МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет водного господарства та природокористування
Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра комп'ютерних технологій та економічної кібернетики

Допущено до захисту:
Завідувач кафедри
комп'ютерних технологій та
економічної кібернетики
д. е. н., проф. П. М. Грицюк
_____
«____» _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА НА ЗДОБУТТЯ ОСВІТНЬО-КВАЛІФІКАЦІЙНОГО РІВНЯ «МАГІСТР»

Інтелектуальна система онлайн тестування з моніторингом поведінки студентів на основі розпізнавання обличчя

Виконав:
Здобувач вищої освіти за ОПП
«Інформаційні технології в бізнесі»
спеціальності 126 «Інформаційні системи та технології», групи ІТБ-61м
Опанасюк Михайло Юрійович

Керівник:
к.е.н., доцент Волошин В.С.

Рецензент:
д.е.н., проф. Грицюк П.М.

Рівне 2024

РЕФЕРАТ

Кваліфікаційна робота магістра: 61 с., 39 рис., 1 табл., 17 літературних джерел.

Актуальність теми даної магістерської роботи полягає у використанні сучасних технологій штучного інтелекту та комп'ютерного зору для створення інтелектуальних систем, що здатні забезпечити ефективність і прозорість процесу онлайн тестування.

Об'єкт дослідження магістерської роботи – система онлайн тестування.

Предметна область роботи – інтеграція алгоритмів розпізнавання обличчя та поведінки студентів для контролю чесності проходження тестувань у реальному часі.

Метою магістерської роботи є розробка інтелектуальної системи для проведення онлайн тестувань із вбудованим моніторингом поведінки студентів, що забезпечує підвищення рівня надійності результатів тестування та мінімізує можливості недобросовісної поведінки.

У магістерській роботі: описано актуальність впровадження інтелектуальних систем у процес дистанційного навчання; проведено аналіз існуючих рішень для розпізнавання облич та моніторингу поведінки студентів; обґрунтовано вибір технологій для реалізації системи, включаючи Python, C# та React; розроблено архітектуру та модель даних системи онлайн тестування; впроваджено алгоритми для виявлення нечесної поведінки студентів під час тестування; реалізовано та протестовано інтелектуальну систему, яка використовує Haar-каскади для розпізнавання обличчя та відстеження напрямку погляду; створено телеграм-бота для учасників тестування із можливістю відстеження поточних результатів та отримання нагадувань про наступні тестування.

КЛЮЧОВІ СЛОВА: ІНТЕЛЕКТУАЛЬНА ІНФОРМАЦІЙНА СИСТЕМА, ОНЛАЙН ТЕСТУВАННЯ, КОНТРОЛЬ ПОВЕДІНКИ, РОЗПІЗНАВАННЯ ОБЛИЧ, HAAR-КАСКАДИ, КОМП’ЮТЕРНИЙ ЗІР, FRONTEND, BACKEND, БАЗИ ДАНИХ.

ЗМІСТ

ВСТУП

Онлайн тестування стало важливою складовою сучасної освітньої системи, відображаючи актуальні тенденції та виклики цифрової епохи. Зростаючий попит на дистанційне навчання потребує розробки інтелектуальних систем, які забезпечують не лише ефективне проведення тестувань, але й контроль за чесністю їх проходження. Це стає особливо актуальним в умовах глобалізації та необхідності підтримувати високі стандарти якості освіти для широкого кола користувачів.

Інтеграція технологій розпізнавання обличчя та моніторингу поведінки студентів дозволяє створити систему, що автоматизує процеси контролю та аналізу під час тестувань. Такі рішення значно підвищують довіру до результатів тестування, забезпечуючи справедливість і прозорість процесу.

Метою даної роботи є проектування та розробка інтелектуальної системи онлайн тестування, яка інтегрує моніторинг поведінки студентів, телеграм-бот для нотифікацій, а також інструменти для автоматизації створення тестів за допомогою OpenAI.

Об’єктом дослідження магістерської роботи є організація проведення онлайн тестувань.

Предметом дослідження є інтеграція сучасних технологій для створення інтелектуальної системи онлайн тестування.

Для досягнення мети магістерської роботи поставлені такі завдання:

1. Визначити концепції онлайн тестування як сучасного формату перевірки знань.
2. Дослідити існуючі технології моніторингу поведінки студентів під час тестувань.
3. Проаналізувати результати учасників НМТ.

4. Розробити архітектуру системи, яка інтегрує розпізнавання обличчя, телеграм-бот для сповіщень і автоматизацію створення тестів.
5. Реалізувати Backend та Frontend системи за допомогою .NET Core 8, EF Core, Hangfire, MS SQL та C#.

Логіка дослідження визначила **структуру** магістерської роботи, яка складається зі вступу, трьох розділів, висновків та списку використаних джерел.

У вступі обґрунтовано актуальність теми, наведено аналіз сучасних тенденцій у використанні інтелектуальних систем для онлайн тестування, визначено об’єкт і предмет дослідження, а також сформульовано мету роботи та основні завдання. Цей розділ також підкреслює важливість впровадження інноваційних технологій, таких як розпізнавання обличчя, моніторинг поведінки та автоматизація тестування, для покращення якості освітнього процесу.

Перший розділ присвячено теоретичним основам систем онлайн тестування. Проведено аналіз ролі онлайн тестування у системі контролю якості знань, спираючись на сучасні дослідження та джерела, включаючи огляди в Інтернеті. Цей розділ також містить огляд і аналіз існуючих платформ для тестування, таких як Google Forms, Moodle, Quizizz, Kahoot тощо, з додаванням ілюстрацій для демонстрації їхнього інтерфейсу та функціональних можливостей.

Другий розділ присвячено інформаційній підтримці функціонування онлайн тестування. У ньому здійснено статистичний аналіз результатів учасників НМТ (Національного мультипредметного тесту) на основі наданих звітів, де вибрано найкращі діаграми та описано їх текстом. У розділі також досліджуються результати учасників НМТ у розрізі навчальних предметів із

використанням відповідних діаграм для ілюстрації різниць у підготовці учасників залежно від предмету.

Розділ 3 фокусується на розробці інтелектуальної системи онлайн тестування. Тут описано програмну реалізацію проєкту, включаючи архітектуру системи, використані технології (.NET Core 8, React, Redux, TypeScript, Python, Hangfire, EF Core, MS SQL) та приклади програмного коду. Також висвітлено інтеграцію телеграм-бота для сповіщень і OpenAI API для автоматизації створення тестів. У підрозділах детально описано функціональні можливості системи з точки зору користувача: реєстрація, авторизація через Google, проходження тестів, моніторинг поведінки за допомогою алгоритмів розпізнавання облич, а також нотифікації через Telegram.

Практична цінність магістерської роботи полягає у розробці ефективної інтелектуальної системи онлайн тестування, що не лише спрощує процес організації тестування, а й забезпечує чесність та прозорість.

Висновки містять підсумки роботи, виконані завдання та рекомендації щодо подальшого вдосконалення системи, зокрема розширення функціональності та інтеграції з іншими платформами.

При виконанні магістерської роботи використовувались наступні програмні інструменти: Visual Studio Code, Visual Studio, .NET Core 8, EF Core, Hangfire, MS SQL, React, Redux, TypeScript, Python, Flask, Telegram API, OpenAI API, Microsoft Word. Ці технології дозволили створити сучасну інтегровану систему з багатим функціоналом для проведення онлайн тестування.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ СИСТЕМ ОНЛАЙН ТЕСТУВАННЯ

1.1. Онлайн-тестування в системі контролю якості знань

Онлайн-тестування стало невід'ємною складовою сучасної системи контролю якості знань, пропонуючи ефективні та об'єктивні методи оцінювання навчальних досягнень студентів.

У статті «Тестування як метод вимірювання якості навчальних досягнень студентів педагогічних спеціальностей» [10] Т. А. Щебликіна досліджує сутність тестування, його переваги та недоліки, а також принципи застосування в освітньому процесі вищих навчальних закладів.

Вона визначає тест як сукупність стандартизованих завдань, що дозволяють вимірювати рівень сформованості певних складників навчальних досягнень студентів. Тестування, у свою чергу, розглядається як науково обґрунтована процедура діагностики, що забезпечує об'єктивність та стандартизацію оцінювання. Серед переваг тестування виділяються можливість одночасного охоплення великої кількості студентів, швидкість отримання результатів та зниження суб'єктивного впливу викладача на оцінку.

Водночас авторка звертає увагу на певні недоліки тестування, зокрема можливість формального підходу студентів до підготовки, орієнтацію на запам'ятовування фактів без глибокого розуміння матеріалу та труднощі у розробці якісних тестових завдань. Для ефективного застосування тестування в освітньому процесі рекомендується дотримуватися принципів валідності, надійності та об'єктивності тестів, а також поєднувати тестування з іншими методами оцінювання для комплексного підходу до контролю знань студентів.

Таким чином, стаття Т. А. Щебликіної надає ґрунтовний аналіз тестування як методу вимірювання якості навчальних досягнень, підкреслюючи його значення в підготовці майбутніх педагогів та необхідність

вдосконалення цього інструменту для підвищення ефективності освітнього процесу.

Також у статті «Тестування як форма контролю та діагностики знань здобувачів вищої освіти» [11] Г. Ю. Мороховець досліджує роль тестування в освітньому процесі, особливо в контексті медичних спеціальностей.

Авторка підкреслює, що тестування є науково обґрунтованим методом діагностики, який забезпечує стандартизацію та об'єктивність оцінювання знань студентів. Вона зазначає, що тестовий контроль дозволяє швидко та ефективно перевіряти рівень засвоєння матеріалу, охоплюючи значний обсяг інформації за короткий час. Це особливо важливо в умовах кредитно-модульної системи навчання, де акцент робиться на самостійному оволодінні знаннями.

Мороховець також розглядає різні форми тестових завдань, зокрема тести відкритої та закритої форм, тест-альтернативи та тести-відповідності. Вона наголошує на важливості дотримання методологічних принципів при розробці тестів, таких як валідність, надійність та об'єктивність. Авторка підкреслює, що якісно розроблені тести сприяють активізації пізнавальної діяльності студентів та підвищенню їхньої мотивації до навчання.

Тож стаття Г. Ю. Мороховець акцентує увагу на значущості тестування як форми контролю та діагностики знань здобувачів вищої освіти, підкреслюючи його роль у забезпеченні якості освітнього процесу та підготовці кваліфікованих фахівців.

Висновок

Онлайн-тестування є важливим інструментом у системі контролю якості знань, який забезпечує стандартизацію, об'єктивність та ефективність оцінювання навчальних досягнень студентів. Аналіз досліджень показав, що тестування дозволяє швидко перевіряти рівень засвоєння матеріалу та

охоплювати значний обсяг інформації, що є особливо актуальним у сучасній освітній системі.

Водночас використання цього методу потребує дотримання методологічних принципів, таких як валідність, надійність та об'єктивність, а також поєднання з іншими формами оцінювання для комплексного підходу до контролю знань. Успішне впровадження онлайн-тестування сприятиме підвищенню якості освітнього процесу та підготовці кваліфікованих фахівців.

1.2. Огляд та аналіз існуючих онлайн платформ для тестування

Онлайн-платформи для тестування займають важливе місце у сучасній освіті, забезпечуючи зручні та ефективні інструменти для контролю знань студентів. Вони дозволяють створювати різноманітні завдання, організовувати тестування в асинхронному чи реальному режимі, автоматизувати оцінювання та забезпечувати швидкий зворотний зв’язок.

Огляд існуючих платформ допоможе зрозуміти їхні функціональні можливості та визначити, які з них найкраще підходять для конкретних потреб. У цьому підрозділі розглядаються найпопулярніші платформи, зокрема Google Forms, Moodle, Quizizz та Kahoot, які широко використовуються у різних навчальних закладах.

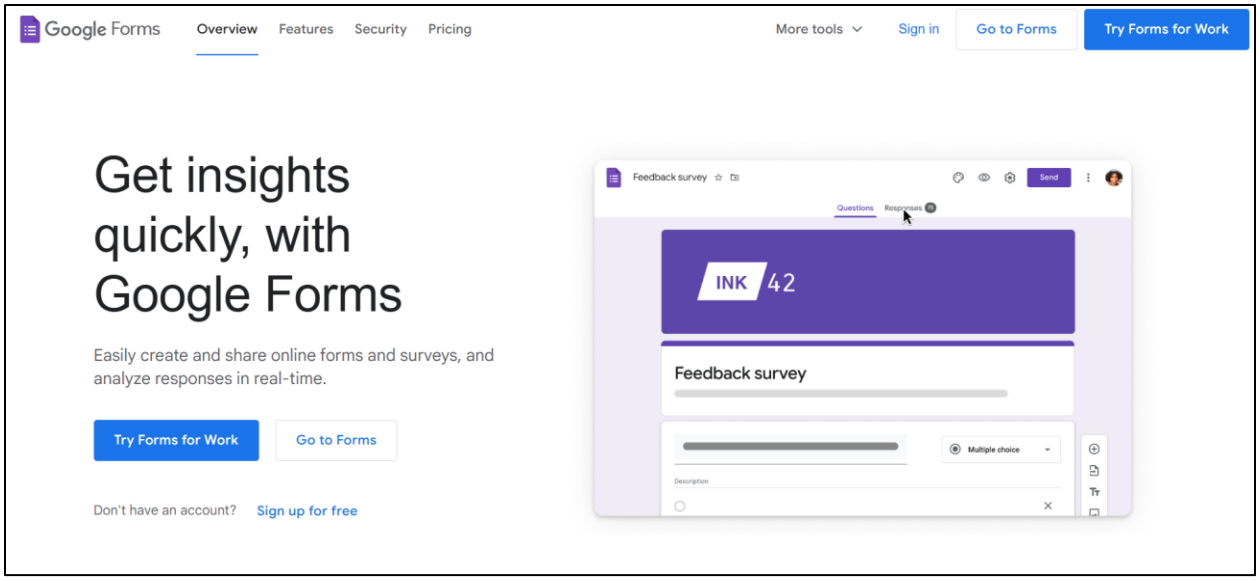

Рис. 1.1. Веб-сайт «Google Forms»

Google Forms – це безкоштовний онлайн-інструмент для створення тестів, опитувань та інших форм збору даних. Завдяки інтеграції з екосистемою Google (Google Drive, Google Sheets) цей інструмент став одним із найпопулярніших у світі. Google Forms пропонує простий у використанні інтерфейс, який дозволяє створювати тести без необхідності спеціальних технічних знань.

Основними перевами Google Forms є інтуїтивно зрозумілий інтерфейс, що дозволяє швидко створювати тести навіть користувачам без технічного досвіду; автоматизація збору та аналізу відповідей – відповіді автоматично збираються в Google Sheets, що забезпечує зручний перегляд і аналіз даних; гнучкість у створенні завдань – підтримка різних типів запитань, зокрема вибір одного чи кількох варіантів відповіді, текстові відповіді, шкали та інше; доступність – платформа є безкоштовною для всіх користувачів із Google-акаунтом.

Однак Google Forms має і певні обмеження. Наприклад, платформа не забезпечує інтегрованих функцій для перевірки списування або захисту тестів

від несанкціонованого доступу. Крім того, функціонал для зворотного зв'язку є доволі базовим порівняно з іншими платформами.

Таким чином, Google Forms є ідеальним рішенням для швидкого створення тестів і збору відповідей, проте для організації складніших тестувань із розширеними функціями може знадобитися використання інших платформ.

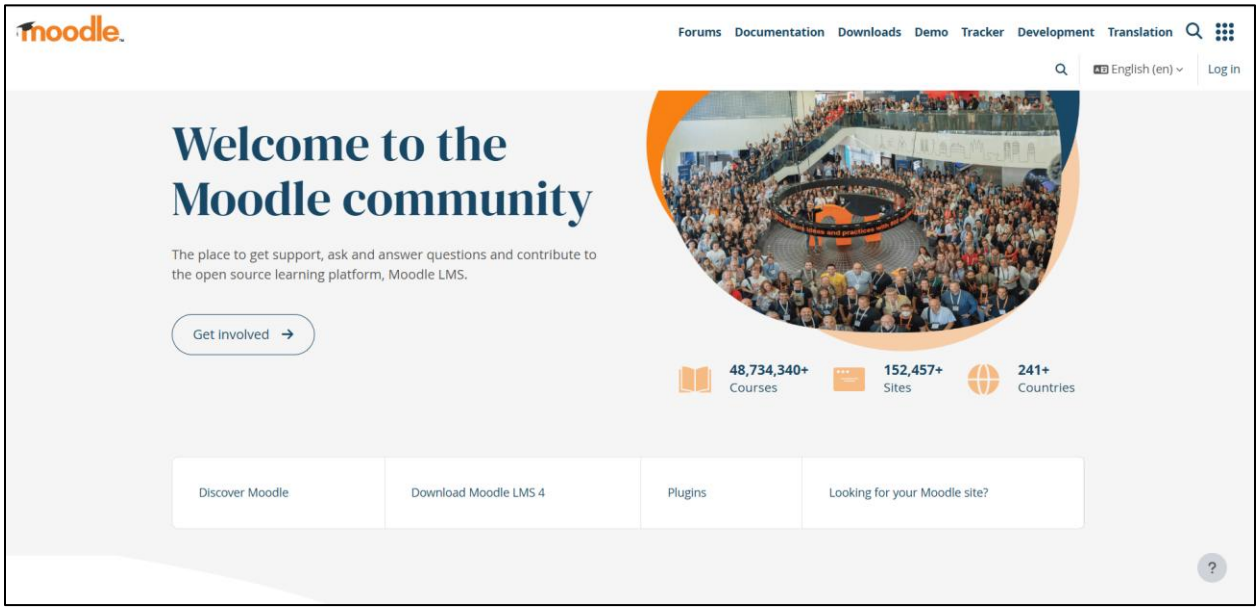

Рис. 1.2. Веб-сайт «Moodle»

Moodle є одним із найпоширеніших інструментів для організації онлайн-навчання та тестування. Як повноцінна система управління навчанням (LMS), вона пропонує широкий спектр функцій, які роблять її популярною серед освітніх закладів у всьому світі. Moodle дозволяє створювати інтерактивні курси, організовувати тестування та забезпечувати безперервний моніторинг успішності студентів.

