється в порівнянні з іншими аналогічними компаніями, які недавно були продані (їх вартість визначена ціною продажу);
- дохідний підхід – він заснований на підрахунку очікуваних доходів від бізнесу;
- підхід, який базується на оцінці активів. В рамках цього підходу вартість бізнесу = вартість активів мінус вартість боргів (зобов'язань).

Таблиця 5.7. Аналіз факторів сприяння

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Час	Витратити багато часу на побудову алгоритму автоматичного виділення понять, тез та їх класифікацію	Необхідно провести гарний аналіз існуючих рішень
2	Технологія	Довгій вибір технології автоматичного класифікування понять та тез	Залучення талановитих працівників
3	Якість	Отримання не дуже якісних тестів	Зосередження на удосконаленні алгоритмів
4	Ринок	Розробки більш універсальних шаблонів для тестування конкурентами	Необхідно запатентувати данну технологію

Проводемо аналіз пропозиції: визначимо загальні риси конкуренції на ринку (таблиця 5.8). Найважливішим аспектом аналізу конкуренції є виявлення

конкурентів, які працюють на ринку. У разі стартапів це часто досить складне завдання, оскільки цей тип починань зазвичай характеризується унікальністю і новаторством.

Таблиця 5.8. Аналіз пропозиції

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Тип конкуренції	Монополістична конкуренція (Конкуренція на рівні декількох великих компаній)	Монополія дозволить трохи завищити ціну
Рівень конкурентної боротьби	Національна (Конкуренція ведеться на рівні усіх країн світу)	Необхідно докладати зусиль для охоплення всього національного ринку.
Галузева ознака	Внутрішньогалузева (Конкуренція ведеться лише в галузі систем тестування)	Необхідно зосередити зусилля на пошуку конкурентних переваг
Конкуренція за видами товарів	Товарно-видова (Конкуренція лише з іншими системами тестування)	Конкуренція між іншими системами тестування
Характер конкурентних переваг	Не цінова (Ставка робиться на якість товару, а не на ціну)	Головною конкурентною перевагою є якість та швидкість системи
Інтенсивність	Марочна	Диференціація глазурованих сирків за мотивом задоволення.

Однак ви повинні розуміти, що конкурентом є не тільки компанія, що пропонує точно такий же продукт або послугу, але і той, хто по-різному задовольняє аналогічні потреби клієнтів. При тестуванні необхідно враховувати не тільки прямих конкурентів, але і непрямих. Більш того, якщо в даний час таких компаній немає, слід припустити, що вони можуть з'явитися в майбутньому.

Проаналізуємо конкуренцію в галузі за моделлю М. Портера (таблиця 5.9).

Таблиця 5.9. Аналіз за моделлю Портера

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Клієнти	Товари-замінники
	iSpring, Indigo	Let's test, Testograf	Клієнтам необхідна система автоматичного контролю знань, при цьому основна увага приділяється якості тестів	Товарів замінників багато, але прямих не існує
Висновки:	Прямих конкурентів мало, конкуренція не дуже інтенсивна	Є усі можливості входу на ринок, строк виходу невеликий	Клієнти диктують умови по якості завдань	Обмеження для роботи на ринку незначні

На основі аналізу конкуренції, проведеного (таблиця 5.9), а також з урахуванням характеристик ідеї проекту (таблиця 5.2), вимог споживачів до товару (таблиця 5.5) та факторів маркетингового середовища (таблиця 5.6-5.7) визначимо та обґрунтовуємо перелік факторів конкурентоспроможності (таблиця 5.10).

Таблиця 5.10. Фактори конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування
1	Якість завдань	Система створює якісні завдання порівняно з іншими рішеннями
2	Рекомендаційна система	Рекомендації щодо необхідності повторення певних тем у найвищій точності
3	Легка інтеграція	Можливість підключення системи як частину до іншої
4	Невеликі часові витрати для автоматичної генерації тестів	Тести генеруються автоматично на відміну від інших рішень

Підіб’ємо підсумки ринкових можливостей за допомогою SWOT аналізу (рисунок 5.1) Метод swot аналізу передбачає, що ви будете збирати дані, а потім аналізувати їх з урахуванням 4 моментів.

