

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Платформа для онлайн тестувань»

Виконав студент групи ПЗ-21-1
САХНО Андрій Олександрович

Керівник роботи:
ВАКАЛЮК Тетяна Анатоліївна

Рецензент:
ЛОКТИКОВА Тамара Миколаївна

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії програмного
забезпечення

Тетяна Вакалюк

«17» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **САХНО Андрій Олександрович**

Керівник роботи: **ВАКАЛЮК Тетяна Анатоліївна**

Тема роботи: **«Платформа для онлайн тестувань»**,

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: Об'єктом дослідження є інформаційні процеси автоматизації освітніх функцій, пов'язаних з організацією, проведенням та оцінюванням онлайн-тестувань. Це включає реєстрацію користувачів, управління тестами, обробку відповідей та формування результатів у цифровому середовищі.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Вакалюк Т.А.	17.02.2025р.	17.02.2025р.
2	Вакалюк Т.А.	17.02.2025р.	17.02.2025р.
3	Вакалюк Т.А.	17.02.2025р.	17.02.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з платформи для онлайн тестувань. Пояснювальна записка до випускної роботи містить 102 сторінок, 50 ілюстрацій та 10 таблиць.

Метою роботи є розробка та впровадження веб-додатку для автоматизації процесів організації та проведення онлайн-тестувань у навчальної середовищі. У проєкті використано сучасні хмарні технології Microsoft Azure. Архітектура побудована за принципами чистої архітектури: Razor Pages відповідає за клієнтську частину, а бізнес-логіка і доступ до даних реалізовані окремими модулями .NET 8, що працюють з базою Azure Cosmos DB через MongoDB API. Додаток підтримує функціонал імпорту та експорту тестів через Excel-файли, а також має інтеграцію з Azure OpenAI для автоматичної генерації тестових запитань.

Інтерфейс платформи побудований з урахуванням ролей користувачів, має адаптивний дизайн та інтуїтивну навігацію, що забезпечує комфортне користування як для учасників тестувань, так і для організаторів і адміністраторів.

КЛЮЧОВІ СЛОВА: ПЛАТФОРМА ДЛЯ ОНЛАЙН-ТЕСТУВАНЬ; RAZOR PAGES; MONGODB; AZURE OPENAI; АВТОМАТИЗАЦІЯ ОСВІТНІХ ПРОЦЕСІВ.

					ІПЗ.КР.Б-121-25-ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		Сахно А.О.			Платформа для онлайн тестувань		Лім.	Арк.	Аркушів
Перевір.		Вакалюк Т. А.						3	102
Керівник				ФІКТ Гр. ІПЗ-21-1					
Н. контр.		Тичина О.П.							
Зав. каф.		Вакалюк Т. А.							

ABSTRACT

Bachelor's qualification thesis consists of an online testing platform. The explanatory note to the final thesis contains 102 pages, 50 illustrations and 10 tables.

The aim of the work is to develop and implement a web application for automating the processes of organizing and conducting online testing in an educational environment. The project uses modern cloud technologies by Microsoft Azure. The architecture is built according to the principles of clean architecture: Razor Pages is responsible for the client side, while the business logic and data access are implemented as separate .NET 8 modules that interact with the Azure Cosmos DB database via the MongoDB API. The application supports the functionality of importing and exporting tests via Excel files and includes integration with Azure OpenAI for automatic generation of test questions.

The platform interface is built with consideration of user roles, has a responsive design and intuitive navigation, ensuring a comfortable experience both for test participants and for organizers and administrators.

KEYWORDS: ONLINE TESTING PLATFORM; RAZOR PAGES; MONGODB; AZURE OPENAI; AUTOMATION OF EDUCATIONAL PROCESSES.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПЛАТФОРМИ ДЛЯ ОНЛАЙН ТЕСТУВАНЬ	10
1.1. Постановка задачі	10
1.2. Аналіз аналогів програмного продукту	11
1.3. Вибір архітектури.....	17
1.4. Обґрунтування вибору технологічного стеку	19
Висновки до першого розділу	23
РОЗДІЛ 2. ПРОЕКТУВАННЯ ПЛАТФОРМИ ДЛЯ ОНЛАЙН ТЕСТУВАНЬ	24
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури платформи для онлайн тестувань.....	24
2.2. Розробка бази даних системи	35
2.3. Проектування та реалізація алгоритмів роботи системи	39
2.4. Застосування архітектурних патернів у побудові системи.....	46
Висновки до другого розділу.....	50
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПЛАТФОРМОЮ ДЛЯ ОНЛАЙН ТЕСТУВАНЬ	52
3.1. Порядок встановлення та налаштування параметрів системи	52
3.2. Структура інтерфейсу платформи для онлайн тестувань.....	53
3.3. Тестування роботи програмного комплексу	70
Висновки до третього розділу.....	72
ВИСНОВКИ	74
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	75

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

MVC – Model-View-Controller; архітектурний шаблон, що поділяє додаток на модель, представлення та контролер

SDK – Software Development Kit; набір інструментів для розробки програмного забезпечення

БД – база даних; система для зберігання, обробки та керування даними

CRUD – Create, Read, Update, Delete; базові операції над даними у будь-якій інформаційній системі

UI – User Interface; інтерфейс користувача, візуальна частина програми

UML – Unified Modeling Language; уніфікована мова моделювання, що використовується для візуалізації систем

DI – Dependency Injection; механізм впровадження залежностей між компонентами у програмі

.NET – програмна платформа від Microsoft для розробки та виконання додатків

LINQ – Language Integrated Query; мова запитів, інтегрована в C# для доступу до даних

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми зумовлена зростаючою потребою у впровадженні сучасних цифрових інструментів для контролю знань в освітньому процесі. В умовах дистанційного або змішаного навчання особливо важливо мати зручний інструмент, який дозволить не лише проводити контроль знань, а й робити це швидко, безпечно та прозоро. Сучасні технології дозволяють автоматизувати рутинні процеси, зменшити навантаження на викладачів і при цьому забезпечити якісну перевірку знань. Завдяки розвитку хмарних платформ та інструментів на основі штучного інтелекту, з'являється можливість створити розумну, гнучку і масштабовану систему тестування.

Мета цієї випускної кваліфікаційної роботи — розробка веб-орієнтованої платформи для онлайн тестувань, яка забезпечує автоматизацію процесів реєстрації та авторизації користувачів із різними ролями (звичайний користувач, організатор, адміністратор), створення та проходження тестів, обробку результатів і перегляд статистики. Особливістю розробленої системи є підтримка імпорту та експорту тестів через Excel-файли, а також генерація тестових запитань із використанням Azure OpenAI.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати предметну область онлайн-тестування, визначити ролі користувачів і сформулювати вимоги до функціональності платформи;
- спроектувати об'єктно-орієнтовану модель системи з чітким поділом відповідальностей між модулями;
- реалізувати структуру взаємодії між клієнтською частиною (Razor Pages) і серверною логікою (.NET 8) із підключенням до бази Azure Cosmos DB через MongoDB API;
- розробити функціонал для створення, редагування та проходження тестів, зберігання результатів і побудови аналітики;
- розробити функціонал імпорту та експорту питань, генерації питань з допомогою Azure OpenAI;

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

– здійснити тестування реалізованої платформи та перевірити відповідність її функціоналу заявленим вимогам.

Об’єктом дослідження є інформаційні процеси автоматизації освітніх функцій, пов’язаних з організацією, проведенням та оцінюванням онлайн-тестувань. Це включає реєстрацію користувачів, управління тестами, обробку відповідей та формування результатів у цифровому середовищі.

Предметом дослідження виступають методи та засоби побудови веб-орієнтованої системи тестування, зокрема архітектурні підходи на базі Razor Pages і .NET 8, використання бази даних Azure Cosmos DB зі структурою MongoDB, а також реалізація бізнес-логіки у вигляді окремих сервісів з чітким розділенням відповідальностей.

Для досягнення поставленої мети у роботі застосовувалися методи функціонального та структурного аналізу предметної області, об’єктно-орієнтоване моделювання із побудовою UML-діаграм, а також практичні підходи до створення інтерактивних інтерфейсів користувача. В рамках дослідження було розроблено прототип платформи та протестовано його на відповідність реальним сценаріям взаємодії користувачів. Інформаційною базою дослідження стали технічна документація з розробки веб-застосунків, рекомендації з організації онлайн-тестувань, а також власний досвід проходження й адміністрування освітніх платформ.

Для валідації технічних рішень було проаналізовано існуючі системи онлайн-тестування, що дозволило обрати найбільш ефективні архітектурні рішення й технології для реалізації поставлених завдань. Окрему увагу приділено інтеграції сервісів штучного інтелекту (Azure OpenAI), що значно розширює функціональні можливості платформи.

Практична цінність одержаних результатів полягає у створенні працюючого веб-додатку, який забезпечує повний цикл проведення онлайн-тестування — від створення тесту до аналізу результатів. Платформа може бути впроваджена в освітніх закладах для підвищення якості й ефективності процесу оцінювання знань.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

Структура роботи побудована логічно та послідовно: у вступі обґрунтовано актуальність теми, визначено мету, завдання, об'єкт і предмет дослідження, описано методи та джерела інформації. Перший розділ присвячено аналізу вимог і архітектурних рішень, другий — моделюванню та розробці основного функціоналу, а третій — фізичній реалізації, деталям імплементації та тестуванню. У завершальних висновках підбито підсумки проєкту та наведено можливі напрями його розвитку.

За результатами роботи було взято участь у 1 конференції та опубліковано тези [32].

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПЛАТФОРМИ ДЛЯ ОНЛАЙН ТЕСТУВАНЬ

1.1. Постановка задачі

У сучасному цифровому середовищі все більшої популярності набувають освітні онлайн-платформи, які забезпечують дистанційний доступ до знань, навчальних матеріалів та перевірки засвоєного матеріалу. Тестування — один із ключових інструментів контролю знань, і саме тому потреба у надійних, зручних і функціональних онлайн-рішеннях зростає щодня. Водночас проведення тестів у дистанційному форматі часто супроводжується низкою труднощів: складністю організації, проблемами з безпекою, обмеженим функціоналом існуючих рішень, або їхньою надмірною складністю для звичайного користувача.

Розробка веб-платформи «Aegos Exam» покликана вирішити ці проблеми шляхом створення інтуїтивно зрозумілого середовища для організації, проходження та оцінювання тестувань. Система орієнтована на три ключові ролі: звичайного користувача (учасника тестування), організатора (того, хто створює і проводить тести), а також адміністратора (який контролює платформу загалом). Усі ролі мають власний набір можливостей, що дозволяє забезпечити гнучке управління системою без втрати зручності у використанні.

Проект реалізовано з використанням сучасних технологій: .NET 8, Razor Pages для фронтенду, Azure Cosmos DB для зберігання даних та інтеграції з Azure OpenAI, яка дозволяє автоматично генерувати запитання на основі заданої тематики. Такий підхід дозволяє організаторам значно скоротити час на підготовку тестів, водночас підтримуючи варіативність завдань. Додатково передбачено можливість імпорту та експорту тестів у форматі Excel, що значно спрощує перенесення або резервування інформації. Веб-додаток розгорнуто в середовищі Azure Web App, що забезпечує масштабованість, безпеку та надійність у роботі системи.

Архітектура платформи дотримується принципів чистої архітектури, що забезпечує чітке розділення відповідальностей між модулями. Бізнес-логіка і доступ до даних винесені в окремі шари, що дозволяє легко вносити зміни або

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

розширювати функціональність без негативного впливу на інші частини системи. Авторизація користувачів реалізована через механізм Claims із підтримкою ролей і використанням cookie-based аутентифікації.

У процесі розробки платформи «Aegos Exam» було визначено низку ключових задач, які охоплюють усі етапи створення сучасного, надійного та функціонального веб-застосунку. Насамперед йшлося про формування чітких вимог до функціональності системи, з урахуванням потреб кожної з ролей — користувача, організатора та адміністратора. Далі було спроектовано об'єктно-орієнтовану модель системи, що базується на принципах чіткого поділу відповідальностей між компонентами.

Паралельно велась реалізація бізнес-логіки, яка забезпечує взаємодію між користувачами та системою, включно зі створенням тестів, їх проходженням, перевіркою та переглядом результатів. Інтеграція механізмів імпорту та експорту у форматі Excel значно спрощує роботу з великою кількістю тестових завдань. Ще одним важливим кроком стало підключення Azure OpenAI, яке дозволяє автоматизувати створення питань і тим самим зменшити навантаження на організаторів. Завершальним етапом стало тестування функціональності та перевірка зручності користування системою з точки зору усіх типів користувачів.

Реалізований програмний комплекс дозволяє створювати повноцінні сценарії онлайн-тестувань у різних навчальних контекстах — від самостійних тренувальних тестів до масштабних контрольних робіт. Завдяки відкритій архітектурі система може бути розширена або адаптована під інші освітні потреби у майбутньому.

1.2. Аналіз аналогів програмного продукту

У сучасному освітньому просторі та корпоративному навчанні активно використовуються онлайн-платформи для створення та проведення тестувань. Аналіз існуючих рішень дозволяє визначити їхні сильні та слабкі сторони, що допомагає сформулювати оптимальну концепцію для розробки власної системи.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

Першим розглянутим аналогом є українська освітня платформа «На Урок» [1] (рис.1.1).

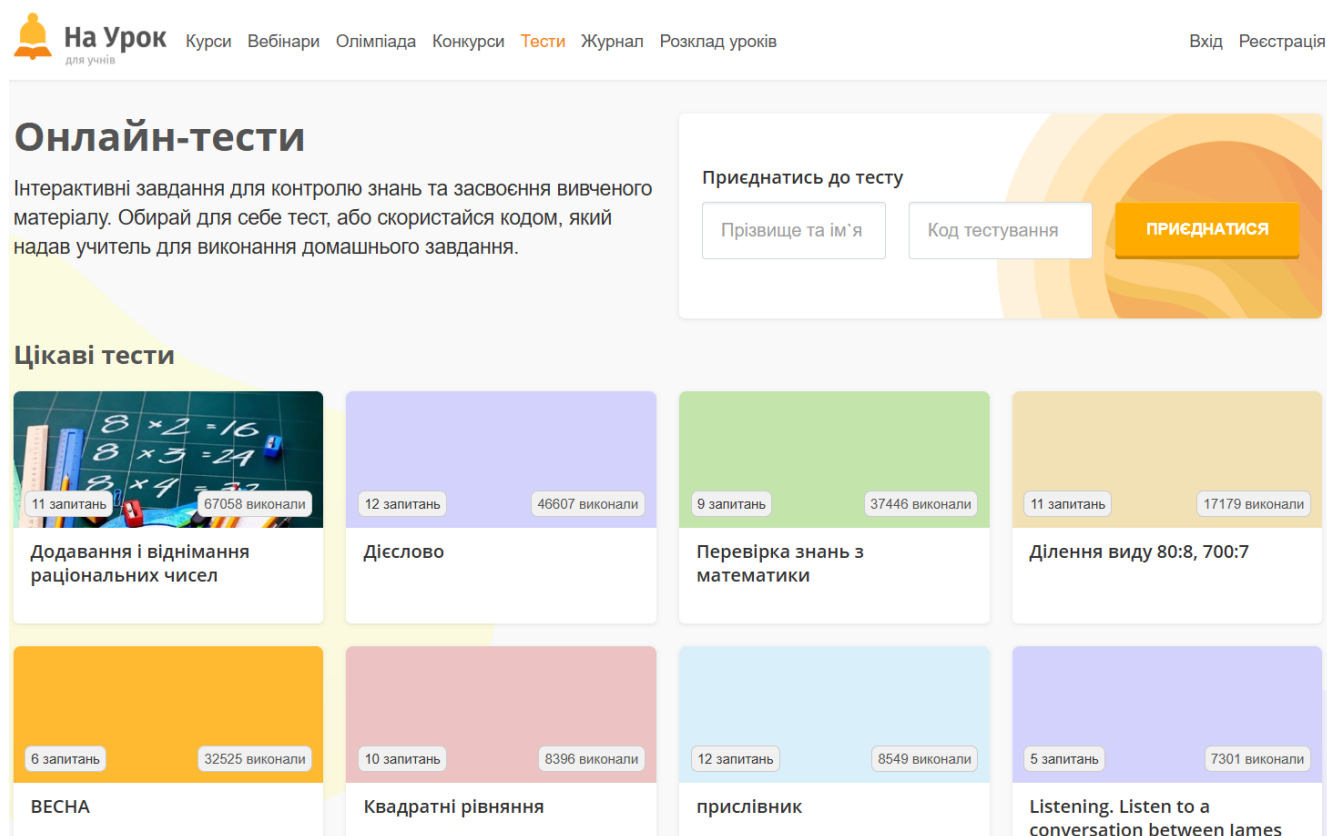


Рис. 1.1 – Платформа «На урок»

Платформа орієнтована насамперед на шкільних учителів і використовується переважно в рамках загальної середньої освіти. Її цільова аудиторія — це викладачі, які прагнуть створювати тести для поточної перевірки знань у зручній та доступній формі.

Формати тестування, які підтримуються системою, включають запитання з одним або кількома правильними варіантами відповіді, а також відкриті запитання, а відповіді учня зберігаються автоматично під час проходження тесту. Присутня базова система оцінювання, фіксація часу виконання й автоматичне збереження результатів. У випадку втрати з'єднання учень не має змоги повернутися до тієї ж сесії — тестування доведеться проходити заново.

Щодо дизайну та користувацького досвіду, інтерфейс платформи загалом зручний, лаконічний та орієнтований на простоту. Він добре сприймається вчителями, які не мають глибоких технічних навичок, однак виглядає дещо

застарілим на сучасних пристроях, а також має обмеження в адаптації до мобільних екранів.

Основним недоліком платформи є її закритість до інтеграції та автоматизації: «На Урок» не підтримує імпорту/експорту у форматі Excel, не має механізмів генерації тестів за допомогою штучного інтелекту. А також неможливість відновлення початої сесії.

Другим розглянутим прикладом є **Google Forms** [2] (рис. 1.2) — онлайн-сервіс від компанії Google, який дозволяє створювати тести у форматі форм.

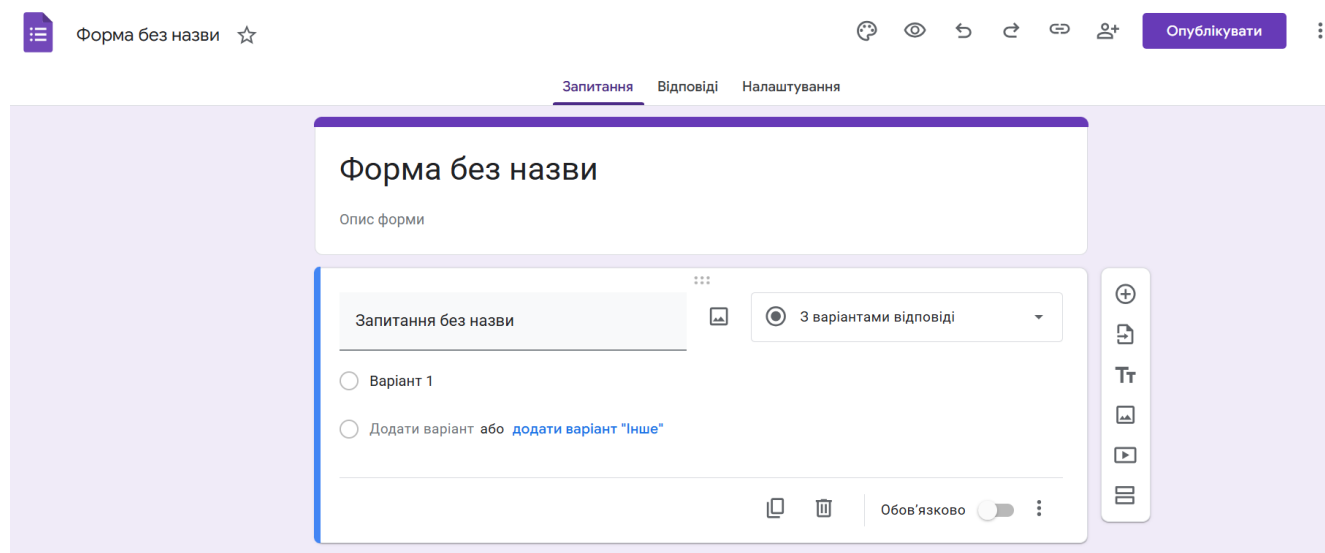


Рис. 1.2 – Google Forms у режимі Quiz

Платформа орієнтована на широке коло користувачів, включаючи викладачів, тренерів, HR-фахівців та організаторів освітніх або комерційних заходів. Завдяки простоті та доступності вона активно використовується у школах, університетах і компаніях.

Google Forms підтримує стандартні формати запитань: вибір одного або кількох варіантів відповіді, відкриті текстові поля, сітки, шкали тощо. У режимі Quiz можливо задавати бали, автоматично перевіряти правильність відповідей та формувати коротку статистику. Система дозволяє встановити обмеження на перегляд результатів або вказати правильні відповіді після завершення.

Щодо гнучкості, Google Forms має обмежений набір налаштувань. Користувач не може призначати таймер, обмежувати кількість спроб або створювати адаптивні сценарії проходження.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

Інтерфейс Google Forms мінімалістичний, добре адаптується до різних пристроїв, швидкий у завантаженні. Завдяки знайомому стилю Google середовище сприймається інтуїтивно.

Важливим обмеженням є те, що відповіді зберігаються лише після натискання кнопки "Надіслати", а під час проходження прогрес не фіксується. У разі втрати з'єднання або випадкового закриття вкладки сесія проходження втрачається, що робить платформу непридатною для серйозних оцінювань. Крім того, Google Forms не підтримує імпорт завдань з Excel, не має інтеграції зі сторонніми системами оцінювання або AI-сервісами.

Загалом, Google Forms добре підходить для простих опитувань і тестів, однак через обмежений функціонал та відсутність підтримки сценаріїв із збереженням сесій, вона не може виступати повноцінною платформою для управління онлайн-тестуванням.

Третім прикладом є платформа **Classtime** [3] (рис. 1.3) — інструмент для проведення інтерактивних уроків, оцінювання знань і контролю навчального процесу в реальному часі.

