

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

ДЛЯ ОРГАНІЗАЦІЇ ТА УПРАВЛІННЯ ТЕСТУВАННЯМ ЗНАНЬ

SOFTWARE FOR KNOWLEDGE

ASSESSMENT AND MANAGEMENT

Виконав(ла): студент(ка) 4 курсу, групи КНТ-112

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

КОЗА Д.А.

(ПРІЗВИЩЕ та ініціали)

Керівник СУББОТІН С.О.

(ПРІЗВИЩЕ та ініціали)

Рецензент КОРОТКИЙ О.В.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

КОЗИ Данила Артемовича

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для організації та управління тестуванням знань. Software for Knowledge Assessment and Management

керівник проєкту (роботи) д.т.н., професор, СУББОТІН Сергій Олександрович,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 07 ” квітня 2026 року № 139

2. Строк подання студентом проєкту (роботи) 10 червня 2026 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) _____

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СУББОТІН С.О., завідувач кафедри		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання “20” квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Данило КОЗА
(Імя ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Сергій СУББОТІН
(Імя ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
97 с., 3 табл., 31 рис., 3 дод., 18 джерел.

INTELLIJ IDEA, KOTLIN, ЗАСТОСУНОК, МОВА
ПРОГРАМУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ТЕСТУВАННЯ.

Об'єкт дослідження – процес розробки програмного забезпечення для організації та управління тестуванням.

Предмет дослідження – програмні засоби для організації та управління тестуванням.

Мета роботи – розробка програмного забезпечення для організації та управління тестуванням знань, що забезпечує зменшення трудомісткості процесу організації тестування.

Матеріали, методи та технічні засоби: мова програмування Kotlin, середовище розробки IntelliJ IDEA.

Результати. Розроблено проєктні рішення для створення програмного забезпечення для організації та управління тестуванням. Створено програмне забезпечення для організації та управління тестуванням. Проведено тестування розробленої програмної системи для організації та управління тестуванням.

Висновки. Мету роботи досягнуто. Розроблено програмне забезпечення для організації та управління тестуванням за допомогою мови програмування Kotlin та середовища розробки IntelliJ IDEA.

Галузь використання – освітні заклади та організації.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 97 pages, 3 tables, 31 figures, 3 appendixes, 18 sources.

INTELLIJ IDEA, KOTLIN, APPLICATION, PROGRAMMING LANGUAGE, SOFTWARE, TESTING.

The object of research is the process of developing software for organizing and managing testing.

The subject of the research is software for organizing and managing testing.

The purpose of this work is to develop the software for organizing and managing testing, which ensures a reduction in the labor intensity of the testing organization process.

Materials, methods and technical tools: Kotlin programming language, IntelliJ IDEA development environment.

Results. Project solutions for the creation of software for organizing and managing testing has been designed. The software for organizing and managing testing has been developed. The developed software system for organizing and managing testing has been tested.

Conclusions. The goal of the work has been achieved. The software tools for organizing and managing testing using the Kotlin programming language and the IntelliJ IDEA development environment have been developed.

Scope of use – educational institutions and organizations.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Програмне забезпечення для організації та управління тестуванням	11
1.2 Аналіз програмного забезпечення для організації та управління тестуванням.....	20
1.3 Висновки за розділом 1	32
2 Матеріали і методи	33
2.1 Вибір мови програмування	33
2.2 Вибір середовища розробки для створення програмного забезпечення для організації та управління тестуванням	37
2.3 Висновки за розділом 2	40
3 Опис програми	41
3.1 Структура програмного забезпечення для організації та управління тестуванням.....	41
3.2 Функціонування програмного забезпечення для організації та управління тестуванням.....	44
3.3 Розробка бази даних програмного забезпечення для організації та управління тестуванням.....	47
3.4 Проєктування інтерфейсу програмного забезпечення для організації та управління тестуванням	50
3.5 Висновки за розділом 3	53
4 Експлуатація, тестування та експериментальне дослідження програми.....	55
4.1 Призначення й умови застосування програми	55
4.2 Характеристики програми для організації та управління тестуванням..	57
4.3 Інструкція по експлуатації програми.....	59
4.3.1 Звернення до програми.....	59

4.3.2 Вхідні й вихідні дані.....	59
4.3.3 Повідомлення	60
4.4 Виконання програмного забезпечення для організації та управління тестуванням.....	60
4.5 Тестування програмного забезпечення для організації та управління тестуванням.....	66
4.6 Висновки за розділом 4	67
Висновки	68
Перелік джерел посилання	71
Додаток А Технічне завдання	73
Додаток Б Фрагмент тексту програми	79
Додаток В Слайди презентації	91

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AI – Artificial Intelligence;

IDE – Integrated Development Environment;

JVM – Java Virtual Machine;

БД – база даних;

ПЗ – програмне забезпечення;

СУБД – система управління базами даних.

ВСТУП

Актуальність програмного забезпечення для організації та управління тестуванням зумовлюється зростанням складності освітніх і професійних процесів оцінювання знань, навичок і компетентностей. В умовах цифрової трансформації суспільства підвищуються вимоги до об'єктивності, прозорості та оперативності контролю результатів навчання або професійної підготовки. Традиційні форми перевірки знань дедалі частіше виявляються недостатньо гнучкими, трудомісткими та залежними від людського чинника, що спричиняє потребу в автоматизованих інструментах, здатних забезпечити стандартизацію процедур тестування та мінімізувати ймовірність помилок [1]-[4].

Необхідність упровадження спеціалізованого програмного забезпечення для організації та управління тестуванням обумовлюється також масштабуванням освітніх платформ і корпоративних систем навчання, де одночасно може проходити оцінювання велика кількість користувачів. У таких умовах виникає потреба в централізованому управлінні банками тестових завдань, налаштуванні різних сценаріїв проведення тестування, автоматичному формуванні варіантів завдань та миттєвому обчисленні результатів [4]-[5].

Важливою складовою актуальності є забезпечення академічної доброчесності та захисту даних. Використання сучасних механізмів автентифікації, випадкового перемішування завдань, обмеження часу виконання та фіксації активності користувачів сприяє зниженню ризику порушень під час проходження тестів. Одночасно реалізуються засоби збереження конфіденційності персональної інформації, що відповідає вимогам нормативно-правового регулювання у сфері обробки даних [5]-[6].

Отже програмне забезпечення для організації та управління тестуванням допомагає сформувати більш об'єктивну та аналітично обґрунтовану систему контролю, яка відповідає сучасним вимогам цифрового середовища та сприяє підвищенню ефективності освітніх і професійних процесів [1]-[6].

Проте впровадження деяких програм для організації та управління тестуванням пов'язано з високими витратами на ліцензування, складною інтеграцією з іншими програмними платформами, навчанням персоналу та адміністрації. За відсутності достатнього рівня підготовки користувачів знижується ефективність використання функціональних можливостей таких програмних систем, особливо у випадках, коли такі системи є досить складними, що призводить до формального застосування інструментів без повної реалізації їх потенціалу. Не менш важливим є питання захисту даних і забезпечення конфіденційності. У разі недостатнього рівня захисту існує ймовірність несанкціонованого доступу до результатів тестування або персональної інформації користувачів. Крім того, автоматизований контроль не повністю усуває можливість недоброчесної поведінки здобувачів освіти або інших учасників освітнього процесу, оскільки зберігається ризик використання сторонніх ресурсів чи технічних засобів під час проходження тестів [1]-[6].

Тому актуальною є розробка програмного забезпечення для організації та управління тестуванням. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки програмного забезпечення для організації та управління тестуванням.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для організації та управління тестуванням;
- здійснити проектування програмного забезпечення для організації та управління тестуванням;
- створити програмне забезпечення для організації та управління тестуванням;
- виконати тестування розробленого програмного забезпечення для організації та управління тестуванням.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення для організації та управління тестуванням

Програмне забезпечення для організації та управління тестуванням являє собою спеціалізований програмний інструмент, призначений для планування, проведення, контролю та аналізу процедур оцінювання знань, умінь або професійних компетентностей. Такі програмні системи створюють надають інформаційне середовище, у межах якого забезпечується підготовка тестових матеріалів, адміністрування доступу учасників, автоматизована перевірка результатів і формування аналітичних висновків. У результаті формується цілісний підхід до оцінювання, що поєднує методичну складову з технологічною підтримкою [1].

Функціонування подібних програмних рішень передбачає централізоване зберігання завдань і результатів, що сприяє впорядкованості процесів та спрощує управління великими обсягами інформації. Забезпечується можливість гнучкого налаштування параметрів тестування відповідно до освітніх або корпоративних потреб. Підтримується створення різних форматів запитань, що дозволяє адаптувати інструменти оцінювання до специфіки навчальних програм чи професійних стандартів [1].

Важливою характеристикою таких програмних систем є автоматизація перевірки відповідей і формування звітності. Це зменшує часові витрати на обробку результатів і підвищує об'єктивність оцінювання. Аналітичні модулі сприяють виявленню тенденцій, визначенню рівня засвоєння матеріалу та прийняттю управлінських рішень щодо вдосконалення навчального процесу. Забезпечується прозорість процедур і можливість простеження динаміки розвитку окремих учасників або груп [1]-[2].

Окрема увага приділяється безпеці та контролю доступу. Реалізуються механізми автентифікації, обмеження несанкціонованого використання

ресурсів і захисту персональних даних. Такий підхід дозволяє підтримувати довіру до результатів оцінювання та зберігати цілісність інформації [1]-[2].

Серед переваг подібних програмних систем відзначається підвищення ефективності організаційних процесів, скорочення паперового документообігу, зручність адміністрування та можливість дистанційної взаємодії. Створюються умови для масштабування оцінювання незалежно від кількості учасників, а також для інтеграції з іншими інформаційними ресурсами. У підсумку програмне забезпечення для організації та управління тестуванням виступає інструментом цифрової трансформації оцінювальної діяльності, поєднуючи технологічну гнучкість із методичною доцільністю [1]-[2].

У період активної цифровізації освітнього середовища Інтернет сервіси для перевірки знань набувають дедалі більшого поширення серед здобувачів освіти. Такі рішення сприймаються як зручний спосіб контролю рівня підготовки та самостійного вдосконалення навичок. Вони поєднують інструменти для створення тестових завдань, моніторингу результатів і аналізу навчальної динаміки, що дозволяє організувати процес оцінювання більш гнучко та технологічно. Незалежно від формату підсумкової перевірки або мети самопідготовки, цифрові платформи забезпечують середовище, у якому навчання поєднується з інтерактивною взаємодією та постійним зворотним зв'язком [1]-[4].

Подібні програмні системи активно впроваджуються закладами освіти, тренінговими центрами та індивідуальними викладачами для вдосконалення процедур контролю знань. Завдяки різноманіттю функціональних можливостей створюються умови для оптимізації оцінювальних процесів і підвищення якості освітнього досвіду. Ринок таких рішень характеризується широким вибором інструментів, орієнтованих як на масові стандартизовані перевірки, так і на спеціалізовані дисципліни або окремі теми. Користувацькі інтерфейси зазвичай розробляються з урахуванням принципів зручності та

інтуїтивності, що полегшує освоєння сервісів і сприяє регулярному використанню [1]-[4].

Перспективи розвитку цього напрямку пов'язані з подальшою інтеграцією інтелектуальних технологій. Очікується розширення можливостей персоналізації навчального процесу, коли алгоритми зможуть адаптувати зміст завдань до індивідуального рівня підготовки. Використання методів аналізу даних і машинного навчання відкриває шлях до автоматичного виявлення прогалин у знаннях та формування рекомендацій для їх усунення. Таким чином створюється основа для більш адресного підходу до навчання, де оцінювання не лише фіксує результат, а й спрямовує подальший розвиток [1]-[4].

Переваги застосування програмних платформ для тестування пов'язані насамперед із доступністю та мобільністю. Проходження перевірок можливе в будь-якому місці за наявності мережевого підключення, що дозволяє інтегрувати підготовку в індивідуальний графік. Значущою перевагою виступає оперативне отримання результатів, завдяки чому одразу визначаються сильні сторони та аспекти, що потребують додаткової уваги. Такий формат скорочує часовий розрив між виконанням завдань і аналізом помилок [1]-[4].

Додаткову цінність становить доступ до великої кількості навчальних матеріалів, які допомагають ознайомитися з різними типами запитань і відпрацювати стратегії виконання тестів. Розширена аналітика дозволяє відстежувати зміни в успішності, аналізувати тенденції та приймати обґрунтовані рішення щодо коригування навчальних підходів. Інтерактивні елементи, мультимедійний контент і симуляційні моделі підсилюють залученість, роблячи освітній процес більш динамічним і мотивуючим. У результаті онлайн платформи для тестування формують сучасний інструментарій, що поєднує контроль знань із підтримкою безперервного навчального розвитку [1]-[4].

У сучасному освітньому середовищі цифрові сервіси для онлайн тестування стали важливим інструментом, який змінює підходи до підготовки студентів і контролю знань. Такі системи надають можливості для адаптивного навчання, оцінювання результатів та відстеження прогресу, забезпечуючи інтерактивність та залучення учнів у процес. Вони сприяють не лише перевірці поточного рівня знань, а й формуванню індивідуальних навчальних траєкторій та розвитку навичок самостійного навчання. Завдяки цим платформам студенти можуть отримувати відразу структуровану інформацію про свої досягнення та слабкі місця, що дозволяє коригувати підготовку та стратегічно планувати навчальний процес [1]-[4].

Програмні платформи для тестування відкривають рівний доступ до навчальних ресурсів, незалежно від географічного розташування чи соціально-економічного статусу користувачів. Віддалені райони та менш забезпечені громади отримують такі ж можливості для практики та оцінки знань, як і користувачі з великих міст або привілейованих навчальних закладів. Використання алгоритмів машинного навчання та штучного інтелекту дозволяє адаптувати тестові завдання до рівня підготовки кожного студента, підбираючи навчальні матеріали та практичні вправи відповідно до індивідуальних потреб. Платформи формують велику кількість даних про успішність, навчальні патерни та прогрес студентів, що дозволяє педагогам та адміністраторам аналізувати ефективність навчальних програм та проводити цілеспрямовані втручання для підвищення результатів [1]-[4].

Безперервна оцінка знань забезпечує регулярний контроль прогресу, дозволяючи використовувати не тільки підсумкові іспити, а й серії малих тестів, які дають постійний зворотний зв'язок. Це сприяє підтримці навчальних цілей, підвищує мотивацію та допомагає студентам відстежувати власні успіхи в режимі реального часу. Використання онлайн платформ також оптимізує розподіл ресурсів навчальних закладів, виявляючи сильні та слабкі сторони програм, що дозволяє ефективніше планувати методи навчання та покращувати загальну академічну успішність [3]-[6].

Щоб максимально використовувати потенціал таких програмних платформ, рекомендується планувати навчальні сесії заздалегідь, організовуючи час для практичних тестів та оцінювання. Варто доповнювати платформу іншими навчальними матеріалами, такими як підручники або онлайн посібники, для глибшого розуміння предмета. Аналітичні інструменти платформ допомагають виявляти проблемні теми та зосереджувати зусилля на них, коригуючи навчальний план. Імітація реальних умов іспиту під час тренувань допомагає адаптуватися до формату перевірки знань і розвивати витривалість до виконання завдань. Зворотний зв'язок від платформ дає можливість аналізувати помилки, розуміти логіку правильних відповідей і вдосконалювати стратегії підготовки для підвищення результативності у навчанні [3]-[6].

В сучасному освітньому процесі цифрові сервіси для онлайн тестування стали невід'ємною складовою підготовки студентів до іспитів і контролю знань. Такі платформи пропонують зручний та гнучкий інструментарій для практики та оцінки навчальних досягнень, забезпечуючи інтерактивність і залучення учнів у навчальний процес. Вони дозволяють оперативно отримувати зворотний зв'язок, виявляти прогалини в знаннях та коригувати підхід до навчання. За допомогою таких систем можливо відстежувати прогрес, порівнювати результати з минулими спробами та планувати подальше навчання з урахуванням індивідуальних потреб [3]-[6].

Платформи онлайн тестування відкривають доступ до навчальних ресурсів незалежно від місця проживання чи соціально-економічного стану користувачів, забезпечуючи рівні можливості для всіх студентів. Використання алгоритмів штучного інтелекту дозволяє підбирати матеріали та завдання відповідно до рівня знань та навчального темпу конкретного учня, надаючи персоналізований підхід до підготовки. Дані, що генеруються платформами, дозволяють освітянам оцінювати ефективність навчальних програм, виявляти слабкі сторони та впроваджувати цілеспрямовані втручання для покращення результатів студентів [3]-[6].

Регулярне проведення оцінювання забезпечує безперервний контроль знань, що знижує залежність від великих підсумкових іспитів і створює можливості для частого отримання зворотного зв'язку. Ці інструменти допомагають оптимізувати навчальний процес, підвищують мотивацію та дозволяють студентам самостійно відстежувати свій прогрес. Платформи також сприяють раціональному використанню ресурсів навчальних закладів, допомагаючи визначати проблемні області програми та ефективніше розподіляти увагу та матеріали для покращення загальної академічної успішності [3]-[6].

Щоб ефективно використовувати можливості онлайн платформ, рекомендується планувати час для практичних тестів і тренувальних іспитів, включати різноманітні навчальні ресурси, аналізувати дані аналітики для визначення слабких місць і зосередження зусиль, відтворювати реальні умови іспиту та користуватися зворотним зв'язком для виправлення помилок і вдосконалення стратегій навчання [3]-[6].

Водночас подібні програмні платформи мають певні обмеження. Вони можуть стикатися з технічними перебоями або проблемами сумісності, вимагати стабільного підключення до Інтернету, мати обмежені заходи безпеки та недостатню особисту взаємодію з викладачем. Деякі платформи потребують додаткових витрат або підписки, що може створювати фінансові бар'єри [3]-[6].

Застосування штучного інтелекту на таких платформах відкриває нові можливості: автоматичний аналіз успішності учнів дозволяє визначати прогалини та надавати персоналізовані рекомендації, адаптивні алгоритми підбирають рівень складності завдань відповідно до прогресу, а чат-боти та інші AI-інструменти забезпечують допомогу й підтримку протягом навчання. Методи машинного навчання, обробки природної мови та розпізнавання зображень дозволяють автоматизувати оцінювання письмових та графічних завдань, економлячи час і підвищуючи точність результатів [3]-[6].

Важливо забезпечувати справедливість і неупередженість алгоритмів, зберігати конфіденційність даних та збалансовувати використання AI із залученням людського контролю, щоб зберегти комплексність та персоналізацію навчального досвіду. У підсумку, онлайн платформи тестування трансформують підготовку до оцінювань, пропонуючи зручність, інтерактивність, швидкий зворотний зв'язок і широкі можливості для персоналізації навчання. Їхнє використання сприяє підвищенню академічної успішності та вдосконаленню освітнього процесу загалом [3]-[6].

Завдяки таким інтелектуальним системам тестування учні можуть отримувати своєчасні рекомендації та оперативний зворотний зв'язок, а викладачі – своєчасно виявляти проблеми, які залишаються непомітними при традиційному спостереженні. У деяких випадках, наприклад при автоматичному оцінюванні письмових робіт, переваги помітні насамперед для організацій, що проводять оцінювання, адже завдяки автоматизації іспити обробляються швидше і ефективніше. У інших ситуаціях безпосередні вигоди відчувають викладачі: застосування алгоритмів обробки природної мови дозволяє інтегрувати оцінювання у навчальні платформи, створюючи звіти про якість письма учнів, що полегшує аналіз результатів і підготовку рекомендацій для індивідуального навчання [3]-[6].

Разом із перевагами використання штучного інтелекту виникають і суттєві виклики. Одним із головних питань є конфіденційність даних: необхідно визначити, як платформи захищають інформацію про користувачів, як отримується згода учнів та батьків на використання технологій і чи варто ділитися анонімізованими даними з науковцями або сторонніми організаціями. Окрім цього, виклики пов'язані з тим, що штучний інтелект не може повністю замінити людську взаємодію та емоційний досвід навчання, який формує вчитель. Важливо також враховувати проблему упередженості алгоритмів. Машинне оцінювання базується на статистичних даних і може некоректно інтерпретувати результати, відображаючи нерівності в доступі до навчальних ресурсів, а не реальні здібності учнів [3]-[6].

Упередженість в інтелектуальних системах проявляється тоді, коли вхідні дані невірно відображають знання або потенціал учнів, що призводить до спотворених оцінок. Ця проблема проявляється у кількох формах: коли історичні результати відображають обмежені можливості для навчання, а не здібності; коли минулі оцінки використовуються для прогнозування майбутніх результатів, створюючи цикл повторюваних низьких показників; коли алгоритми автоматично розповсюджують упередження через швидкість обробки даних, і коли нерівність у доступі до навчальних ресурсів підтримується, незважаючи на технології [3]-[6].

Експерти зазначають, що повне усунення упередженості є малоімовірним і, можливо, недоцільним завданням. Натомість регулятори та розробники платформ мають визначити допустимий рівень упередженості та розробити механізми, які забезпечать, щоб алгоритми не призводили до нерівного впливу на учнів залежно від раси, етнічного походження, рівня доходу, наявності інвалідності чи рівня володіння англійською мовою. Це дозволяє інтегрувати ШІ в освітні процеси відповідально, зберігаючи баланс між технологією та людським контролем [3]-[6].

