МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

дискретного аналізу та інтелектуальних систем
(повна назва кафедри)

Дипломна робота

РОЗРОБКА ІНТЕРНЕТ ПЛАТФОРМИ ДЛЯ ТЕСТУВАННЯ ЗНАНЬ

Виконав: студент IV курсу, групи ПМі-43с
напряму підготовки (спеціальності)
122 «Комп'ютерні науки»
(шифр і назва спеціальності)

Стахів Р.М.
(підпис) (прізвище та ініціали)

Керівник Олійник Р.М.
(підпис) (прізвище та ініціали)

Рецензент
(підпис) (прізвище та ініціали)

2023

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет Прикладної математики та інформатики _____
Кафедра Дискретного аналізу та інтелектуальних систем _____
Спеціальність 122 «Комп'ютерні науки» _____ __
(шифр і назва)

«ЗАТВЕРДЖУЮ»
Завідувач кафедри _____

"31 "серпня 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНУ У РОБОТУ СТУДЕНТУ

Стахіву Ростиславу Михайловичу
(прізвище, ім'я, по батькові)

1 .Тема роботи Розробка інтернет платформи для тестування знань _____
_____
_____

керівник роботи_____ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені Вченою радою факультету від **" 13" вересня 2022 року № 15** _____

2. Строк подання студентом роботи **13.06.2023р.** _____

3. Вихідні дані до роботи __документація по мові програмування C#, документація по JavaScript фреймворку React.js, IDE Visual Studio 2022, IDE Visual Studio Code, інтернет ресурси_____
_____
_____

4. Зміст дипломної роботи (перелік питань, які потрібно розробити) ознайомитись з існуючими платформами для тестування знань, знайти переваги і недоліки платформ, сформувати вимоги до своєї розробки, обрати мову програмування для серверу і фреймворк для клієнтської частини, розробити серверну частину, розробити клієнтську частину

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
__ілюстрації життєвого циклу компонента React.js, ілюстрації шаблону дизайну сайту, ілюстрації багатошарової архітектури сервера, ER-діаграми баз даних_____
_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання **31 серпня 2022 р.**

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1.	Дослідження існуючих платформ для тестування	15.10.2022	
2.	Пошук переваг і недоліків платформ для тестування	20.10.2022	
3.	Формування вимог до своєї розробки	02.11.2022	
4.	Розробка дизайну платформи	12.11.2022	
5.	Розробка базової архітектури платформи	20.12.2022	
6.	Розробка функціоналу для роботи з базою даних	17.01.2023	
7.	Розробка логічного функціоналу для обробки даних	06.03.2023	
8.	Розробка функціоналу для отримання запитів і відповіді на них	20.03.2023	
9.	Розробка клієнтської частини платформи	24.05.2023	
10.	Тестування платформи	28.05.2023	

Студент _____ **Стахів Р.М.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Олійник Р.М.**
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Робота складається з вступу,трьох розділів, висновків і списку використаної літератури. Метою цієї роботи була розробка зручної та інтуїтивно-зрозумілої платформи, яка має забезпечити можливість для викладачів створювати питання і тести, які складатимуться з цих питань, встановлювати складність, обмеження по часу та максимальну оцінку за тест.

Зі сторони студента має бути можливість зареєструватись на тест з допомогою запрошувального посилання, яке йому поширить викладач, вибрати складність для проходження тесту і власне пройти тест, після чого студент повинен мати можливість побачити свій результат. Після проходження тесту студентом, викладач також повинен мати можливість переглядати результати студентів по певному тесту.

Іншим напрямком під час розробки платформи було забезпечення максимальної прозорості і справедливості тестування. Було реалізовано декілька запобіжних засобів, які спрямовані на те, щоб сильно ускладнити процес списування або фальсифікації своїх результатів.

Ця розробка є дуже корисною в теперішніх умовах, адже зараз викладачі стикаються з багатьма проблемами при оцінюванні студентів, адже не всі студенти є доброчесними і часто вдаються до спроб фальсифікувати свої знання задля вищої оцінки.

ЗМІСТ

ВСТУП

З березня 2020 року через пандемію коронавірусу в Україні було встановлено карантин, який заборонив здобувачам освіти відвідувати навчальні заклади. З часом навчальні заклади продовжили навчання у дистанційному форматі. Такий формат навчання вплинув на кожного студента, адже тепер замість звичних парт і аудиторій був комп'ютер, втратилось живе спілкування з викладачем і одногрупниками, а у викладачів почали виникати питання як проводити контроль знань студента, адже стало неможливим проведення контрольних робіт і екзаменів у навчальному закладі. Рішенням цієї проблеми стали інтернет платформи для тестування. Вони забезпечують студентам зручні умови проходження тестів, а викладачам надають хороші можливості по оцінюванню.

Перевірка і оцінка знань студента завжди були одними з найважливіших складових навчального процесу. Неможливість проводити тести очно у навчальних закладах спричинає багато проблем як викладачам, так і студентам.

Першою, і основною проблемою, є повна відсутність контролю, студент може знайти відповіді до тесту в інтернеті або серед навчальних матеріалів, які йому надіслав викладач. Ця проблема стосується і викладачів, адже важко зрозуміти, який студент на яку оцінку заслуговує, так і студентів, адже знання обезцінюються перед вмінням швидко шукати відповіді в інтернеті.

Другою проблемою є те, що бувають часті програмні збої, наприклад, не відкривається картинка або відповідь не збереглась. У найкращому випадку подібний збій призведе до переписування студентом тесту, якщо в нього буде знімок екрану, який підтвердить програмний збій, в найгіршому випадку збій призведе до втрати балів, якщо в студента не буде доказів збою.

Також втратити бали можна через нестабільність інтернету, адже більшість тест платформ для зміни кожного питання надсилають окремий запит на сервер, який при нестабільному інтернеті займає багато часу, якого у студентів під час тесту і так обмаль. У таких платформ також є проблема з максимальною кількістю

запитів на сервер, адже, під час сесії, дуже багато студентів одночасно проходять тести і, якщо для кожного студента для кожної зміни питання потрібно один запит, то сервер не справляється з такою кількістю і тести у студентів “зависають”.

В той час як забезпечити контроль доволі складно, але все ж можливо, з іншими двома проблемами впоратись набагато простіше. Потрібно зменшити кількість запитів для написання одного тесту, тобто відправляти цілий тест одним запитом, а змінювати питання і зберігати відповіді вже на клієнтській частині програми, тим самим переклавши частину роботи на комп’ютер користувача. Такий крок не лише покращує швидкодію програми, а й запобігає такому програмному збою як не збережена відповідь через незавершений запит, або відмову в обслуговуванні сервером.

1. ІСНУЮЧІ СИСТЕМИ ТЕСТУВАННЯ

На сьогоднішній день існує багато рішень для проведення дистанційного навчання. Одним із таких рішень є онлайн платформи для тестування. Таких платформ є дуже багато і з кожним днем їх кількість збільшується. Розглянемо найпопулярніші з них.

1.1. Платформа для тестування Moodle

Moodle – це навчальна платформа, яка дозволяє викладачам налагодити навчальний процес у дистанційному форматі. Вона дозволяє викладачам поширювати навчальні матеріали та новини, обмінюватись повідомленнями, а також проводити тести. Ця платформа забезпечує студентам проходження тестів в будь-який час, який буде вкладатись в межі, визначені викладачем, зручну навігацію між питаннями, приємний і інтуїтивно-зрозумілий інтерфейс.

Значною перевагою Moodle є те, що платформа не дозволяє випадково пропустити питання, адже після натискання на кнопку завершення тесту з’являється вікно, у якому можна перевірити відповіді, на які питання було збережено відповідь і, якщо студент помітить, що відповідь на якесь питання не було збережено, він може повернутись до тесту та відповісти на пропущене питання.

Але у цієї платформи також є недоліки, основним з яких є швидкодія, адже, кожного разу, коли студент переходить на інше питання, відправляється запит на сервер, що суттєво знижує продуктивність, особливо, коли багато студентів проходять тест одночасно, наприклад, під час сесії.

1.2. Платформа для тестування Microsoft Forms

Microsoft Forms – це сервіс для створення і проведення опитувань і тестів, створений компанією Microsoft у червні 2016. Ця платформа дозволяє оперативно створити вікторину, опитування чи тест і запросити інших користувачів відповісти на це завдання. При цьому власник завдання може відразу переглядати результати запрошених, і, за допомогою вбудованих інструментів, він може проводити аналітику, щоб оцінити відповіді. Також сервіс дозволяє експортувати результати у Excel-файл для додаткового аналізу.

Ця платформа є дуже зручною, але вона в основному призначена для створення опитувань, щоб дізнатись думку більшості, або для проведення невеликих навчальних тестів для закріплення матеріалу, наприклад, після лекції, але у цьому сервісі немає змоги обмежити кількість спроб, тому його не варто використовувати для фінальних тестів для контролю знань.

2. ВИКОРИСТАНІ ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТИ

2.1. PostgreSQL

PostgreSQL - це популярна і широко використовувана система керування базами даних з відкритим вихідним кодом, яка заслуговує на увагу завдяки своїй потужності та гнучкості. Вона була розроблена університетом Берклі в Каліфорнії ще в далекому 1987 році і з того часу пройшла великий шлях розвитку та популяризації.

Однією з основних переваг PostgreSQL над іншими СКБД є можливість використання функцій, які дозволяють виконувати код безпосередньо на сервері бази даних. Це означає, що ви можете створювати складні логічні операції, використовуючи мову SQL разом з іншими мовами програмування, які підтримуються PostgreSQL, такими як:

- PL/pgSQL
- PL/Perl
- PL/Python
- PL/Tcl
- PL/Ruby
- PL/sh

Також функції можна написати такими мовами програмування, як C, C++ і Java за допомогою PL/Java. Це дає розробникам неймовірну гнучкість та можливість реалізувати найскладніші логічні операції безпосередньо на рівні бази даних.

Окрім того, PostgreSQL підтримує багато типів індексів, таких як:

- B-дерева,
- хеш-індекси,
- R-дерева,

— GiST,

— GIN.

Це дозволяє оптимізувати швидкість і продуктивність запитів до бази даних, вибираючи найбільш підходящий тип індексу для конкретної ситуації.

І що ще цікаво, PostgreSQL дає можливість розширювати його функціонал у багатьох аспектах. Розробник може створювати власні перетворення типів, типи даних, домени, функції (включаючи агрегатні функції), індекси, оператори (включаючи перевизначення вже існуючих) та процедурні мови. Це означає, що можна реалізувати практично будь які власні специфічні функціональні можливості та логіку, які відповідають вашим потребам і вимогам.

Крім того, PostgreSQL пропонує цікавий механізм успадкування таблиць, що дозволяє таблиці успадковувати характеристики та поля від інших таблиць. Це дає можливість швидко створювати нові таблиці, які успадковуються від вже існуючих, і отримувати доступ до даних та запитів батьківських таблиць автоматично. Хоча цей функціонал ще не є повністю завершеним, він вже може бути успішно використаний у практичних сценаріях.

А щодо тригерів, то PostgreSQL дозволяє визначати тригери як функції, які автоматично спрацьовують при виконанні певних операцій зміни даних (DML), таких як вставка, оновлення або видалення записів. Ви можете написати тригери на різних мовах програмування, вони будуть пов'язані з конкретною таблицею. Важливим моментом є те, що множинні тригери виконуються в алфавітному порядку, що дозволяє контролювати послідовність їх виконання.

В цілому, PostgreSQL є досить розширюваною та потужною системою керування базами даних, яка надає широкий спектр можливостей для роботи з даними, логіки та оптимізації запитів. Її гнучкість та можливості дозволяють задовольняти різноманітні потреби користувачів та реалізовувати складні логічні сценарії в контексті баз даних.

2.2. Entity Framework Core

Entity Framework Core (EF Core) – це технологія доступу до даних, розроблена компанією Microsoft, яка пропонує розширені можливості для роботи з об'єктами. Як ORM-інструмент, EF Core надає високорівневу абстракцію для роботи з базами даних, що дозволяє нам працювати з даними незалежно від типу сховища, абстрагуючись від деталей роботи з таблицями, індексами та ключами. Це означає, що ми працюємо з об'єктами на концептуальному рівні, незалежно від фізичної реалізації бази даних.

