Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

магістра
(назва освітньо-кваліфікаційного рівня)

студента *Янишівського В'ячеслава В'ячеславовича*
(ПІБ)

академічної групи *121м-22-4*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *«Інженерія програмного забезпечення»*
(назва освітньої програми)

на тему: *Розробка та дослідження ефективності застосування контрольно-діагностичної системи для перевірки знань з математичних дисциплін*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
розділів кваліфікаційної роботи				
спеціальний	*проф. Мороз Б. І.*			
Рецензент				
Нормоконтролер	*проф. Лактіонов І.С.*	74	добре	

Дніпро
2023

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 20 23 Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

студенту *121м-22-4* *Янишівський В'ячеслав В'ячеславович*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Розробка та дослідження ефективності застосування контрольно-діагностичної системи для перевірки знань з математичних дисциплін*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 09.10.2023 р. № 1227-с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – процес розробки та застосування контрольно-діагностичної системи для перевірки знань з математичних дисциплін

Предмет досліджень – система оцінювання та діагностика знань учнів або студентів заснована на математичних завданнях та тестах

Мета НДР – дослідити існуючі методи перевірки знань та розробити контрольно-діагностичну систему для допомоги студентам у процесі перевірки знань викладачами.

Вихідні дані для проведення роботи – теоретичні дослідження та перелік питань і відповідей до математичних задач та тестів.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Новизна запропонованих рішень полягає в представлені в даному дослідженні рішення яке відзначається своєю виразною новизною шляхом обґрунтування та вирішення актуальної проблеми, що полягає в відсутності зручних та

універсальних діагностичних систем, призначених для перевірки знань з математичних дисциплін.

Практична цінність полягає в їх здатності ефективно вплинути на процес створення математичних тестів для викладачів. Ця система відкриває нові можливості та переваги для різних сфер, де перевірка математичних знань має важливе значення.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати досліджень мають бути подані у вигляді, що дозволяє побачити та оцінити безспосереднє використання обраних методів. В результаті роботи повинна бути розроблена контрольно-діагностична система на основі фреймворку Spring Boot, що працює у всіх сучасних браузерах.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз теми та постановка задачі	12.09.2023-30.09.2023
Розробка та дослідження ефективності контрольно-діагностичної системи	01.10.2023-31.10.2023
Використання програми та аналіз отриманих результатів	01.11.2023-12.12.2023

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект включає вигоди від зменшення витрат, підвищення продуктивності, покращення якості та зменшення ризиків у зв'язку з впровадженням системи для контролю та діагностики знань.

Соціальний ефект від реалізації результатів роботи очікується позитивним, завдяки полегшенню процесів контролю перевірки знань студентів.

7 ДОДАТКОВІ ВИМОГИ

Завдання видав _____ *Мороз Б. І.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Янишівський В. В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *12.09.2023 р.*

Термін подання кваліфікаційної роботи до ЕК *1 .12.2023*

РЕФЕРАТ

Пояснювальна записка: 96 сторінок, 35 рисунків, 5 таблиць, 2 додатки, 20 джерел.

Об’єкт дослідження: модуль для перевірки знань з математичних дисциплін.

Мета магістерської роботи: дослідити існуючі методи перевірки знань та розробити контрольно-діагностичну систему для допомоги студентам у процесі перевірки знань викладачами.

Методи дослідження: дослідити існуючі методи перевірки знань та розробити контрольно-діагностичну систему для допомоги студентам у процесі перевірки знань викладачами.

Наукова новизна полягає представлені в даному дослідженні рішення яке відзначається своєю виразною новизною шляхом обґрунтування та вирішення актуальної проблеми, що полягає в відсутності зручних та універсальних діагностичних систем, призначених для перевірки знань з математичних дисциплін.

Практична цінність полягає в їх здатності ефективно вплинути на процес створення математичних тестів для викладачів. Ця система відкриває нові можливості та переваги для різних сфер, де перевірка математичних знань має важливе значення.