Серед ключових можливостей Moodle виділяється його здатність підтримувати різноманітні формати тестів, включаючи закриті та відкриті питання, завдання на встановлення відповідності чи числові розрахунки. Гнучка система налаштувань дозволяє викладачам визначати специфічні

правила оцінювання для кожного тесту або курсу, встановлювати часові обмеження та забезпечувати автоматичну перевірку відповідей.

Важливою перевагою Moodle є підтримка плагінів, що дозволяють розширити функціонал платформи залежно від потреб конкретного навчального процесу. Наприклад, можна додати модулі для відеоконференцій, інтерактивних вправ або детального аналізу успішності студентів.

Разом із цим, Moodle забезпечує високий рівень захисту даних, дозволяючи встановлювати обмеження доступу до матеріалів та тестів, шифрувати дані й налаштовувати права користувачів.

Однак використання Moodle потребує певних технічних навичок, особливо під час початкового налаштування та адміністрування платформи. Також хоча сама система є безкоштовною, хостинг і технічна підтримка можуть викликати додаткові витрати.

Загалом Moodle є універсальним рішенням для організації комплексного навчального процесу, що особливо добре підходить для великих освітніх закладів чи організацій, які шукають гнучкість і багатофункціональність.

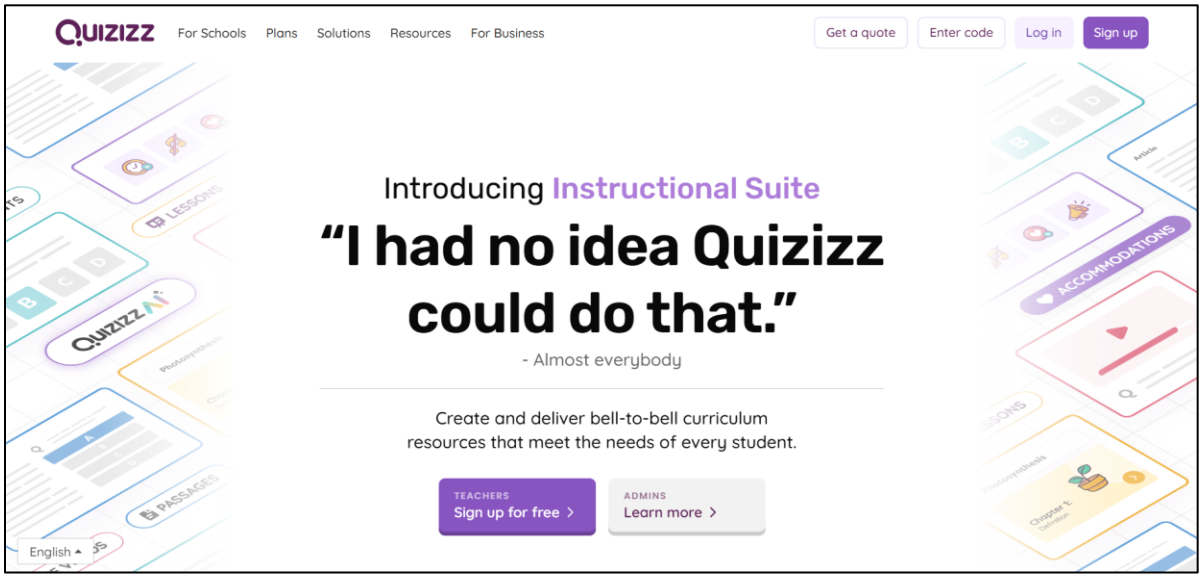

Рис. 1.3. Веб-сайт «Quizizz»

Quizizz – це інтерактивна онлайн-платформа, яка поєднує тестування та елементи гейміфікації, створюючи динамічне навчальне середовище. Завдяки своєму яскравому дизайну та орієнтації на взаємодію з учнями, Quizizz активно використовується у школах і на тренінгах для залучення учасників до навчального процесу.

Головна особливість Quizizz полягає в її здатності проводити тести в реальному часі або асинхронно. Це означає, що викладачі можуть організовувати змагання в класі чи дати можливість студентам проходити завдання у зручний для них час. Система автоматично збирає відповіді, надає зворотний зв’язок і генерує звіти про успішність.

Користувачам доступний широкий вибір шаблонів для створення завдань, включаючи запитання з варіантами відповіді, відкриті завдання чи завдання на встановлення відповідності. Крім того, Quizizz інтегрує елементи гейміфікації, як-от очки, рейтинги та музичний супровід, що допомагає мотивувати учнів та створює захопливу атмосферу.

Однією з переваг платформи є її простота у використанні: для початку роботи достатньо створити акаунт і вибрати відповідний формат завдань. Платформа також підтримує інтеграцію з Google Classroom та іншими сервісами, що робить її універсальним інструментом для викладачів.

Незважаючи на свої переваги, Quizizz має певні обмеження, як-от відсутність розширених інструментів для аналізу чи можливостей автоматичного запобігання списуванню. Водночас її основний акцент на гейміфікацію робить платформу менш придатною для проведення серйозних академічних оцінювань.

Quizizz – це чудовий вибір для створення тестів, які одночасно навчають і розважають. Її інтерактивний підхід сприяє підвищенню зацікавленості

учнів, що особливо важливо в умовах дистанційного навчання чи неформальних освітніх заходів.

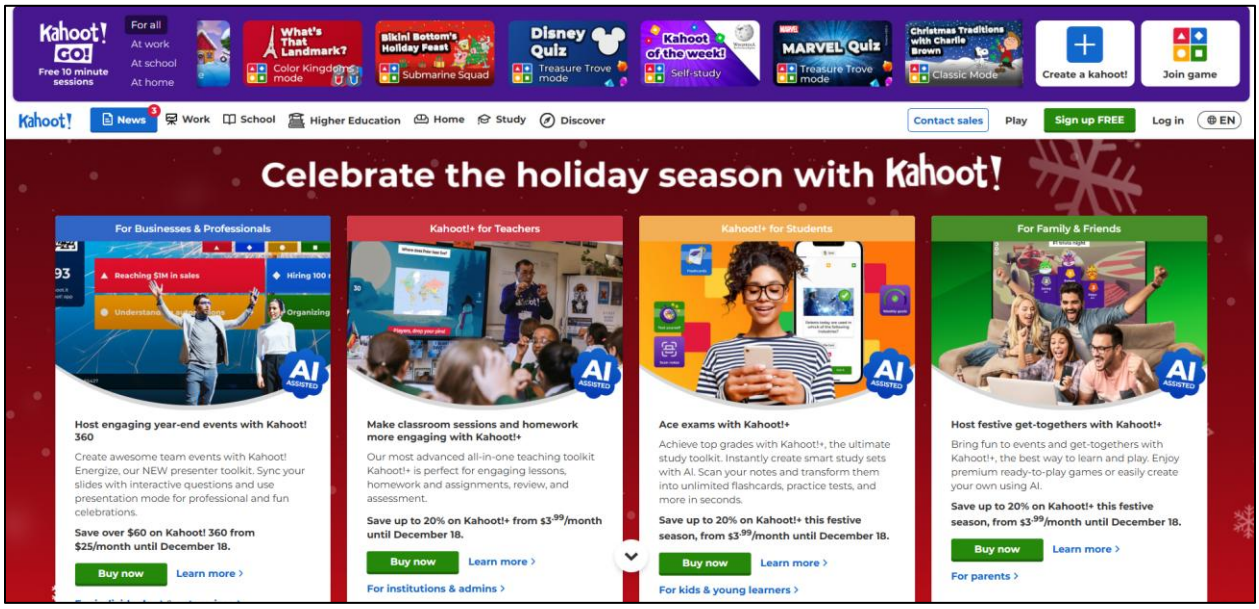

Рис. 1.4. Веб-сайт «Kahoot»

Kahoot — це онлайн-платформа, створена для інтерактивного навчання та тестування, яка орієнтована на залучення аудиторії через гейміфікацію. Завдяки веселому і яскравому дизайну, Kahoot особливо популярна серед викладачів шкіл та організаторів тренінгів, які прагнуть зробити навчання цікавим і динамічним.

Головна ідея Kahoot полягає у створенні вікторин, які можна проводити у формі гри в реальному часі. Учасники підключаються до тесту через спеціальний код на своїх пристроях, а запитання з’являються на загальному екрані. Відповіді учасників реєструються миттєво, а платформа формує рейтинг на основі правильності та швидкості відповідей.

Kahoot підтримує різноманітні формати запитань, зокрема вибір однієї або кількох правильних відповідей, порядкові завдання чи навіть інтерактивні опитування. Викладачі можуть створювати власні вікторини або використовувати готові шаблони з бібліотеки платформи.

Особливістю Kahoot є акцент на соціальну взаємодію. Групова динаміка, конкуренція та інтерактивний підхід сприяють активному залученню учнів, навіть під час складних тем. Крім того, платформа пропонує функції для дистанційного використання, що робить її актуальною для онлайн-навчання.

Серед недоліків Kahoot варто зазначити її обмежений функціонал для детального аналізу результатів і перевірки знань у серйозних академічних контекстах. Крім того, платформа більше підходить для коротких ігрових сесій, ніж для комплексних оцінювань.

Загалом, Kahoot — це відмінний інструмент для створення живого й інтерактивного освітнього досвіду. Її використання сприяє підвищенню мотивації учнів, полегшує засвоєння матеріалу та додає елемент гри в навчальний процес.

Висновок

Онлайн-платформи для тестування відіграють важливу роль у сучасному освітньому процесі, забезпечуючи зручність, гнучкість і ефективність у контролі знань. Проаналізовані платформи демонструють різні підходи до організації тестувань, адаптовані до специфічних потреб користувачів.

Google Forms виділяється своєю простотою, доступністю та інтеграцією з іншими сервісами Google, що робить її ідеальною для швидкого створення базових тестів. Moodle, у свою чергу, пропонує розширений функціонал і можливості для комплексного управління навчанням, що підходить для великих освітніх закладів. Quizizz поєднує в собі тестування та гейміфікацію, сприяючи підвищенню мотивації та зацікавленості учнів. Kahoot акцентує увагу на інтерактивності та груповій динаміці, роблячи навчальний процес веселим і динамічним.

Кожна з платформ має свої переваги й обмеження, що підкреслює важливість вибору відповідного інструменту залежно від цілей та

особливостей навчального середовища. Завдяки онлайн-платформам, освітяни мають змогу забезпечити якісний контроль знань, інтегруючи сучасні технології у свої методи роботи.

РОЗДІЛ 2. ІНФОРМАЦІЙНА ПІДТРИМКА ФУНКЦІОНУВАННЯ ОНЛАЙН ТЕСТУВАННЯ

2.1. Загальний статистичний аналіз результатів учасників НМТ

Національний мультипредметний тест (НМТ) є ключовим інструментом оцінювання рівня знань випускників закладів загальної середньої освіти, необхідних для вступу до закладів вищої освіти. Аналіз результатів НМТ дозволяє отримати цінну інформацію про рівень підготовки учнів, регіональні особливості, ефективність навчального процесу та вплив різних чинників на успішність тестування.

У цьому підрозділі буде розглянуто статистичні дані, представлені у звіті за результатами НМТ 2024 року [17], які охоплюють різні аспекти участі випускників у тестуванні: розподіл за регіонами, типами закладів освіти, рівнями успішності та іншими показниками. Представлені діаграми та їх аналіз допоможуть виявити основні тенденції та проблемні аспекти, що потребують уваги для вдосконалення освітнього процесу.

Діаграма, що зображена на рис. 2.1 відображає розподіл учасників НМТ 2024 року [17, с. 50], які є випускниками поточного року, за регіонами розташування закладу освіти, де вони здобули повну загальну середню освіту (ПЗСО). Найбільший відсоток учасників зареєстровано в місті Київ (10,9%), що свідчить про значну концентрацію освітніх закладів та високий рівень участі у столиці. Наступними за кількістю є Львівська (8,4%) та Дніпропетровська області (7,9%), які також демонструють високі показники через розвинену інфраструктуру освіти.

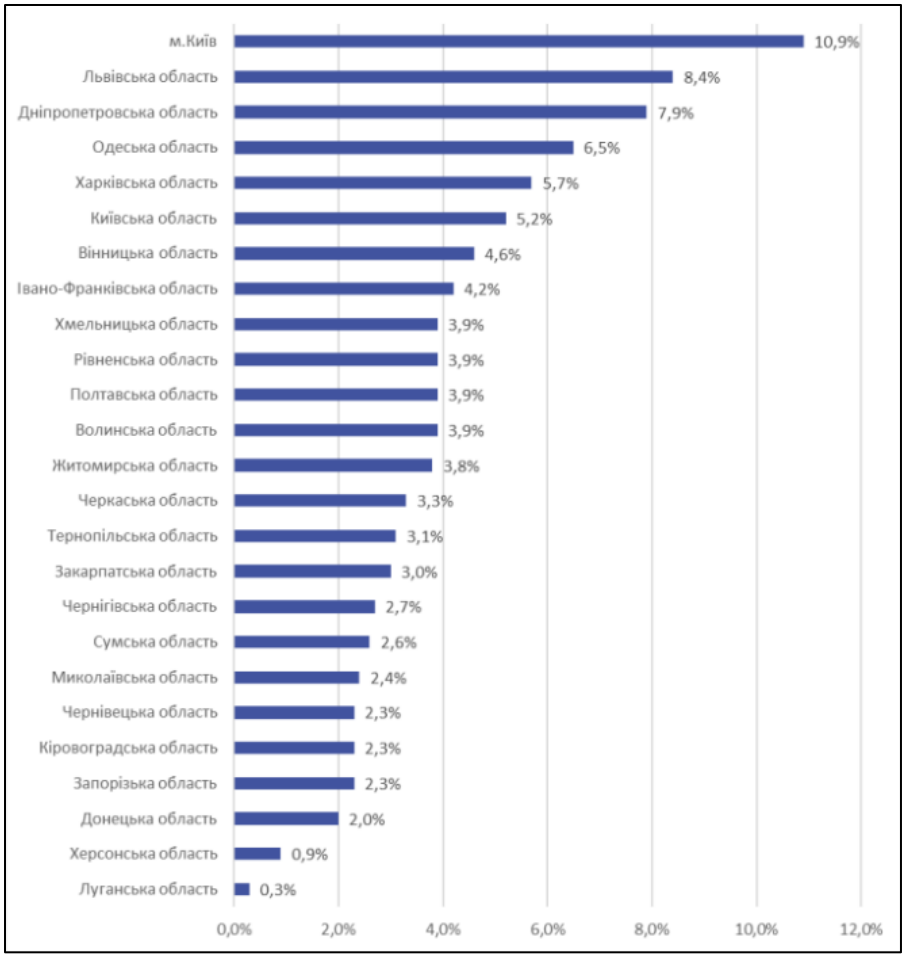

Рис. 2.1. Розподіл учасників НМТ за регіоном розташування закладу освіти

Водночас найменший відсоток учасників зафіксовано у Луганській області (0,3%) та Херсонській області (0,9%), що може бути зумовлено складною соціально-економічною ситуацією та військовими діями. Такий розподіл дозволяє аналізувати регіональні відмінності у доступності та участі в освітніх процесах.

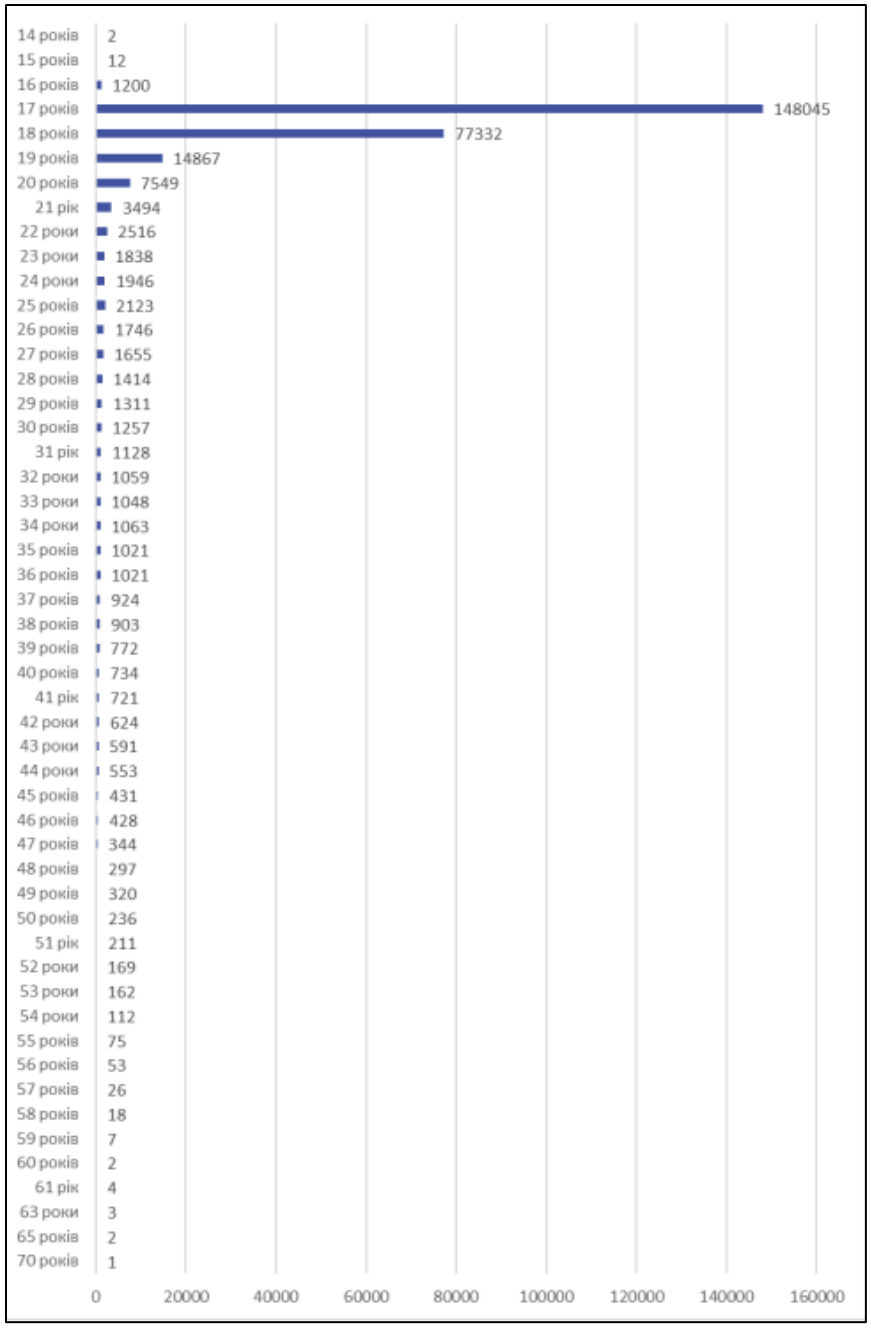

Рис. 2.2. Розподіл учасників НМТ за віком

На діаграмі (рис. 2.2) представлено розподіл учасників Національного мультипредметного тесту (НМТ) за віком [17, с. 48]. Найбільша частка учасників припадає на випускників поточного року, яким виповнилося 17 років – це 148 045 осіб. Другою за чисельністю віковою групою є 18-річні учасники,

кількість яких становить 77 332 осіб. Дещо менше представлено 19-річних – 14 867 осіб.