EF Core підтримує широкий спектр систем управління базами даних, і ми можемо використовувати його з будь-якою СКБД, якщо доступний відповідний провайдер. Microsoft постачає вбудовані провайдери для таких СКБД, як MS SQL Server, SQLite та PostgreSQL, а також є провайдери від сторонніх постачальників, наприклад MySQL.

Одним з ключових переваг EF Core є його універсальний API для роботи з даними. Якщо ми вирішимо змінити цільову СКБД, основні зміни стосуватимуться конфігурації та підключення до провайдерів, але код, що працює з даними, залишиться без змін.

EF Core успадкував багато речей від свого попередника, Entity Framework 6, але він представляє собою зовсім нову технологію зі своїми власними принципами роботи. Поточна версія EF Core - 5.0, випущена у листопаді 2020 року, і технологія продовжує активно розвиватись.

EF Core можна використовувати на різних платформах стека .NET, включаючи Windows Forms, консольні програми, WPF, UWP та ASP.NET Core. Його кросплатформова природа дозволяє використовувати його не лише на ОС Windows, але і на Linux і Mac OS X.

Центральною концепцією Entity Framework є сутність (entity). Сутність відображає набір даних, пов'язаних з конкретним об'єктом. Замість роботи з таблицями, EF Core дозволяє працювати з об'єктами та їх колекціями. Кожна сутність має набір властивостей, які описують дані про об'єкт. Ці властивості

можуть бути простими типами даних, такими як рядки чи числа, або складнішими типами.

EF Core також підтримує відношення між сутностями, такі як один-до-багатьох, один-до-одного та багато-до-багатьох, аналогічно до зв'язків у базах даних за допомогою зовнішніх ключів.

Одним з унікальних аспектів EF Core як ORM-технології є можливість використання LINQ (Language Integrated Query) для складання запитів до бази даних. За допомогою LINQ, ми можемо створювати різноманітні запити для вибірки об'єктів з бази даних, включаючи запити, які охоплюють пов'язані об'єкти. Під час виконання запиту, EF Core перетворює LINQ-вирази на вирази, зрозумілі для конкретної СКБД (зазвичай, це SQL-вирази).

Загалом, Entity Framework Core є потужним інструментом доступу до даних, який надає високорівневу абстракцію для роботи з базами даних, дозволяючи розробникам працювати з об'єктами та використовувати мову LINQ для складання запитів. Він є гнучким і може бути використаний на різних платформах, що робить його популярним в розробці програмного забезпечення на стеку.[6]

2.3. .NET Core

.NET Core – це потужна і сучасна платформа з відкритим вихідним кодом, розроблена компанією Microsoft для побудови кросплатформних додатків. Її створення було оголошено 12 листопада 2014 року, і з тих пір .NET Core набула широкої популярності серед розробників.

Одна з головних особливостей .NET Core полягає у її кросплатформній природі. Вона розроблена таким чином, щоб працювати на операційних системах Windows, Linux і macOS, надаючи розробникам гнучкість у виборі платформи для своїх додатків. Це робить .NET Core ідеальним вибором для розробників, які прагнуть створити додатки, що працюють на різних операційних системах.

.NET Core володіє потужним інструментарієм, який дозволяє розробникам ефективно створювати різноманітні типи додатків. Наприклад, розробники можуть будувати веб-додатки з використанням ASP.NET Core, що надає широкий набір функціональності для розробки веб-сторінок і веб-служб. Також, .NET Core підтримує розробку консольних додатків, що дає змогу створювати потужні інструменти для автоматизації рутинних завдань. Крім того, розробники можуть створювати бібліотеки, які можуть бути використані в різних проектах, а також програми для універсальної платформи Windows, що дозволяють створювати додатки, що працюють на різних пристроях, включаючи ПК, планшети, смартфони та інші.

Після випуску .NET Core 5.0 в листопаді 2020 року, платформа стала ще більш потужною і функціональною. У цій версії були вирішені патентні проблеми, пов'язані з попередніми версіями .NET Framework, що дозволило розробникам використовувати .NET Core з більшою впевненістю. Крім того, в .NET Core 5.0 було внесено численні покращення і оптимізації, що сприяють підвищенню продуктивності та забезпеченню надійності додатків.

Застосування .NET Core може бути дуже широким, від невеликих персональних проектів до складних корпоративних рішень. Вона знаходить застосування в різних сферах, включаючи фінанси, медицину, розваги, торгівлю та інші галузі. Розробники, які використовують .NET Core, мають доступ до великого набору інструментів, бібліотек і фреймворків, що сприяє швидкому розгортанню та розширенню їх проектів.

Усе це робить .NET Core одним з найпопулярніших виборів серед розробників, які прагнуть створити потужні, ефективні і кросплатформні додатки. Завдяки своїм можливостям, гнучкості і підтримці відкритого вихідного коду, .NET Core продовжує займати важливу позицію в розробці програмного забезпечення.

2.4. ASP.NET Core

ASP.NET Core – це відносно новий веб-фреймворк з відкритим вихідним кодом, розроблений компанією Microsoft. Він був створений з метою надання розробникам гнучкості та ефективності у побудові сучасних веб-додатків.

Однією з ключових переваг ASP.NET Core є його кросплатформність. Фреймворк може працювати як на операційних системах Windows, так і на Linux та macOS. Це дає розробникам можливість створювати веб-додатки для різних платформ без необхідності змінювати код або інфраструктуру.

ASP.NET Core пропонує модульну структуру, яка дозволяє розробникам використовувати лише компоненти та функціональність, які необхідні. Це сприяє підвищенню продуктивності та зменшенню зайвого навантаження на додаток.

Окрім того, ASP.NET Core підтримує сучасні технології, такі як обробка запитів в реальному часі, веб-сокети, вбудована підтримка для розгортання в хмарному середовищі та контейнерах Docker. Це дозволяє розробникам будувати масштабовані та високопродуктивні веб-додатки з мінімальними зусиллями.

Крім того, фреймворк має велику спільноту розробників, яка активно підтримує його розвиток та надає безліч корисних ресурсів, таких як документація, уроки та статті. Це дозволяє новим розробникам швидко освоїти фреймворк та отримати підтримку у вирішенні потенційних проблем.

ASP.NET Core також інтегрується з іншими популярними технологіями, такими як Entity Framework Core для роботи з базами даних, SignalR для реалізації двостороннього зв'язку між клієнтом та сервером, і Identity для автентифікації та авторизації користувачів. Це забезпечує розробникам готові рішення для широкого спектру завдань у веб-розробці.

Загалом, ASP.NET Core є потужним і сучасним веб-фреймворком, який надає розробникам всі необхідні інструменти та можливості для ефективної розробки високоякісних веб-додатків на різних платформах. Його гнучкість, продуктивність та активна спільнота роблять його популярним вибором серед розробників.[7]

2.5. ASP.NET Core Web Api

ASP.NET Core Web API - це спеціалізований метод розробки програм, який зосереджений на створенні програм з урахуванням REST-підходу. REST-архітектура використовує різні HTTP-запити або методи, такі як:

— GET

— POST

— DELETE

— PUT

— PATCH

Часто REST-стиль виявляється особливо зручним при створенні різноманітних односторінкових додатків (Single Page Application), які часто використовують популярні JavaScript-фреймворки, наприклад Angular, React або Vue.js. По суті, Web API представляє собою веб-службу, до якої можуть звертатися інші програми. Ці програми можуть використовувати будь-які технології та платформи, включаючи веб-додатки, мобільні додатки або десктопні клієнти.

2.6. SMTP Client

SMTP Client - це бібліотека, що дозволяє програмам відправляти електронні листи через протокол Simple Mail Transfer Protocol (SMTP). Для створення та відправлення листів за допомогою SmtpClient потрібно вказати наступну інформацію:

— Сервер SMTP, який буде використовуватись для надсилання листів.

— Облікові дані для аутентифікації, якщо потрібно для використання SMTP-сервера.

— Адреса електронної пошти відправника.

— Адреси електронної пошти отримувачів.

— Зміст повідомлення.

Щоб додати вкладення до листа, спочатку потрібно створити об'єкт Attachment за допомогою класу Attachment, а потім додати його до повідомлення за допомогою властивості MailMessage.Attachments. Після додавання всієї необхідної інформації можна викликати метод Send або SendAsync для відправлення електронного листа.[9]

2.7. AutoMapper

AutoMapper є однією з бібліотек, яка забезпечує можливість перетворення об'єктів з одного типу в інший шляхом копіювання значень їх полів. Вона дозволяє створити копію об'єкта, змінивши його тип. Для кожної пари типів можна налаштувати відображення, вказавши, в яке поле першого типу треба записати конкретне поле другого типу.

Крім того, застосувавши метод ReverseMap() під час конфігурації, можна зробити двостороннє відображення, що дозволяє перетворювати об'єкт з типу A в тип B і навпаки. AutoMapper зберігає багато часу під час розробки програми, оскільки вона виконує всю рутинну роботу з перетворення об'єктів, звільняючи розробника від цих завдань.

Часто AutoMapper застосовується в багатошаровій архітектурі для передачі моделей між рівнями, але також може бути використаний у звичайних проектах для зручності та покращення читабельності коду.[4]

2.8. React.js

React, також відомий як React.js або ReactJS, є відкритою JavaScript бібліотекою для розробки користувацьких інтерфейсів. Вона дозволяє створювати односторінкові та мобільні додатки, надаючи високу швидкість, простоту

використання та масштабованість. React зазвичай використовується в поєднанні з іншими бібліотеками, такими як MobX, Redux і GraphQL, для розробки інтерфейсів користувача.

React був створений Джорданом Валке, розробником програмного забезпечення в Facebook, і його вихідний код був опублікований у травні 2013 року на конференції "JSConf US". Ця бібліотека використовує віртуальний DOM (Data Object Model), що дозволяє ефективно обчислювати різницю між попереднім і поточним станом інтерфейсу для оптимального оновлення DOM браузера. Вона створює кеш-структуру в пам'яті, яка автоматично вирішує, які компоненти сторінки потребують оновлення, звільняючи програміста від цієї рутинної задачі.

Кожен компонент у React має свій життєвий цикл, який складається з різних стадій. Розробнику надаються спеціальні методи, які можна використовувати на різних етапах життєвого циклу компонента. Наприклад, метод “shouldComponentUpdate” дозволяє уникнути зайвого перемальовування компонента, повертаючи значення false, якщо оновлення не є необхідним. Метод “componentDidMount” викликається після першого відмальовування компонента і часто використовується для отримання даних з віддаленого джерела за допомогою API. Найважливішим методом життєвого циклу є метод “render”, який відповідає за відображення даних компонента в інтерфейсі та викликається при зміні даних компонента для їх перемальовування.[2][8]

2.9. Axios

Axios - це клієнт для виконання HTTP-запитів, який працює на основі Promise як у середовищі Node.js, так і у браузері. На серверній стороні він використовує вбудований модуль http в Node.js, а на клієнтській стороні використовує XMLHttpRequests.[5]

Основні особливості Axios:

— Дозволяє виконувати запити з використанням XMLHttp у браузері.

— Дозволяє виконувати HTTP-запити у середовищі Node.js.
— Підтримує Promise API, що дозволяє зручно працювати з асинхронними операціями.
— Можливість перехоплювати запити та відповіді для здійснення додаткових операцій.
— Забезпечує можливість трансформувати (парсити) дані запиту та відповіді у зручний формат.
— Дозволяє скасовувати запити у разі необхідності.
— Дозволяє автоматично перетворювати дані у формат JSON.
— Забезпечує підтримку на клієнтській стороні для захисту від атаки XSRF (Cross-Site Request Forgery).

3. ОПИС ПРОГРАМИ

3.1. Програмна реалізація

3.1.1. Загальний опис реалізації

У програмі користувач може мати одну з трьох ролей: адміністратор, студент,викладач.

Після реєстрації у будь-якого користувача буде можливість:

1. Війти в систему;
2. Редагувати особисту інформацію про себе;
3. Зіграти міні гру на головній сторінці;
4. Вийти з системи.

У студента також буде можливість:

1. Зарахування на тест по посиланню викладача;
2. Перегляд всіх тестів, на які користувач був зарахований;
3. Перегляд інформації про конкретний тест;
4. Проходження тесту.