Область застосування контрольно-діагностичної системи для перевірки знань з математичних дисциплін включає використання в навчальних закладах (школи, університети) для регулярного оцінювання та моніторингу рівня математичних знань учнів та студентів, надає можливості для самостійного вивчення матеріалу та самоперевірки знань, що сприяє активному навчанню, а також викладачам та студентам готуватися до іспитів:

Значення роботи та висновки: полягає в створенні ефективного, індивідуалізованого та мотиваційного середовища для вивчення математики.

Прогнози щодо розвитку досліджень: організація чат-спілкування між авторизованими користувачами.

Список ключових слів: контрольно-діагностична система, відображення, сторінка, база даних, Spring Boot, IntelliJ IDEA.

ABSTRACT

Explanatory note: 96 pages, 35 figures, 5 tables, 2 appendices, 20 sources.

Object of research: a module for testing knowledge of mathematical disciplines.

The purpose of the master's work: to investigate the existing methods of knowledge verification and to develop a control and diagnostic system to help students in the process of knowledge verification by teachers.

Research methods: to investigate existing knowledge testing methods and to develop a control-diagnostic system to help students in the process of knowledge testing by teachers.

The scientific novelty is the solution presented in this study, which is distinguished by its expressive novelty by substantiating and solving the actual problem, which consists in the lack of convenient and universal diagnostic systems designed to test knowledge of mathematical disciplines.

The practical value lies in their ability to effectively influence the process of creating mathematics tests for teachers. This system opens up new opportunities and advantages for various fields where the verification of mathematical knowledge is important.

The field of application of the control-diagnostic system for testing knowledge of mathematical disciplines includes use in educational institutions (schools, universities) for regular assessment and monitoring of the level of mathematical knowledge of pupils and students, provides opportunities for independent study of the material and self-testing of knowledge, which promotes active learning, and also for teachers and students to prepare for exams:

The value of the work and conclusions: is to create an effective, individualized and motivational environment for learning mathematics.

Forecasts for the development of research: the organization of chat communication between authorized users.

Keywords: monitoring system, display, page, database, Spring Boot, IntelliJ IDEA.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

MVC – Model-View-Controller

ОС – операційна система

ЗМІСТ

ВСТУП

В даний час освіта є важливою сферою в діяльності людини, одним з тих соціальних інститутів, значимість яких постійно зростає в міру розвитку суспільства в бік інформаційно-технологічного та соціально-економічного прогресу.

Тестування як форма контролю навчальних досягнень студентів має своїх прихильників і тих, хто її активно критикує, але, незважаючи на це, в останні роки вона все частіше стала застосовуватися в системі професійної освіти як технологія, яка дозволяє об'єктивно і швидко оцінити рівень досягнень кожного студента.

Найважливішою перевагою електронного тестування є можливість моделювання тестових завдань (їх послідовності, варіативності і навіть самих умов) на основі заданого алгоритму.

До інших переваг слід віднести оперативність при підведенні підсумків і їх опублікування, неупередженість оцінок, меншу трудомісткість при редакції тестів, простоту і економічність їх тиражування, можливість здійснення самоконтролю, дистанційна взаємодія з студентами які навчаються з урахуванням індивідуального вибору часу і місця.

Застосування інтернет-технологій в електронному тестуванні не просто розширює географічні рамки, але, перш за все, являє додатковий інструментарій оцінки взаємодії з які навчаються. Електронне тестування дозволяє автоматично узагальнити ряд характеристик доступу і локального робочого місця студента (включаючи як характеристики апаратного і програмного забезпечення, використовуваного для підключення до Інтернет, так і допоміжних ресурсів).

Завдяки існуючим значним можливостям та своїй зручності для розробки контрольно-діагностичної системи було обрано універсальне середовище IntelliJ IDEA 2019.1.1 (Community Edition). Платформою для розробки було обрано Spring MVC (Model View Control).

РОЗДІЛ 1
АНАЛІЗ ТЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Сутність контрольно-діагностичної системи

Контрольно-діагностична система для перевірки знань – це комплекс програмних інструментів та методів, спрямованих на об'єктивну оцінку та аналіз рівня засвоєння матеріалу студентами чи учнями. Вона служить інструментом для постійного моніторингу прогресу та виявлення можливих труднощів у засвоєнні матеріалу.

