МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МИКОЛАЇВСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

Факультет менеджменту

Кафедра економічної кібернетики, комп'ютерних наук та інформаційних технологій

Кваліфікаційна наукова
праця на правах рукопису

КУДІН Костянтин Сергійович

УДК 621.392.71:004.451.1(043.2)

КВАЛІФІКАЦІЙНА РОБОТА

РОЗРОБКА АДАПТИВНОГО ВЕБ-ЗАСТОСУНКУ ДЛЯ СТВОРЕННЯ, АДМІНІСТРУВАННЯ ТА ПРОХОДЖЕННЯ ОСВІТНІХ ТЕСТІВ

Спеціальність 122 «Комп'ютерні науки»
Галузь знань -12 «Інформаційні технології»

Подається на здобуття освітнього ступеня «Бакалавр»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Кудін К. С.

Науковий керівник: Крайній Володимир Олексійович кандидат економічних наук, в.о. доцент.

Завідувач кафедри: Тищенко Світлана Іванівна, кандидат педагогічних наук, доцент.

Миколаїв–2025

АНОТАЦІЯ

Кудін К. С. Веб-застосунок для створення та адміністрування тестів. - Кваліфікаційна наукова праця на правах рукопису. Робота на здобуття освітнього ступеня бакалавра за спеціальністю 122 «Комп'ютерні науки». -Миколаївський національний аграрний університет, Миколаїв, 2025.

Об'єктом дослідження є процеси комп'ютерного тестування в освітньому середовищі.

Предмет дослідження -технології створення веб-застосунку для генерації, адміністрування та проходження тестів, адаптованого під різні типи пристроїв.

Метою роботи є розроблення інтуїтивно зрозумілого, адаптивного та продуктивного веб-застосунку для створення та адміністрування тестів, що забезпечує ефективну взаємодію між користувачем і системою, підтримує перегляд результатів і налаштування тестів, а також демонструє кросплатформену адаптацію.

Робота складається з фахового розділу і спеціального розділу з охорони праці. Пояснювальна записка включає вступ, три розділи, висновки, список використаних джерел та додатки.

У першому розділі досліджено сучасні підходи до створення інформаційних систем для тестування, описано існуючі аналоги, їх функціональні особливості, переваги й недоліки.

У другому розділі представлено концепцію розробки веб-застосунку, спроєктовано структуру інтерфейсу, визначено функціональні модулі системи, технології та середовище розробки.

У третьому розділі описано реалізацію користувацького інтерфейсу, тестування застосунку на реальних даних, адаптацію до мобільних пристроїв, перевірку продуктивності та відображення на різних екранах.

Результати дослідження обговорювалися під час підготовки до випускної атестації, частина функціоналу була протестована на реальних користувачах у межах навчального процесу.

Кваліфікаційна робота викладена на 61 сторінках, містить 2 таблиць, 20 рисунків, 4 схем, список використаних джерел включає 20 найменувань.

Ключові слова: веб-застосунок, тестування, JavaScript, React, Node.js, адаптивний дизайн, Material UI, база даних, результат, користувач.

ANNOTATION

Kudin K. S. Web application for creating and administering tests. - Qualification scientific work in the form of a manuscript.

Work for obtaining a bachelor's degree in specialty 122 "Computer Science". - Mykolaiv National Agrarian University, Mykolaiv, 2025.

The object of the study is the processes of computer testing in the educational environment.

The subject of the study is the technology of creating a web application for generating, administering and passing tests, adapted for different types of devices.

The purpose of the work is to develop an intuitive, adaptive and productive web application for creating and administering tests, which provides effective interaction between the user and the system, supports viewing results and setting up tests, and also demonstrates cross-platform adaptation.

The work consists of a professional section and a special section on labor protection. The explanatory note includes an introduction, three sections, conclusions, a list of sources used, and appendices.

The first section examines modern approaches to creating information systems for testing, describes existing analogues, their functional features, advantages, and disadvantages.

The second section presents the concept of developing a web application, designs the interface structure, defines the functional modules of the system, technologies, and development environment.

The third section describes the implementation of the user interface, testing the application on real data, adaptation to mobile devices, performance testing, and display on different screens.

The results of the study were discussed during preparation for the final certification, part of the functionality was tested on real users within the educational process.

The qualification work is presented on 61 pages, contains 2 tables, 20 figures, 4 diagrams, the list of used sources includes 20 items.

Keywords: web application, testing, JavaScript, React, Node.js, adaptive design, Material UI, database, result, user.

Перелік найбільш уживаних умовних позначень, символів і термінів

с.	сторінка
табл.	таблиця
рис.	Рис.
UI	користувацький інтерфейс (User Interface)
UX	користувацький досвід (User Experience)
SPA	односторінковий застосунок (Single Page Application)
API	програмний інтерфейс додатків (Application Programming Interface)
JSON	формат обміну даними (JavaScript Object Notation)
CSS	каскадні таблиці стилів
MUI	Material UI - бібліотека інтерфейсів
AOS	бібліотека анімацій при скролі (Animate On Scroll)
Node.js	середовище виконання JavaScript
React.js	JavaScript бібліотека для побудови інтерфейсу
ORM	об’єктно-реляційне відображення (Object-Relational Mapping)
PWA	прогресивний веб-застосунок (Progressive Web App)
HTTP	протокол передачі гіпертексту
HTTPS	захищений HTTP
CDN	мережа доставки контенту (Content Delivery Network)

ЗМІСТ

ВСТУП

У сучасних умовах цифровізації освіти особливої актуальності набуває розробка ефективних веб-застосунків, які забезпечують інтерактивне, адаптивне та зручне навчальне середовище. Одним із ключових напрямів у цьому контексті є створення систем онлайн-тестування, що дозволяють організовувати об'єктивне оцінювання знань, незалежно від місця та часу проведення тестування. В умовах постійного зростання обсягу навчальної інформації та необхідності швидкої перевірки знань, автоматизовані системи оцінювання стають незамінними інструментами для закладів освіти всіх рівнів.

Актуальність теми дослідження полягає у необхідності створення універсального, адаптивного та масштабованого веб-застосунку для освітнього тестування, який забезпечить якісний зворотний зв’язок для студентів і викладачів, підвищить ефективність освітнього процесу та дасть змогу оптимізувати ресурси викладача. Підвищення вимог до зручності користування, безпеки персональних даних, адаптивності під різні пристрої та інтеграції з іншими освітніми платформами актуалізують потребу у нових технологічних рішеннях.

Мета дослідження полягає у розробці, обґрунтуванні та впровадженні адаптивного веб-застосунку для створення, адміністрування та проходження освітніх тестів, який забезпечує ефективну взаємодію між користувачем та системою, високу продуктивність, адаптивність до різних пристроїв та можливість подальшої інтеграції в освітні середовища.

Завдання дослідження:

- проаналізувати існуючі онлайн-системи тестування та визначити їх переваги й недоліки;
- дослідити особливості розробки адаптивних веб-застосунків;
- визначити методи генерації та перевірки тестових завдань;

- здійснити порівняльний аналіз технологій для розробки адаптивних освітніх платформ;
- спроєктувати структуру та логіку веб-застосунку;
- реалізувати функціональні модулі для створення, адміністрування та проходження тестів;
- провести тестування застосунку на реальних даних;
- оптимізувати інтерфейс під різні типи пристроїв.

Об’єкт дослідження - процес онлайн-тестування в цифровому освітньому середовищі.

Предмет дослідження - технології та алгоритми розробки адаптивного веб-застосунку для освітніх тестів.

Методи дослідження: теоретичні (аналіз, синтез, узагальнення, порівняння), емпіричні (спостереження, моделювання), програмно-інженерні (проектування, реалізація, тестування), інструментальні (використання JavaScript, React.js, Node.js, Material UI, AOS, Chrome DevTools).

Інформаційна база дослідження включає наукові публікації вітчизняних та зарубіжних авторів у сфері комп’ютерних наук, нормативно-правові акти, документацію до програмних засобів (React, Node.js, MUI), статистичні дані, а також результати практичної реалізації та тестування створеного застосунку.

Практичне значення дослідження полягає у можливості впровадження розробленого веб-застосунку в навчальний процес як інструменту для самостійного оцінювання, проведення контрольних робіт або практикумів. Результати дослідження також можуть бути корисними розробникам освітнього програмного забезпечення для створення гнучких та масштабованих систем оцінювання знань.

РОЗДІЛ 1

ОНЛАЙН-СИСТЕМИ ТЕСТУВАННЯ ТА ЇХ РОЛЬ В СУЧАСНІЙ ОСВІТІ

1.1.Огляд існуючих систем для онлайн-тестування

Зі стрімким розвитком інформаційних технологій та широким впровадженням дистанційного навчання зростає значення онлайн-тестування як ключового інструменту для оцінювання знань. На сьогодні існує велика кількість онлайн-платформ для проведення тестування в освітніх закладах, вибір яких залежить від потреб закладу, особливостей навчального процесу, необхідного функціоналу та бюджету [1].

Однією з найбільш поширених систем є Moodle - це відкрита платформа для електронного навчання, яка підтримує різноманітні типи завдань: тести з одним або кількома правильними варіантами, відкриті питання, завдання на відповідність, числові задачі та заповнення пропусків. Система управління навчанням Moodle надає широких можливостей для створення тестів різного типу: визначена кількість спроб проходження тесту; встановлення тимчасових затримок між спробами; вибір методу оцінювання у випадку кількох спроб (вища/середня оцінка, перша/остання спроба); перемішування як самих питань у тесті, так і варіантів відповідей; побудова тесту на основі випадкового вибору питань з категорій; навчальний режим (можливість відповісти на питання кілька разів у межах однієї спроби, нарахування штрафних балів за кожну неправильну відповідь); налаштування режиму перегляду результатів (що і коли саме буде показано студенту); надання коментарів до кожного питання, варіанту відповіді чи всього тесту [2]. Вона має гнучкі налаштування та великий вибір додаткових модулів і плагінів, що дає змогу адаптувати систему під конкретні потреби навчального закладу. Однією з головних переваг Moodle є активна

міжнародна спільнота розробників, яка забезпечує постійну підтримку та регулярні оновлення [3].

Google Forms -проста і безкоштовна платформа, популярна завдяки зручності створення форм для швидкого оцінювання знань [4]. Google Forms ефективно використовується для коротких контрольних робіт, збору зворотного зв'язку, реєстраційних форм або швидких тестів після навчальних модулів. Незважаючи на простоту, вона має обмеження щодо налаштування складних логік перевірки відповідей та аналітики результатів, але залишається ідеальним рішенням для оперативного тестування та збору статистики.

Quizlet - це сервіс, який дозволяє легко запам'ятовувати будь-яку інформацію, яку можна подати у вигляді навчальних карток. Все що потрібно - це знайти в базі або створити власні картки. У безкоштовній версії обмежений функціонал, а у платних тарифах до карток можна додавати своє аудіо, картинки, створюючи більш інтерактивний матеріал. Інтерактивна платформа, що ефективно використовується учнями середньої та старшої школи, а також студентами університетів [5]. Особливо підходить для вивчення іноземних мов, термінології з природничих та гуманітарних дисциплін і для підготовки до стандартизованих тестів. Quizlet пропонує різні режими навчання: флеш-картки, тести, ігри на відповідність і вправи для запам'ятовування, що робить процес навчання інтерактивним і захоплюючим.

Kahoot - платформа для створення інтерактивних вікторин з елементами гейміфікації, особливо популярна серед молодших школярів і підлітків. Завдяки яскравому дизайну і змагальному формату, Kahoot значно підвищує мотивацію учнів та забезпечує активну участь у процесі навчання [6].

Testportal - комерційна платформа з різноманітними тарифними планами, що дозволяє адаптуватися до бюджету та вимог освітнього закладу. Вона пропонує потужні інструменти для створення складних тестових

сценаріїв з детальною аналітикою [7]. Testportal особливо підходить для професійних освітніх установ, де важливими є поглиблений аналіз результатів тестування, високий рівень безпеки та захист персональних даних.

Також на ринку представлені інші рішення:

- Blackboard -комплексне рішення для університетів з широкими можливостями адміністрування навчання;
- ClassMarker -спеціалізується на професійному тестуванні з високим рівнем безпеки;
- Edmodo - освітня технологічна платформа, що пропонує комунікацію, співпрацю та можливість тренерської роботи для загальноосвітніх шкіл, коледжів та викладачів. Мережа Edmodo дає вчителям змогу ділитися вмістом, створювати тести, вікторини та опитування, керувати спілкуванням з учнями, колегами та батьками. За пів року користування Edmodo вже склалися певні традиції навики, прийоми. Є свої розробки тестів. Пропонується до уваги опис досвіду роботи у освітній соціальній мережі Edmodo;
- Socrative - Освітоорієнтований сервіс для створення і проведення опитувань, тестів, голосувань, формативних завдань. Пропонує швидкі інтерактивні тести з миттєвим зворотним зв'язком, що ідеально підходить для класних занять.

Вибір оптимальної системи онлайн-тестування повинен базуватися на детальному аналізі потреб освітньої установи, врахуванні функціональних можливостей платформ, їх зручності використання та економічної ефективності. Такий підхід забезпечить максимальну ефективність і комфорт користувачів у процесі онлайн-тестування.

1.2. Особливості адаптивних веб-застосунків

Адаптивні веб-застосунки є важливою складовою сучасного онлайн-навчання, оскільки дозволяють користувачам ефективно взаємодіяти із

системами незалежно від типу пристрою, на якому вони працюють. Головна особливість таких застосунків полягає у здатності автоматично налаштовувати інтерфейс та контент відповідно до розмірів екранів різних пристроїв (персональні комп’ютери, ноутбуки, планшети та смартфони), а також забезпечувати сумісність із різними операційними системами та браузерами. Завдяки адаптивності студенти можуть ефективніше взаємодіяти з навчальним контентом, що підвищує їхню залученість та мотивацію до навчання, адже навчання можливе будь-де і будь-коли без втрати функціоналу та зручності .

Однією з ключових переваг адаптивних веб-застосунків є покращення користувацького досвіду (User Experience, UX), що забезпечує зручність і простоту використання платформи. Завдяки цьому студенти можуть комфортно проходити тести або навчальні модулі незалежно від місця перебування. Це особливо важливо в умовах дистанційного навчання та сучасного мобільного стилю життя. Додатково, адаптивні веб-застосунки зменшують навантаження на користувачів, мінімізуючи необхідність додаткових налаштувань.

User experience, або користувацький досвід, являє собою сприйняття продукту, будь то веб-сайт чи мобільний додаток, з боку користувача. UX охоплює комплекс емоцій, вражень та дій, які відвідувач отримує при взаємодії із продуктом. При цьому він може отримати як позитивний, так і негативний досвід. Розуміння цього дозволяє бренду вдосконалювати продукт так, щоб він відповідав та задовольняв потреби та очікування відвідувача.

Для реалізації адаптивності активно використовуються сучасні технології веб-розробки, такі як CSS-фреймворки Bootstrap та Tailwind [8]. Ці інструменти надають розробникам гнучкі можливості створювати адаптивні інтерфейси з використанням гнучких сіток та готових компонентів, що значно прискорює процес розробки та полегшує підтримку проєкту. Також активно використовуються JavaScript-бібліотеки та фреймворки, зокрема

React, Vue.js та Angular, які дозволяють створювати інтерактивні та динамічні елементи, що коректно працюють на різних пристроях. До популярних інтерактивних елементів у навчальних застосунках належать інтерактивні тести та опитування, модулі із зворотним зв'язком у реальному часі, анімовані візуалізації матеріалів, інтерактивні завдання типу drag-and-drop, а також динамічні таблиці й графіки, що оновлюються залежно від введених користувачем даних.

Важливою характеристикою адаптивних веб-застосунків є також швидкість завантаження сторінок і стабільність роботи. Для оптимізації медіаконтенту використовуються методи стиснення зображень та відео, адаптивні формати (наприклад, WebP), CDN (Content Delivery Network) для швидкої доставки контенту, а також ліниве завантаження (lazy loading). Додатково застосовується асинхронне завантаження JavaScript-файлів і мінімізація коду, що забезпечує стабільну роботу навіть на пристроях із низькою швидкістю інтернет-з'єднання. Також широко використовуються технології прогресивних веб-додатків (Progressive Web Apps, PWA), що дозволяють користувачам отримувати доступ до контенту навіть без активного інтернет-з'єднання завдяки кешуванню ресурсів [9].