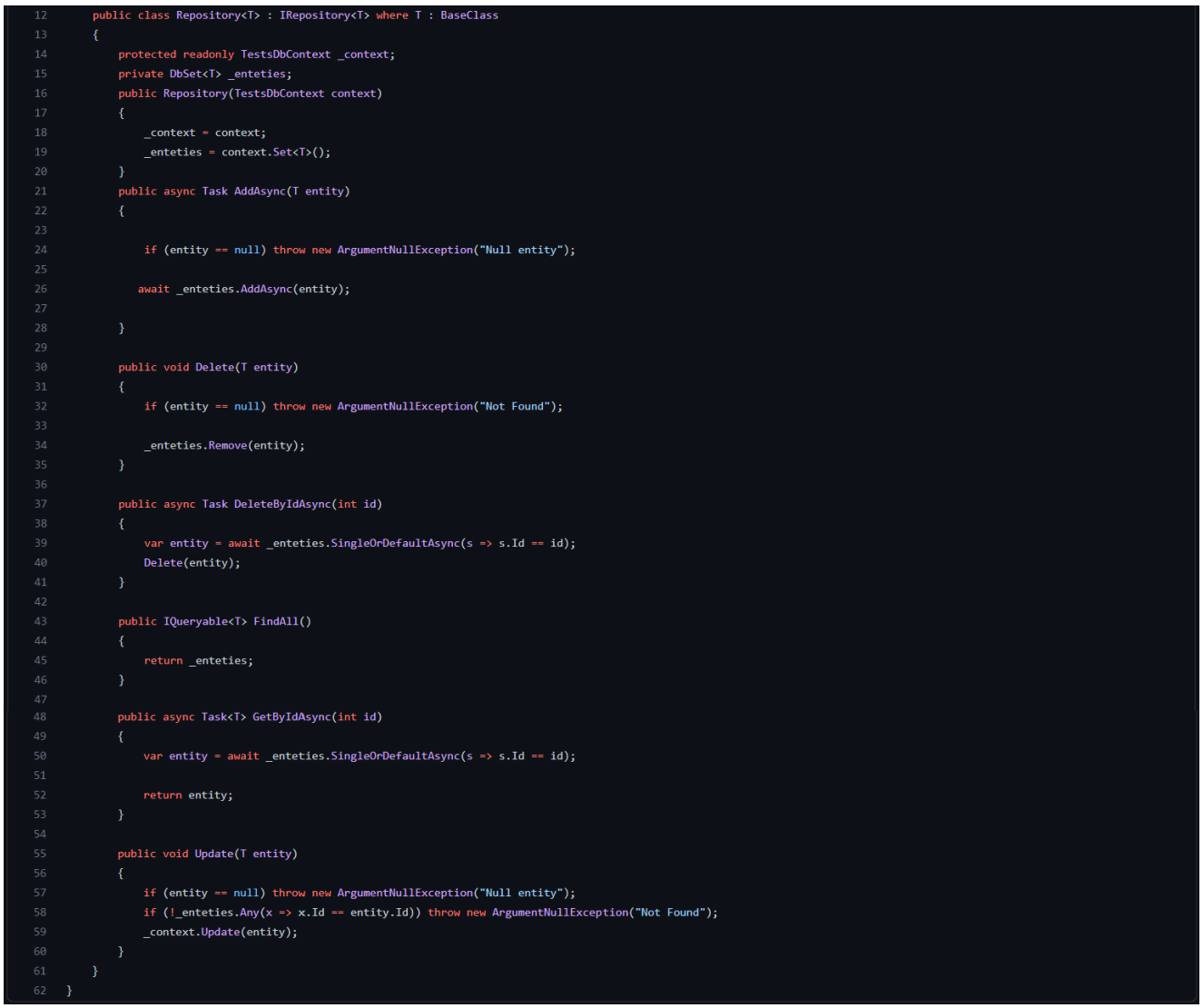
Якщо у користувача роль адміністратора, то він може:

1. Ввійти в систему;
2. Переглядати всіх користувачів;
3. Редагувати інформацію про іншого користувача;
4. Змінювати користувачу роль зі студента на викладача.

Після призначення користувачу ролі викладача в нього з’являється такий функціонал:

1. Створити нове питання;
2. Переглядати список власних питань;
3. Переглядати загальний список питань;
4. Переглядати конкретне питання;
5. Видаляти власні питання;
6. Копіювати публічні питання іншого викладача;
7. Створити тест;
8. Переглядати список своїх тестів;
9. Переглядати конкретну інформацію про тест;
10. Копіювати посилання для зарахування на тест;
11. Надсилати запрошення на тест електронною поштою;
12. Переглядати журнал оцінок за певний проміжок часу;

Для обробки даних я використовую Entity Framework Core. Для роботи з окремими сутностями я реалізував патерн Generic Repository, в якому містяться методи для створення сутності, оновлення та видалення вже існуючої, а також для отримання всіх існуючих сутностей і отримання конкретної сутності по ідентифікатору.



```
    public class Repository<T> : IRepository<T> where T : BaseClass
    {
        protected readonly TestsDbContext _context;
        private DbSet<T> _enteties;
        public Repository(TestsDbContext context)
        {
            _context = context;
            _enteties = context.Set<T>();
        }
        public async Task AddAsync(T entity)
        {

            if (entity == null) throw new ArgumentNullException("Null entity");

           await _enteties.AddAsync(entity);

        }

        public void Delete(T entity)
        {
            if (entity == null) throw new ArgumentNullException("Not Found");

            _enteties.Remove(entity);
        }

        public async Task DeleteByIdAsync(int id)
        {
            var entity = await _enteties.SingleOrDefaultAsync(s => s.Id == id);
            Delete(entity);
        }

        public IQueryable<T> FindAll()
        {
            return _enteties;
        }

        public async Task<T> GetByIdAsync(int id)
        {
            var entity = await _enteties.SingleOrDefaultAsync(s => s.Id == id);

            return entity;
        }

        public void Update(T entity)
        {
            if (entity == null) throw new ArgumentNullException("Null entity");
            if (!_enteties.Any(x => x.Id == entity.Id)) throw new ArgumentNullException("Not Found");
            _context.Update(entity);
        }
    }
}
```

Рисунок 3.1 – Реалізація Generic Repository

Оскільки сутності Test і Question складніші ніж решта і містять в собі полязі складними типами, то для роботи з ними функціоналу цього репозиторію недостатньо, тому для кожної з цих сутностей було розроблено окремий репозиторій, який розширює Generic Repository. В кожному з них є 2 додаткові методи, а саме отримання всіх сутностей з деталями і отримання сутності з деталями по ідентифікатору

Основним завданням моєї програми є генерування випадкового варіанту тесту і перевірка тесту з подальшим оцінюванням студента.

Для генерації тесту я дістаю список тем, які викладач призначив при створенні тесту, і кількість питань для кожної з них. Далі для кожної теми, з допомогою LINQ, дістаються всі питання, які належать викладачу і складність яких нижча або рівна складності тесту, яку обрав студент. Після цього випадковим чином обирається необхідна кількість питань і додається до тесту. Після додавання питань зі всіх тем, тест видається студенту, а після його проходження повертається на сервер для перевірки.

Під час перевірки для кожного питання дістаються всі правильні відповіді для нього і порівнюються з відповідями студента. В залежності від типу питання можливі три варіанти:

1. Якщо питання має лише одну правильну відповідь, то студенту додається максимальна оцінка за питання, якщо його відповідь є правильною, або не додається нічого якщо відповідь не правильна.
2. Якщо питання має декілька правильних відповідей, то за кожну правильну відповідь студенту додається часткова оцінка, яка дорівнює максимальній оцінці за питання, поділеній на кількість правильних відповідей, а за кожну неправильну відповідь від оцінки за питання віднімається часткова оцінка.
3. Якщо питання має вільну відповідь (студенту потрібно самому її надрукувати), то якщо відповідь студента міститься в списку можливих правильних відповідей, то студенту буде додана максимальна оцінка за питання

Після проходження тесту знаходиться коефіцієнт, який дорівнює максимальній оцінці за тест, поділеній на суму максимальних оцінок за всі питання, після чого фактична оцінка за питання студента множиться на цей коефіцієнт і отримується оцінка студента за тест.

3.1.2. База даних

У своїй програмі я використовую СКБД PostgreSQL. Моя база даних містить 7 таблиць

У таблиці Roles міститься інформація про всі ролі користувачів.

У таблиці Users міститься інформація про всіх зареєстрованих користувачів.

Кожен користувач має свою роль.

У таблиці Questions міститься інформація про питання, які були створені викладачами. Кожен викладач може мати багато питань

У таблиці Answers зберігаються відповіді до питань. Кожне питання може мати одну або більше відповідей.

У таблиці SuggestedAnswers зберігаються відповіді до питань, з яких студенту буде запропоновано обрати відповідь.

У таблиці Tests зберігається інформація про всі тести, як і створені викладачами так і пройдені студентами. Тест посилається на двох користувачів: викладача, який його створив, та студента, якому він призначений.

У таблиці ThemeCounts зберігається інформація про те, скільки питань з якої теми має включати в себе конкретний тест. Кожен тест посилається на один або більше об’єктів цієї таблиці.

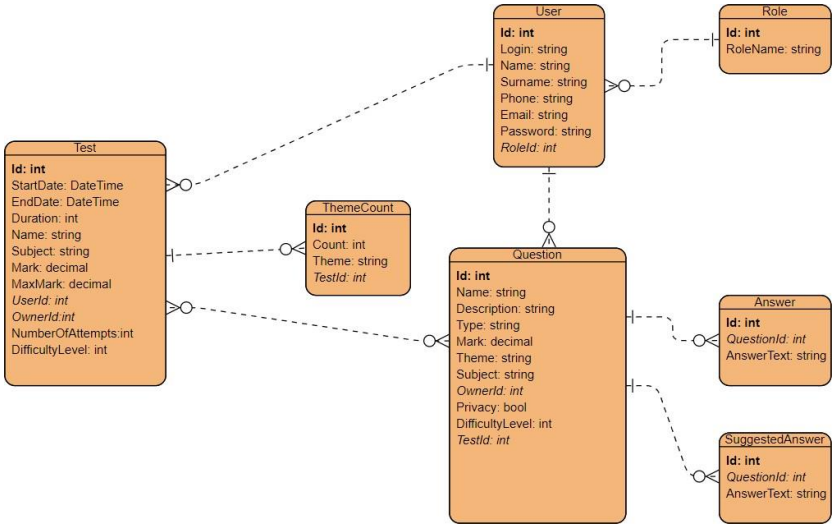


Рисунок 3.2 – ER-діаграма

3.1.3. Архітектура програми

Сервер програми побудований з використанням тришарової архітектури. Перший шар – Data Acces Layer містить у собі реалізацію підключення до Бази Даних і роботи з нею з використанням Entity Framework Core.

Другий шар – Business Logic Layer містить в собі всі необхідні методи для опрацювання даних, такі як згенерувати тест, отримати всі питання певного користувача, зареєструватись і т.д.

Третій шар – Presentation Layer містить у собі необхідну логіку для спілкування з клієнтською частиною програми, а також перевіряє аутентифікацію і авторизацію користувача, який використовує програму.

Для перетворення типів і передачі об’єктів між шарами я використовую AutoMapper.