Контрольно-діагностична система може працювати як інструмент самооцінки, так і засіб для оцінки зовнішніми організаціями. Контрольно-діагностична система важлива для створення ефективного та індивідуалізованого середовища навчання, сприяючи досягненню найкращих результатів у навчанні. Контрольно-діагностична система також включає в себе інструменти для аналізу та зіставлення даних з іншими учнями чи групами. Це дозволяє вчителям та адміністраторам здійснювати загальний моніторинг ефективності навчання та розробляти стратегії для покращення якості освіти.

Контрольно-діагностична система включає в себе методи тестування, аналізу відповідей, створення зворотного зв'язку та можливості адаптації до потреб кожного користувача.

1.2 Мета розробки та область використання

Контрольно-діагностична системи для перевірки знань з математичних дисциплін призначена для викладачів і студентів будь-яких навчальних закладів. Основна мета створення цієї системи — полегшити роботу викладача та надати студентам можливість самостійно перевірити свої знання.

Кожний викладач за допомогою такої системи може самостійно створювати тести, а студент може самостійно перевірити свої знання на цих тестах, які відкриті для усіх студентів.

Завдяки контрольно-діагностичної системі викладачу не треба перевіряти чи виконані тести, оскільки це зробить сама система одразу після завершення тестування. Також викладач на моніторі свого комп'ютера може побачити результати тестування і проконтролювати, скільки завдань виконав кожний студент, на які питання відповів, чи правильно відповів і скільки балів за це отримав.

Галузь застосування цієї системи широка і може включати наступні сфери:

– Алгебра та Геометрія: оцінка розуміння алгебраїчних концепцій, геометричних формул та взаємодії між ними;

– Математичний аналіз**:** вимірювання здатності застосовувати аналітичні методи до розв'язання математичних проблем та обчислень;

– Теорія чисел та комбінаторика**:** перевірка розуміння властивостей чисел, комбінаторних методів і ймовірнісних понять;

– Статистика та ймовірність**:** оцінка здатності використовувати статистичні методи та розуміння ймовірнісних розподілів;

– Комп'ютерне програмування та використання математичного програмного забезпечення**:** оцінка навичок використання комп'ютерних інструментів для вирішення математичних завдань;

– Дискретна математика: оцінка розуміння теорії графів, комбінаторних структур та логічних операцій;

– Математичну логіку та теорію множин: перевірка здатності розв'язувати завдання, пов'язані з логікою та властивостями множин;

– Математичні концепції в інших науках**:** перевірка здатності застосовувати математичні концепції в інших наукових галузях.

1.3 Постановка задачі

Темою роботи є «Розробка та дослідження ефективності застосування контрольно-діагностичної системи для перевірки знань з математичних дисцплін».

В межах даного магістерського проєкту потрібно організувати базу даних для збереження інформації про тести з математичних дисциплін, розробити контрольно-діагностичну систему, яка дозволяє працювати з розробленою базою даних.

Контрольно-діагностична система повинна бути кросбраузерна та підтримувати масштабування.

Програма повинна виконувати такі функції по веденню бази даних:

- додавання, редагування та видалення тестів;
- додавання, редагування та видалення запитань;
- додавання, редагування та видалення відповідей;
- додавання, редагування та видалення результатів;
- можливість завантаження тестів на персональний комп’ютер;
- пошук необхідного тесту за назвою або описом;
- підтримка завантаження зображень до питань тестів;

Контрольно-діагностична система повинна перевіряти усі введені дані на коректність та видавати повідомлення у випадку помилкового вводу.

Завдання магістерського проєкту повинно бути реалізоване з використанням таких ролей користувачів: «Адміністратор» та «Студент».

Для ролі користувача «Адміністратор» повинні бути реалізовані функції створення, редагування та видалення тестових завдань для студентів.

Для ролі користувача «Студент» повинні бути реалізовані функції для надання перегляду та проходження тестів.

Для легшого опрацювання інформації щодо матеріалу, посилання на які надає контрольно-діагностична система, повинна бути реалізована функція пошуку тестів за його назвою, або описом.