Значна кількість учасників також представлена віком 20-22 роки, однак після цієї вікової групи спостерігається поступове зниження чисельності учасників із кожним роком. Наприклад, 25-річних осіб лише 2 123, тоді як кількість 30-річних становить 1 257 осіб. Особи старшого віку (понад 40 років) також брали участь у тестуванні, однак їхня кількість значно зменшується – від 621 учасника у віці 42 років до кількох одиниць у віковій групі 60 років і старше.

Ця діаграма відображає домінування молоді серед учасників НМТ, що логічно пояснюється тим, що тест орієнтований на випускників закладів освіти, які планують вступати до закладів вищої освіти. Присутність старших учасників свідчить про бажання деяких осіб старшого віку здобути нову освіту або перекваліфікуватися.

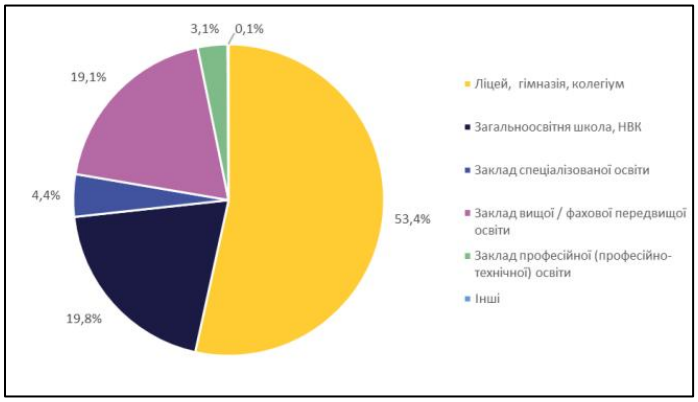

Рис. 2.3. Розподіл учасників НМТ – випускників поточного року за типом закладу освіти, у якому вони здобули ПЗСО

На цій круговій діаграмі (рис. 2.3) зображено розподіл учасників НМТ – випускників поточного року за типом закладу освіти, у якому вони здобули повну загальну середню освіту (ПЗСО) [17, с. 52]. Найбільший відсоток

випускників (53,4%) здобули освіту в ліцеях, гімназіях і колегіумах, що демонструє високу популярність таких типів закладів освіти серед учнів.

Другою за чисельністю групою є випускники загальноосвітніх шкіл та навчально-виховних комплексів (НВК) – їх частка становить 19,8%. Це свідчить про значний внесок традиційних шкіл у забезпечення освіти для випускників.

Заклади вищої або фахової передвищої освіти займають третє місце за кількістю випускників – 19,1%, що вказує на поширеність отримання середньої освіти в таких закладах для тих, хто обирає професійний або технічний напрям.

Менше випускників здобули ПЗСО у закладах спеціалізованої освіти (4,4%) та закладах професійної (професійно-технічної) освіти (3,1%). Ці заклади обирають переважно учні, які планують продовжувати освіту у вузькоспеціалізованих сферах.

Найменша частка випускників (0,1%) належить до категорії «Інші», що може включати рідкісні або нетипові заклади освіти. Цей розподіл демонструє загальну структуру системи освіти в Україні та пріоритети сучасних випускників у виборі навчальних закладів.

2.2. Дослідження результатів учасників НМТ у розрізі навчальних предметів

Розглянемо також розподіл учасників НМТ за сумою результатів за всіма субтестами, виконаними учасником, зображених на рис. 2.4 [17, с. 53].

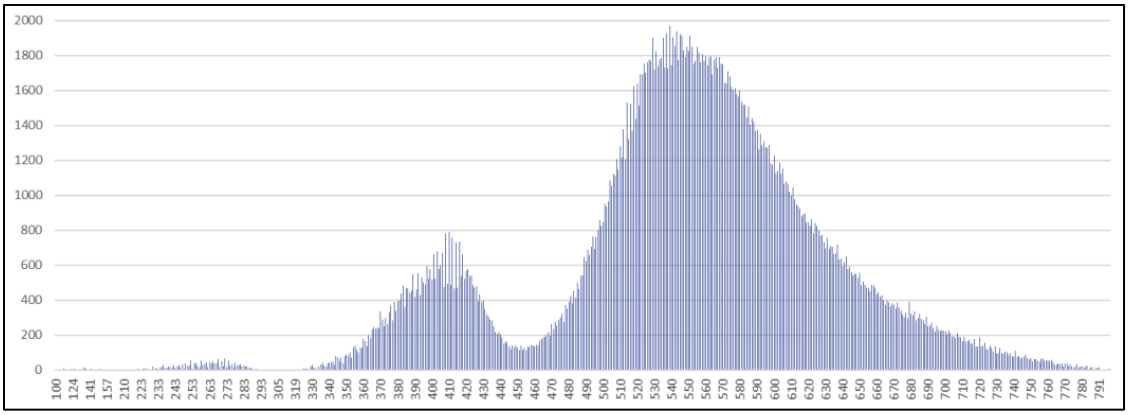

Рис. 2.4. Розподіл учасників НМТ за сумою результатів за всіма субтестами, виконаними учасником.

Діаграма ілюструє розподіл учасників НМТ 2024 року за сумою балів, отриманих за всі субтести, які виконувалися. Загалом на графіку показані результати 283 358 учасників, які набрали щонайменше 100 балів за шкалою 100-200 з одного або більше навчальних предметів. Максимальна кількість балів, яку міг отримати учасник за чотири субтести, становила 800. Проте на розподіл балів впливає той факт, що не всі учасники проходили чотири субтести, а також частина з них не подолала поріг із певних предметів. Розподіл має дві яскраво виражені моди, що свідчить про різницю у підготовці учасників та вплив різних факторів, таких як складність завдань і кількість складених субтестів.

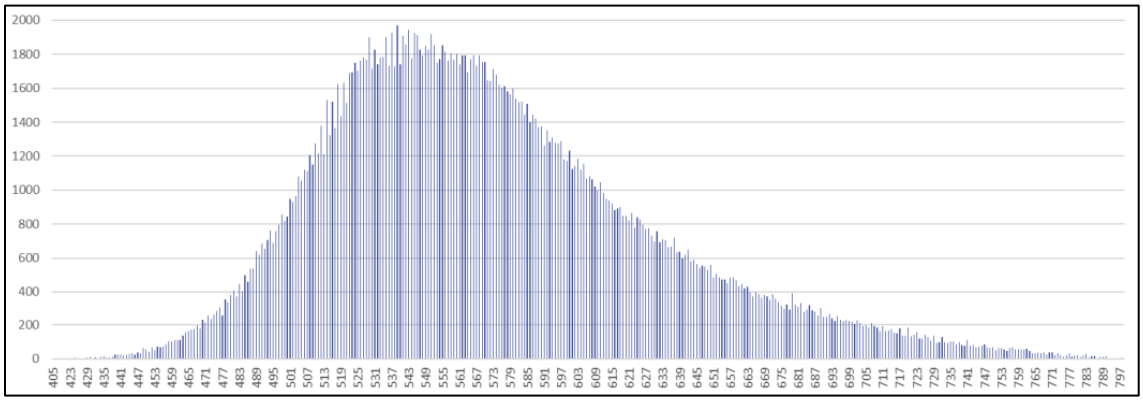

Рис. 2.5. Розподіл учасників НМТ за сумою результатів за всіма чотирма субтестами, виконаними учасником

Якщо ж роглядати випадок, коли розподіл учасників за сумою результатів уже містить всі чотири субтести (рис. 2.5) [17, с. 54], то можна побачити рівномірне зменшення кількості учасників із вищими балами, що свідчить про складність виконання завдань та обмежену кількість високих результатів. На діаграмі зображено розподіл учасників НМТ 2024 року, які пройшли тестування із чотирьох предметів, за сумою отриманих балів. Загальна кількість таких учасників становить 242 553 особи, серед яких більшість (194 830) – це випускники поточного року, які завершили здобуття повної загальної середньої освіти (ПЗСО) у 2024 році. Максимальна сума балів за чотири субтести могла досягати 800, однак розподіл демонструє пік у межах 500-550 балів, що відображає середній рівень підготовки учасників. Лише ті учасники, які отримали результат за всі чотири субтести, могли претендувати на вступ до закладів вищої освіти у 2024 році.

РОЗДІЛ 3. ІНТЕЛЕКТУАЛЬНА СИСТЕМА ОНЛАЙН ТЕСТУВАННЯ

3.1. Програмна реалізація проекту

У цьому розділі описано процес розробки програмного забезпечення для системи онлайн-тестування. Програмна реалізація охоплює створення архітектури проєкту, побудову бази даних, розробку серверної частини на основі .NET Core 8 із використанням Entity Framework Core, а також клієнтської частини, реалізованої на React із застосуванням TypeScript та Redux.

Розробка програмного забезпечення виконувалася з урахуванням принципів модульності, масштабованості та зручності підтримки. Усі основні компоненти системи інтегруються в єдину інфраструктуру для забезпечення безперервної роботи сервісу. Особлива увага приділялася захисту даних, відповідності вимогам сучасних технологій та зручності використання.

Основними етапами програмної реалізації були:

- проєктування та реалізація схеми бази даних;
- створення API для взаємодії клієнтської та серверної частин;
- розробка клієнтського інтерфейсу з використанням сучасних технологій веб-розробки;
- впровадження системи авторизації та аутентифікації користувачів із використанням токенів;
- інтеграція модулів для управління тестуванням і зберігання результатів;
- реалізація фонових завдань за допомогою Hangfire.

Цей підрозділ детально описує кожен із перелічених етапів, підкреслюючи технічні рішення, що забезпечують реалізацію функціоналу системи онлайн-тестування.

Створення схеми бази даних

Для реалізації проєкту системи онлайн-тестування було розроблено базу даних із використанням MS SQL Server. Він забезпечує високу швидкість обробки запитів, підтримує великий обсяг даних і транзакційність, що робить його оптимальним вибором для сучасних проєктів, включаючи онлайн-тестування.

Схема бази даних (рис. 3.1) охоплює основні таблиці, необхідні для збереження інформації про користувачів, ролі, тести, результати тестування, а також підозрілу активність (таблиця 3.1). Кожна таблиця структурована з урахуванням вимог до цілісності даних, використання унікальних ідентифікаторів (Id) і підтримки зв’язків між таблицями через зовнішні ключі. Поля таблиць містять необхідну інформацію для забезпечення функціонування системи, а також дати створення та оновлення записів для зручності управління даними.

Ця структура дозволяє ефективно управляти даними, забезпечує масштабованість і адаптивність системи до подальших удосконалень.

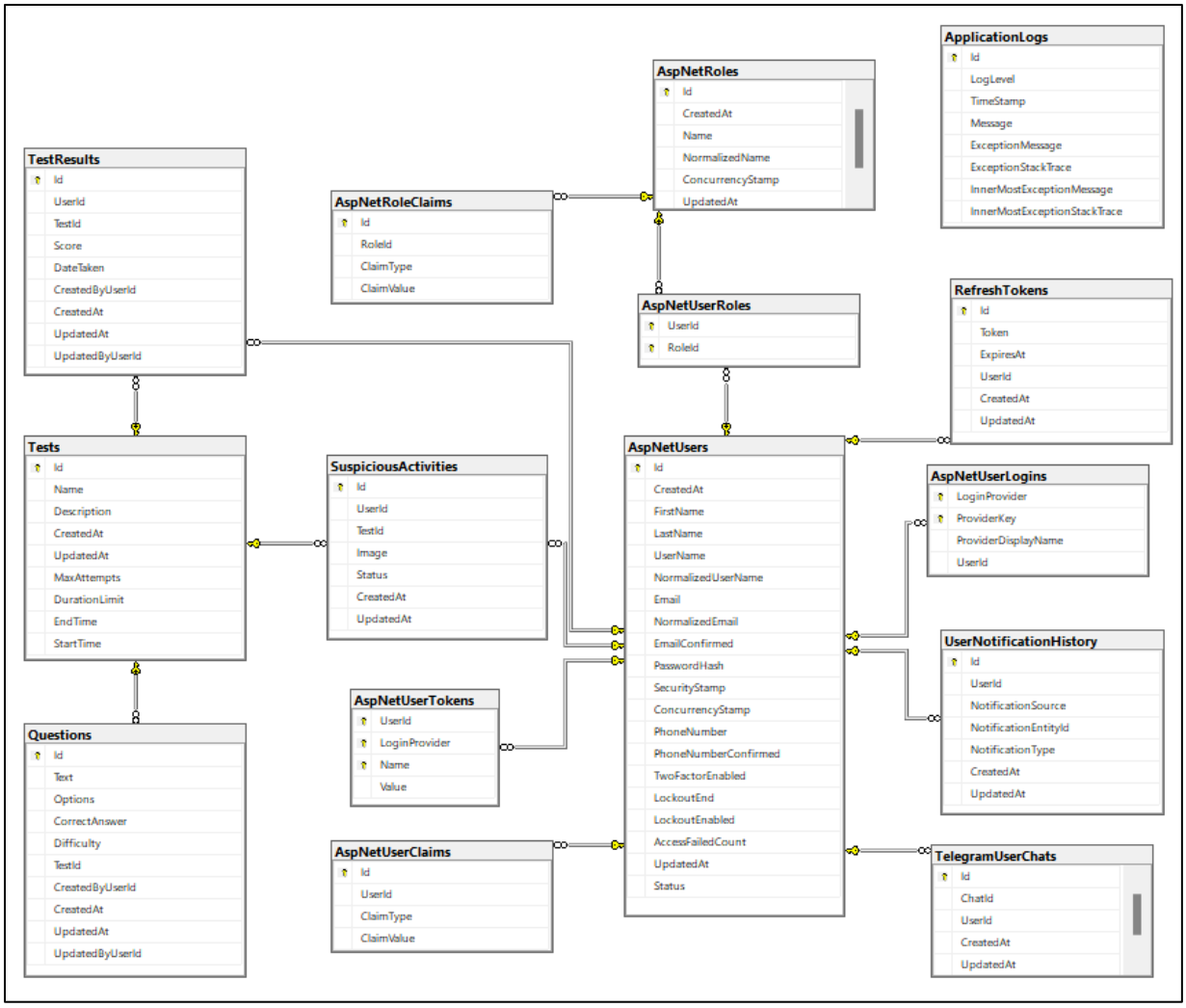

Рис. 3.1. Схема бази даних застосунку

Таблиця 3.1

Таблиця структури бази даних

Назва таблиці	**Призначення таблиці**	**Тип зв'язків з іншими таблицями**	**Поля, їх призначення та тип даних**
AspNetUsers	Зберігання інформації про користувачів системи.	Один-до-багатьох з AspNetUserRoles, TestResults, SuspiciousActivities, TelegramUserChats, UserNotificationHistory	Id – унікальний ідентифікатор користувача (uniqueidentifier) FirstName – ім’я користувача (nvarchar); LastName – прізвище користувача (nvarchar); UserName – логін користувача (nvarchar);

			NormalizedUserName – нормалізований логін користувача (nvarchar); Email – адреса електронної пошти (nvarchar); NormalizedEmail – нормалізована адреса електронної пошти (nvarchar); EmailConfirmed – підтвердження електронної пошти (bit); PasswordHash – хеш від паролю (nvarchar); SecurityStamp – штамп безпеки (nvarchar); ConcurrencyStamp – штамп для відстеження змін (nvarchar); PhoneNumber – номер телефону (nvarchar); PhoneNumberConfirmed – підтвердження номера телефону (bit); TwoFactorEnabled – увімкнення двофакторної аутентифікації (bit); LockoutEnd – дата закінчення блокування (datetimeoffset); LockoutEnabled – увімкнення блокування (bit); AccessFailedCount – кількість невдалих спроб входу (int); Status – статус користувача (int); CreatedAt – дата створення (datetime2); UpdatedAt – дата оновлення (datetime2).
AspNetRoles	Зберігання інформації про ролі користувачів.	Один-до-багатьох з AspNetUserRoles	Id – унікальний ідентифікатор ролі (uniqueidentifier); Name – назва ролі (nvarchar); NormalizedName – нормалізована назва ролі (nvarchar); ConcurrencyStamp – штамп для відстеження змін (nvarchar); CreatedAt – дата створення (datetime2);

			UpdatedAt – дата оновлення (datetime2).
AspNetUserRoles	Проміжна таблиця для реалізації зв’язку багато-до-багатьої між таблицями AspNetUsers та AspNetRoles	Багато-до-одного з AspNetUsers і AspNetRoles	UserId – унікальний ідентифікатор користувача (uniqueidentifier); RoleId – унікальний ідентифікатор ролі (uniqueidentifier).
Tests	Зберігання інформації про тести.	Один-до-багатьох з Questions, TestResults, SuspiciousActivities	Id – унікальний ідентифікатор тесту (uniqueidentifier); Name – назва тесту (nvarchar) Description – опис тесту (nvarchar); MaxAttempts – максимальна кількість спроб (int); DurationLimit – ліміт часу на виконання (time); StartTime – час початку тесту (datetime2); EndTime – час завершення тесту (datetime2); CreatedAt – дата створення (datetime2); UpdatedAt – дата оновлення (datetime2).
Questions	Зберігання запитань тестів.	Багато-до-одного з Tests	Id – унікальний ідентифікатор запитання (uniqueidentifier); Text – текст запитання (nvarchar); Options – варіанти відповідей (nvarchar); CorrectAnswer – правильна відповідь (nvarchar); Difficulty – складність запитання (int); TestId – ідентифікатор тесту (uniqueidentifier); CreatedByUserId – ідентифікатор користувача, який створив запитання (uniqueidentifier);

			CreatedAt – дата створення (datetime2); UpdatedAt – дата оновлення (datetime2); UpdatedByUserId – ідентифікатор користувача, який оновив запитання (uniqueidentifier).
TestResults	Зберігання результатів тестів користувачів.	Багато-до-одного з AspNetUsers і Tests	Id – унікальний ідентифікатор результату (uniqueidentifier); UserId – ідентифікатор користувача (uniqueidentifier); TestId – ідентифікатор тесту (uniqueidentifier); Score – результат тесту (float); DateTaken – дата проходження тесту (datetime2); CreatedByUserId – ідентифікатор користувача, який створив запис (uniqueidentifier); CreatedAt – дата створення (datetime2); UpdatedAt – дата оновлення (datetime2); UpdatedByUserId – ідентифікатор користувача, який оновив запис (uniqueidentifier).
SuspiciousActivities	Виявлення підозрілої активності під час тестування.	Багато-до-одного з AspNetUsers і Tests	Id – унікальний ідентифікатор запису (uniqueidentifier); UserId – ідентифікатор користувача (uniqueidentifier); TestId – ідентифікатор тесту (uniqueidentifier); Image – зображення підозрілої активності (nvarchar); Status – статус активності (nvarchar); CreatedAt – дата створення (datetime2); UpdatedAt – дата оновлення (datetime2).