Використання штучного інтелекту для покращення процесів оцінювання потребує системного підходу, що охоплює кілька напрямків одночасно. По-перше, важливо інвестувати у дослідження механізмів упередженості в тестуванні та зрозуміти, де саме вона виникає. Результати оцінювань повинні максимально точно відображати знання та навички учнів, а не демонструвати закономірності, що пов'язані з расою, соціально-економічним статусом або іншими нерелевантними чинниками. Аналіз даних та наукові дослідження дозволяють визначити, на якому етапі формуються упередження – у способі вимірювання, інтерпретації відповідей чи процедурі оцінювання – і розробити інструменти для їхнього зменшення [3]-[6].

По-друге, необхідні значні інвестиції у створення нових технологій оцінювання, які б інтегрували можливості штучного інтелекту, віртуальної та змішаної реальності, а також навчальних ігор. Такі ресурси дозволяють

розробляти адаптивні системи тестування, що підлаштовуються під індивідуальні потреби кожного учня. Для реалізації таких проектів потрібне державне фінансування, яке забезпечує масштабні дослідження та впровадження інновацій у системи освіти. Підтримка грантових програм і спеціалізованих ініціатив дозволяє створювати нові методи оцінювання, орієнтовані на ефективність навчання та розвиток персоналізованих підходів [3]-[6].

По-третє, важливо спрямувати ресурси на професійний розвиток педагогів у сфері ефективного застосування інструментів оцінювання. Вчителі повинні володіти навичками створення власних оцінювань, аналізу результатів та використання цих даних для оптимізації навчальної підтримки учнів [3]-[6].

Правильно розроблені засоби оцінювання, що використовують сучасні технології та штучний інтелект, дозволяють прискорити та покращити навчання учнів, надаючи можливості для індивідуальної роботи та різноманітних підходів до практики. Використання автоматизованих міні-тестів, прихованих оцінювань, інтерактивних ігор і віртуальних симуляцій створює широкий спектр засобів для залучення студентів і підвищення якості навчального процесу. Досягнення цього рівня ефективності неможливе без суттєвих інвестицій у дослідження та впровадження нових технологій тестування, що дозволяють надавати викладачам та учням інструменти для більш гнучкого, адаптивного та персоналізованого оцінювання [3]-[6].

Визначено, що програмне забезпечення для організації та управління тестуванням являє собою спеціалізований програмний інструмент, призначений для планування, проведення, контролю та аналізу процедур оцінювання знань, умінь або професійних компетентностей. Такі програмні системи створюють надають інформаційне середовище, у межах якого забезпечується підготовка тестових матеріалів, адміністрування доступу учасників, автоматизована перевірка результатів і формування аналітичних висновків. У результаті формується цілісний підхід до оцінювання, що поєднує методичну складову з технологічною підтримкою.

1.2 Аналіз програмного забезпечення для організації та управління тестуванням

Проаналізуємо програмне забезпечення для організації та управління тестуванням [7]-[14].

Програмне забезпечення Mentimeter (рис. 1.1) є цифровим рішенням, орієнтованим на організацію активної взаємодії під час проведення оцінювання знань. Платформа трансформує звичний формат подання матеріалу в динамічний процес спільної участі, де слухачі не залишаються пасивними спостерігачами. Забезпечується можливість перетворення звичайної презентації на інтерактивне середовище, у якому комунікація відбувається у двосторонньому напрямку [7], [8], [12].

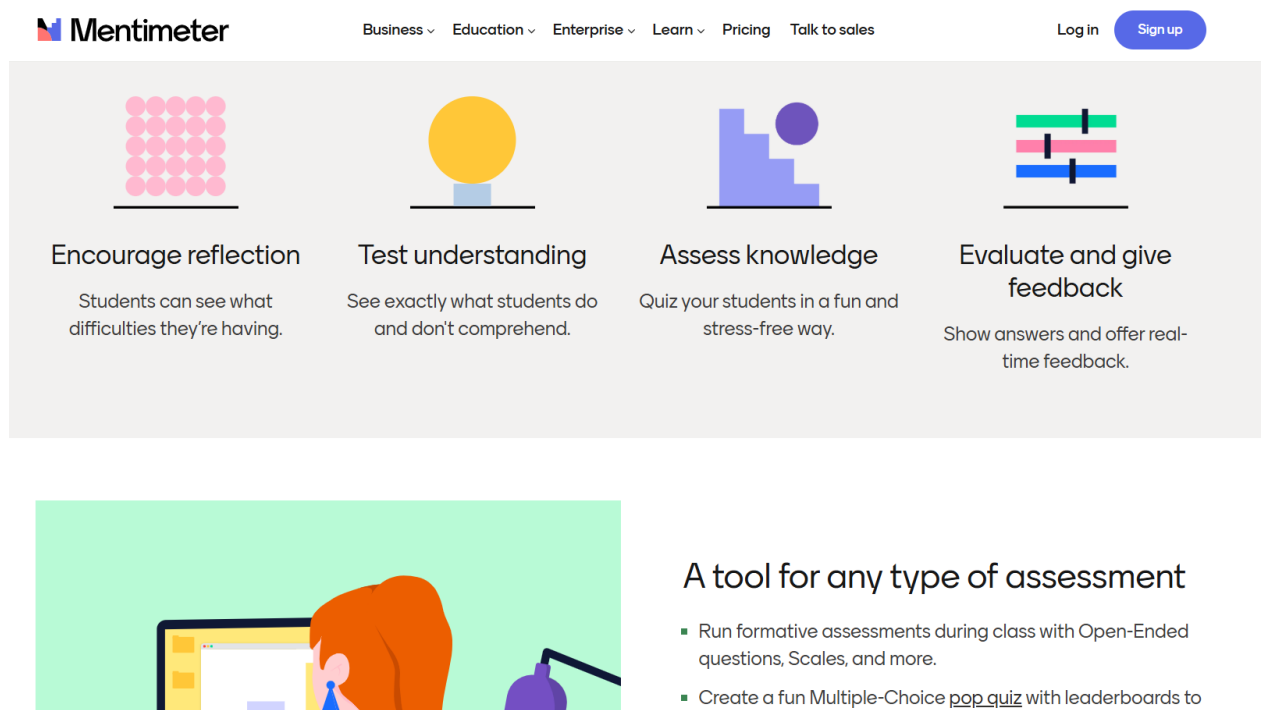


Рисунок 1.1 – Програмне забезпечення Mentimeter [12]

Функціональність програмного забезпечення Mentimeter передбачає створення запитань різних типів, проведення опитувань та організацію перевірки знань із миттєвим отриманням відповідей від учасників. Взаємодія відбувається синхронно, що дозволяє одразу аналізувати реакцію аудиторії та

коригувати подальший перебіг заходу. Участь забезпечується через персональні пристрої, що спрощує підключення та робить процес доступним без потреби у спеціальному обладнанні. Такий підхід сприяє залученню максимальної кількості присутніх і створює атмосферу відкритого обміну думками [7], [8], [12].

Особлива увага у програмному забезпеченні Mentimeter приділяється візуальному відображенню результатів. Отримані відповіді автоматично узагальнюються та подаються у наочній формі, що полегшує сприйняття інформації та допомагає швидше виявити тенденції, рівень засвоєння матеріалу або загальні настрої аудиторії. Завдяки цьому процес оцінювання у програмному забезпеченні Mentimeter набуває прозорості, а обговорення результатів стає більш аргументованим [7], [8], [12].

У програмному забезпеченні Mentimeter передбачено гнучкість у використанні, що дозволяє застосовувати платформу в освітньому середовищі, під час професійних тренінгів, робочих зустрічей або публічних заходів різного масштабу. Програмне забезпечення Mentimeter органічно поєднується з іншими програмними засобами для створення презентацій, що дає змогу інтегрувати його в уже налагоджений робочий процес без складних технічних налаштувань [7], [8], [12].

Програма Mentimeter орієнтована на викладачів, тренерів, фасилітаторів та спікерів, які прагнуть активізувати учасників і підвищити рівень їхньої включеності. Використання такого підходу сприяє формуванню інклюзивного простору, де кожному надається можливість висловити позицію. Інтерактивність стає невід'ємною складовою оцінювання, а подання матеріалу набуває більш сучасного й ефективного характеру [7], [8], [12].

Основними характеристиками програмного забезпечення Mentimeter є такі [7], [8], [12]:

- інтерактивне опитування в реальному часі: у програмному забезпеченні Mentimeter підтримується створення запитань різних форматів із

миттєвим збором відповідей від аудиторії через персональні пристрої без потреби встановлення додаткового програмного забезпечення [7], [8], [12];

- візуалізація результатів: у програмному забезпеченні Mentimeter відповіді автоматично відображаються у вигляді діаграм, шкал та інших графічних форм, що полегшує аналіз та інтерпретацію отриманих результатів тестування [7], [8], [12];

- підтримка різних типів запитань: у програмному забезпеченні Mentimeter реалізовано можливість використання тестових завдань, відкритих відповідей, хмар слів та шкал оцінювання для різноманітних освітніх і професійних потреб [7], [8], [12];

- зручність доступу: робота програмного забезпечення Mentimeter забезпечується через веббраузер із будь-якого сучасного пристрою, що спрощує підключення та розширює охоплення аудиторії [7], [8], [12];

- інтеграція з презентаційними сервісами: у програмному забезпеченні Mentimeter передбачено поєднання з популярними інструментами для створення слайдів, що дозволяє органічно включати інтерактивні елементи в готові матеріали [7], [8], [12];

- аналітика та збереження даних: результати сесій зберігаються та можуть бути експортовані для подальшого опрацювання й оцінювання ефективності заходів [7], [8], [12];

- універсальність застосування: використання програмного забезпечення Mentimeter можливе під час навчальних занять, тренінгів, ділових зустрічей та публічних виступів різного формату [7], [8], [12];

- підвищення залученості: створюються умови для активної участі кожного присутнього, що сприяє формуванню відкритого діалогу та більш глибокого засвоєння матеріалу [7], [8], [12];

- простота налаштування: інтерфейс у програмному забезпеченні Mentimeter орієнтований на швидке створення сесій без складних технічних дій або спеціалізованих знань [7], [8], [12];

– масштабованість використання: програмне забезпечення Mentimeter придатно як для невеликих груп, так і для великих аудиторій без втрати функціональності [7], [8], [12].

Серед недоліків програмного забезпечення Mentimeter можна відзначити залежність від стабільного Інтернет з'єднання, обмеження функціональності у безкоштовній версії, а також необхідність попередньої підготовки матеріалів для досягнення максимальної ефективності інтерактивної взаємодії [7], [8], [12].

Програмне забезпечення ExamSoft (рис. 1.2) представляє собою комплексне рішення для організації, проведення та аналізу оцінювання в освітньому середовищі. Система ExamSoft орієнтується на забезпечення академічної доброчесності, підвищення якості навчання та підтримку процедур внутрішнього й зовнішнього контролю. У межах платформи ExamSoft створюється цілісна екосистема, у якій процес підготовки іспитів поєднується з глибоким аналізом результатів та формуванням аналітичних висновків для вдосконалення освітніх програм [9], [11], [13].



[Why ExamSoft](#) [Solutions](#) [Who We Serve](#) [Resources](#) [About](#) [Contact](#)

One Core Platform
Two Simple Interfaces

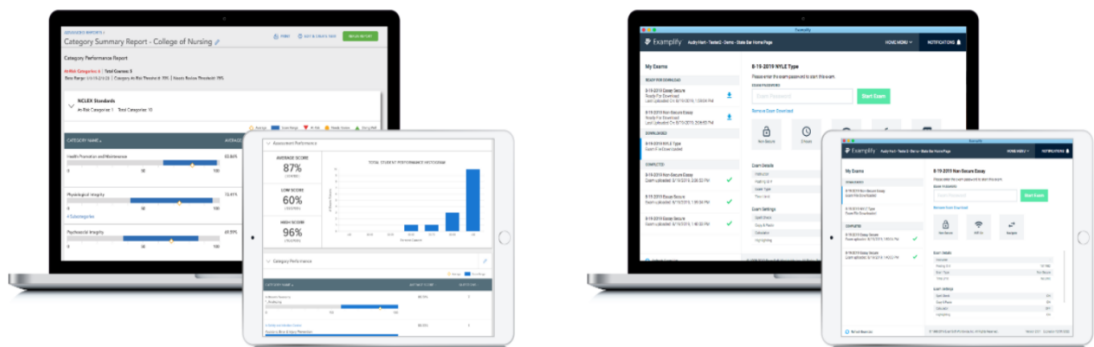


Рисунок 1.2 – Програмне забезпечення ExamSoft [13]

Передбачено інструментарій для розроблення великої кількості тестових матеріалів із різними форматами завдань. Проведення оцінювання може здійснюватися в мережевому середовищі або без підключення до нього, що розширює можливості застосування в різних умовах. Автоматизована перевірка результатів у програмному забезпеченні ExamSoft дозволяє оперативно отримувати підсумкові дані, а також формувати структуровані звіти щодо досягнень окремих здобувачів освіти та груп загалом. Усі відомості акумулюються в єдиному інформаційному просторі, що спрощує систематизацію та подальший аналіз [9], [11], [13].

Значна увага у програмному забезпеченні ExamSoft приділяється індивідуалізації оцінювання. Налаштування параметрів тестування дає змогу адаптувати зміст і форму завдань відповідно до освітніх потреб, а також здійснювати зіставлення результатів із навчальними цілями та компетентностями. Отримані дані можуть бути використані для коригування методик викладання, визначення сильних сторін і виявлення напрямів для покращення навчальних результатів. Передбачено формування детального зворотного зв'язку, який сприяє усвідомленню власного рівня підготовки та стимулює подальший розвиток [9], [11], [13].

Платформа ExamSoft вирізняється розвиненими механізмами захисту. У програмному забезпеченні ExamSoft Реалізовано засоби ідентифікації учасників, контроль за дотриманням правил проходження іспиту та обмеження доступу до сторонніх ресурсів під час виконання завдань. Забезпечується блокування браузера, контроль активності на пристрої та застосування алгоритмів для виявлення підозрілих дій. Додатково використовується варіативність порядку подання запитань, що знижує ризик недобросовісної поведінки. Такий підхід сприяє збереженню цілісності результатів і підвищує довіру до процедури оцінювання [9], [11], [13].

Інструмент підтримує різні формати проведення іспитів, зокрема комп'ютерний і мобільний, а також допускає використання традиційних варіантів. Це дозволяє інтегрувати систему в різноманітні освітні контексти

без суттєвих змін усталених процесів. Окремо варто відзначити можливість імпорту завдань із зовнішніх джерел, що пришвидшує підготовчий етап та оптимізує роботу викладача [9], [11], [13].

Водночас застосування розширених засобів безпеки може іноді впливати на стабільність роботи користувацьких пристроїв, що потребує належної технічної підтримки. Крім того, інтерфейс у програмному забезпеченні ExamSoft може вимагати додаткового часу для адаптації та освоєння. Незважаючи на це, система розглядається як ефективний інструмент управління оцінюванням, який поєднує контроль, аналітику та методичну підтримку в межах єдиного програмного середовища [9], [11], [13].

Програмне забезпечення ExamSoft має такі особливості [9], [11], [13]:

- комплексне управління оцінюванням: у програмному забезпеченні ExamSoft забезпечується повний цикл створення, проведення та аналізу тестувань у межах єдиного цифрового середовища з централізованим зберіганням результатів [9], [11], [13];

- розширена аналітика: у програмному забезпеченні ExamSoft надається деталізована звітність із можливістю відстеження навчальних досягнень, визначення прогалин у знаннях та підтримки прийняття педагогічних рішень [9], [11], [13];

- безпечне тестування: у програмному забезпеченні ExamSoft реалізовано інструменти контролю доступу, блокування сторонніх ресурсів і моніторингу активності для забезпечення академічної учасників освітнього процесу [9], [11], [13];

- перевірка особи: у програмному забезпеченні ExamSoft передбачено механізми ідентифікації учасників оцінювання з метою підтвердження автентичності проходження іспиту [9], [11], [13];

- гнучкість форматів: у програмному забезпеченні ExamSoft підтримується проведення оцінювань у різних форматах, включаючи комп'ютерний та мобільний варіанти, що розширює можливості використання програми [9], [11], [13];

- індивідуалізація завдань: у програмному забезпеченні ExamSoft допускається налаштування параметрів тестування відповідно до освітніх потреб та особливостей навчальних програм [9], [11], [13];

- інтеграція з навчальними програмами: у програмному забезпеченні ExamSoft забезпечується зіставлення результатів із визначеними компетентностями та освітніми цілями [9], [11], [13];

- централізований збір даних: усі результати оцінювання акумулюються в єдиній системі для подальшого аналізу та формування підсумкових висновків [9], [11], [13];

- автоматизоване оцінювання: у програмному забезпеченні ExamSoft передбачено швидку обробку відповідей із мінімізацією ручної перевірки та зменшенням ризику помилок [9], [11], [13];

- імпорт і масштабування контенту: у програмному забезпеченні ExamSoft підтримується завантаження великої кількості тестових матеріалів із зовнішніх джерел для оптимізації підготовчого процесу [9], [11], [13].

Серед недоліків програмного забезпечення ExamSoft можна відзначити підвищені вимоги до технічної стабільності пристроїв під час проходження тестування, складність початкового налаштування окремих функцій безпеки, а також інтерфейс, який може потребувати додаткового часу для повного освоєння користувачами [9], [11], [13].

Програмне забезпечення Kaldin (рис. 1.3) є гнучким цифровим рішенням для проведення дистанційного оцінювання в освітніх установах і корпоративному середовищі. Платформа Kaldin з відкритим програмним кодом забезпечує широкі можливості адаптації відповідно до організаційних потреб, що дозволяє змінювати структуру модулів, розширювати функціональність і впроваджувати власні налаштування без обмежень, характерних для закритих систем. Такий підхід створює умови для глибокої інтеграції з іншими інформаційними ресурсами та внутрішніми сервісами установи [10], [14].

Create Exams, Schedule & Evaluate Easily.

- ✓ Highly customizable
- ✓ Safe & Secure data
- ✓ Import/Export data
- ✓ Free for Everyone
- ✓ Readymade Questions



How it works

Questions

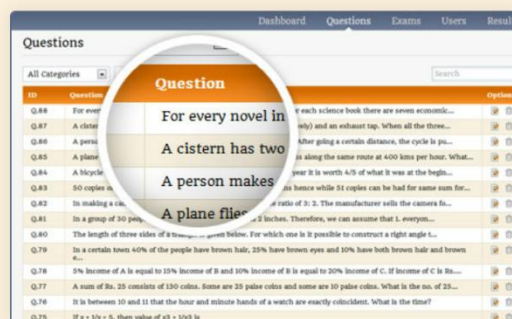
Use existing hundreds of questions or add new multiple choice questions in multiple categories.

Exam

Invite group of users or email link of Exam to be given on scheduled date and time.

Results

Real-time notifications, analysis by user, exam and category with ability to download certificates.

[Signup Now](#)
[Exam Demo](#)


Readymade Questions

Use existing questions in categories including Analytical Skills, General Aptitude, English Language, Technology and continuously growing set of questions.

Customizable questions

Multiple or single choice, two or more options, define complexity and specify correct answer.

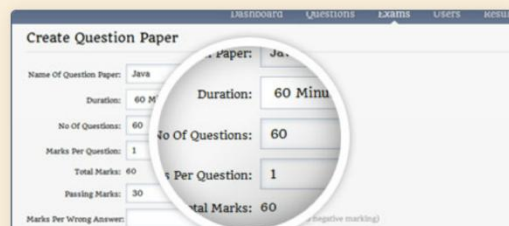


Рисунок 1.3 – Програмне забезпечення Kaldin [14]

Інтерфейс програмного забезпечення Kaldin орієнтовано на персоналізацію, що дає змогу змінювати візуальне оформлення, додавати корпоративну символіку та формувати індивідуалізовані варіанти тестових матеріалів. Передбачено використання великої бази завдань із різних тематичних напрямів, серед яких аналітичне мислення, мовна підготовка, загальні компетентності та технічні дисципліни. Водночас не встановлюються обмеження щодо наповнення контенту, оскільки користувачі можуть самостійно створювати нові розділи, редагувати питання та оновлювати навчальні матеріали відповідно до власних стандартів [10], [14].

У програмному забезпеченні Kaldin підтримується варіативність типів завдань, що дозволяє формувати тести з альтернативним вибором, множинними відповідями та іншими форматами перевірки знань. Надається можливість визначення рівня складності, зазначення правильних відповідей і структурування матеріалів за допомогою додаткових категорій та підкатегорій. Це спрощує адміністрування великих банків запитань і забезпечує впорядкованість екзаменаційного процесу [10], [14].

Вбудований механізм планування у програмному забезпеченні Kaldin дає змогу організовувати проведення оцінювання з попереднім визначенням параметрів сесії та інформуванням учасників через електронні повідомлення. Запрошення надсилаються централізовано, що полегшує координацію та зменшує адміністративне навантаження. Процедура роботи платформи передбачає створення або вибір запитань, налаштування умов проходження іспиту та подальше автоматичне формування результатів. Після завершення тестування здійснюється автоматизована перевірка відповідей і генерується підсумкова інформація щодо кожного учасника [10], [14].

Програмне забезпечення Kaldin підтримує багатокористувацький режим, що дозволяє одночасну роботу адміністраторів, викладачів і здобувачів освіти в межах одного середовища. Реалізовано функцію випадкового формування запитань і відповідей для мінімізації ризиків недоброчесності. Додаткові засоби захисту спрямовані на забезпечення цілісності екзаменаційних матеріалів та запобігання несанкціонованому доступу. У підсумку програмне рішення постає як адаптивна платформа для організації тестування, що поєднує індивідуалізацію, автоматизацію та контроль у межах єдиної інформаційної системи [10], [14].