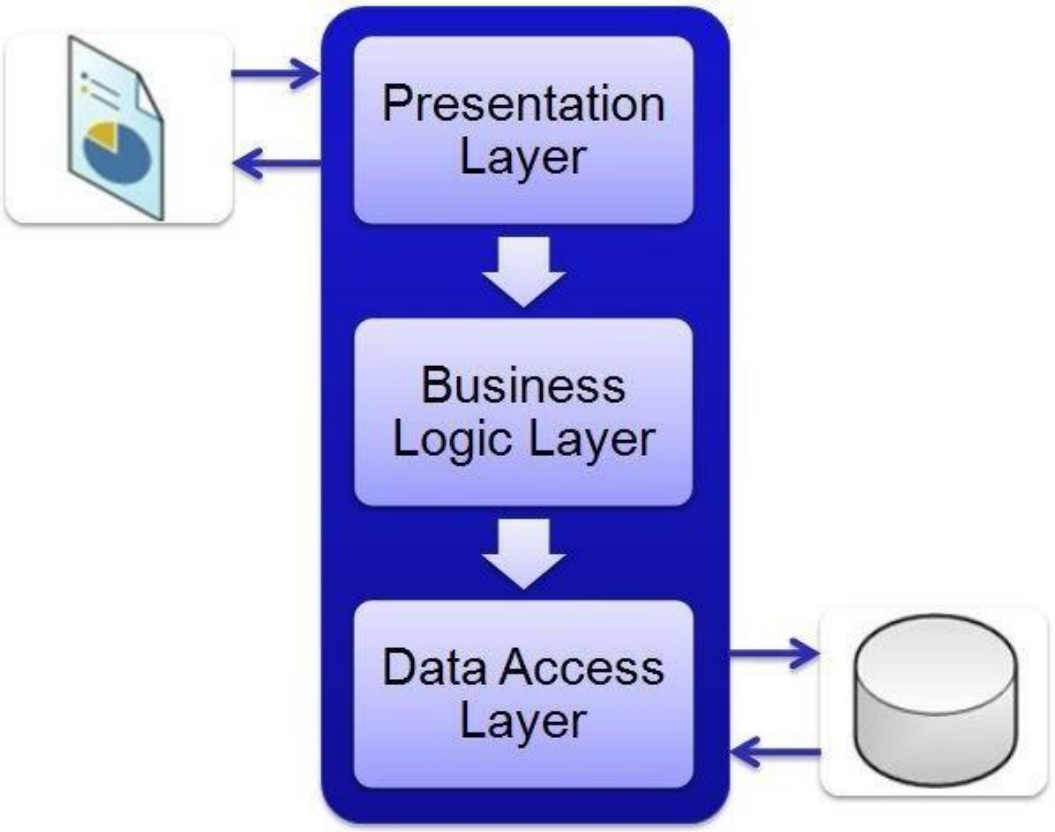


Рисунок 3.3 – Загальна схема тришарової архітектури

3.2. Функціональні можливості

3.2.1. Система керування обліковим записом

Моя програма дозволяє користувачу зареєструватись, заповнивши просту форму. Цю процедуру мають виконувати як студенти, так і викладачі.

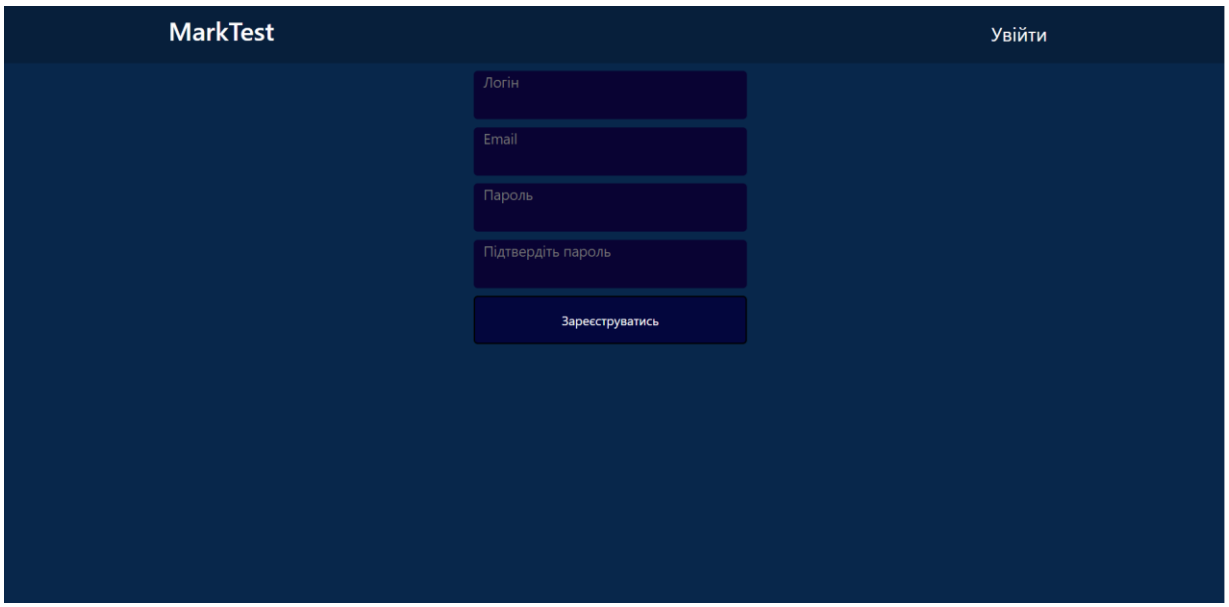


Рисунок 3.4 – Сторінка реєстрації

Після реєстрації користувач може перейти в особистий кабінет, в якомубудуть відображатись необхідні йому посилання.

Перейшовши по посиланню для редагування профілю, користувач може вказати нову або відредагувати вже існуючу додаткову інформацію про себе.

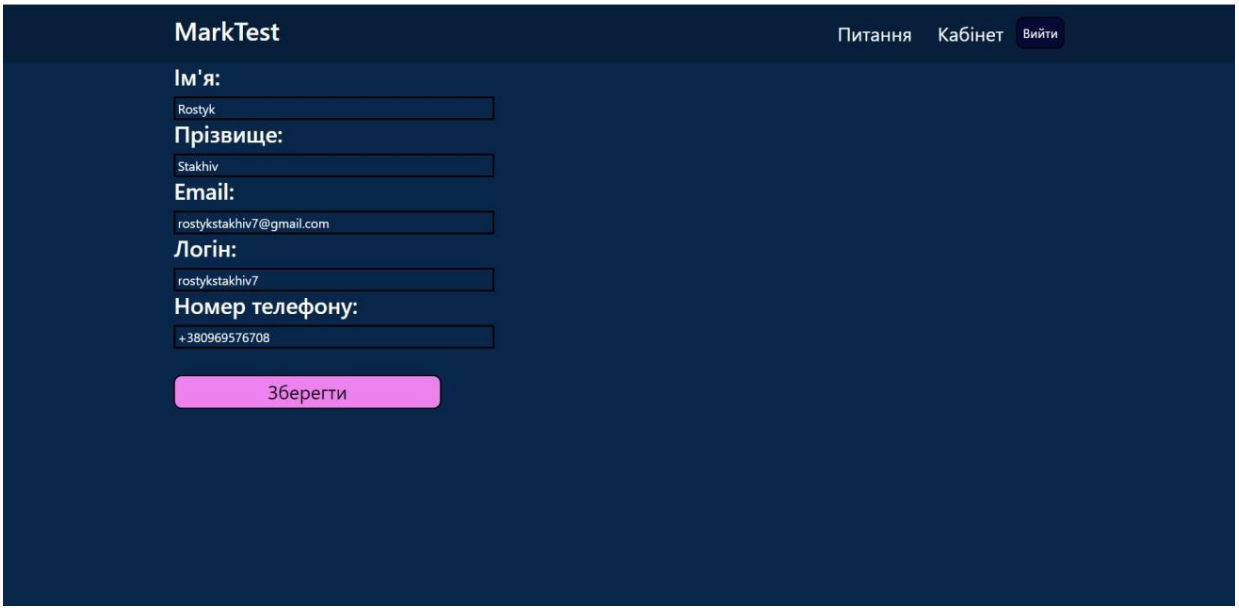


Рисунок 3.5 – Сторінка редагування користувача

У студента також відображається посилання на тести, на які він зареєструвався.

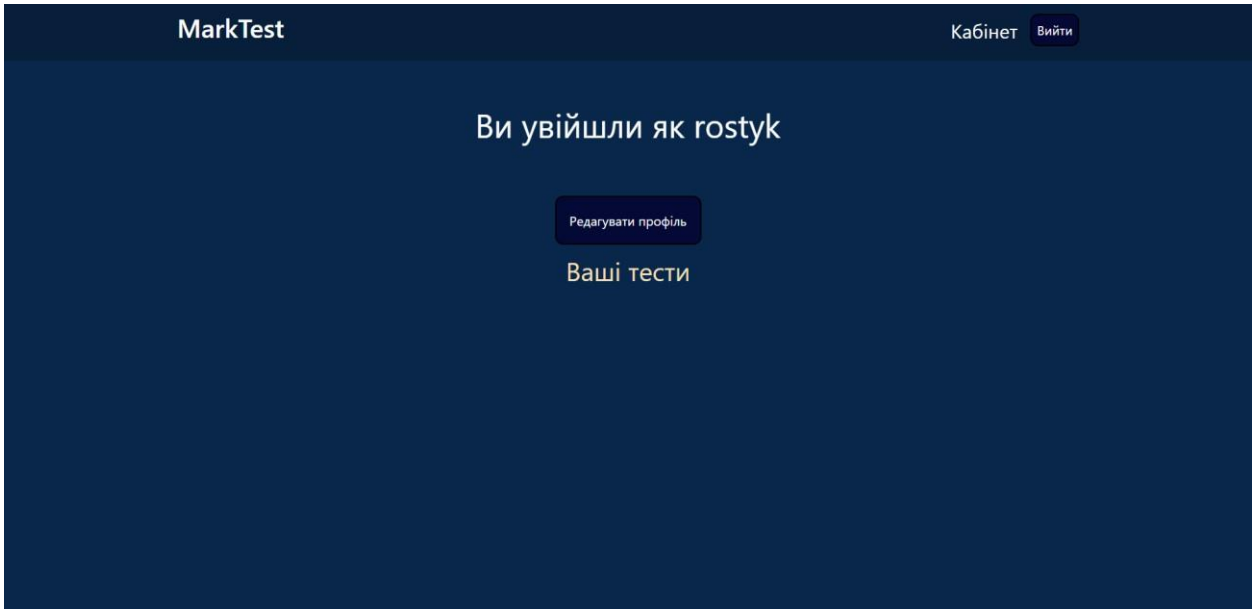


Рисунок 3.6 – Особистий кабінет студента

У викладача є посилання на його власні тести та питання.

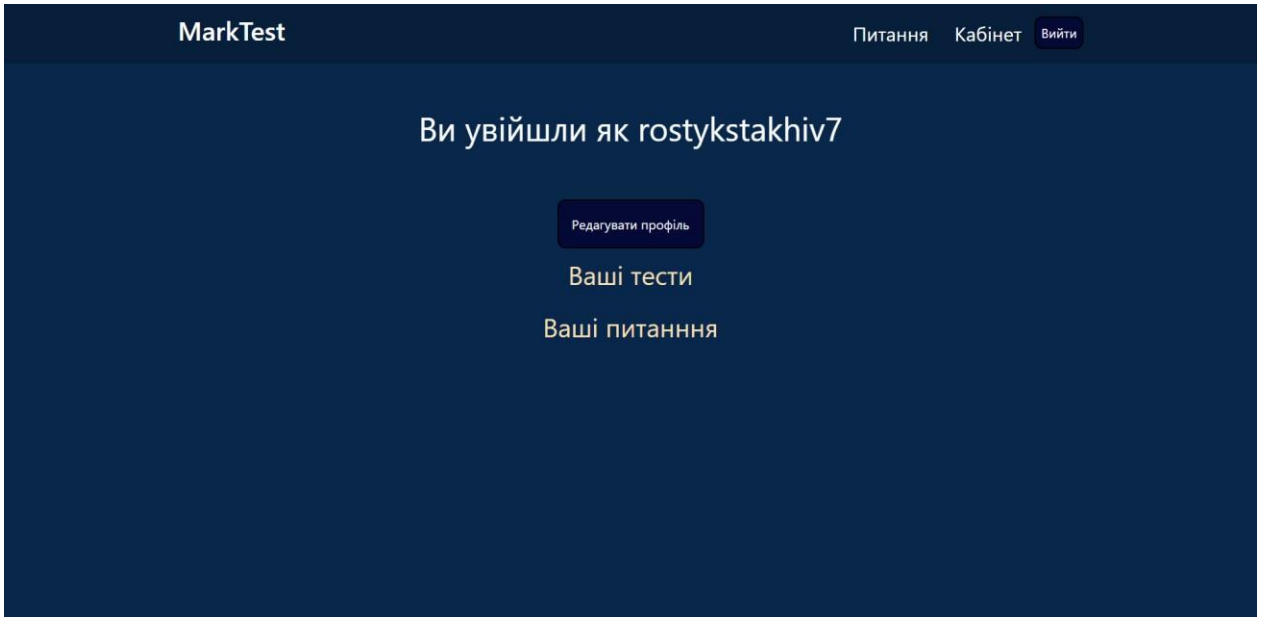


Рисунок 3.7 – Особистий кабінет викладача

Для надання профілю прав викладача, користувач має надіслати лист-заявкуна електронну пошту, в якому буде вказано логін профілю, з допомогою якого адміністратори платформи можуть ідентифікувати профіль і надати йому відповідні права, а також, при необхідності, змінити інформацію про користувача.