Особлива увага приділяється питанням безпеки адаптивних веб-застосунків. Освітні платформи часто стикаються з такими загрозами, як несанкціонований доступ, витік персональних даних, DDoS-атаки та шкідливий код. Тому застосовуються сучасні протоколи та методи шифрування даних, такі як HTTPS, інтегрується багатофакторна аутентифікація та регулярно проводиться аудит безпеки, що дозволяє надійно захищати конфіденційну інформацію користувачів.

HTTP - це протокол передачі даних, на основі якого працює всесвітня павутина. Завдяки цьому протоколу ми можемо заходити на сайти в браузері та взаємодіяти з ними: переходити з однієї сторінки на іншу, завантажувати файли та переглядати зображення, обмінюватися повідомленнями та оплачувати покупки.

Адаптивні веб-застосунки дозволяють освітнім закладам значно розширити свої можливості в онлайн-освіті, забезпечуючи доступність навчального контенту широкій аудиторії, комфортну взаємодію з навчальними ресурсами та високий рівень безпеки і надійності.

1.3. Методи генерації та перевірки тестів онлайн

Методи генерації та перевірки тестів є важливими елементами для ефективного функціонування систем онлайн-тестування. Вони дозволяють автоматизувати процес контролю знань, забезпечуючи швидкість, якість і точність результатів.

Існує кілька основних підходів до створення тестових завдань. Перший і найтрадиційніший -це ручне формування тестів, коли викладач або адміністратор самостійно розробляє питання, базуючись на змісті навчальних матеріалів. Наприклад, такий підхід може бути особливо ефективним при підготовці до підсумкової атестації або при розробці тестів з гуманітарних дисциплін, де важливо врахувати нюанси формулювання запитань і відповідей. Даний метод дозволяє детально контролювати зміст, складність і тематичну спрямованість завдань. Однак він є досить трудомістким, особливо при великій кількості користувачів або великому обсязі матеріалів, що може знизити оперативність проведення тестування. Утім, існують інструменти, які частково автоматизують процес створення тестів - наприклад, шаблони, майстри формування запитань або системи підтримки банку питань, що дозволяють зменшити навантаження на викладача без втрати якості змісту.

Другим поширеним методом є автоматична генерація тестів. Цей підхід передбачає випадковий вибір питань із заздалегідь підготовленого банку завдань. Перевагою автоматичної генерації є забезпечення унікальності та варіативності тестових завдань для кожного користувача, зменшення ймовірності повторення питань та списування. Однак недоліком цього методу може бути ризик втрати контролю над точністю та якістю підібраних завдань, оскільки випадковий вибір не завжди враховує оптимальну логічну послідовність питань або рівномірний розподіл за складністю. Попри це, автоматична генерація значно економить час викладачів та адміністраторів,

дозволяючи більше зосередитись на якості навчальних матеріалів та аналізі результатів тестування.

Ще одним важливим і перспективним методом є адаптивна генерація тестів. При цьому підході питання для кожного користувача підбираються відповідно до результатів його попередніх відповідей. Це означає, що рівень складності завдань динамічно змінюється залежно від успішності користувача в процесі проходження тесту. Така технологія дозволяє максимально точно визначити рівень знань, персоналізувати навчальний процес, підвищити мотивацію та ефективність навчання.

Наприклад, адаптивне тестування є особливо ефективним у вивченні іноземних мов, де система може поступово ускладнювати граматичні чи лексичні завдання відповідно до рівня володіння мовою, або у математичних дисциплінах, де поступовий перехід від базових до складніших понять дозволяє краще зрозуміти індивідуальні потреби кожного учня.

Що стосується методів перевірки тестів, існує кілька варіантів, які можуть використовуватись окремо або у комплексі. Автоматична перевірка використовується для закритих типів завдань, таких як завдання з вибором відповіді, встановлення відповідності чи послідовності. Головною перевагою автоматичної перевірки є її швидкість і повна об'єктивність, що дозволяє миттєво отримувати результати і оперативно реагувати на них. Однак варто враховувати, що у випадку неоднозначних формулювань або завдань із подібними за змістом варіантами відповідей, автоматична перевірка може виявитися недостатньо гнучкою, оскільки не завжди здатна коректно інтерпретувати логіку міркувань користувача. Тому під час проєктування тестів такого типу важливо ретельно опрацьовувати формулювання, щоб уникнути неоднозначності.

Для перевірки відкритих питань часто застосовується напівавтоматична перевірка, яка передбачає автоматичний аналіз відповідей за ключовими словами або фразами, такими як визначення, основні терміни, ключові поняття або дати подій. Цей метод потребує подальшого підтвердження чи

коригування результатів викладачем чи адміністратором, що дозволяє скоротити час перевірки та враховувати індивідуальні особливості та нюанси відповідей користувачів, які важко врахувати автоматично.

Одним з найсучасніших і інноваційних підходів є застосування штучного інтелекту для аналізу відкритих відповідей. Використання алгоритмів обробки природної мови (Natural Language Processing, NLP) дозволяє проводити глибокий і детальний аналіз текстових відповідей, розуміти контекст і значення сказаного, а також значно знижувати суб'єктивність оцінювання. Популярні сервіси, що використовують такі алгоритми -Grammarly, IBM Watson і Turnitin, які ефективно застосовуються для автоматизованого оцінювання складних текстів у навчальних середовищах. Водночас, застосування штучного інтелекту має певні обмеження: алгоритми можуть не враховувати емоційне забарвлення, іронію або нестандартні формулювання, що характерні для відкритих відповідей.

Також існують питання етичного характеру, пов’язані з конфіденційністю даних та прозорістю алгоритмів оцінювання, які потребують ретельного регламентування при впровадженні таких технологій у навчальні процеси. Для мінімізації цих ризиків рекомендується застосовувати локальне зберігання даних, впроваджувати політики обмеженого доступу, а також використовувати відкриті або прозоро задокументовані алгоритми, щоб забезпечити довіру користувачів.

Таблиця 1.1 Методи перевірки тестових завдань та їх особливості

Метод перевірки	**Тип завдання**	**Особливості та переваги**
Автоматична перевірка	Закриті питання, відповідності	Швидкість, об'єктивність, миттєві результати
Напівавтоматична перевірка	Відкриті питання	Часткова автоматизація, необхідність коригування
Перевірка із застосуванням ШІ	Відкриті складні питання	Глибокий аналіз, контекстуальність,

		зменшення суб'єктивності

Правильний вибір методів генерації та перевірки тестів залежить від потреб освітнього процесу, технологічних можливостей освітньої установи та специфіки предмету чи курсу. Це дозволить забезпечити не тільки ефективність навчання, але й високий рівень мотивації користувачів завдяки точному та об'єктивному оцінюванню їхніх знань і навичок.

1.4. Порівняльний аналіз технологій для розробки адаптивних систем

Вибір технологій є одним із ключових факторів при розробці адаптивних веб-застосунків, оскільки саме вони визначають продуктивність, зручність використання та можливості масштабування проєкту. Розглянемо сучасні інструменти для фронтенд- та бекенд-розробки, що активно застосовуються в розробці адаптивних систем.

Технології фронтенд-розробки:

React - один із найпопулярніших фреймворків, який забезпечує високу продуктивність завдяки віртуальному DOM. Використання цього підходу дозволяє оновлювати лише ті частини інтерфейсу, які змінилися, значно прискорюючи завантаження сторінок. React підтримує велика активна спільнота, що надає багато готових компонентів і бібліотек, полегшуючи процес розробки [10].

Angular - комплексний фреймворк, створений компанією Google. Він містить у собі широкі стандартні можливості: керування станом додатку, маршрутизацію, обробку форм та валідацію даних. Angular добре структурований, що робить його особливо ефективним для великих та складних проєктів. Водночас, одним із недоліків цього фреймворку є високий поріг входження, який може створювати труднощі для початківців [11].

Vue.js характеризується простотою та гнучкістю. Він дозволяє швидко створювати як прості прототипи, так і повноцінні додатки з адаптивним інтерфейсом. Vue.js є чудовим рішенням для малих і середніх проєктів

завдяки низькому порогу входження та швидкій адаптації розробників до його функціоналу [12].

Технології бекенд-розробки:

Node.js - платформа, яка дозволяє використовувати JavaScript як на фронтенді, так і на бекенді. Головною перевагою Node.js є асинхронність, завдяки якій досягається висока швидкість роботи додатків із великою кількістю одночасних з'єднань. Відомими додатками на Node.js є Netflix, Uber та LinkedIn, що підтверджує надійність цієї технології у високонавантажених умовах [13].

Django - повноцінний фреймворк на мові Python, який пропонує готові рішення для автентифікації користувачів, адміністративної панелі, роботи з базами даних і безпеки інформації. Django найбільше підходить для великих і складних додатків, де важлива організація великого обсягу даних і стабільна робота [14].

Flask - легкий мікрофреймворк, також на Python, що добре підходить для невеликих та середніх додатків. Flask надає гнучкість у налаштуванні та легко розширюється завдяки модульній структурі. Це робить його оптимальним для експериментальних проєктів, які можуть швидко змінюватись у процесі розробки [15].

Таблиця 2.1 Порівняльний аналіз технологій

Технологія	**Тип**	**Складність освоєння**	**Масштабованість**	**Сфера використання**
React	Фронтенд	Середня	Висока	Швидка розробка, висока продуктивність
Angular	Фронтенд	Висока	Висока	Великі, складні проєкти, командна робота
Vue.js	Фронтенд	Низька	Середня	Малі та середні проєкти, швидке прототипування
Node.js	Бекенд	Середня	Висока	Високонавантажені додатки, реальний час
Django	Бекенд	Висока	Висока	Великі, складні додатки, бази даних
Flask	Бекенд	Низька	Середня	Малі та середні додатки, швидке налаштування

Отже, правильний вибір технологій залежить від особливостей проєкту, кваліфікації команди, доступних ресурсів і планів щодо масштабування застосунку. Врахування цих факторів дозволяє створити якісний, зручний і ефективний адаптивний веб-застосунок.

РОЗДІЛ 2

ПРОЕКТУВАННЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ СТВОРЕННЯ ТА АДМІНІСТРУВАННЯ ТЕСТІВ

2.1. Аналіз вимог до адаптивного веб-застосунку для освітніх тестів

У сучасному освітньому середовищі онлайн-тестування стало важливим інструментом для контролю знань та оцінювання навчальних досягнень. З огляду на динамічний розвиток цифрових технологій, виникає потреба у створенні веб-застосунків, які є не лише функціональними, але й зручними, надійними та безпечними. Адаптивні веб-застосунки повинні відповідати широкому спектру вимог, щоб забезпечити ефективність та гнучкість в умовах сучасного навчального процесу.

Функціональні вимоги охоплюють базову поведінку системи. Веб-застосунок повинен забезпечувати створення, редагування та збереження тестів з можливістю додавання різних типів питань: вибір правильної відповіді, встановлення відповідності, заповнення пропусків, відкрита відповідь тощо. Користувачі повинні мати змогу реєструватися, авторизуватись, обирати тест зі списку доступних, проходити його та переглядати результати. Система повинна автоматично зберігати відповіді й результати тестування, надаючи викладачу змогу їх аналізувати.

Нефункціональні вимоги визначають технічні характеристики системи. Однією з основних є продуктивність: усі сторінки повинні завантажуватись швидко, а взаємодія користувача із застосунком має відбуватись без затримок. Це стосується генерації тестових завдань, перевірки відповідей, формування результатів. Масштабованість системи дозволяє використовувати її у великих освітніх закладах або під час масового доступу, не втрачаючи стабільності роботи.

Безпека користувацьких даних - ключовий аспект у проєктуванні освітніх систем. Застосунок має використовувати шифрування даних під час

зберігання та передачі. Автентифікація користувачів повинна включати перевірку облікових записів, а для викладачів або адміністраторів - додаткові рівні доступу. Важливою вимогою є збереження цілісності результатів тестування та протидія спробам маніпуляцій, що може бути реалізовано шляхом обмеження кількості спроб, таймерів на виконання, генерації унікальних варіантів тестів.

Адаптивність інтерфейсу є необхідною умовою для зручного доступу користувачів з різних пристроїв. Інтерфейс повинен коректно відображатися на настільних комп'ютерах, планшетах і смартфонах, підтримуючи респонсивну верстку. При цьому зберігається повна функціональність системи: кнопки, форми, поля вводу, меню повинні бути зручними незалежно від розміру екрана. Особливо важливо врахувати навігацію на сенсорних пристроях.

Інтерактивність та зворотний зв'язок підвищують мотивацію користувача. Після завершення тесту система повинна надати результати з поясненнями або коментарями до помилок. Це може включати виділення правильних відповідей, посилання на додаткові матеріали, графічне відображення прогресу. Для адміністратора варто передбачити доступ до аналітики: середній бал, кількість спроб, типові помилки тощо. Застосунок має дозволяти створення адаптивних тестів, у яких складність завдань змінюється залежно від відповідей користувача.

Інтеграція з іншими освітніми платформами дозволяє підвищити функціональність системи. Необхідна підтримка стандартів обміну даними, наприклад LTI, SCORM або API для взаємодії з LMS, такими як Moodle, Google Classroom, Microsoft Teams. Результати тестування мають експортуватися у зручних форматах (CSV, XLSX, PDF) для подальшої обробки або зберігання в архіві.

З урахуванням широкої аудиторії, важливо також враховувати вимоги доступності: інтерфейс має бути придатним для використання людьми з

порушеннями зору, слуху або моторики. Це включає підтримку масштабування, озвучування елементів, навігацію з клавіатури тощо.

Аналіз вимог до адаптивного веб-застосунку для освітніх тестів дозволяє визначити необхідні технічні, функціональні та інтерфейсні характеристики майбутньої системи. Веб-застосунок повинен бути швидким, адаптивним, зручним у використанні, безпечним і гнучким для інтеграції.

Реалізація усіх перелічених вимог є запорукою якості та ефективності розробленої системи, що забезпечить комфорт як для студентів, так і для викладачів у процесі освітнього тестування.

2.2. Проектування логіки роботи адаптивного веб-застосунку

Функціонування системи ґрунтується на поетапній взаємодії між користувачем, сервером і базою даних. Основні дії включають завантаження тестів, відповіді користувача, перевірку на сервері та збереження результатів.