Програмне забезпечення Kaldin має такі особливості [10], [14]:

- відкрита архітектура коду: програмне забезпечення Kaldin надає можливість модифікації функціональних модулів відповідно до потреб установи та адаптації системи під специфіку навчального або корпоративного середовища [10], [14];

– гнучке налаштування інтерфейсу: підтримується персоналізація зовнішнього вигляду, додавання корпоративної символіки та формування індивідуалізованих шаблонів тестування [10], [14];

– багатоформатні запитання: у програмному забезпеченні Kaldin реалізовано створення завдань із різними типами відповідей, із можливістю визначення складності [10], [14];

– керування банком питань: у програмному забезпеченні Kaldin передбачено структурування матеріалів за категоріями та підкатегоріями для впорядкованого зберігання та швидкого пошуку [10], [14];

– автоматизоване оцінювання: результати перевіряються системою без додаткового втручання, що прискорює обробку відповідей та формування підсумків [10], [14];

– планування іспитів: інтегрований механізм програмного забезпечення Kaldin дозволяє встановлювати параметри проведення тестування та надсилати запрошення учасникам електронними засобами [10], [14];

– випадкова генерація завдань: у програмному забезпеченні Kaldin підтримується перемішування питань і варіантів відповідей для підвищення об'єктивності та зменшення ризиків порушень [10], [14];

– багатокористувацький режим: у програмному забезпеченні Kaldin забезпечується одночасна робота адміністраторів, викладачів і учасників у межах єдиного цифрового простору [10], [14];

– інтеграційні можливості: спрощується поєднання з іншими інформаційними системами завдяки відкритій структурі рішення [10], [14];

– засоби захисту даних: у програмному забезпеченні Kaldin впроваджено механізми контролю доступу та безпеки для збереження конфіденційності екзаменаційних матеріалів [10], [14].

Серед недоліків програмного забезпечення Kaldin можна відзначити потребу в технічних ресурсах для повноцінного розгортання та підтримки відкритої системи, можливу складність первинного налаштування для

користувачів без спеціалізованих знань, а також залежність стабільності роботи від конфігурації серверного середовища [10], [14].

Порівняльну характеристику програмного забезпечення для організації та управління тестуванням наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для організації та управління тестуванням

Критерій порівняння	Mentimeter [12]	ExamSoft [13]	Kaldin [14]
Безпечне оцінювання	+—	+	+—
Гнучкість налаштування	+—	+—	+
Візуалізація даних	+	+—	+—
Зручність інтеграції з іншими системами	+—	+—	—
Простота використання	+	+—	+—
Масштабованість для великих груп	+—	+	+—

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для організації та управління тестуванням. Проте впровадження деяких програм для організації та управління тестуванням пов'язано з високими витратами на ліцензування, складною інтеграцією з іншими програмними платформами, навчанням персоналу та адміністрації. За відсутності достатнього рівня підготовки користувачів знижується ефективність використання функціональних можливостей таких програмних систем, особливо у випадках, коли такі системи є досить складними, що призводить до формального застосування інструментів без повної реалізації їх потенціалу. Не менш важливим є питання захисту даних і забезпечення конфіденційності. У разі недостатнього рівня захисту існує ймовірність несанкціонованого доступу до результатів

тестування або персональної інформації користувачів. Крім того, автоматизований контроль не повністю усуває можливість недоброчесної поведінки здобувачів освіти або інших учасників освітнього процесу, оскільки зберігається ризик використання сторонніх ресурсів чи технічних засобів під час проходження тестів. Тому актуальною є розробка програмного забезпечення для організації та управління тестуванням.

При розробці програмного забезпечення для організації та управління тестуванням необхідно забезпечити такі функціональні вимоги:

- підтримка можливості реєстрації та авторизації користувачів із розмежуванням прав доступу відповідно до ролей у системі;
- підтримка можливості зберігання даних про користувачів у базі даних із забезпеченням цілісності інформації;
- можливість створення, налаштування та адміністрування тестів із визначенням параметрів проходження та критеріїв оцінювання;
- підтримка формування та редагування банку запитань із можливістю встановлення різних типів відповідей і вагових коефіцієнтів оцінювання;
- можливість автоматичного обчислення результатів тестування з урахуванням встановлених правил та збереження отриманих показників у базі даних;
- можливість обмеження доступу до тестів відповідно до встановлених правил і параметрів повторного проходження;
- можливість взаємодії клієнтської та серверної частин системи через програмний інтерфейс із забезпеченням безпечної передачі даних;
- підтримка масштабованості системи для забезпечення одночасної роботи значної кількості користувачів без втрати продуктивності;
- підтримка коректного відображення інтерфейсу в настільному та мобільному клієнтах із забезпеченням зручності використання програмного забезпечення.

1.3 Висновки за розділом 1

Визначено, що програмне забезпечення для організації та управління тестуванням являє собою спеціалізований програмний інструмент, призначений для планування, проведення, контролю та аналізу процедур оцінювання знань, умінь або професійних компетентностей. Такі програмні системи створюють надають інформаційне середовище, у межах якого забезпечується підготовка тестових матеріалів, адміністрування доступу учасників, автоматизована перевірка результатів і формування аналітичних висновків.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для організації та управління тестуванням. Проте впровадження деяких програм для організації та управління тестуванням пов'язано з високими витратами на ліцензування, складною інтеграцією з іншими програмними платформами, навчанням персоналу та адміністрації. За відсутності достатнього рівня підготовки користувачів знижується ефективність використання функціональних можливостей таких програмних систем, особливо у випадках, коли такі системи є досить складними, що призводить до формального застосування інструментів без повної реалізації їх потенціалу. Не менш важливим є питання захисту даних і забезпечення конфіденційності. У разі недостатнього рівня захисту існує ймовірність несанкціонованого доступу до результатів тестування або персональної інформації користувачів. Крім того, автоматизований контроль не повністю усуває можливість недобросовісної поведінки здобувачів освіти або інших учасників освітнього процесу, оскільки зберігається ризик використання сторонніх ресурсів чи технічних засобів під час проходження тестів. Тому актуальною є розробка програмного забезпечення для організації та управління тестуванням.

Сформульовано функціональні вимоги до програмного забезпечення для організації та управління тестуванням.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для організації та управління тестуванням було використано мову програмування Kotlin [15]-[16].

Kotlin – це мова програмування та платформа для створення програмних рішень різного рівня складності. Її поява пов’язана з діяльністю компанії JetBrains, яка орієнтувалася на усунення типових недоліків традиційних мов і підвищення продуктивності розробки без втрати сумісності з уже наявною інфраструктурою [15]-[16].

Мова Kotlin дозволяє спільно використовувати частину бізнес-логіки між серверною, мобільною та навіть вебкомпонентою. Завдяки цьому скорочуються витрати на підтримку кількох незалежних кодових баз, забезпечується узгодженість логіки обробки даних та підвищується цілісність архітектури програмного продукту. У сучасних умовах, коли програмні системи мають працювати одночасно на різних типах пристроїв, така можливість набуває стратегічного значення [15]-[16].

Окремої уваги заслуговує інтеграція Kotlin з екосистемою Android. Після офіційної підтримки з боку Google ця мова стала пріоритетною для створення мобільних застосунків на Android. Це сприяло стрімкому зростанню спільноти розробників, розширенню кількості бібліотек і підвищенню рівня документації. Активна підтримка з боку великої технологічної компанії позитивно впливає на стабільність розвитку мови та її довгострокову перспективу [15]-[16].

З точки зору архітектурного проектування Kotlin підтримує як об’єктно-орієнтовану, так і функціональну парадигми програмування. Наявність лямбда-виразів, функцій вищого порядку, data-класів і розширень класів дозволяє формувати гнучкі та виразні моделі предметної області. Це особливо важливо під час розробки складних інформаційних систем, де

необхідно забезпечити чітке розмежування відповідальностей, масштабованість і легкість модифікації коду [15]-[16].

Крім того, мова характеризується високим рівнем інструментальної підтримки. Середовище розробки, системи статичного аналізу, засоби автоматичного тестування та інтеграції з системами безперервної інтеграції забезпечують контроль якості програмного продукту на всіх етапах життєвого циклу. Розвинена система типізації та механізми перевірки на етапі компіляції сприяють ранньому виявленню помилок і підвищенню надійності кінцевого рішення [15]-[16].

Не менш важливим є аспект продуктивності. Оскільки Kotlin компілюється у байт-код, сумісний із Java Virtual Machine, забезпечується ефективне виконання програм і можливість використання оптимізацій, притаманних JVM. При цьому зберігається гнучкість розгортання як у традиційних серверних середовищах, так і в хмарній інфраструктурі [15]-[16].

Таким чином, Kotlin можна охарактеризувати як сучасну, еволюційно розвинену мову програмування, що поєднує лаконічність синтаксису, безпечність типізації, мультиплатформеність і глибоку інтеграцію з усталеними технологічними екосистемами. Її застосування доцільно розглядати не лише з позиції зручності написання коду, а й як стратегічний вибір для створення довготривалих, масштабованих і технологічно стійких програмних продуктів [15]-[16].

Доцільність вибору мови програмування Kotlin для розробки програмного забезпечення, призначеного для організації та управління тестуванням, обумовлюється сукупністю технологічних, архітектурних та експлуатаційних переваг. З огляду на необхідність створення надійної, масштабованої та безпечної системи, що обробляє значні обсяги даних і забезпечує одночасну роботу багатьох користувачів, важливого значення набувають стабільність платформи виконання та підтримка сучасних парадигм програмування [15]-[16].

Мова Kotlin функціонує на платформі Java Virtual Machine (JVM), що надає доступ до зрілої екосистеми бібліотек, фреймворків і засобів інтеграції. Завдяки повній сумісності з кодом, написаним мовою Java, забезпечується можливість використання існуючих рішень без необхідності повного перепроєктування системи. Це сприяє скороченню витрат часу на розробку та спрощує інтеграцію з базами даних, серверами застосунків і сторонніми сервісами [15]-[16].

Важливою перевагою є лаконічність синтаксису та зменшення шаблонного коду, що позитивно впливає на читабельність і підтримуваність програмного продукту. У процесі створення систем управління тестуванням, де реалізуються модулі авторизації, формування тестових завдань, автоматизованого оцінювання та аналітики результатів, зменшення обсягу технічного коду знижує ймовірність помилок і полегшує подальшу модифікацію функціоналу. Підвищується продуктивність роботи розробників і спрощується командна взаємодія [15]-[16].

Суттєве значення має вбудована підтримка безпечної роботи з null-значеннями, що мінімізує ризик виникнення типових помилок виконання. Для систем тестування, які оперують персональними даними користувачів, результатами оцінювання та статистичною інформацією, підвищення надійності обробки даних є критично важливим. Додатково мова підтримує сучасні підходи до асинхронного програмування, зокрема механізм корутин, що дозволяє ефективно реалізовувати багатопоточну обробку запитів без надмірного ускладнення коду [15]-[16].

У разі розробки веборієнтованої або мобільної складової системи Kotlin забезпечує універсальність застосування. За допомогою відповідних фреймворків можливе створення серверної логіки, клієнтських вебінтерфейсів і мобільних застосунків у межах єдиної технологічної стекової екосистеми. Такий підхід сприяє уніфікації архітектури та спрощує супровід програмного продукту протягом життєвого циклу [15]-[16].

Обґрунтування вибору мови програмування для розробки програмного забезпечення для організації та управління тестуванням наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення для організації та управління тестуванням

Критерій порівняння мов програмування	Мова програмування		
	Kotlin	Python	Java
Мультиплатформеність (сервер/мобільні/веб)	+	+—	+—
Сумісність з існуючою інфраструктурою JVM	+	—	+
Лаконічність синтаксису	+	+	—
Асинхронне програмування та обробка потоків	+	+—	+—
Продуктивність виконання	+	+—	+

Отже, для реалізації програмного забезпечення для організації та управління тестуванням обрано мову програмування Kotlin, що є сучасною, еволюційно розвиненою мовою, яка поєднує лаконічність синтаксису, безпечність типізації, мультиплатформеність і глибоку інтеграцію з усталеними технологічними екосистемами, що дозволяє розробляти програми, які характеризуються високою надійністю, масштабованістю, інтеграційними можливостями та зручністю супроводу. Поєднання сучасних мовних конструкцій Kotlin, сумісності з усталеною інфраструктурою та орієнтації на безпечну розробку формує технологічну основу для створення ефективної та стійкої інформаційної системи.

2.2 Вибір середовища розробки для створення програмного забезпечення для організації та управління тестуванням

В якості середовища розробки для створення програмного забезпечення для організації та управління тестуванням було обрано середовище Android Studio [17]-[18].

Середовище розробки IntelliJ IDEA розроблено компанією JetBrains, що орієнтувалася на інтегровану підтримку мов JVM та сучасних технологій, тому середовище забезпечує глибоку сумісність з мовою Kotlin і дозволяє максимально використовувати її функціональні можливості [17]-[18].

Важливою особливістю є потужний механізм статичного аналізу коду, який не лише підказує синтаксичні помилки, а й виявляє потенційні логічні недоліки, що особливо цінно при реалізації складних алгоритмів обробки тестових даних або автоматизованого оцінювання. Інструмент рефакторингу дозволяє безпечно перенесення імен, методів та класів, що сприяє підтримці чистої архітектури і полегшує масштабування системи [17]-[18].

IDE пропонує інтегровані засоби налагодження, профілювання та моніторингу продуктивності додатків, що дозволяє виявляти «вузькі місця» в логіці обробки запитів або генерації статистики результатів тестування. Це критично для великих навчальних або корпоративних платформ, де одночасно обробляються дані сотень або тисяч користувачів [17]-[18].

Середовище розробки IntelliJ IDEA підтримує багатоплатформеність та роботу з різними фреймворками. IntelliJ IDEA дозволяє створювати серверні застосунки, вебінтерфейси та мобільні компоненти в єдиній екосистемі, що спрощує інтеграцію різних частин системи управління тестуванням. Одночасно забезпечується робота з базами даних, REST API, хмарними сервісами та системами контролю версій без необхідності додаткової конфігурації [17]-[18].

Ще одним важливим аспектом є екосистема плагінів, які розширюють можливості IDE відповідно до специфічних завдань: від інтеграції з системами

тестування і контейнеризації до додаткових засобів аналізу та візуалізації даних. Це дозволяє створювати налаштоване робоче середовище, оптимізоване під конкретний проєкт або команду [17]-[18].

Таким чином, середовище IntelliJ IDEA не лише полегшує написання та підтримку коду, а й створює комплексну платформу для реалізації надійного, безпечного, масштабованого та високопродуктивного програмного забезпечення. Його можливості охоплюють весь життєвий цикл проєкту — від проектування та кодування до тестування, розгортання та супроводу, що робить середовище ключовим інструментом для професійної розробки систем управління тестуванням [17]-[18].

Вибір середовища розробки IntelliJ IDEA для створення програмного забезпечення для організації та управління тестуванням є доцільним через комплексну інтеграцію з мовою Kotlin і можливість ефективно реалізовувати всі етапи життєвого циклу розробки програмного продукту. IntelliJ IDEA надає широкий спектр інструментів, які спрощують написання, налагодження, тестування та підтримку коду, що особливо важливо для складних систем, які обробляють великі обсяги даних та одночасно працюють з багатьма користувачами [17]-[18].

Середовище відзначається потужними засобами автодоповнення коду, інтегрованим аналізом якості та рефакторингу, що зменшує кількість помилок ще на етапі написання програми. Для систем управління тестуванням це критично, оскільки дозволяє швидко розробляти модулі формування тестів, обробки результатів та генерації аналітики без ризику втрати логічної цілісності програми. Крім того, IDE забезпечує підтримку корутин і багатопоточної обробки, що сприяє ефективному виконанню запитів від великої кількості користувачів у реальному часі [17]-[18].

IntelliJ IDEA також забезпечує глибоку інтеграцію з системами контролю версій, системами безперервної інтеграції та різноманітними фреймворками, що дозволяє підтримувати автоматичне тестування, збірку та розгортання продукту. Це підвищує ефективність командної роботи, скорочує

час на супровід і обслуговування програмного забезпечення та забезпечує стабільність функціонування системи навіть при масштабуванні [17]-[18].

Окремим аргументом є висока якість документації, велика спільнота розробників та регулярні оновлення середовища, які забезпечують підтримку новітніх технологій, бібліотек і стандартів. Завдяки цьому IntelliJ IDEA дозволяє реалізовувати проєкти сучасного рівня, підтримувати високі стандарти безпеки, зручності використання та продуктивності, що робить її оптимальним середовищем для створення надійного та масштабованого програмного забезпечення для управління тестуванням [17]-[18].

Обґрунтування вибору середовища розробки для створення програмного забезпечення для організації та управління тестуванням наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення для організації та управління тестуванням

Критерій порівняння середовищ розробки	Середовища розробки		
	IntelliJ IDEA	Eclipse	NetBeans
Інструменти автодоповнення та рефакторингу	+	+—	+—
Інтеграція з системами контролю версій	+	+	+—
Підтримка багатоплатформеності	+	+—	+—
Плагіни та розширюваність	+	+	+—
Підтримка серверних і веб-проєктів	+	+—	+—

Таким чином, для створення програмного забезпечення для організації та управління тестуванням обрано середовище розробки IntelliJ IDEA через комплексну інтеграцію з мовою Kotlin і можливість ефективно реалізовувати всі етапи життєвого циклу розробки програмного продукту. IntelliJ IDEA надає широкий спектр інструментів, які спрощують написання, налагодження, тестування та підтримку коду, що особливо важливо для складних систем, які обробляють великі обсяги даних та одночасно працюють з багатьма користувачами.

2.3 Висновки за розділом 2

Для реалізації програмного забезпечення для організації та управління тестуванням обрано мову програмування Kotlin, що є сучасною, еволюційно розвиненою мовою, яка поєднує лаконічність синтаксису, безпечність типізації, мультиплатформеність і глибоку інтеграцію з усталеними технологічними екосистемами, що дозволяє розробляти програми, які характеризуються високою надійністю, масштабованістю, інтеграційними можливостями та зручністю супроводу. Поєднання сучасних мовних конструкцій Kotlin, сумісності з усталеною інфраструктурою та орієнтації на безпечну розробку формує технологічну основу для створення ефективної та стійкої інформаційної системи.

Для створення програмного забезпечення для організації та управління тестуванням обрано середовище розробки IntelliJ IDEA через комплексну інтеграцію з мовою Kotlin і можливість ефективно реалізовувати всі етапи життєвого циклу розробки програмного продукту. IntelliJ IDEA надає широкий спектр інструментів, які спрощують написання, налагодження, тестування та підтримку коду, що особливо важливо для складних систем, які обробляють великі обсяги даних та одночасно працюють з багатьма користувачами.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення для організації та управління тестуванням

Структуру програмного забезпечення для організації та управління тестуванням наведено на рис. 3.1.

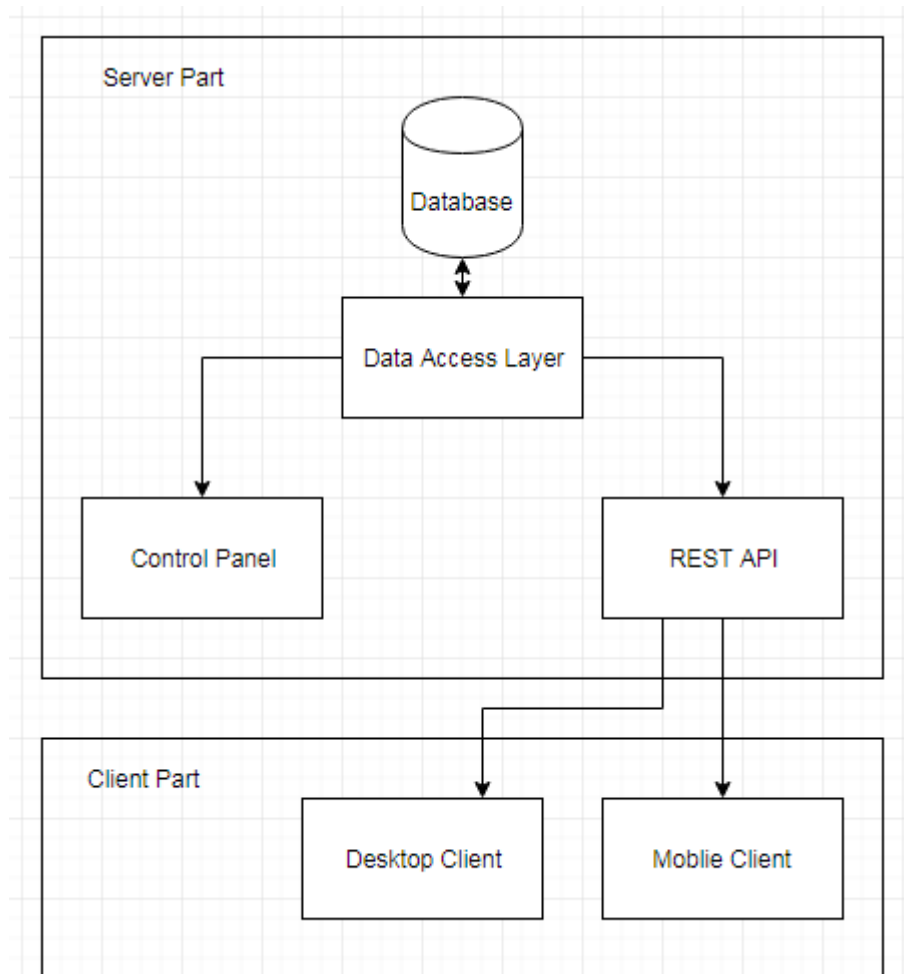


Рисунок 3.1 – Структура програмного забезпечення для організації та управління тестуванням

Структура програмного забезпечення для організації та управління тестуванням являє собою дворівневу систему, що поєднує серверну та клієнтську частини й забезпечує цілісність зберігання даних, обробки запитів і взаємодії з користувачами. Така архітектура орієнтована на розмежування

відповідальностей між рівнями, підвищення масштабованості та спрощення супроводу програмного продукту.