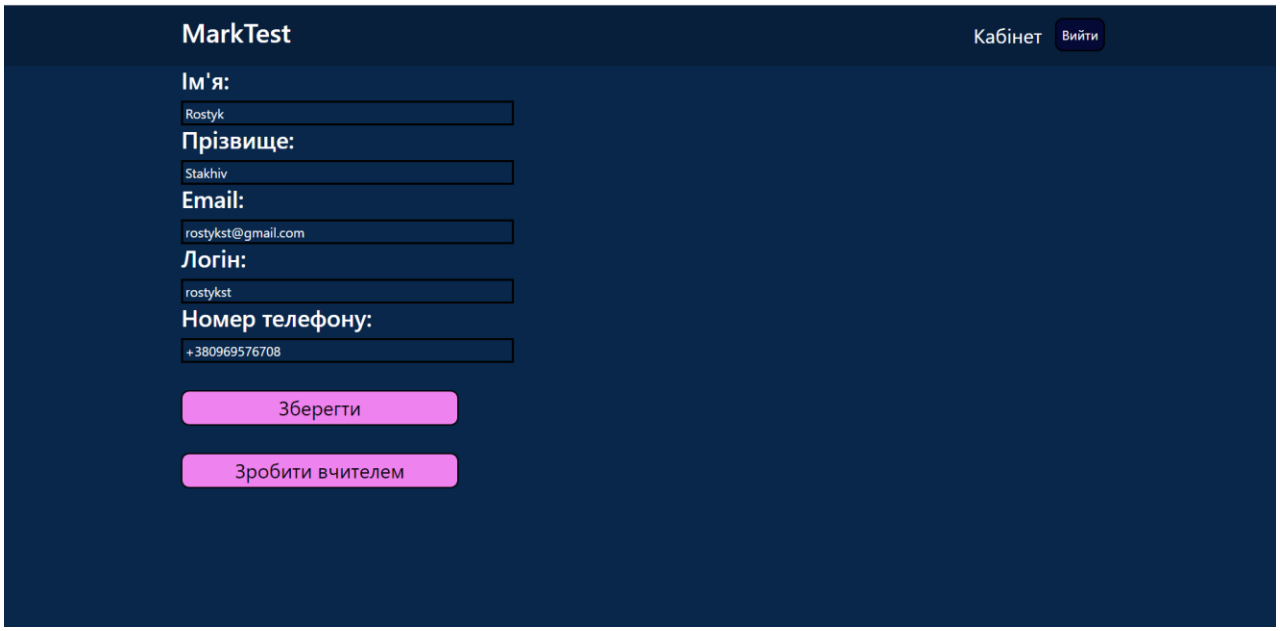


Рисунок 3.8 – Сторінка редагування користувача адміністратором

У адміністратора є посилання на користувачів.

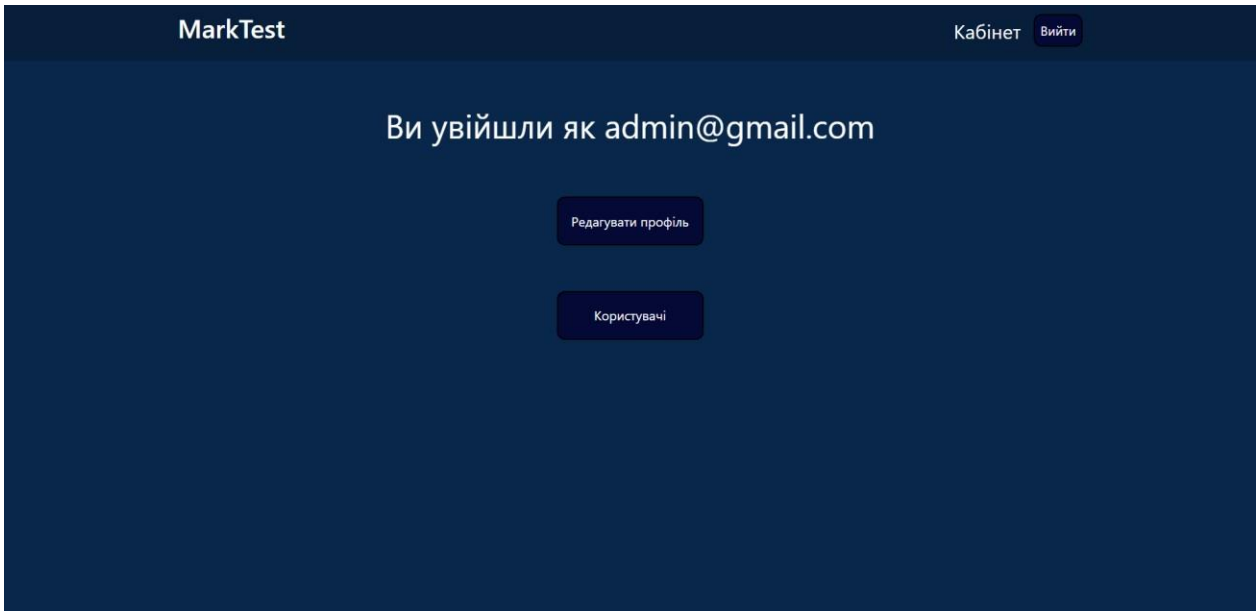


Рисунок 3.9 – Особистий кабінет адміністратора

Перейшовши по цьому посиланню, адміністратор може переглядати всіх користувачів і, клікнувши по користувачу, може перейти на сторінку його профілю.

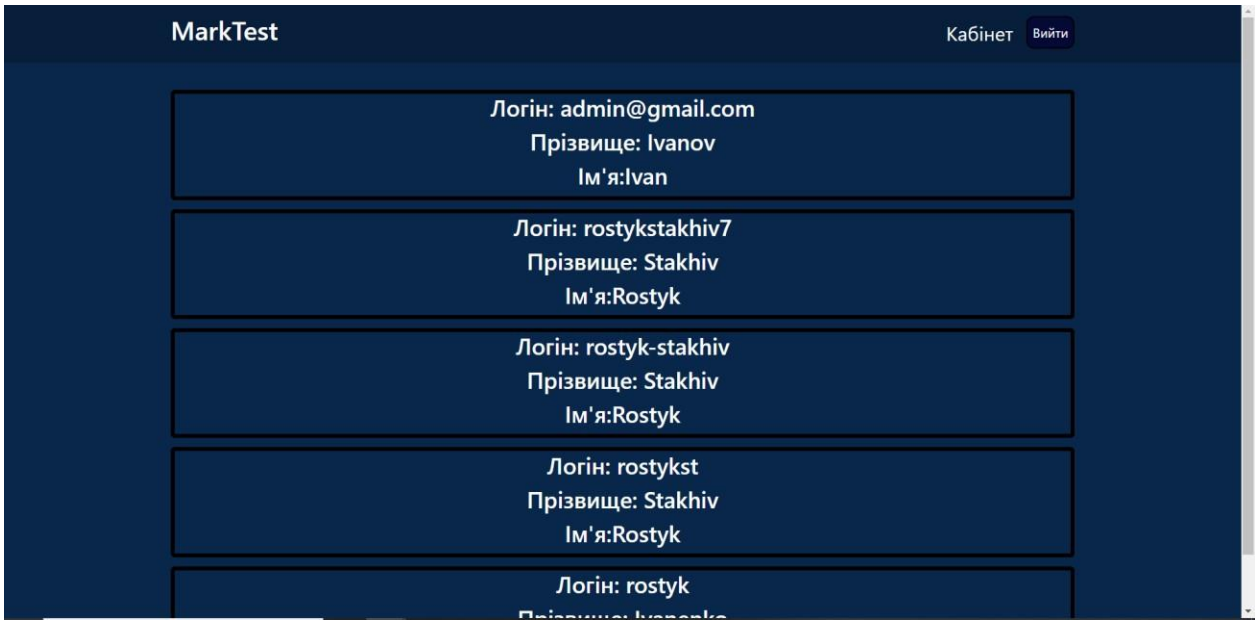


Рисунок 3.10 – Сторінка користувачів

3.2.2. Система керування питаннями

Викладач може створювати свої питання, кожному з яких має бути присвоєний предмет і тема, до яких воно відноситься. Для питання можна обрати один з трьох типів відповіді: єдина відповідь, декілька відповідей, або запропонувати студенту самому ввести відповідь. У спеціальному вікні відразу можна вказати, яка відповідь буде правильною. Також можна визначити чи питання є приватним чи публічним.

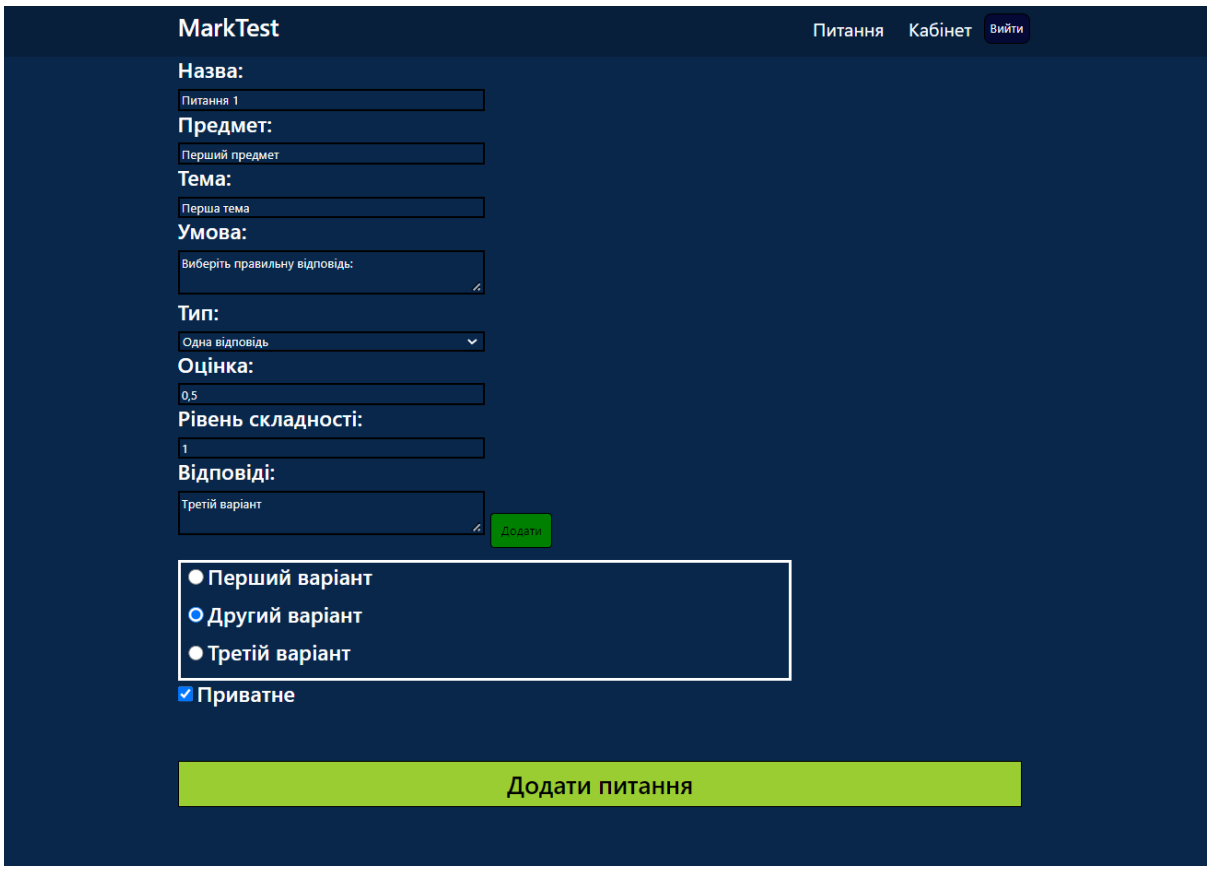


Рисунок 3.11 – Сторінка створення питання

Викладач може переглядати список своїх питань, перейшовши по відповідному посиланню у своєму особистому кабінеті, а також список питань інших викладачів, перейшовши по посиланню в заголовку сайту.