Нижче наведено базову логіку роботи веб-застосунку під час проходження тесту:

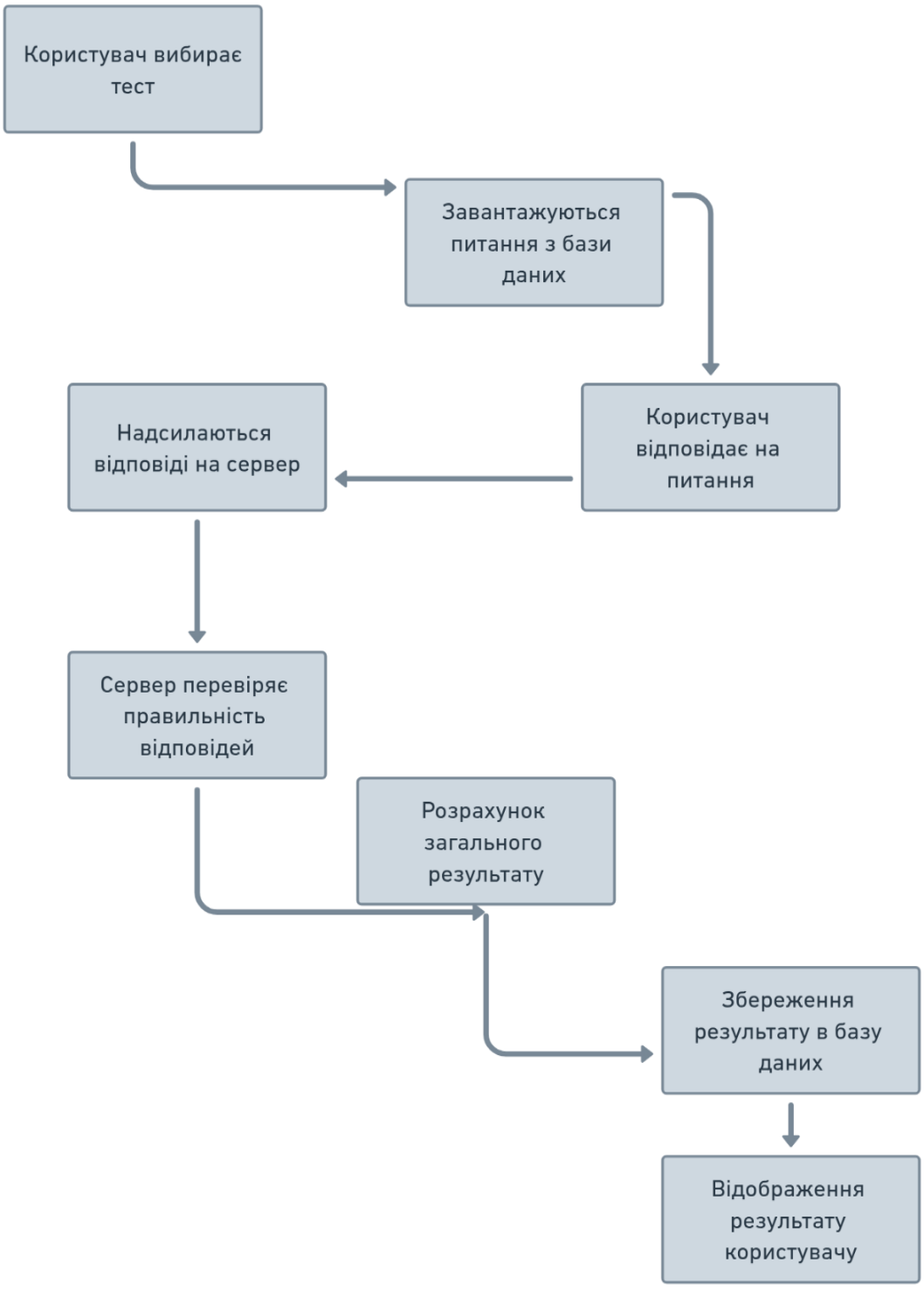

Рис. 2.1 -Логіка взаємодії користувача з веб-застосунком під час проходження тесту

Пояснення етапів логіки роботи системи:

1. Користувач заходить у систему та обирає зі списку доступних тестів той, який хоче пройти.
2. Веб-застосунок надсилає запит до сервера, який у свою чергу звертається до бази даних та завантажує питання, пов’язані з вибраним тестом.
3. Питання відображаються на інтерфейсі користувача, і він починає відповідати на них у зручному форматі (текстові, варіант вибору, drag-and-drop тощо).
4. Після завершення тесту відповіді користувача надсилаються назад на сервер, де здійснюється перевірка на правильність.
5. Сервер розраховує загальний результат, враховуючи кількість правильних відповідей, кількість спроб, складність запитань (за потреби - адаптивну логіку).
6. Результат зберігається в базу даних користувача.
7. Після збереження результатів користувачеві миттєво відображається підсумкова оцінка разом із додатковою інформацією (пояснення, розбір помилок, поради).

Проєктування логіки адаптивного веб-застосунку дозволяє чітко структурувати весь процес проходження тестів. Представлений Рис. забезпечує розуміння ключових етапів та ролі кожного компоненту в системі. Це дозволяє не лише оптимізувати процес реалізації, а й забезпечити зручність, точність та ефективність взаємодії з користувачем.

2.3. Архітектура веб-застосунку для адаптивного тестування

Проєктування архітектури веб-застосунку є одним із ключових етапів розробки, оскільки саме вона визначає структуру, взаємозв’язки між модулями та розподіл функціональності системи. Для адаптивної системи онлайн-тестування важливо забезпечити не лише логічний розподіл обов’язків між компонентами, але й стабільність, масштабованість та

гнучкість. Веб-застосунок повинен мати модульну структуру, щоб полегшити оновлення, відлагодження та подальше розширення функціоналу без необхідності переробки всієї системи. У цьому підрозділі розглядається архітектура серверної частини веб-застосунку, а також допоміжні механізми забезпечення надійної роботи з базою даних.

Система будується за принципом багаторівневої архітектури, яка передбачає розділення відповідальності між компонентами. Основні рівні включають контролери (API), бізнес-логіку (сервіси) та рівень доступу до даних (репозиторії або ORM). Такий підхід дозволяє чітко структурувати програмний код, уникати дублювання логіки та підвищувати якість підтримки проєкту.

На наступному рисунку зображено логіку взаємодії між компонентами серверної частини веб-застосунку, а також інструменти для забезпечення моніторингу та резервування бази даних.

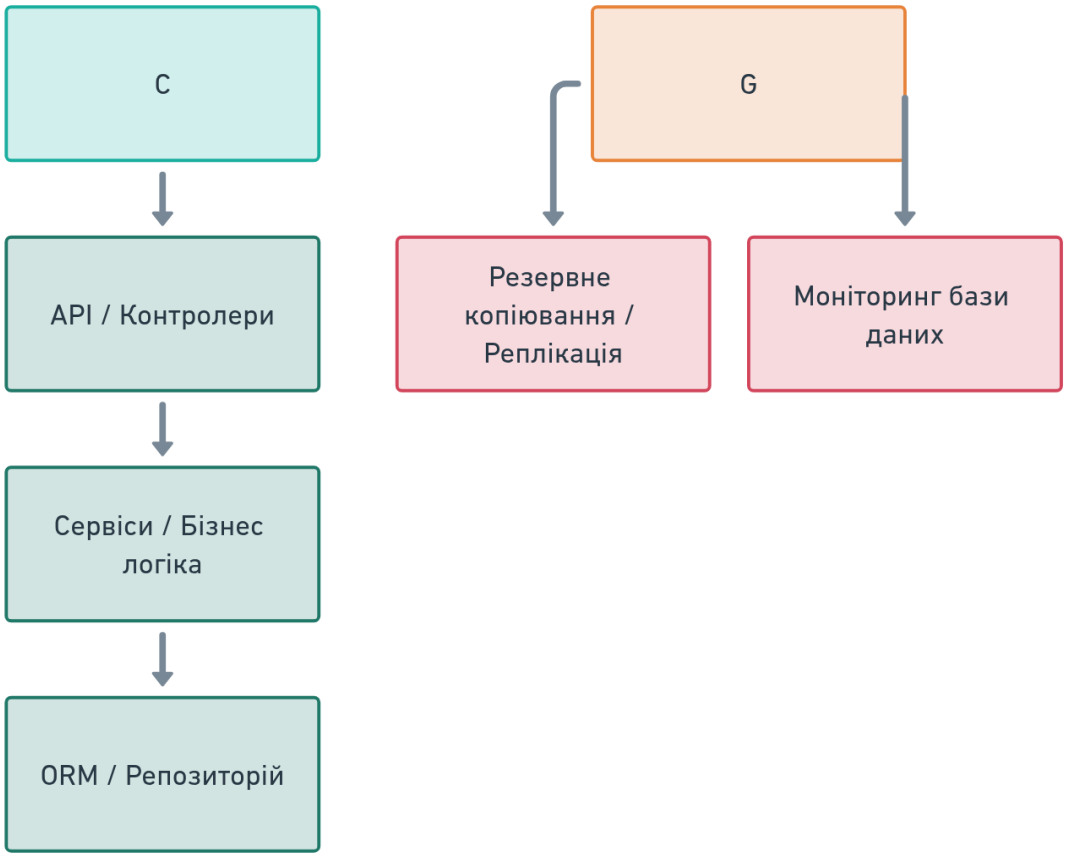

Рис. 2.2 -Структура логіки застосунку та обробки даних на сервері

Пояснення компонентів:

1. Контролери (API) відповідають за обробку HTTP-запитів від клієнта. Вони отримують дані, перевіряють їх на коректність та передають далі у відповідні сервіси. Наприклад, якщо користувач надсилає відповіді на тест, саме контролер приймає ці дані та ініціює процес перевірки.
2. Сервіси (Бізнес-логіка) реалізують правила роботи застосунку. Тут відбувається основна обробка запитів: перевірка правильності відповідей, обрахунок балів, визначення рівня складності наступного завдання, формування результату. Саме цей рівень можна адаптувати для реалізації адаптивної логіки - наприклад, змінювати питання відповідно до попередніх відповідей користувача.
3. Репозиторії / ORM (Object-Relational Mapping) відповідають за доступ до бази даних. Вони ізолюють прямі SQL-запити від решти частини програми, забезпечуючи чисту й контрольовану взаємодію з таблицями. Це спрощує зміну структури БД у майбутньому без значного впливу на інші модулі.
4. G (Блок обслуговування БД) включає два важливі процеси - резервне копіювання/реплікація та моніторинг бази даних. Резервне копіювання гарантує, що дані користувачів не буде втрачено в разі збою. Реплікація дає змогу розподіляти навантаження або створювати копії даних для читання. Моніторинг бази даних дозволяє своєчасно виявляти навантаження, відставання, помилки або спроби несанкціонованого доступу, що підвищує загальну надійність системи.

Цей підхід також сприяє масштабуванню - наприклад, можна винести базу даних на окремий сервер, розділити API на мікросервіси, або підключити сторонні сервіси аналітики через зовнішні API.

Важливо, що подібна архітектура дає змогу паралельної розробки кількома командами - фронтенд і бекенд розробляються незалежно, а тестування та впровадження нових функцій відбуваються модульно, без впливу на основний код.

Архітектура дозволяє чітко структурувати застосунок, розмежовуючи зони відповідальності. Вона сприяє масштабованості, полегшує тестування й супровід системи. Стабільності бази даних, є критичним аспектом для платформи онлайн-тестування. Такий підхід дозволяє ефективно реалізувати адаптивну систему, здатну до подальшого розширення та інтеграції. Завдяки дотриманню принципів модульності та ізоляції компонентів система набуває гнучкості, що особливо важливо в умовах динамічних змін в освітньому середовищ.

2.4. Модуль адміністрування та аналізу результатів

Адміністративна частина веб-застосунку відіграє ключову роль у процесі організації та керування навчальним контентом. Основна відповідальність адміністратора полягає у створенні тестів, управлінні користувачами та здійсненні контролю за процесами оцінювання. Цей модуль має бути логічно побудований, інтуїтивно зрозумілим, а його функціональні можливості - відповідати сучасним освітнім вимогам. Теоретичною основою адміністрування є концепція системного підходу до управління навчальними процесами, де адміністратор виступає як координатор освітнього середовища в межах системи.

Відповідно до принципів освітнього менеджменту, адміністратор має мати змогу впливати на контент, налаштовувати доступи, формувати статистику і здійснювати рефлексію процесів навчання. Веб-застосунок, що відповідає цим завданням, повинен реалізовувати логічну структуру адміністрування з урахуванням ролей, прав доступу, керованості та звітності.

Після входу до системи адміністратор отримує доступ до панелі керування, яка є центральним елементом адміністративного модуля. Тут реалізовано можливості зі створення нових тестів, редагування наявних, додавання запитань, керування варіантами відповідей і збереження всіх змін у базу даних. Призначення тестів певним користувачам, формування груп

доступу, визначення строків і кількості спроб - усе це адміністратор здійснює через відповідні інтерфейсні модулі.

Суттєвим компонентом є керування користувачами. Адміністратор може створювати нові облікові записи, редагувати ролі, обмежувати або надавати доступ до певних ресурсів. Користувачі можуть бути згруповані за категоріями, що дозволяє швидко здійснювати призначення тестів, формувати вибірки результатів і аналізувати ефективність навчання в межах певної групи.

Модуль перегляду результатів тестування реалізовано через гнучку систему фільтрації. Адміністратор має змогу вибирати тести, переглядати результати за конкретним користувачем або групою, аналізувати типові помилки, кількість спроб і прогрес користувачів. Усі дані зберігаються в базі й доступні для подальшого аналізу або експорту. Узагальнена схема функціональності адміністратора наведена нижче.

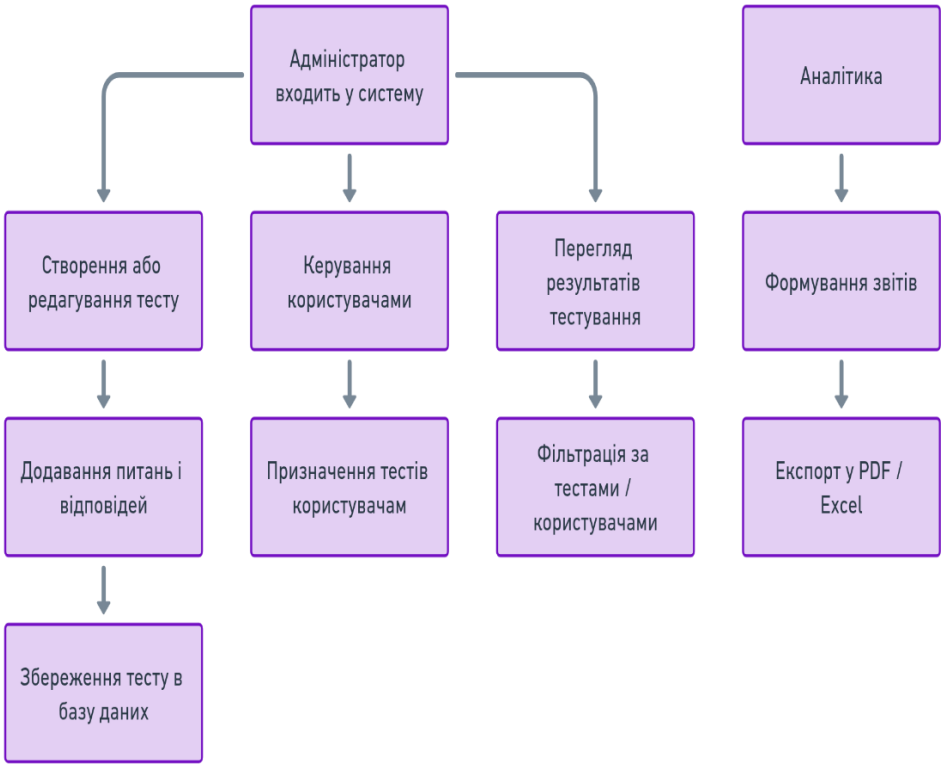

Рис. 2.3 - Функціональна схема адміністративного модуля веб-застосунку

Систематичне створення тестів - одна з основних функцій адміністратора. Починаючи з введення загальної інформації, адміністратор додає запитання, визначає правильні відповіді, налаштовує параметри проходження та публікує тест. Процес має бути інтуїтивно зрозумілим, а інтерфейс - таким, що підтримує всі типи запитань.

Нижче представлено покрокову візуалізацію процесу:

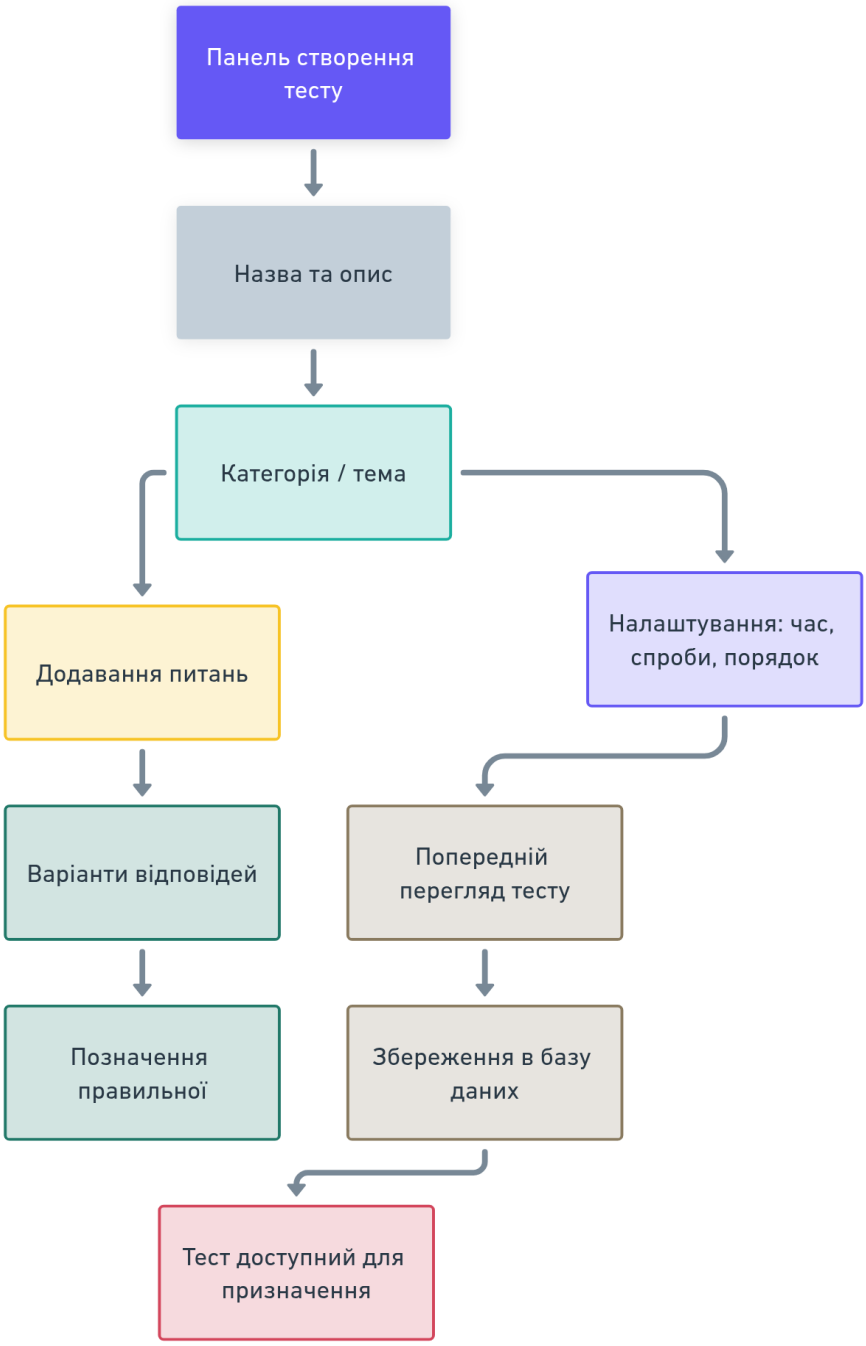

Рис. 2.4 –процес створення тесту адміністратором

Цей процес передбачає введення назви, опису, вибір тематики, додавання запитань, варіантів відповідей, визначення правильних варіантів, встановлення параметрів часу, кількості спроб, а також перегляд і збереження готового тесту до бази. Після цього тест стає доступним для користувачів відповідно до налаштувань доступу.