Серверна частина програмного забезпечення для організації та управління тестуванням є ядром системи й реалізує логіку зберігання, обробки та керування інформаційними потоками.

Важливим компонентом серверної частини є модуль Data Access Layer, який виконує роль посередника між фізичним рівнем зберігання даних і прикладною логікою. Його призначення полягає у формалізації операцій читання, запису, оновлення та видалення інформації, а також у забезпеченні абстракції доступу до бази даних. Завдяки цьому зменшується залежність інших модулів від конкретної реалізації сховища, підвищується гнучкість системи та спрощується її модифікація або міграція на іншу платформу зберігання даних. Data Access Layer також реалізує перевірку коректності запитів, обробку виняткових ситуацій і оптимізацію виконання транзакцій.

Модуль Control Panel призначений для адміністративного керування системою. У його межах реалізується створення та редагування тестів, формування структури запитань, налаштування критеріїв оцінювання, управління ролями користувачів, перегляд звітності та статистики. Функціонально цей модуль забезпечує інтерфейс для адміністраторів і викладачів або менеджерів, які відповідають за організацію процесу тестування. Він взаємодіє з Data Access Layer для отримання та збереження змін, формуючи централізований механізм керування всією системою.

Модуль REST API виконує функцію прикладного інтерфейсу взаємодії із зовнішніми клієнтами. Його призначення полягає в обробці HTTP-запитів, авторизації користувачів, передачі тестових завдань, прийнятті відповідей і поверненні результатів у стандартизованому форматі. REST API забезпечує ізоляцію серверної логіки від клієнтських застосунків, що дозволяє підтримувати різні типи клієнтів без дублювання бізнес-логіки. У межах цього модуля реалізуються механізми аутентифікації, контролю доступу, журналювання дій та забезпечення безпеки передачі даних.

Важливим елементом є модуль роботи з базою даних, який призначений для довготривалого збереження облікових записів користувачів, банків тестових завдань, результатів проходження тестів, статистичних показників і налаштувань системи. У межах цього модуля забезпечується цілісність даних, механізми транзакційності, резервне копіювання та контроль доступу до інформації. Структура сховища даних проєктується таким чином, щоб підтримувати як стандартні запити вибірки результатів, так і складні аналітичні операції.

Клієнтська частина програмного забезпечення для організації та управління тестуванням представлена модулями Desktop Client та Mobile Client, які орієнтовані на різні сценарії використання системи. Модуль Desktop Client призначений для роботи на персональних комп'ютерах або ноутбуках і забезпечує розширений функціонал для проходження тестування, перегляду детальної статистики та взаємодії з аналітичними звітами. Його функції включають відображення тестових завдань різних типів, контроль часу виконання, локальну валідацію введених відповідей і обмін даними із сервером через REST API. Інтерфейс цього модуля оптимізується для роботи з великим екраном, що дозволяє одночасно відображати додаткову інформацію, підказки або результати попередніх спроб.

Модуль Mobile Client орієнтований на використання системи з мобільних пристроїв і забезпечує адаптивний інтерфейс, оптимізований під сенсорне керування. Його основне призначення полягає в наданні можливості проходження тестів у будь-який час і з будь-якого місця, що розширює доступність системи. Функціонально мобільний клієнт підтримує авторизацію користувачів, отримання тестових завдань із сервера, передачу відповідей, відображення підсумкових результатів і збереження історії проходження. При цьому забезпечується мінімізація обсягу переданих даних та оптимізація продуктивності з урахуванням обмежених ресурсів мобільних пристроїв.

Таким чином, структура програмного забезпечення для організації та управління тестуванням базується на чіткому розмежуванні серверної логіки

та клієнтських інтерфейсів, що забезпечує масштабованість, надійність і гнучкість системи. Взаємодія модулів через визначені інтерфейси дозволяє підтримувати різні платформи, централізовано керувати даними та гарантувати стабільне функціонування процесу організації й управління тестуванням у різних умовах експлуатації.

3.2 Функціонування програмного забезпечення для організації та управління тестуванням

Функціонування програмного забезпечення для організації та управління тестуванням подамо за допомогою схеми, зображеної на рис. 3.2.

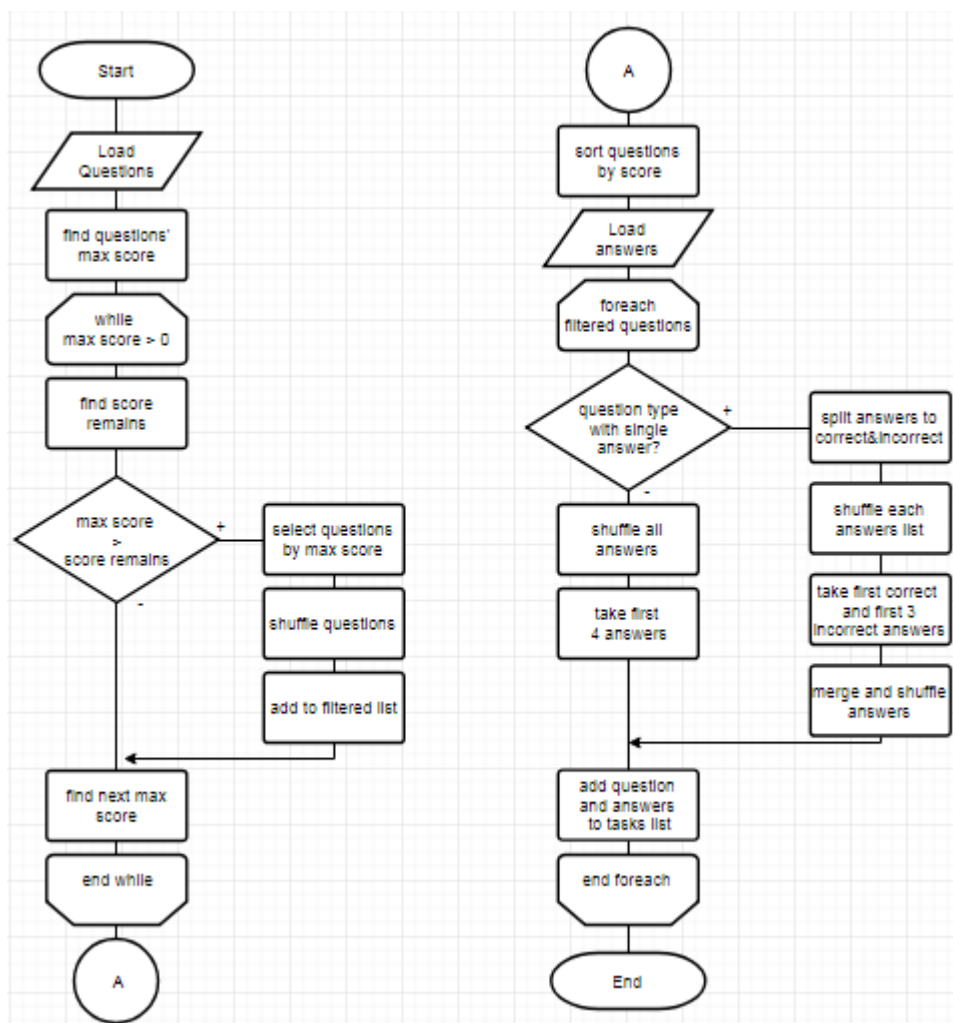


Рисунок 3.2 – Схема функціонування програмного забезпечення для організації та управління тестуванням

Функціонування програмного забезпечення для організації та управління тестуванням базується на реалізації ключових механізмів безпеки доступу та інтелектуального формування індивідуальних варіантів завдань. У межах прикладної логіки системи було реалізовано два базові алгоритмічні напрями: механізм підтвердження особи користувача та процедуру автоматизованого конструювання тесту відповідно до заданих параметрів.

Під час створення облікового запису введений користувачем пароль не зберігається у відкритому вигляді. На етапі передавання реєстраційних даних серверна частина ініціює перетворення пароля у хеш-значення із застосуванням відповідного алгоритму. Отриманий результат фіксується у базі даних як єдине представлення пароля. Таким чином, навіть у разі несанкціонованого доступу до сховища даних безпосереднє відтворення первинного тексту пароля ускладнюється. Під час входу до системи введена користувачем комбінація символів проходить аналогічну процедуру хешування, після чого відбувається зіставлення обчисленого значення із записом, що зберігається у базі. Співпадіння результатів підтверджує автентичність облікового запису, а невідповідність блокує доступ до функціоналу.

Після успішної перевірки облікових даних формується спеціальний ідентифікаційний маркер, який використовується для подальшої взаємодії клієнта із сервером. До його складу входять ідентифікатор користувача у базі даних, логін та хешоване значення пароля. Сукупність цих параметрів знову піддається криптографічному перетворенню за тим самим алгоритмом, після чого до сформованого результату додається ідентифікатор користувача з метою зменшення ймовірності колізій. Створений токен передається клієнтському застосунку і надалі супроводжує всі запити, виконуючи функцію цифрового підтвердження особи без необхідності повторного введення пароля.

Іншим ключовим аспектом роботи системи є алгоритм побудови варіанта тесту. Після вибору конкретного тесту за його ідентифікатором із

бази даних завантажується перелік пов'язаних із ним запитань. На початковому етапі визначається запитання з найвищим рівнем складності або максимальною кількістю балів. Далі розпочинається ітеративний процес відбору, під час якого система аналізує, скільки балів необхідно набрати для формування повного варіанта відповідно до параметрів тесту. Якщо поточний рівень складності не перевищує залишкову кількість балів, до набору включаються випадково обрані запитання цього рівня. Після цього допустимий рівень складності знижується, і процедура повторюється до моменту, коли подальший відбір стає недоцільним. Такий підхід дозволяє сформувати збалансований варіант із урахуванням різних рівнів складності.

Після завершення добору запитання впорядковуються за зростанням або спаданням складності, що забезпечує логічну послідовність виконання. Для кожного запитання додатково завантажуються відповідні варіанти відповідей. Подальша обробка залежить від типу завдання. Якщо запитання передбачає кілька правильних відповідей, перелік варіантів випадково перемішується, після чого відбираються перші чотири позиції, що теоретично може призвести до відсутності правильної відповіді серед запропонованих. Якщо ж умовою передбачено лише одну правильну відповідь, система розділяє варіанти на коректні та некоректні, окремо перемішує кожну групу та формує підсумковий набір із одного правильного і трьох неправильних варіантів, гарантуючи наявність єдиного коректного вибору.

Сформований тестовий пакет передається клієнтові разом із фіксацією часу початку виконання. Цей часовий маркер надалі використовується для контролю тривалості проходження, обчислення результатів і формування статистики. Взаємодія клієнтської та серверної частин відбувається у режимі обміну структурованими даними, що забезпечує цілісність інформації та стабільність роботи всієї системи.

Отже, функціонування програмного забезпечення поєднує механізми захищеної автентифікації та алгоритмічно обґрунтованого формування тестових варіантів, що забезпечує одночасно безпеку доступу та

індивідуалізацію оцінювання. Така організація логіки роботи сприяє надійності, адаптивності та ефективності процесу управління тестуванням у різних умовах експлуатації.

3.3 Розробка бази даних програмного забезпечення для організації та управління тестуванням

Базу даних програмного забезпечення для організації та управління тестуванням розроблено як структуроване реляційне сховище, призначене для впорядкованого збереження облікових записів користувачів, параметрів тестів, наборів запитань, варіантів відповідей та результатів проходження. Логічна модель цієї бази даних сформована з урахуванням забезпечення цілісності даних, мінімізації дублювання інформації та підтримки зв'язків між сутностями різних рівнів.

Схему бази даних програмного забезпечення для організації та управління тестуванням наведено на рис. 3.3.

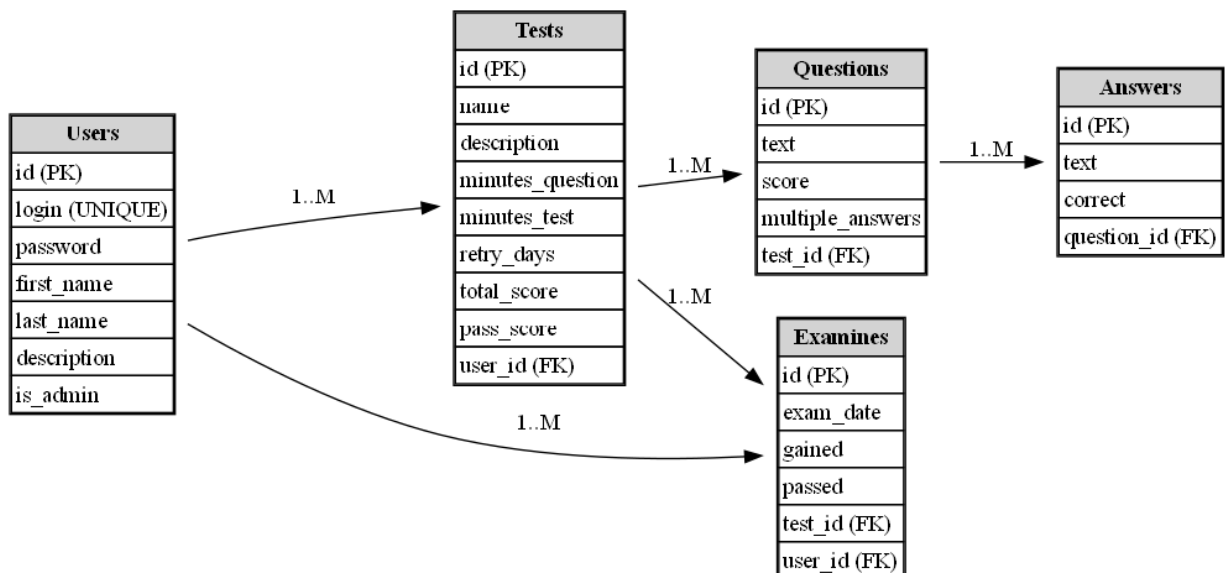


Рисунок 3.3 – Схема бази даних програмного забезпечення для організації та управління тестуванням

Центральним елементом моделі бази даних програмного забезпечення для організації та управління тестуванням є таблиця Users, у якій акумулюються відомості про зареєстрованих учасників системи. Для кожного запису передбачається унікальний ідентифікатор, що використовується як первинний ключ і забезпечує однозначну ідентифікацію користувача. Атрибут login визначає ім'я облікового запису та підлягає обмеженню унікальності, що унеможливило б повторну реєстрацію з однаковими даними доступу. Поле password містить не сам пароль, а його криптографічне представлення, що застосовується під час процедури автентифікації. Додаткові атрибути first_name, last_name та description дозволяють зберігати персональні відомості та довільну характеристику користувача. Логічне поле is_admin визначає рівень привілеїв і використовується для розмежування доступу до адміністративних функцій. Надання відповідних прав передбачається лише користувачем із чинним статусом адміністратора, що реалізує механізм ієрархії повноважень.

З таблицею користувачів пов'язується таблиця Tests, яка відображає створені в системі тести. Кожен тест має власний первинний ключ id та містить назву і текстовий опис, що дозволяє ідентифікувати його змістове призначення. Параметри minutes_question і minutes_test визначають часові обмеження відповідно для окремого завдання та для тесту загалом. Поле retry_days встановлює мінімальний інтервал між повторними спробами складання, що забезпечує контроль частоти проходження. Атрибути total_score і pass_score задають максимальну суму балів та поріг успішності. Зовнішній ключ user_id пов'язує кожен тест із конкретним користувачем-автором, формуючи відношення «один-до-багатьох», у межах якого один користувач може створити декілька тестів.

Подальша деталізація структури тестів реалізується через таблицю Questions, у якій зберігаються запитання, пов'язані з конкретним тестом. Кожен запис має унікальний ідентифікатор та текст запитання. Поле score визначає кількість балів, що нараховується за правильне виконання. Атрибут

`multiple_answers` відображає тип запитання, дозволяючи фіксувати, чи передбачено одну правильну відповідь, чи декілька. Зовнішній ключ `test_id` встановлює зв'язок із таблицею `Tests`, утворюючи ієрархію, за якої одному тесту відповідає множина запитань.

Для зберігання конкретних варіантів відповідей використовується таблиця `Answers`. Кожен варіант має власний первинний ключ, текстове представлення та логічний показник `correct`, що сигналізує про його правильність. Через зовнішній ключ `question_id` реалізується зв'язок із таблицею `Questions`, що формує ще одне відношення «один-до-багатьох»: одному запитанню відповідає декілька варіантів відповідей. Така побудова дозволяє гнучко змінювати набір відповідей без порушення цілісності структури тесту.

Окреме місце у логічній моделі займає таблиця `Examines`, призначена для фіксації фактів проходження тестів користувачами. У цій таблиці зберігається дата початку спроби, кількість отриманих балів та індикатор успішності. Поля `test_id` і `user_id` виконують роль зовнішніх ключів, що пов'язують спробу з конкретним тестом і конкретним користувачем. У результаті формується відношення «один-до-багатьох» як з боку `Tests`, так і з боку `Users`. Водночас ця таблиця виконує функцію проміжної сутності, яка логічно реалізує зв'язок «багато-до-багатьох» між користувачами та тестами, оскільки один користувач може проходити багато тестів, а один тест може складатися багатьма користувачами.

У межах такої структури бази даних програмного забезпечення для організації та управління тестуванням забезпечується цілісність через використання первинних і зовнішніх ключів, а також через визначення відповідних обмежень унікальності та зв'язків між таблицями. Логічна модель дозволяє ефективно виконувати запити щодо формування тестів, обчислення статистики, аналізу результатів і контролю доступу. Дані організовано таким чином, щоб підтримувати як оперативні транзакції під час проходження тестування, так і аналітичні операції, пов'язані з оцінюванням успішності.

Отже, база даних програмного забезпечення для організації та управління тестуванням формує структуроване реляційне середовище, у якому всі ключові сутності системи пов'язані між собою через чітко визначені відношення. Така модель забезпечує узгодженість інформації, гнучкість розширення функціоналу та можливість ефективного управління процесами тестування в межах єдиної інформаційної системи.

3.4 Проєктування інтерфейсу програмного забезпечення для організації та управління тестуванням

Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для організації та управління тестуванням виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення для організації та управління тестуванням враховані функціональні вимоги до програми:

- підтримка можливості реєстрації та авторизації користувачів із розмежуванням прав доступу відповідно до ролей у системі;
- підтримка можливості зберігання даних про користувачів у базі даних із забезпеченням цілісності інформації;
- можливість створення, налаштування та адміністрування тестів із визначенням параметрів проходження та критеріїв оцінювання;
- підтримка формування та редагування банку запитань із можливістю встановлення різних типів відповідей і вагових коефіцієнтів оцінювання;
- можливість автоматичного обчислення результатів тестування з урахуванням встановлених правил та збереження отриманих показників у базі даних;
- можливість обмеження доступу до тестів відповідно до встановлених правил і параметрів повторного проходження;
- можливість взаємодії клієнтської та серверної частин системи через програмний інтерфейс із забезпеченням безпечної передачі даних;

- підтримка масштабованості системи для забезпечення одночасної роботи значної кількості користувачів без втрати продуктивності;

- підтримка коректного відображення інтерфейсу в настільному та мобільному клієнтах із забезпеченням зручності використання програмного забезпечення.

Інтерфейс головної форми програмного забезпечення для організації та управління тестуванням у режимі звичайного користувача наведено на рис. 3.4.

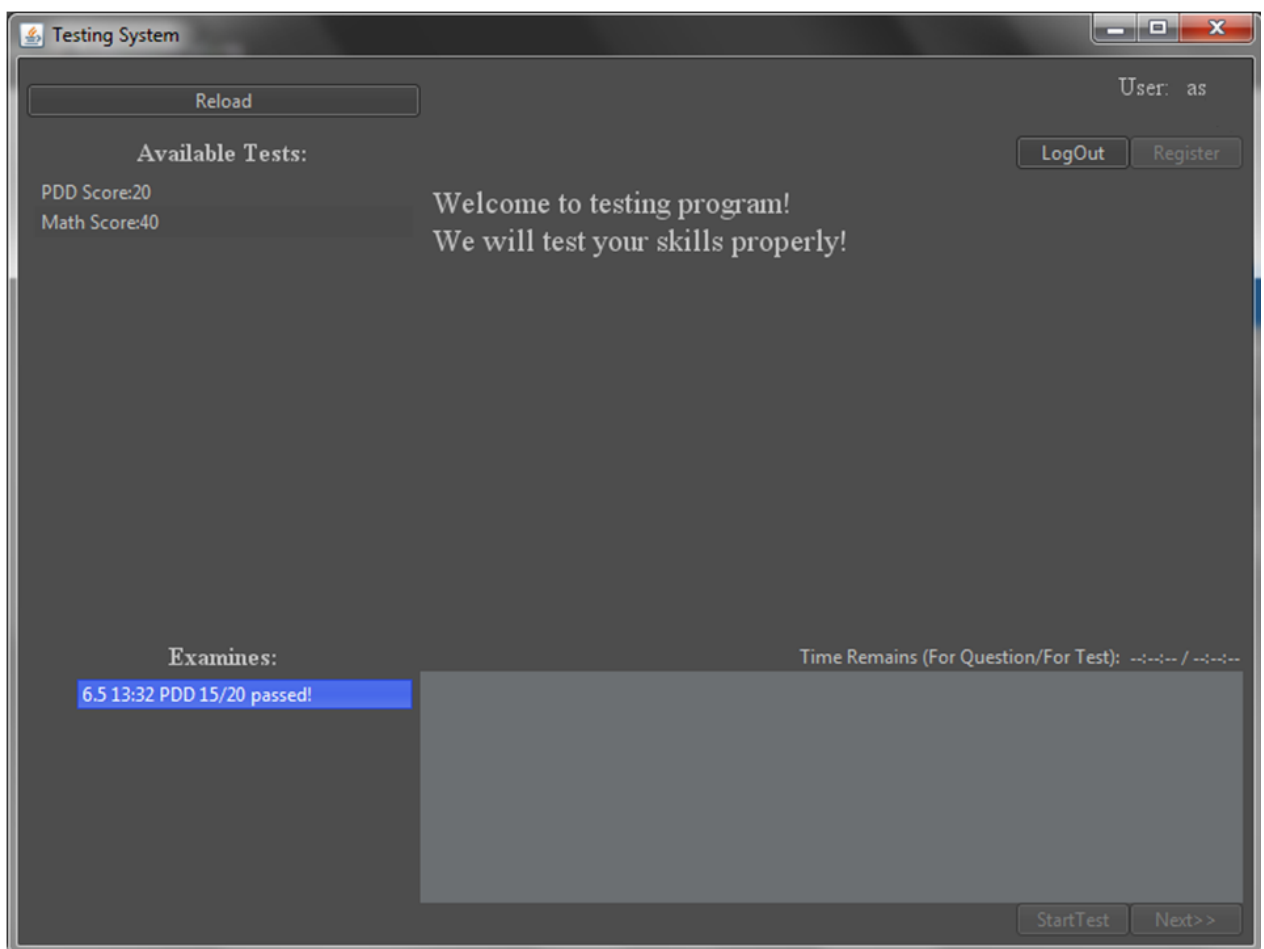


Рисунок 3.4 – Інтерфейс головної форми програмного забезпечення для організації та управління тестуванням у режимі звичайного користувача

Інтерфейс головної форми програмного забезпечення для організації та управління тестуванням у режимі адміністратора наведено на рис. 3.5.