- S - сили (Strengths);
- W - слабкості (Weaknesses);
- O - можливості (Opportunities);
- T - загрози (Threats).

Swot аналіз організації розділяє наведені вище категорії на 2 групи: внутрішні чинники і зовнішні чинники впливу. До внутрішніх відносяться сили і слабкості, тобто те, що лежить в області піддається зміні з боку стратегів бізнесу. До зовнішніх – можливості і загрози, які підносить бізнесу «навколишнє середовище», в якій він діє.

Рисунок 5.1. SWOT аналіз

5.4. Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку (таблиця 5.11)

Табл 5.11. Ринкова стратегія стартап-проекту

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Студенти	Висока	Низький	Низька	Висока
2	Навчальні заклади	Середня	Середній	Низька	Середня
3	IT-компанії	Середня	Високий	Середня	Низька
Можна обрати усі три групи, оскільки кожна з них має свої недоліки, проте й переваги					

Ринкова стратегія також може бути оцінена шляхом розробки бізнес моделі

стартап-проекту (рисунок 5.2). За більш складною моделлю працюють великі інформаційні портали – їм видається безліч варіантів монетизації (починаючи продажем рекламного інвентарю, закінчуючи написанням замовних матеріалів), що вимагає аналізу джерел прибутку і їх оптимізації з метою підвищення доходу.