Після проходження тестів адміністратор може здійснювати повноцінний аналіз результатів, що є основою для прийняття рішень щодо змісту освітніх програм, методів викладання чи зміни системи оцінювання. Система підтримує фільтрацію, порівняння, формування динаміки змін у знаннях користувачів. Аналітичні можливості забезпечуються як табличними, так і візуальними засобами - графіками, діаграмами, хмарами слів тощо. Результати можуть експортуватися в різні формати, що дозволяє адаптувати їх під вимоги звітності.

Також важливо, що модуль може підтримувати формування агрегованих звітів за періоди, категоріями тестів або рівнями складності. Це дозволяє не лише оцінювати індивідуальні досягнення, але й здійснювати загальний контроль ефективності навчання в межах курсу чи програми.

Адміністративний модуль адаптивного веб-застосунку є ядром управління системою. Його функціонал охоплює весь спектр дій - від створення контенту до глибокої аналітики. Забезпечення зручності, надійності та прозорості адміністрування - передумова якісної роботи всієї освітньої платформи. Використання теоретичних засад управління знаннями та освітнього менеджменту дозволяє реалізувати системний підхід до адміністрування, що забезпечує ефективне керування навчальним процесом у цифровому середовищі.

РОЗДІЛ 3

РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ ТА ТЕСТУВАННЯ ЙОГО ФУНКЦІОНАЛЬНОСТІ

3.1. Реалізація користувацького інтерфейсу для створення тестів та адміністрування

Розробка адміністративного інтерфейсу веб-застосунку здійснювалася у середовищі Visual Studio Code використанням Node.js для реалізації серверної логіки та React.js - для побудови клієнтської частини. Такий підхід забезпечив розмежування відповідальностей між frontend та backend, підвищив гнучкість архітектури та спростив масштабування системи [Див. Рис. 1.1 у додатку А). Додатково було використано бібліотеки Material UI та Framer Motion, які дозволили створити сучасний, адаптивний та візуально привабливий інтерфейс, із підтримкою анімацій та інтерактивних компонентів.

На початковому етапі було налаштоване середовище розробки у Visual Studio Code, в якому реалізовано структуру проєкту у вигляді окремих папок та файлів для компонентів, стилів, маршрутизації, службових функцій і налаштувань. Це сприяло зрозумілій навігації по проєкту та спростило подальший супровід. Серверна частина обробляє запити до бази даних, авторизацію користувачів та збереження результатів проходження тестів. Клієнтська частина у React працює як **SPA (Single Page Application)**, забезпечуючи швидке оновлення контенту без перезавантаження сторінки та плавну взаємодію з користувачем.

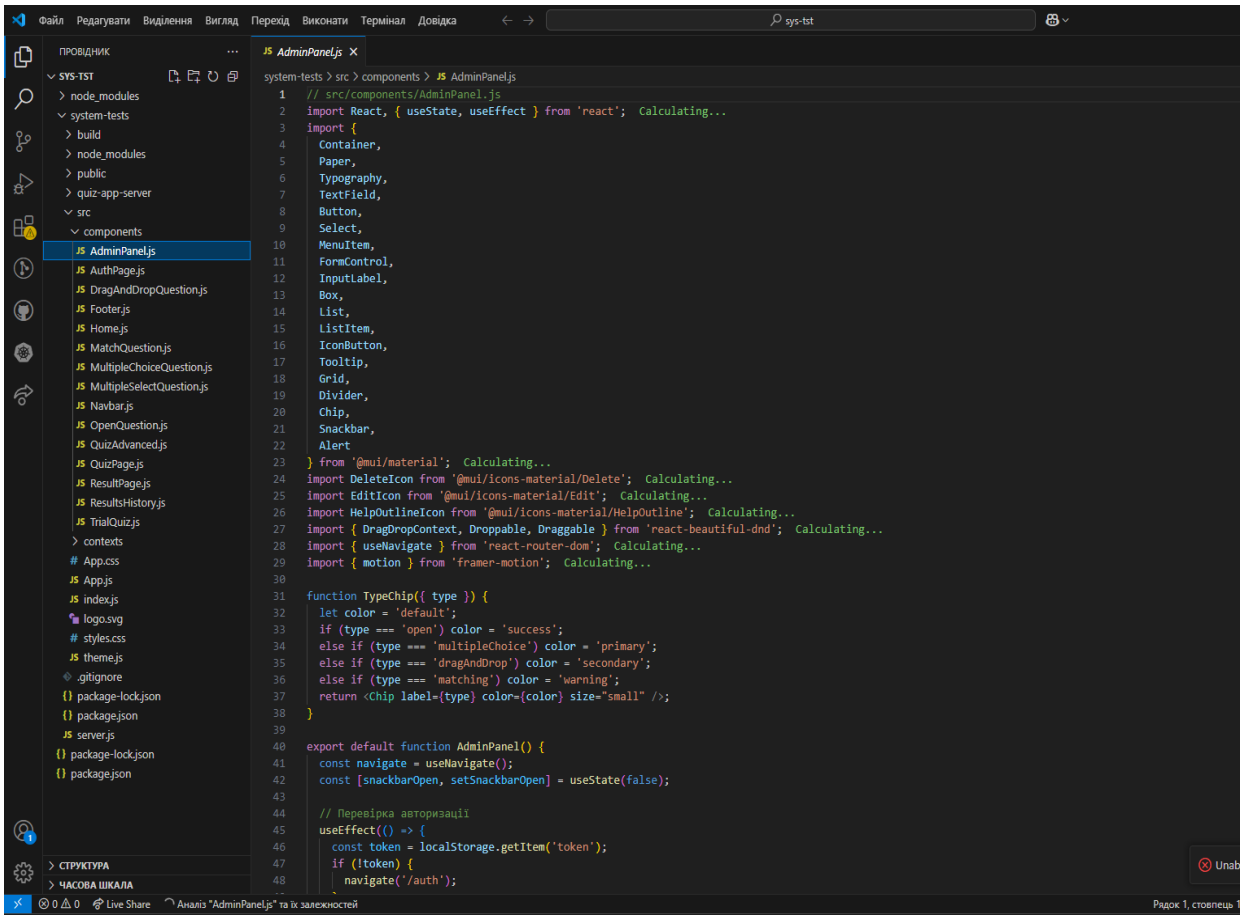

Рис. 3.0 -Середовище розробки у Visual Studio Code з відкритим проєктом

Після запуску веб-застосунку користувач потрапляє на головну сторінку, яка виконує роль навігаційного центру. Звідси користувач може перейти до проходження тестів, перегляду результатів або, у разі авторизації, до адміністративного інтерфейсу для створення та керування тестами.

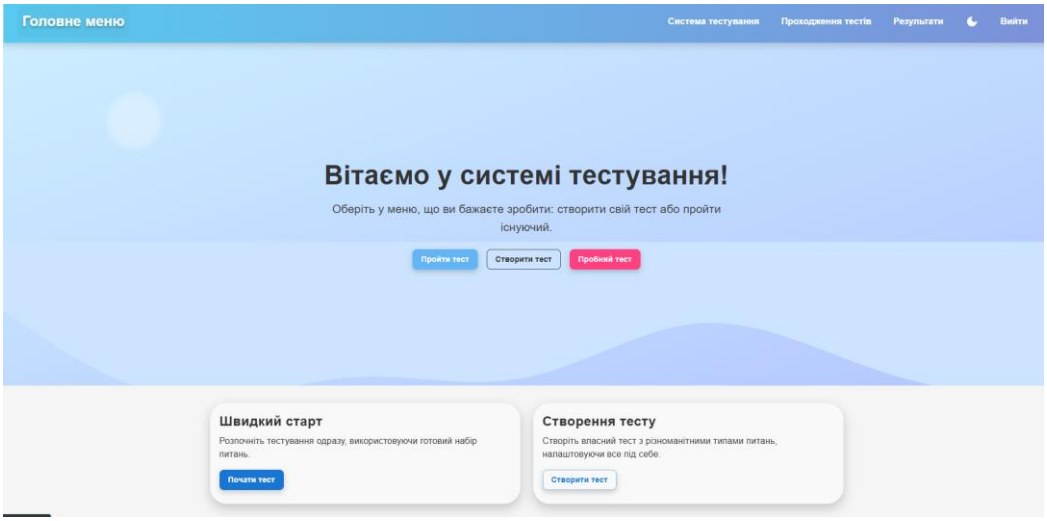

Рис. 3.1 -Головна сторінка веб-застосунку

Інтерфейс адміністратора дозволяє повноцінно керувати процесом створення тестових запитань. Після авторизації (перевірки токена у localStorage), адміністратор отримує доступ до функціоналу, який дозволяє створювати нові питання, обирати тип питання, редагувати та видаляти вже наявні запитання. Форма динамічно змінюється залежно від обраного типу запитання: відкриті, з вибором, на відповідність або типу drag-and-drop.

Використання локального сховища браузера (localStorage) забезпечує збереження створених запитань між сесіями, дозволяє редагувати та доповнювати тести в зручному режимі. Кожне питання серіалізується у форматі JSON, що забезпечує гнучке зберігання та обробку. Додавання запитань супроводжується миттєвим оновленням інтерфейсу без потреби взаємодії з сервером, що дозволяє проводити попередню підготовку тестів в автономному режимі.

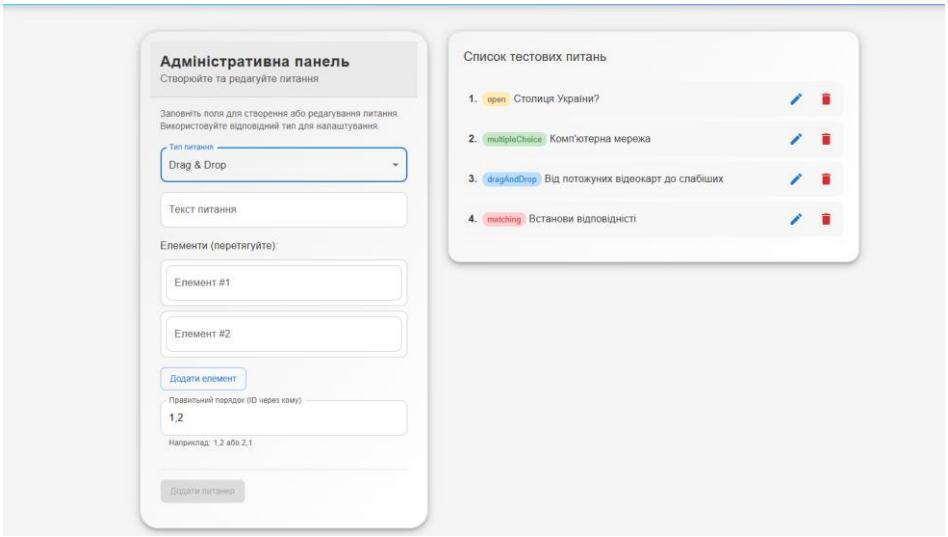

Рис. 3.2 -Загальний вигляд адміністративної панелі

Інтерфейс включає механізм редагування та перегляду питань у вигляді списку, кожне з яких можна відкрити для редагування натисканням на відповідну іконку. Зміни, внесені у форму, зберігаються натисканням кнопки, а відповідний об'єкт оновлюється у localStorage. Видалення питань також реалізовано з можливістю миттєвого оновлення інтерфейсу. Усі операції відбуваються без перезавантаження сторінки.

Під час створення запитань типу drag-and-drop застосовується бібліотека react-beautiful-dnd, яка дозволяє візуально перетягувати елементи, змінюючи їх порядок. Кожна зміна супроводжується плавною анімацією, що покращує взаємодію з інтерфейсом.

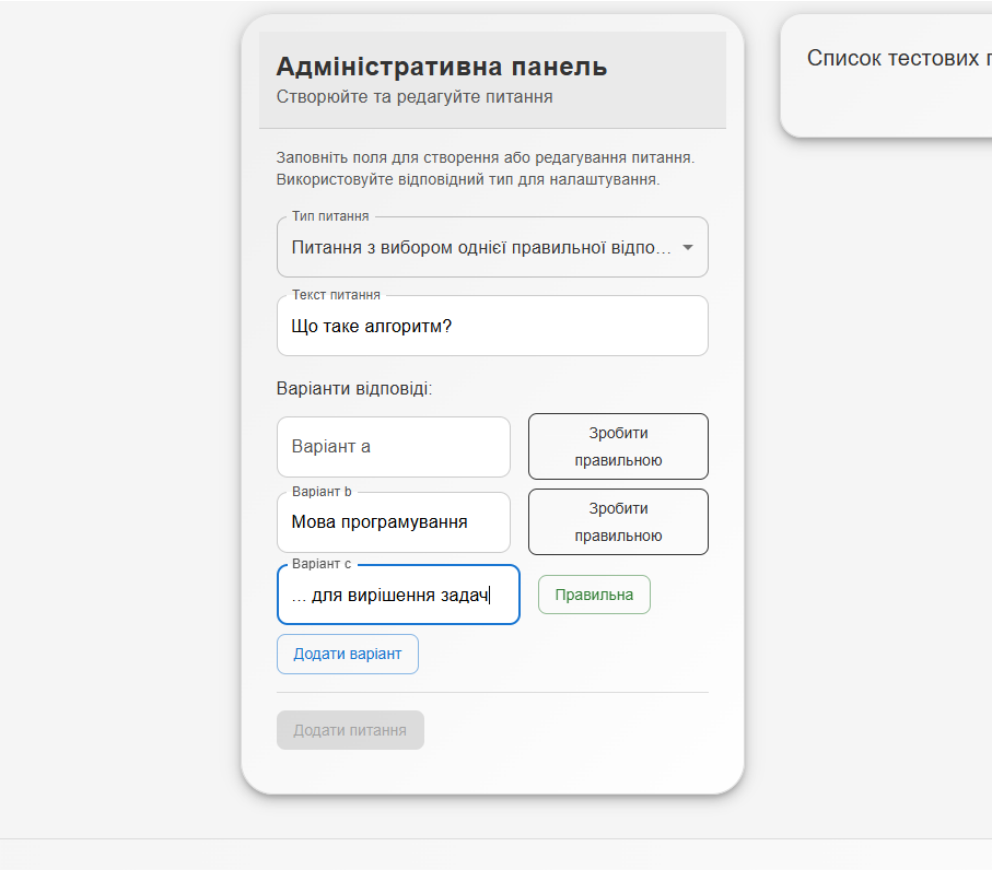

Рис. 3.3 - Приклад створення запитання з множинним вибором

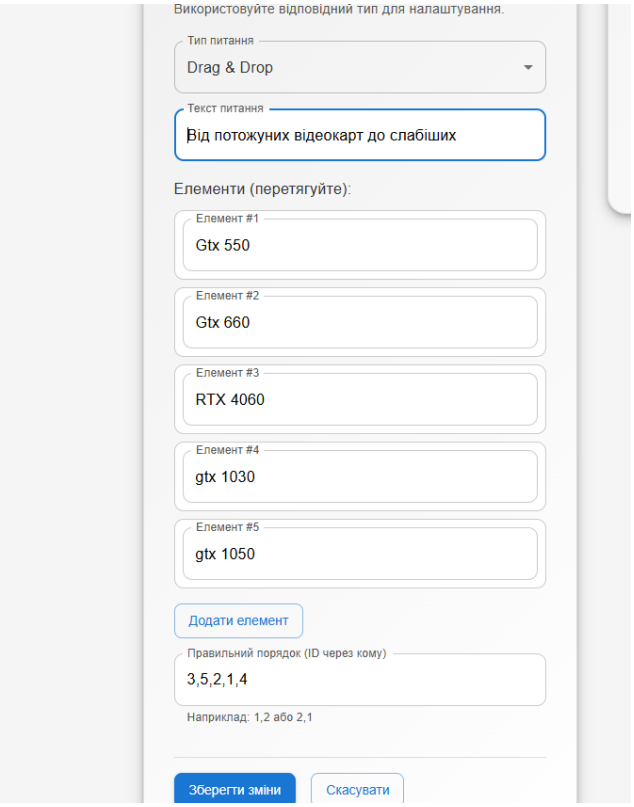

Рис. 3.4 - Редактор запитань з типом «Перетягування»