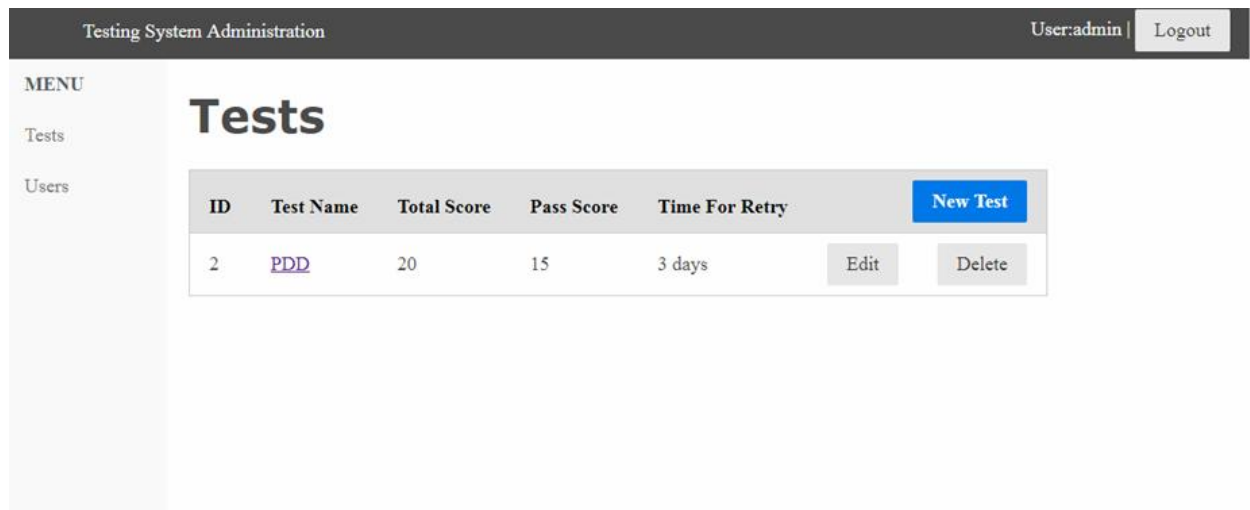


Рисунок 3.5 – Інтерфейс головної форми програмного забезпечення для організації та управління тестуванням у режимі адміністратора

Сторінку з прикладом проходження тестування користувачем наведено на рис. 3.6.

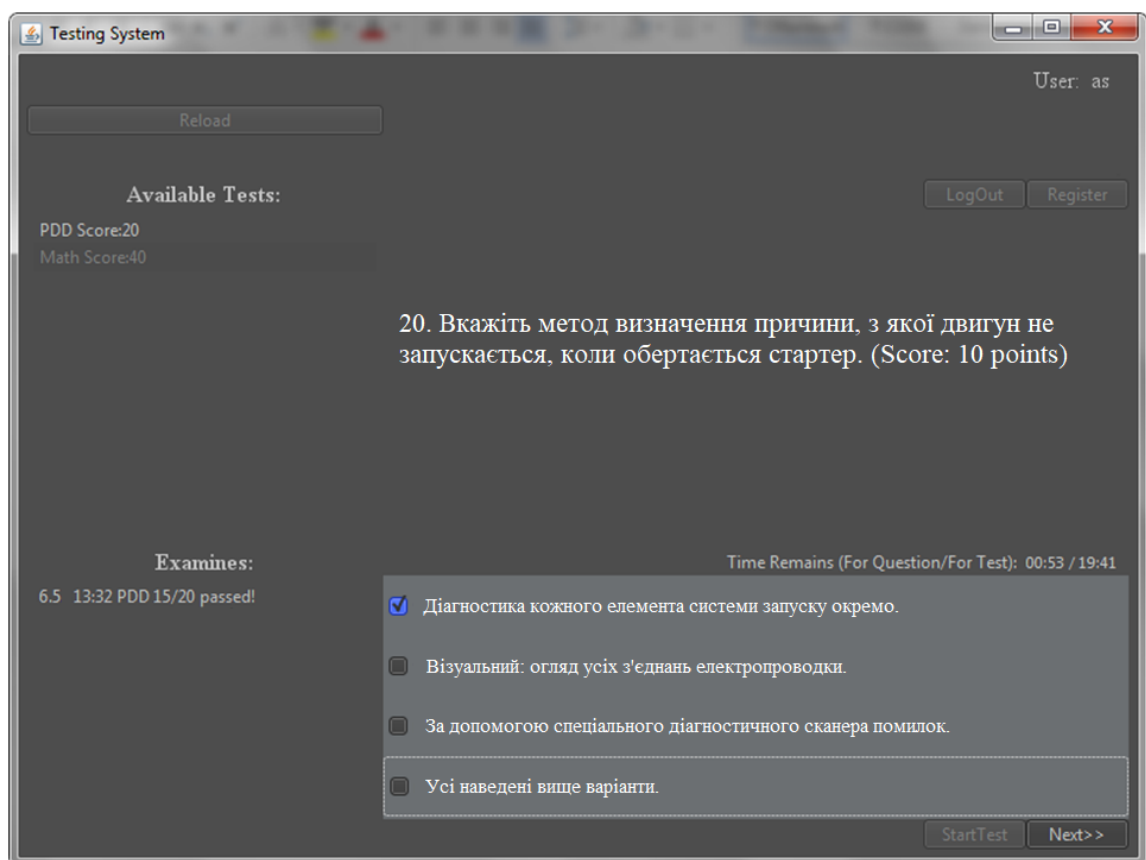


Рисунок 3.6 – Сторінка з прикладом проходження тестування

Сторінку з прикладом подання результатів тестування наведено на рис. 3.7.

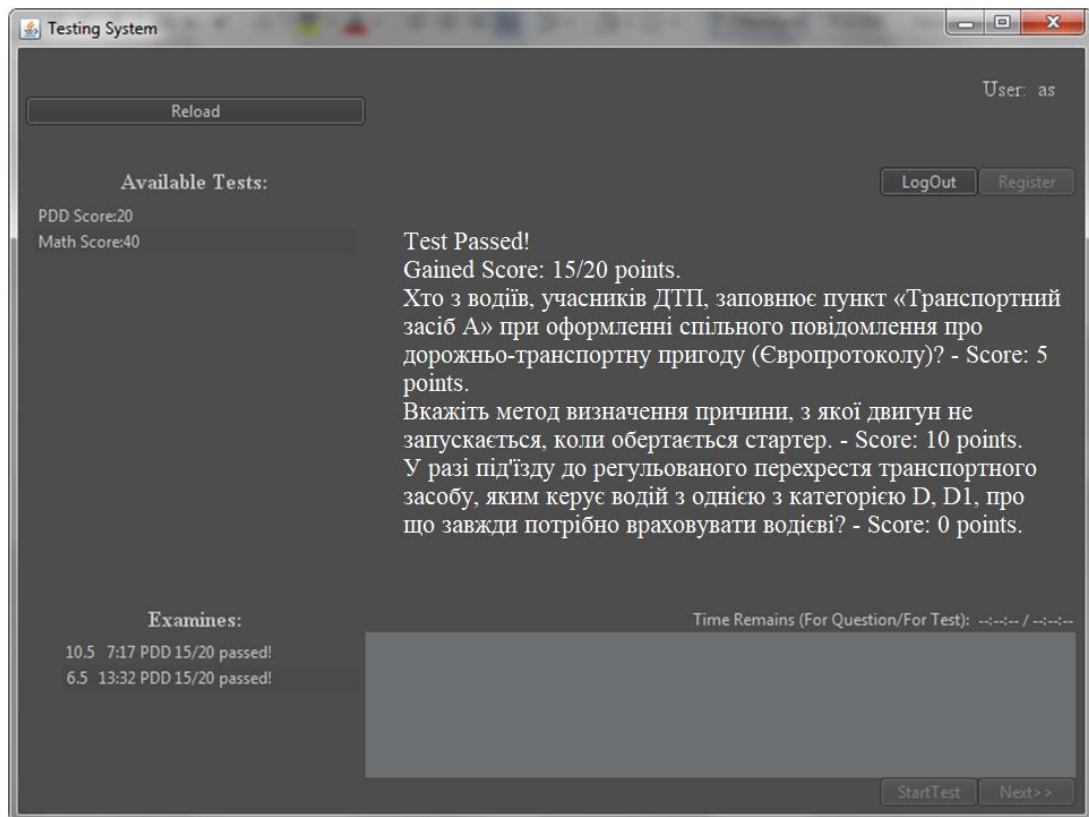


Рисунок 3.7 – Сторінка з прикладом подання результатів тестування

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для організації та управління тестуванням.

Запропоновано структуру програмного забезпечення для організації та управління тестуванням. Визначено, що структура програмного забезпечення для організації та управління тестуванням базується на чіткому розмежуванні серверної логіки та клієнтських інтерфейсів, що забезпечує масштабованість, надійність і гнучкість системи. Взаємодія модулів через визначені інтерфейси дозволяє підтримувати різні платформи, централізовано керувати даними та гарантувати стабільне функціонування процесу організації й управління тестуванням у різних умовах експлуатації.

Описано функціонування програмного забезпечення для організації та управління тестуванням. Визначено, що функціонування програмного забезпечення поєднує механізми захищеної автентифікації та алгоритмічно обґрунтованого формування тестових варіантів, що забезпечує одночасно безпеку доступу та індивідуалізацію оцінювання. Така організація логіки роботи сприяє надійності, адаптивності та ефективності процесу управління тестуванням у різних умовах експлуатації.

Розроблено базу даних програмного забезпечення для організації та управління тестуванням як структуроване реляційне сховище, призначене для впорядкованого збереження облікових записів користувачів, параметрів тестів, наборів запитань, варіантів відповідей та результатів проходження. Логічна модель цієї бази даних сформована з урахуванням забезпечення цілісності даних, мінімізації дублювання інформації та підтримки зв'язків між сутностями різних рівнів.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для організації та управління тестуванням.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для організації та управління тестуванням. Програма написана на мові Kotlin, що є сучасною, еволюційно розвиненою мовою, яка поєднує лаконічність синтаксису, безпечність типізації, мультиплатформеність і глибоку інтеграцію з усталеними технологічними екосистемами, що дозволяє розробляти програми, які характеризуються високою надійністю, масштабованістю, інтеграційними можливостями та зручністю супроводу. В якості середовища розробки для створення програмного забезпечення для організації та управління тестуванням обрано IntelliJ IDEA через комплексну інтеграцію з мовою Kotlin і можливість ефективно реалізовувати всі етапи життєвого циклу розробки програмного продукту.

Програмне забезпечення для організації та управління тестуванням забезпечує виконання таких функцій:

- підтримка можливості реєстрації та авторизації користувачів із розмежуванням прав доступу відповідно до ролей у системі;
- підтримка можливості зберігання даних про користувачів у базі даних із забезпеченням цілісності інформації;
- можливість створення, налаштування та адміністрування тестів із визначенням параметрів проходження та критеріїв оцінювання;
- підтримка формування та редагування банку запитань із можливістю встановлення різних типів відповідей і вагових коефіцієнтів оцінювання;
- можливість автоматичного обчислення результатів тестування з урахуванням встановлених правил та збереження отриманих показників у базі даних;
- можливість обмеження доступу до тестів відповідно до встановлених правил і параметрів повторного проходження;

- можливість взаємодії клієнтської та серверної частин системи через програмний інтерфейс із забезпеченням безпечної передачі даних;
- підтримка масштабованості системи для забезпечення одночасної роботи значної кількості користувачів без втрати продуктивності;
- підтримка коректного відображення інтерфейсу в настільному та мобільному клієнтах із забезпеченням зручності використання програмного забезпечення.

Для роботи програмного забезпечення для організації та управління тестуванням потрібні такі технічні та програмні ресурси: процесор з тактовою частотою не нижче 2,4 ГГц із підтримкою багатопоточності та наявністю щонайменше чотирьох ядер, що дозволяє ефективно обробляти паралельні HTTP-запити та транзакції. Обсяг оперативної пам'яті для сервера має становити не менше 8 ГБ, однак для систем із підвищеним навантаженням рекомендовано 16 ГБ і більше, що забезпечить коректну роботу сервера застосунків та системи керування базами даних без зниження продуктивності.

Для зберігання даних необхідним є наявність твердотілого накопичувача обсягом не менше 256 ГБ, що гарантує достатній простір для розміщення самої системи, журналів подій, резервних копій і бази даних. Мережева інфраструктура має забезпечувати стабільне підключення до мережі з пропускнуою здатністю не нижче 100 Мбіт/с для серверного вузла, що дозволяє обслуговувати клієнтів без затримок при передачі даних.

Клієнтська частина програмного забезпечення для організації та управління тестуванням може функціонувати на персональних комп'ютерах із процесором не нижче 2,0 ГГц та оперативною пам'яттю від 4 ГБ, що є достатнім для роботи настільного застосунку або сучасного веббраузера. Для мобільних пристроїв мінімальні вимоги передбачають наявність процесора та оперативної пам'яті від 3 ГБ, а також актуальної версії мобільної операційної системи. Дисплей із роздільною здатністю не нижче 1280×720 пікселів забезпечує коректне відображення інтерфейсу та елементів керування.

Програмне забезпечення серверної частини передбачає встановлення операційної системи сімейства Windows або Linux із підтримкою сучасних засобів мережевої безпеки та багатокористувацького режиму. Необхідною є наявність встановленого середовища виконання Java Virtual Machine відповідної версії, що забезпечує коректну роботу серверних компонентів, розроблених із використанням мови програмування Kotlin. Також потребується система керування базами даних, яка підтримує реляційну модель та механізми транзакційності, резервного копіювання і відновлення даних.

Для клієнтської частини необхідною є підтримка веббраузера з актуальними оновленнями або встановлене середовище виконання настільного застосунку. Додатково передбачається наявність стабільного підключення до мережі Інтернет для синхронізації з сервером та передачі результатів тестування. Використання антивірусного програмного забезпечення та засобів захисту мережі підвищує рівень інформаційної безпеки системи та мінімізує ризики несанкціонованого доступу до персональних даних користувачів.

Функціональні характеристики розробленого програмного забезпечення для організації та управління тестуванням наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для організації та управління тестуванням наведено у додатку Б.

4.2 Характеристики програми для організації та управління тестуванням

Програмне забезпечення для організації та управління тестуванням характеризується комплексністю, модульністю та орієнтацією на автоматизацію всіх ключових етапів контролю знань. У його основі закладається цілісна архітектура, що забезпечує взаємодію між інтерфейсом користувача, серверною логікою та системою зберігання даних. Такий підхід

дозволяє гарантувати узгодженість операцій, надійність обробки результатів і стабільність функціонування в умовах одночасного доступу багатьох користувачів.

Програмне забезпечення для організації та управління тестуванням відзначається гнучкістю налаштувань, що дає змогу адаптувати параметри тестування відповідно до потреб освітнього чи корпоративного середовища. Передбачається можливість конфігурування тривалості проходження тестів, визначення критеріїв оцінювання, встановлення правил повторного складання та формування індивідуальних сценаріїв перевірки знань. Завдяки цьому забезпечується варіативність використання програмного продукту без необхідності змінювати його внутрішню структуру.

Важливою характеристикою програмного забезпечення для організації та управління тестуванням є безпека обробки інформації. Реалізовано механізми аутентифікації та авторизації користувачів, розмежування прав доступу та захисту персональних даних. Передбачено контроль цілісності результатів тестування та запобігання несанкціонованому втручанню в базу даних.

Функціональні можливості охоплюють створення та редагування тестів, формування банку запитань із підтримкою різних типів відповідей, автоматичне оцінювання результатів і генерацію аналітичних звітів. Забезпечується накопичення статистичних даних, що дозволяє аналізувати успішність користувачів, виявляти типові помилки та оцінювати ефективність навчального процесу. Інтерфейс орієнтується на інтуїтивність і зручність, що зменшує час адаптації користувачів до системи.

Серед технічних характеристик відзначається масштабованість, яка дає змогу розширювати систему відповідно до зростання кількості користувачів або обсягу тестових матеріалів. Архітектурні рішення підтримують інтеграцію з іншими інформаційними ресурсами, що відкриває можливості для включення програмного забезпечення до ширшого цифрового середовища

установи. Висока продуктивність забезпечується оптимізацією обробки запитів і раціональною організацією зберігання даних.

Отже, програмне забезпечення для організації та управління тестуванням вирізняється поєднанням функціональної повноти, адаптивності, безпечності та технічної стійкості, що робить його ефективним інструментом контролю та аналізу результатів навчальної діяльності.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення для організації та управління тестуванням на серверній частини необхідно запустити відповідний файл. Для звертання до клієнтської сторони необхідно перейти на відповідну вебсторінку.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для організації та управління тестуванням є облікові дані користувачів, що вводяться під час автентифікації, інформація про створювані тести, тексти запитань, варіанти відповідей та параметри оцінювання. До вхідної інформації також належать результати вибору відповідей під час проходження тестування, налаштування тривалості виконання завдань, правила допуску до повторного складання та інші параметри, що визначають логіку функціонування системи.

Вихідними даними програмного забезпечення для організації та управління тестуванням є результати проходження тестів, інформація про успішність виконання завдань. Система формує підсумкові оцінки, відображає правильні та неправильні відповіді відповідно до встановлених правил, а також надає узагальнену інформацію. Вихідні дані відображаються у вигляді інтерактивних сторінок інтерфейсу.

4.3.3 Повідомлення

Користувач програмного забезпечення для організації та управління тестуванням може отримати такі повідомлення: повідомлення про успішну або неуспішну автентифікацію, підтвердження збереження змін, сповіщення про початок або завершення тестування, попередження про помилки введення даних. Також можуть відображатися повідомлення про відсутність доступу до певних функцій, необхідність оновлення сторінки або виникнення технічних збоїв під час обробки запиту. Окремо передбачаються інформаційні сповіщення щодо результатів перевірки знань.

4.4 Виконання програмного забезпечення для організації та управління тестуванням

Після запуску програмного забезпечення для організації та управління тестуванням користувачу відображається основне вікно інтерфейсу (рис. 4.1), у якому подається привітальне повідомлення, перелік тестів, доступних для проходження, а також інформація про раніше виконані спроби. У цьому ж вікні розміщуються елементи керування для створення облікового запису та входу до системи. До моменту проходження автентифікації функціональні можливості залишаються обмеженими, що забезпечує захист даних і розмежування доступу.

Після вибору команди входу (кнопка Login) відкривається форма (рис. 4.2), у якій необхідно ввести облікові дані для підтвердження особи користувача.