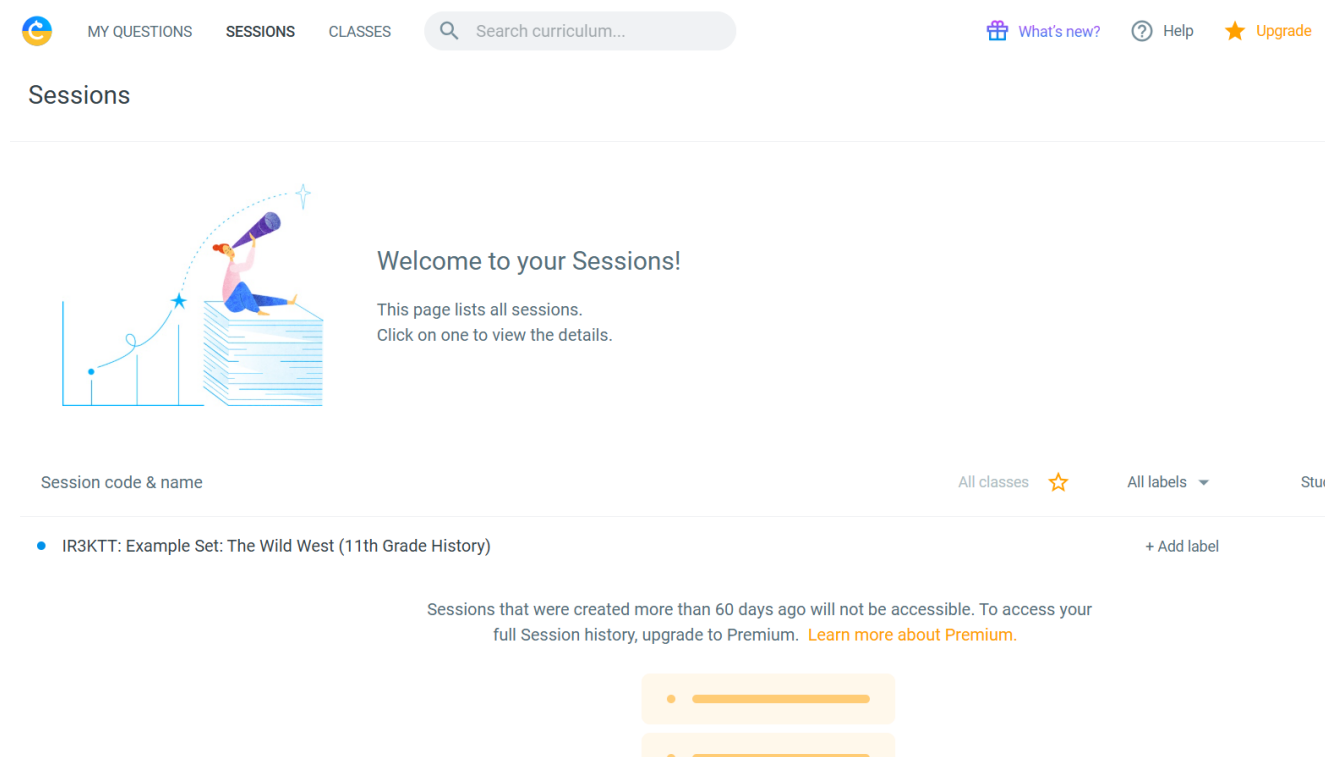


Рис. 1.3 – Панель управління сесіями у Classtime

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

Classtime орієнтована переважно на освітні установи, що проводять онлайн-навчання, а також викладачів, які працюють у змішаному або дистанційному форматі. Платформа дозволяє організовувати як контрольні роботи, так і тренувальні сесії для груп будь-якого розміру.

Classtime підтримує різноманітні формати запитань: одиночний та множинний вибір, відкриту відповідь, співставлення пар тощо. Існує підтримка автоматичного оцінювання, налаштування обмеження часу, а також змішування порядку запитань і відповідей.

Платформа має розвинену гнучкість налаштувань. Доступна авторизація за кодом або через обліковий запис, підтримуються кілька спроб, адаптивні сесії, а також командні ігрові режими для залучення учнів. Організатор має змогу зберегти та повторно використовувати сеанси, переглядати розгорнуту статистику по кожному учаснику.

Інтерфейс Classtime сучасний і добре оптимізований як для настільних, так і для мобільних пристроїв. Особлива увага приділена UX у контексті роботи викладача: система дозволяє швидко перемикатися між класами, дублювати сесії, проводити оцінювання наживо без зайвих переходів.

Серед сильних сторін Classtime — надійна система збереження відповідей: прогрес фіксується автоматично після кожної дії учня. У разі від'єднання учень може повернутись до тестування і продовжити з того місця, де зупинився. Це робить платформу надійною у контексті нестабільного інтернет-з'єднання.

Основним обмеженням є умовно безкоштовна модель: деякі розширені функції (наприклад, аналітика, експорт результатів, синхронізація з LMS) доступні лише у платній версії. Крім того, платформа не підтримує генерацію запитань за допомогою AI і має обмежену кастомізацію вигляду тесту для кінцевого користувача.

Загалом, Classtime — потужний інструмент для проведення інтерактивного тестування з високим рівнем стабільності та функціональності, однак для повноцінного адміністрування та масштабованості в корпоративному середовищі її можливості є обмеженими.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

Для наочності нижче наведено порівняльну таблицю (табл.1.1), яка відображає ключові характеристики кожної з розглянутих платформ. Це дозволяє узагальнити результати аналізу та виявити переваги й недоліки, що були враховані при розробці власної системи.

Таблиця 1.1

Таблиця порівняння платформ для онлайн тестувань

Характеристика	На Урок	Google Forms	Classtime	Aegos Exam
Основна аудиторія	Освітні заклади	Загальне використання	Освітні заклади, тренінги	Загальне використання
Формати тестування	Один/декілька варіантів, відкриті	Один/декілька варіантів, відкриті	Один/декілька варіантів, відкриті, співставлення	Один/декілька варіантів, відкриті
Інтерфейс	Простий, мінімалістичний	Мінімалістичний, знайомий	Сучасний, функціональний	Сучасний, функціональний
Збереження відповідей під час проходження	Так	Ні	Так	Так
Повернення до сесії після від'єднання	Ні	Ні	Так	Так
Імпорт/експорт запитань	Немає	Немає	Частково (в преміум)	Так (Excel)
AI-генерація питань	Немає	Немає	Немає	Так (Azure OpenAI)

Аналіз існуючих рішень показав, що жодна з представлених платформ не забезпечує повноцінного охоплення усіх потреб, пов'язаних з організацією, проведенням та адмініструванням онлайн-тестувань. Саме тому розробка «Aegos Exam» була спрямована на подолання цих обмежень і створення комплексного продукту, який поєднує в собі зручність, функціональність та інноваційні підходи.

У порівнянні з іншими платформами, «Aegos Exam» забезпечує не лише підтримку стандартних форматів тестових запитань і сучасного інтерфейсу, а й критично важливі можливості, які часто відсутні в аналогах: автоматичне збереження відповідей, повернення до сесії після втрати з'єднання, повноцінну підтримку ролевої моделі доступу, зручний імпорту/експорту через Excel, а також інтеграцію з Azure OpenAI для генерації запитань.

Завдяки такому функціональному наповненню платформа може бути корисною не лише для навчальних закладів, а й для команд або організаторів, які потребують зручного інструменту для перевірки знань у різних форматах [4].

1.3. Вибір архітектури

У процесі розробки веб-платформи для онлайн-тестування «Aegos Exam» ключовими критеріями архітектурного вибору стали масштабованість, модульність, зручність обслуговування та можливість інтеграції із сучасними хмарними сервісами. Проект орієнтований на підтримку широкого спектра функціональностей — від авторизації користувачів до автоматичної генерації запитань, — тому вимоги до архітектури були досить комплексними [5].

На основі аналізу вимог було ухвалено рішення реалізувати систему за принципами трирівневої архітектури [6] з чітким поділом відповідальностей між клієнтським, логічним і дата-рівнями. Такий підхід дозволяє розвивати систему поступово, не порушуючи вже існуючий функціонал, і дає змогу масштабувати кожен з рівнів незалежно. Саме ця гнучкість відіграє важливу роль при роботі з хмарною інфраструктурою Azure [7].

Користувацький інтерфейс платформи реалізовано на основі Razor Pages [8] — компонента веб-розробки в межах ASP.NET [9], який поєднує декларативний підхід до формування сторінок із прямою інтеграцією обробників подій. Це дозволяє досягти балансу між простотою реалізації й контролем над логікою представлення. Razor Pages ідеально підходить для додатків, де основний фокус — на формуванні зручного багаторівневого інтерфейсу з численними сценаріями взаємодії користувача.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

Серверна логіка побудована на основі .NET 8 [10] із дотриманням принципів чистої архітектури. Контролери відповідають лише за прийом HTTP-запитів [11] і делегування дій відповідним сервісам. Вся прикладна логіка інкапсульована у сервісних класах, а доступ до даних реалізовано через репозиторії. Така побудова забезпечує слабку зв'язаність компонентів, високу тестованість і можливість незалежного розвитку окремих функціональних модулів. Застосування патерна "Репозиторій" дозволяє забезпечити абстракцію над джерелами даних, що критично важливо при роботі з NoSQL-базою в хмарному середовищі.

У якості основи для зберігання даних обрано Azure Cosmos DB [12], який підтримує API для MongoDB [13]. Це рішення дозволяє поєднати зручність документоорієнтованої моделі з гнучким масштабуванням і високою доступністю, яку забезпечує інфраструктура Microsoft Azure. Схема зберігання адаптована до специфіки тестових систем: кожен тест, сесія, користувач або відповідь представлені у вигляді окремого документа з можливістю швидкого доступу та фільтрації.

На етапі вибору архітектурного підходу розглядалися також варіанти створення односторінкового застосунку (SPA [14]) з використанням фреймворків на кшталт React [15] або Angular [16], а також мікросервісна архітектура. Проте з урахуванням наявних ресурсів, очікуваної навантаженості й потреб у швидкому розгортанні було обрано монолітну структуру на Razor Pages. Вона демонструє оптимальне співвідношення між продуктивністю, простотою підтримки та адаптивністю.

Крім того, система вже адаптована до подальшого масштабування. Всі сервіси реалізовані як окремі інтерфейси з чіткими контрактами, що дозволяє при потребі винести частину функціональності у зовнішні мікросервіси без зміни основної логіки.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

1.4. Обґрунтування вибору технологічного стеку

У проєкті «Aegos Exam», який реалізує платформу для онлайн-тестування, ключовими критеріями відбору технологій стали надійність, масштабованість і простота підтримки. Система повинна обслуговувати різні типи користувачів, працювати стабільно під навантаженням та легко адаптуватися до змін бізнес-вимог. Однією з перших і найважливіших задач стало визначення підходу до реалізації веб-інтерфейсу.

У рамках платформи .NET було обрано ASP.NET Core Razor Pages як основний засіб побудови веб-інтерфейсу. На відміну від класичного ASP.NET MVC [17], Razor Pages орієнтована на сторінково-центричну модель, де вся логіка прив'язана до конкретної сторінки. Такий підхід дозволяє спростити структуру коду, прискорити розробку і зробити її більш інтуїтивно зрозумілою, що особливо актуально при роботі з великою кількістю форм і взаємодій на рівні користувача. Ця модель дозволила уникнути зайвої складності, пов'язаної з розподіленням контролерів по загальних секціях, і краще адаптувалась до ролей, що мають відокремлені інтерфейси.

У процесі створення «Aegos Exam» Razor Pages продемонструвала високу ефективність при реалізації функціоналу з чіткими сценаріями: створення тесту, редагування запитань, проходження сесії, відображення результатів. Це дало змогу будувати інтерфейс у логічно завершених частинах без надлишкової абстракції. Кожна сторінка містить модель ('PageModel'), що відповідає за обробку запитів та підготовку даних для представлення, що забезпечує кращу читаність і підтримуваність коду.

До переваг Razor Pages також можна віднести гнучке налаштування маршрутизації, повну підтримку форм, просту інтеграцію з компонентами UI та можливість використання всіх функцій .NET Core [18] (DI, middleware, фільтри тощо). Ці можливості дозволили реалізувати всі ключові вимоги системи — від безпечної авторизації до зручного управління контентом і підтримки генерації даних через сторонні сервіси.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

Проте у процесі аналізу також розглядалася можливість використання ASP.NET Core MVC. Цей підхід реалізує архітектурний шаблон Model-View-Controller і забезпечує повну гнучкість структурування системи, що особливо корисно при створенні складних багатомодульних додатків. MVC дозволяє суворо розмежувати контролери, моделі й уявлення, тим самим спрощуючи незалежне тестування й модифікацію логіки. Але для «Aegos Exam» така гнучкість виявилася надмірною, адже більшість сторінок орієнтовані на конкретний сценарій, і Razor Pages краще підходить для їхньої реалізації.

Зважаючи на всі технічні й організаційні аспекти, Razor Pages було обрано як оптимальний варіант для побудови інтерфейсу. Водночас для об'єктивного вибору було проведено системне порівняння між Razor Pages та ASP.NET Core MVC за низкою важливих параметрів [19] (табл.1.2).

Таблиця 1.2

Порівняння Razor Pages та ASP.NET Core MVC

Критерій	Razor Pages	ASP.NET Core MVC
Архітектурна модель	Сторінково-центрична (Page-based)	Компонентно-центрична (Model-View-Controller)
Простота структури	Простіша, менш конфігурована	Більш складна, потребує ручної організації контролерів та маршрутів
Маршрутизація	Просте налаштування автоматично пов'язане зі шляхами сторінок	Гнучка, але потребує явного визначення маршрутів
Швидкість розробки	Вища при реалізації CRUD-функцій та окремих сценаріїв	Менша при простих задачах, зате більш контрольована при складних
Підтримка ролей у навігації	Легко умовно відображати елементи меню прямо у .cshtml для кожної ролі	Потребує винесення логіки у ViewBag/ViewModel або окремі компоненти

Однією з ключових складових архітектури системи «Aegos Exam» є надійне, гнучке й масштабоване сховище даних. Оскільки платформа працює в

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

умовах великої кількості короткотривалих транзакцій (проходження тестів, збереження відповідей, ведення історії спроб), вибір технології бази даних був особливо критичним. Розглядалися кілька альтернатив: традиційні реляційні СУБД [20] (наприклад, Microsoft SQL Server або PostgreSQL), а також сучасні NoSQL-рішення [21] (MongoDB, Firebase, Amazon DynamoDB).

Попри високу узагальненість і підтримку транзакцій у реляційних СУБД, вони вимагають складнішого структурування схем, суворого нормалізованого дизайну та частіше стикаються з продуктивністю при роботі з динамічною структурою даних. У випадку «Aegos Exam», де дані мають документоподібну природу (тест, сесія, питання, відповідь), документоорієнтований підхід виявився значно ефективнішим.

У результаті аналізу було обрано Azure Cosmos DB із підтримкою API для MongoDB як основну технологію зберігання даних. Це дозволяє поєднати гнучкість документоорієнтованої моделі з перевагами хмарної інфраструктури Microsoft Azure: автоматичне масштабування, висока доступність, резервне копіювання й підтримка глобального розгортання. Формат документів MongoDB ідеально відповідає потребам платформи: дані сесії тестування, списки відповідей, шаблони запитань — усе зберігається у вигляді окремих об'єктів, з можливістю швидкого доступу, фільтрації та оновлення.

Окрім MongoDB, розглядалися й інші NoSQL-рішення. Наприклад, Firebase [22] від Google має швидку інтеграцію з мобільними додатками та вбудовану підтримку реального часу, але не надає зручних механізмів для складної агрегації даних або побудови незалежних бізнес-шарів. Amazon DynamoDB [23] демонструє хорошу масштабованість, але складніший у використанні, менш поширений серед .NET-розробників і потребує адаптації архітектури під key-value модель [24].

З огляду на все вищевикладене, Azure Cosmos DB з MongoDB API виявився оптимальним вибором для задач AegosExam. Для підтвердження доцільності рішення було підготовлено порівняльну таблицю між основними альтернативами (табл. 1.3).

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 1.3

Порівняння технологій зберігання даних

Критерій	Azure Cosmos DB	SQL Server	Firebase Realtime
Модель зберігання	Документоорієнтована (JSON)	Реляційна (таблична)	Документоорієнтована (JSON)
Складність інтеграції з .NET	Мінімальна, повна підтримка MongoDB бібліотек	Рідна підтримка в .NET	Обмежена, потребує сторонніх SDK
Механізми агрегації/запитів	Aggregation pipeline, гнучкі фільтри	SQL-запити з JOIN	Обмежені, без складної агрегації

У межах функціоналу автоматизованої генерації тестових запитань було інтегровано сервіс штучного інтелекту Azure OpenAI Service [25]. Цей сервіс надає можливість звертатися до передових мовних моделей через стабільний API від Microsoft, зберігаючи високі стандарти безпеки, доступності та масштабованості. Основною перевагою Azure OpenAI є повний контроль над середовищем виконання: дані обробляються в хмарній інфраструктурі Microsoft, доступ до ключів здійснюється через Azure Key Vault [26], а запити можуть бути детально логовані й обмежені на рівні політик безпеки.

Для реалізації генеративного механізму в «Aegos Exam» було обрано модель GPT-35-Turbo [27], яка демонструє відмінний баланс між продуктивністю, вартістю та якістю результатів. Ця модель здатна формувати логічно узгоджені, змістовні запитання як з одним, так і з кількома правильними відповідями, підтримує системні промпти та дає змогу контролювати стиль і структуру відповідей через параметри запиту. Інтеграція виконана з використанням SDK Azure.AI.OpenAI, що дозволяє гнучко керувати параметрами генерації та легко масштабувати рішення в майбутньому. і здатність створювати змістовні, логічно узгоджені тестові запитання як з одним, так і з кількома правильними варіантами.

Висновки до першого розділу

У першому розділі в результаті проведеного аналізу проблематики та огляду аналогічних платформ було виявлено, що жодне з наявних рішень не задовольняє повною мірою потреб у надійному, зручному й функціональному середовищі для онлайн-тестування. Саме це стало поштовхом до розробки власної системи, яка б об'єднала зручність інтерфейсу, стабільність роботи та розширений набір функцій.

Платформа «Aegos Exam» спроектована з урахуванням досвіду використання існуючих рішень і орієнтована на покриття потреб трьох основних категорій користувачів: учасників, організаторів та адміністраторів. Усі рішення — від вибору архітектури до механізмів авторизації — були спрямовані на створення системи, яка є не лише технічно стабільною, а й інтуїтивно зрозумілою.

Особливу увагу було приділено вибору технологічного стеку. Замість універсального, але складного ASP.NET MVC, було обрано Razor Pages — легшу та більш придатну для сценарного підходу до реалізації інтерфейсу. Для зберігання даних використано Azure Cosmos DB із MongoDB API, що забезпечує високу швидкодію та гнучкість при роботі з документоподібними структурами. Це рішення дозволило оптимально організувати збереження сесій, запитань і відповідей.

Інноваційною перевагою системи стала інтеграція з Azure OpenAI, яка дозволяє автоматизувати створення тестових запитань на основі заданої тематики. Обрана модель GPT-35-Turbo виявилася ефективною у створенні релевантного і різноманітного контенту, що значно спрощує роботу організаторів тестування.

Таким чином, основні концепції, закладені в основу архітектури та функціональності «Aegos Exam», створюють підґрунтя для масштабованої, безпечної та зручної системи онлайн-тестування, готової до впровадження в реальні освітні та професійні сценарії.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПЛАТФОРМИ ДЛЯ ОНЛАЙН
ТЕСТУВАНЬ

2.1. Визначення варіантів використання та об’єктно-орієнтованої
структури платформи для онлайн тестувань

На першому етапі проектування платформи «Aegos Exam» було визначено основні ролі користувачів системи та сценарії їх взаємодії з функціоналом платформи. Система підтримує три типи користувачів: **RegularUser** (звичайний користувач), **TestOrganizer** (організатор тестування) та **Admin** (адміністратор системи). Кожна з ролей має власний набір прав доступу та функціональних можливостей, що реалізовані згідно з принципами спадкування.

Роль звичайного користувача передбачає можливість входу в систему, проходження тестів, перегляду результатів, а також базові дії з профілем (редагування, видалення, зміна паролю). Основний сценарій взаємодії цієї ролі — отримання коду доступу до тесту, проходження тестування та перегляд оцінки. Також передбачено можливість повторного проходження та позначення запитань під час проходження тесту.

Роль організатора тестування включає всі функції звичайного користувача та додатковий функціонал для створення, редагування, публікації тестів, додавання запитань вручну або імпортом з Excel, а також генерації запитань за допомогою інтегрованого сервісу Azure OpenAI. Організатор може задавати обмеження на проходження (час, кількість спроб), переглядати результати студентів і експортувати створені тести.

Роль адміністратора системи успадковує можливості організатора і надає додаткові інструменти адміністрування: управління всіма користувачами та тестами, перегляд журналів дій, доступ до загальної статистики роботи системи. Це дозволяє реалізувати централізований контроль над функціонуванням платформи та її наповненням.

Варіанти використання описані у вигляді UML-діаграми [28] (рис. 2.1). Завдяки використанню механізму спадкування акторів, діаграма відображає логіку повторного використання функціональності між ролями.



Рис. 2.1 – Діаграма варіантів використання системи «Aegos Exam»

Окрім формалізації варіантів використання системи, було розроблено об'єктно-орієнтовану модель, яка відображає структуру основних сутностей платформи та забезпечує її розширюваність і легкість у супроводі. Базовим класом виступає User, що інкапсулює загальні властивості всіх користувачів, тоді як специфічні дані для ролей TestOrganizer та Admin реалізовані через розширення цієї моделі. Такий підхід дозволяє мінімізувати дублювання коду й підтримує логічну ієрархію ролей [29]. Крім того, об'єкти, пов'язані з тестуванням, зокрема Test, Question, AnswerOption, TestSession та ClientAnswer, реалізовані як окремі класи з чітко визначеними властивостями та зв'язками. Усі класи спроектовані відповідно до принципів SOLID, що дозволяє централізовано змінювати логіку, не порушуючи стабільність інших компонентів. Детальну UML-діаграму класів із відображенням зв'язків та структурних особливостей наведено в підрозділі 2.3.

Функціональні вимоги

Система «Aegos Exam» розробляється з урахуванням ролей користувачів, кожна з яких має доступ до певного набору функцій. Усі користувачі можуть зареєструватися та увійти в систему, при цьому автентифікація реалізована на основі cookie-based механізму з використанням ClaimsIdentity. Після авторизації користувач отримує доступ до власного профілю, який включає персональні дані та налаштування. Профіль можна редагувати, змінювати пароль або повністю видалити обліковий запис. Також доступна функція виходу з системи.

Звичайний користувач після входу до системи має змогу долучитися до тестування за допомогою унікального коду доступу, який генерується для кожного тесту. Перед початком тестування користувач ознайомлюється з описом, а у разі встановлених обмежень — бачить інформацію про максимальний час виконання та доступну кількість спроб. Під час тестування користувач взаємодіє з різними типами запитань: з однією правильною відповіддю, з кількома правильними відповідями, а також з відкритими текстовими полями. Передбачено можливість позначення запитань для

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

повторного перегляду (функція "помітити"), очищення введених відповідей, а також вільне перемикання між сторінками тесту зі збереженням даних. Після завершення сесії, якщо це дозволено організатором, користувач може ознайомитися з результатами або повторно пройти тест.

Організатор тестування має доступ до розширеного функціоналу, який охоплює створення, редагування, видалення та публікацію тестів. Для кожного тесту організатор визначає його назву, опис, час виконання, кількість спроб, опції випадкового перемішування питань і відповідей, доступність перегляду відповідей після завершення, а також параметри видимості. Запитання до тесту можуть бути додані вручну через інтерфейс, імпортовані з Excel-файлів (підтримується кілька режимів), або згенеровані автоматично за допомогою AI-моделі GPT-35-Turbo. Організатор має змогу редагувати та видаляти окремі запитання в межах тесту. Після завершення тестування організатор може переглядати як узагальнену статистику, так і результати окремих учасників, що дозволяє аналізувати успішність тестування за ключовими метриками.