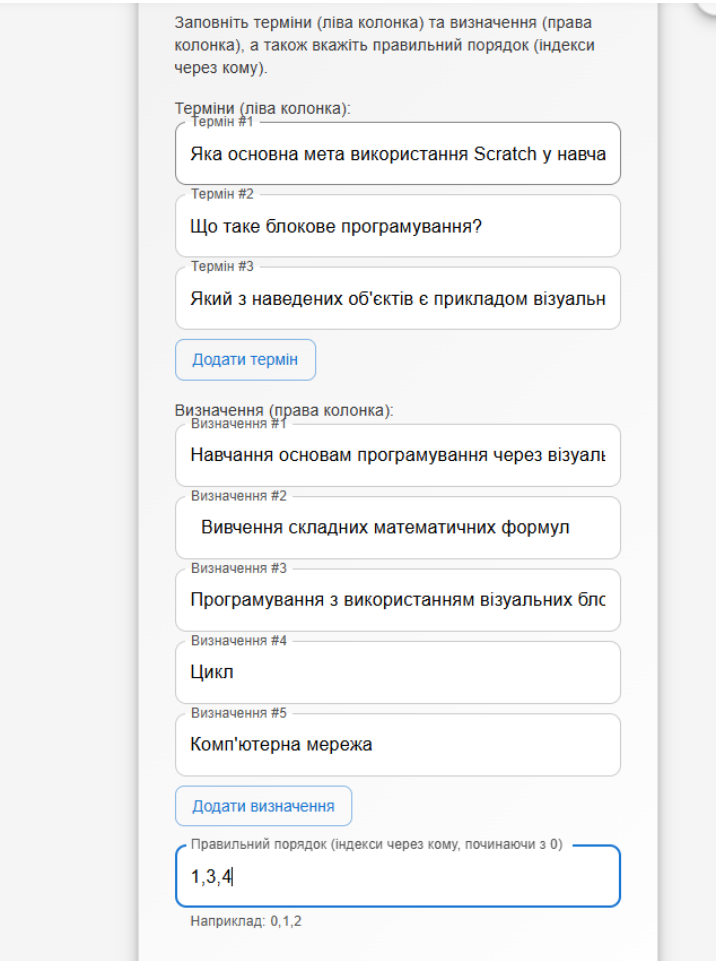

Рис. 3.5 - Приклад інтерфейсу для питань на відповідність

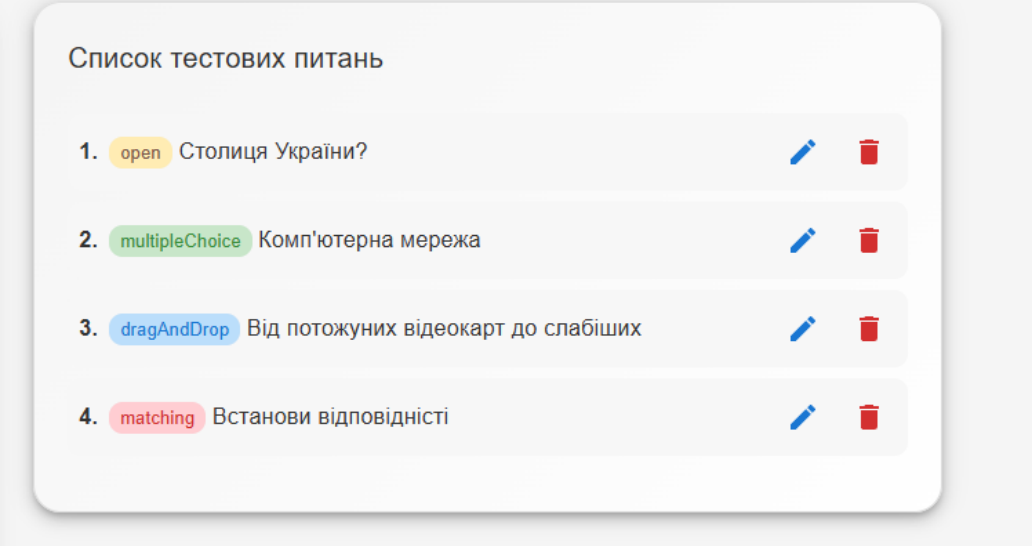

Рис. 3.6 - Список створених запитань адміністратора

Інтерфейс враховує принципи адаптивного дизайну: компоненти масштабуються відповідно до ширини екрана, забезпечуючи зручну взаємодію як на мобільних пристроях, так і на десктопах. Framer Motion забезпечує плавні анімаційні ефекти під час завантаження та взаємодії з елементами: появу блоків, кнопок, списків [17]. Це додає інтерактивності та підвищує загальну якість UX.

Окрему роль відіграють інформаційні підказки, що допомагають орієнтуватися в інтерфейсі: при наведенні курсору на іконку з’являється коротке пояснення до поля чи дії. Повідомлення про успішне збереження реалізовано за допомогою **Snackbar** - ненав’язливого спливаючого повідомлення, яке автоматично зникає через кілька секунд і не заважає подальшій роботі користувача [18].

Таким чином, інтерфейс адміністративного модуля реалізовано відповідно до сучасних вимог до UX/UI: зручність, інтерактивність, гнучкість, безпека та розширюваність. У поєднанні з адаптивним дизайном та візуальними ефектами інтерфейс забезпечує ефективну взаємодію

користувача з системою тестування, роблячи процес створення тестів інтуїтивно зрозумілим і технологічно якісним.

3.2. Впровадження алгоритмів для перевірки відповідей і генерації результатів

На цьому етапі розробки веб-застосунку було реалізовано повноцінний механізм перевірки відповідей користувача на тестові запитання, а також детальну логіку формування та виводу результатів проходження тесту. Увесь функціонал було розроблено у середовищі Visual Studio Code із використанням середовища виконання Node.js на серверній стороні та бібліотеки React.js - на клієнтській. Подібне архітектурне рішення дало змогу забезпечити ефективну взаємодію між клієнтською і серверною частинами, прискорити обмін даними та гарантувати надійну і точну обробку відповідей у реальному часі.

Тестування розпочинається зі стартової сторінки, що виконує роль психологічної та інформаційної підготовки користувача. Тут міститься коротке повідомлення про початок тесту та кнопка, що активує старт процесу.

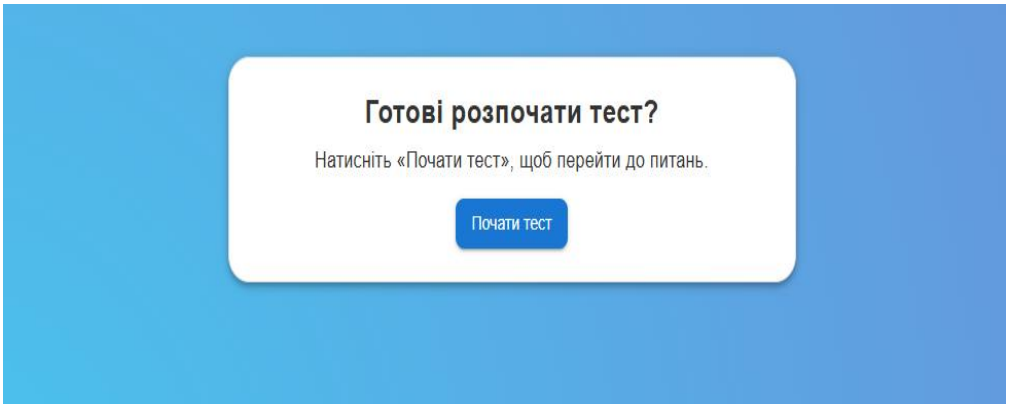

Рис. 3.7 - Стартова сторінка тесту

Після натискання на кнопку «Почати тест», користувач переходить до першого запитання. Залежно від типу запитання, інтерфейс змінюється відповідно до сценарію взаємодії. Усі типи запитань мають окремі алгоритми

перевірки, розроблені спеціально для забезпечення високої точності оцінювання.

1. **Відкриті запитання**. У таких випадках перевірка полягає у порівнянні введеного користувачем тексту з правильною відповіддю, що зберігається в базі. Під час перевірки враховується нечутливість до регістру, наявність або відсутність пробілів, можливість альтернативних формулювань. Це забезпечує більшу гнучкість у випадку відкритих питань.

Фрагмент коду:

```
const isCorrect =
  userAnswer.trim().toLowerCase() ===
question.correctAnswer.trim().toLowerCase();
onAnswer(isCorrect, userAnswer);
```

У цьому фрагменті функція trim() видаляє пробіли з обох боків введеної відповіді, а toLowerCase() перетворює її в нижній регістр. Це дозволяє уникнути помилок, пов’язаних із різними варіаціями введення правильних відповідей. Змінна isCorrect набуває значення true, якщо відповідь користувача повністю відповідає очікуваній. Потім викликається функція onAnswer, яка фіксує результат.

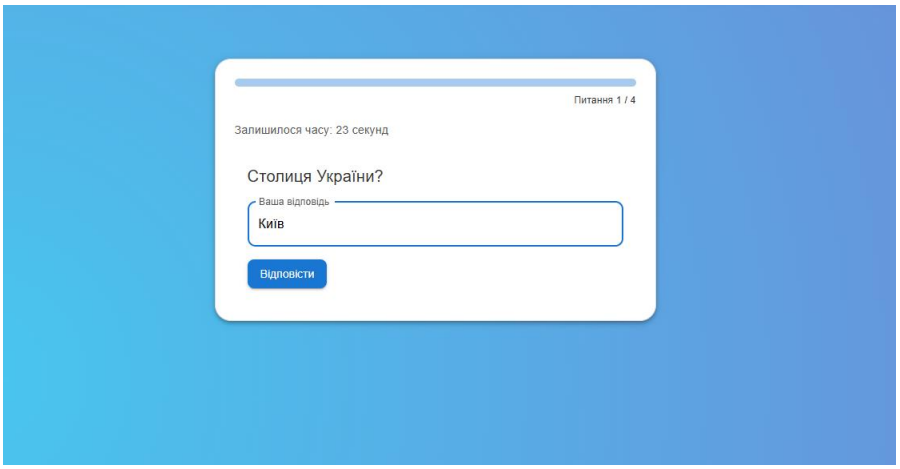

Рис. 3.8 -Приклад відкритого запитання

2. **Питання з вибором однієї правильної відповіді**. Алгоритм перевіряє, чи обраний варіант відповідає збереженому id правильної відповіді. Після відправки відповіді система миттєво надає зворотній зв’язок, підсвічуючи відповідь. Реалізована візуалізація обраного варіанта робить процес більш зручним і зрозумілим для користувача.

Фрагмент коду:

```
const isCorrect = selected === question.correctOptionId;
const selectedOptionText =
  question.options.find(opt => opt.id === selected)?.text || '';
onAnswer(isCorrect, selectedOptionText);
```

Тут selected - це значення, вибране користувачем, а question.correctOptionId - правильна відповідь. Якщо вони збігаються, відповідь вважається правильною. За допомогою методу find система знаходить текст обраного варіанту, який також передається у функцію onAnswer для фіксаці

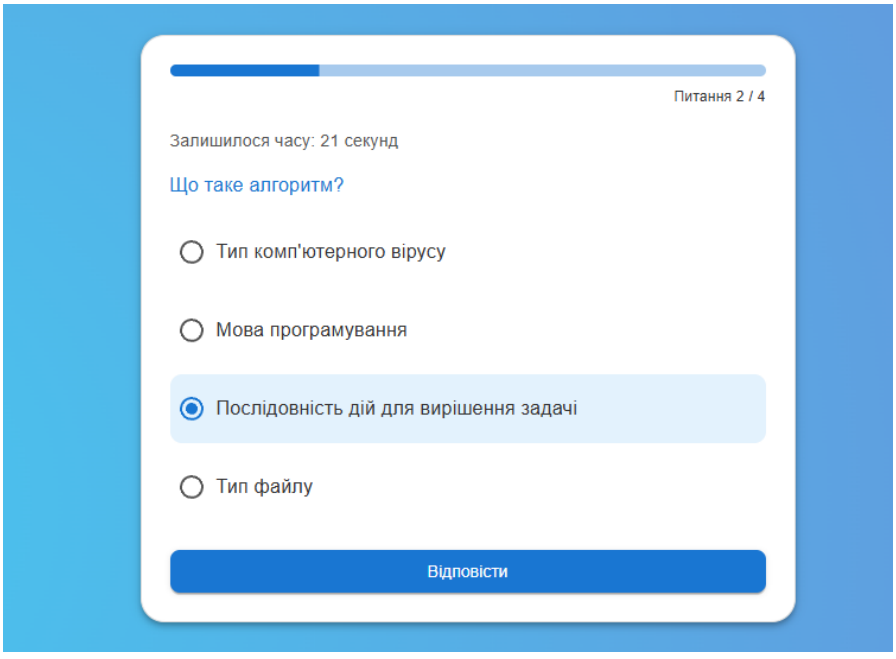

Рис. 3.9 - Приклад запитання з вибором однієї відповіді

3. **Drag-and-drop запитання**. Від користувача вимагається розмістити елементи у правильному порядку. Алгоритм порівнює два масиви - поточну послідовність та еталонну. Якщо порядок збігається, відповідь вважається правильною. Для реалізації функції перетягування використовувалась бібліотека react-beautiful-dnd.

   Фрагмент коду:

```
const handleOnDragEnd = useCallback((result) => {
  if (!result.destination) return;
  const newItems = Array.from(items);
  const [movedItem] = newItems.splice(result.source.index, 1);
  newItems.splice(result.destination.index, 0, movedItem);
  setItems(newItems);
}, [items]);
const userOrderIds = items.map(item => item.id);
const isCorrect = JSON.stringify(userOrderIds) ===
JSON.stringify(question.correctOrder);
```

Після завершення перетягування елемента, функція handleOnDragEnd оновлює порядок елементів у списку items. Потім обчислюється масив ідентифікаторів у поточному порядку і порівнюється з правильним порядком question.correctOrder. Якщо ці масиви ідентичні, то відповідь правильна.