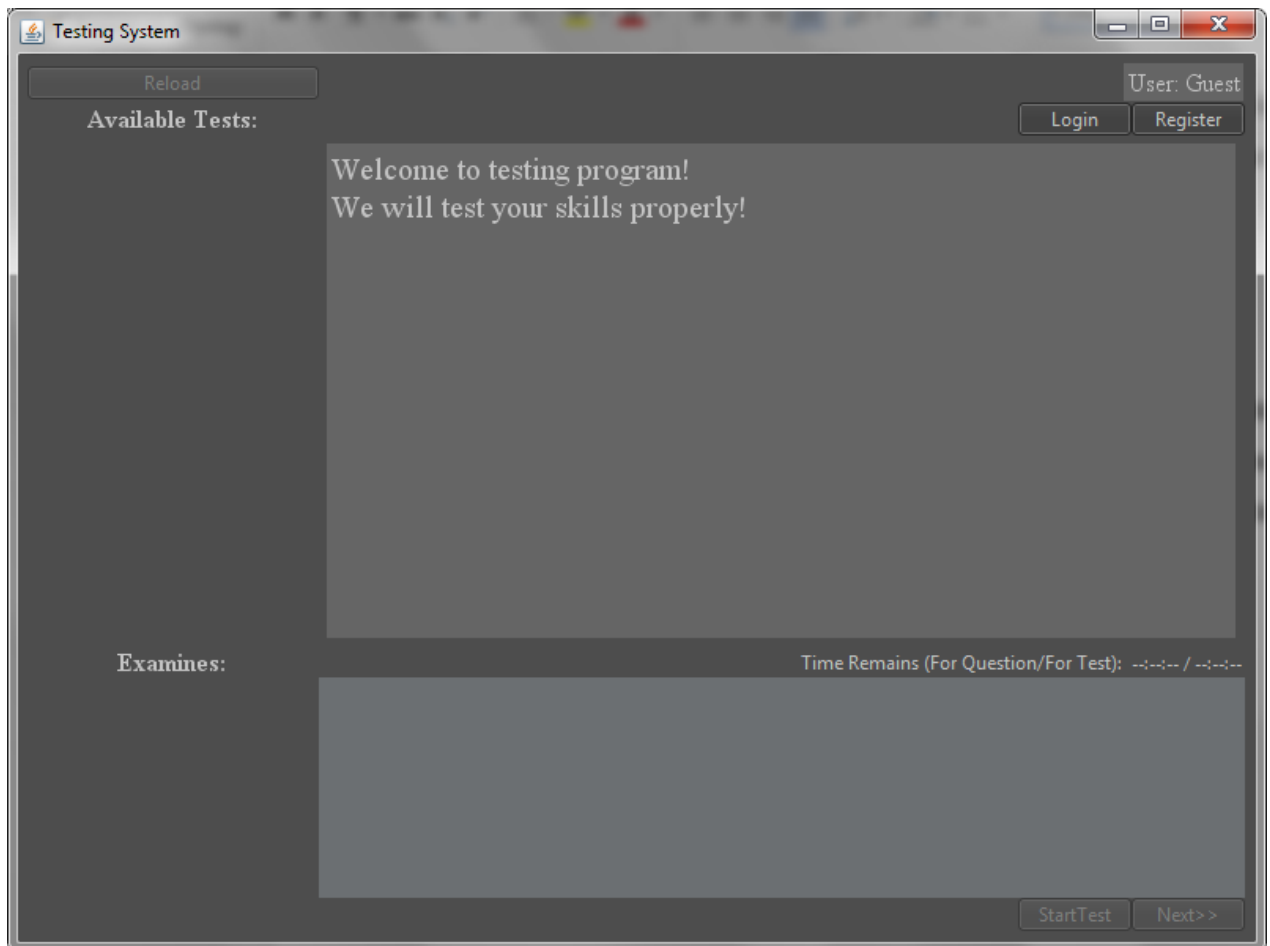


Рисунок 4.1 – Головна форма програмного забезпечення для організації та управління тестуванням

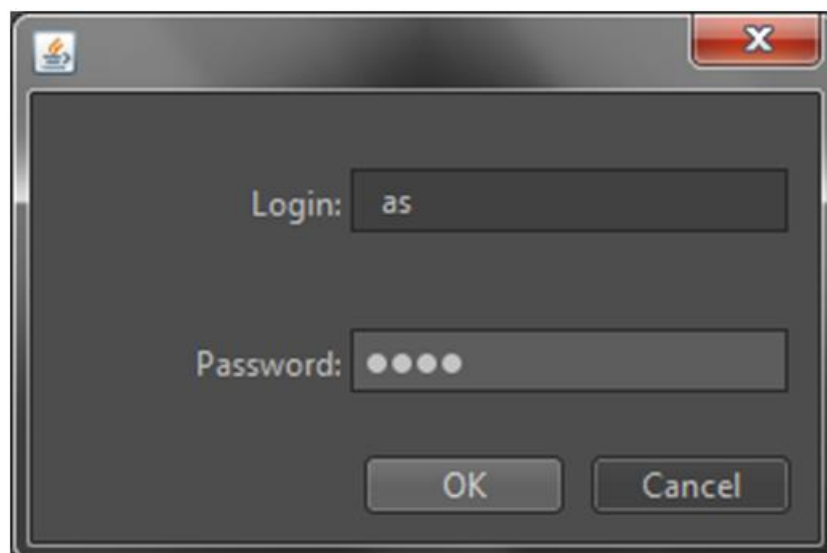


Рисунок 4.2 – Форма авторизації

Після успішної авторизації здійснюється автоматичне завантаження актуального переліку тестів і відображення інформації про попередні результати (рис. 4.3).

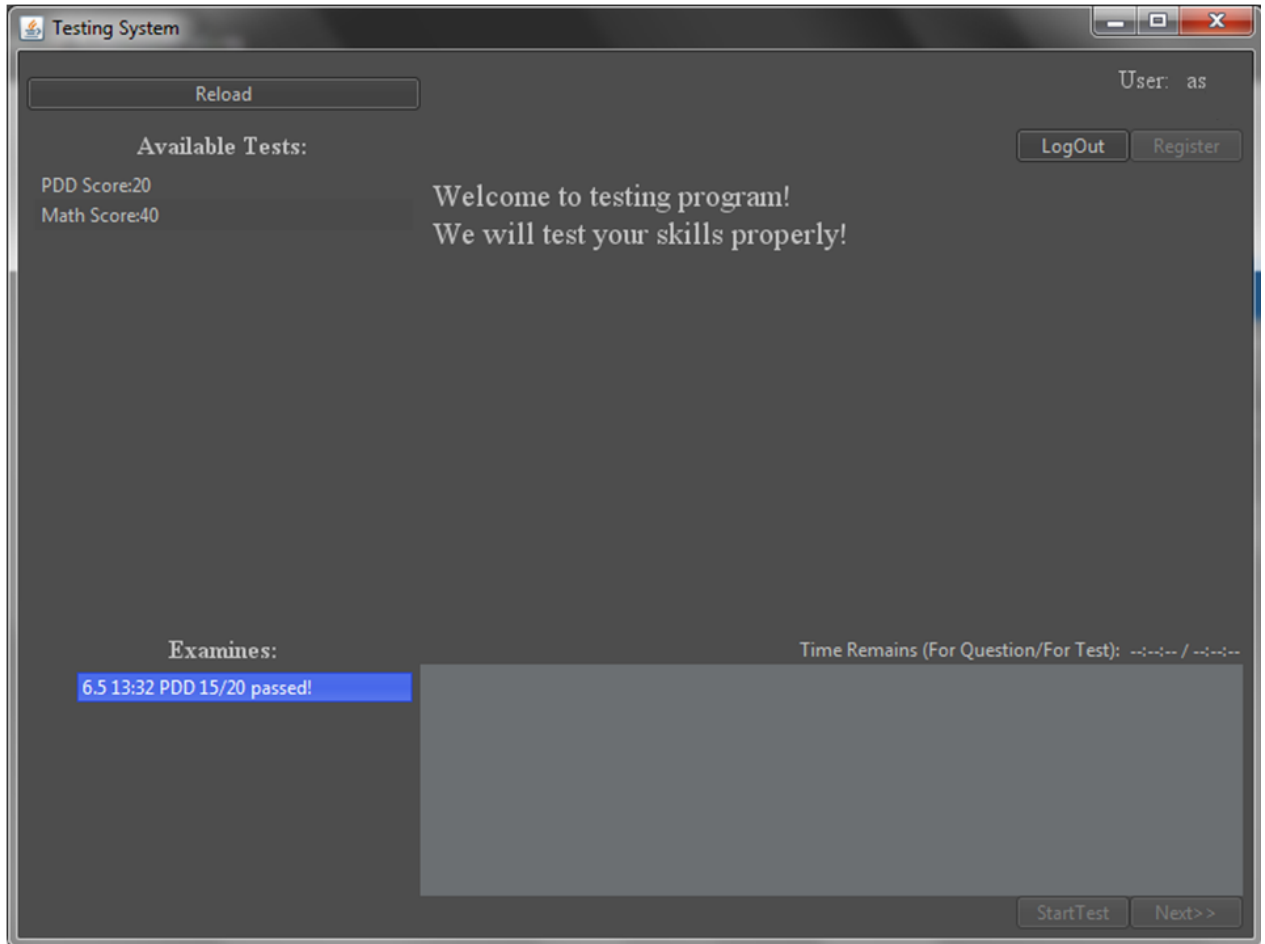


Рисунок 4.3 – Головна форма програми після авторизації

У разі вибору конкретного тесту на центральній панелі інтерфейсу демонструється його опис та основні параметри (рис. 4.4).

Ініціювання проходження тесту супроводжується завантаженням запитань і запуском механізму контролю часу, який обмежує тривалість надання відповіді (рис. 4.5). Під час виконання завдань обрана відповідь фіксується системою, після чого відображається наступне запитання.

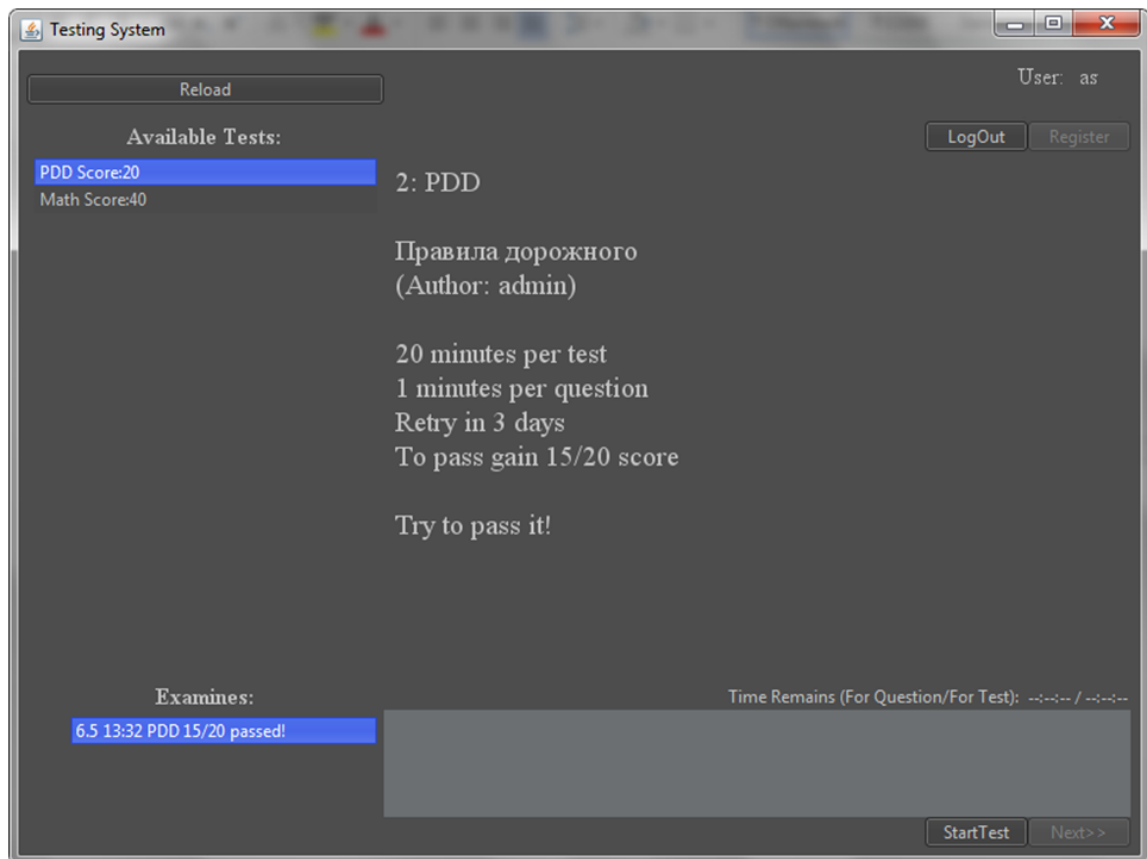


Рисунок 4.4 – Інформація про тестування

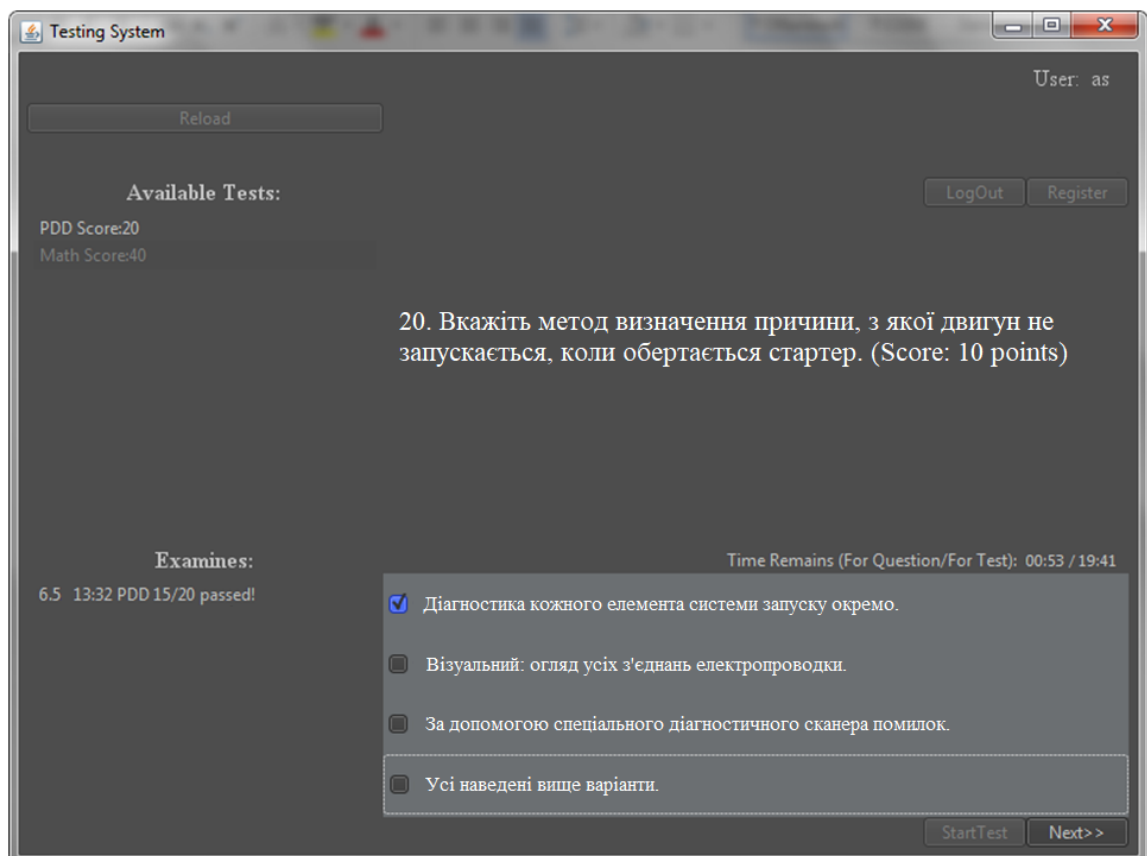


Рисунок 4.5 – Проходження тесту користувачем

По завершенні процесу всі відповіді передаються на сервер для автоматизованої перевірки, а користувачу надається детальний підсумок із відображенням отриманих балів та загального результату (рис. 4.6). Після завершення тестування повторний доступ до цього ж тесту тимчасово блокується відповідно до встановлених правил.

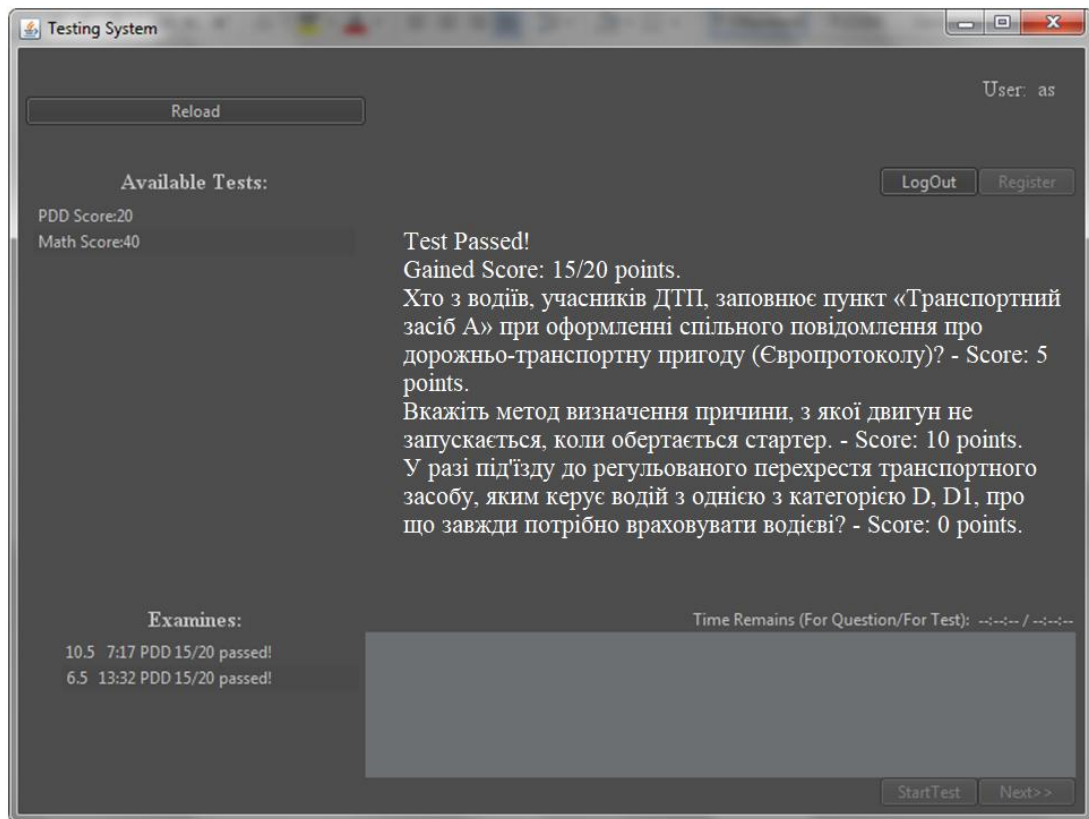


Рисунок 4.6 – Результати тестування

Мобільна версія застосунку зберігає повний обсяг функціональних можливостей настільного клієнта, однак інтерфейс адаптовано до особливостей портативних пристроїв. Відображення інформації структуровано таким чином, щоб забезпечити зручну навігацію та оптимальне використання екранного простору.

Для переходу до адміністративної частини використовується відповідна адреса застосунку у веббраузері. Після входу до панелі керування відкривається доступ до інструментів створення нових тестів і редагування наявних (рис. 4.7).

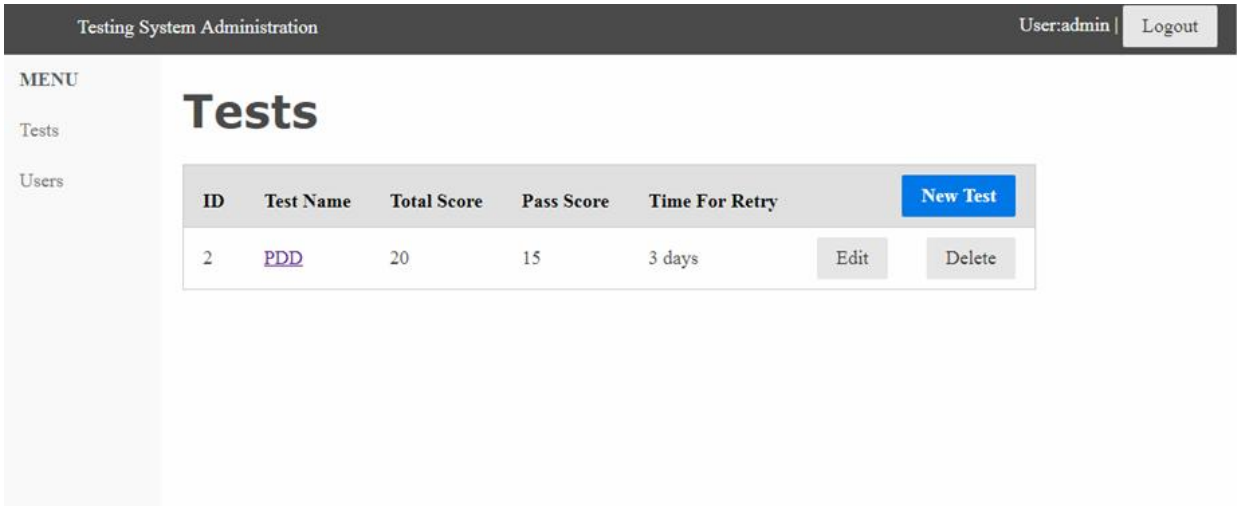


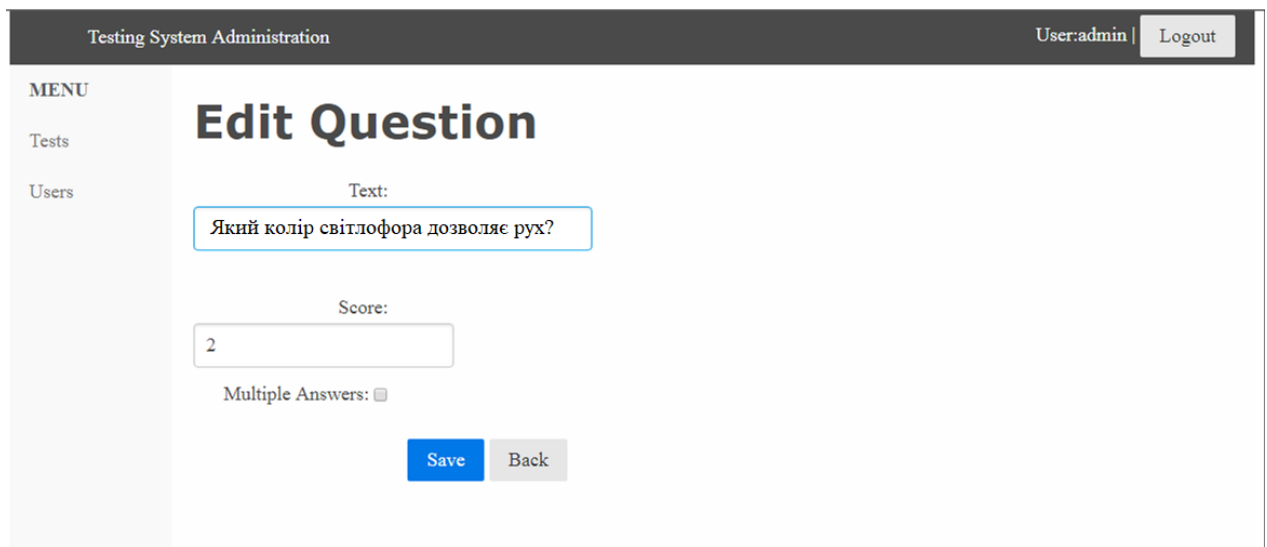
Рисунок 4.7 – Головна форма програми у режимі адміністратора

Перегляд детальної інформації про тест здійснюється через вибір його назви, що відкриває сторінку з описом та переліком пов’язаних запитань. У межах цієї сторінки передбачено можливість додавання нових запитань, внесення змін до вже створених або їх видалення (рис. 4.8).