Адміністратор системи має найширший обсяг прав у межах платформи. Він може здійснювати повне адміністрування користувачів — переглядати облікові записи, блокувати або розблоковувати доступ, а також остаточно видаляти користувачів із бази. Адміністратор також має доступ до всіх створених тестів і може керувати ними незалежно від організатора. Крім того, він може переглядати загальну системну статистику — кількість користувачів, кількість активних тестів, частоту проходжень, — а також працювати з журналами дій, де фіксуються ключові події: авторизації, створення тестів, запуск сесій тестування тощо.

Викладені функціональні вимоги стали основою для побудови логіки взаємодії між модулями та класами системи, забезпечили чіткий розподіл відповідальностей між ролями та дозволили закласти основу для масштабованої, безпечної та зручної веб-платформи. Детальний опис реалізації відповідного функціоналу подано у наступних підрозділах. під час побудови загальної

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

структури системи та слугували основою для реалізації відповідних модулів, класів і взаємодій, що докладно розглядаються в наступних підрозділах.

Нефункціональні вимоги

Нефункціональні вимоги до платформи «Aegos Exam» охоплюють аспекти, що забезпечують якість, стабільність, зручність та безпеку функціонування системи. Однією з ключових вимог є зручність використання (usability). Інтерфейс має бути інтуїтивно зрозумілим, логічно структурованим та доступним на різних пристроях — як на десктопах, так і на смартфонах чи планшетах. Навігація побудована за принципом мінімального кліку, що дає змогу користувачу досягати цільової дії за 2–3 кроки. Колірна палітра, типографіка та структура елементів відповідають принципам сучасного UX-дизайну.

Система демонструє високу надійність у роботі: вона здатна стабільно функціонувати за умов значного навантаження. У разі збоїв у роботі передбачено механізм резервного копіювання та можливість відновлення даних, а також окремі сторінки для обробки помилок. Розгортання в середовищі Azure Web App дозволяє підтримувати безперервну доступність із гарантованим SLA і проводити технічне обслуговування без суттєвого впливу на користувачів.

Продуктивність системи оцінюється за швидкістю реакції інтерфейсу на стандартні дії. Всі базові операції — вхід у систему, навігація сторінками, відображення тестів — мають виконуватися в межах 1–2 секунд. Обробка складних запитів, зокрема імпорту/експорту великої кількості запитань, або ж їх генерації з допомогою Azure OpenAI, не повинна перевищувати 20 секунд. Цього вдалося досягти шляхом застосування асинхронної обробки, кешування, а також оптимізації запитів до MongoDB через відповідні фільтраційні механізми.

Масштабованість — ще один важливий показник. Завдяки використанню документоорієнтованої моделі MongoDB та хмарної інфраструктури Azure, система може обробляти збільшену кількість тестів, користувачів та результатів без зміни архітектури. Горизонтальне масштабування бази, поділ

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

відповідальностей між модулями, а також реалізація окремих сервісів у вигляді інтерфейсів із подальшою DI-ін'єкцією — все це дозволяє розвивати платформу без шкоди для стабільності.

Питання безпеки займає провідне місце у переліку нефункціональних вимог. Реалізовано шифрування паролів з використанням хеш-функцій (SHA-256), захищене передавання даних через HTTPS, контроль доступу до сторінок на основі Claims із перевіркою ролей. Усі критичні змінні підключення зберігаються у Azure Key Vault.

Крім того, система повинна бути легко підтримуваною й оновлюваною. Завдяки модульній структурі, чітко виділеній логіці сервісів, використанню інтерфейсів для слабкого зв'язування компонентів і підтримці тестування за допомогою xUnit, платформа готова до інтеграції нових функцій без ризику порушення вже реалізованого функціоналу. Саме такі нефункціональні характеристики дозволяють вважати «Aegos Exam» готовим до використання в умовах реального навчального процесу. Для побудови логіки взаємодії між модулями та класами системи, забезпечили чіткий розподіл відповідальностей між ролями та дозволили закласти основу для масштабованої, безпечної та зручної веб-платформи. Детальний опис реалізації відповідного функціоналу подано у наступних підрозділах. під час побудови загальної структури системи та слугували основою для реалізації відповідних модулів, класів і взаємодій, що докладно розглядаються в наступних підрозділах.

Системні вимоги

Для забезпечення стабільної роботи веб-платформи «Aegos Exam» передбачено розгортання у хмарному середовищі Microsoft Azure, що дозволяє досягти високої доступності, автоматичного масштабування та централізованого управління інфраструктурою. Платформа функціонує в рамках ресурсної групи AegosExamRG у регіоні North Europe, використовуючи середовище розгортання Azure App Service на базі Windows. Домен розміщено за адресою aegos-exam.azurewebsites.net. На поточному етапі використовується план розгортання

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

Free (F1), що забезпечує базову функціональність і є придатним для тестового середовища.

Серверна частина реалізована з використанням технології .NET 8, яка дозволяє створювати високопродуктивні веб-додатки з використанням сучасного стеку інструментів. Система сумісна як із Windows, так і з основними Linux-дистрибутивами, зокрема Ubuntu 22.04 LTS або новішими. Для локального розгортання або розгортання поза межами Azure сервер має відповідати таким технічним характеристикам: процесор з мінімум чотирма ядрами, не менше восьми гігабайт оперативної пам'яті, твердотільний накопичувач із вільним простором від 50 ГБ, стабільне з'єднання з мережею та наявність публічної IP-адреси. Додатково необхідна наявність встановленого середовища виконання .NET SDK відповідної версії та доступного веб-сервера (наприклад, IIS або Nginx з reverse proxy).

У складі архітектури системи також задіяно кілька ключових хмарних сервісів. Для зберігання даних використовується Azure Cosmos DB з підтримкою MongoDB API, що дає змогу реалізувати гнучку документоорієнтовану модель із можливістю масштабування. Компонент Azure Key Vault відповідає за безпечне зберігання секретів, які не зберігаються у відкритому вигляді в конфігурації. Окремий компонент — Azure OpenAI — інтегрований для генерації тестових запитань на базі моделі GPT-35-Turbo. Уся взаємодія між сервісами виконується через захищені канали, а доступ до ресурсів регламентується через політики доступу та рольові обмеження.

Поточна інфраструктура платформи, що працює на безкоштовному тарифному плані Azure App Service (F1), є придатною для ознайомлення, тестування та індивідуального використання. Водночас вона має певні обмеження щодо продуктивності, кількості одночасних запитів і обсягу ресурсів. У разі впровадження платформи у реальному навчальному процесі або комерційному використанні доцільно перейти на стандартні чи преміум-плани, що забезпечують виділені ресурси, підтримку масштабування інстансів і

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

можливість інтеграції додаткових сервісів (наприклад, Application Insights, Load Balancer або Front Door для балансування навантаження).

Завдяки централізованому керуванню у середовищі Azure, платформа підтримує механізми резервного копіювання, моніторингу продуктивності та автоматичної доставки оновлень. Застосування Azure Key Vault дозволяє повністю уникнути витoku конфіденційної інформації завдяки зберіганню секретів поза межами коду. Контроль доступу до ресурсів відбувається на основі ролей і політик, що унеможливорює несанкціоноване втручання. Інтеграція з GitLab CI/CD забезпечує автоматичне тестування та розгортання змін, що значно підвищує стабільність і безперервність розробки.

На стороні клієнта система не висуває жорстких технічних вимог. Достатньо наявності сучасного веб-браузера з підтримкою HTML5, CSS3 та JavaScript (ES6).

Для локальної розробки, супроводу та модифікації системи рекомендується використовувати інтегроване середовище розробки Visual Studio 2022 або новішу версію. Це середовище забезпечує зручну роботу з Razor Pages, підтримку ASP.NET, інтеграцію з Azure, а також дозволяє ефективно керувати залежностями через NuGet. Для візуалізації структури бази даних і ручної роботи з документами MongoDB доцільно застосовувати MongoDB Compass. Розробка ведеться з використанням системи контролю версій Git, а зберігання коду організовано в зовнішньому репозиторії GitLab, що забезпечує гнучкість при командній роботі та можливість налаштування CI/CD-процесів.

Аналіз функціональних вимог дозволив виділити наступні сутності, що забезпечать програмну реалізацію додатку. Щоб візуалізувати взаємозв'язки між цими сутностями, доцільно побудувати діаграму класів [30] (рис. 2.2), яка наочно демонструє структуру даних, успадкування, композицію та основні властивості кожного класу.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				31
Змн.	Арк.	№ докум.	Підпис	Дата		

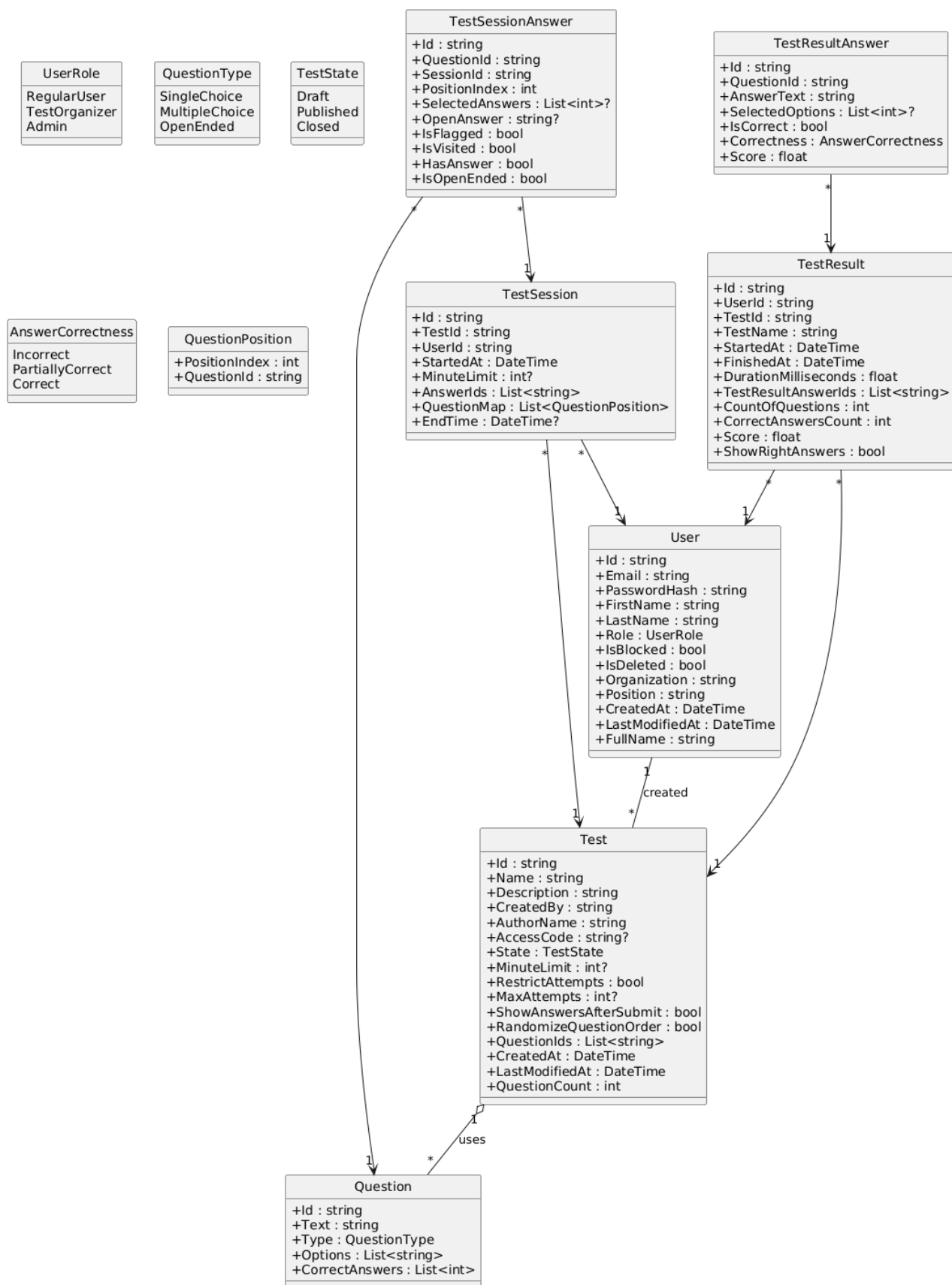


Рис. 2.2 – Діаграма класів основних сутностей

У системі «Aegos Exam» клас User є центральною сутністю для управління обліковими записами. Він об'єднує всі ролі — звичайних користувачів,

організаторів та адміністраторів — через поле Role (перелічуваний тип UserRole). Унікальний ідентифікатор Id зберігається як об'єкт MongoDB (ObjectId у вигляді рядка). Авторизація реалізується через поле Email, а пароль зберігається у вигляді хешу (PasswordHash), що гарантує безпечне зберігання облікових даних. Статус облікового запису визначається прапорцями IsBlocked (тимчасове блокування) і IsDeleted (логічне видалення).

Для персоналізації інтерфейсу передбачено поля FirstName, LastName, а для організаторів додатково Organization і Position. Час створення профілю (CreatedAt) та час останньої модифікації (LastModifiedAt) зберігаються в UTC-форматі. Властивість FullName, яка об'єднує ім'я та прізвище, спрощує відображення імені в інтерфейсі. Така структура дозволяє централізовано зберігати всі облікові записи в одній колекції MongoDB і масштабувати доступ до функцій системи на рівні логіки.

Клас Test моделює сутність тесту. Він включає базову інформацію: Name, Description, унікальний AccessCode для доступу, а також ідентифікатори автора (CreatedBy, AuthorName). Для керування логікою проходження передбачено поля MinuteLimit, RestrictAttempts, MaxAttempts, ShowAnswersAfterSubmit, RandomizeQuestionOrder. Статус тесту визначається через поле State (перелік TestState: Draft, Published, Closed). Для підтримки аналітики зберігаються часові мітки створення (CreatedAt) та останньої зміни (LastModifiedAt). Список QuestionIds посилається на запитання, що входять до тесту.

Клас Question моделює окреме тестове запитання. Він містить унікальний ідентифікатор, текст запитання (Text), тип (QuestionType — SingleChoice, MultipleChoice, OpenEnded), список варіантів (Options) і правильні відповіді у вигляді списку індексів (CorrectAnswers). Це забезпечує гнучке зберігання як закритих, так і відкритих типів запитань.

Допоміжна структура QuestionPosition дозволяє зберігати порядок відображення запитань у тесті або сесії. Вона складається з полів PositionIndex (порядковий номер) та QuestionId. Такий підхід забезпечує як фіксовану, так і випадкову послідовність запитань під час проходження тесту.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

Клас `TestSession` відповідає за сесію проходження тесту конкретним користувачем. Він містить `TestId`, `UserId`, часову мітку початку (`StartedAt`) та, за потреби, ліміт часу (`MinuteLimit`). Список `AnswerIds` зберігає ідентифікатори відповідей (`TestSessionAnswer`), а поле `QuestionMap` — список `QuestionPosition`, який відображає структуру конкретної сесії. Властивість `EndTime` визначає допустимий момент завершення сесії з урахуванням обмеження часу.

`TestSessionAnswer` є моделлю для збереження відповідей у межах сесії. Вона зберігає `QuestionId`, `SessionId`, порядковий індекс (`PositionIndex`), список обраних варіантів (`SelectedAnswers`) або відкриту відповідь (`OpenAnswer`). Додатково реалізовано прапорці `IsFlagged` (позначено як важливе) та `IsVisited` (переглянуто), а також обчислювані властивості `HasAnswer` (чи є відповідь) і `IsOpenEnded` (чи відкрите запитання).

`TestResult` — це узагальнена модель результату проходження тесту. Вона зберігає ідентифікатори користувача (`UserId`) та тесту (`TestId`), назву тесту (`TestName`), часові мітки початку та завершення (`StartedAt`, `FinishedAt`) і обчислюване поле `DurationMilliseconds`, яке визначає тривалість проходження. Результат включає список відповідей (`TestResultAnswerIds`), загальну кількість запитань (`CountOfQuestions`), кількість правильних відповідей (`CorrectAnswersCount`), фінальний бал (`Score`) та прапорець `ShowRightAnswers`, який визначає, чи буде показано правильні відповіді після завершення тесту.

Клас `TestResultAnswer` деталізує відповідь користувача на конкретне запитання в рамках результату. Він включає `QuestionId`, текст відповіді (`AnswerText`), список обраних варіантів (`SelectedOptions`), логічне поле `IsCorrect` (повністю правильна відповідь), числову оцінку (`Score`) та поле `Correctness`, що реалізоване як перелік `AnswerCorrectness` (`Incorrect`, `PartiallyCorrect`, `Correct`). Це дає змогу реалізувати часткове оцінювання при складних запитаннях із кількома правильними варіантами або відкритими відповідями.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

2.2. Розробка бази даних системи

У процесі розробки платформи «Aegos Exam» важливою частиною стало проєктування структури бази даних. Обрана документоорієнтована модель MongoDB дозволяє зручно зберігати складні вкладені структури, зокрема тести з масивами запитань, відповіді користувачів, сесії проходження та результати. Гнучка схема MongoDB значно спрощує розширення та адаптацію системи до нових вимог без потреби в складних міграціях.

Структура бази даних побудована з урахуванням логічного поділу на функціональні блоки. Для кожної сутності використовується окрема колекція, структура якої представлена у вигляді таблиці з переліком полів, типами та призначенням. Такий підхід дозволяє швидко орієнтуватися у моделі, забезпечує зрозумілу документацію та полегшує супровід.

У наступних таблицях 2.1-2.7 наведено опис основних колекцій MongoDB, які реалізовані в системі «Aegos Exam».

Таблиця 2.1

Опис колекції Users

Поле	Тип	Призначення
Id	string	Унікальний ідентифікатор користувача
Email	string	Адреса електронної пошти
PasswordHash	string	Хеш пароля користувача
FirstName	string	Ім'я користувача
LastName	string	Прізвище користувача
Role	enum UserRole	Роль користувача (RegularUser, TestOrganizer, Admin)
IsBlocked	bool	Ознака блокування облікового запису
IsDeleted	bool	Ознака логічного видалення облікового запису
Organization	string	Назва організації
Position	string	Посада

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

Продовження таблиці 2.1

CreatedAt	datetime	Дата створення запису
LastModifiedAt	datetime	Дата останньої модифікації

Таблиця 2.2

Опис колекції Tests

Поле	Тип	Призначення
Id	string	Унікальний ідентифікатор тесту
Name	string	Назва тесту
Description	string	Опис тесту
CreatedBy	string	Ідентифікатор автора тесту
AuthorName	string	Ім'я автора
AccessCode	string	Код доступу до тесту
State	enum TestState	Статус тесту
MinuteLimit	int?	Ліміт часу на проходження
RestrictAttempts	bool	Ознака обмеження кількості спроб
MaxAttempts	int?	Максимальна кількість спроб
ShowAnswersAfterSubmit	bool	Показ відповідей після проходження
RandomizeQuestionOrder	bool	Рандомізація порядку запитань
QuestionIds	List<string>	Список ідентифікаторів запитань
CreatedAt	datetime	Дата створення тесту
LastModifiedAt	datetime	Дата останньої модифікації тесту
LastSessionAt	datetime	Дата останньої сесії проходження

Таблиця 2.3

Опис колекції Questions

Поле	Тип	Призначення
Id	string	Унікальний ідентифікатор запитання
Text	string	Текст запитання
Type	enum QuestionType	Тип запитання
Options	List<string>	Список варіантів відповіді
CorrectAnswers	List<int>	Індекси правильних відповідей

Таблиця 2.4

Опис колекції TestSessions

Поле	Тип	Призначення
Id	string	Унікальний ідентифікатор сесії
TestId	string	Ідентифікатор тесту
UserId	string	Ідентифікатор користувача
StartedAt	datetime	Дата та час початку проходження
MinuteLimit	int?	Ліміт часу (якщо задано)
AnswerIds	List<string>	Список ідентифікаторів відповідей
QuestionMap	List<QuestionPosition>	Порядок запитань у сесії

Таблиця 2.5

Опис колекції TestSessionAnswers

Поле	Тип	Призначення
Id	string	Ідентифікатор відповіді
QuestionId	string	Ідентифікатор запитання
SessionId	string	Ідентифікатор сесії
PositionIndex	int	Позиція запитання в межах сесії

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

Продовження таблиці 2.5

SelectedAnswers	List<int>?	Список вибраних варіантів (для варіантів)
OpenAnswer	string?	Відкрита відповідь (текст)
IsFlagged	bool	Позначено для перегляду пізніше
IsVisited	bool	Чи було відвідано запитання

Таблиця 2.6

Опис колекції TestResults

Поле	Тип	Призначення
Id	string	Унікальний ідентифікатор результату
UserId	string	Ідентифікатор користувача
TestId	string	Ідентифікатор тесту
TestName	string	Назва тесту
StartedAt	datetime	Час початку проходження
FinishedAt	datetime	Час завершення
TestResultAnswerIds	List<string>	Список ідентифікаторів відповідей у результаті
CountOfQuestions	int	Загальна кількість запитань
CorrectAnswersCount	int	Кількість правильних відповідей
Score	float	Набраний бал
ShowRightAnswers	bool	Чи показувати правильні відповіді

Таблиця 2.7

Опис колекції TestResultAnswers

Поле	Тип	Призначення
Id	string	Ідентифікатор відповіді в результаті
QuestionId	string	Ідентифікатор запитання

Продовження таблиці 2.7

AnswerText	string	Відповідь текстом
SelectedOptions	List<int>?	Обрані варіанти (індекси)
IsCorrect	bool	Оцінка правильності
Correctness	enum AnswerCorrectness	Рівень правильності (повна, часткова, неправильна)
Score	float	Оцінка за це запитання

Отже, структура бази даних системи «Aegos Exam» побудована з урахуванням вимог до гнучкості та масштабованості. Використання документоорієнтованого підходу на основі MongoDB дозволило організувати дані у вигляді логічно розділених колекцій, які відображають ключові сутності платформи — користувачів, тести, запитання, сесії проходження, відповіді та результати. Така структура забезпечує зручну інтеграцію з бізнес-логікою системи, підтримує швидке зчитування інформації, полегшує реалізацію аналітичного функціоналу й дозволяє безперешкодно розширювати функціональні можливості без необхідності суттєвих змін у моделі даних.