UserNotification History	Зберігання історії повідомлень користувачів.	Багато-до-одного з AspNetUsers	Id – унікальний ідентифікатор запису (uniqueidentifier); UserId – ідентифікатор користувача (uniqueidentifier); NotificationSource – джерело повідомлення (int); NotificationEntityId – ідентифікатор сутності повідомлення (uniqueidentifier); NotificationType – тип повідомлення (int); CreatedAt – дата створення (datetime2); UpdatedAt – дата оновлення (datetime2).
TelegramUserChats	Зберігання зв’язків між користувачами та чатами Telegram.	Багато-до-одного з AspNetUsers	Id – унікальний ідентифікатор запису (uniqueidentifier); ChatId – ідентифікатор чату в Telegram (bigint); UserId – ідентифікатор користувача (uniqueidentifier); CreatedAt – дата створення (datetime2); UpdatedAt – дата оновлення (datetime2).
RefreshTokens	Зберігання токенів-оновлення для можливості оновлення токенів автентифікації.	Багато-до-одного з AspNetUsers	Id – унікальний ідентифікатор токена (uniqueidentifier); Token – токен авторизації (nvarchar); ExpiresAt – дата закінчення дії токена (datetime2); UserId – ідентифікатор користувача (uniqueidentifier); CreatedAt – дата створення (datetime2); UpdatedAt – дата оновлення (datetime2).

Створення серверної частини

Серверна частина проєкту розроблена з використанням платформи .NET Core та Python Flask. Основна мета серверної частини — забезпечити ефективну взаємодію між клієнтськими додатками та базою даних, обробку

бізнес-логіки, а також інтеграцію інструментів для аналізу поведінки користувачів.

Архітектура побудована за принципами мікросервісної структури, що сприяє гнучкості розробки, тестування та подальшого масштабування системи. Серверна частина включає:

.NET мікросервіс для управління базою даних, автентифікації користувачів, створення та виконання тестів.

Python Flask сервіс для аналізу зображень на наявність підозрілої активності (розпізнавання обличчя та руху очей).

Обидва сервіси забезпечують обробку відповідних даних та інтегруються між собою через HTTP-запити.

Розробка .NET мікросервісу

.NET мікросервіс реалізує наступні основні функції:

- Управління базою даних через ORM EF Core (підхід Code First).
- Автентифікацію та авторизацію користувачів за допомогою JWT.
- Обробку бізнес-логіки, включаючи створення тестів, управління ролями користувачів, запис результатів тестування.
- Взаємодію з Python-сервісом для отримання інформації про підозрілу активність.

Основні компоненти включають:

- TestSystem.Common: бібліотека загальних компонентів, що включає константи, утиліти, винятки, сервіси часу тощо.
- TestSystem.Core: бібліотека бізнес-логіки з сервісами, такими як AuthService, JwtService тощо.
- TestSystem.Data: реалізація бази даних, включаючи моделі та конфігурацію DbContext.

- TestSystem.WebApi: ASP.NET Core Web API застосунок для роботи з API.

Приклад методу реалізації генерації токенів для автентифікації користувачів із врахуванням їхніх ролей сервісом JwtService зображено нижче:

```
public string GenerateToken(string userId, string[] userRoles, int tokenLifetimeInMinutes)
    {
```

Генерація підпису токена на основі секретного ключа реалізується за допомогою коду:

```
                            var authSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_jwtConfiguration.JwtKey));
        var signingCredentials = new SigningCredentials(authSigningKey, SecurityAlgorithm);
```

Після чого створюється список клеймів (параметрів-вимог), куди входить ідентифікатор користувача та його ролі:

```
   var tokenClaims = new List<Claim>()
        {
            new(Claims.Id, userId)
        };

        foreach (var role in userRoles)
        {
            tokenClaims.Add(new Claim(ClaimTypes.Role, role));
        }

        var descriptor = new SecurityTokenDescriptor
        {
            Subject = new ClaimsIdentity(tokenClaims),
                                                    Expires = currentDateTime.UtcNow.AddMinutes(tokenLifetimeInMinutes),
            SigningCredentials = signingCredentials
        };
```

```
            var tokenExpirationTime = currentDateTime.UtcNow +
TimeSpan.FromMinutes(_jwtConfiguration.AccessTokenLifeTimeInMinutes);

        var token = new JwtSecurityToken(
           expires: tokenExpirationTime,
           claims: tokenClaims,
           audience: _jwtConfiguration.JwtAudience,
           issuer: _jwtConfiguration.JwtIssuer,
           signingCredentials: new SigningCredentials(
               key: authSigningKey,
               algorithm: SecurityAlgorithm)
       );
```

Та повернення згенерованого токена сервісом:

```
            return new JwtSecurityTokenHandler().WriteToken(token);
  }
```

Цей метод GenerateToken сервісу JwtService викликається у AuthService – сервісі, що відповідає за реєстрацію та аутентифікацію користувачів використовуючи внутрішню аутентифікацію, або ж, зовнішню, як у випадку із Google аутентифікацією. Розглянемо метод реєстрації користувача:

```
public async Task<UserDto> RegisterUser(RegisterUserRequest
registerUserDto, CancellationToken cancellationToken)
    {
```

Перетворення моделі даних користувача із запиту контролера до моделі, що зберігається на сервері відбувається за допомогою коду:

```
        var user = mapper.Map<UserEntity>(registerUserDto);
        user.Email = registerUserDto.UserName;
```

Створення користувача та збереження у базі даних:

```
   var result = await userManager.CreateAsync(user,
registerUserDto.Password);
```

Перевіряємо, чи користувача успішно створено:

```
        if (!result.Succeeded)
        {
            throw ThrowHelper.GetBusinessLogicException("User with same
user name already exists");
        }
```

Якщо користувач реєструється як викладач:

```
    if (registerUserDto.IsTeacher)
        {
```

Додаємо відповідну роль:

```
            await                        userManager.AddToRoleAsync(user,
AppConstant.Roles.Teacher);
        }
        else
        {
```

У інакшому випадку автоматично активуємо користувача, оскільки для викладача потрібне підтвердження активації адміністратором:

```
            user.Status = UserStatus.Enabled;
```

Та додаємо роль студента:

```
            await                        userManager.AddToRoleAsync(user,
AppConstant.Roles.Student);
        }
```

У результаті повертаємо перетвореного у нову модель даних користувача:

```
    return mapper.Map<UserDto>(user);
     }
```

Також розглянемо контролер, через який мікросервіс дозволяє викликати через HTTP запит ендпоінт реєстрації.

Атрибут, що дозволяє виклик неавтентифікованим користувачам:

```
  [AllowAnonymous]
```

Вказуємо тип HTTP виклику – POST та назву ендпоінту “register”:

```
    [HttpPost("register")]
```

```
    public async Task<ActionResult<UserDto>> Register([FromBody]
```

Із тіла запиту отримуємо модель RegisterUserRequest та токен відміни:

```
RegisterUserRequest registerUserDto, CancellationToken
cancellationToken)
    {
```

Викликаємо відповідно AuthService для реєстрації користувача:

```
        return await authService.RegisterUser(registerUserDto,
cancellationToken);
    }
```

Розробка Python мікросервісу на Flask

Python Flask застосунок відповідає за розпізнавання обличчя та аналіз руху очей для виявлення підозрілої активності під час тестування.

Функціонал включає:

- Обробку зображень, отриманих із клієнтського додатка.
- Використання бібліотек OpenCV для розпізнавання облич і очей.
- Визначення, чи є поведінка користувача підозрілою.

Розглянемо основні частини коду цього застосунку:

Підключення класифікатору для розпізнавання обличчя з haar каскадів відбувається за допомогою наступного коду:

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
```

Далі відбувається підключення класифікатору для розпізнавання очей з haar каскадів:

```
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_eye.xml')
def get_eye_center(eye):
```

Перетворення зображення ока у відтінки сірого для спрощення обробки:

```
    gray_eye = cv2.cvtColor(eye, cv2.COLOR_BGR2GRAY)
```

Застосування розмиття Гауса для зменшення шуму та згладжування зображення:

```
gray_eye = cv2.GaussianBlur(gray_eye, (7, 7), 0)
```

Бінаризація зображення: виділення об'єктів за пороговим значенням:

```
_, threshold_eye = cv2.threshold(gray_eye, 30, 255, cv2.THRESH_BINARY)
```

Пошук контурів на бінаризованому зображенні:

```
contours, _ = cv2.findContours(threshold_eye, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

Якщо знайдено контури, обробляємо найбільший:

```
if len(contours) > 0:
```

Вибір контуру з найбільшою площею:

```
largest_contour = max(contours, key=cv2.contourArea)
```

Розрахунок моментів контуру для визначення центру:

```
moments = cv2.moments(largest_contour)
```

Перевірка, чи не є площа моменту нульовою (уникнення ділення на 0):

```
if moments['m00'] != 0:
```

Визначення координат центру мас (cx, cy):

```
cx = int(moments['m10'] / moments['m00'])
cy = int(moments['m01'] / moments['m00'])
```

Повернення координат центру

```
return (cx, cy
```

Якщо контури не знайдені або центр не визначено, повертаємо None

```
return None
```

Та наявний ендпоінт, котрий отримує зображення, та повертає булевий результат в залежності від того, чи на даному зображенні виявлено підозрілу активність:

```
@app.route('/suspicious', methods=['POST'])
def get_suspicious_status():
```

Перевіряємо, чи файл із зображенням присутній у запиті:

```
if 'image' not in request.files:
    return jsonify({"error": "No image part in the request"}), 400

file = request.files['image']
```

Перевіряємо, чи файл був вибраний (не порожня назва):

```
if file.filename == '':
    return jsonify({"error": "No selected file"}), 400

try:
```

Зчитуємо байтові дані з файлу в масив NumPy:

```
    npimg = np.frombuffer(file.read(), np.uint8)
```

Перевіряємо, чи буфер зображення порожній:

```
    if npimg.size == 0:
        print("Received empty image buffer")
        return jsonify({"error": "Empty image buffer"}), 400
```

Декодуємо зображення у форматі BGR:

```
    img = cv2.imdecode(npimg, cv2.IMREAD_COLOR)
```

Перевіряємо, чи зображення вдалося декодувати:

```
    if img is None:
        print("Image decoding failed")
        return jsonify({"error": "Image decoding failed"}), 400
```

Викликаємо функцію для перевірки підозрілої активності:

```
    suspicious = detect_suspicious_activity(img)
```

Повертаємо результат у форматі JSON:

```
    return jsonify({"suspicious": suspicious})
except Exception as e:
```

Обробляємо будь-які винятки та повертаємо помилку:

```
print(f"Exception: {e}")
    return jsonify({"error": str(e)}), 500
```

Запускаємо Flask-додаток на порту 5050:

```
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5050)
```

Розробка фронтенд частини на React

Фронтенд-частина системи онлайн-тестування розроблена за допомогою React, TypeScript, і Redux. Основна мета цієї частини — забезпечити інтуїтивно зрозумілий та ефективний інтерфейс для користувачів, а також інтеграцію з серверною частиною для отримання та обробки даних.

Проєкт має чітко структуровану архітектуру, яка базується на розподілі відповідальностей між різними папками. Нижче наведено детальний опис основних директорій у проєкті:

- api – містить логіку для взаємодії з сервером (API-запити). Тут знаходяться всі функції або класи для роботи з API через бібліотеки, такі як Axios.
- assets – зберігає статичні ресурси, які використовуються у фронтенді, такі як зображення, шрифти, або файли стилів.
- common – містить загальні утиліти, константи або типи, які можуть використовуватися у багатьох частинах проєкту
- components – містить реюзабельні React-компоненти, які можуть використовуватися на різних сторінках
- hooks – містить кастомні React-хуки для абстрагування логіки
- pages – містить основні сторінки додатка, які об'єднують компоненти для формування повноцінних екранів
- store – містить Redux-сховище та логіку управління станом програми
- utils – містить утилітарні функції для обробки даних, роботи з датами, форматування тощо.

Така структура має переваги, оскільки код розділений на логічні блоки, що полегшує розуміння і тестування; загальні функції, компоненти та сервіси можна легко повторно використовувати; додаток легко масштабувати, додаючи нові функціональні частини; зрозуміла структура дозволяє швидко орієнтуватися в коді навіть новим розробникам. Ця структура є зразком ефективної організації проєкту з використанням сучасних підходів до розробки на React і TypeScript.

3.2 Функціональні можливості системи

Система онлайн-тестування створена з метою забезпечення ефективного, зручного та надійного процесу контролю знань для освітніх установ, викладачів та студентів. Вона пропонує широкий набір функціональних можливостей, які сприяють організації тестування, зберіганню результатів, аналізу даних та підвищенню прозорості оцінювання.

Головними цілями функціоналу забезпечення зручності користувачів через інтуїтивно зрозумілий інтерфейс, реалізація гнучких механізмів для створення, редагування та управління тестами, забезпечення прозорості та точності результатів шляхом використання автоматизованих алгоритмів і виявлення підозрілих дій під час тестування за допомогою модулів з комп'ютерним зором.

Цей підрозділ розглядає ключові функціональні можливості системи, включаючи авторизацію користувачів, управління тестами, моніторинг активності під час проходження тестів, а також аналіз результатів.

Користувача зустрічає сторінка «Про нас» (рис. 3.2), де користувач може ознайомитись із системою та функціями, які доступні для нього.

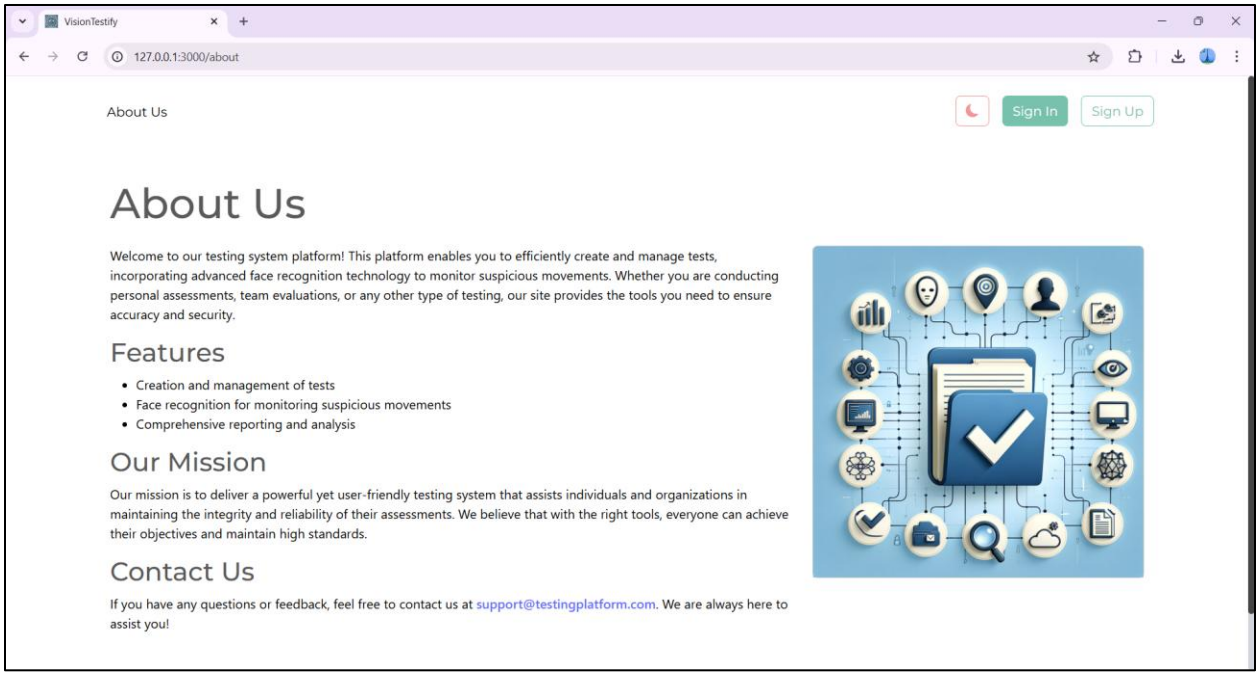

Рис. 3.2. Початкова сторінка для користувачів

Оскільки система містить 3 ролі – студента, викладача та адміністратора, спочатку зареєструємось як студент та поглянемо на функціонал, що доступний саме для цієї ролі. Для цього натиснемо кнопку Sign Up зверху праворуч на екрані.