The screenshot shows the 'New Test' form within the 'Testing System Administration' application. The header and sidebar are identical to the previous screenshot. The main content area is titled 'New Test'. It contains several input fields with labels: 'Name:' (with the value 'Криптограф'), 'Total Score:' (with the value '100'), 'Pass limit:' (with the value '65'), 'Time per Test:' (with the value '45' and the unit 'minutes' below it), 'Time per Question:' (with the value '5' and the unit 'minutes' below it), and 'Retry Days:' (with the value '30' and the unit 'days' below it). There is also a 'Description:' label followed by a large text area. At the bottom of the form are two buttons: 'Save' and 'Back'.

Рисунок 4.8 – Створення нового тесту

Для кожного запитання доступний перегляд і коригування варіантів відповідей через спеціалізовані форми введення (рис. 4.9).



The screenshot shows the 'Edit Question' form in the 'Testing System Administration' interface. The top header bar is dark grey with 'Testing System Administration' on the left and 'User:admin | Logout' on the right. A left sidebar contains a 'MENU' section with 'Tests' and 'Users' links. The main content area has the title 'Edit Question' in large bold font. Below the title, there is a 'Text:' label followed by a text input field containing the question 'Який колір світлофора дозволяє рух?'. Below this is a 'Score:' label followed by a numeric input field containing the value '2'. Underneath the score field is a 'Multiple Answers:' label with an unchecked checkbox. At the bottom of the form are two buttons: a blue 'Save' button and a grey 'Back' button.

Рисунок 4.9 – Створення та редагування питання до тесту

Крім того, у розділі тесту відображаються результати проходження, що дає змогу аналізувати успішність користувачів. Користувач із розширеними правами має змогу працювати з переліком зареєстрованих осіб, переглядати їхні профілі та вносити необхідні зміни через відповідний розділ адміністративного меню.

4.5 Тестування програмного забезпечення для організації та управління тестуванням

Виконано тестування програмного забезпечення для організації та управління тестуванням.

Виявлено, що програма для організації та управління тестуванням функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу підтримки процесу тестування.

Розроблене програмне забезпечення дозволяє забезпечувати автоматизацію процесів, пов'язаних з підтримкою процесу тестування здобувачів закладів освіти.

4.6 Висновки за розділом 4

Виконано тестування розробленого програмного забезпечення для організації та управління тестуванням. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу тестування.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розробки програмного забезпечення для організації та управління тестуванням.

Визначено, що програмне забезпечення для організації та управління тестуванням являє собою спеціалізований програмний інструмент, призначений для планування, проведення, контролю та аналізу процедур оцінювання знань, умінь або професійних компетентностей. Такі програмні системи створюють надають інформаційне середовище, у межах якого забезпечується підготовка тестових матеріалів, адміністрування доступу учасників, автоматизована перевірка результатів і формування аналітичних висновків. У результаті формується цілісний підхід до оцінювання, що поєднує методичну складову з технологічною підтримкою.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для організації та управління тестуванням. Проте впровадження деяких програм для організації та управління тестуванням пов'язано з високими витратами на ліцензування, складною інтеграцією з іншими програмними платформами, навчанням персоналу та адміністрації. За відсутності достатнього рівня підготовки користувачів знижується ефективність використання функціональних можливостей таких програмних систем, особливо у випадках, коли такі системи є досить складними, що призводить до формального застосування інструментів без повної реалізації їх потенціалу. Не менш важливим є питання захисту даних і забезпечення конфіденційності. У разі недостатнього рівня захисту існує ймовірність несанкціонованого доступу до результатів тестування або персональної інформації користувачів. Крім того, автоматизований контроль не повністю усуває можливість недоброчесної поведінки здобувачів освіти або інших учасників освітнього процесу, оскільки зберігається ризик використання сторонніх ресурсів чи технічних засобів під

час проходження тестів. Тому актуальною є розробка програмного забезпечення для організації та управління тестуванням.

Сформульовано функціональні вимоги до програмного забезпечення для організації та управління тестуванням.

Для реалізації програмного забезпечення для організації та управління тестуванням обрано мову програмування Kotlin, що є сучасною, еволюційно розвиненою мовою, яка поєднує лаконічність синтаксису, безпечність типізації, мультиплатформеність і глибоку інтеграцію з усталеними технологічними екосистемами, що дозволяє розробляти програми, які характеризуються високою надійністю, масштабованістю, інтеграційними можливостями та зручністю супроводу. Поєднання сучасних мовних конструкцій Kotlin, сумісності з усталеною інфраструктурою та орієнтації на безпечну розробку формує технологічну основу для створення ефективної та стійкої інформаційної системи.

Для створення програмного забезпечення для організації та управління тестуванням обрано середовище розробки IntelliJ IDEA через комплексну інтеграцію з мовою Kotlin і можливість ефективно реалізовувати всі етапи життєвого циклу розробки програмного продукту. IntelliJ IDEA надає широкий спектр інструментів, які спрощують написання, налагодження, тестування та підтримку коду, що особливо важливо для складних систем, які обробляють великі обсяги даних та одночасно працюють з багатьма користувачами.

Розроблено програмне забезпечення для організації та управління тестуванням.

Запропоновано структуру програмного забезпечення для організації та управління тестуванням. Визначено, що структура програмного забезпечення для організації та управління тестуванням базується на чіткому розмежуванні серверної логіки та клієнтських інтерфейсів, що забезпечує масштабованість, надійність і гнучкість системи. Взаємодія модулів через визначені інтерфейси дозволяє підтримувати різні платформи, централізовано керувати даними та

гарантувати стабільне функціонування процесу організації й управління тестуванням у різних умовах експлуатації.

Описано функціонування програмного забезпечення для організації та управління тестуванням. Визначено, що функціонування програмного забезпечення поєднує механізми захищеної автентифікації та алгоритмічно обґрунтованого формування тестових варіантів, що забезпечує одночасно безпеку доступу та індивідуалізацію оцінювання. Така організація логіки роботи сприяє надійності, адаптивності та ефективності процесу управління тестуванням у різних умовах експлуатації.

Розроблено базу даних програмного забезпечення для організації та управління тестуванням як структуроване реляційне сховище, призначене для впорядкованого збереження облікових записів користувачів, параметрів тестів, наборів запитань, варіантів відповідей та результатів проходження. Логічна модель цієї бази даних сформована з урахуванням забезпечення цілісності даних, мінімізації дублювання інформації та підтримки зв'язків між сутностями різних рівнів.

Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для організації та управління тестуванням.

Виконано тестування розробленого програмного забезпечення для організації та управління тестуванням. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою процесу тестування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. The Best Online Testing Platforms for Students. URL: <https://scholarly.so/blog/the-best-online-testing-platforms-for-students> (date of access: 02.05.2026).
2. School Testing Assesses Knowledge Gains and Gaps. URL: <https://www.thoughtco.com/the-purpose-of-tests-7688> (date of access: 02.05.2026).
3. The Power of Testing in Education. URL: <https://www.numberanalytics.com/blog/the-power-of-testing-in-education> (date of access: 02.05.2026).
4. Understanding Assessment Testing: A Comprehensive Guide for Educators and Students. URL: <https://www.learndogrow.org/post/understanding-assessment-testing-a-comprehensive-guide-for-educators-and-students> (date of access: 02.05.2026).
5. Future of Testing in Education: Artificial Intelligence. URL: <https://www.americanprogress.org/article/future-testing-education-artificial-intelligence/> (date of access: 02.05.2026).
6. Testing, testing. How educators use assessments to help students learn. URL: <https://www.schoolnewsnetwork.org/2023/11/22/testing-testing-how-educators-use-assessments-to-help-students-learn/> (date of access: 02.05.2026).
7. Best Student Assessment Software for Teachers. URL: <https://onlineexammaker.com/kb/8-best-student-assessment-software-for-teachers/> (date of access: 02.05.2026).
8. Best Free Assessment Software. URL: <https://www.softwareworld.co/free-assessment-software/> (date of access: 02.05.2026).
9. Best Assessment Software. URL: <https://research.com/software/best-assessment-software> (date of access: 02.05.2026).

10. Free Open Source Exam Software for Online Assessment. URL: <https://www.techjockey.com/blog/7-free-open-source-exam-software> (date of access: 02.05.2026).

11. Best Exam Software & Tools. URL: <https://www.proprofs.com/quiz-school/blog/best-exam-software/> (date of access: 02.05.2026).

12. Mentimeter. URL: <https://www.mentimeter.com/education/assessments> (date of access: 02.05.2026).

13. ExamSoft. URL: <https://examsoft.com/solutions/> (date of access: 02.05.2026).

14. Kaldin. URL: <https://www.kaldin.in/> (date of access: 02.05.2026).

15. Kotlin. Concise. Multiplatform. Fun. URL: <https://kotlinlang.org/> (date of access: 02.05.2026).

16. Kotlin Tutorial. URL: <https://www.geeksforgeeks.org/kotlin/kotlin-programming-language/> (date of access: 02.05.2026).

17. IntelliJ IDEA Learn and get assistance. URL: <https://www.jetbrains.com/idea/resources/> (date of access: 02.05.2026).

18. Everything you need to know about IntelliJ IDEA. URL: <https://www.polimetro.com/en/What-is-IntelliJ-Idea/> (date of access: 02.05.2026).

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для організації та управління тестуванням.

А.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмне забезпечення для організації та управління тестуванням знань», затверджене наказом Національного університету «Запорізька політехніка» № 139 від 07 квітня 2026 р.

А.2 Призначення розробки

Програмний продукт призначений для підтримки процесу тестування під час перевірки знань у здобувачів освіти.

А.3 Основні вимоги до програми, що розробляється

А.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для організації та управління тестуванням забезпечує виконання таких функцій:

- підтримка можливості реєстрації та авторизації користувачів із розмежуванням прав доступу відповідно до ролей у системі;
- підтримка можливості зберігання даних про користувачів у базі даних із забезпеченням цілісності інформації;
- можливість створення, налаштування та адміністрування тестів із визначенням параметрів проходження та критеріїв оцінювання;
- підтримка формування та редагування банку запитань із можливістю встановлення різних типів відповідей і вагових коефіцієнтів оцінювання;

- можливість автоматичного обчислення результатів тестування з урахуванням встановлених правил та збереження отриманих показників у базі даних;
- можливість обмеження доступу до тестів відповідно до встановлених правил і параметрів повторного проходження;
- можливість взаємодії клієнтської та серверної частин системи через програмний інтерфейс із забезпеченням безпечної передачі даних;
- підтримка масштабованості системи для забезпечення одночасної роботи значної кількості користувачів без втрати продуктивності;
- підтримка коректного відображення інтерфейсу в настільному та мобільному клієнтах із забезпеченням зручності використання програмного забезпечення.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для організації та управління тестуванням повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Програмне забезпечення для організації та управління тестуванням повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для організації та управління тестуванням необхідна наявність персонального комп'ютера.

А.3.5 Вимоги до складу та параметрів технічних засобів

Для роботи програмного забезпечення для організації та управління тестуванням потрібні такі технічні та програмні ресурси: процесор з тактовою частотою не нижче 2,4 ГГц із підтримкою багатопоточності та наявністю щонайменше чотирьох ядер, що дозволяє ефективно обробляти паралельні HTTP-запити та транзакції. Обсяг оперативної пам'яті для сервера має становити не менше 8 ГБ, однак для систем із підвищеним навантаженням рекомендовано 16 ГБ і більше, що забезпечить коректну роботу сервера застосунків та системи керування базами даних без зниження продуктивності.

Для зберігання даних необхідним є наявність твердотілого накопичувача обсягом не менше 256 ГБ, що гарантує достатній простір для розміщення самої системи, журналів подій, резервних копій і бази даних. Мережева інфраструктура має забезпечувати стабільне підключення до мережі з пропускну здатністю не нижче 100 Мбіт/с для серверного вузла, що дозволяє обслуговувати клієнтів без затримок при передачі даних.

Клієнтська частина програмного забезпечення для організації та управління тестуванням може функціонувати на персональних комп'ютерах із процесором не нижче 2,0 ГГц та оперативною пам'яттю від 4 ГБ, що є достатнім для роботи настільного застосунку або сучасного веббраузера. Для мобільних пристроїв мінімальні вимоги передбачають наявність процесора та оперативної пам'яті від 3 ГБ, а також актуальної версії мобільної операційної системи. Дисплей із роздільною здатністю не нижче 1280×720 пікселів забезпечує коректне відображення інтерфейсу та елементів керування.

Програмне забезпечення серверної частини передбачає встановлення операційної системи сімейства Windows або Linux із підтримкою сучасних засобів мережевої безпеки та багатокористувацького режиму. Необхідною є наявність встановленого середовища виконання Java Virtual Machine відповідної версії, що забезпечує коректну роботу серверних компонентів, розроблених із використанням мови програмування Kotlin. Також

потребується система керування базами даних, яка підтримує реляційну модель та механізми транзакційності, резервного копіювання і відновлення даних.

Для клієнтської частини необхідною є підтримка веббраузера з актуальними оновленнями або встановлене середовище виконання настільного застосунку. Додатково передбачається наявність стабільного підключення до мережі Інтернет для синхронізації з сервером та передачі результатів тестування. Використання антивірусного програмного забезпечення та засобів захисту мережі підвищує рівень інформаційної безпеки системи та мінімізує ризики несанкціонованого доступу до персональних даних користувачів.

A.3.6 Вимоги до маркування і пакування

Програма для організації та управління тестуванням може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для організації та управління тестуванням».

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для організації та управління тестуванням є облікові дані користувачів, що вводяться під час автентифікації, інформація про створювані тести, тексти запитань, варіанти відповідей та параметри оцінювання. До вхідної інформації також належать результати вибору відповідей під час проходження тестування, налаштування тривалості виконання завдань, правила допуску до повторного складання та інші параметри, що визначають логіку функціонування системи.

Вихідними даними програмного забезпечення для організації та управління тестуванням є результати проходження тестів, інформація про

успішність виконання завдань. Система формує підсумкові оцінки, відображає правильні та неправильні відповіді відповідно до встановлених правил, а також надає узагальнену інформацію. Вихідні дані відображаються у вигляді інтерактивних сторінок інтерфейсу.

ДОДАТОК Б**Фрагмент тексту програми**

```

class DtbsMngr {

    private val prsnlst: MutableList<Prsn> = mutableListOf()
    private val tstlst: MutableList<Tst> = mutableListOf()
    private val rsltlst: MutableList<Rslt> = mutableListOf()

    fun ddprsn(prsn: Prsn) {
        var fnd = false
        for (p in prsnlst) {
            if (p.lgn == prsn.lgn) {
                fnd = true
            }
        }
        if (!fnd) {
            prsnlst.add(prsn)
        }
    }

    fun gtprsn(lgn: String, psswr: String): Prsn? {
        var rs: Prsn? = null
        for (p in prsnlst) {
            if (p.lgn == lgn && p.psswr == psswr) {
                rs = p
            }
        }
        return rs
    }

    fun ddtst(tst: Tst) {
        tstlst.add(tst)
    }

    fun gttstlst(): MutableList<Tst> {
        val rs: MutableList<Tst> = mutableListOf()
        for (t in tstlst) {
            rs.add(t)
        }
        return rs
    }

    fun ddrslt(rslt: Rslt) {
        rsltlst.add(rslt)
    }

    fun gtrslts(prsncd: String): MutableList<Rslt> {
        val rs: MutableList<Rslt> = mutableListOf()
        for (r in rsltlst) {

```

```

        if (r.prsncd == prsncd) {
            rs.add(r)
        }
    }
    return rs
}

}

class Prsn(
    val prsncd: String,
    val lgn: String,
    val psswr: String,
    val rll: String
)

class Tst(
    val tstcd: String,
    val ttl: String,
    val dscr: String,
    val mxscr: Int
) {
    val qstnlst: MutableList<Qstn> = mutableListOf()

    fun ddqstn(qstn: Qstn) {
        qstnlst.add(qstn)
    }
}

class Qstn(
    val qstncd: String,
    val txt: String,
    val scr: Int
) {
    val nswrlst: MutableList<Nsw> = mutableListOf()

    fun ddnsrw(nsw: Nsw) {
        nswrlst.add(nsw)
    }
}

class Nsw(
    val nswrcd: String,
    val txt: String,
    val crct: Boolean
)

class Rslt(
    val rsltcd: String,

```

```

        val prsncd: String,
        val tstcd: String,
        val scr: Int,
        val pssd: Boolean
    )

class TstSrvc(private val dtbsmngr: DtbsMngr) {

    fun crttst(tstcd: String, ttl: String, dscr: String, mxscr:
Int): Tst {
        val tst = Tst(tstcd, ttl, dscr, mxscr)
        dtbsmngr.ddtst(tst)
        return tst
    }

    fun ddqstntst(tstcd: String, qstn: Qstn) {
        val lst = dtbsmngr.gttstlst()
        for (t in lst) {
            if (t.tstcd == tstcd) {
                t.ddqstn(qstn)
            }
        }
    }

    fun gttst(tstcd: String): Tst? {
        var rs: Tst? = null
        val lst = dtbsmngr.gttstlst()
        for (t in lst) {
            if (t.tstcd == tstcd) {
                rs = t
            }
        }
        return rs
    }
}

class RsltSrvc(private val dtbsmngr: DtbsMngr) {

    fun prcst(prsncd: String, tst: Tst, rspns: Map<String,
String>): Rslt {
        var ttl = 0
        for (q in tst.qstnlst) {
            val ch = rspns[q.qstncd]
            for (n in q.nswrlst) {
                if (n.nswrcd == ch && n.crct) {
                    ttl += q.scr
                }
            }
        }
    }
}

```

```

    }
    val pssd = ttl >= tst.mxscr / 2
    val rslt = Rslt(
        rsltcd = prsncd + "_" + tst.tstcd,
        prsncd = prsncd,
        tstcd = tst.tstcd,
        scr = ttl,
        pssd = pssd
    )
    dtbsmgr.ddrslt(rslt)
    return rslt
}

fun gtrslts(prsncd: String): MutableList<Rslt> {
    return dtbsmgr.gtrslts(prsncd)
}

}

class CntlrRst(
    private val dtbsmgr: DtbsMgr,
    private val tstsrv: TstSrv,
    private val rsltsrv: RsltSrv
) {

    fun rgsprsn(prsncd: String, lgn: String, psswd: String, rll:
String): String {
        val prsn = Prsn(prsncd, lgn, psswd, rll)
        dtbsmgr.ddprsn(prsn)
        return "PRSN_RGSTRD"
    }

    fun lgnprsn(lgn: String, psswd: String): String {
        val prsn = dtbsmgr.gtprsn(lgn, psswd)
        return if (prsn != null) {
            "LGN_SCCSS"
        } else {
            "LGN_FLR"
        }
    }

    fun crttstreq(tstcd: String, ttl: String, dscr: String,
mxscr: Int): String {
        tstsrv.crttst(tstcd, ttl, dscr, mxscr)
        return "TST_CRTD"
    }

    fun ddqstnreq(tstcd: String, qstncd: String, txt: String,
scr: Int): String {

```

```

        val q = Qstn(qstncd, txt, scr)
        tstsrv.ddqstntst(tstcd, q)
        return "QSTN_DD"
    }

    fun ddnsrreq(tstcd: String, qstncd: String, nswrcd: String,
txt: String, crct: Boolean): String {
        val tst = tstsrv.gttst(tstcd)
        if (tst != null) {
            for (q in tst.qstnlst) {
                if (q.qstncd == qstncd) {
                    val n = Nsw(nswrcd, txt, crct)
                    q.ddnsr(n)
                }
            }
        }
        return "NSWR_DD"
    }

    fun prcstreq(prsncd: String, tstcd: String, rspns:
Map<String, String>): String {
        val tst = tstsrv.gttst(tstcd)
        return if (tst != null) {
            val rslt = rsltsrv.prcst(prsncd, tst, rspns)
            if (rslt.pssd) {
                "PSSD_${rslt.scr}"
            } else {
                "FLD_${rslt.scr}"
            }
        } else {
            "TST_NFND"
        }
    }

    fun gtrsltsreq(prsncd: String): String {
        val lst = rsltsrv.gtrslts(prsncd)
        val bldr = StringBuilder()
        for (r in lst) {
            bldr.append(r.tstcd)
            bldr.append("_")
            bldr.append(r.scr)
            bldr.append("_")
            bldr.append(r.pssd)
            bldr.append(";")
        }
        return bldr.toString()
    }
}