2.3. Проектування та реалізація алгоритмів роботи системи

Одним із ключових алгоритмів у системі «Aegos Exam» є створення тестування, яке виконується організатором через інтерфейс платформи. Цей процес включає не лише створення заголовка тесту та налаштувань (час, кількість спроб, доступ за кодом), а й послідовне додавання запитань, які зберігаються окремо в колекції. На діаграмі послідовності нижче (рис.2.3) зображено послідовність дій, що відбуваються при створенні тесту, починаючи з ініціації запиту до фінальної публікації.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

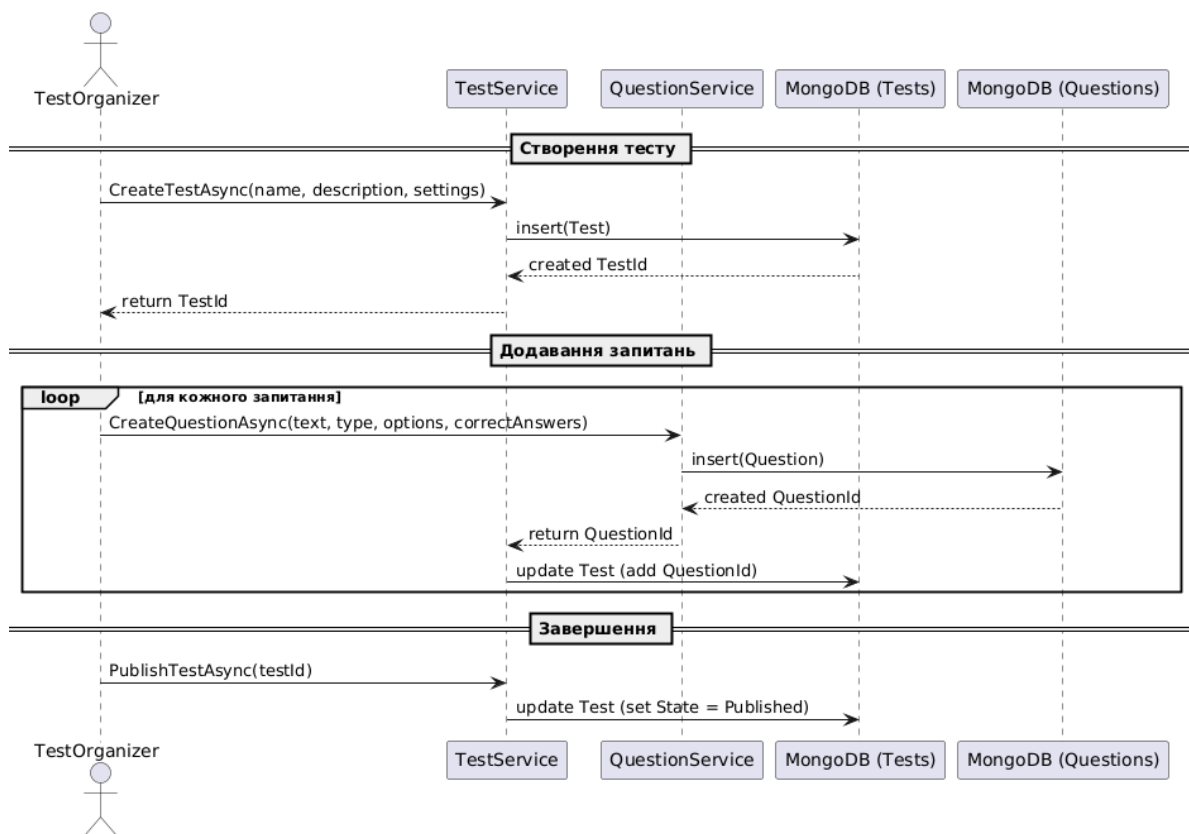


Рис. 2.3 – Діаграма послідовності створення тестування Організатором Тестів

У цьому сценарії ініціатором дії є користувач з роллю TestOrganizer, який викликає метод CreateTestAsync() у сервісі TestService. На цьому етапі формується об'єкт Test, що містить назву, опис, параметри доступу та інші налаштування. Цей об'єкт передається до колекції Tests у MongoDB:

```

var test = new Test {
    Name = input.Name,
    Description = input.Description,
    AccessCode = GenerateAccessCode(),
    CreatedBy = userId,
    CreatedAt = DateTime.UtcNow,
    State = TestState.Draft
};

await _testRepository.AddAsync(test);
  
```

На цьому етапі важливо, що тест створюється у стані «чернетка» (Draft) — це дозволяє поступово додавати до нього запитання, змінювати налаштування та редагувати структуру до моменту публікації. Така модель збереження є

надзвичайно гнучкою: вона дозволяє зберігати основний документ тесту навіть без жодного питання, підтримуючи можливість багатокрокової взаємодії. Це особливо корисно у випадках, коли організатор тестування не має змоги сформувати повний тест одразу, а хоче повертатися до його редагування протягом тривалого часу.

Після створення тесту користувач може додавати запитання. Кожне запитання створюється через окремий сервіс `QuestionService`, а повернені `QuestionId` додаються до тесту:

```
var question = new Question {  
    Text = input.Text,  
    Type = input.Type,  
    Options = input.Options,  
    CorrectAnswers = input.CorrectAnswers  
};  
  
await _questionRepository.AddAsync(question);  
await _testRepository.AddQuestionIdAsync(testId, question.Id);
```

Цей цикл може повторюватися кілька разів — кожен запит створення питання супроводжується вставкою у відповідну колекцію та оновленням документа тесту.

Завершується процес викликом методу `PublishTestAsync()`, який змінює стан тесту на `Published`. Після цього тест стає доступним для проходження.

Завдяки такій структурі реалізації та послідовності взаємодії між компонентами система забезпечує чисту модульну логіку, високу гнучкість і можливість масштабування в майбутньому.

Ще одним ключовим алгоритмом, реалізованим у системі «Aegos Exam», є проходження тесту звичайним користувачем. Цей процес охоплює початок сесії тестування, покрокове надання відповідей на запитання, збереження відповідей у базі даних та формування фінального результату після завершення. На діаграмі послідовності нижче (рис.2.4) зображено цей процес.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

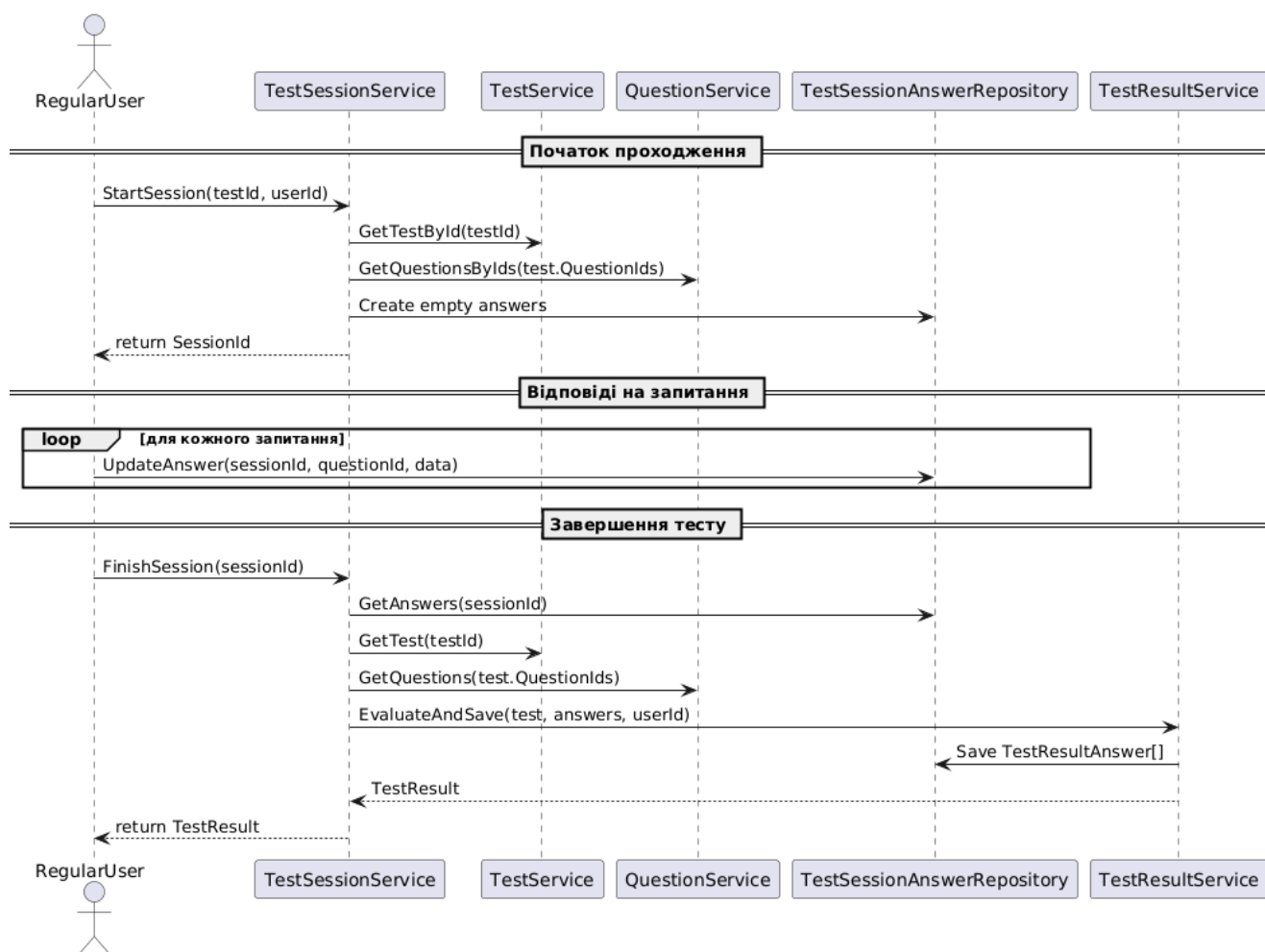


Рис. 2.4 – Діаграма послідовності проходження тестування

На початку взаємодії користувач ініціює проходження тесту, викликаючи метод `StartSession()` сервісу `TestSessionService`, передаючи ідентифікатор тесту та свій `userId`. Сервіс отримує відповідний тест і пов'язані з ним запитання, після чого створює порожні об'єкти відповідей (`TestSessionAnswer`) для кожного запитання в окремій колекції. Користувачу повертається ідентифікатор сесії (`SessionId`), який буде використовуватись під час проходження.

Під час проходження тесту, кожна відповідь користувача зберігається окремо через виклик методу оновлення: `UpdateAnswer(sessionId, questionId, data)` у відповідному репозиторії. Завдяки цьому реалізовано покрокове збереження, яке дозволяє користувачу не втрачати прогрес при оновленні сторінки або відключенні.

Після завершення тесту користувач викликає метод `FinishSession()`. Сервіс зчитує всі надані відповіді (`GetAnswers`), повторно отримує структуру тесту й

перелік запитань, після чого передає їх у `TestResultService`. Оцінювання здійснюється відповідно до правил, заданих у тесті (врахування правильності, часткової відповідності тощо), після чого зберігаються результати по кожному запиту (TestResultAnswer) і формується загальний об'єкт результату (TestResult).

Цей підхід дозволяє реалізувати гнучкий механізм оцінювання, підтримку різних типів запитань, а також надає можливість адаптації для відновлення сесії, перевірки відповідей, аналітики тощо. Результат тесту повертається користувачеві для відображення на відповідній сторінці інтерфейсу.

Однією з ключових функцій системи є автоматичний підрахунок результатів тестування після завершення сесії. Цей процес є повністю автоматизованим і запускається після дій користувача на клієнтському інтерфейсі. Нижче представлено діаграму послідовності (рис.2.5), яка демонструє алгоритм обробки запитів, оцінювання відповідей та збереження результату тестування.

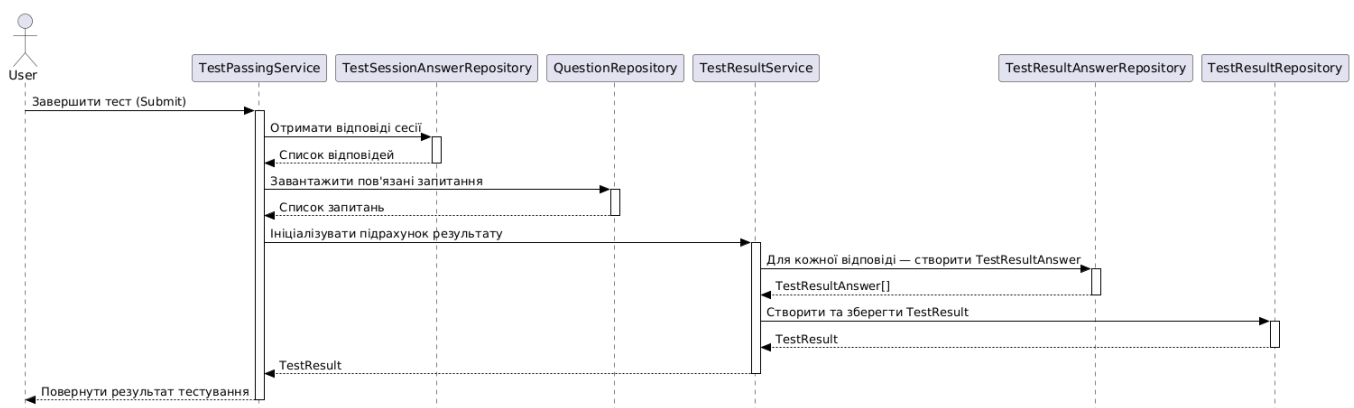


Рис. 2.5 – Діаграма послідовності підрахунку результатів

У процесі проходження тесту користувач поступово надає відповіді, які зберігаються в окремій колекції `TestSessionAnswer`. Після натискання кнопки "Завершити тест" ініціюється метод `SubmitAsync` у `TestPassingService`. Цей виклик починає ланцюг обробки, мета якого — сформувати об'єкт результату (`TestResult`), оцінити правильність відповідей і зберегти результати для подальшого аналізу. На діаграмі послідовності цей виклик представлено стрілкою від актора `RegularUser` до об'єкта `TestPassingService`.

Першим кроком `TestPassingService` звертається до `TestSessionAnswerRepository` для отримання всіх відповідей, які користувач надав у межах сесії. У коді це виглядає як `var answers = await _sessionAnswerRepo.GetBySessionIdAsync(sessionId);`. Отриманий список використовується для наступного кроку — завантаження всіх відповідних запитань із бази даних. Це реалізується через метод `GetByIdsAsync`, викликаний у `QuestionRepository`: `var questions = await _questionRepo.GetByIdsAsync(ids);`.

Після отримання як відповідей користувача, так і повної інформації про запитання, сервіс `TestPassingService` передає їх до `TestResultService`. Цей сервіс відповідає за оцінювання: зіставлення наданих відповідей із правильними, розрахунок балів, визначення коректності (включаючи часткову правильність) і створення об'єктів `TestResultAnswer`. Цей процес може бути реалізований через LINQ-вираз: `var resultAnswers = answers.Select(ans => CreateResultAnswer(ans, questions)).ToList();`.

Створені об'єкти `TestResultAnswer` зберігаються за допомогою `TestResultAnswerRepository`. Після цього формується основний об'єкт результату — `TestResult`, який містить узагальнену інформацію: час проходження, кількість запитань, кількість правильних відповідей, підсумковий бал і інші метадані. У коді створення виглядає приблизно так: `var result = new TestResult { ... };`, а збереження — `await _testResultRepo.AddAsync(result);`.

На завершальному етапі об'єкт `TestResult` повертається назад до `TestPassingService`, а той — до інтерфейсу користувача, де результат виводиться на сторінці підсумків: `return View("Result", result);`. Таким чином, реалізовано повний автоматичний цикл перевірки результатів без втручання адміністратора або організатора.

На діаграмі нижче (рис.2.6) представлено загальну архітектуру системи «Aegos Exam» у вигляді діаграми компонентів. Такий тип діаграми дозволяє наочно продемонструвати основні модулі програмного забезпечення, їх взаємозв'язки та залежності. Компонентна структура системи розроблена відповідно до принципів чистої архітектури (Clean Architecture), що забезпечує

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

поділ відповідальностей, гнучкість при внесенні змін і легкість у тестуванні окремих частин системи.

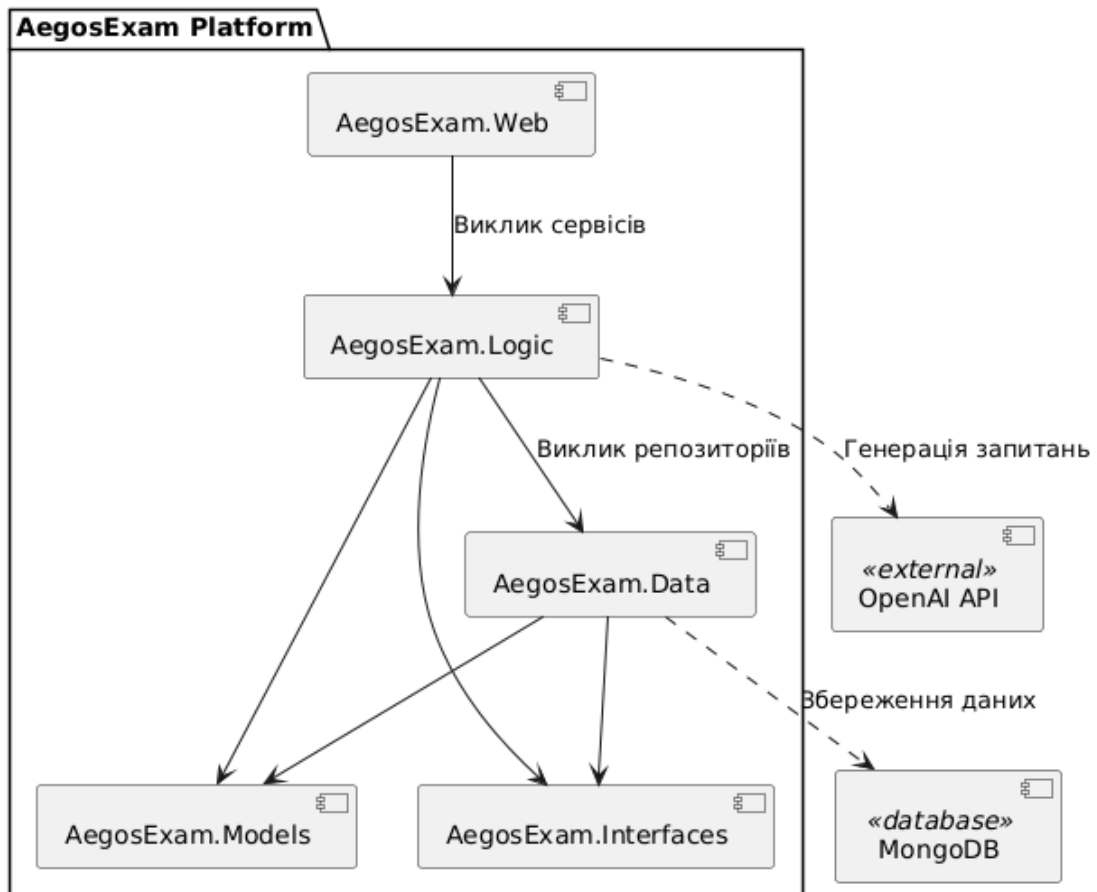


Рис. 2.6 – Діаграма компонентів

Центральним елементом є модуль AegosExam.Web, який реалізує користувацький інтерфейс на основі Razor Pages. Цей компонент відповідає за обробку HTTP-запитів, відображення сторінок, обробку форм та інтеграцію з бізнес-логікою через відповідні сервіси.

Всю логіку обробки даних і реалізацію сценаріїв взаємодії із системою зосереджено в модулі AegosExam.Logic. Саме цей компонент реалізує ключову функціональність, зокрема створення тестів, управління сесіями проходження, генерацію запитань, обчислення результатів та авторизацію користувачів. Для взаємодії з іншими частинами системи (Web, Data) бізнес-логіка використовує інтерфейси з модуля AegosExam.Interfaces, що дозволяє досягти слабкого зв'язування та підтримати принцип залежності від абстракцій.

Доступ до бази даних реалізується в компоненті AegosExam.Data, який містить класи-репозиторії для роботи з MongoDB. Репозиторії інкапсулюють всі CRUD-операції над колекціями та слугують єдиною точкою взаємодії з базою даних. Всі запити до сховища проходять через цей шар, що дозволяє чітко розмежувати бізнес-логіку та технічну реалізацію зберігання.

Окремим компонентом є AegosExam.Models, де описані всі основні сутності системи: користувачі, тести, запитання, сесії, відповіді, результати тощо. Вони визначають структуру документів MongoDB, використовуються у всіх модулях та є спільним типізованим контрактом для передачі даних.

Зовнішні залежності системи винесені за межі основного пакету: MongoDB виступає основним документоорієнтованим сховищем, у якому зберігається вся бізнес-інформація, а OpenAI API підключається до модуля Logic у випадку використання функціоналу генерації тестових запитань. Таким чином, система легко масштабована та дозволяє підключати сторонні сервіси без порушення загальної архітектури.

Подібна компонентна побудова забезпечує гнучкість розробки, модульність, повторне використання коду, а також розділення відповідальностей. Завдяки чіткому структуруванню та використанню інтерфейсів платформа «Aegos Exam» може бути легко розширена новими функціональними модулями без значного втручання в існуючий код.

2.4. Застосування архітектурних патернів у побудові системи

У процесі розробки веб-платформи "Aegos Exam" особливу увагу було приділено організації внутрішньої архітектури системи, що визначає структуру взаємодії між компонентами та забезпечує стабільність, масштабованість і зручність підтримки. В умовах розвитку проєкту, що охоплює створення тестів, проходження сесій, генерацію результатів, а також зовнішні інтеграції (наприклад, з OpenAI API), важливою задачею стало впровадження архітектурних рішень, які дозволяють ізолювати бізнес-логіку, спростити тестування та полегшити розширення функціоналу.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

З цією метою в системі було застосовано два класичних архітектурних патерни — "Репозиторій" (Repository Pattern) і "Сервісний шар" (Service Layer). Обидва патерни мають чітко окреслену відповідальність та спрощують реалізацію принципів розділення обов'язків (Separation of Concerns), що, у свою чергу, позитивно впливає на модульність і зрозумілість коду.

Першим розглянемо патерн Репозиторію. Його завдання — ізолювати логіку доступу до бази даних і надати уніфікований інтерфейс для взаємодії з колекціями MongoDB. Замість того, щоб здійснювати виклики до бази безпосередньо в контролерах або сервісах, у проєкті було реалізовано відповідні інтерфейси (наприклад, `ITestRepository`), які описують доступні операції над сутностями. Реалізації цих інтерфейсів (`TestRepository`) інкапсулюють виклики до MongoDB та відповідають за CRUD-операції.

На діаграмі класів нижче (рис.2.7) представлено реалізацію патерну Репозиторій у контексті роботи з об'єктами тестів. Вона ілюструє взаємозв'язки між інтерфейсом, конкретною реалізацією та моделлю `Test`, а також логічне відокремлення бази даних від бізнес-логіки додатку.