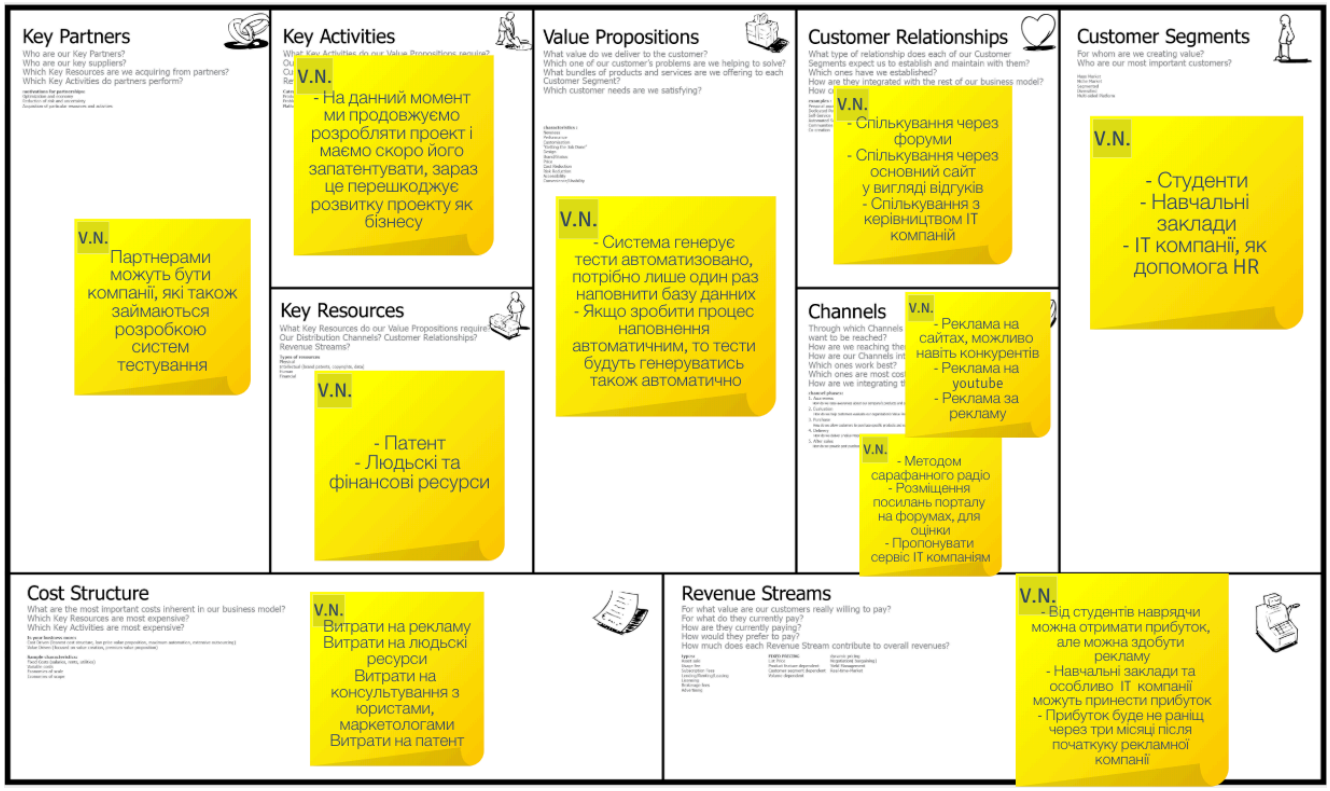

Рисунок 5.2. Бізнес модель стартап-проекту

5.5. Висновки

В даному розділі було подано опис ідеї стартап проекту автоматизованої системи контролю знань на онтологічно-орієнтованому порталі з програмування, його технологічний аудит. Здійснено аналіз ринкових можливостей запуску системи, для цього розроблена ринкова стратегія проекту, яка включає в себе аналіз конкурентів, можливих партнерів, цільову аудиторію. Описано слабкі та сильні сторони завдяки SWOT-аналізу. Розроблено бізнес модель розвитку стартапу.

6. ВИСНОВКИ

У даній роботі було проаналізовано методи автоматизованої побудови тестових завдань з перевагами та недоліками кожного з них. Розглянуто концепції, які необхідні для забезпечення валідності та якості тестів, а також описано один з варіантів їх калібрування. Запропоновано модифікацію методу автоматизованої побудови тестів на базі понятійно-тезисної моделі, що забезпечило диференціацію контролю за рівнем складності із застосуванням перевірки досягнення когнітивних цілей відповідно до модифікованої шкали Блума. Введено нові типи завдань, модифіковано структуру шаблонів тестових завдань та подано формальну модель генерації тесту з урахуванням рівня складності. Здійснено програмну реалізацію запропонованого формального апарату та проведено пілотне тестування, що підтвердило адекватність та перспективність здійснених модифікацій. Реалізовано рекомендаційний функціонал, що сприяє більш швидкому процесу навчання для студентів. Загалом можна казати про успіх даної модернізації ПТМ. Серед напрямків подальших досліджень – детальна формалізація розв’язання задачі достатності даних для генерації тесту, підвищення якості тестових завдань різних типів та побудова нових шаблонів завдань. Особливої уваги потребує забезпечення додаткових засобів перевірки досягнення когнітивних цілей відповідно до всієї шкали Блума [7]. Було здійснено програмну реалізацію системи, з описом кожного компоненту програми з різних точок зору. Описано зв’язки між кожною структурною одиницею програми. Розроблено адаптивну архітектуру завдяки модульності, що забезпечило високу надійність системи. Інтерфейс програми реалізований таким чином, що є зрозумілим користувачу та має швидкий відгук. Створено модель розвитку системи як стартап-проекту з описом переваг та недоліків, сфер застосування та перспективами подальшого розвитку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Аванесов В.С. Композиция тестовых заданий. Учебная книга для преподавателей вузов, учителей школ, аспирантов и студентов пед. вузов. 2 изд.,испр.. и доп.М.:Адепт.1998-217с.

2. Титенко, С. В. Генерація тестових завдань у системі дистанційного навчання на основі моделі формалізації дидактичного тексту / С. В. Титенко // Наукові вісті НТУУ "КПІ". – 2009. – No 1(63). – С. 47–57.

3. Титенко С.В., Гагарін О.О. Практична реалізація технології автоматизації тестування на основі понятійно-тезисної моделі. Образование и виртуальность – 2006. Сборник научных трудов 10-й Международной конференции Украинской ассоциации дистанционного образования / Под общ. ред. В.А. Гребенюка, Др Киншука, В.В. Семенца.– Харьков-Ялта: УАДО, 2006.– С. 401-412.