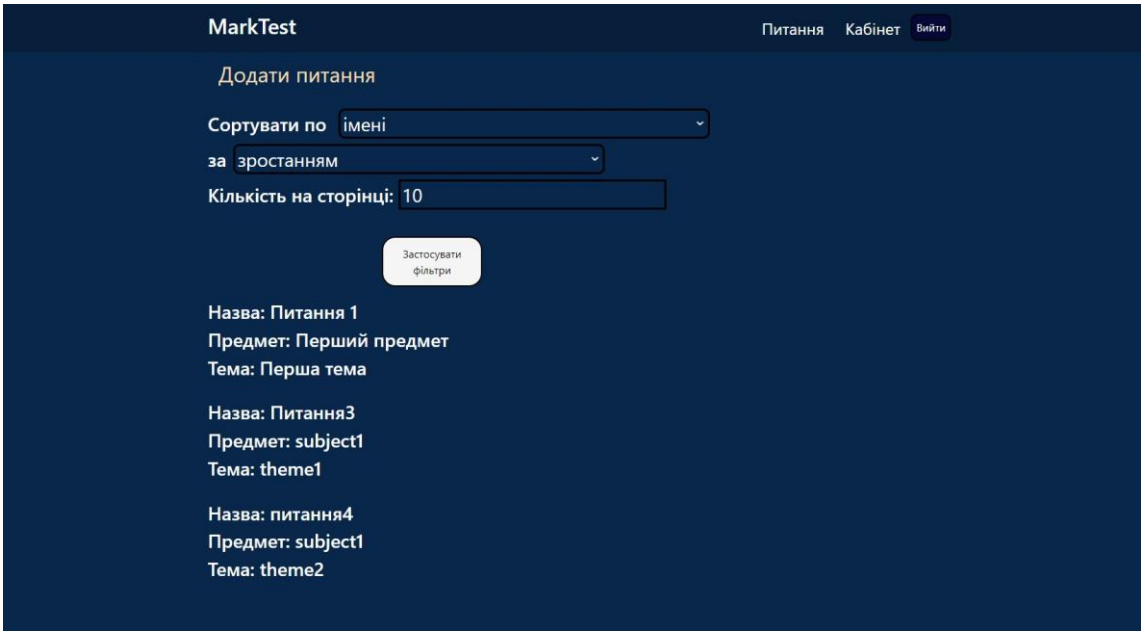


Рисунок 3.12 – Сторінка перегляду питань

Клікнувши по питанні у списку, викладач відкриє попередній перегляд питання, де також буде кнопка видалення, якщо викладач є власником питання

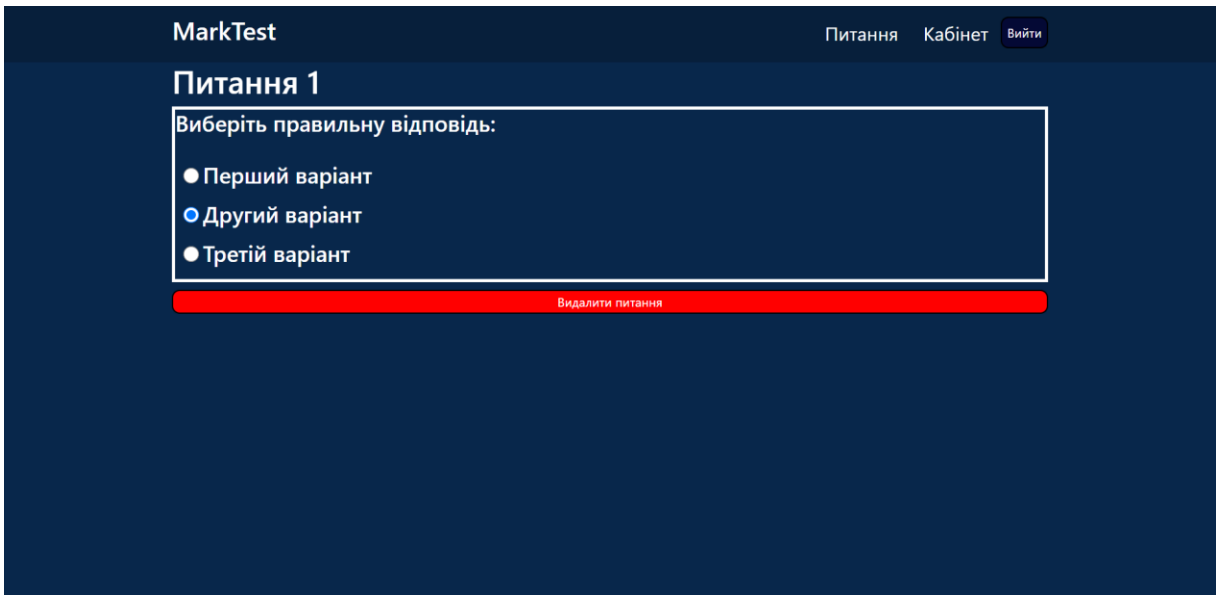


Рисунок 3.13 – Сторінка попереднього перегляду власного питання

Викладач може скопіювати питання до своєї колекції, якщо воно є публічним і викладач не є його власником.

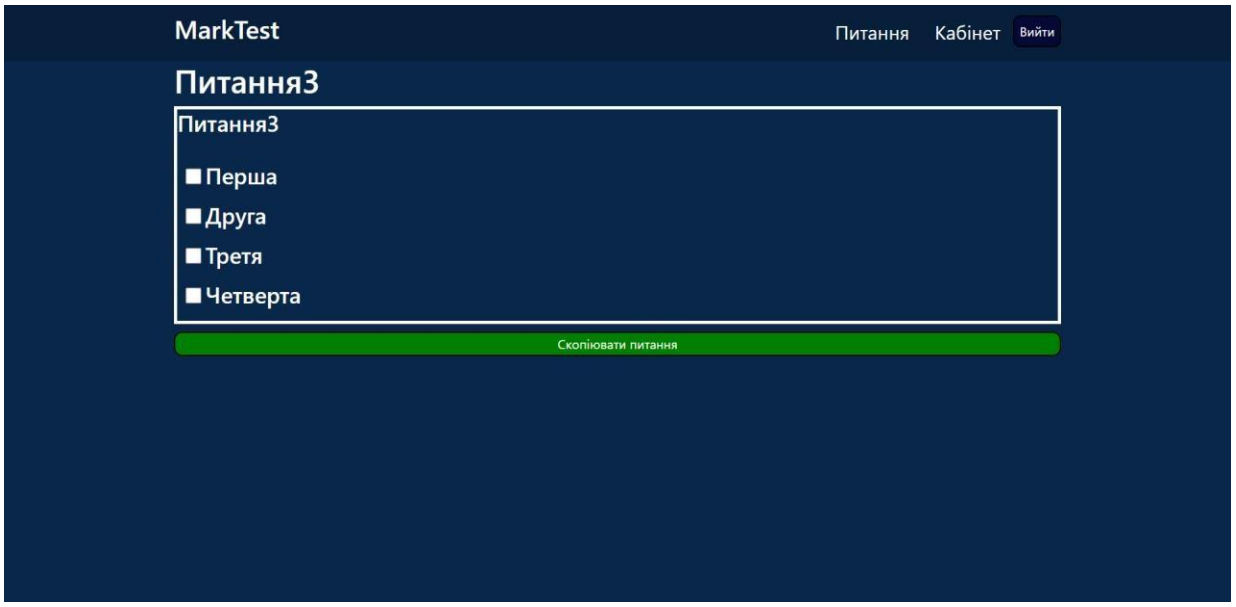


Рисунок 3.14 – Сторінка попереднього перегляду публічного питання іншого викладача

3.2.3. Система керування тестами

Викладач може створювати тести, кожен має відноситись до якогось предмету, викладач може вибирати скільки питань і з якої теми має містити тест, ще можна обрати дату і час початку і завершення тесту, а також його тривалість в хвилинах.

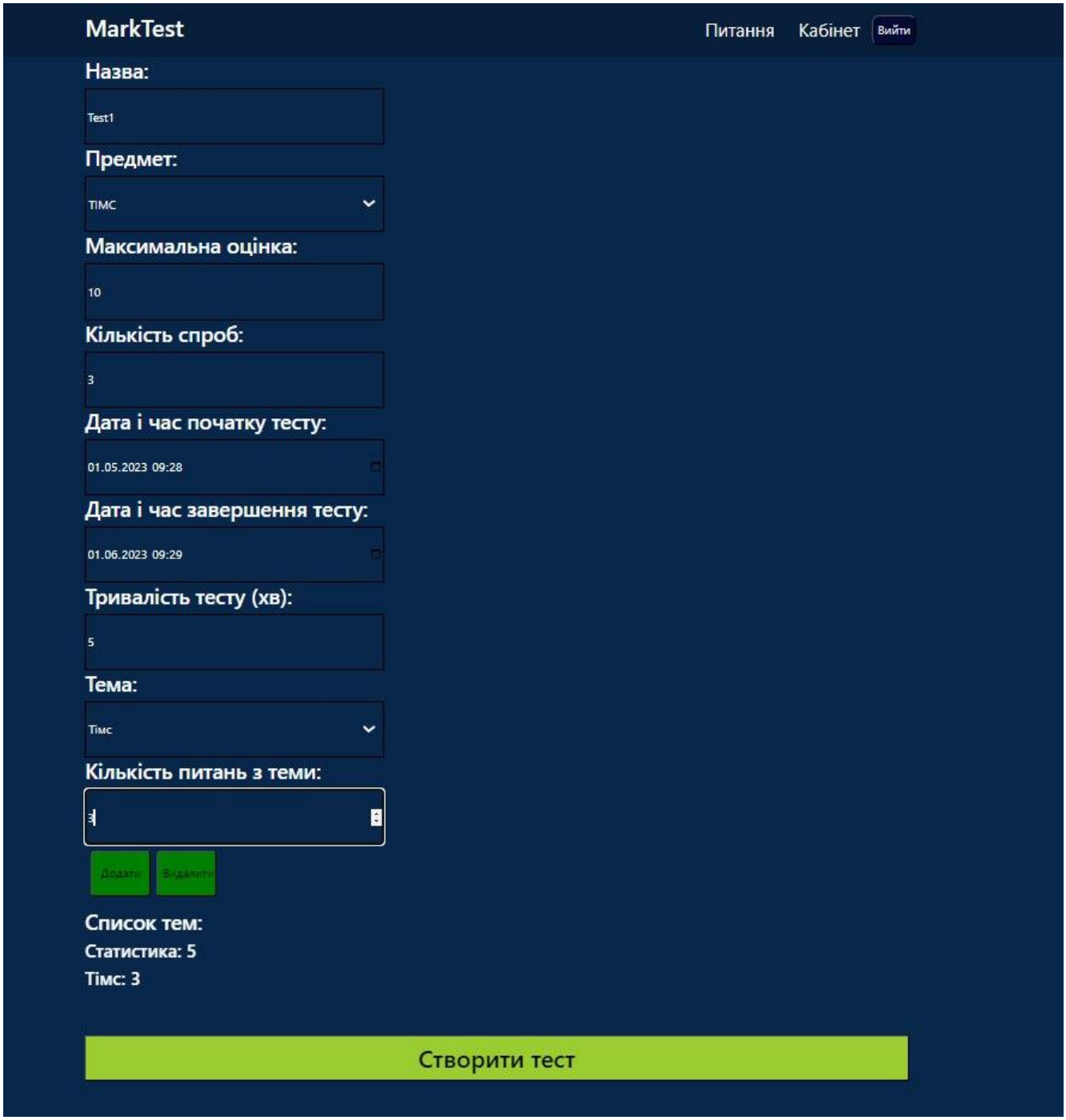


Рисунок 3.15 – Сторінка створення тесту

Після створення тест відобразиться на сторінці тестів викладача, де можна знайти всі власні тести, вказавши назву предмету.