На діаграмі (рис.2.7) видно, що інтерфейс `ITestRepository` декларує основні методи доступу до колекції тестів, зокрема отримання одного чи всіх елементів, додавання, оновлення й видалення. Клас `TestRepository` реалізує ці методи й містить приватне поле `_collection`, яке відповідає за пряме звернення до MongoDB. Така реалізація дозволяє сервісам взаємодіяти з репозиторієм через інтерфейс, не залежачи від конкретної реалізації або типу сховища. Це не лише підвищує модульність, а й полегшує тестування (через можливість мокування інтерфейсу), а також дозволяє змінити джерело даних без переписування бізнес-логіки.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

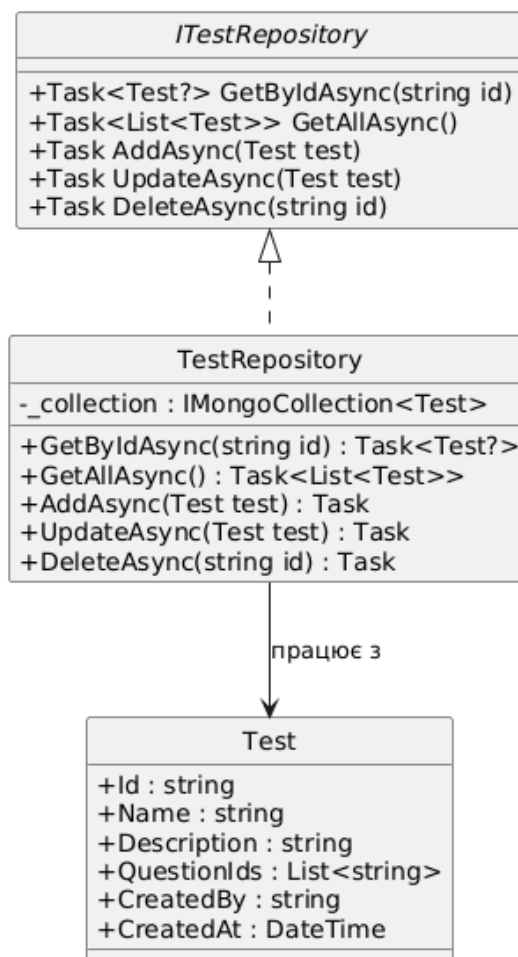


Рис. 2.7 – Діаграма класів реалізації патерну Репозиторій

Другим важливим структурним елементом є "Сервісний шар" — патерн, що відповідає за інкапсуляцію всієї бізнес-логіки системи та виступає посередником між контролерами (або іншими споживачами логіки) та репозиторіями. Замість того, щоб вбудовувати складні правила прямо в контролери або DAO, вони зосереджені у відповідних сервісах. Наприклад, клас `TestResultService` обробляє всю логіку створення результатів проходження тесту, включаючи підрахунок балів, обробку частково правильних відповідей і збереження агрегованої інформації в базі даних.

Сервіси реалізуються через інтерфейси (наприклад, `ITestResultService`), що полегшує тестування, підвищує модульність і дозволяє легко змінити реалізацію. Нижче наведено спрощену UML-діаграму [31] (рис.2.8), яка ілюструє реалізацію цього патерну у рамках створення результату тестування.

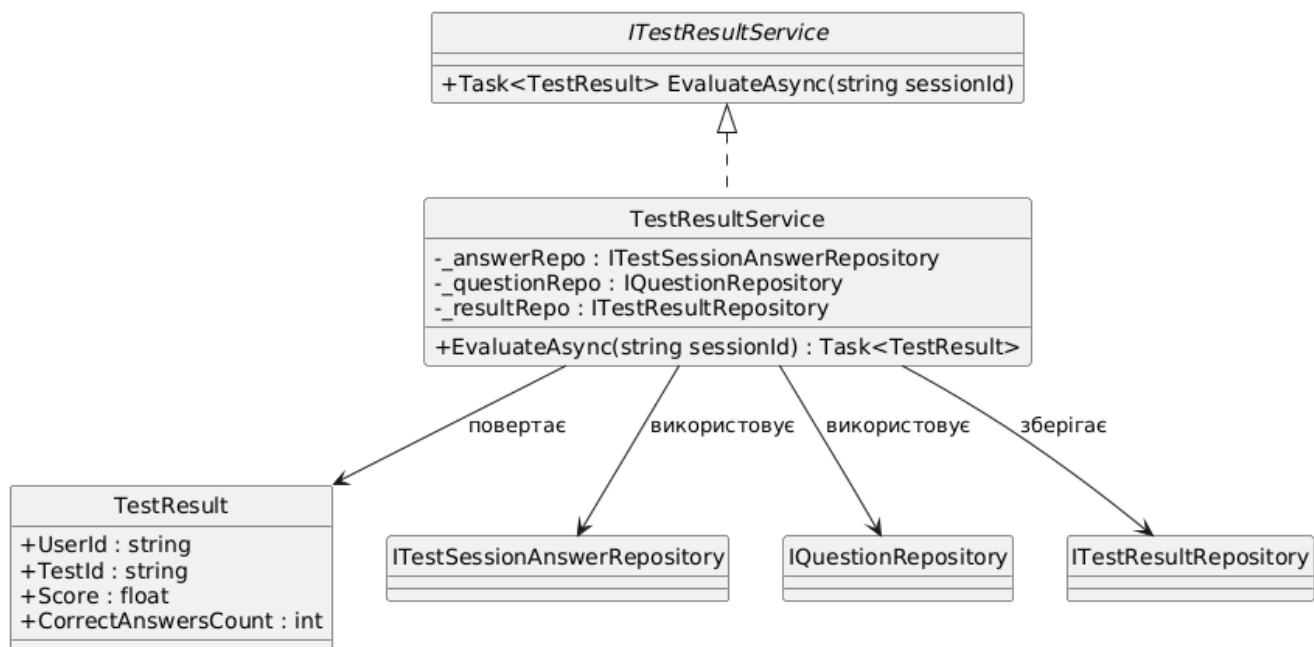


Рис. 2.8 – Діаграма класів реалізації патерну "Сервісний шар"

Діаграма демонструє чітку ізоляцію бізнес-логіки у класі `TestResultService`, який використовує інші сервіси або репозиторії для доступу до даних, виконує підрахунки та повертає модель `TestResult`. Завдяки такому підходу функціональність легко розширюється — наприклад, можна впровадити нову стратегію оцінювання або змінити джерело даних, не змінюючи інші частини системи. Така організація коду відповідає принципам чистої архітектури та дозволяє масштабувати й підтримувати систему в довгостроковій перспективі.

Щоб продемонструвати, як взаємодіють усі шари системи на практиці, розглянемо типовий приклад виклику — створення нового тесту. Цей сценарій охоплює всі рівні: інтерфейс користувача, контролер, сервіс, репозиторій та збереження в базі даних. Контролер приймає модель вводу (наприклад, `TestInputModel`) та викликає відповідний метод у сервісі. Сервіс виконує валідацію, створює екземпляр моделі `Test`, та делегує збереження до `ITestRepository`. Це дозволяє легко масштабувати логіку в сервісі без зміни контролера або нижчого шару доступу до даних. Логіку ввиконання можна переглянути на діаграмі нижче(рис.2.9).

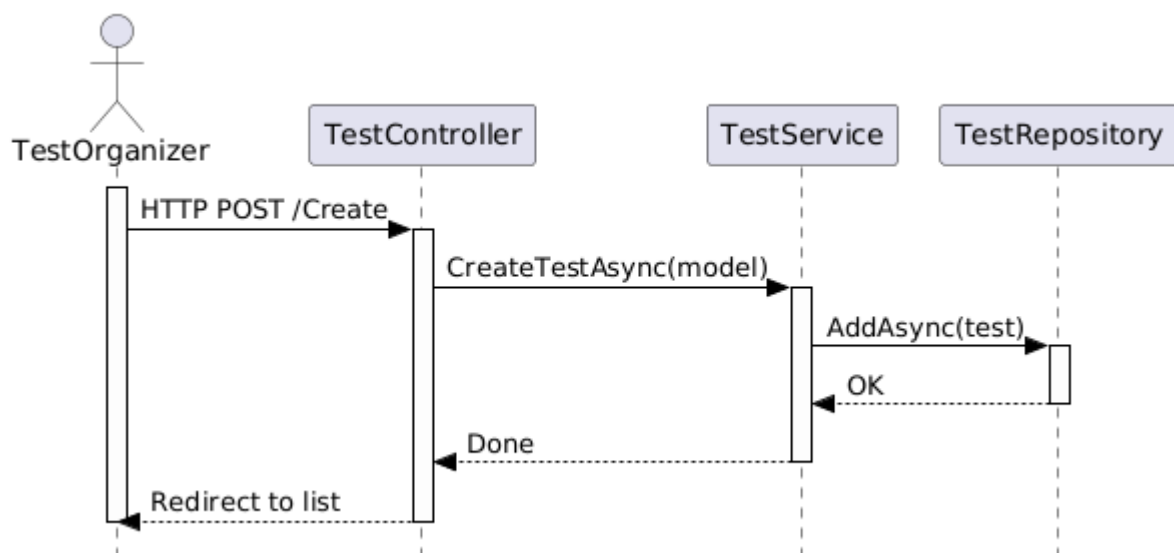


Рис. 2.9 – Діаграма активності типового виклику

Висновки до другого розділу

У третьому розділі було розглянуто архітектурну побудову платформи для онлайн тестувань та її реалізацію на рівні структурних компонентів, логіки взаємодії, сховища даних і патернів проєктування. Система побудована відповідно до принципів чистої архітектури, що дозволяє чітко розділити відповідальність між модулями, забезпечити слабке зв'язування та створити зручні умови для супроводу та розширення функціоналу.

Розділ починається з визначення ключових ролей користувачів (RegularUser, TestOrganizer, Admin), кожна з яких взаємодіє з платформою у межах своїх повноважень. Це стало основою для формалізації варіантів використання через UML-діаграми, які відображають основні сценарії дій у системі. Всі дії користувачів реалізовано у вигляді модульних сервісів, що дозволяє легко оновлювати функціональність без впливу на інші частини системи.

Особливу увагу приділено об'єктно-орієнтованому моделюванню: класи користувачів, тестів, запитань, сесій і результатів спроектовано згідно з принципами інкапсуляції та розширюваності. Усі сутності реалізовані як окремі класи з чітко визначеними властивостями та логічними зв'язками, що дозволило ефективно організувати зберігання в документоорієнтованій базі MongoDB.

Розробка бази даних ґрунтується на логічному поділі інформації за колекціями, кожна з яких відповідає певній бізнес-сутності. Завдяки використанню MongoDB API у поєднанні з чітко спроектованими моделями, система забезпечує як гнучкість у зміні схеми, так і швидкість при виконанні запитів, що особливо важливо в умовах масштабування.

Важливим аспектом розділу стало описання алгоритмів взаємодії між модулями: створення тесту, проходження, оцінювання результату. Усі етапи реалізовані з урахуванням принципів асинхронності, захисту даних та безперервного збереження, що дозволяє гарантувати цілісність і збереження прогресу навіть при нестабільному підключенні.

Крім того, у проєкті реалізовано два ключові архітектурні патерни — Репозиторій і Сервісний шар. Вони дозволяють ізолювати логіку зберігання та обробки, забезпечити повторне використання коду та значно полегшити юніт-тестування компонентів. Розмежування логіки між інтерфейсами, сервісами та сховищем даних дозволяє розробникам легко вносити зміни на будь-якому рівні без впливу на решту системи.

Таким чином, реалізована структура платформи забезпечує гнучкість, стабільність і високу масштабованість. Використання сучасних технологій .NET 8, MongoDB та Azure, разом з добре спроектованими модулями, дозволяє вважати платформу «Aegos Exam» готовою до розгортання в реальному середовищі з великим навантаженням і широким колом користувачів.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПЛАТФОРМОЮ ДЛЯ
ОНЛАЙН ТЕСТУВАНЬ

3.1. Порядок встановлення та налаштування параметрів системи

Налаштування та запуск платформи для онлайн тестувань вимагають попередньої підготовки кількох ключових компонентів: клієнтського інтерфейсу, веб-сервера, бази даних та зовнішніх інтеграцій. Кожен із цих елементів відіграє свою роль у забезпеченні цілісної та стабільної роботи платформи, тому їх розгортання та налаштування мають бути узгодженими.

Щоб краще зрозуміти архітектуру розгортання, варто розглянути відповідну діаграму розгортання (рис.3.1). Вона ілюструє логічне та фізичне розміщення компонентів, а також спосіб їхньої взаємодії в межах єдиної системи.

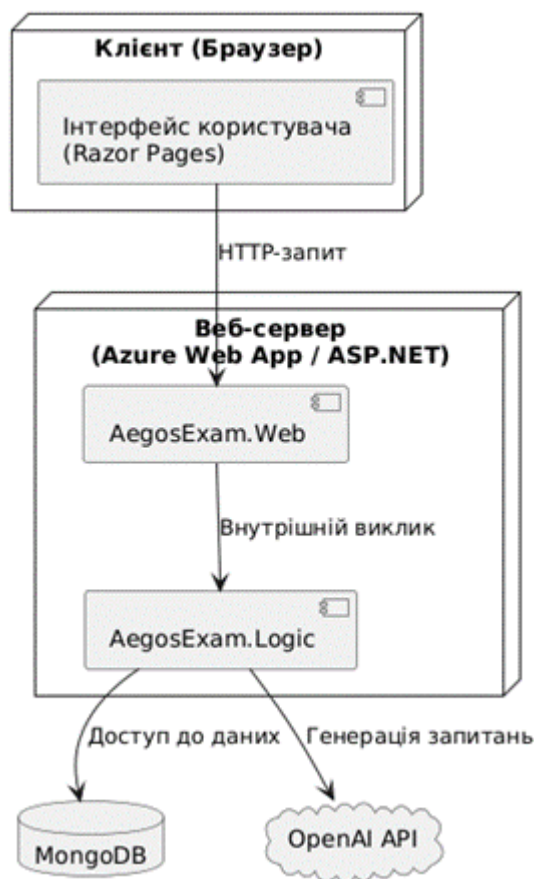


Рис. 3.1 – Діаграма розгортання

Згідно з діаграмою, користувач взаємодіє із застосунком через веб-браузер, де відображається інтерфейс, реалізований за допомогою технології Razor Pages. Усі дії (перегляд, створення або проходження тестів) надсилаються на веб-сервер у вигляді HTTP-запитів. На сервері розгорнуто два основні модулі:

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

AegosExam.Web, який обробляє маршрути й відповіді, та AegosExam.Logic, де зосереджена бізнес-логіка системи.

У процесі обробки запитів модуль AegosExam.Logic взаємодіє з базою даних MongoDB, яка зберігає усі сутності: користувачів, тести, сесії, відповіді тощо. Крім того, реалізовано інтеграцію з хмарним сервісом OpenAI API, що дозволяє автоматично генерувати запитання на основі наданого промпту. Запити до зовнішнього API ініціюються лише за потреби.

Подібна структура дає змогу масштабувати систему, розділяючи навантаження між клієнтом, сервером та сховищем. Наприклад, базу можна розгорнути як керований кластер у MongoDB Atlas, а веб-сервер — як окремий екземпляр Azure Web App. Такий підхід забезпечує гнучкість, стабільність та можливість інтеграції нових сервісів у майбутньому.

3.2. Структура інтерфейсу платформи для онлайн тестувань

При відкритті веб-сайту, користувач потрапляє на сторінку авторизації, що зображена на рис. 3.2. На платформі авторизація є обов'язковою.

Рис. 3.2 – Сторінка авторизації

Користувач переходить на сторінку створення акаунту, що зображена на рис. 3.4, в нього є дві опції: створення студентського або ж організаторського акаунту (рис. 3.5). В залежності від вибраного варіанту змінюється набір полів для заповнення.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

Андрій

Прізвище *

Сахно

Електронна пошта *

Введіть електронну пошту

Електронна пошта є обов'язковою.

Організація *

Житомирська політехніка

Посада *

Організатор

Пароль *

.....

Повторіть пароль *

.....

Паролі не співпадають.

Зареєструватися

Рис. 3.6 – Валідація полів форми

Після успішного заповнення усіх полів користувача перенаправляють на сторінку входу в обліковий запис, де він може ввести дані входу до свого новоствореного облікового запису (рис. 3.7).

Вхід до облікового запису

Електронна пошта *

andrii.org@gmail.com

Пароль *

.....

Не маєте акаунту? [Перейти до реєстрації.](#)

Увійти

Рис. 3.7 – Форма входу з даними входу

При успішній авторизації користувача перенаправляють на головну сторінку (рис. 3.8), де відображається привітання з його іменем та текстбокс для вводу коду теста для приєднання. Також у правому верхньому куті відображається іконка організатора та ім'я.

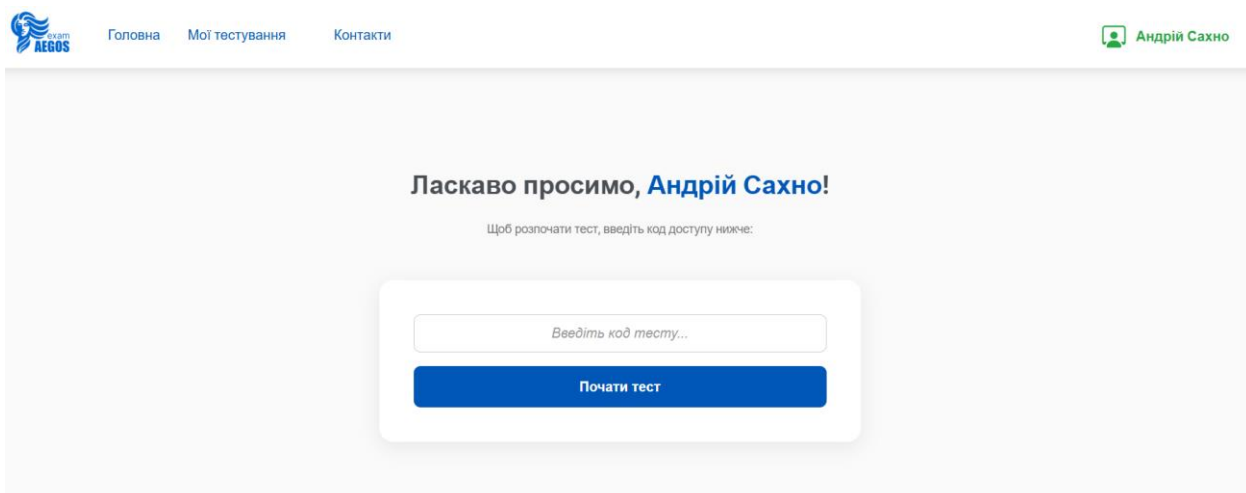


Рис. 3.8 – Головна сторінка платформи

Але перед початком проходження тест потрібно створити, тому користувач переходить на сторінку «Мої тестування» (рис. 3.9), де поки немає ні одного теста.

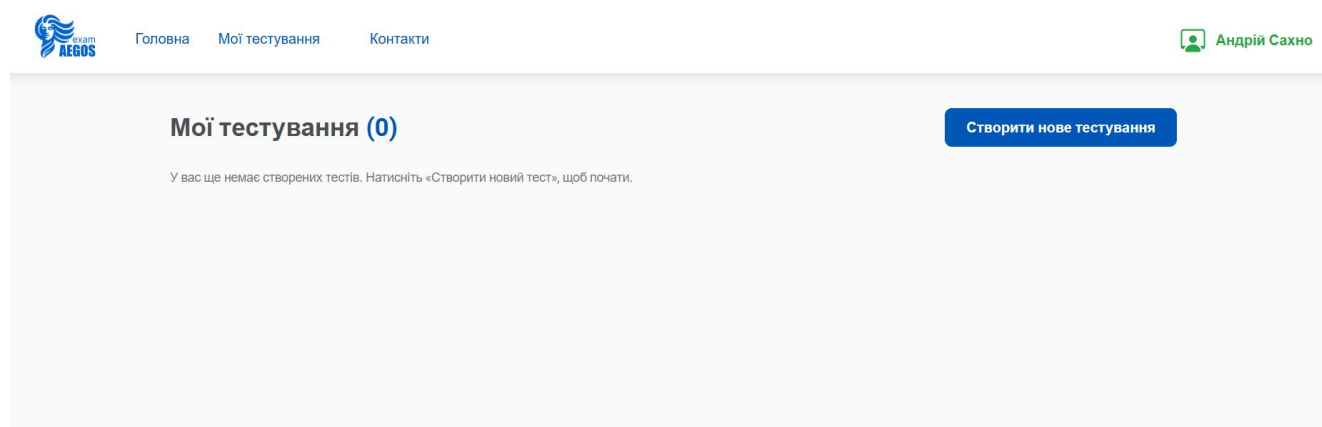


Рис. 3.9 – Сторінка мої тестування

Користувач натискає «Створити нове тестування», система перенаправляє його на сторінку створення, де він може задати назву, опис та задати налаштування, натискаючи відповідні візуальні компоненти (рис. 3.10, рис. 3.11). Доступні налаштування «Увімкнути обмеження часу», «Обмежити кількість спроб», «Відобразити правильні відповіді після завершення», «Перемішати порядок запитань». При натисканні на компоненти пов'язані з обмеженням часу або спроб, є можливість задати числове значення обмеження. Усі поля мають валідацію, у випадку невведення якогось з полів висвічується відповідне повідомлення.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				56
Змн.	Арк.	№ докум.	Підпис	Дата		

Мої тестування > Створення тестування

Створення нового тестування

Назва тестування *

Моє перше тестування

Опис

Тестування щоб протестувати =)

Налаштування тестування

☒ Увімкнути обмеження часу
Після завершення часу тест автоматично завершиться.
Тривалість (у хвиликах) *
10
Вкажіть від 1 до 600 хвилин.

Рис. 3.10 – Сторінка створення тестування (Назва і опис)

☒ Увімкнути обмеження часу
Після завершення часу тест автоматично завершиться.
Тривалість (у хвиликах) *
10
Вкажіть від 1 до 600 хвилин.

☒ Обмежити кількість спроб
Користувач зможе пройти тест лише задану кількість разів
Кількість дозволених спроб *
3
Вкажіть число від 1 до 20. Після вичерпання спроб повторне проходження буде недоступне.

☒ Відобразити правильні відповіді після завершення
Після завершення тесту користувач побачить правильні відповіді.

☒ Перемішати порядок запитань
Кожен користувач отримає випадковий порядок запитань

Створити тестування

Рис. 3.11 – Сторінка створення тестування (Налаштування)

Після натискання кнопки «Створити тестування, система перенаправляє користувача на сторінку з інформацією про нього (рис. 3.12), можливістю додавання питань різного типу (рис. 3.13), їх редагування та видалення.

Мої тестування > Тестування "Моє перше тестування"

Моє перше тестування

Імпорт/Експорт Excel

AI генерація

Редагувати

Опублікувати

Створено:
22:09:44 17/06/2025

Код доступу:
SCZ9CV

Опис:
Тестування щоб протестувати =)

Кількість запитань:
0

Обмеження часу:
✓ Увімкнено (10 хв)

Обмеження спроб:
✓ Увімкнено (3 спроби)

Перемішування питань:
✓ Увімкнено

Показ відповідей:
✓ Увімкнено

Поточний стан:
Чернетка

Остання зміна:
22:09:44 17/06/2025

Додати нове запитання

Текст запитання *

Наприклад: Яка столиця Франції?

Рис. 3.12 – Інформація про тестування

Додати нове запитання

Текст запитання *

Наприклад: Яка столиця Франції?