1.4 Ефективність застосування контрольно-діагностичної системи

Застосування контрольно-діагностичної системи для перевірки знань з математичних дисциплін може виявитися ефективним з точки зору об'єктивності, швидкості та індивідуалізації. Вона дозволяє автоматизувати оцінювання, надаючи об'єктивні результати та забезпечуючи можливість адаптуватися до потреб кожного студента. Такий підхід дозволяє ефективно визначати рівень розуміння матеріалу, виявляти і усувати помилки та надавати студентам індивідуалізовані рекомендації для подальшого вивчення.

При цьому системи контролю можуть виявити сильні та слабкі сторони кожного студента, дозволяючи оптимізувати процес вивчення. Автоматизовані засоби також забезпечують швидкість оцінювання та зворотний зв'язок, що сприяє більш ефективному використанню часу та ресурсів. В результаті контрольно-діагностичні системи можуть значно покращити якість навчання та створити сприятливі умови для розвитку математичних навичок студентів.

Крім того, контрольно-діагностична система може бути використана для створення персоналізованих навчальних планів, враховуючи індивідуальні потреби та темпи навчання студентів. Це сприяє більш ефективній адаптації до різних рівнів здібностей та стилів вивчення.

Застосування контрольно-діагностичної системи може стимулювати студентів до більш активного участі у власному навчанні, оскільки вони отримують миттєвий зворотний зв'язок та можливість самостійно вдосконалювати свої навички.

Узагальнюючи, застосування контрольно-діагностичної системи в математичних дисциплінах може сприяти підвищенню якості навчання,

стимулюванню саморозвитку студентів та створенню більш гнучкого та індивідуалізованого середовища навчання.

Контрольно-діагностичні системи можуть слугувати не лише інструментами оцінювання, а й засобами активної педагогіки. Їх можна використовувати для впровадження інтерактивних завдань, групової роботи та інших форм співпраці, що сприяє залученню студентів та розвитку їхніх творчих та критичних мисленнєвих навичок.

За допомогою контрольно-діагностичних систем можна стежити за динамікою зростання знань студентів, що дозволяє адаптувати навчальні плани та методи навчання на ходу. Це особливо важливо в умовах швидкозмінюваних вимог ринку праці та наукового прогресу.

У підсумку, контрольно-діагностичні системи є потужним інструментом для оцінювання, підтримки та розвитку знань студентів в математичних дисциплінах, що сприяє покращенню якості навчання та формуванню висококваліфікованих фахівців.

Контрольно-діагностичні системи можуть також допомагати виявляти студентів, які можуть виявити інтерес до конкретних аспектів математики чи відгалуження в глибше вивчення певних тем. Це може сприяти ранньому виявленню та розвитку обдарованих учнів.

У кінцевому рахунку, контрольно-діагностичні системи є не лише інструментом для перевірки знань, а й інтегральною частиною сучасного навчального процесу, спрямованого на максимізацію навчального потенціалу кожного студента та підготовку його до викликів сучасного світу.

Додатково, контрольно-діагностичні системи можуть створювати можливості для розвитку критичного мислення та проблемного підходу. Вони можуть пропонувати завдання, спрямовані на розвиток аналітичних та творчих навичок, а також вміння застосовувати математичні знання до розв'язання реальних проблем.

Застосування технологій та інтерактивних методів у контрольно-діагностичних системах може робити навчання математики більш привабливим та зрозумілим для студентів, залучаючи їх до процесу навчання.

Усе це допомагає створювати навчальне середовище, сприятливе для розвитку всебічно освічених особистостей, здатних не лише засвоювати математичні знання, а й застосовувати їх у різних сферах життя.

1.5 Проблематика відсутності контрольно-діагностичних систем

Відсутність контрольно-діагностичних систем з математичних дисциплін може породжувати декілька проблем, які впливають на різні рівні освіти та навчання:

– Неефективне оцінювання: відсутність систематичного та об’єктивного методу оцінювання знань студентів у математичних дисциплінах може призводити до суб’єктивності в оцінок та несправедливості в оцінюванні;

– Важкість виявлення слабких місць: систематичного контролю важко виявити та вирішити проблемні питання та слабкі місця у знаннях студенів;

– Забруднення навчання: відсутність чіткого механізму оцінювання може сприяти "вивченню тесту" замість глибокого розуміння та застосування математичних концепцій;

– Недостатній розвиток аналітичних та проблемних навичок: може бути відсутнітий стимул для розвитку аналітичних та проблемних навичок, які є важливими в математичній освіті;

– Невідомість прогресу: вчителям може бути важко відстежувати прогрес кожного учня та реагувати на його потреби без відповідного інструментарію;

– Втрата можливостей для інновацій: відсутність цифрових технологій та інноваційних методів оцінювання може обмежувати можливості вдосконалення навчального процесу;

– Недостатній фокус на ключових концепціях: може виникнути ситуація, коли навчальні програми не належним чином фокусуються на ключових математичних концепціях.