4. Петрова Л. Г. Використання модифікованої понятійно-тезисної моделі для автоматизованого формування бази тестових запитань в системах комп’ютеризації освіти / Л. Г. Петрова, С. А. Петров // Інформаційні технології і засоби навчання. - 2012. - № 4 (30). - С. 1-13.

5. Мельник А.М. Метод генерації тестових завдань на основі системи семантичних класів /Мельник А.М., Пасічник Р.М. // Вісник ТДТУ. — 2010. — Том 15. — № 1. — С. 187-193

6. Танченко С. С. Усунення мовної неузгодженості в тестових завданнях, згенерованих на основі понятійно-тезисної моделі/ С. С. Танченко, С. В. Титенко, О. О. Гагарін// XII международная научная конференция имени Т. А. Таран «Интеллектуальный анализ информации ИАИ-2014», Киев, 14-16 мая 2014 г. : сб. тр./ гл. ред. С.В.Сирота. – К. : Просвіта, 2014. – С. 196-200.

7. Krathwohl, D. A revision of Bloom's taxonomy: an overview, Theory into Practice, 41(4). - 2002.- 212 - 218.

8. Is this a trick question?: A short guide to writing effective test questions / B.

Clay, E. Root. — Kansas Curriculum Center, 2001.

9. Nurgabyl D. Construction of a mathematical model for calibrating test task parameters and the knowledge level scale of university students by means of testing / D. Nurgabyl, G. Kalzhanova, N. Ualiyev, G. Abdoldinova // Eurasia Journal of Mathematics, Science and Technology Education. — 2017. — Vol. 13, No. 11. — P. 7421–7429.

10. Brusilovsky P. and Miller P. Web-based testing for distance education // Proceedings of WebNet'99, World Conference of the WWW and Internet, Honolulu, HI, Oct. 24—30, 1999, AACE / P. De Bra and J. Leggett (eds.). — P. 149—154.

11. Pathak S., Brusilovsky P. Assessing Student Programming Knowledge with Webbased Dynamic Parameterized Quizzes // ED-MEDIA'2002 — World Conference on Educational Multimedia, Hypermedia and Telecommunications, Denver, CO, June 24—29, 2002 / Barker P. and Rebelsky S. (eds.). — P. 1548—1553.

12. Sosnovsky S. Web-based Parameterized Questions as a Tool for Learning / S. Sosnovsky, O. Shcherbinina, P. Brusilovsky // Proceedings of E-Learn 2003, Arizona USA. — 2003. — [p. 2151-2154].

13. Левинская М.А. Автоматизированная генерация заданий по математике для контроля знаний учащихся [Электронный ресурс] // Educational Technology & Society. — 2002. — 5(4). — [С. 214-221] . — http://ifets.ieee.org/russian/depository/ v5_i4/html/3.html.

14. Кручинин В. В. Модели и алгоритмы генерации задач в компьютерном тестировании [Электронный ресурс] / В. В. Кручинин, Ю. В. Морозова // Известия Томского политехнического университета [Известия ТПУ] / Томский политехнический университет (ТПУ) . — 2004 . — Т. 307, № 5 . — [С. 127-131] . — Заглавие с титульного листа. — Электронная версия печатной публикации.

15. Кручинин В. В. Методы генерации тестовых заданий по информатике / В.

В. Кручинин // «Информатика и образование». – 2005 . – № 2.

16. Кручинин В.В. Методы построения алгоритмов генерации и нумерации комбинаторных объектов на основе деревьев И/ИЛИ// Томск: «В-Спектр», 2007. – 200 с.

17. Зорин Ю. А, Кручинин В.В. Система генерации тестовых заданий на основе деревьев И/ИЛИ [Электронный ресурс] — Томск., 2012. — Т. 2. — [С. 313-314].

18. Елизаренко Г.Н. Проектирование компьютерных курсов обучения: концепция, язык, структура. — К.: НТУУ “КПИ”, 2001.