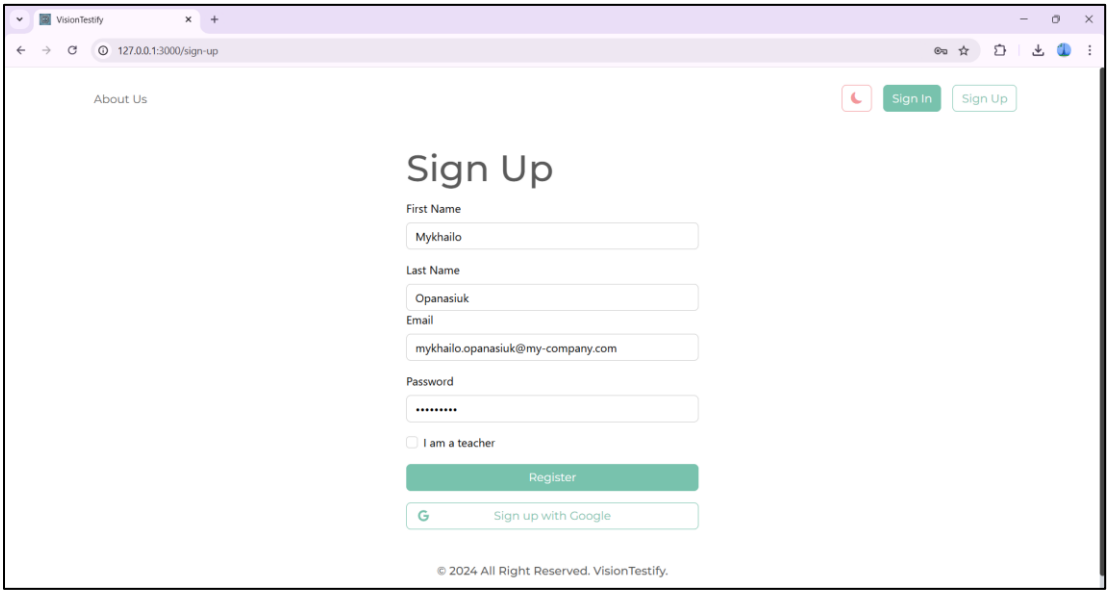

Рис. 3.3. Форма реєстрації

Для користувача доступні два можливості реєстрації – через форму та через Google акаунт. Спочатку зареєструємо через форму, попередньо ввівши усі поля та натиснемо кнопку Register. (рис. 3.3) Після реєстрації користувача перенаправить на сторінку логіну (рис. 3.4), де введемо попередні логін та пароль, після чого натиснемо Sign In.

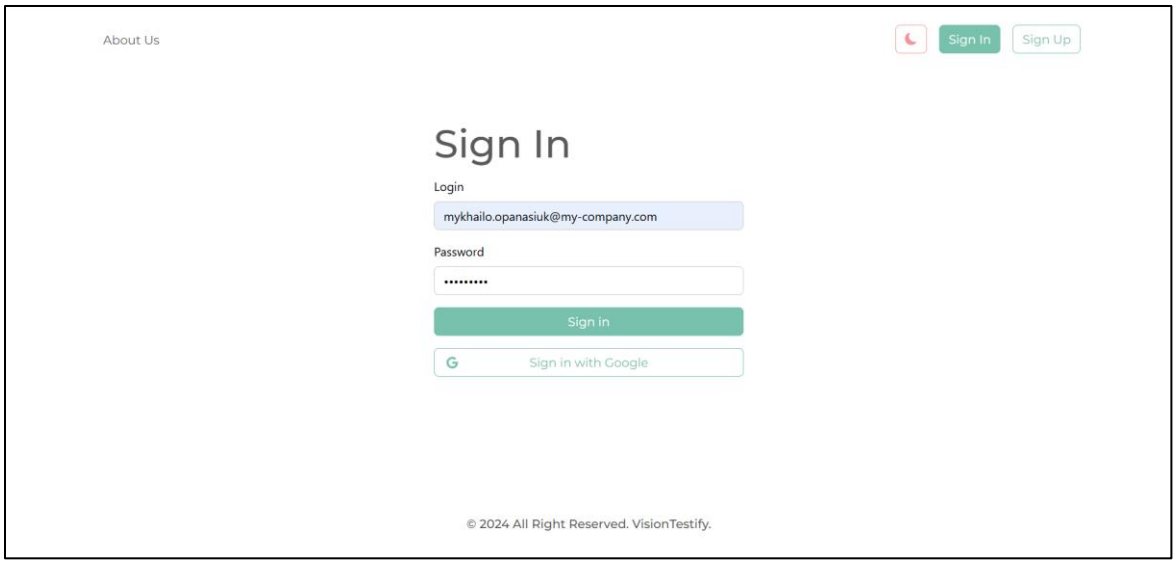

Рис. 3.4. Форма входу в систему

Після входу в систему користувача зустрічає форма з тестами, яка у стані завантаження, де відображаються картки перед завантаженням реальних тестів (рис. 3.5) та реальні тести (рис. 3.6) коли ми отримали відповідь від серверу.

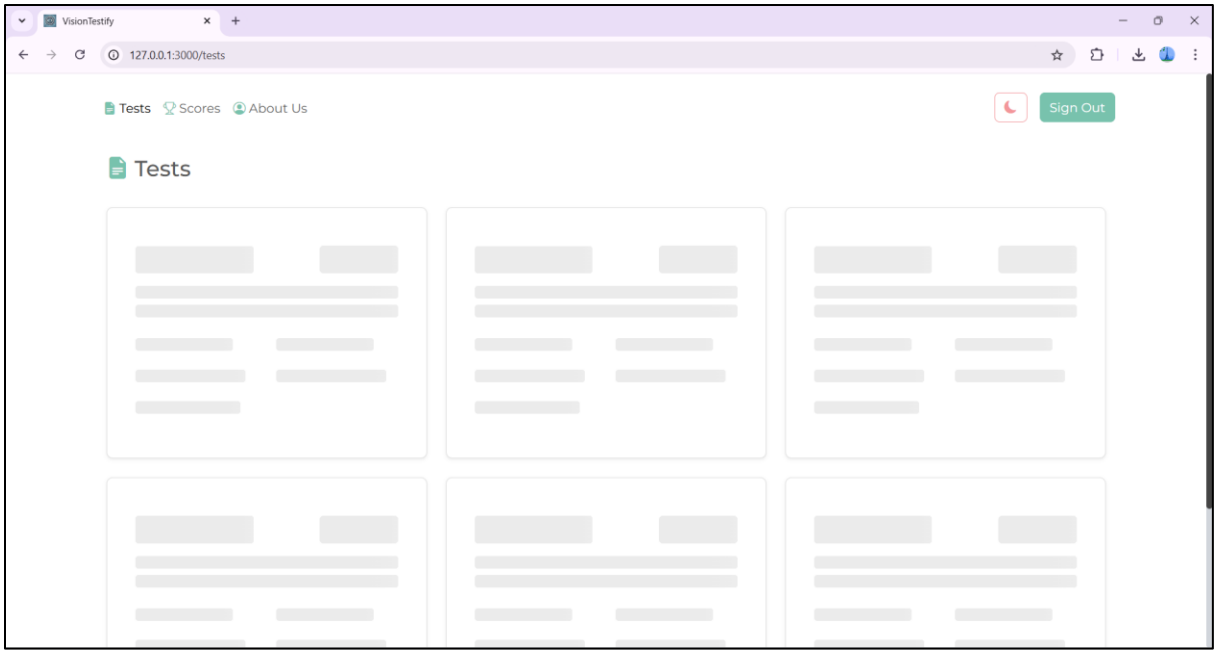

Рис. 3.5. Сторінка з тестами, поки тести підвантажуються з серверу

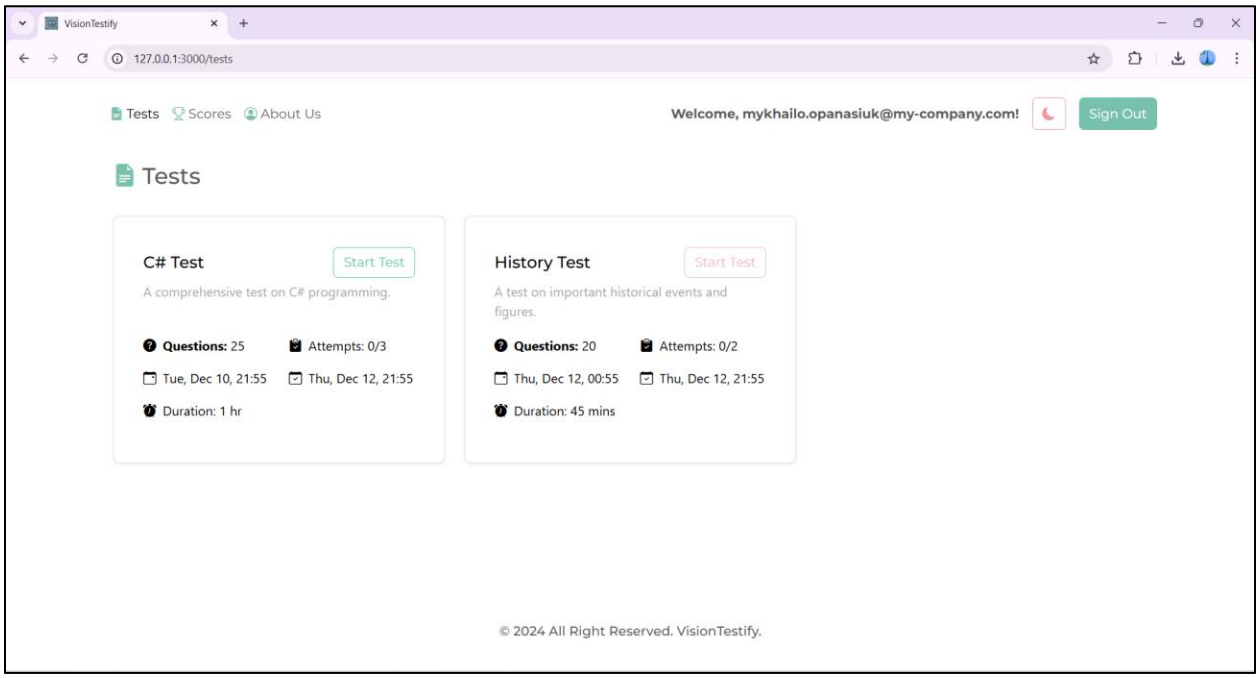

Рис. 3.6. Сторінка з тестами після підвантаження тестів із серверу

Такий вид підвантаження дозволяє користувачу інтуїтивно зрозуміти, який інтерфейс та розмір карток на нього очікують після підвантаження даних. Також не менш важливим аспектом є адаптивність сайту під будь-які девайси, наприклад під телефони, як це зображено на рис. 3.7.

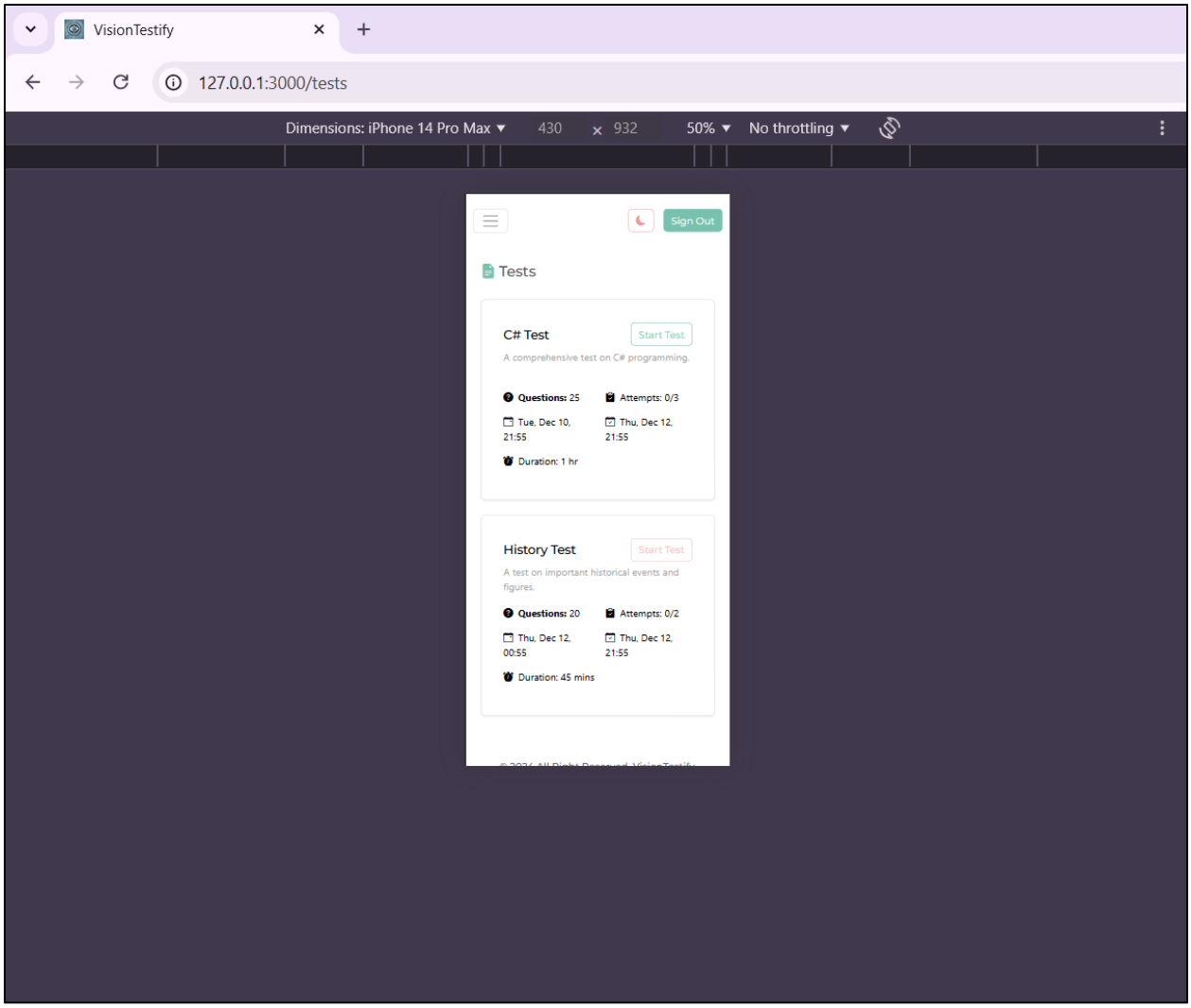

Рис. 3.7. Адаптивність сайту під розмір екрану телефону

Як ми бачили на рис. 3.6, для користувача із роллю Студент доступні пункти меню Tests (список доступних тестів) та Scores (результати тестів). Спробуймо пройти тест, натиснувши кнопку Start Test.

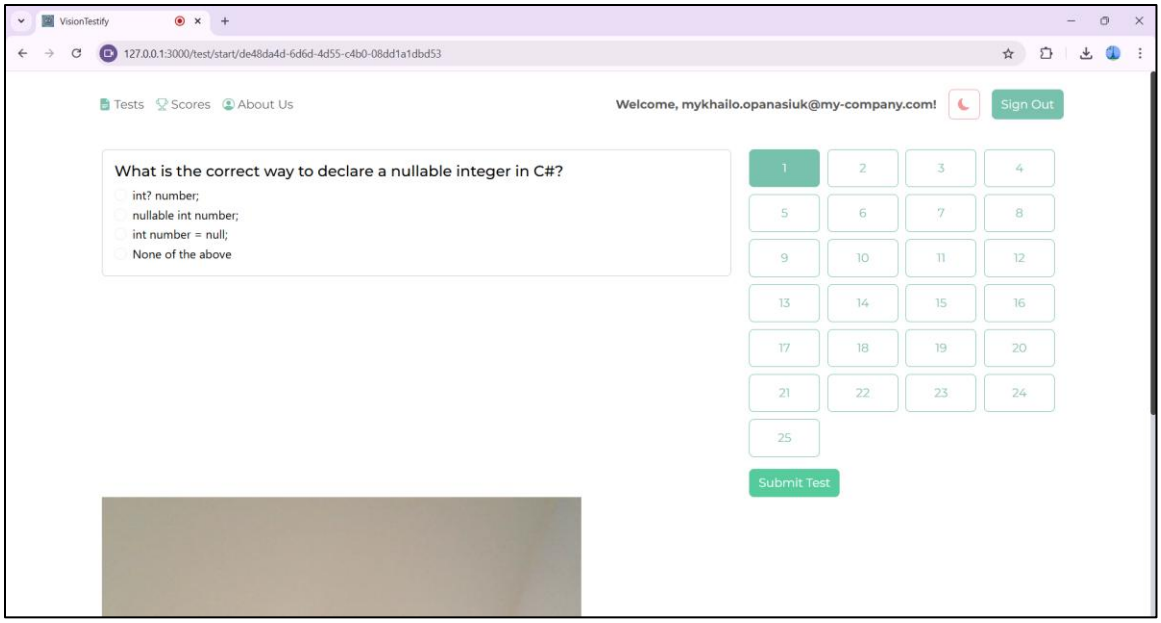

Рис. 3.8. Форма проходження тесту

Під час початку проходження тесту вмикається камера (зображена нижче та обрізана, щоб не показувати лице учасника) та учасник тестування проходить тести. Після проходження тесту варто натиснути Submit Test.