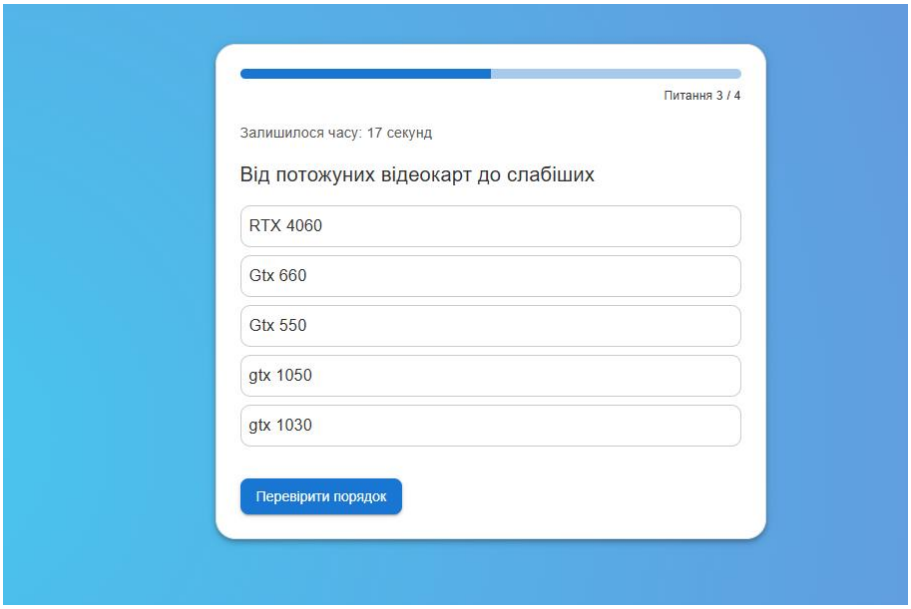

Рис. 3.10 - Приклад drag-and-drop запитання

4. **Запитання на відповідність**. Тут перевірка виконується через обробку пари "ключ - значення", де кожен елемент лівої частини має бути співставлений з відповідним елементом правої. Система обчислює кількість правильних пар, що дозволяє встановити часткову або повну правильність відповіді.

   Фрагмент коду:

```
const [userMatches, setUserMatches] = useState(
  question.matchLeft.map(() => -1)
);
const handleChange = (leftIndex, newValue) => {
  const updated = [...userMatches];
  updated[leftIndex] = newValue;
  setUserMatches(updated);
};
const isCorrect =
  JSON.stringify(userMatches) === JSON.stringify(question.correctMatches);
```

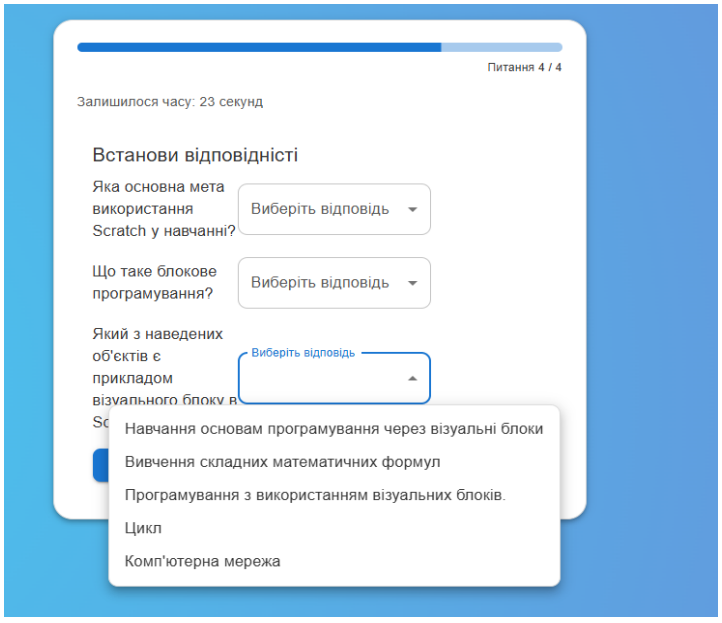

Рис. 3.11 - Приклад запитання на відповідність

Після проходження усіх запитань, користувач отримує підсумкову інформацію про результати: кількість правильних відповідей, відсотковий результат та візуальне підтвердження завершення тесту.

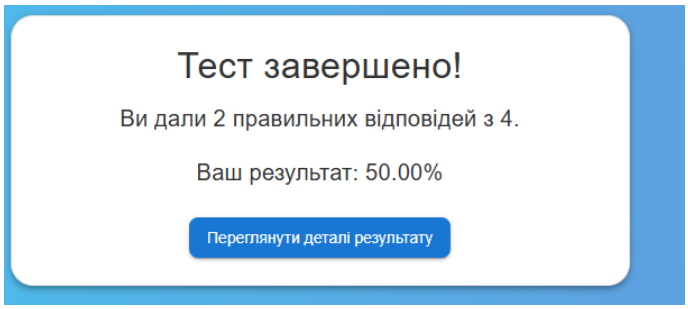

Рис. 3.12 - Підсумок проходження тесту

Окрім загального результату, платформа надає можливість переглянути деталізований звіт - які саме відповіді були надані, які з них правильні, а також показує правильні відповіді для кожного з питань. Це дозволяє не лише оцінити власні знання, але й виправити можливі помилки у майбутньому.

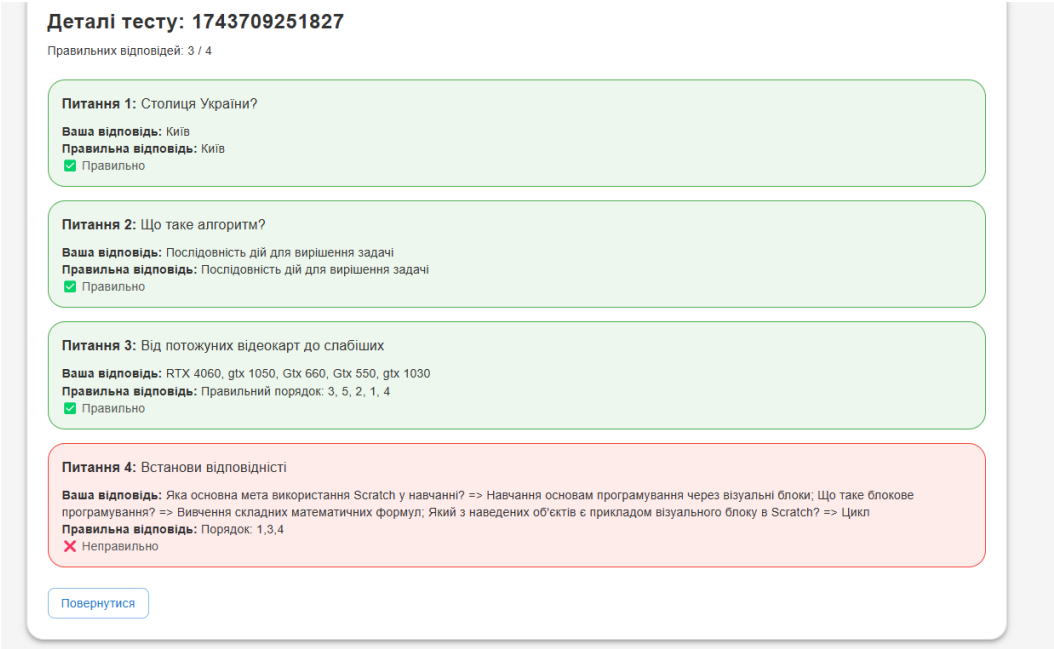

Рис. 3.13 - Детальний звіт про проходження тесту

Впроваджені алгоритми перевірки відповідей розроблені з урахуванням типології запитань, що дозволяє масштабувати систему та легко додавати нові типи без необхідності переробки базової логіки. Усі перевірки спочатку відбуваються на клієнтській стороні для забезпечення швидкості, а потім - за потреби - результати надсилаються на сервер, де можуть зберігатися централізовано для подальшого аналізу або збереження статистики.

Використання локального сховища localStorage забезпечує збереження відповідей навіть у випадку тимчасового зникнення підключення. Завдяки цьому система демонструє високу стійкість до втрати даних та може працювати у режимі офлайн, що особливо важливо для освітніх установ із нестабільним інтернет-з'єднанням.

Загалом, реалізований механізм оцінювання дозволяє створити прозору, адаптивну та зручну систему перевірки знань, яка відповідає сучасним вимогам до цифрових освітніх платформ і забезпечує як точність перевірки, так і комфорт користувача під час проходження тестування.

Крім детального звіту про кожне тестування, система також реалізує механізм збереження історії проходжень. Це дозволяє користувачеві бачити свої попередні результати, аналізувати динаміку успішності та за потреби - повторно переглядати відповіді. У таблиці з історією результатів відображається унікальний ідентифікатор тесту (Test ID), кількість правильних відповідей, загальна кількість запитань, підрахована оцінка, дата проходження та можливість перегляду деталізації.

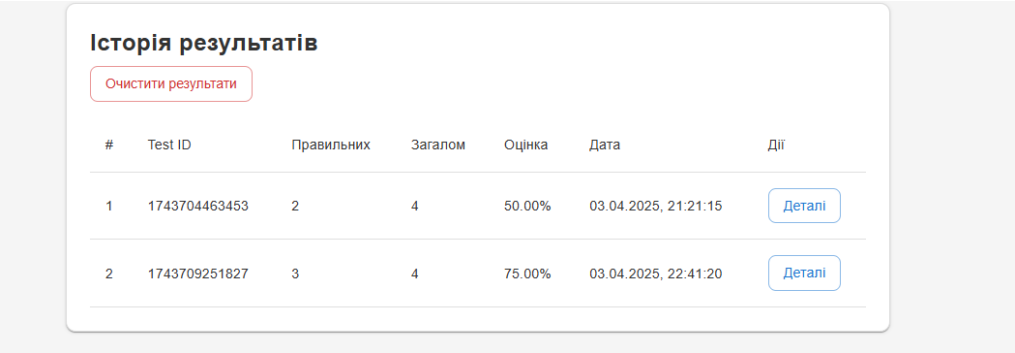

Рис. 3.14 - Історія результатів тестувань

3.3. Адаптивність інтерфейсу: зручність для користувачів ПК, планшетів та мобільних пристроїв

Під час створення веб-застосунку для створення, адміністрування та проходження тестів одним із пріоритетних завдань стало забезпечення

зручного та інтуїтивно зрозумілого інтерфейсу для користувачів з різними типами пристроїв. Сучасні вимоги до UI/UX вимагають врахування широкого спектру розмірів екранів - від настільних моніторів до невеликих мобільних дисплеїв. Таким чином, була впроваджена стратегія адаптивного дизайну, яка дозволила створити єдину гнучку платформу.

Для реалізації цієї мети використовувалися найкращі практики адаптивної верстки, зокрема можливості CSS Flexbox та компонентної бібліотеки Material UI (MUI) [19]. Flexbox дозволив забезпечити динамічне вирівнювання, масштабування та порядок елементів на сторінці, а компоненти MUI дали змогу створити інтерфейс, що самостійно підлаштовується до змін ширини екрана [20]. Бібліотека MUI також підтримує сучасні теми та адаптивні медіа-запити, що дозволило покращити доступність і зробити застосунок зручним у користуванні на будь-якому пристрої.

Особливу роль у формуванні адаптивного дизайну відіграли компоненти Box, Container, Grid та інші, які використовувалися для побудови модульної структури сторінки. Вся система побудована на принципі mobile-first, що означає первинну оптимізацію під мобільні пристрої та подальше масштабування під ширші екрани. Відступи, розміри, кольори та типографіка автоматично змінюються відповідно до поточного breakpoint’у, що значно покращує взаємодію.

Інтерфейсні елементи - кнопки, поля вводу, повідомлення, списки, заголовки, картки та блоки з контентом - адаптовані для змінення свого розміру та розташування в залежності від пристрою. У мобільному варіанті все переходить до вертикального макету з одним стовпцем, що спрощує навігацію, зменшує візуальне навантаження та виключає необхідність горизонтальної прокрутки. Також інтерфейсні компоненти аналогічно виглядають і працюють на планшетах, що забезпечує послідовний досвід взаємодії користувача незалежно від діагоналі пристрою.

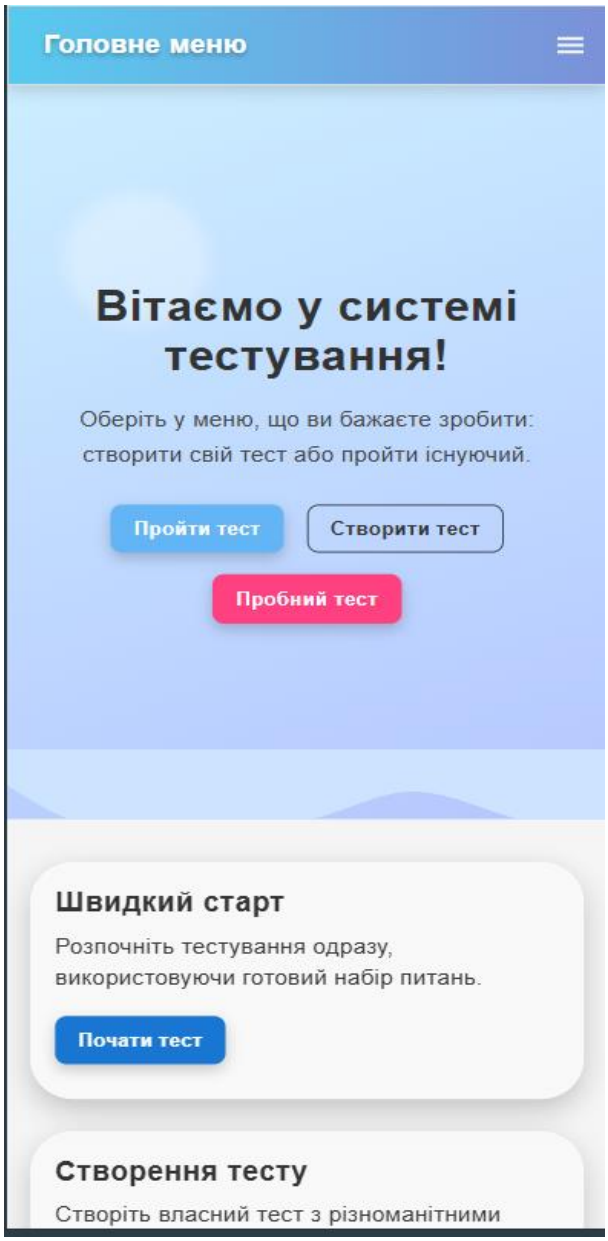

Рис. 3.14 - Головна сторінка в мобільному вигляді

Окрему увагу приділено навігації. У мобільній версії вона реалізується через компактне «бургер-меню», яке відкриває бокову панель з усіма доступними пунктами. Це дозволяє максимально заощадити простір на екрані та забезпечити швидкий доступ до основних функцій.

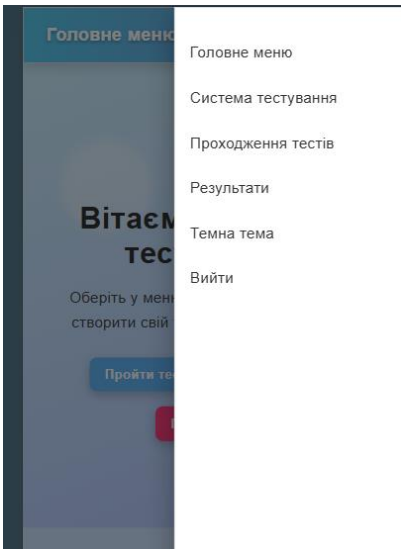

Рис. 3.15 - Випадаюче навігаційне меню на смартфоні

Компоненти для проходження тестів, зокрема ті, що підтримують drag-and-drop, вибір варіантів або введення відповіді, також масштабуються відповідно до розміру екрана. Навіть на невеликих пристроях вони зберігають інтуїтивність, читаємість і зручність у взаємодії.

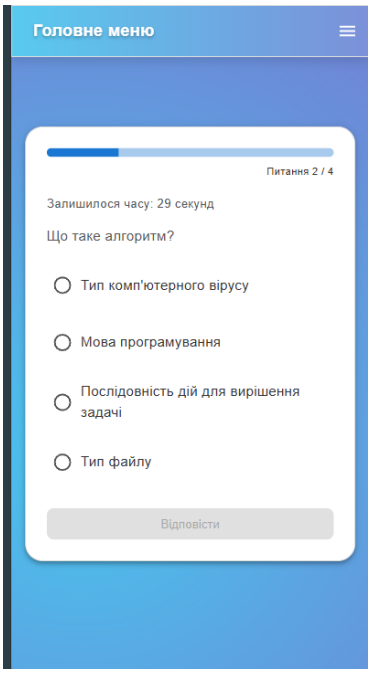

Рис. 3.16 -Мобільна версія інтерфейсу запитання

Важливим елементом стало також коректне відображення результатів. Таблиця з результатами тестів, що зазвичай має багато стовпців, у

мобільному вигляді трансформується у вертикальні блоки. Це дозволяє легко переглядати результати без втрати інформаційного навантаження.