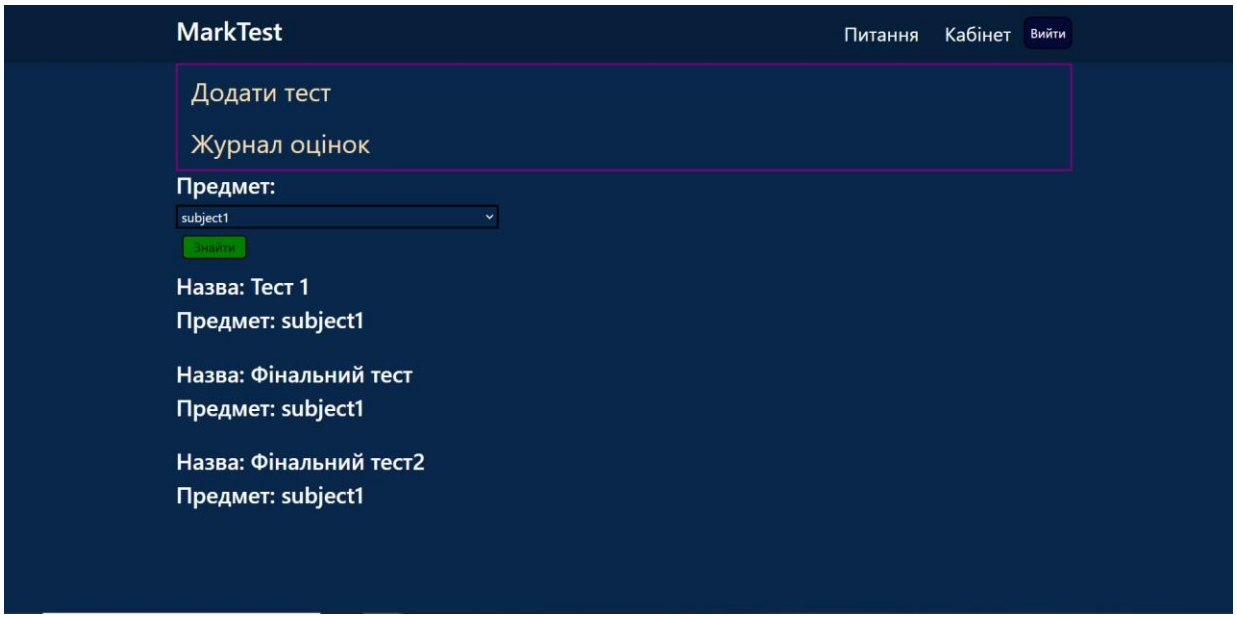


Рисунок 3.16 – Сторінка перегляду власних тестів

Клікнувши на якийсь тест зі списку, можна буде перейти на його сторінку. На сторінці тесту буде відображатись посилання для зарахування на цей тест, яке викладач зможе скопіювати і поширити серед студентів. Також у викладача є можливість відправити запрошення на електронну пошту студентів, для цього йому потрібно ввести електронні адреси студентів у спеціальне поле.

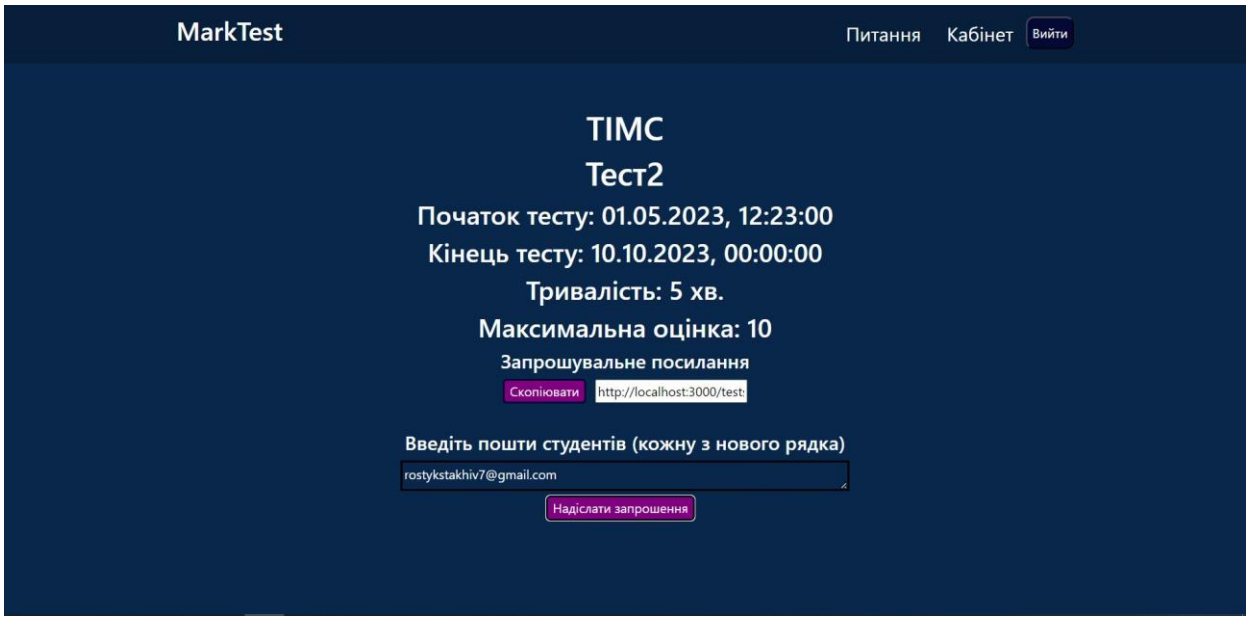


Рисунок 3.17 – Сторінка перегляду тесту викладачем

У такому випадку студенти отримають лист-запрошення на тест**.** Якщо студент перейде по посиланню, то його буде зараховано на тест, про що його буде сповіщено, а також тест відобразиться у списку тестів, де можна буде перейти на його сторінку і пройти у час, визначений викладачем.

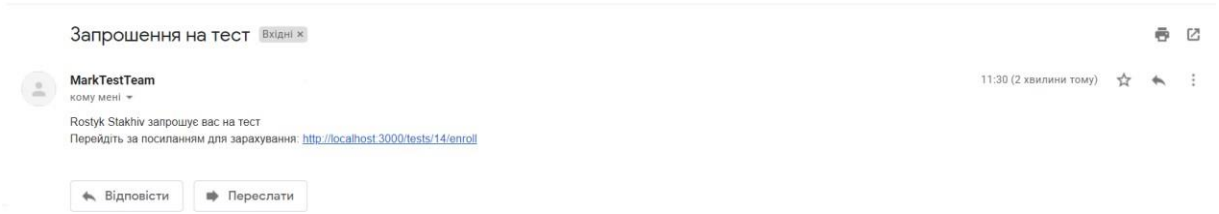


Рисунок 3.18 – Лист запрошення на тест

Перед проходженням тесту студенту буде запропоновано обрати рівень складності для тесту. Всього рівнів є три: легкий, який пропонує найпростіші питання і максимальна оцінка якого складає 30% від загальної максимальної оцінки тесту, посередній, який пропонує суміш з простих і середніх питань і максимальну оцінку в 60% від загальної і складний, який буде складатись зі питань будь-якої складності і максимальна оцінка за який буде рівна загальній.

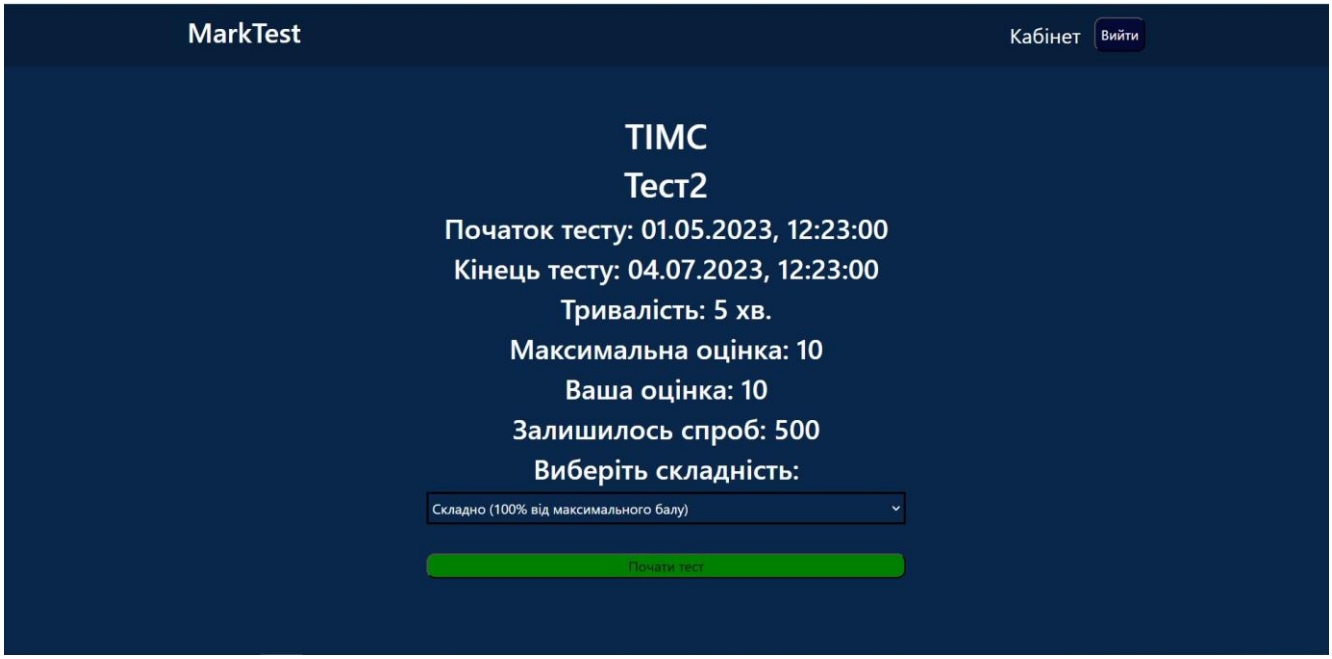


Рисунок 3.19 – Сторінка перегляду тесту студентом

Під час проходження тесту студент може вільно переміщатись між питаннями, а також питання, на які студент вже дав відповідь будуть відмічатись спеціальною відміткою.

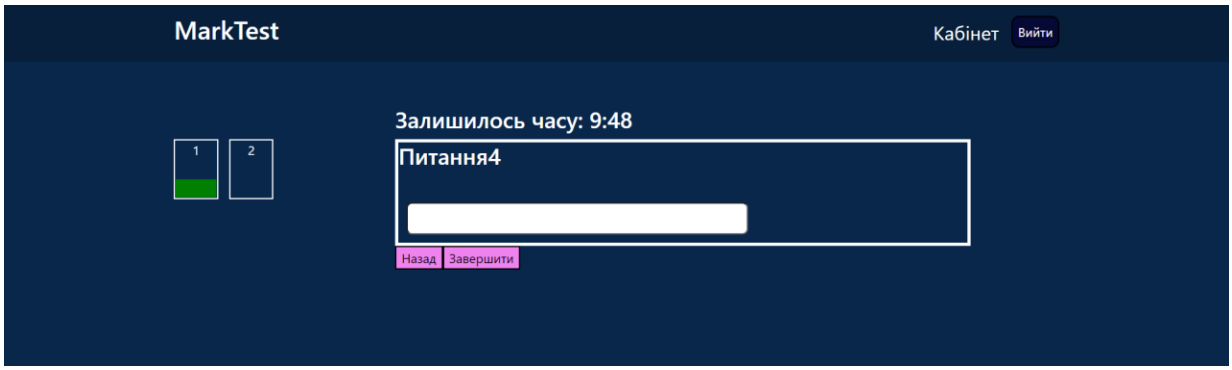


Рисунок 3.20 – Сторінка проходження тесту

Після завершення тесту, студенту відобразиться результат тесту, який такожможна буде побачити, якшо перейти на сторінку тесту.