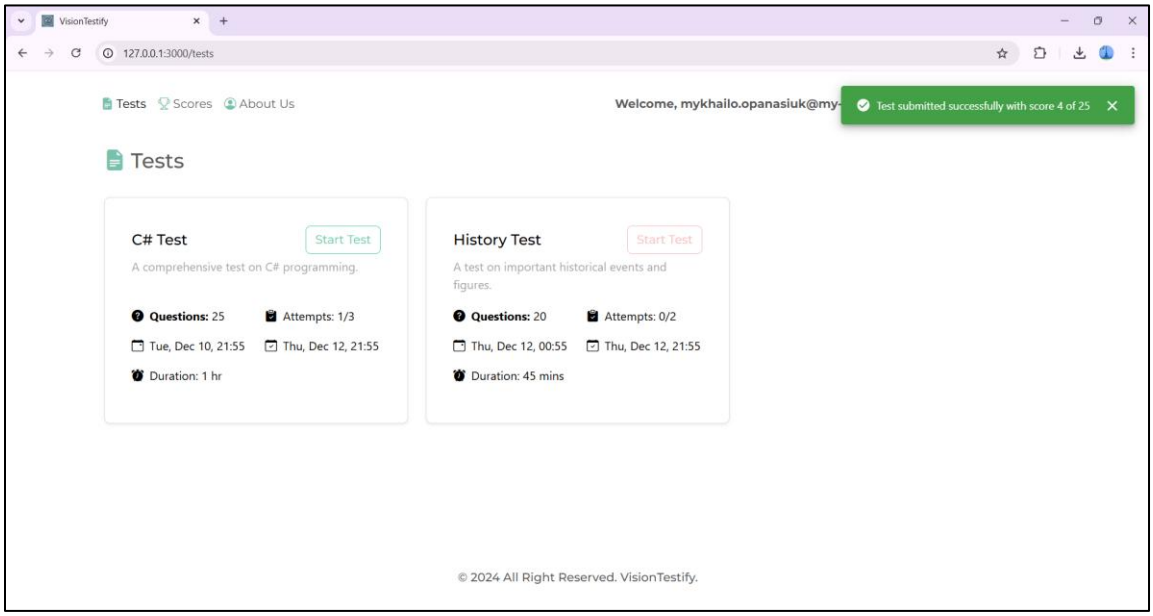

Рис. 3.8. Сторінка із тестами із відображенням результату у формі спливаючої підказки на сайті.

Можемо також впевнитись, що кількість спроб на тест зменшилась (1/3). Перейдемо у розділ Scores, щоб побачити цей результат (рис. 3.9)

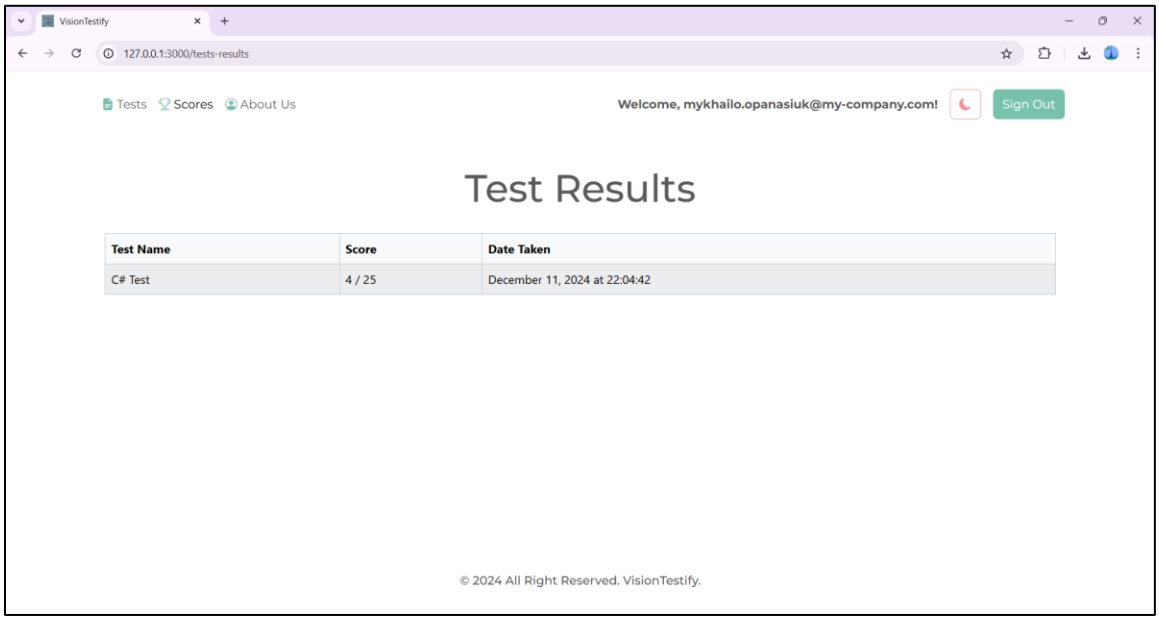

Рис. 3.9. Сторінка із результатами тесту учасника

Також для усіх користувачів доступна можливість змінювати тему сайту. Доступні дві – темна та світла. Для зміни теми варто натиснути на іконку місяцю праворуч зверху на екрані та змінимо тему на темну (рис. 3.10)

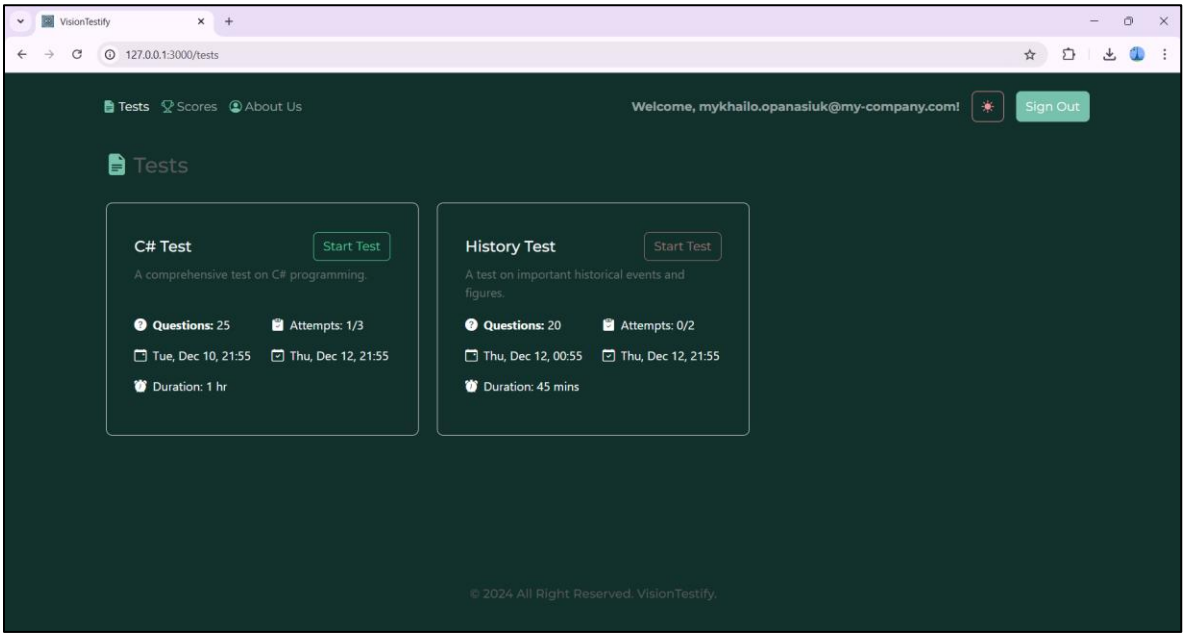

Рис. 3.10. Темна тема застосунку

До речі, для студента також доступна можливість отримати інформацію про тести, які доступні зараз, підключити нотифікації та дізнатись результати тестувань у телеграм боті VisionTestifyBot. Для цього відкриємо месенджер Telegram та знайдемо відповідного бота (рис. 3.11) а потім почнемо з ним роботу (рис. 3.12).

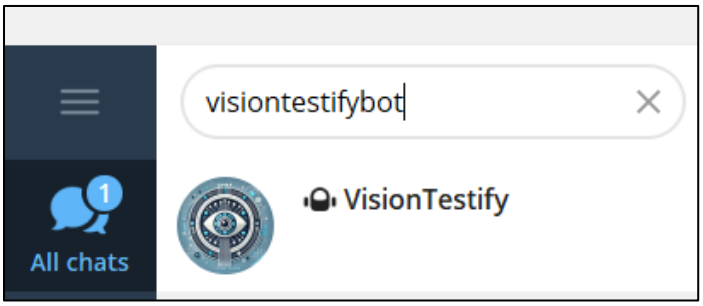

Рис. 3.11. Пошук телеграм бота до застосунку

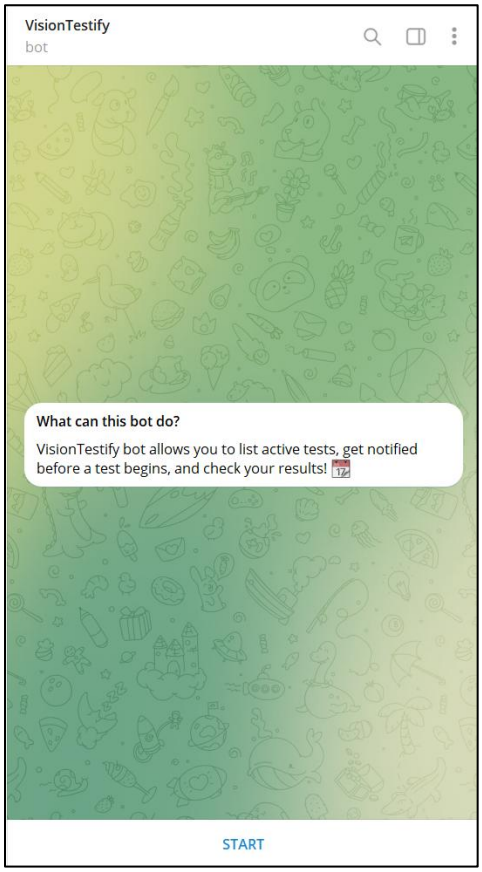

Рис. 3.12. Телеграм-бот VisionTestify перед початком роботи

Для початку роботи натиснемо кнопку Start.

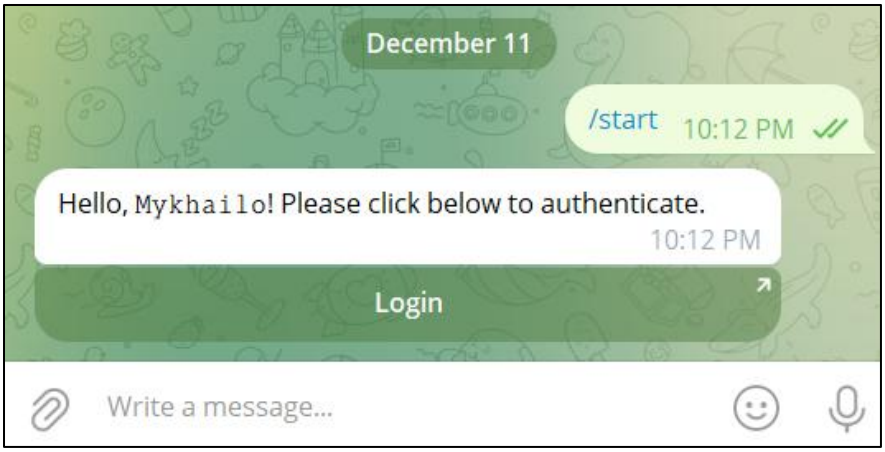

Рис. 3.13. Початок роботи із телеграм-ботом

Користувачу необхідно аутентифікуватись через бот, щоб мати доступ до функціоналу. Для цього натиснемо на Login кнопку. Телеграм запропонує перенаправити у браузер для аутентифікації – погоджуємось, натиснувши Open кнопку (рис. 3.14)

Рис. 3.14 – Вікно-погодження для перенаправлення у браузер

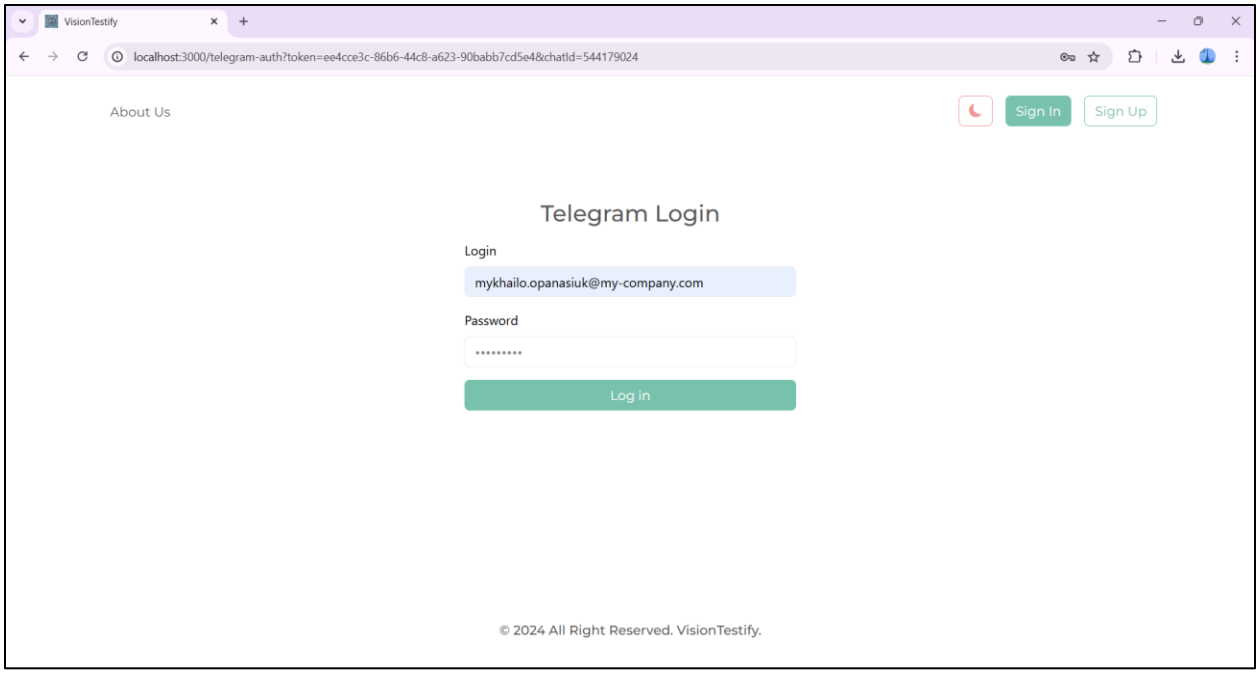

Рис. 3.15. Сторінка аутентифікації для телеграм-боту

Вводимо логін та пароль користувача та натискаємо Log in, як це зображено на рис. 3.15.

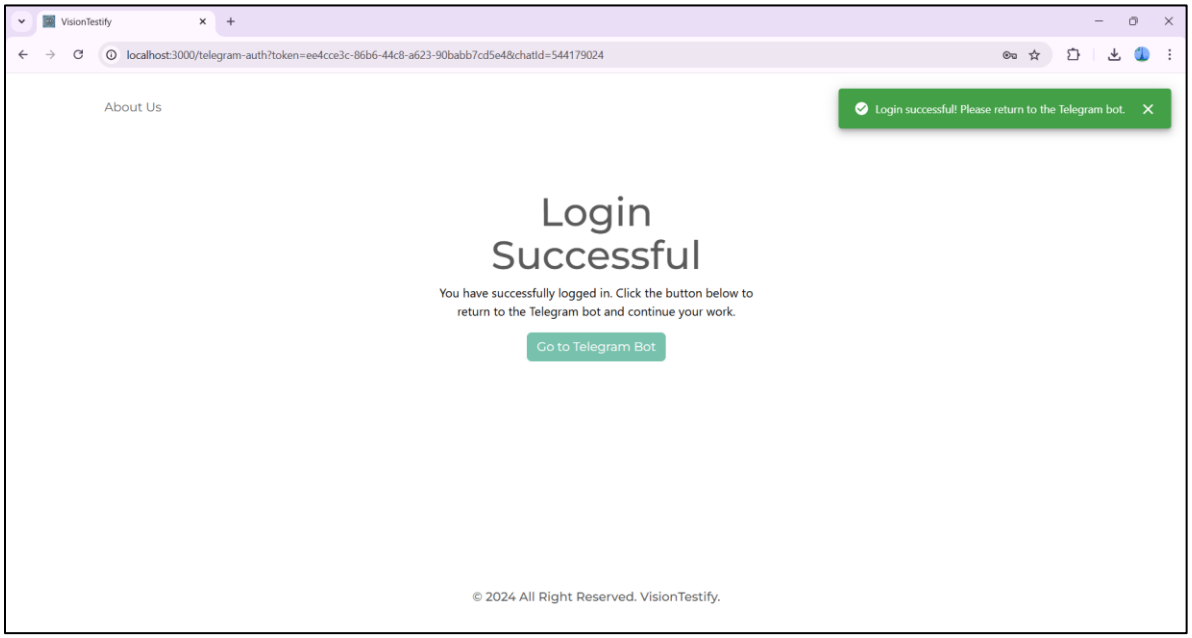

Рис. 3.16. Сторінка, що з’являється після успішної аутентифікації через телеграм-бот.

Після аутентифікації, натиснемо кнопку Go to Telegram Bot, що перенаправить нас у телеграм бот та побачимо, що для користувача з’явились нові функції, що були описані вище (рис. 3.17).

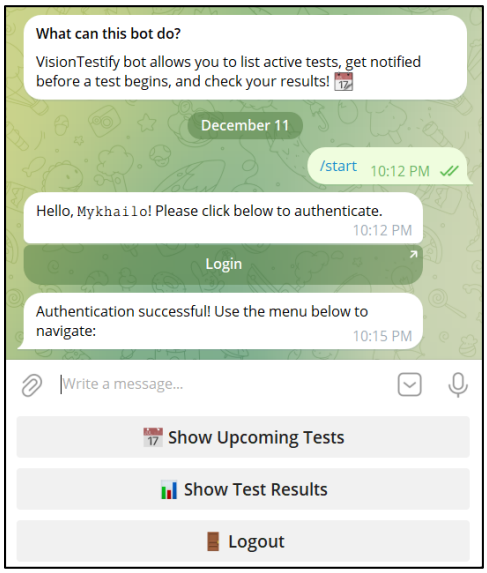

Рис. 3.17. Меню телеграм-боту після аутентифікації

Спробуймо натиснути Show Upcoming Tests, щоб побачити, які тести доступні зараз для користувача (рис 3.18).