Тип запитання *

Один варіант
Користувач повинен вибрати лише одну правильну відповідь із запропонованих

Кілька варіантів
Користувач може вибрати одну або кілька правильних відповідей із списку

Відкрите
Користувач вводить відповідь самостійно у вигляді тексту

Варіанти відповіді (макс. 5 варіантів): *

☐
Варіант відповіді 1

Додати варіант

- Текст запитання має містити від 5 до 300 символів
- Додайте хоча б один варіант відповіді
- Позначте хоча б одну правильну відповідь

Додати запитання

Існуючі запитання (0)

Вибрати всі

Очистити вибір

Видалити вибрані

Рис. 3.13 – Додавання та редагування питань тесту

Усього налічується 3 види питань: з однією, кількома та відкритою відповіддю. Щоби додати питання (рис. 3.14), потрібно виконати усі пункти, що знаходяться біля кнопки додати запитання. Для питань з вибором потрібно

задати текст, хоча б один варіант відповіді, хоча б одну правильну. Для відкритої – текст і хоча б один правильний варіант. Максимальне обмеження для усіх питань – 5 варіантів. Після виконання усіх пунктів – питання додається до тесту без перезавантажень.

Додати нове запитання

Текст запитання *

2 + 2 =

Тип запитання *



Один варіант

Користувач повинен вибрати лише одну правильну відповідь із запропонованих



Кілька варіантів

Користувач може вибрати одну або кілька правильних відповідей із списку



Відкрите

Користувач вводить відповідь самостійно у вигляді тексту

Варіанти відповіді (макс. 5 варіантів): *



8



3



4



5



Додати варіант

• Текст запитання має містити від 5 до 300 символів

• Додайте хоча б один варіант відповіді

• Позначте хоча б одну правильну відповідь

Додати запитання

Рис. 3.14 – Редагування питання

Користувач може редагувати (рис. 3.15) та видаляти питання абсолютно без перезавантажень. Є можливість вибору та видалення питання серією.

Існуючі запитання (1)

Вибрати всі

Очистити вибір

Видалити вибрані

Тип: Одна відповідь

Текст запитання *

2 + 2 =

Варіанти відповіді: *

8



3



4



5



Додати варіант

• Текст запитання має містити від 5 до 300 символів

• Додайте хоча б один варіант відповіді

• Позначте хоча б одну правильну відповідь

Зберегти

Скасувати

Рис. 3.15 – Редагування питання

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

Для пришвидшення процесу наповнення тесту, користувач має досить сучасну та унікальну функцію «AI генерація». При переході на сторінку генерації є детальна інструкція для генерації питань у тест (рис. 3.16), а також форма з полями для заповнення (рис. 3.17). Організатор повинен задати тему тесту, цільову аудиторію, мову запитань, ключові слова, тип запитань (усі можливі типи на платформі), а також кількість запитань (до 20). Усі поля відповідно підтримують валідацію.

Мої тестування > Тестування "Моє перше тестування" > Генерація запитань за допомогою AI

Генерація запитань за допомогою AI

Приклад заповнення форми

Щоб отримати якісні запитання, рекомендується чітко сформулювати тему, аудиторію та ключові слова. Нижче наведено приклад:

- **Тема тесту:** Основи інформаційної безпеки
- **Цільова аудиторія:** Студенти 2 курсу спеціальності Кібербезпека
- **Мова запитань:** Українська
- **Ключові слова:** фішинг, антивіруси, шифрування, соціальна інженерія
- **Тип запитань:** Декілька правильних відповідей
- **Кількість:** 10

Після натискання кнопки «Згенерувати», запитання автоматично додадуться до тесту. Їх можна буде переглянути та відредагувати.

⚠ Зверніть увагу: AI не гарантує 100% правильність фактів або відповідей. Ретельно перевіряйте створені запитання перед публікацією!

Рис. 3.16 – Інструкція для генерації питань

Параметри генерації запитань

Тема тесту *

Кібербезпека

Цільова аудиторія *

Інформаційні технології, 1 курс

Мова запитань *

Українська

Ключові слова *

Кібербезпека, інформаційна безпека, захист даних, доступність

Тип запитань *

Одна правильна відповідь

Кількість запитань (до 20) *

8

Згенерувати (gpt-35-turbo)

Рис. 3.17 – Форма параметрів генерації запитань

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

При натисканні кнопки «Згенерувати», почнеться процес генерації. Це займе деякий час, тому буде відображатись спінер, таймер та прохання не оновлювати сторінку (рис. 3.18). А після повідомлення про успішну генерацію питань, та скільки це зайняло (рис. 3.19).

Тип запитань *

Одна правильна відповідь

Кількість запитань (до 20) *

8

Згенерувати

(gpt-35-turbo)

00:03

Будь ласка, не оновлюйте сторінку під час генерації запитань.

Рис. 3.18 – Процес генерації запитань

Мої тестування > Тестування "Моє перше тестування" > Генерація запитань за допомогою AI

Генерація запитань за допомогою AI

✓

Згенеровано 8 запитання та успішно додано до тесту. Це зайняло 00:09

Приклад заповнення форми

Щоб отримати якісні запитання, рекомендується чітко сформулювати тему, аудиторію та ключові слова. Нижче наведено приклад:

•Тема тесту:

Основи інформаційної безпеки

•Цільова аудиторія:

Студенти 2 курсу спеціальності Кібербезпека

Рис. 3.19 – Повідомлення про успішне генерування

При переході до списку питань, користувач може помітити, що запитання на задану тему згенерувались (рис. 3.20).

Існуючі запитання (9)

Вибрати всі

Очистити вибір

Видалити вибрані

Які переваги має використання двофакторної аутентифікації?

Тип: Одна відповідь

Зменшення безпеки облікового запису

Додатковий шар захисту від несанкціонованого доступу ✓

Збільшення кількості паролів для запам'ятовування

Неможливість входу в обліковий запис

Редагувати

Видалити

Чому важливо мати оновлене програмне забезпечення?

Тип: Одна відповідь

Для зменшення швидкості роботи комп'ютера

Для забезпечення безпеки від нових загроз ✓

Для накопичення вірусів

Для витрати більшої кількості пам'яті

Редагувати

Видалити

Що таке бекапування даних?

Тип: Одна відповідь

Видалення даних

Створення резервних копій даних для запобігання втраті ✓

Публічне розміщення даних

Перенесення даних на зовнішній пристрій

Редагувати

Видалити

Рис. 3.20 – Розширений список запитань після успішної генерації

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

Після чого користувач може перейти до секції «Імпорт/Експорт Excel». Ця сторінка має широкі можливості імпорту і експорту питань з використанням Microsoft Excel. Угорі відображається дві карточки з функціоналом та порадами (рис. 3.21), у нижній частині є обширна карточка з детальними інструкціями користування (рис. 3.22).

Імпорт запитань

Потрібен **порожній шаблон** для заповнення? [Кликайте для завантаження \(.xlsx\)](#)

Для редагування шаблону рекомендуємо використовувати:

- **Microsoft Excel** (версія 2010 або новіша)
- **Google Таблиці** (через імпорт XLSX-файлу)
- **LibreOffice Calc** (збереження у форматі .xlsx без макросів)

У тесті "**Моє перше тестування**" вже є **9** запитань.
Ви можете додати ще **491** (загальний максимум — **500**).

Режим імпорту *

Додати усі запитання з файлу

Файл Excel (.xlsx) *

Choose File
No file chosen

Оберіть файл у форматі .xlsx. Не використовуйте формули або об'єднані клітинки.

Імпортувати запитання з файлу

Експорт запитань

Ви можете завантажити Excel-файл з усіма запитаннями цього тесту.

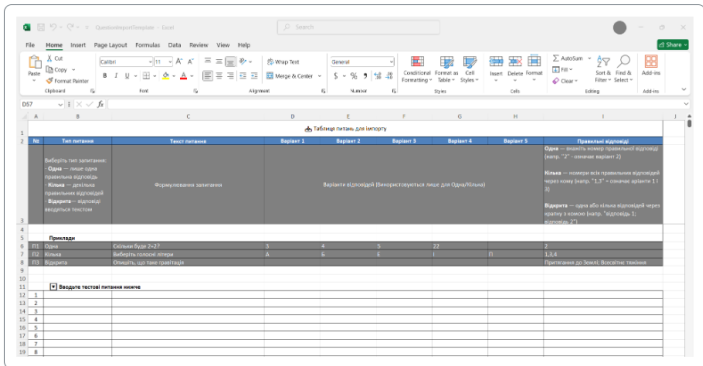
Файл можна використовувати для перегляду, резервного копіювання або повторного імпорту.

У тесті "Моє перше тестування" усього **9** запитань.

Експортувати наявні питання у файл

Рис. 3.21 – Функціонал імпорту, експорту питань

Огляд Excel-шаблону



Загальна інформація:

- Шаблон дозволяє максимум **500** запитань — як і сам тест.
- Містить окремий аркуш з інструкціями під назвою "**Інструкції**" — рекомендується прочитати перед використанням.
- Кожен рядок з ID (від 1 до 500) відповідає одному запитанню.

Позначки типів запитань:

- "**Одна**" — одна правильна відповідь. Наприклад: "2".

Рис. 3.22 – Інструкція використання шаблону

Для початку користувач може експортувати наявні питання в цьому тесті, натиснувши відповідну кнопку. Відкривши згенерований файл, на основному аркуші користувач бачить об'ємну таблицю з специфічною структурою (рис. 3.23). В ній перелічені усі, наявні у тесті, питання. Також присутній аркуш детальних інструкцій для використання (рис. 3.24).

№	Тип питання	Текст питання	Варіант 1	Варіант 2	Варіант 3	Варіант 4	Варіант 5	Правильні відповіді
		Виберіть тип запитання: - Одна — лише одна правильна відповідь - Кілька — декілька правильних відповідей - Відкрита — відповіді вводяться текстом						Одна — вкажіть номер правильної відповіді (напр. "2" - означає варіант 2) Кілька — номери всіх правильних відповідей через кому (напр. "1,3" - означає варіанти 1 і 3) Відкрита — одна або кілька відповідей через крапку з комою (напр. "Відповідь 1; відповідь 2")
	Приклади							
П1	Одна	Скільки буде 2+2?	3	4	5	22		2
П2	Кілька	Виберіть голосні літери	А	Б	Є		П	1,3,4
П3	Відкрита	Опишіть, що таке гравітація						Притягання до Землі; Всесвітнє тяжіння
	Вводьте тестові питання нижче							
1	Одна	2 + 2 =	6	3	4	5		3
	Одна	Що таке фішинг?	Відправлення шкідливих вірусів	Атака на інформаційну систему	Шахрайство з метою вибуття конфіденційних даних	Заблокування доступу до даних		3
	Одна	Які методи можна використовувати для захисту від вірусів?	Використання антивірусного програмного забезпечення	Зміна IP-адреси	Відправлення шкідливих посилань	Нічого з переліченого		1
	Одна	Що таке DDoS-атака?	Атака на персональні мережі запитими, що пошкоджують до	Переповнення мережі запитими, що пошкоджують до	Атака на веб-камер	Відправлення спаму		2

Рис. 3.23 – Таблиця експортованих питань

№	Тип питання	Текст питання	Варіант 1	Варіант 2	Варіант 3	Варіант 4	Варіант 5	Правильні відповіді
		Як заповнити шаблон для імпорту запитань						
		1. Почніть вводити запитання нижче рядка "Вводьте свої питання нижче".						
		2. У колонці "Тип" виберіть один із варіантів:						
		- Одна — якщо тільки одна правильна відповідь						
		- Кілька — якщо правильних відповідей кілька						
		- Відкрита — якщо учень сам пише відповідь текстом						
		3. У колонках "Варіант 1–5" вписуйте варіанти відповідей:						
		- Якщо тип "Одна" або "Кілька" — заповніть від 1 до 5 варіантів						
		- Якщо тип "Відкрита" — залиште ці клітинки порожніми						

Рис. 3.24 – Таблиця експортованих питань

При недовгому ознайомленні, користувач може додати деякі питання сюди, для копіювання і розширення, для пришвидшення процесу створення тестування, для створення резервної копії тесту перед редагуванням. Після редагування, користувач може підгрузити файл у відповідний компонент і обрати тип імпорту в залежності від його потреби (рис. 3.25). При успішному імпорті, висвітиться відповідне повідомлення про успіх. Якщо виникнуть якісь проблеми з синтаксисом або ж переповненням тесту, система пропустить невалідні питання та виведе їх ідентифікатори для подальшого редагування користувачем.

Режим імпорту *

Перезаписати — очистити існуючі і додати запитання з ▾

Файл Excel (.xlsx) *

Choose File ExportedQ...ns (15).xlsx

Оберіть файл у форматі .xlsx . Не використовуйте формули або об'єднані клітинки.

Імпортувати запитання з файлу

Рис. 3.25 – Заповнена форма імпорту

Після додавання двох питань у файл, результат імпорту можна бачити на рис. 3.26.

Існуючі запитання (11) Вибрати всі Очистити вибір Видалити вибрані

Які переваги має використання двофакторної аутентифікації? Тип: Одна відповідь

☐ Зменшення безпеки облікового запису
Додатковий шар захисту від несанкціонованого доступу ✓
Збільшення кількості паролів для запам'ятовування
Неможливість входу в обліковий запис

Редагувати
Видалити

Чому важливо мати оновлене програмне забезпечення? Тип: Одна відповідь

☐ Для зменшення швидкості роботи комп'ютера
Для забезпечення безпеки від нових загроз ✓
Для накопичення вірусів
Для витрати більшої кількості пам'яті

Редагувати
Видалити

Які переваги має використання двофакторної аутентифікації? Тип: Одна відповідь

☐ Зменшення безпеки облікового запису
Додатковий шар захисту від несанкціонованого доступу ✓
Збільшення кількості паролів для запам'ятовування
Неможливість входу в обліковий запис

Редагувати
Видалити

Рис. 3.26 – Результат імпорту

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				64
Змн.	Арк.	№ докум.	Підпис	Дата		

Після наповнення тесту – його можна публікувати та робити доступним для проходження користувачам. Це організатор може зробити з допомогою кнопки «Опублікувати». Система його перенаправить на сторінку сесії тестування (рис. 3.27).

Рис. 3.27 – Сторінка керування сесією тестування

Користувачі при вводі цього коду (рис. 3.28), отримують доступ до тестування. Система перенаправляє їх на сторінку інформації та рекомендацій про проходження (рис. 3.29).

Рис. 3.28 – Вписування коду тестування

Головна · Інформація про тестування

Інформація про проходження тестування "Моє перше тестування"

Тестування щоб протестувати =>

Кількість запитань:
11

Обмеження часу:
✓ Увімкнено (10 хв)

Обмеження спроб:
✓ Увімкнено (3 спроби)

Виконано спроб:
0

▶ Спроби проходження користувача **Andre Brown** (0)

Інструкція з проходження тесту

- Загальні положення**
Цей тест призначений для оцінювання ваших знань з відповідної теми. Він містить **11** запитань різних типів: з однією правильною відповіддю, кількома правильними відповідями або відкриті запитання.
- Обмеження часу**
На виконання тесту відведено: **10 хв.**
Після завершення часу тест буде автоматично завершено.

Рис. 3.29 – Сторінка інформації проходження

При натисканні «Почати тестування», користувач почне сесію і запустить таймер. Система переадресує його на сторінку проходження тесту (рис. 3.30), яка включатиме в себе блок питання, функціонал позначення питання прапорцем, очищення відповіді, пагінації та таймеру.

Запитання 1 із 11
Тип: Одна відповідь

Залишилось часу: **8 хв. 12 с.**

Які методи можна використовувати для захисту від вірусів?

Позначити

Очистити

☐ Використання антивірусного програмного забезпечення
 ☐ Зміна IP-адреси
 ☐ Відправлення шкідливих посилань
 ☐ Нічого з переліченого

Назад

1 2 3 4 5 6 7 8 9 10 11

Далі

Відповіли на 0 із 11 запитань

Рис. 3.30 – Сторінка проходження тестування

На якомусь етапі проходження, користувач бачитиме видозмінену панель пагінації (рис. 3.31), в залежності від того, які питання він прочитав, помітив і на які відповів. Усі дані про проходження зберігаються в базі даних в процесі, тому

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				66
Змн.	Арк.	№ докум.	Підпис	Дата		

у випадку втрати з'єднання з інтернетом або зміни пристрою користувач зможе відновити сесію і продовжити з місця де почав (рис. 3.32).

Запитання 9 із 11 Тип: Одна відповідь

Залишилось часу: 6 хв. 17 с.

Що таке шифрування даних?

Позначити Очистити

☐ Перетворення даних у нечитабельний формат для забезпечення конфіденційності

☐ Відправлення даних без захисту

☐ Видалення даних з пристрою

☐ Публічне розміщення даних

Назад 1 2 3 4 5 6 7 8 9 10 11 Далі

Відповіли на 7 із 11 запитань

Рис. 3.31 – Видозмінена пагінація в процесі проходження

4. Завершення тесту

Після проходження останнього запитання натисніть **“Завершити тест”**

5. Рекомендації

- Забезпечте стабільне інтернет-з'єднання.
- Не відкривайте тест в кількох вкладках одночасно.

Продовжити тестування

Залишилось: 4 хв. 17 с.

Рис. 3.32 – Відновлення сесії

При натисканні «Завершити» користувача перенаправить на фінальну сторінку перед завершенням тесту (рис. 3.33), де він може переглянути усі дані в процесі свого проходження. Ті питання, на які не відповів, які позначив, як відповів, тощо. У ситуації вичерпання часу (рис. 3.34), система автоматично заблокує можливість додавати відповіді та перенаправить на сторінку результатів (рис. 3.35).

Проходження тестування > Перевірка перед завершенням

Перевірка перед завершенням тестування "Моє перше тестування"

Ви не відповіли на 4 запитань!
За ці запитання ви отримаєте 0 балів, якщо завершите тестування зараз.
Ви можете повернутись і дати відповіді перед завершенням.

✗ Без відповіді (4)

Запитання 3 Тип: Одна відповідь Без відповіді

Перейти

Що таке DDoS-атака?

Запитання 6 Тип: Одна відповідь Без відповіді

Перейти

2 + 2 =

0 хв. 33 с.



Рис. 3.33 – Сторінка перевірки перед завершенням тестування

Зменшення безпеки облікового запису

Запитання 11 Тип: Одна відповідь Відповідь є

Перейти

Які переваги має використання двофакторної аутентифікації?

Обрано варіанти:

Збільшення кількості паролів для запам'ятовування

Після натискання кнопки тест буде завершено, і ви не зможете змінити відповіді.

Завершити тестування

Залишилось часу: ⌚ Час вичерпано!

Рис. 3.34 – Вичерпання часу

Головна > Інформація про тестування > Проходження тестування > Результат тестування

Результати тестування "Моє перше тестування"

Результат користувача
Andre Brown

18.2%

Тест завершено

В інших шкалах оцінювання

12-бальна система: 0
5-бальна (незадовільно) система: 1
Letter grade: F

Перевірка результату

Скануйте цей QR-код, щоб перевірити дійсність результатів користувача Andre Brown.



Початок:
23:24:44 17/06/2025

Завершення:
23:34:47 17/06/2025

Тривалість:
10 хв 2 сек

Кількість запитань:
11

Правильних відповідей:
2

Частково правильних відповідей:
0

Неправильних відповідей:
9

Середній час на запитання:
54 сек

Список використаних матеріалів Andre Brown (4)

Рис. 3.35 – Сторінка результату

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

Сторінка результату відображає максимально детально усю інформацію про проходження, відображає усі надані відповіді із зазначенням правильності. Якщо організатор дозволяє – є можливість переглянути правильні відповіді.

Також користувач має доступ до особистого кабінету (рис. 3.36), де може переглядати і редагувати інформацію про себе, змінювати пароль, або ж взагалі видалити свій акаунт.

Скріншот сторінки профілю користувача в системі AEOSS. Вгорі є навігаційне меню: Головна, Мої тестування, Контакти. Праворуч вказано ім'я користувача: Андрій Сахно. Основна частина сторінки має заголовок 'Профіль користувача' і містить наступні дані:

Ім'я Андрій	Прізвище Сахно	Email andrii.org@gmail.com	Роль TestOrganizer	Статус акаунта Активний
Дата створення 17.06.2025 21:34	Останнє оновлення 17.06.2025 21:34	Організація Житомирська політехніка	Посада Організатор	

Нижче є секція 'Оновити особисту інформацію' з формами для введення:

- Ім'я * (введено: Андрій)
- Прізвище * (введено: Сахно)
- Організація (введено: Житомирська політехніка)

Рис. 3.36 – Особистий профіль користувача

Платформа також включає в себе адмінську панель (рис. 3.37) з усім потрібним функціоналом адміністрування користувачів (рис. 3.38) та тестувань (рис. 3.39).

Панель адміністратора

Скріншот панелі адміністратора. Вгорі є статистика:

Кількість користувачів 7	Адміністратори 2	Організатори тестів 3	Звичайні користувачі 2	Заблоковані акаунти 0
Нові реєстрації (7 днів) 2	Усього тестів 1	Опубліковані тести 1		

Нижче є секція 'Останні події' з наступними записами:

- Опубліковано тест: Моє перше тестування 23:10:05 17/06
- Створено новий тест: Моє перше тестування 22:09:44 17/06
- Новий користувач: Андрій Сахно 21:34:27 17/06
- Новий користувач: Андрій Сахно 21:29:48 17/06

Внизу є дві кнопки: Керування користувачами, Керування тестами.

Рис. 3.37 – Панель адміністратора

[Головна](#) > [Панель адміністратора](#) > Керування користувачами

Керування користувачами

Показати записів

Пошук:

ПІБ	Email	Роль	Статус	Організація	Посада	Створено	Дії
Andre Brown	1234@email.com	Admin	Активний	—	—	10.06.2025 15:39	Переглянути Заблокувати
ghij jgfhkjhg	1jkfgjkkkyuky234@email.com	TestOrganizer	Активний	—	—	10.06.2025 15:39	Переглянути Заблокувати
Kateryna Petruk	katyaloveandriusha@gmail.com	RegularUser	Активний	—	—	10.06.2025 15:39	Переглянути Заблокувати
Test Organizer	organizer@email.com	TestOrganizer	Активний	56345	7657456	10.06.2025 15:39	Переглянути Заблокувати
Андрій Сахно	andrii.sakhno@gmail.com	RegularUser	Активний	—	—	17.06.2025 21:29	Переглянути Заблокувати

Рис. 3.38 – Керування користувачами

[Головна](#) > [Панель адміністратора](#) > Керування тестами

Керування тестами

Показати записів

Пошук:

Назва	Автор	Код доступу	Питань	Статус	Спроби	Час	Створено	Змінено	Дії
Моє перше тестування	Андрій Сахно	SCZ9CV	11	Published	3 спроб(и)	10 хв	17.06.2025 22:09	17.06.2025 23:10	Переглянути Редагувати Видалити

Показано 1–1 з 1 записів

Початок Кінець

Рис. 3.39 – Керування тестуваннями

3.3. Тестування роботи програмного комплексу

У процесі тестування системи «Aegos Exam» основну увагу було зосереджено на перевірці ключового функціоналу: створення та проходження тестів, перевірки результатів, а також роботи інтерфейсу користувача в різних сценаріях. Було протестовано, як система реагує на типові дії користувача: відкриття сторінки з тестами, додавання питань, початок сесії та збереження

відповідей. Крім того, були перевірені граничні ситуації, зокрема спроби пройти тест із недійсним кодом доступу, або отримати доступ до сторінки без прав.