19. Stankov S., *Žitko* B. and Grubi*šič* A. Ontology as a Foundation for Knowledge Evaluation in Intelligent E-learning Systems // AIED’05 Workshop SW-EL’05: Applications of Semantic Web Technologies for E-Learning. Papers of 12th International Conference on Artificial Intelligence in Education (AIED 2005). — Amsterdam, 2005. — http://hcs. science.uva.nl/AIED2005/W3proc.pdf

20. Berners-Lee T. Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. — The MI Press, 2005.

21. Гагарін О.О., Гайдаржи В.І., Титенко С.В. Концептуальний підхід до подання знань в інтелектуальній освітній системі // Сучасні тенденції розвитку інформаційних технологій в науці, освіті та економіці: Матер. Всеукр. наук.-практич. конф., 11—13 грудня 2006 р., м. Луганськ. — Луганськ: Альма-матер, 2006. — С. 17—19.

22. Артур К. Как изучать Библию. — СПб., 1998.

23. Gordon D. Fee, Douglas Stuart. How to Read the Bible For All its Worth: A Guide to Understanding the Bible. — Micchigan, 1982.

24. Gagarin A., Tytenko S. Complex model of educational hypermedia environment for ongoing learning // Образование и виртуальность-2007: Сб. науч. тр. 11-й Междунар. конф. Укр. ассоциации дистанционного образования / Под общ. ред. В.А. Гребенюка, Др. Киншука и В.В. Семенца. — Харьков—Ялта: УАДО, 2007. — С. 140—145.

25. Гагарін О.О., Титенко С.В. Проблеми створення гіпертекстового навчаючого середовища // Вісн. Схід- ноукр. нац. ун-ту ім. Володимира Даля. — 2007. — Ч. 2, No 4(110). — С. 6—15.

26. Гагарин А.А., Луценко А.Н., Титенко С.В. Организация дистанционного обучения как информационный фактор реализации научно-технологической составляющей экономической безопасности государства // Экономическая безопасность государства и информационные технологии в ее обеспечении / Под общ. ред. Г.К. Вороновского, И.В. Недина. — К.: Знания Украины, 2005. — С. 608—619.

27. Титенко С.В., Гагарін О.О. Семантична модель знань для цілей організації контролю знань у навчальній системі // Сб. тр. междунар. конф. “Интеллектуальный анализ информации-2006”. — К.: Просвіта, 2006. — С. 298—307.

28. Портал знань — застосування ПТМ-підходів для автоматизації Web-тестування. — http://www.znannya.org

29. Лабораторія СЕТ — віртуальна лабораторія новітніх інформаційних технологій. Дослідження в області дистанційного навчання. — http://www.setlab.net

30. Аванесов В. С. Основные элементы заданий в тестовой форме с двумя ответами. "Управление школой" №10, март, 2000г. http://testolog.narod.ru/Theory15.html

Додаток А

Система автоматизованого контролю знань на онтологічно-орієнтованому порталі з програмування

Копії публікацій

УКР.НТУУ “КПІ ім. Ігоря Сікорського”.ТІ41155_18М

Аркушів 1

Київ 2019

Ministry of Education and Science of Ukraine

National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
Heat and Power Engineering Faculty

Glushkov Institute of Cybernetics, National Academy of Sciences of Ukraine

Polytechnic Institute of Tomar (Portugal)

Smart Cities Research Center (Portugal)

MODERN ASPECTS OF SOFTWARE DEVELOPMENT

Proceedings of
VI International Scientific and Practical Virtual Conference of
Software Development Specialists

June, 24, 2019

Kyiv, Ukraine

Modern Aspects of Software Development: Proceedings of VI International Scientific and Practical Virtual Conference of Software Development Specialists, June, 24 2019 p. – Kyiv: Igor Sikorsky KPI, 2019. – 189 p.

The Proceedings of VI International Scientific and Practical Virtual Conference of Software Development Specialists "Modern Aspects of Software Development" consists of the results of scientific researches and practical developments in the following directions: software engineering, computer ecological-economic monitoring, computerized design

Text is printed in the authors' edition.

ISBN

CONTENT