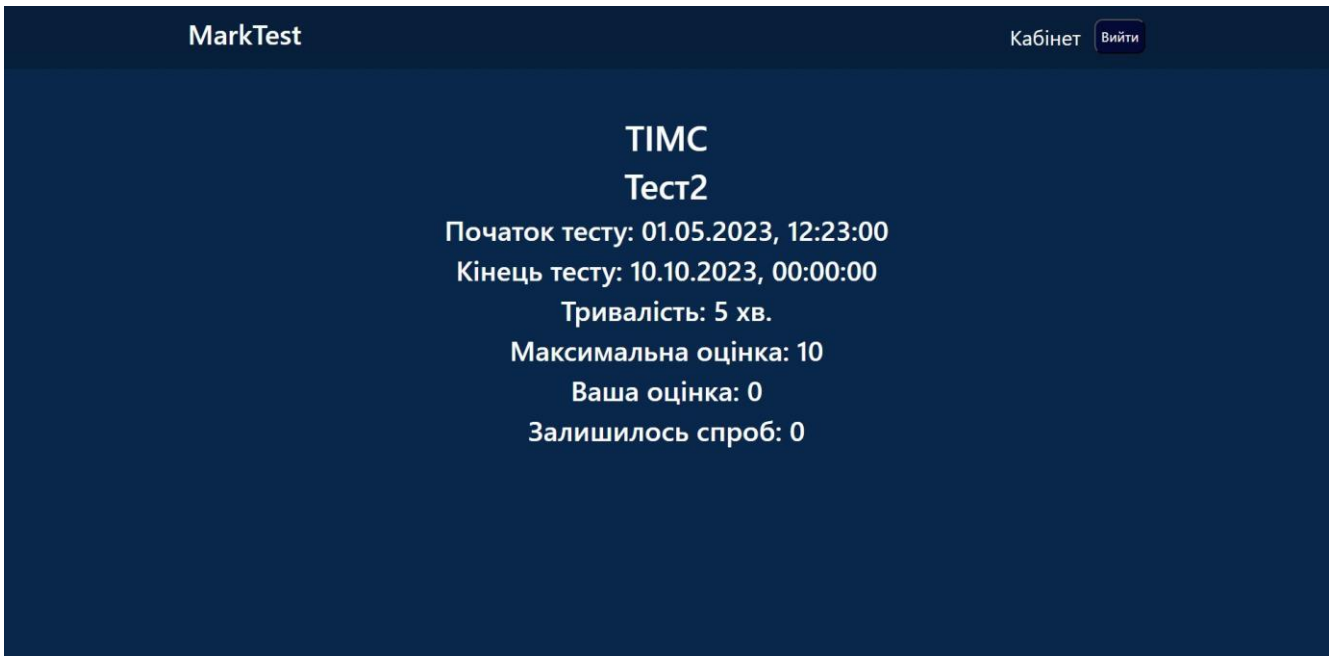


Рисунок 3.21 – Сторінка перегляду тесту після його проходження

Після того як студенти пройдуть тест, викладач може зайти в журнал оцінокі вибравши предмет, назву тесту і проміжок часу, переглянути результати студентів.

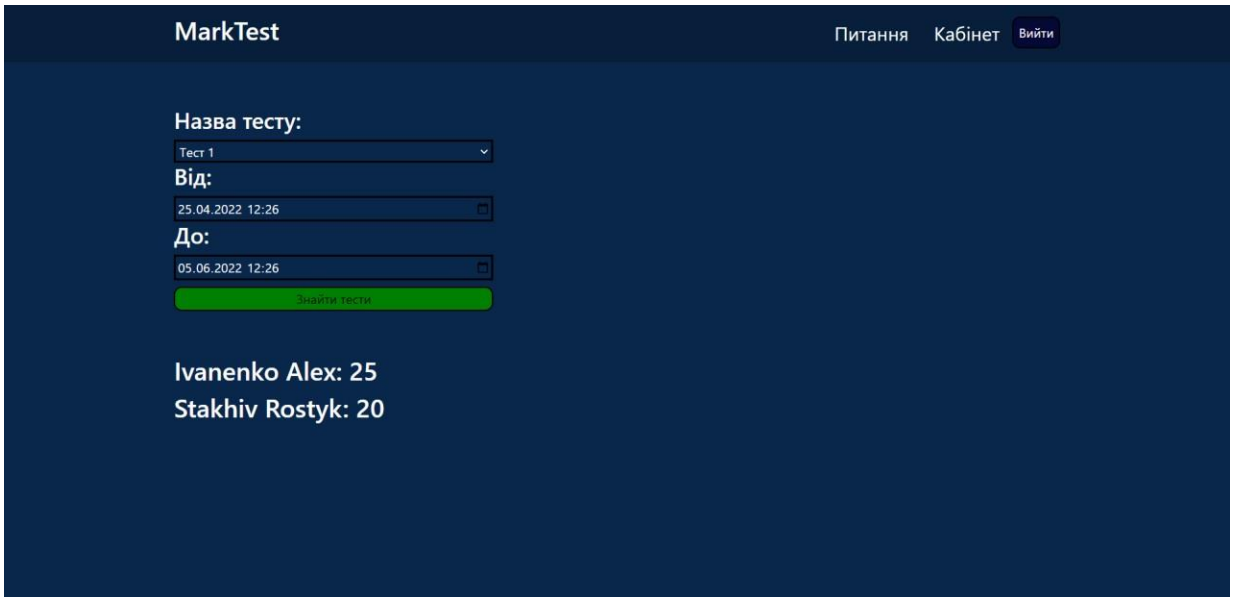


Рисунок 3.22 – Журнал оцінок

3.2.4. Додатковий функціонал

На головній сторінці сайту будь-який користувач може зіграти у міні гру у випадку, якщо потрібно почекати початку тесту, або у випадку надмірного стресу

Рисунок 3.23 – Головна сторінка

Також було створено механізми захисту від шахрайства. При переході на інше вікно або вкладку під час проходження тесту на час, більший 10 секунд, тест завершиться. При спробі зробити скріншот або скопіювати текст питання, зміст питання чи знімок екрану не збережеться в буфер обміну, замість них туди потрапить прохання не списувати на тестах.

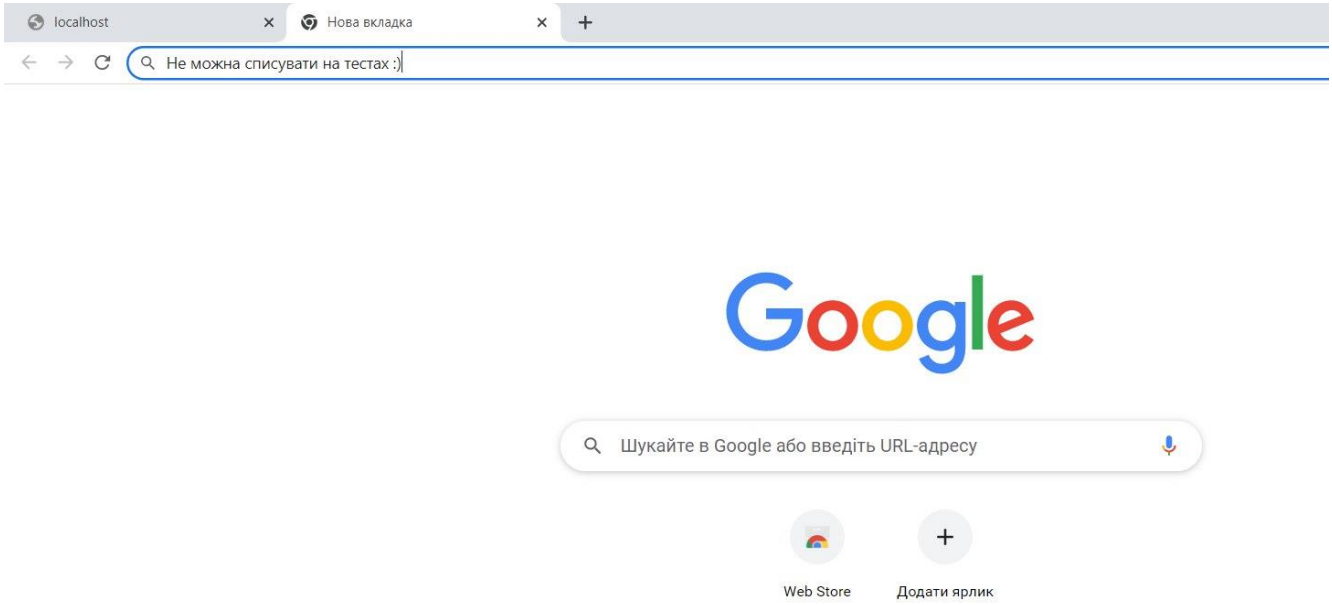


Рисунок 3.24 – Результат копіювання питання

ВИСНОВКИ

Головною задачею кожної людини є здобуття освіти для того, щоб знайти себе в житті і працювати на тій роботі, яка їй подобається. А задачею розробників програмного забезпечення, в тому числі інструментів для навчання, є забезпечення зручних умов для здобувачів освіти, адже, коли навчання викликатиме в людини позитивні емоції, у неї буде бажання навчатись, розвивати себе і приносити користь суспільству. Моя розробка – платформа MarkTest забезпечує зручний інтерфейс як для студента, так і для викладача.

У порівнянні з іншими платформами, MarkTest дозволяє студенту вибрати рівень складності тесту, в відповідності з яким будуть обрані питання, з яких складатиметься тест. Ця можливість мінімізує шанс провалити студенту, який претендує на задовільну оцінку, тест, у який можуть потрапити багато складних питань, які має знати студент, який претендує на кращу оцінку ніж задовільно.

Також значною перевагою MarkTest є те, що для проходження тесту він надсилає лише два запити на сервер (один, щоб згенерувати тест і один, щоб його здати та перевірити), що робить її більш оптимізованою і дає можливість одночасно витримувати більшу кількість тестів, в той час як аналоги надсилають запит на сервер при кожній зміні питань, внаслідок чого працюють повільніше. Також у моєї програми є можливість розсилати запрошення на тест електронною поштою, а це зручніше ніж поширювати посилання через месенджери.

У сучасному світі тестування онлайн є одним з найкращих способів перевірити знання і є перспективним напрямом розвитку в освіті, адже технології невпинно розвиваються і постійно з’являється щось нове. З кожним днем залученість людини зменшується, а сучасні технології забезпечують що раз ефективніше та надійніше проведення тестування.

Ще однією значною перевагою онлайн тестування є те, що студент може побачити свою оцінку відразу, а не чекати поки викладач перевірить його роботу.

Викладачеві ж не треба перевіряти роботу кожного студента, йому достатньо зайти на сторінку і переглянути результати, а це непогана економія часу.

Платформи для тестування також є універсальними, тобто студенти можуть проходити тести в будь який період часу практично в будь якому місці, неважливо чи вони знаходяться на робочому місці чи в закладі харчування чи навіть в транспортному засобі. Щоб пройти тест їм достатньо мати стабільний інтернет і смартфон. Також такий підхід до оцінювання є дуже зручним для студентів, які поєднують навчання з роботою, адже їм не потрібно їхати в навчальний заклад для проходження тесту.

Написання тесту є швидшим ніж написання контрольної роботи з описовими завданнями, а також дає більш точну і справедливішу оцінку, адже на оцінку не будуть впливати такі фактори, як поганий почерк студента або запізнення здачі через повільну манеру письма.

Отже, тестування – це потужний інструмент, який можна використовувати не лише для оцінювання, але й як спосіб навчатися. Воно має переваги як для студентів, так і для викладачів, однак буде дійсно ефективним тоді, коли студенти сприйматимуть його не як покарання, а як корисну навчальну вправу.

Вміле використання дистанційного тестування є неперевершеним здобутком висококваліфікованого викладача. Добре розроблений тест дає критичну інформацію про роботу студентів. Завдяки швидкому зворотному зв’язку тести стають важливими інструментом для студентів, бо забезпечують оцінку їх власної діяльності. Тестування слід зробити звичною і зручною формою регулярного контролю знань студентів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Використання онлайн тестування для організації дистанційного навчання в умовах карантину [Електронний ресурс] – Режим доступу:http://new.makinfo.org.ua/index.php/novyny-viddilennia-zahalnoosvitnia- pidhotovka/569-vikoristannya-onlajn-testuvannya-dlya-organizatsiji- distantsijnogo-navchannya-v-umovakh-karantinu
2. Посібник: знайомство з React [Електронний ресурс] – Режим доступу: https://uk.reactjs.org/tutorial/tutorial.html
3. Тестування в системі дистанційного навчання. Колонтаєвський О.П. [Електронний ресурс] – Режим доступу: https://2018.moodlemoot.in.ua/course/view.php?id=11
4. AutoMapper Overview [Електронний ресурс] – Режим доступу: https://automapper.org/
5. Axios Introduction [Електронний ресурс] – Режим доступу: https://axios-http.com/docs/intro
6. Entity Framework Documentation [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/uk-ua/ef/
7. Overview to ASP.NET Core [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/uk-ua/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0
8. Robin Wieruch. The Road to React: Your journey to master plain yet pragmaticReact.js 2022
9. SmtpClient Class [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/uk-ua/dotnet/api/system.net.mail.smtpclient?view=net-6.0