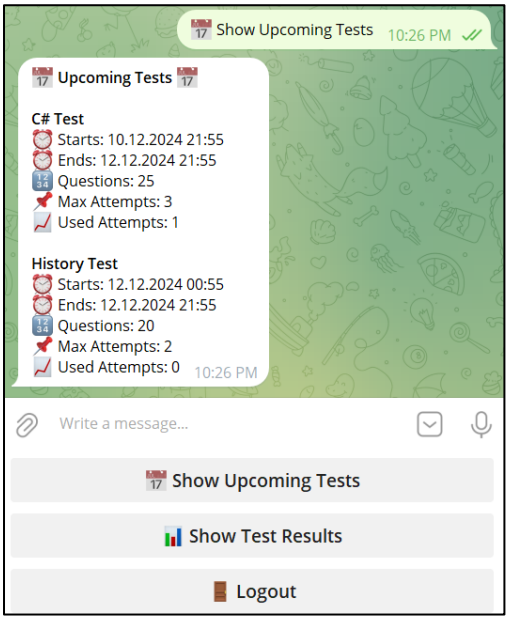

Рис. 3.18. Результат роботи кнопки Show Upcoming Tests

Як бачимо, ми отримали ті ж тести, що і у браузері, але без необхідності відвідувати сайт тепер. Спробуймо отримати результати тестів, натиснувши Show Test Results (рис. 3.19).

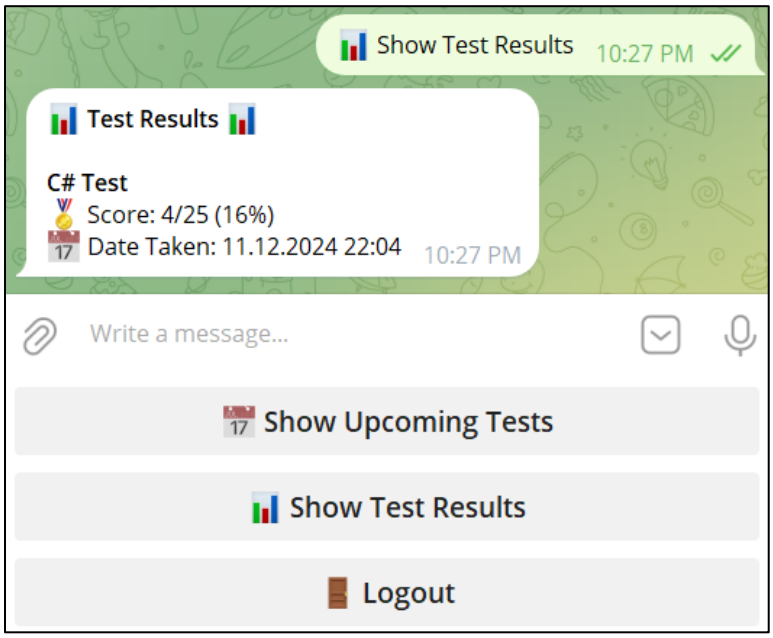

Рис. 3.19. Результат роботи кнопки меню Show Test Results

Для учасників тестування також приходять нотифікації за 15 хвилин до тесту, для цього мануально sql скриптом змінимо дату тесту на потрібну нам, щоб не чекати саме потрібного часу, виконавши sql скрипт:

```
UPDATE Tests SET StartTime = DATEADD(MINUTE, 15, GETUTCDATE());
```

Після чого вручну запустимо планувальника, що працює на бекенді, щоб він розіслав нотифікації (рис. 3.20) поставивши checkbox біля send-notifications-for-users-before-tests-begin-job елементом та натиснувши Trigger now.

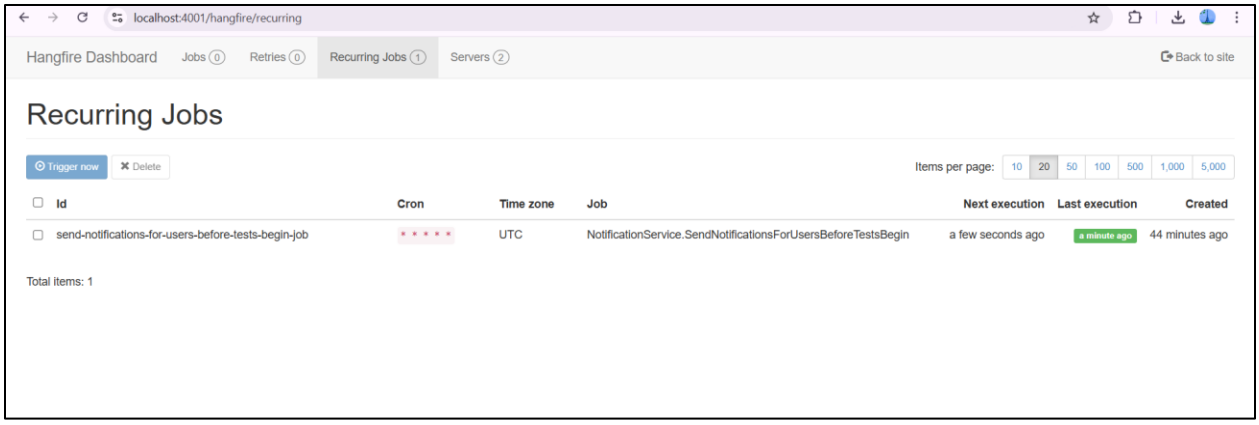

Рис. 3.20. Тригер джоби, що надсилає нотифікації

Планувальник і так щохвилини перевіряє та автоматично надсилає нотифікації, але для спрощення ми виконали цей алгоритм та отримали нотифікацію у боті (рис. 3.21)

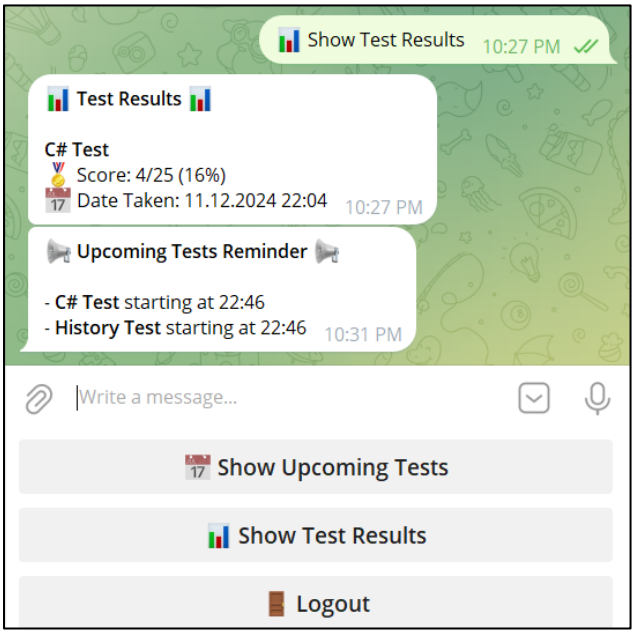

Рис. 3.21. Нотифікація, що з’явилась автоматично для нагадування про наступні тести.

Як бачимо на рис. 3.21, нотифікація з’явилась автоматично, що дозволить сповістити учасників про початок тесту. Якщо користувач хоче від’єднати профіль, варто просто натиснути кнопку Logout (рис. 3.22).

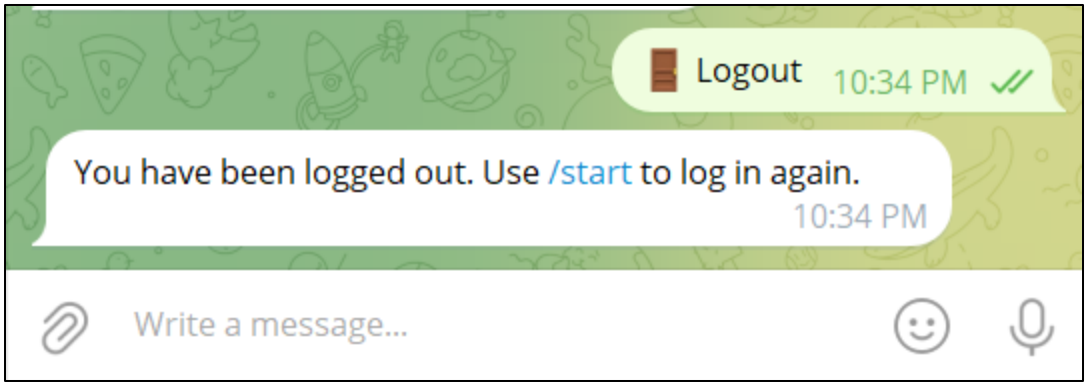

Рис. 3.22. Від’єднання профілю від телеграм-боту

Повернемось до двох інших ролей – адміністратора та вчителя. Оскільки адміністратор має усі ті ж ролі, що й вчитель і навіть більше, то увійдемо в систему під адміністратором, щоб не дублювати документацію по застосунку. Для цього на формі логіну введемо логін admin@gmail.com та пароль AdminQwerty_1. Цей користувач за замовчуванням створюється при першому запуску бекенду, тож нам не потрібно його реєструвати.

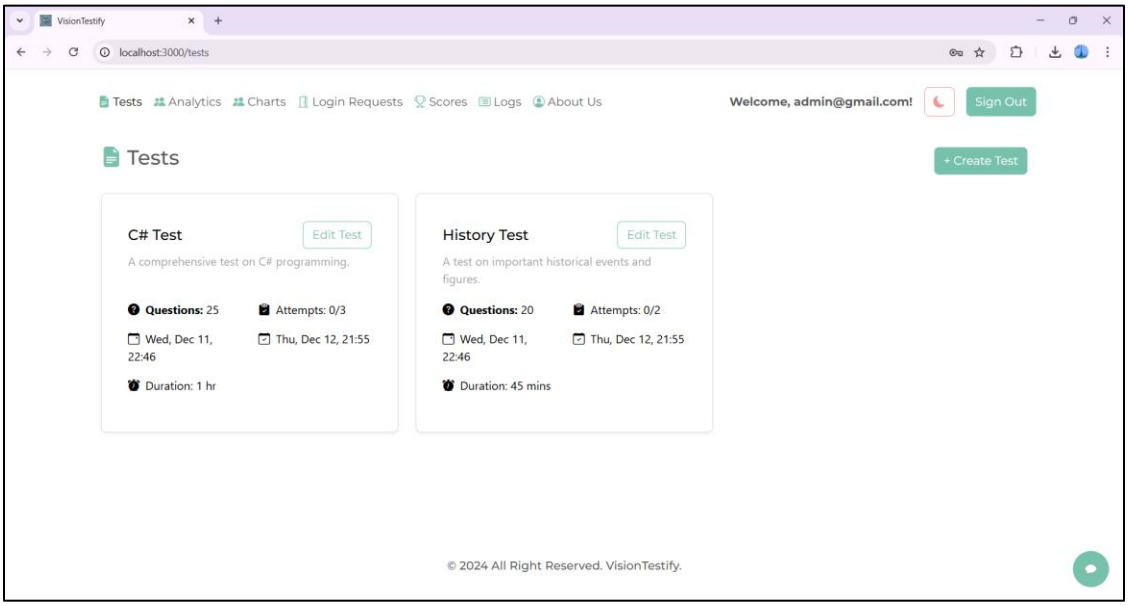

Рис. 3.23. Домашня сторінка адміністратора

У адміністратора доступно більше вкладок, аніж у студента та викладача (у викладача недоступна вкладка Logs, оскільки вона технічна та дає змогу

переглянути логи застосунку). Спробуймо відредагувати тест натиснувши кнопку Edit Test на одному із тестів (рис. 3.24).

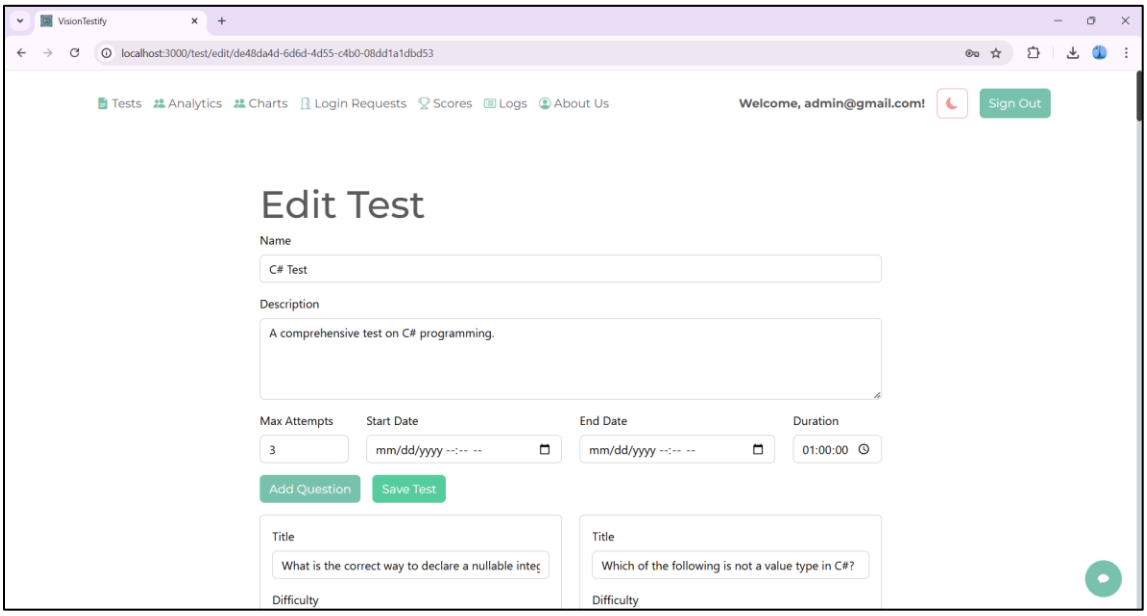

Рис. 3.24. Форма редагування тесту

Також, лише для користувачів із ролями Викладач та Адміністратор доступний помічник-асистент, що дозволить швидше створити тести, або ж допоможе відповісти на запитання. Для цього справа знизу варто натиснути на значок чату (рис. 3.25) та відкриється діалогове вікно із помічником.

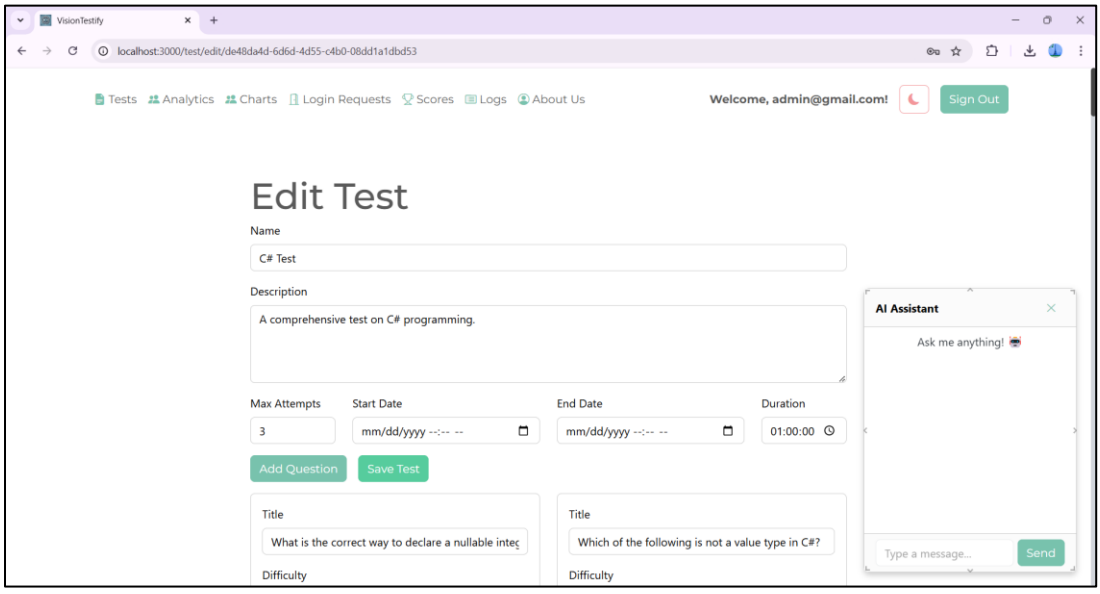

Рис. 3.25. Діалогове вікно із помічником-асистентом

Для зручності у користувача є можливість змінювати розміри модального вікна та переміщати його на сторінці, що й продемонстровано на рис. 3.26. Протестуймо його, запитавши у нього згенерувати 2 запитання про Moq бібліотеку у c# та отримаємо результат.