Для забезпечення глибокого охоплення тестування застосовувались різні підходи, серед яких метод аналізу граничних значень і метод "причина-наслідок". Це дало змогу змодельовати типові ситуації, в яких користувач виконує тест за умовами з обмеженням часу, із великою кількістю питань або з відкритими відповідями. Система мала забезпечити стабільну роботу при будь-яких комбінаціях параметрів, коректно обробляючи кожну дію користувача.

Окремо використовувався підхід "передбачення помилок", що дозволив виявити ситуації, у яких користувач міг залишити обов'язкові поля порожніми або ввести некоректні дані. Було перевірено, чи система надає валідаційні повідомлення, чи блокує збереження невалідної інформації та чи зберігає при цьому поточний стан інтерфейсу без втрати даних. Це особливо важливо для підвищення юзабіліті застосунку, адже дозволяє уникнути розчарування користувача в разі помилки.

У результаті тестування було виявлено та усунуто низку дрібних помилок, пов'язаних з перевіркою обов'язкових полів, переходами між сторінками та обробкою виняткових ситуацій (наприклад, закінчення часу під час проходження тесту). Завдяки цьому вдалося значно підвищити стабільність роботи платформи та поліпшити взаємодію з інтерфейсом. Застосунок успішно пройшов усі основні сценарії, що підтверджує його готовність до використання у реальному середовищі.

Нижче подано основні етапи проведеного тестування.

У межах функціонального тестування перевірялась робота основних сценаріїв користувача. Зокрема, було протестовано реєстрацію та авторизацію користувачів, включно з обробкою коректних та некоректних даних під час введення. Окремо перевірялись обмеження доступу до функцій залежно від ролі: звичайний користувач, організатор або адміністратор. Важливою частиною функціонального тестування було проходження тестів: перевірка збереження відповідей, навігації між запитаннями, обрахунку результату, а також

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

блокування можливості повторного проходження, якщо це заборонено налаштуваннями. Сценарії з обмеженням часу також були протестовані, включно з авто-завершенням сесії після тайм-ауту.

Оцінювалась зручність та інтуїтивність інтерфейсу для різних типів користувачів. Перевірялась коректність відображення елементів управління, зрозумілість підказок, логіка розташування кнопок і загальний вигляд сторінок. Тестування проводилось як на десктопах, так і на мобільних пристроях, щоб переконатися у адаптивності дизайну. Було виявлено кілька недоліків у мобільному поданні форми з відповідями, які були оперативно усунені. Додатково перевірялися сценарії редагування тесту організатором, навігація по вкладках у режимі результатів та перегляд підсумкових балів.

Особлива увага приділялась безпеці облікових записів та захисту персональних даних. Тестувались механізми хешування паролів, обробка токенів сесій, а також обмеження доступу до API-методів відповідно до ролі користувача. Було перевірено стійкість системи до XSS-атак (введення небезпечного HTML у форму). Оскільки база даних реалізована на основі MongoDB, також було проведено перевірку на Mongo-ін'єкції. Усі критичні загрози було усунуто, а для запитів впроваджено захищену обробку параметрів.

Загалом, тестування охопило функціональні, візуальні та захисні аспекти системи, що дозволило виявити та усунути значну кількість потенційних проблем ще до моменту запуску платформи у реальному середовищі.

Висновки до третього розділу

У третьому розділі було детально розглянуто архітектурну побудову, інтерфейсну частину та результати тестування платформи «Aegos Exam». На основі діаграми розгортання описано ключові компоненти системи та принципи їх взаємодії — веб-інтерфейс на Razor Pages, бізнес-логіка, MongoDB як основа зберігання даних, а також інтеграція з OpenAI для автоматизованого створення тестових завдань. Така архітектура забезпечує модульність, масштабованість і гнучкість під час подальшого розвитку платформи.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

Значну увагу приділено користувацькому інтерфейсу: покроковий процес створення, редагування та проходження тесту супроводжується валідацією, збереженням стану та інформативними повідомленнями. Інтерфейс враховує потреби трьох основних ролей (RegularUser, TestOrganizer, Admin), а також адаптований до різних пристроїв. Особливо цінними є можливості генерації запитань за допомогою GPT-35-Turbo та зручна система імпорту/експорту у форматі Excel, що значно спрощує роботу організаторів.

Під час тестування системи було перевірено всі основні функції: від реєстрації до завершення тестування. Протестовано поведінку у випадках обмеженого доступу, некоректних дій, завершення часу, а також перевірено візуальні та адаптивні аспекти інтерфейсу. Виявлені помилки було оперативно усунуто, що дозволило досягти високого рівня стабільності й зручності використання. Особлива увага приділялась безпеці: реалізовано хешування паролів, обмеження доступу за ролями та захист від поширених атак.

Таким чином, третій розділ демонструє, що реалізована система відповідає сучасним вимогам до онлайн-платформ: вона технічно надійна, зручна у використанні, добре протестована та готова до повноцінного застосування у навчальному або професійному середовищі.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було спроектовано та реалізовано платформу для онлайн-тестування «Aegos Exam», що поєднує сучасні технології, зручний інтерфейс і гнучку архітектуру. Система орієнтована на три ролі користувачів — учасників, організаторів та адміністраторів — і забезпечує повний цикл проведення тестування: від створення до оцінювання результатів.

У першому розділі досліджено існуючі рішення, визначено недоліки аналогів та сформульовано технічні вимоги. Обґрунтовано вибір Razor Pages для реалізації інтерфейсу, MongoDB для зберігання даних і Azure OpenAI для автоматичної генерації запитань. Обрані технології забезпечили гнучкість, масштабованість та швидкодію системи.

Другий розділ присвячено архітектурі платформи. Детально описано структуру модулів, об'єктно-орієнтоване моделювання та реалізацію репозиторіїв і сервісів. Чітке розмежування логіки дозволило спростити розробку, тестування та подальший супровід системи.

У третьому розділі проаналізовано інтерфейсну частину та проведено тестування. Перевірено ключові функції: реєстрацію, створення та проходження тестів, збереження результатів і безперервність сесії. Виявлені недоліки усунуті, протестовано безпеку, адаптивність і рольовий доступ. Система продемонструвала стабільність і готовність до використання.

У підсумку, платформа «Aegos Exam» є сучасним інструментом для проведення онлайн-тестування, що поєднує простоту використання, багатий функціонал і технічну надійність. Вона готова до впровадження у навчальні та професійні процеси, де потрібна ефективна система контролю знань.

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Тести - Онлайн-школа «На Урок». *Онлайн-школа «На Урок» для учнів.* URL: <https://naurok.ua/student/tests> (дата звернення: 17.06.2025).
2. Google Forms. URL: <https://docs.google.com/forms/u/0/> (date of access: 17.06.2025).
3. Classtime. URL: <https://www.classtime.com/uk> (date of access: 17.06.2025).
4. Сучасні методи контролю і оцінки знання учнів :: Mach mit! Mach besser!. URL: <https://denglisch-vid-desyatnik.webnode.com.ua/news/suchasni-metodi-kontrolyu-i-otsinki-znannya-uchniv/> (дата звернення: 17.06.2025).
5. Architectural principles - .NET. *Microsoft Learn: Build skills that open doors in your career.* URL: <https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/architectural-principles> (date of access: 17.06.2025).
6. IBM. What Is Three-Tier Architecture? | IBM. *IBM - United States.* URL: <https://www.ibm.com/think/topics/three-tier-architecture> (date of access: 17.06.2025).
7. Azure documentation. *Microsoft Learn: Build skills that open doors in your career.* URL: <https://learn.microsoft.com/en-us/azure/?product=popular> (date of access: 17.06.2025).
8. Introduction to Razor Pages in ASP.NET Core. *Microsoft Learn: Build skills that open doors in your career.* URL: <https://learn.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-9.0&tabs=visual-studio> (date of access: 17.06.2025).
9. ASP.NET documentation. *Microsoft Learn: Build skills that open doors in your career.* URL: <https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-9.0> (date of access: 17.06.2025).
10. .NET documentation. *Microsoft Learn: Build skills that open doors in your career.* URL: <https://learn.microsoft.com/en-us/dotnet/> (date of access: 17.06.2025).

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

11. HTTP request methods. *MDN Web Docs*.
URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods> (date of access: 17.06.2025).
12. Azure Cosmos DB. *Microsoft Learn: Build skills that open doors in your career*. URL: <https://learn.microsoft.com/en-us/azure/cosmos-db/> (date of access: 17.06.2025).
13. Azure Cosmos DB. *Microsoft Learn: Build skills that open doors in your career*. URL: <https://learn.microsoft.com/en-us/azure/cosmos-db/> (date of access: 17.06.2025).
14. Team M. D. MongoDB Documentation - Homepage. *MongoDB: The World's Leading Modern Database* | *MongoDB*.
URL: <https://www.mongodb.com/docs/> (date of access: 17.06.2025).
15. SPA (Single-page application). *MDN Web Docs*.
URL: <https://developer.mozilla.org/en-US/docs/Glossary/SPA> (date of access: 17.06.2025).
16. Quick Start. *React*. URL: <https://react.dev/learn> (date of access: 17.06.2025).
17. Angular. *Home • Angular*. URL: <https://angular.dev/> (date of access: 17.06.2025).
18. ASP.NET MVC Pattern | .NET. *Microsoft*.
URL: <https://dotnet.microsoft.com/en-us/apps/aspnet/mvc> (date of access: 17.06.2025).
19. What is ASP.NET Core?. *Microsoft Learn: Build skills that open doors in your career*. URL: <https://learn.microsoft.com/en-us/shows/back-end-web-development-with-dotnet-for-beginners/what-is-aspnet-core-back-end> (date of access: 17.06.2025).
20. ASP.NET Razor Pages vs MVC: Benefits and Code Comparisons. *Stackify*. URL: <https://stackify.com/asp-net-razor-pages-vs-mvc/> (date of access: 17.06.2025).

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

21. Система управління базами даних: сучасні тенденції у розробці. *FoxmindEd*. URL: <https://foxminded.ua/systema-upravlinnia-bazamy-danykh/> (дата звернення: 17.06.2025).

22. What Is NoSQL? NoSQL Databases Explained. *MongoDB*. URL: <https://www.mongodb.com/resources/basics/databases/nosql-explained> (date of access: 17.06.2025).

23. Firebase Documentation. URL: <https://firebase.google.com/docs> (date of access: 17.06.2025).

24. Amazon DynamoDB Documentation. URL: <https://docs.aws.amazon.com/dynamodb/> (date of access: 17.06.2025).

25. What Is a Key-Value Database?. *Amazon Web Services, Inc.* URL: <https://aws.amazon.com/nosql/key-value/> (date of access: 17.06.2025).

26. Azure OpenAI documentation - Quickstarts, Tutorials, API Reference. *Microsoft Learn: Build skills that open doors in your career*. URL: <https://learn.microsoft.com/en-us/azure/ai-services/openai/> (date of access: 17.06.2025).

27. Azure Key Vault documentation. *Microsoft Learn: Build skills that open doors in your career*. URL: <https://learn.microsoft.com/en-us/azure/key-vault/> (date of access: 17.06.2025).

28. Work with chat completion models - Azure OpenAI in Azure AI Foundry Models. *Microsoft Learn: Build skills that open doors in your career*. URL: <https://learn.microsoft.com/en-us/azure/ai-services/openai/how-to/chatgpt> (date of access: 17.06.2025).

29. Діаграми UML для моделювання процесів і архітектури проекту. *Evergreen - web розробка і діджиталізація бізнесу за допомогою AI продуктів*. URL: <https://evergreens.com.ua/ua/articles/uml-diagrams.html> (дата звернення: 17.06.2025).

30. Role-based authorization in ASP.NET Core. *Microsoft Learn: Build skills that open doors in your career*. URL: <https://learn.microsoft.com/en->

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

[us/aspnet/core/security/authorization/roles?view=aspnetcore-9.0](https://aspnet/core/security/authorization/roles?view=aspnetcore-9.0) (date of access: 17.06.2025).

31. UML Class Diagram Tutorial. *Lucidchart*.
URL: <https://www.lucidchart.com/pages/uml-class-diagram> (date of access: 17.06.2025).

32. Сахно А.О., Вакалюк Т.А. АНАЛІЗ ІСНУЮЧИХ ПЛАТФОРМ ДЛЯ ОНЛАЙН ТЕСТУВАНЬ. Матеріали П'ятнадцятої Міжнародної наукової конференції студентів та молодих вчених «Сучасні інформаційні технології – 2025» «Modern Information Technology – 2025» (15-16 травня 2025 р., м. Одеса) / МОН України; Національний університет «Одеська політехніка»; Ін-т комп'ют. систем. – Одеса : Наука і техніка, 2025. С. 215-217

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

ДОДАТКИ

		Сахно А. О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public class Test
{
    [BsonId]
    [BsonRepresentation(BsonType.ObjectId)]
    public string Id { get; set; } = string.Empty;

    [BsonElement("name")]
    public string Name { get; set; } = string.Empty;

    [BsonElement("description")]
    public string Description { get; set; } = string.Empty;

    [BsonElement("questionIds")]
    public List<string> QuestionIds { get; set; } = new();

    [BsonIgnore]
    public int QuestionCount => QuestionIds.Count;

    [BsonElement("createdBy")]
    public string CreatedBy { get; set; } = string.Empty;

    [BsonElement("authorName")]
    public string AuthorName { get; set; } = string.Empty;

    [BsonElement("accessCode")]
    public string? AccessCode { get; set; }

    [BsonElement("state")]
    public TestState State { get; set; } = TestState.Draft;

    [BsonElement("minuteLimit")]
    public int? MinuteLimit { get; set; }

    [BsonElement("restrictAttempts")]
    public bool RestrictAttempts { get; set; }

    [BsonElement("maxAttempts")]
    public int? MaxAttempts { get; set; }

    [BsonElement("showAnswersAfterSubmit")]
    public bool ShowAnswersAfterSubmit { get; set; }

    [BsonElement("randomizeQuestionOrder")]
    public bool RandomizeQuestionOrder { get; set; }

    [BsonElement("createdAt")]
    [BsonDateTimeOptions(Kind = DateTimeKind.Utc)]
    public DateTime CreatedAt { get; set; } = DateTime.UtcNow;

    [BsonElement("lastModifiedAt")]
    [BsonDateTimeOptions(Kind = DateTimeKind.Utc)]

```

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    public DateTime LastModifiedAt { get; set; } = DateTime.UtcNow;
}

public enum TestState
{
    Draft,
    Published,
    Closed
}

namespace AegosExam.Models
{
    public class Question
    {
        [BsonId]
        [BsonRepresentation(BsonType.ObjectId)]
        public string Id { get; set; } = string.Empty;

        [BsonElement("text")]
        public string Text { get; set; } = string.Empty;

        [BsonElement("type")]
        public QuestionType Type { get; set; } = QuestionType.SingleChoice;

        [BsonElement("options")]
        public List<string> Options { get; set; } = new();

        [BsonElement("correctAnswers")]
        public List<int> CorrectAnswers { get; set; } = new();
    }

    public enum QuestionType
    {
        SingleChoice,
        MultipleChoice,
        OpenEnded
    }
}

namespace AegosExam.Models
{
    public class QuestionPosition
    {
        [BsonElement("positionIndex")]
        public int PositionIndex { get; set; }

        [BsonElement("questionId")]
        public string QuestionId { get; set; } = string.Empty;
    }
}

```

```

namespace AegosExam.Models
{
    public class TestResult
    {
        [BsonId]
        [BsonRepresentation(BsonType.ObjectId)]
        public string Id { get; set; } = string.Empty;

        [BsonElement("userId")]
        public string UserId { get; set; } = string.Empty;

        [BsonElement("testId")]
        public string TestId { get; set; } = string.Empty;

        [BsonElement("testName")]
        public string TestName { get; set; } = string.Empty;

        [BsonElement("startedAt")]
        public DateTime StartedAt { get; set; }

        [BsonElement("finishedAt")]
        public DateTime FinishedAt { get; set; }

        [BsonIgnore]
        public float DurationMilliseconds => (float)(FinishedAt - StartedAt).TotalMilliseconds;

        [BsonElement("userAnswerIds")]
        public List<string> TestResultAnswerIds { get; set; } = new();

        [BsonElement("totalQuestions")]
        public int CountOfQuestions { get; set; }

        [BsonElement("correctAnswersCount")]
        public int CorrectAnswersCount { get; set; }

        [BsonElement("score")]
        public float Score { get; set; }

        [BsonElement("showAnswers")]
        public bool ShowRightAnswers { get; set; }
    }
}

namespace AegosExam.Models
{
    public class TestResultAnswer
    {
        [BsonId]
        [BsonRepresentation(BsonType.ObjectId)]
        public string Id { get; set; } = string.Empty;
    }
}

```

```

[BsonElement("questionId")]
public string QuestionId { get; set; } = string.Empty;

[BsonElement("answerText")]
public string AnswerText { get; set; } = string.Empty;

[BsonElement("selectedOptions")]
public List<int>? SelectedOptions { get; set; }

[BsonElement("isCorrect")]
public bool IsCorrect { get; set; }

[BsonElement("correctness")]
public AnswerCorrectness Correctness { get; set; }

[BsonElement("score")]
public float Score { get; set; }
}

public enum AnswerCorrectness
{
    Incorrect = 0,
    PartiallyCorrect = 1,
    Correct = 2
}

}

namespace AegosExam.Models
{
    public class TestSession
    {
        [BsonId]
        [BsonRepresentation(BsonType.ObjectId)]
        public string Id { get; set; } = string.Empty;

        [BsonElement("testId")]
        public string TestId { get; set; } = string.Empty;

        [BsonElement("userId")]
        public string UserId { get; set; } = string.Empty;

        [BsonElement("startedAt")]
        public DateTime StartedAt { get; set; }

        [BsonElement("minuteLimit")]
        [BsonIgnoreIfDefault]
        public int? MinuteLimit { get; set; }

        [BsonElement("answerIds")]
        [BsonRepresentation(BsonType.ObjectId)]
        public List<string> AnswerIds { get; set; } = new();
    }
}

```

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				83
Змн.	Арк.	№ докум.	Підпис	Дата		

```

[BsonElement("questionMap")]
public List<QuestionPosition> QuestionMap { get; set; } = new();

[BsonIgnore]
public DateTime? EndTime
{
    get {
        if (MinuteLimit.HasValue && MinuteLimit > 0)
            return StartedAt.AddMinutes(MinuteLimit.Value);
        //return StartedAt.AddSeconds(10);
        return null;
    }
}
}

namespace AegosExam.Models
{
    public class TestSessionAnswer
    {
        [BsonId]
        [BsonRepresentation(BsonType.ObjectId)]
        public string Id { get; set; } = string.Empty;

        [BsonElement("questionId")]
        public string QuestionId { get; set; } = string.Empty;

        [BsonElement("sessionId")]
        public string SessionId { get; set; } = string.Empty;

        [BsonElement("positionIndex")]
        public int PositionIndex { get; set; }

        [BsonElement("selectedAnswers")]
        [BsonIgnoreIfNull]
        public List<int>? SelectedAnswers { get; set; }

        [BsonElement("openAnswer")]
        [BsonIgnoreIfNull]
        public string? OpenAnswer { get; set; }

        [BsonElement("isFlagged")]
        public bool IsFlagged { get; set; } = false;

        [BsonElement("IsVisited")]
        public bool IsVisited { get; set; } = false;

        [BsonIgnore]
        public bool HasAnswer =>
            (SelectedAnswers is { Count: > 0 }) || !string.IsNullOrEmpty(OpenAnswer);
    }
}

```

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				84
Змн.	Арк.	№ докум.	Підпис	Дата		

```

[BsonIgnore]
public bool IsOpenEnded => OpenAnswer != null;
}
}
namespace AegosExam.Models
{
    public class User
    {
        [BsonId]
        [BsonRepresentation(BsonType.ObjectId)]
        public string Id { get; set; } = string.Empty;

        [BsonElement("email")]
        public string Email { get; set; } = string.Empty;

        [BsonElement("passwordHash")]
        public string PasswordHash { get; set; } = string.Empty;

        [BsonElement("firstName")]
        public string FirstName { get; set; } = string.Empty;

        [BsonElement("surname")]
        public string LastName { get; set; } = string.Empty;

        [BsonElement("role")]
        public UserRole Role { get; set; } = UserRole.RegularUser;

        [BsonElement("isBlocked")]
        public bool IsBlocked { get; set; } = false;

        [BsonElement("isDeleted")]
        public bool IsDeleted { get; set; } = false;

        [BsonElement("organization")]
        public string Organization { get; set; } = string.Empty;

        [BsonElement("position")]
        public string Position { get; set; } = string.Empty;

        [BsonElement("createdAt")]
        [BsonDateTimeOptions(Kind = DateTimeKind.Utc)]
        public DateTime CreatedAt { get; set; }

        [BsonElement("lastModifiedAt")]
        [BsonDateTimeOptions(Kind = DateTimeKind.Utc)]
        public DateTime LastModifiedAt { get; set; }

        [BsonIgnore]
        public string FullName => $"{FirstName} {LastName}";
    }
}

```

```

public enum UserRole
{
    RegularUser,
    TestOrganizer,
    Admin
}

namespace AegosExam.Data.Repositories
{
    public class QuestionRepository : IQuestionRepository
    {
        private readonly IMongoCollection<Question> _questions;

        public QuestionRepository(MongoDbContext dbContext)
        {
            _questions = dbContext.Database.GetCollection<Question>("questions");
        }

        public async Task<Question?> GetQuestionByIdAsync(string id)
        {
            return await _questions.Find(q => q.Id == id).FirstOrDefaultAsync();
        }

        public async Task<List<Question>> GetManyByIdsAsync(IEnumerable<string> ids)
        {
            var filter = Builders<Question>.Filter.In(q => q.Id, ids);
            return await _questions.Find(filter).ToListAsync();
        }

        public async Task CreateQuestionAsync(Question question)
        {
            await _questions.InsertOneAsync(question);
        }

        public async Task CreateManyAsync(List<Question> questions)
        {
            if (questions == null || questions.Count == 0)
                return;

            await _questions.InsertManyAsync(questions);
        }

        public async Task UpdateQuestionAsync(Question question)
        {
            var result = await _questions.ReplaceOneAsync(q => q.Id == question.Id, question);
            if (!result.IsAcknowledged || result.ModifiedCount == 0)
            {
                throw new InvalidOperationException($"Failed to update question with ID {question.Id}. It may not exist.");
            }
        }
    }
}