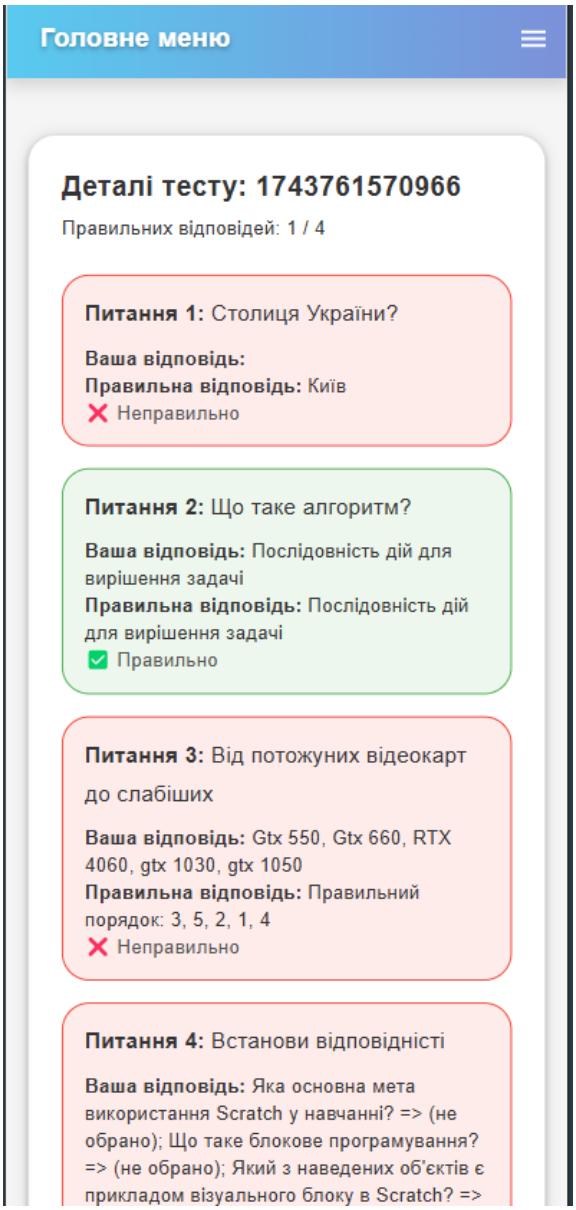

Рис. 3.17 - Мобільне відображення детального звіту

Окремо слід згадати підтримку темної теми. Вона реалізована за допомогою можливостей MUI, де стилі змінюються автоматично відповідно до системних налаштувань користувача. Це не лише підвищує комфорт, але й зменшує навантаження на очі, особливо при роботі в умовах недостатнього освітлення.

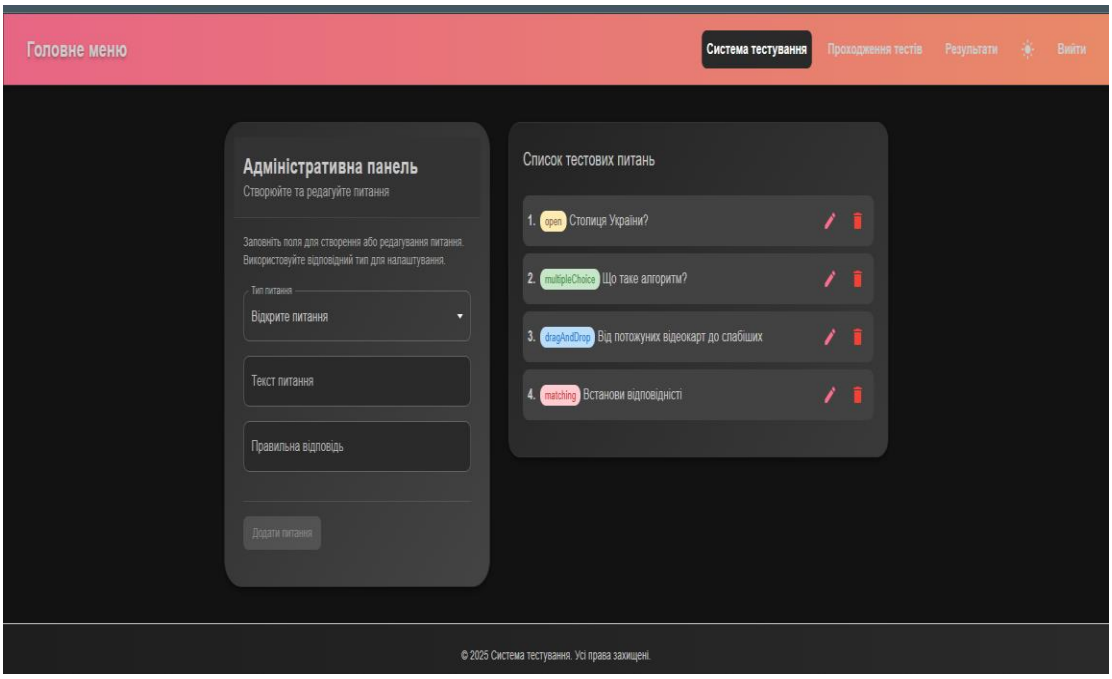

Рис. 3.18 - Адаптивний вигляд у темному режимі

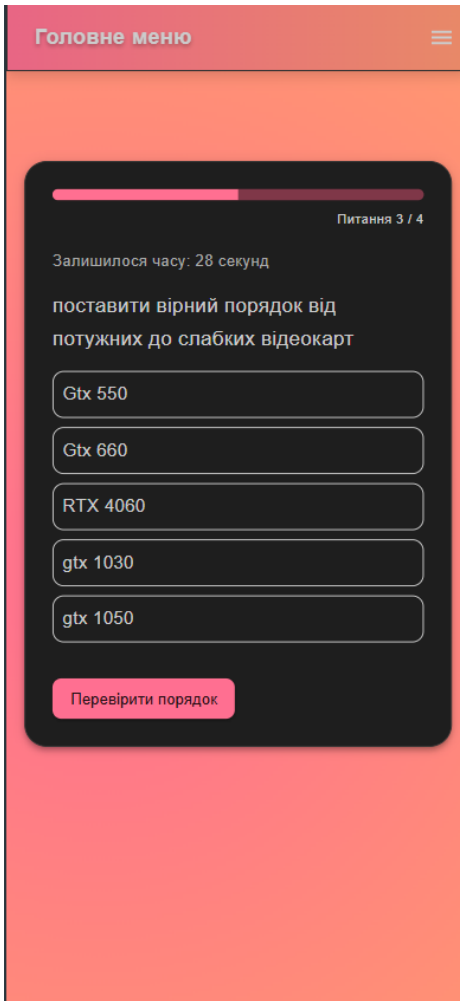

Рис. 3.19 - Приклад запитання у темній темі на мобільному

Завдяки комплексному підходу до адаптації -використанню Flexbox, компонентів MUI, вбудованих медіа-запитів та тематичних стилів -веб-застосунок забезпечує зручний, сучасний і універсальний інтерфейс. Це не лише покращує користувацький досвід, а й робить платформу доступною для ширшого кола користувачів, незалежно від їхнього пристрою.

3.4. Тестування застосунку на реальних даних, оптимізація швидкодії та коректності відображення на різних екранах

Після завершення основної фази розробки веб-застосунку було розпочато комплексне тестування, метою якого стала оцінка функціональної правильності, адаптивності інтерфейсу, стабільності роботи та продуктивності на різних пристроях. Було розроблено набір сценаріїв для перевірки основних функціональних модулів, таких як створення, проходження тестів, формування результатів та їх візуалізація.

Тестування проводилось як вручну, так і з використанням інструментів розробника браузера Google Chrome, зокрема Chrome DevTools. Вручне тестування дозволяло безпосередньо оцінити зручність використання, тоді як Chrome DevTools використовувався для виявлення помилок в адаптивній верстці та поведінці елементів на різних екранах.

Тестування охоплювало широкий спектр пристроїв, зокрема смартфони Samsung Galaxy A51, iPhone X, планшети iPad Mini, а також десктопи з різними розширеннями екрану. Завдяки Chrome DevTools було змодельовано умови використання на вказаних пристроях, що дало змогу детально перевірити відображення текстів, кнопок, форм, навігаційних елементів та поведінку адаптивного лейауту (вживається щодо графічного дизайну і означає вигляд елементів сторінки та їх розташування).

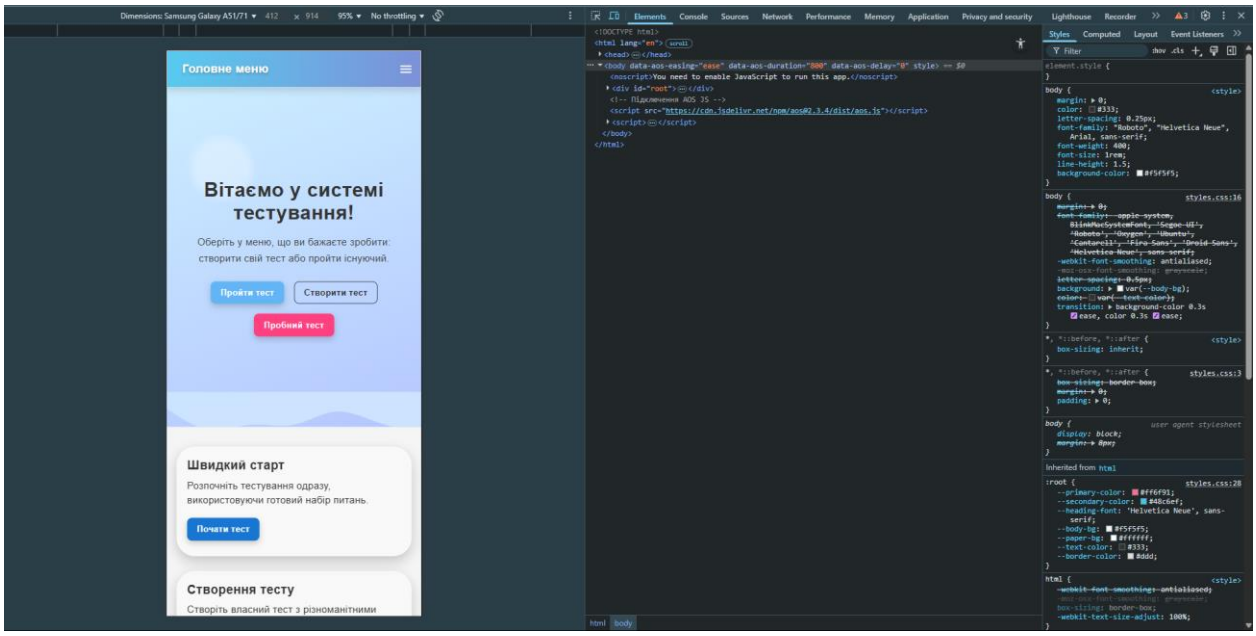

Рис. 3.20 - Тестування мобільної версії інтерфейсу в Chrome DevTools (модель: Samsung Galaxy A51)

Особлива увага приділялась тестуванню адаптивності: елементи інтерфейсу залишаються зрозумілими й легкодоступними на пристроях з малими екранами, кнопки не перекриваються, а поля вводу не виходять за межі екрану. На всіх перевірених пристроях застосунок демонстрував стабільну роботу, без візуальних збоїв або логічних помилок [Див. 5.1., 5.2. у додатку Б].

Для покращення користувацького досвіду були впроваджені легкі анімації з використанням бібліотеки AOS (Animate On Scroll). Ці анімації застосовувались до ключових елементів інтерфейсу, зокрема карток результатів, запитань та кнопок. Анімаційні ефекти протестовано на пристроях із середнім та низьким рівнем продуктивності. За результатами перевірки встановлено, що анімації не впливають на швидкість завантаження сторінок та не створюють затримок у взаємодії.

У процесі тестування було виявлено низку дрібних недоліків у верстці: неправильне вирівнювання елементів при нестандартних пропорціях екрану, надмірні відступи, горизонтальна прокрутка. Для усунення вад були

скориговані медіа-запити, відступи та ширини контейнерів. Також уточнено правила використання CSS Flexbox та Grid Layout для досягнення максимальної гнучкості макету.

Крім цього, була вдосконалена логіка підключення стилів залежно від breakpoints, що забезпечило стабільну і логічну структуру сторінок при зміні розміру вікна. Це дало змогу уникнути дублювання стилів та забезпечити рівномірне відображення вмісту.

Елементи інтерфейсу були перевірені на зручність використання сенсорного введення. Було підтверджено, що вони достатньо великі для зручного натискання пальцем, тексти залишаються читабельними, а структура екранів - послідовною та інтуїтивною.

Проведені заходи з оптимізації дозволили суттєво підвищити стабільність та швидкодію застосунку. Всі функції коректно працюють без помітних затримок. Завдяки ретельному тестуванню та врахуванню специфіки роботи на різних пристроях, система повністю готова до масштабного використання як на десктопах, так і на планшетах і смартфонах, забезпечуючи однаково якісний досвід для всіх користувачів.

ВИСНОВОК

Підсумовуючи результати виконаної кваліфікаційної роботи, можна стверджувати, що створений веб-застосунок для створення, адміністрування та проходження освітніх тестів повністю відповідає сучасним вимогам цифрового середовища. Його реалізація враховує тенденції адаптивного дизайну, інтуїтивної взаємодії з користувачем та високої функціональності. Проведене дослідження охоплює всі етапи -від аналітики до тестування -що дозволило досягти поставленої мети та реалізувати систему, готову до практичного застосування.

У межах дослідження було послідовно реалізовано низку етапів:

1) **Аналіз предметної області.** Проведено детальний огляд сучасних інструментів онлайн-тестування. Було вивчено їх функціональні можливості, зручність користування та технічну реалізацію. Це дало змогу виявити наявні обмеження та сформувати бачення ефективного освітнього продукту.

2) **Формулювання вимог до системи.** Визначено як функціональні, так і нефункціональні вимоги до майбутнього застосунку. Серед ключових вимог - доступність, інтуїтивність інтерфейсу, безпека обробки даних, адаптивність інтерфейсу до різних пристроїв, а також простота подальшого масштабування.

3) **Проєктування архітектури.** Реалізовано розподілену архітектуру з чітким розмежуванням клієнтської та серверної логіки. Клієнтська частина реалізована на основі React.js, а серверна - на Node.js. Такий підхід забезпечив гнучкість системи, зручність модифікацій та розширення функціоналу.

4) **Розробка функціоналу.** Було створено функціональні інтерфейси для двох ролей користувачів: адміністратора та звичайного користувача. Адміністратор має змогу керувати базою тестів (створювати,

редагувати, видаляти), а також переглядати статистику проходження. Користувачі мають змогу проходити тести різних форматів: із вибором однієї правильної відповіді, встановленням відповідності, відкритими запитаннями, drag-and-drop завданнями. Передбачено зворотний зв’язок одразу після проходження.

5) **Адаптивність інтерфейсу.** Було реалізовано повністю адаптивний дизайн, що забезпечує комфортну взаємодію на різних пристроях: комп’ютерах, планшетах і смартфонах. Застосовано сучасні технології веб-розробки - CSS Flexbox, Grid Layout, а також бібліотеку Material UI. Завдяки цьому інтерфейс автоматично підлаштовується під розміри екрана, зберігаючи зручність навігації та читабельність інформації.

6) **Тестування та оптимізація.** Застосунок проходив комплексне тестування за допомогою браузера Google Chrome і середовища Chrome DevTools. Було перевірено коректність логіки роботи, відповідність очікуванням користувача, а також здійснено виявлення та усунення помилок, що виникали при взаємодії з різними елементами інтерфейсу. Впроваджено легкі анімації - зокрема, ефекти появи елементів - для підвищення привабливості застосунку та покращення юзабіліті. Оптимізація коду дозволила знизити час завантаження і покращити швидкодію інтерфейсу.

7) **Практична цінність розробки.** Розроблений застосунок може бути легко інтегрований в освітній процес закладів середньої та вищої освіти. Він забезпечує ефективний інструмент для перевірки знань, підвищує мотивацію людей до навчання та спрощує роботу викладачів із моніторингу результатів.

Розроблений веб-застосунок продемонстрував високу ефективність і стабільність роботи. Він відповідає сучасним стандартам цифрової освіти, є гнучким до подальшого розвитку та потенційно може масштабуватися під потреби різних освітніх середовищ. Таким чином, результати роботи не лише

мають теоретичну цінність, а й забезпечують вагомий практичний внесок у впровадження цифрових інструментів в освітній процес.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бойко І.В. Розробка і організаційно-технічні методи забезпечення тестового контролю знань в системі сучасної шкільної освіти: кваліфікаційна робота магістра / І.В. Бойко; наук. кер. П.Б. Митрофанов. -Полтава: Нац. ун-т «Полтавська політехніка імені Юрія Кондратюка», 2025. -100 с.