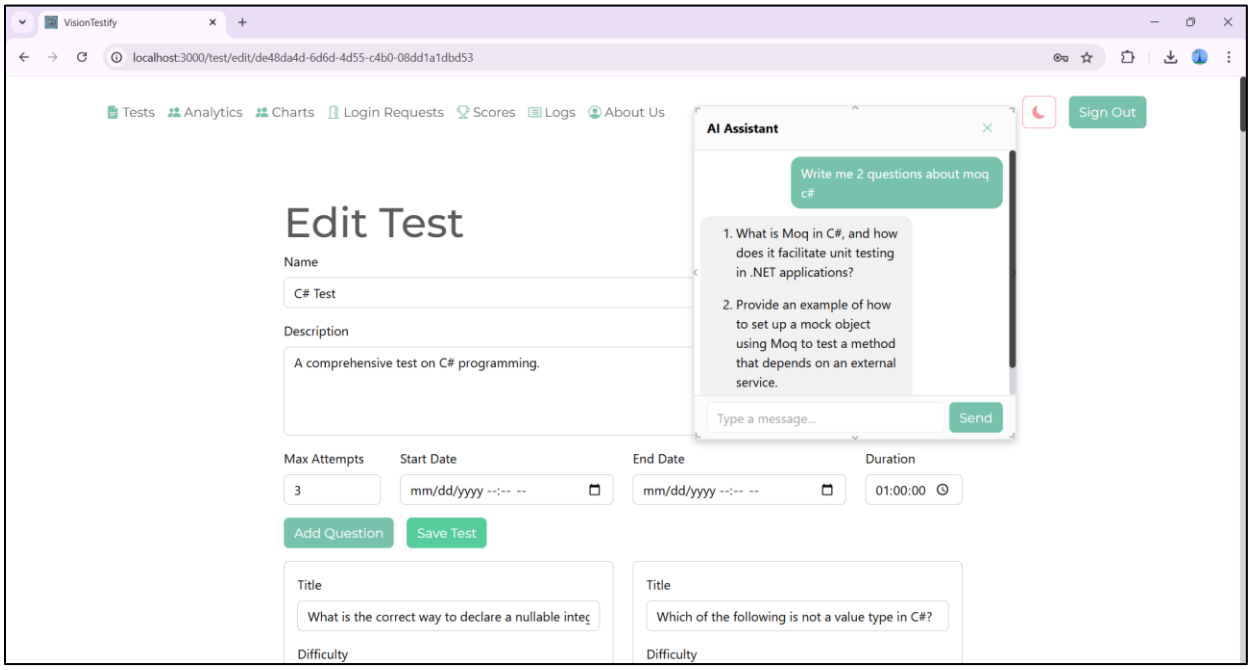

Рис. 3.26. Результат роботи помічника-асистента

Для адміністратора та викладача доступні сторінки Analytics (рис. 3.27) та Charts (3.28), що дозволяють переглянути аналітику, що була згенерована під час проходження тесту студентами. Відповідно, вкладка Analytics показує погруповану по тестами та учасникам аналітику із фотографіями, коли було помічено підозрілу активність, а вкладка Charts показує вже у графіках дану аналітику.

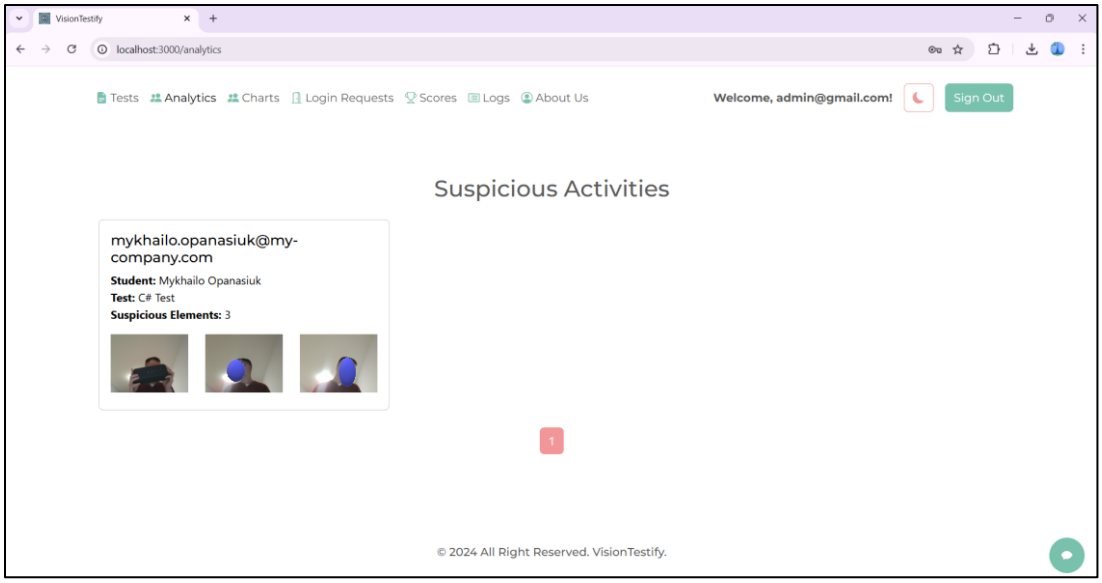

Рис. 3.27. Сторінка з аналітикою (лице учасника замальовано у редакторі)

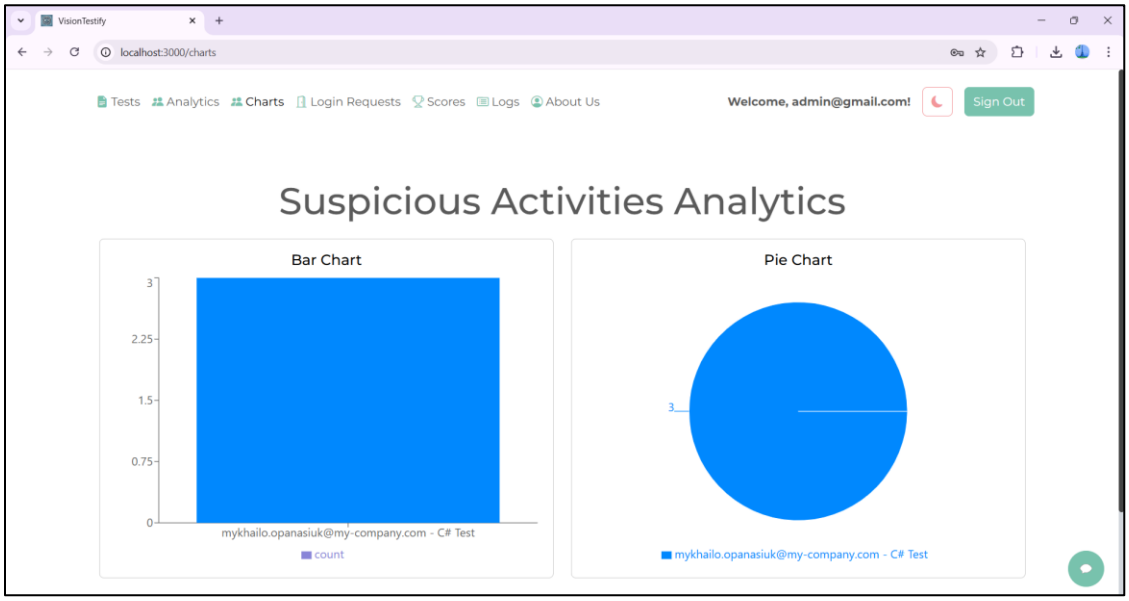

Рис. 3.28. Сторінка із графіками

Лише для адміністратора доступна вкладка Login Requests (рис. 3.29), що дозволяє переглянути запити на реєстрацію від викладачів, оскільки вони потребують підтвердження. Лише після активації адміністратором акаунт викладача стає активним та користувач із такою роллю зможе увійти на сайт.

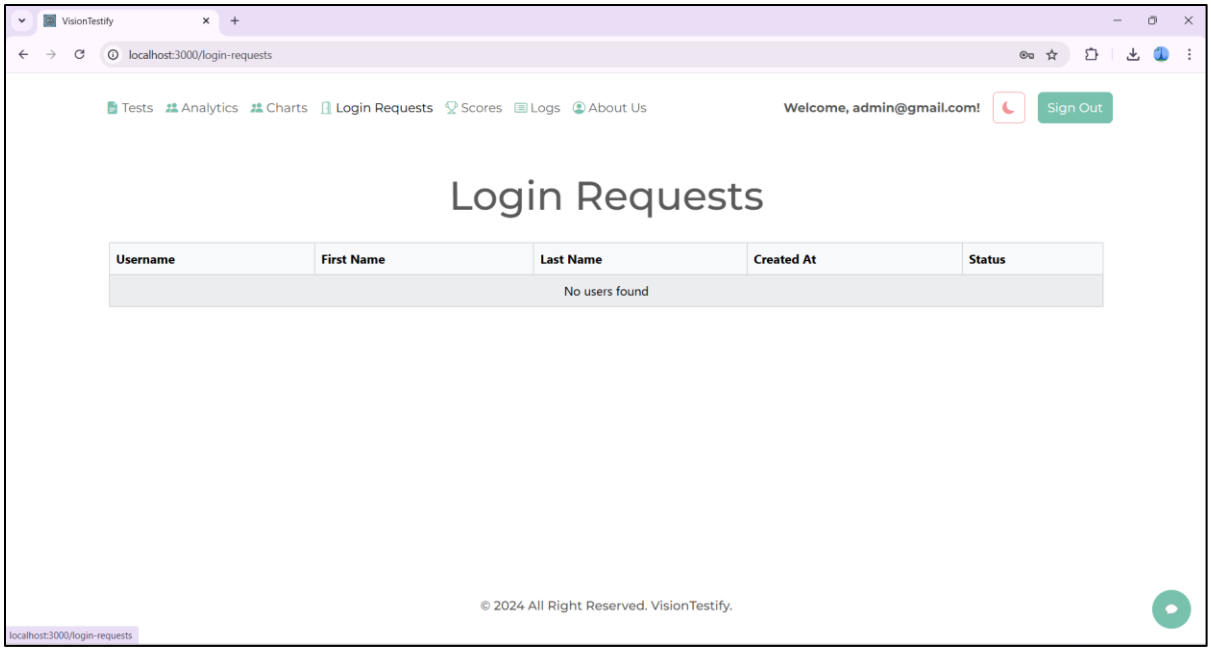

Рис. 3.29. Сторінка запитів на реєстрацію

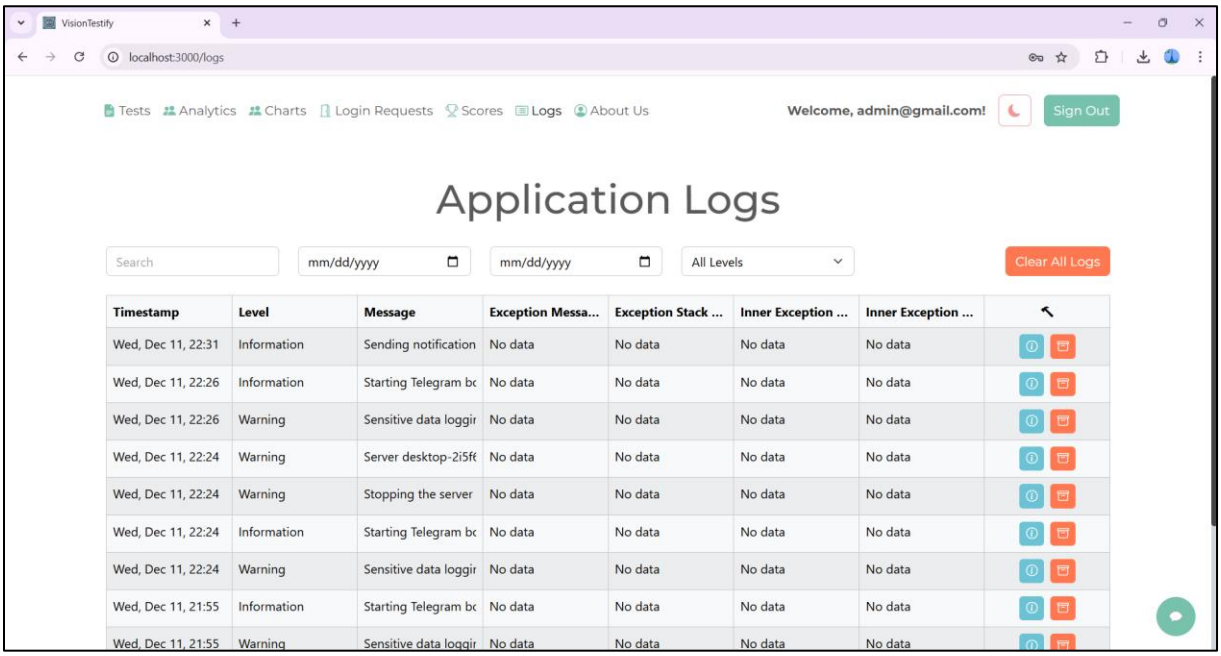

Рис. 3.30. Вкладка логів у системі, що доступна для адміністратора

Для можливості перегляду логів та подій у системі для адміністратора є додаткова панель, де він може переглянути події, якщо щось у додатку пішло не так та швидко відреагувати, як це зображено на рис. 3.30.

Висновок

У третьому розділі було детально розглянуто процес реалізації системи онлайн-тестування, включаючи проектування бази даних, серверної частини та фронтенд-компонентів. Було створено чітку структуру бази даних, яка забезпечує ефективне зберігання даних про користувачів, тести, результати та інші важливі аспекти роботи системи. Серверна частина реалізована на основі мікросервісної архітектури з використанням .NET Core, що дозволило досягти високої продуктивності та масштабованості. Окремо було створено Python-застосунок на Flask для інтеграції функціоналу розпізнавання облич та виявлення підозрілої активності.

Фронтенд-частина реалізована за допомогою React, що забезпечує динамічний та зручний користувацький інтерфейс із сучасним дизайном. Система дозволяє виконувати базові та розширені функції, такі як реєстрація, авторизація, управління тестами та їх проходження, а також інтеграцію зі сторонніми сервісами, такими як Telegram.

Таким чином, розроблена система є повнофункціональним рішенням, яке відповідає сучасним вимогам до онлайн-тестування, забезпечуючи зручність, точність і безпеку. Це рішення може бути легко адаптоване під потреби різних освітніх установ і організацій.

ВИСНОВКИ

У процесі виконання цієї роботи було розроблено систему онлайн-тестування, яка відповідає сучасним вимогам до організації ефективного та прозорого контролю знань. Головною метою проєкту було створення інструменту, який забезпечить зручність у використанні для викладачів і студентів, а також інтегрує сучасні технології для підвищення точності й безпеки тестування.

У ході роботи було досліджено теоретичні аспекти онлайн-тестування, зокрема його роль у системі контролю якості знань. Проведений огляд наявних платформ дозволив оцінити переваги та недоліки існуючих рішень, що дало можливість виділити ключові напрями для вдосконалення функціоналу. Це стало основою для проєктування власної системи, яка враховує сучасні технологічні тенденції та потреби користувачів.

Особлива увага була приділена вивченню технологій моніторингу поведінки студентів під час тестування. Розглянуто сучасні підходи до розпізнавання обличчя та поведінкової аналітики, зокрема використання комп’ютерного зору для виявлення підозрілої активності. Це дозволило інтегрувати модулі на основі комп’ютерного зору, що сприяє підвищенню рівня прозорості й справедливості оцінювання, зменшуючи можливість списування.

Проаналізовано результати учасників національного мультипредметного тесту (НМТ), що дало змогу оцінити типові проблеми та поведінкові особливості студентів під час тестування. Зібрані дані були враховані під час розробки алгоритмів системи, зокрема для адаптації її до різних сценаріїв використання та підвищення точності моніторингу.

Розроблена система включає кілька ключових компонентів: серверну частину, клієнтську частину та інтеграцію модулів для аналізу поведінки користувачів. Архітектура побудована з використанням сучасних технологій, таких як .NET Core, React та Python. Впровадження телеграм-бота забезпечує сповіщення про результати тестування, тоді як автоматизація створення тестів зменшує навантаження на викладачів.

Технічна реалізація проєкту здійснена з використанням сучасного стеку технологій, що включає .NET Core 8, EF Core, Hangfire, MS SQL та C#. Використання підходу Code First у фреймворку Entity Framework Core дозволило автоматизувати створення структури бази даних і забезпечити ефективне зберігання та обробку великих обсягів даних. Інтеграція клієнтської частини на основі React забезпечує зручний і динамічний інтерфейс для користувачів.

Реалізована система має широке практичне застосування. Вона може використовуватися у навчальних закладах, корпоративних тренінгах та інших сферах, де необхідний контроль знань і аналіз результатів. Гнучкість у налаштуванні та інтеграції зі сторонніми сервісами дозволяє адаптувати систему до різних потреб користувачів.

У підсумку можна стверджувати, що розроблена система є сучасним рішенням для організації онлайн-тестування. Вона створює основу для подальшого вдосконалення, включаючи інтеграцію новітніх алгоритмів штучного інтелекту та розширення функціональних можливостей для задоволення вимог майбутніх користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Albahari J., Albahari B. C# 10 in a Nutshell: The Definitive Reference. O'Reilly Media, 2022;
2. Troelsen A., Japikse P. Pro C# 9 with .NET 5: Foundational Principles and Practices in Programming. Apress, 2021;
3. Freeman E., Robson E. Head First Design Patterns: A Brain-Friendly Guide. O'Reilly Media, 2020;
4. Fowler M. Refactoring: Improving the Design of Existing Code. Addison-Wesley Professional, 2018;
5. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, 1994;
6. McConnell S. Code Complete: A Practical Handbook of Software Construction. Microsoft Press, 2004;
7. Metz S. Practical Object-Oriented Design: An Agile Primer Using Ruby. Addison-Wesley Professional, 2012;
8. Richter J. CLR via C#: Applying C# to the Microsoft .NET Framework. Microsoft Press, 2012;
9. Dreyfus S., Eisenberg E. Eloquent JavaScript: A Modern Introduction to Programming. No Starch Press, 2018;
10. Щебликіна Т. А. Тестування як метод вимірювання якості навчальних досягнень студентів педагогічних спеціальностей: стаття. URL: https://lib.chmnu.edu.ua/pdf/naukpraci/pedagogika/2014/246-234-27.pdf (дата звернення: 11.11.2024);
11. Мороховець Г. Ю. Тестування як форма контролю та діагностики знань здобувачів вищої освіти: стаття. URL: https://otr.iod.gov.ua/images/pdf/2018/3/04.pdf (дата звернення: 11.11.2024);

12. Freeman A., Sanderson A. Pro ASP.NET Core 6: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages. Apress, 2022;
13. Sedgewick R., Wayne K. Algorithms. Addison-Wesley Professional, 2011;
14. Telegram API: офіційний веб-сайт. URL: https://core.telegram.org/api (дата звернення: 22.11.2024);
15. OpenAI API: офіційний веб-сайт. URL: https://platform.openai.com/docs/ (дата звернення: 22.11.2024);
16. Haar Cascades: документація OpenCV. URL: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html (дата звернення: 10.10.2024);
17. Офіційний звіт про результати. URL: https://testportal.gov.ua/wp-content/uploads/2024/09/ZVIT-NMT_2024-Tom-1.pdf (дата звернення: 12.12.2024).