```

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				86
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public async Task DeleteQuestionAsync(string id)
{
    var result = await _questions.DeleteOneAsync(q => q.Id == id);
    if (!result.IsAcknowledged || result.DeletedCount == 0)
    {
        throw new InvalidOperationException($"Failed to delete question with ID {id}. It
may not exist.");
    }
}

public async Task DeleteManyAsync(IEnumerable<string> questionIds)
{
    if (questionIds == null || !questionIds.Any())
        return;

    var filter = Builders<Question>.Filter.In(q => q.Id, questionIds);
    await _questions.DeleteManyAsync(filter);
}

}

namespace AegosExam.Data.Repositories
{
    public class TestRepository : ITestRepository
    {
        private readonly IMongoCollection<Test> _tests;

        public TestRepository(MongoDbContext dbContext)
        {
            _tests = dbContext.Database.GetCollection<Test>("tests");
        }

        public async Task<List<Test>> GetAllTestsAsync()
        {
            return await _tests.Find(_ => true).ToListAsync();
        }

        public async Task<Test?> GetTestByIdAsync(string id)
        {
            return await _tests.Find(t => t.Id == id).FirstOrDefaultAsync();
        }

        public async Task CreateTestAsync(Test test)
        {
            await _tests.InsertOneAsync(test);
        }

        public async Task UpdateTestAsync(Test test)
        {
            var result = await _tests.ReplaceOneAsync(t => t.Id == test.Id, test);

```

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				87
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (!result.IsAcknowledged || result.ModifiedCount == 0)
        {
            throw new InvalidOperationException($"Failed to update test with ID {test.Id}. It
may not exist.");
        }
    }

    public async Task DeleteTestAsync(string id)
    {
        var result = await _tests.DeleteOneAsync(t => t.Id == id);
        if (!result.IsAcknowledged || result.DeletedCount == 0)
        {
            throw new InvalidOperationException($"Failed to delete test with ID {id}. It may
not exist.");
        }
    }

    public async Task<bool> IsAccessCodeUsedAsync(string accessCode)
    {
        var test = await _tests.Find(t => t.AccessCode ==
accessCode).FirstOrDefaultAsync();
        return test != null;
    }
}

namespace AegosExam.Data.Repositories
{
    public class TestResultRepository : ITestResultRepository
    {
        private readonly IMongoCollection<TestResult> _results;

        public TestResultRepository(MongoDbContext dbContext)
        {
            _results = dbContext.Database.GetCollection<TestResult>("test_results");
        }

        public async Task CreateAsync(TestResult result)
        {
            await _results.InsertOneAsync(result);
        }

        public async Task<TestResult?> GetByIdAsync(string id)
        {
            return await _results.Find(r => r.Id == id).FirstOrDefaultAsync();
        }

        public async Task<List<TestResult>> GetUserAndTestIdAsync(string userId, string
testId)
        {
            var filter = Builders<TestResult>.Filter.Eq(r => r.UserId, userId) &
Builders<TestResult>.Filter.Eq(r => r.TestId, testId);

```

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				88
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return await _results.Find(filter).ToListAsync();
    }
}

namespace AegosExam.Data.Repositories
{
    public class TestSessionAnswerRepository : ITestSessionAnswerRepository
    {
        private readonly IMongoCollection<TestSessionAnswer> _answers;

        public TestSessionAnswerRepository(MongoDbContext dbContext)
        {
            _answers =
dbContext.Database.GetCollection<TestSessionAnswer>("test_session_answers");
        }

        public async Task<TestSessionAnswer?> GetAnswerByIdAsync(string id)
        {
            return await _answers.Find(a => a.Id == id).FirstOrDefaultAsync();
        }

        public async Task<List<TestSessionAnswer>>
GetManyByIdsAsync(IEnumerable<string> ids)
        {
            var filter = Builders<TestSessionAnswer>.Filter.In(a => a.Id, ids);
            return await _answers.Find(filter).ToListAsync();
        }

        public async Task CreateAnswerAsync(TestSessionAnswer answer)
        {
            await _answers.InsertOneAsync(answer);
        }

        public async Task CreateManyAsync(List<TestSessionAnswer> answers)
        {
            if (answers == null || answers.Count == 0)
                return;

            await _answers.InsertManyAsync(answers);
        }

        public async Task UpdateAnswerAsync(TestSessionAnswer answer)
        {
            var result = await _answers.ReplaceOneAsync(a => a.Id == answer.Id, answer);
            if (!result.IsAcknowledged || result.ModifiedCount == 0)
            {
                throw new InvalidOperationException($"Failed to update answer with ID
{answer.Id}. It may not exist.");
            }
        }
    }
}

```

```

public async Task DeleteAnswerAsync(string id)
{
    var result = await _answers.DeleteOneAsync(a => a.Id == id);
    if (!result.IsAcknowledged || result.DeletedCount == 0)
    {
        throw new InvalidOperationException($"Failed to delete answer with ID {id}. It
may not exist.");
    }
}

public async Task DeleteManyAsync(IEnumerable<string> answerIds)
{
    if (answerIds == null || !answerIds.Any())
        return;

    var filter = Builders<TestSessionAnswer>.Filter.In(a => a.Id, answerIds);
    await _answers.DeleteManyAsync(filter);
}
}

namespace AegosExam.Data.Repositories
{
    public class TestSessionRepository : ITestSessionRepository
    {
        private readonly IMongoCollection<TestSession> _collection;

        public TestSessionRepository(MongoDbContext dbContext)
        {
            _collection = dbContext.Database.GetCollection<TestSession>("test_sessions");
        }

        public async Task<TestSession?> GetByIdAsync(string id)
        {
            return await _collection.Find(s => s.Id == id).FirstOrDefaultAsync();
        }

        public async Task<List<TestSession>> GetByUserIdAsync(string userId)
        {
            return await _collection.Find(s => s.UserId == userId).ToListAsync();
        }

        public async Task<TestSession?> GetByUserAndTestIdAsync(string userId, string
testId)
        {
            var filter = Builders<TestSession>.Filter.And(
                Builders<TestSession>.Filter.Eq(s => s.UserId, userId),
                Builders<TestSession>.Filter.Eq(s => s.TestId, testId)
            );

            var session = await _collection.Find(filter).FirstOrDefaultAsync();

```

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return session;
    }

    public async Task CreateAsync(TestSession session)
    {
        await _collection.InsertOneAsync(session);
    }

    public async Task UpdateAsync(TestSession session)
    {
        var result = await _collection.ReplaceOneAsync(s => s.Id == session.Id, session);
        if (!result.IsAcknowledged || result.ModifiedCount == 0)
        {
            throw new InvalidOperationException($"Failed to update session with ID {session.Id}.");
        }
    }

    public async Task DeleteAsync(string id)
    {
        var result = await _collection.DeleteOneAsync(s => s.Id == id);
        if (!result.IsAcknowledged || result.DeletedCount == 0)
        {
            throw new InvalidOperationException($"Failed to delete session with ID {id}.");
        }
    }

    public async Task<List<TestSession>> GetExpiredSessionsAsync(DateTime
currentTimeUtc)
    {
        var gracePeriod = TimeSpan.FromSeconds(30);

        var sessions = await _collection
            .Find(s => s.MinuteLimit.HasValue && s.MinuteLimit > 0)
            .ToListAsync();

        return sessions
            .Where(s => s.EndTime < currentTimeUtc - gracePeriod)
            .ToList();
    }
}

namespace AegosExam.Data.Repositories
{
    public class UserAnswerRepository : IUserAnswerRepository
    {
        private readonly IMongoCollection<TestResultAnswer> _answers;

        public UserAnswerRepository(MongoDbContext dbContext)
        {

```

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        _answers
dbContext.Database.GetCollection<TestResultAnswer>("user_answers");
    }

    public async Task<List<TestResultAnswer>> GetManyByIdsAsync(List<string> ids)
    {
        var filter = Builders<TestResultAnswer>.Filter.In(a => a.Id, ids);
        return await _answers.Find(filter).ToListAsync();
    }

    public async Task CreateManyAsync(List<TestResultAnswer> answers)
    {
        if (answers == null || !answers.Any()) return;
        await _answers.InsertManyAsync(answers);
    }
}

namespace AegosExam.Data.Repositories
{
    public class UserRepository : IUserRepository
    {
        private readonly IMongoCollection<User> _users;

        public UserRepository(MongoDbContext dbContext)
        {
            _users = dbContext.Database.GetCollection<User>("users");
        }

        public async Task<List<User>> GetAllUsersAsync()
        {
            return await _users.Find(_ => true).ToListAsync();
        }

        public async Task<User?> GetUserByEmailAsync(string email)
        {
            return await _users.Find(u => u.Email == email).FirstOrDefaultAsync();
        }

        public async Task<User?> GetUserByIdAsync(string id)
        {
            return await _users.Find(u => u.Id == id).FirstOrDefaultAsync();
        }

        public async Task InsertUserAsync(User user)
        {
            await _users.InsertOneAsync(user);
        }

        public async Task UpdateUserAsync(User user)
        {
            await _users.ReplaceOneAsync(u => u.Id == user.Id, user);
        }
    }
}

```

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				92
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    public async Task DeleteUserAsync(string id)
    {
        await _users.DeleteOneAsync(u => u.Id == id);
    }
}

namespace AegosExam.Data
{
    public class MongoDBContext
    {
        private readonly IMongoDatabase _database;

        public MongoDBContext(IOptions<MongoDbSettings> settings)
        {
            var mongoSettings = MongoClientSettings.FromUrl(new
MongoUrl(settings.Value.ConnectionString));
            mongoSettings.SslSettings = new SslSettings { EnabledSslProtocols =
System.Security.Authentication.SslProtocols.Tls12 };

            var client = new MongoClient(mongoSettings);
            _database = client.GetDatabase(settings.Value.DatabaseName);
        }

        public IMongoDatabase Database => _database;
    }
}

namespace AegosExam.Data
{
    public class MongoDBSettings
    {
        public string ConnectionString { get; set; } = string.Empty;
        public string DatabaseName { get; set; } = string.Empty;
    }
}

namespace AegosExam.Logic.Services
{
    public class UserService : IUserService
    {
        private readonly IUserRepository _userRepository;

        public UserService(IUserRepository userRepository)
        {
            _userRepository = userRepository;
        }

        public async Task<List<User>> GetAllAsync()
        {

```

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				93
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return await _userRepository.GetAllUsersAsync();
    }

    public async Task RegisterUserAsync(User user, string password)
    {
        var existingUser = await _userRepository.GetUserByEmailAsync(user.Email);
        if (existingUser != null)
        {
            throw new InvalidOperationException($"Помилка реєстрації: користувач з електронною поштою '{user.Email}' вже існує. Будь ласка, використайте іншу адресу.");
        }

        user.PasswordHash = BCrypt.Net.BCrypt.HashPassword(password);
        user.CreatedAt = DateTime.UtcNow;
        user.LastModifiedAt = DateTime.UtcNow;

        await _userRepository.InsertUserAsync(user);
    }

    public async Task<User> AuthenticateUserAsync(string email, string password)
    {
        var user = await _userRepository.GetUserByEmailAsync(email);
        if (user == null || !BCrypt.Net.BCrypt.Verify(password, user.PasswordHash))
        {
            throw new UnauthorizedAccessException("Помилка входу: електронна пошта або пароль вказані невірно.");
        }

        if (user.IsBlocked)
        {
            throw new UnauthorizedAccessException("Ваш обліковий запис заблоковано. Зверніться до адміністратора.");
        }

        if (user.IsDeleted)
        {
            throw new UnauthorizedAccessException("Ваш обліковий запис видалено.");
        }

        return user;
    }

    public async Task<User?> GetByIdAsync(string id)
    {
        return await _userRepository.GetByIdAsync(id);
    }

    public async Task UpdateUserAsync(User user)
    {
        user.LastModifiedAt = DateTime.UtcNow;
        await _userRepository.UpdateUserAsync(user);
    }

```

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				94
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        public async Task DeleteUserAsync(string id)
        {
            await _userRepository.DeleteUserAsync(id);
        }
    }
}

namespace AegosExam.Logic.Services
{
    public class TestSessionService : ITestSessionService
    {
        private readonly ITestSessionRepository _sessionRepo;
        private readonly ITestSessionAnswerRepository _answerRepo;
        private readonly ITestRepository _testRepo;
        private readonly IQuestionRepository _questionRepo;

        private readonly IMemoryCache _cache;

        public TestSessionService(
            ITestSessionRepository sessionRepo,
            ITestSessionAnswerRepository answerRepo,
            ITestRepository testRepo,
            IQuestionRepository questionRepo,
            IMemoryCache cache)
        {
            _sessionRepo = sessionRepo;
            _answerRepo = answerRepo;
            _testRepo = testRepo;
            _questionRepo = questionRepo;
            _cache = cache;
        }

        public async Task<TestSession?> GetSessionByIdAsync(string sessionId)
        {
            return await _sessionRepo.GetByIdAsync(sessionId);
        }

        public async Task<TestSession?> GetSessionByUserAndTestIdAsync(string userId,
string testId)
        {
            return await _sessionRepo.GetByUserAndTestIdAsync(userId, testId);
        }

        public async Task<List<TestSession>> GetSessionsByUserAsync(string userId)
        {
            return await _sessionRepo.GetByUserIdAsync(userId);
        }

        public async Task<(Test? test, TestSession? session)> GetTestAndSessionAsync(string
userId, string testId)
        {

```

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк. Т. А.				95
Змн.	Арк.	№ докум.	Підпис	Дата		

```

var testTask = _testRepo.GetTestByIdAsync(testId);
var sessionTask = _sessionRepo.GetByUserAndTestIdAsync(userId, testId);

await Task.WhenAll(testTask, sessionTask);

return (testTask.Result, sessionTask.Result);
}

public async Task<TestSession> StartSessionAsync(string userId, Test test, List<string>
questionIds)
{
    if (test.RandomizeQuestionOrder)
    {
        questionIds = questionIds.OrderBy(_ => Guid.NewGuid()).ToList();
    }

    var session = new TestSession
    {
        TestId = test.Id,
        UserId = userId,
        StartedAt = DateTime.UtcNow,
        MinuteLimit = test.MinuteLimit,
    };

    await _sessionRepo.CreateAsync(session);

    var answers = new List<TestSessionAnswer>();
    int positionIndex = 0;
    foreach (var questionId in questionIds)
    {
        var answer = new TestSessionAnswer
        {
            QuestionId = questionId,
            SessionId = session.Id,
            PositionIndex = positionIndex++
        };
        answers.Add(answer);
    }
    await _answerRepo.CreateManyAsync(answers);

    session.AnswerIds = answers.Select(answer => answer.Id).ToList();
    session.QuestionMap = questionIds
        .Select((qid, index) => new QuestionPosition
        {
            PositionIndex = index,
            QuestionId = qid
        }).ToList();

    await _sessionRepo.UpdateAsync(session);

    var questions = await _questionRepo.GetManyByIdsAsync(questionIds);

```

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				96
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        _cache.Set(GetQuestionCacheKey(session.Id), questions.ToDictionary(q => q.Id));

        return session;
    }

    public async Task<Question?> GetQuestionFromCacheOrDbAsync(TestSession
session, string questionId)
    {
        var cacheKey = GetQuestionCacheKey(session.Id);

        if (_cache.TryGetValue(cacheKey, out var obj) && obj is Dictionary<string,
Question> questionDict)
        {
            if (questionDict.TryGetValue(questionId, out var cachedQuestion))
                return cachedQuestion;
        }

        return await _questionRepo.GetQuestionByIdAsync(questionId);
    }

    private string GetQuestionCacheKey(string sessionId)
    {
        return $"questions_{sessionId}";
    }

    public async Task FinishSessionAsync(TestSession session)
    {
        if (session.AnswerIds.Any())
            await _answerRepo.DeleteManyAsync(session.AnswerIds);

        await _sessionRepo.DeleteAsync(session.Id);

        _cache.Remove(GetQuestionCacheKey(session.Id));
    }

    public async Task<List<TestSessionAnswer>>
GetAnswersInSessionAsync(TestSession testSession)
    {
        return await _answerRepo.GetManyByIdsAsync(testSession.AnswerIds);
    }

    public async Task<List<int>> GetPositionIndexesInSessionAsync(TestSession
testSession)
    {
        var answers = await GetAnswersInSessionAsync(testSession);
        return answers.Select(a => a.PositionIndex).ToList();
    }

    public Task<string> GetQuestionIdByPositionIndexAsync(TestSession session, int
positionIndex)
    {

```

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				97
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        var question = session.QuestionMap.FirstOrDefault(q => q.PositionIndex ==
positionIndex);
        if (question == null)
            throw new ArgumentOutOfRangeException(nameof(positionIndex), $"No
question found at index {positionIndex} in the session.");

        return Task.FromResult(question.QuestionId);
    }

    private string GetAnswerIdByPositionIndex(TestSession session, int positionIndex)
    {
        if (positionIndex < 0 || positionIndex >= session.AnswerIds.Count)
            throw new ArgumentOutOfRangeException(nameof(positionIndex), $"No answer
at index {positionIndex} in session {session.Id}.");

        return session.AnswerIds[positionIndex];
    }

    public async Task SaveSelectedAsync(TestSession session, int positionIndex, List<int>
selectedAnswers)
    {
        var answerId = GetAnswerIdByPositionIndex(session, positionIndex);
        var answer = await _answerRepo.GetAnswerByIdAsync(answerId);

        if (answer == null)
            throw new InvalidOperationException($"Answer with ID {answerId} not found for
session {session.Id}.");

        if ((answer.SelectedAnswers ?? new List<int>()).SequenceEqual(selectedAnswers))
            return;

        answer.SelectedAnswers = selectedAnswers;
        answer.OpenAnswer = null;
        answer.IsVisited = true;

        await _answerRepo.UpdateAnswerAsync(answer);
    }

    public async Task SaveOpenEndedAsync(TestSession session, int positionIndex, string
openAnswer)
    {
        var answerId = GetAnswerIdByPositionIndex(session, positionIndex);
        var answer = await _answerRepo.GetAnswerByIdAsync(answerId);

        if (answer == null)
            throw new InvalidOperationException($"Answer with ID {answerId} not found for
session {session.Id}.");

        var trimmed = openAnswer?.Trim();

        if (string.Equals(answer.OpenAnswer, trimmed, StringComparison.Ordinal))

```

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				98
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return;

        answer.OpenAnswer = trimmed;
        answer.SelectedAnswers = null;
        answer.IsVisited = true;

        await _answerRepo.UpdateAnswerAsync(answer);
    }

    public async Task MarkVisitedAsync(TestSession session, int positionIndex)
    {
        var answerId = GetAnswerIdByPositionIndex(session, positionIndex);
        var answer = await _answerRepo.GetAnswerByIdAsync(answerId);

        if (answer == null)
            throw new InvalidOperationException($"Answer not found for position
{positionIndex} in session {session.Id}");

        if (!answer.IsVisited)
        {
            answer.IsVisited = true;
            await _answerRepo.UpdateAnswerAsync(answer);
        }
    }

    public int GetQuestionCount(TestSession session)
    {
        return session.AnswerIds.Count;
    }

    public async Task<TestSessionAnswer?> GetAnswerAtIndexAsync(TestSession
session, int positionIndex)
    {
        var answerId = GetAnswerIdByPositionIndex(session, positionIndex);
        return await _answerRepo.GetAnswerByIdAsync(answerId);
    }

    public async Task<bool> ToggleFlagAsync(TestSession session, int positionIndex)
    {
        var answerId = GetAnswerIdByPositionIndex(session, positionIndex);
        var answer = await _answerRepo.GetAnswerByIdAsync(answerId);

        if (answer == null)
            throw new InvalidOperationException($"Answer with ID {answerId} not found in
session {session.Id}.");

        answer.IsFlagged = !answer.IsFlagged;

        await _answerRepo.UpdateAnswerAsync(answer);
        return answer.IsFlagged;
    }
}

```

```

public async Task<bool> ClearAnswerAsync(TestSession session, int positionIndex)
{
    var answerId = GetAnswerIdByPositionIndex(session, positionIndex);
    var answer = await _answerRepo.GetAnswerByIdAsync(answerId);

    if (answer == null)
        throw new InvalidOperationException($"Answer with ID {answerId} not found in
session {session.Id}.");

    // Nothing to clear
    if (string.IsNullOrEmpty(answer.OpenAnswer) && (answer.SelectedAnswers
== null || answer.SelectedAnswers.Count == 0))
        return false;

    answer.OpenAnswer = null;
    answer.SelectedAnswers = null;

    await _answerRepo.UpdateAnswerAsync(answer);
    return true;
}

public async Task<List<TestSession>> GetExpiredSessionsAsync()
{
    return await _sessionRepo.GetExpiredSessionsAsync(DateTime.UtcNow);
}
}

namespace AegosExam.Logic.Services
{
    public class TestService : ITestService
    {
        private readonly ITestRepository _testRepository;
        private readonly IQuestionRepository _questionRepository;

        public TestService(ITestRepository testRepository, IQuestionRepository
questionRepository)
        {
            _testRepository = testRepository;
            _questionRepository = questionRepository;
        }

        public async Task<List<Test>> GetAllTestsAsync()
        {
            return await _testRepository.GetAllTestsAsync();
        }

        public async Task<Test?> GetTestByIdIfExistsAsync(string id)
        {
            return await _testRepository.GetTestByIdAsync(id);
        }
    }
}

```

```

public async Task<List<Question>> GetQuestionsLinkedToTheTestAsync(string id)
{
    var test = await _testRepository.GetTestByIdAsync(id);
    if (test == null)
        throw new ArgumentException($"Test with ID {id} not found.");

    return await _questionRepository.GetManyByIdsAsync(test.QuestionIds);
}

public async Task<List<Question>> GetQuestionsByIdsAsync(List<string> ids)
{
    return await _questionRepository.GetManyByIdsAsync(ids);
}

public async Task CreateTestAsync(Test test)
{
    string accessCode;
    do
    {
        accessCode = AccessCodeGenerator.Generate();
    } while (await _testRepository.IsAccessCodeUsedAsync(accessCode));

    test.AccessCode = accessCode;
    test.CreatedAt = DateTime.UtcNow;
    test.LastModifiedAt = DateTime.UtcNow;

    await _testRepository.CreateTestAsync(test);
}

public async Task UpdateTestAsync(Test test)
{
    test.LastModifiedAt = DateTime.UtcNow;
    await _testRepository.UpdateTestAsync(test);
}

public async Task DeleteTestAsync(string id)
{
    var test = await _testRepository.GetTestByIdAsync(id);
    if (test == null)
        throw new ArgumentException($"Test with ID {id} not found.");

    if (test.QuestionCount > 0)
    {
        await _questionRepository.DeleteManyAsync(test.QuestionIds);
    }

    await _testRepository.DeleteTestAsync(id);
}

public async Task ChangeTestStateAsync(string id, TestState testState)
{
    var test = await _testRepository.GetTestByIdAsync(id);

```

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				101
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (test == null)
            throw new ArgumentException($"Test with ID {id} not found.");

        test.State = testState;
        await UpdateTestAsync(test);
    }
}

```

		Сахно А.О.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Вакалюк Т. А.				102
Змн.	Арк.	№ докум.	Підпис	Дата		