2. Войченко В.С. Аналіз платформ онлайн–навчання та їх можливостей для тестування результатів навчання: дипломна робота бакалавра / В.С. Войченко; наук. кер. О.О. Капшук. -Київ: Нац. техн. ун-т України «Київський політехнічний інститут імені Ігоря Сікорського», 2021. -99 с.

3. Адаптивне тестування в системі Moodle. MoodleMoot Ukraine 2015.

4. Авраменко А.С., Авраменко В.С., Косенюк Г.В. Тестування програмного забезпечення: навч. посіб. / А.С. Авраменко, В.С. Авраменко, Г.В. Косенюк. -Черкаси: Черкас. нац. ун-т ім. Богдана Хмельницького, 2017. -284 с.

5. Бондар Г.А. Використання системи онлайн-тестів для розвитку соціальної та комунікативної компетентностей учнів початкових класів: матеріали до уроків / Г.А. Бондар.

6. Kahoot! | Learning games | Make learning awesome!. *Kahoot!*. URL: https://kahoot.com/.

7. Testportal -online assessment platform | Create your own tests, quizzes and exams. Testportal (офіц. сайт). URL: https://www.testportal.net/

8. W3.CSS Тренди. Уроки для початківців. W3Schools українською. *W3Schools українською. Безплатні уроки онлайн для початківців, школярів та студентів*. URL: https://w3schoolsua.github.io/w3css/w3css_trends.html.

9. Важливість адаптивного дизайну: як створити веб-сайт, який добре виглядатиме на всіх пристроях While Web Production. *While Web Production*. URL: https://whileweb.com/uk/blog/vazhlivist-adaptivnogo-dizajnu-yak-stvoriti-veb-sajt-yakij-dobre-viglyadatime-na-vsih-pristroyah/.

10. React -JavaScript-бібліотека для створення користувацьких інтерфейсів. *React -JavaScript-бібліотека для створення користувацьких інтерфейсів*. URL: https://uk.legacy.reactjs.org/.

11. Angular. *Home • Angular*. URL: https://angular.dev/.

12. Vue.js. - *The Progressive JavaScript Framework | Vue.js*. URL: https://vuejs.org/.

13. Node.js -Запускайте JavaScript будь-де. *Node.js -Run JavaScript Everywhere*. URL: https://nodejs.org/uk.

14. Django. *Django Project*. URL: https://www.djangoproject.com/.

15. Welcome to Flask -Flask Documentation (3.1.x). *Welcome to Flask - Flask Documentation (3.1.x)*. URL: https://flask.palletsprojects.com/en/stable/.

16. Microsoft. Visual Studio Code - Code Editing. Redefined. *Visual Studio Code - Code Editing. Redefined*. URL: https://code.visualstudio.com/.

17. Framer-motion. *npm*. URL: https://www.npmjs.com/package/framer-motion.

18. React Snackbar component - Material UI. *MUI: The React component library you always wanted*. URL: https://mui.com/material-ui/react-snackbar/?srsltid=AfmBOoqvdGTmxYbihAEnsbNNwPUX6RWNKt3nuPt24jh_iUJ8C4n_C8kt.

19. Material UI: React components that implement Material Design. *MUI: The React component library you always wanted*. URL: https://mui.com/material-ui/?srsltid=AfmBOoq0AJN-tXC2YX7-grMK-3Cw8Cih2hpdodEMtv4TWNsxYvdSfm33.

20. Flexbox -що це в CSS і в чому переваги. *Корисні матеріали: Статті та новини IT-індустрії | Комп'ютерна школа Hillel*. URL: https://blog.ithillel.ua/articles/what-is-flexbox.

21. Мови програмування для створення сайтів: порівняння. *FoxmindEd*. URL: https://foxminded.ua/movy-prohramuvannia-dlia-stvorennia-saitiv/.

22. Пасинков С. Стас Пасинков 3 рівень Київ 4 серпня 2023 870 views 0 comments Створення простого веб-додатку на сервлетах та jsp. *JavaRush*. URL: https://javarush.com/ua/groups/posts/uk.328.stvorennja-prostogo-veb-dodatku-na-servletakh-ta-jsp-chastina-1.

23. Розробка веб сайтів для початківців html - css - javascript. URL: https://nvkarta.com/project/library/uploads/engineering/programming/(uk)_rozrobka_veb-saitiv_dlia_pochatkivtsiv_html_css_javascript.pdf.

24. Що таке npm (Node Package Manager)? Як встановити та розмістити пакети. *Highload.tech - медіа для розробників*. URL: https://highload.tech/uk/npm/.

25. Drag'n'Drop з подіями миші. *Сучасний підручник з JavaScript*. URL: https://uk.javascript.info/mouse-drag-and-drop.

26. JavaScript: як створити інтерактивний сайт - Рейтинг Конструкторів Сайтів. *Рейтинг Конструкторів Сайтів*. URL: https://landinglist.com.ua/javascript-yak-stvoryty-interaktyvnyj-sajt/.

27. Microsoft. Visual studio code - code editing. redefined. *Visual Studio Code - Code Editing. Redefined*. URL: https://code.visualstudio.com/.

28. Npm | home. *npm | Home*. URL: https://www.npmjs.com/.

29. Ukrainian W. Уроки від W3Schools українською онлайн. *W3Schools українською. Для початківців, школярів та студентів*. URL: https://w3schoolsua.github.io/css/index.html#gsc.tab=0.

30. Viacheslav. Java. *JavaRush*. URL: https://javarush.com/ua/groups/posts/uk.2125.veb-dodatok-na-java.

ДОДАТОК А

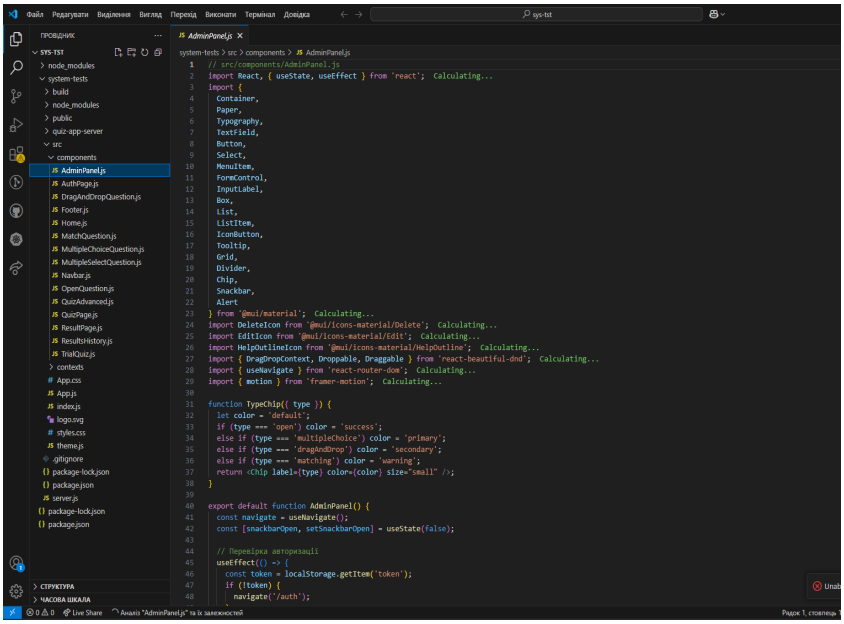

Рис. 1.1 – Visual code

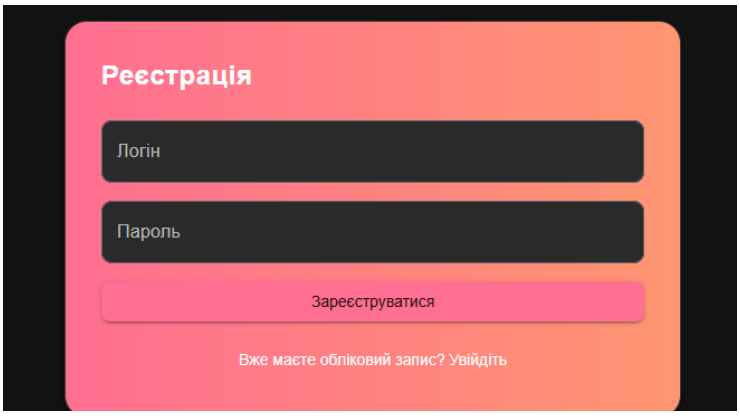

Рис. 2.1 - Вікно Реєстрації

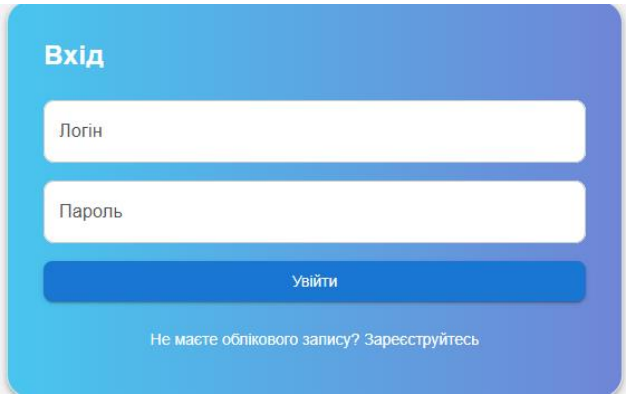

Рис. 3.1 - Вікно авторизації

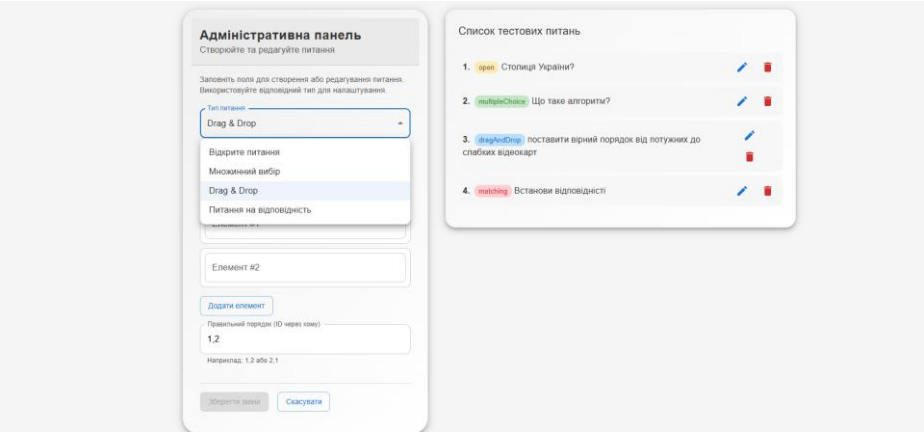

Рис. 4.1 - Інтерфейс додавання нового тесту

ДОДАТОК Б

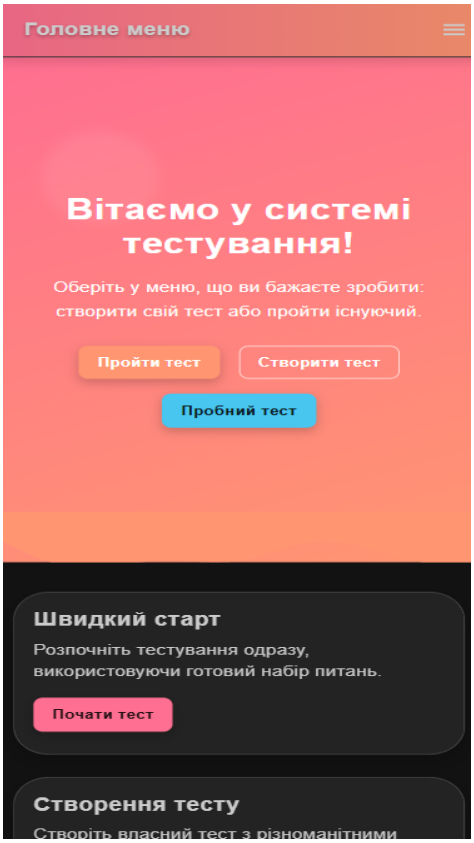

Рис. 5.2 - Вигляд інтерфейсу на смартфоні (Chrome DevTools, режим мобільного перегляду з темним режимо)

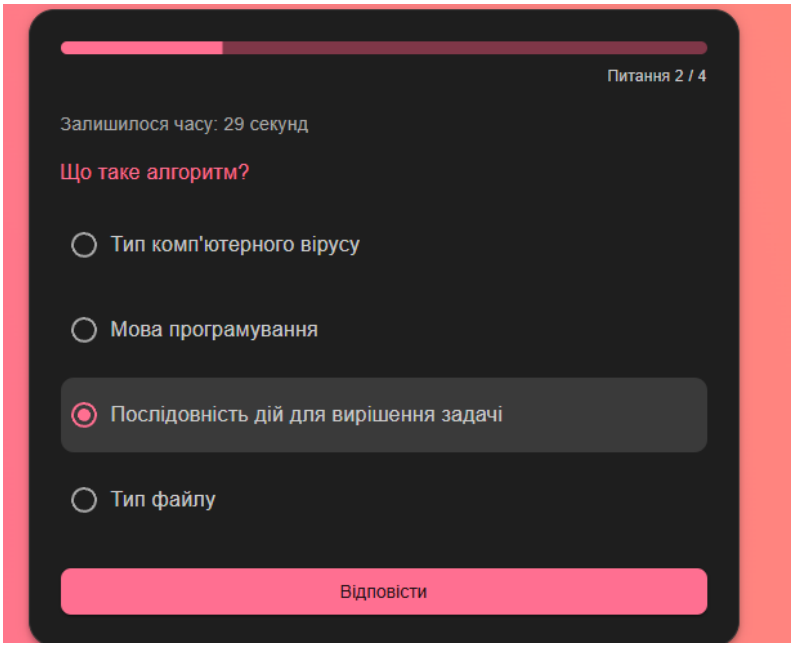

Рис. 6.2 - Інтерфейс для користувача, який проходить тест у темному режимі

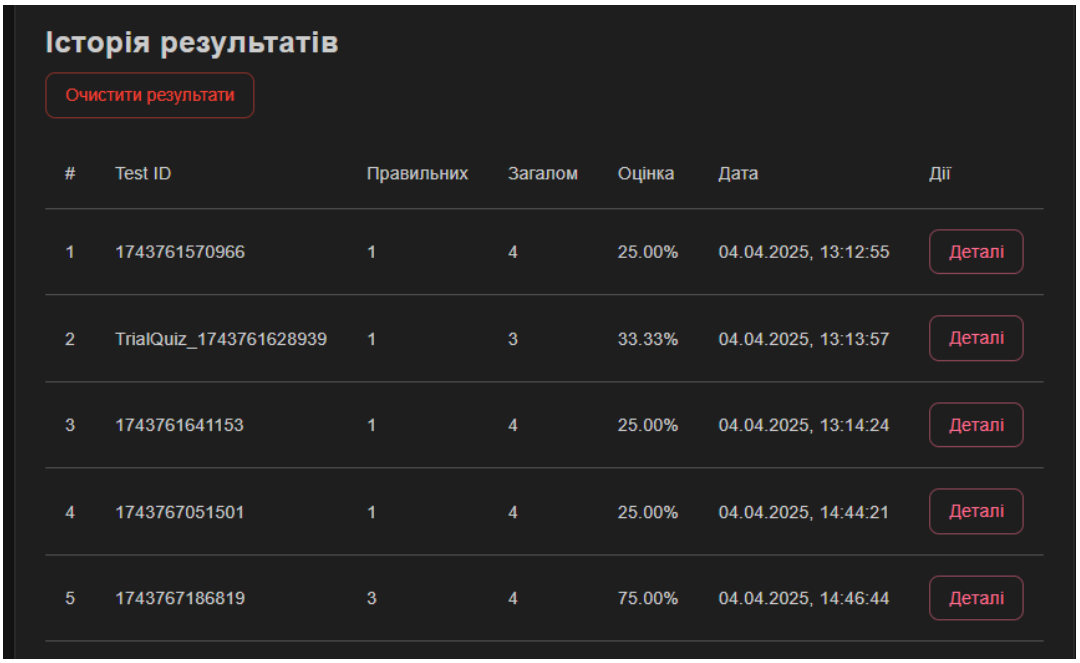

Рис. 7.2 - Перегляд результатів тестування у темному режимі