```

```

class SttsMngr(private val dtbsmngr: DtbsMngr) {

    fun gnstts(): String {
        val tsts = dtbsmngr.gttstlst()
        val bldr = StringBuilder()
        for (t in tsts) {
            bldr.append(t.tstcd)
            bldr.append(":")
            var ttlq = 0
            for (q in t.qstnlst) {
                ttlq++
            }
            bldr.append(ttlq)
            bldr.append("|")
        }
        return bldr.toString()
    }

    fun gnrprt(prsncd: String): String {
        val rslts = dtbsmngr.gtrslts(prsncd)
        val bldr = StringBuilder()
        var sm = 0
        var cnt = 0
        for (r in rslts) {
            sm += r.scr
            cnt++
        }
        bldr.append("CNT:")
        bldr.append(cnt)
        bldr.append(";SM:")
        bldr.append(sm)
        return bldr.toString()
    }
}

class SssnMngr {

    private val sssnmp: MutableMap<String, String> =
mutableMapOf()

    fun crtsssn(sssncd: String, prsncd: String) {
        sssnmp[sssncd] = prsncd
    }

    fun gtprsncd(sssncd: String): String? {
        return sssnmp[sssncd]
    }
}

```

```

    fun rmvsssn(sssncd: String) {
        sssnmp.remove(sssncd)
    }
}

class CfgMgr {

    private var mxqstnprpg: Int = 5
    private var mxprsn: Int = 100

    fun stmxxqstn(vl: Int) {
        if (vl > 0) {
            mxqstnprpg = vl
        }
    }

    fun stmxxprsn(vl: Int) {
        if (vl > 0) {
            mxprsn = vl
        }
    }

    fun gtcfgtxt(): String {
        val bldr = StringBuilder()
        bldr.append("MXQ:")
        bldr.append(mxqstnprpg)
        bldr.append(";MXP:")
        bldr.append(mxprsn)
        return bldr.toString()
    }
}

class DshbrdSrvc(private val dtbsmgr: DtbsMgr) {

    fun gndshbrd(prsncd: String): String {
        val bldr = StringBuilder()
        val rslts = dtbsmgr.gtrslts(prsncd)
        bldr.append("PRSN:")
        bldr.append(prsncd)
        bldr.append("|RSLT_CNT:")
        bldr.append(rslts.size)
        bldr.append("|")
        for (r in rslts) {
            bldr.append(r.tstcd)
            bldr.append(":")
            bldr.append(r.scr)
            bldr.append(":")
        }
    }
}

```

```

        bldr.append(r.pssd)
        bldr.append(";")
    }
    return bldr.toString()
}

fun gnsqry(): String {
    val bldr = StringBuilder()
    val tsts = dtbsmgr.gttstlst()
    bldr.append("TST_CNT:")
    bldr.append(tsts.size)
    bldr.append("|")
    for (t in tsts) {
        bldr.append(t.tstcd)
        bldr.append("_")
        bldr.append(t.qstnlst.size)
        bldr.append(";")
    }
    return bldr.toString()
}
}

class PrsnMgr(private val dtbsmgr: DtbsMgr) {

    fun updprsn(prsncd: String, nwpsswr: String, nwrl: String):
Boolean {
        val prsn = dtbsmgr.gtrslts(prsncd)
        var upd = false
        for (r in prsn) {
            if (r.prsncd == prsncd) {
                upd = true
            }
        }
        if (upd) {
            val pr = dtbsmgr.gtprsn(prsncd, nwpsswr)
            if (pr == null) {
                dtbsmgr.ddprsn(Prsn(prsncd, prsncd, nwpsswr,
nwrl))
            }
        }
        return upd
    }

    fun vrfyprsn(lgn: String): Boolean {
        val pr = dtbsmgr.gtprsn(lgn, "")
        return pr != null
    }
}

```

```

class TstAdmSrvc(private val dtbsmngr: DtbsMngr) {

    fun rmvtst(tstcd: String): Boolean {
        val lst = dtbsmngr.gttstlst()
        var rmv = false
        val itr = lst.iterator()
        while (itr.hasNext()) {
            val t = itr.next()
            if (t.tstcd == tstcd) {
                itr.remove()
                rmv = true
            }
        }
        return rmv
    }

    fun updtst(tstcd: String, nwttl: String, nwdscr: String,
nwmxscr: Int): Boolean {
        val lst = dtbsmngr.gttstlst()
        var upd = false
        for (t in lst) {
            if (t.tstcd == tstcd) {
                val tmp = Tst(t.tstcd, nwttl, nwdscr, nwmxscr)
                for (q in t.qstnlst) {
                    tmp.ddqstn(q)
                }
                upd = true
            }
        }
        return upd
    }

    fun vldtst(tstcd: String): Boolean {
        val lst = dtbsmngr.gttstlst()
        for (t in lst) {
            if (t.tstcd == tstcd) {
                if (t.qstnlst.isEmpty()) {
                    return false
                }
                for (q in t.qstnlst) {
                    if (q.nswrlst.isEmpty()) {
                        return false
                    }
                }
                return true
            }
        }
        return false
    }
}

```

```

}

class RprtSrvc(private val dtbsmngr: DtbsMngr) {

    fun gnrprttst(tstcd: String): String {
        val bldr = StringBuilder()
        val tsts = dtbsmngr.gttstlst()
        for (t in tsts) {
            if (t.tstcd == tstcd) {
                bldr.append("TST:")
                bldr.append(t.ttl)
                bldr.append("|QSTN_CNT:")
                bldr.append(t.qstnlst.size)
                bldr.append("|")
                for (q in t.qstnlst) {
                    bldr.append(q.qstncd)
                    bldr.append(":")
                    bldr.append(q.scr)
                    bldr.append(";")
                }
            }
        }
        return bldr.toString()
    }

    fun gnrprtscr(prsncd: String): String {
        val rslts = dtbsmngr.gtrslts(prsncd)
        var mx = 0
        var mn = Int.MAX_VALUE
        var sm = 0
        for (r in rslts) {
            if (r.scr > mx) {
                mx = r.scr
            }
            if (r.scr < mn) {
                mn = r.scr
            }
            sm += r.scr
        }
        val bldr = StringBuilder()
        bldr.append("MX:")
        bldr.append(mx)
        bldr.append("|MN:")
        bldr.append(if (mn == Int.MAX_VALUE) 0 else mn)
        bldr.append("|SM:")
        bldr.append(sm)
        return bldr.toString()
    }
}

```

```

class SrtSrvc(private val dtbsmngr: DtbsMngr) {

    fun srttstbyttrl(): List<Tst> {
        val lst = dtbsmngr.gttstlst()
        return lst.sortedBy { it.ttl }
    }

    fun srtrsltsbyscr(prsncd: String): List<Rslt> {
        val lst = dtbsmngr.gtrslts(prsncd)
        return lst.sortedByDescending { it.scr }
    }

    fun fltrpssd(prsncd: String): List<Rslt> {
        val lst = dtbsmngr.gtrslts(prsncd)
        val rs: MutableList<Rslt> = mutableListOf()
        for (r in lst) {
            if (r.pssd) {
                rs.add(r)
            }
        }
        return rs
    }
}

```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет “Запорізька політехніка”
Кафедра програмних засобів

Програмне забезпечення для організації та управління тестуванням знань

Виконав
ст. гр. КНТ-112

Данило КОЗА

Керівник
д.т.н., професор

Сергій СУББОТІН

Рисунок В.1 – Слайд 1

Об’єкт, предмет та мета роботи

Об’єктом дослідження кваліфікаційної роботи бакалавра є процес розробки програмного забезпечення для організації та управління тестуванням.

Предмет дослідження – програмні засоби для організації та управління тестуванням.

Мета роботи – розробка програмного забезпечення для організації та управління тестуванням знань, що забезпечує зменшення трудомісткості процесу організації тестування.

2

Рисунок В.2 – Слайд 2

Завдання роботи

Необхідно було вирішити такі завдання:

- виконати аналіз предметної області та програмних засобів для організації та управління тестуванням;
- здійснити проєктування програмного забезпечення для організації та управління тестуванням;
- створити програмне забезпечення для організації та управління тестуванням;
- виконати тестування розробленого програмного забезпечення для організації та управління тестуванням.

3

Рисунок В.3 – Слайд 3

Програмне забезпечення для організації та управління тестуванням

Критерій порівняння	Mentimeter	ExamSoft	Kaldin
Безпечне оцінювання	+-	+	+-
Гнучкість налаштування	+-	+-	+
Візуалізація даних	+	+-	+-
Зручність інтеграції з іншими системами	+-	+-	-
Простота використання	+	+-	+-
Масштабованість для великих груп	+-	+	+-

4

Рисунок В.4 – Слайд 4

Вибір мови програмування

Критерій порівняння мов програмування	Мова програмування		
	Kotlin	Python	Java
Мультиплатформеність (сервер/мобільні/веб)	+	+-	+-
Сумісність з існуючою інфраструктурою JVM	+	-	+
Лаконічність синтаксису	+	+	-
Асинхронне програмування та обробка потоків	+	+-	+-
Продуктивність виконання	+	+-	+

5

Рисунок В.5 – Слайд 5

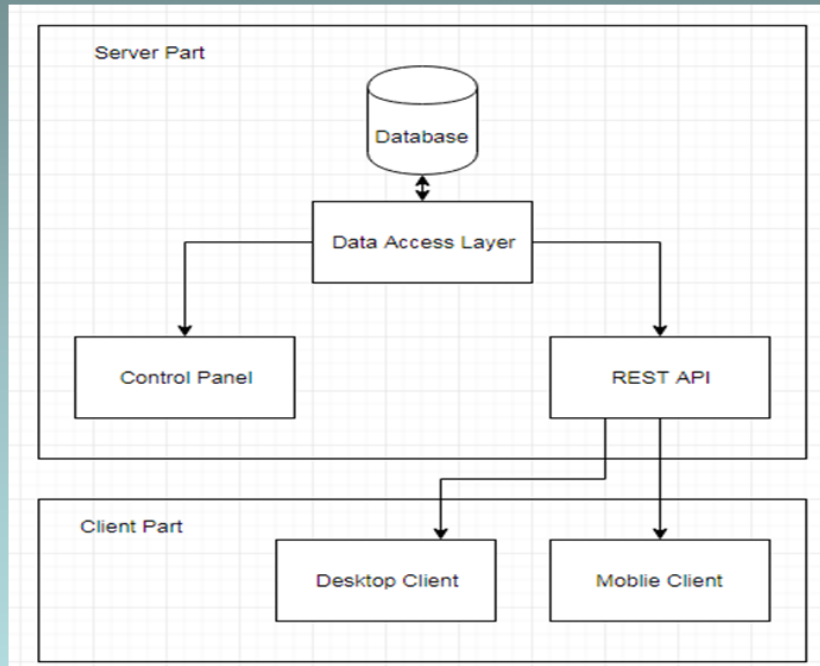
Обґрунтування вибору середовища розробки

Критерій порівняння середовищ розробки	Середовища розробки		
	IntelliJ IDEA	Eclipse	NetBeans
Інструменти автодоповнення та рефакторингу	+	+-	+-
Інтеграція з системами контролю версій	+	+	+-
Підтримка багатоплатформеності	+	+-	+-
Плагіни та розширюваність	+	+	+-
Підтримка серверних і веб-проектів	+	+-	+-

6

Рисунок В.6 – Слайд 6

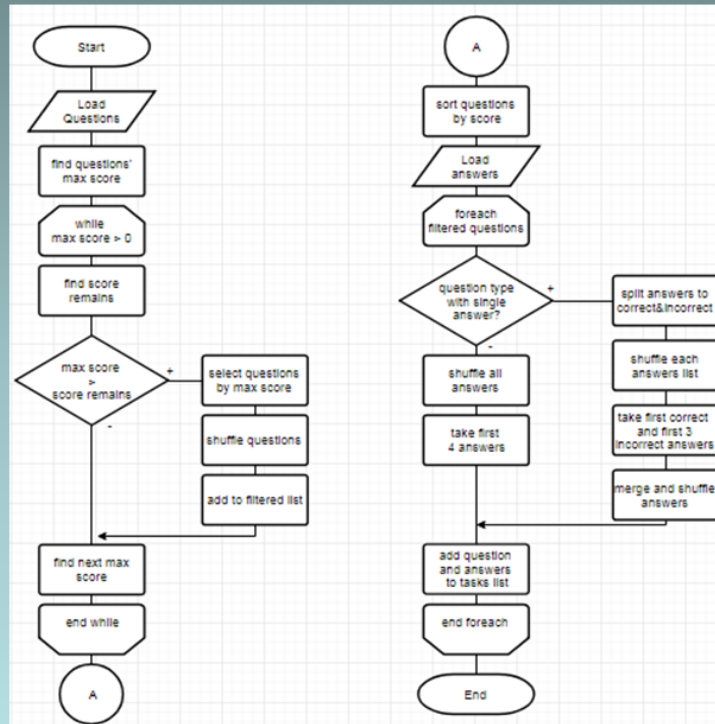
Структура програми



7

Рисунок В.7 – Слайд 7

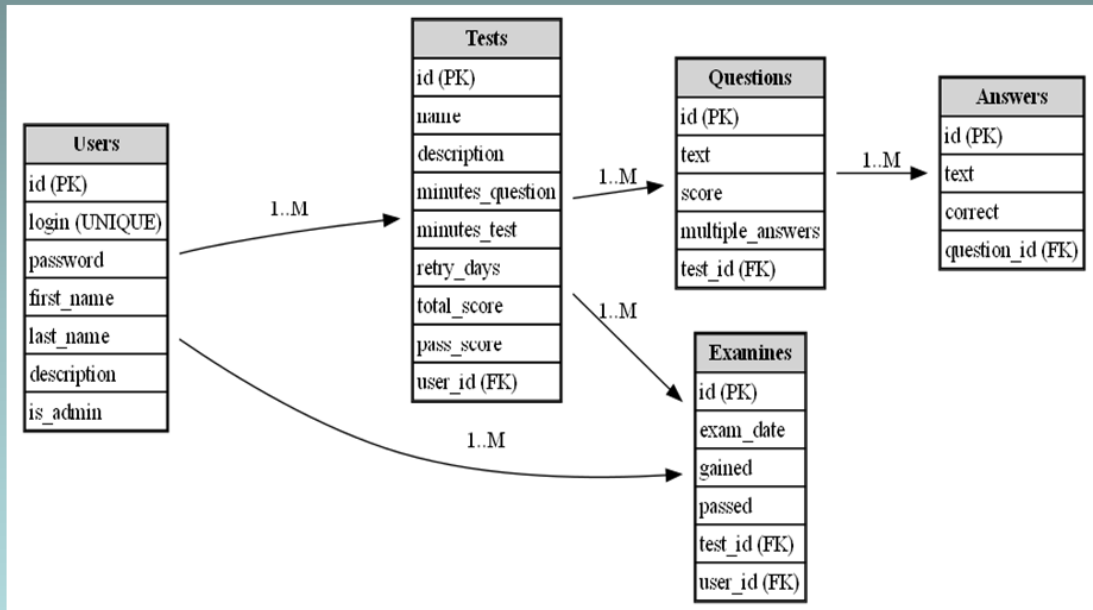
Схема функціонування програми



8

Рисунок В.8 – Слайд 8

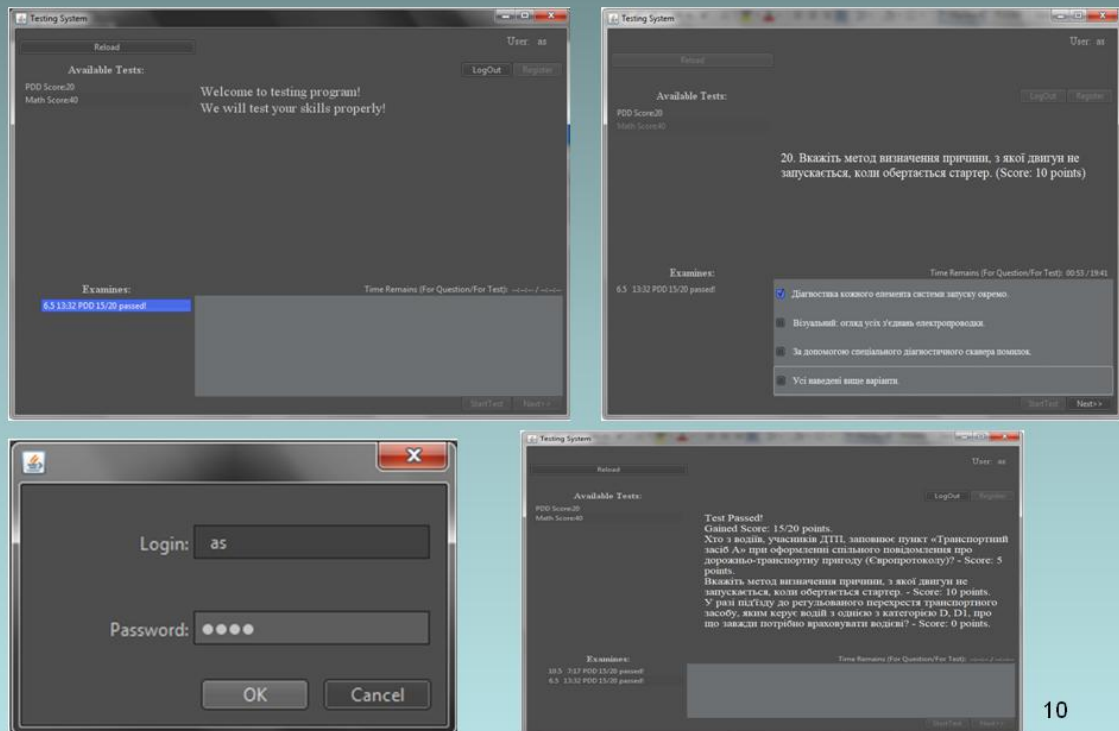
Схема бази даних



9

Рисунок В.9 – Слайд 9

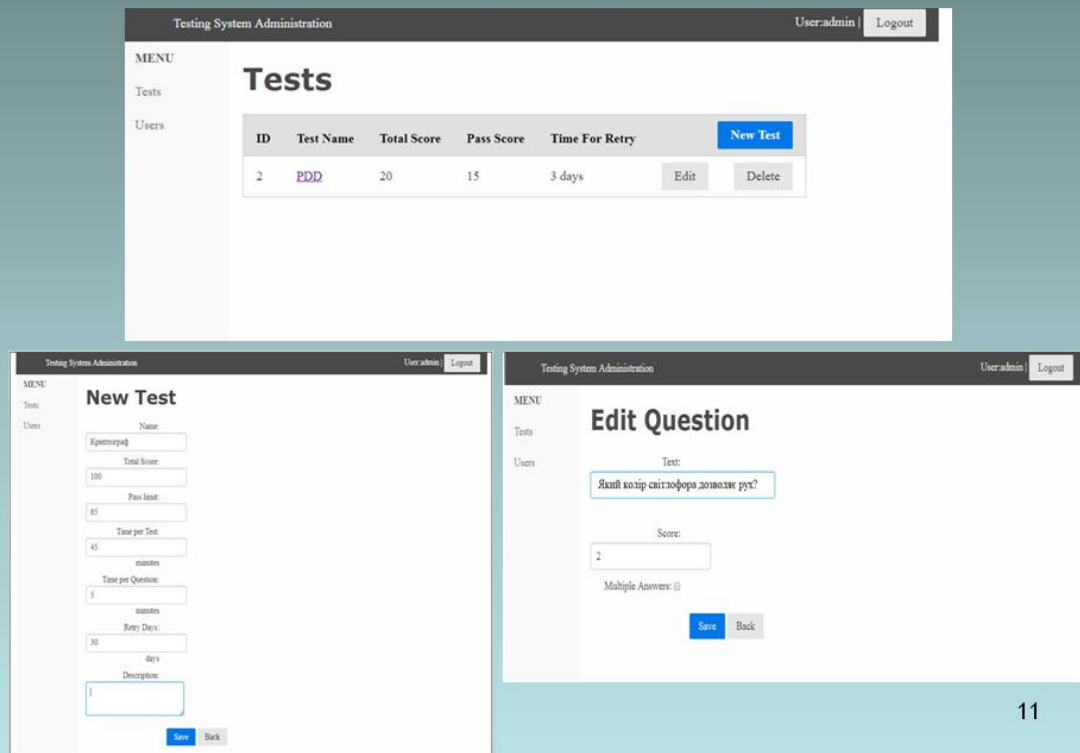
Робота користувача з програмою



10

Рисунок В.10 – Слайд 10

Робота адміністратора з програмою



11

Рисунок В.11 – Слайд 11

Висновки

В результаті виконаної роботи було проаналізовано та досліджено процес розробки програмного забезпечення для організації та управління тестуванням. Сформульовано функціональні вимоги до програмного забезпечення для організації та управління тестуванням.

Для розробки програмного забезпечення для організації та управління тестуванням обрано мову програмування Kotlin та середовище розробки IntelliJ IDEA.

Запропоновано структуру програмного забезпечення для організації та управління тестуванням, яка базується на чіткому розмежуванні серверної логіки та клієнтських інтерфейсів, що забезпечує масштабованість, надійність і гнучкість системи. Розроблено базу даних програмного забезпечення для організації та управління тестуванням.

Результати тестування програмного забезпечення для організації та управління тестуванням показали, що програма функціонує коректно та відповідає технічному завданню.

12

Рисунок В.12 – Слайд 12