Розробка та впровадження контрольно-діагностичних систем у математичну освіту може сприяти вирішенню цих проблем і забезпечити більш ефективний та індивідуалізований підхід до навчання математики.

РОЗДІЛ 2
ТЕХНОЛОГІЇ ЯКІ БУЛИ ЗАДІЯНІ В РОЗРОБЦІ ПРОГРАМИ

2.1 IntelliJ IDEA

IntelliJ IDEA є інтегрованим середовищем розробки, розробленоюкомпанією JetBrains, і визначається своєю високою продуктивністю та широким набором функціоналу. Це середовище призначено для розробки програмного забезпечення на мовах програмування, таких як Java, Kotlin, Groovy, Scala, та інші [2].

IntelliJ IDEA відзначається високою ефективністю та інтелектуальним підходом до розробки, надаючи розробникам широкий спектр інструментів для рефакторингу, аналізу коду, автоматичної підказки та вирішення різноманітних завдань. Вона підтримує розробку на різних мовах програмування, що розширює її універсальність для розробників.

Однією з ключових особливостей IntelliJ IDEA є вбудована підтримка різноманітних фреймворків та технологій, що полегшує розробку програм під конкретні потреби. Вона також надає інтеграцію з системами контролю версій, що сприяє спільній роботі команд розробників [3].

IntelliJ IDEA підтримує розробку мобільних додатків для Android і iOS, надаючи інструменти для створення високоякісних мобільних застосунків. Крім того, вона має зручний інтерфейс та можливості розширення, що роблять її популярним вибором серед розробників програмного забезпечення.

IntelliJ IDEA активно використовується в індустрії програмування, включаючи розробку веб-додатків, мобільних додатків, корпоративних систем та інших типів програм. Вона відома своєю здатністю прискорювати процес розробки завдяки інтегрованим інструментам, таким як система автоматичного виявлення помилок, автоматичний аналіз коду та високий рівень підтримки рефакторингу [4].

Окрім того, IntelliJ IDEA підтримує роботу у спільноті розробників і надає можливості розширення за допомогою плагінів, що дозволяє користувачам розширювати функціоналість IDE відповідно до своїх потреб. IntelliJ IDEA продовжує активно розвиватися і оновлюватися, впроваджуючи нові функції та виправляючи помилки. Крім того, вона забезпечує інтеграцію з сучасними інструментами розробки, такими як системи управління версіями, системи збірки, а також розподілені та хмарні сервіси.

Багато розробників обирають IntelliJ IDEA завдяки її спрощеному використанню, високій продуктивності та вдалому поєднанню інтелектуальних інструментів розробки. Вона також підтримує стандарти індустрії, що робить її зручною для інтеграції у різноманітні проєкти.

Ідея створення IntelliJ IDEA полягала в тому, щоб полегшити та покращити роботу розробників, надаючи їм потужний інструмент для швидкої та ефективної розробки програмного забезпечення. І це завдання вона успішно виконує, продовжуючи бути однією з найбільш популярних ідей для розробки на мовах Java та інших мовах програмування.

Загалом, IntelliJ IDEA є інструментом вибору для багатьох розробників, які цінують продуктивність, зручний інтерфейс та багатий функціонал для створення високоякісного програмного забезпечення.

2.2 MySQL Server

MySQL Server – це реляційна система управління базами даних, яка використовує мову структурованого запитання SQL (Structured Query Language) для взаємодії з базами даних. MySQL є відкритим програмним забезпеченням і доступний для розгортання на різноманітних операційних системах, таких як Windows, Linux та macOS [5].

Основні характеристики MySQL Server:

– Мова запитань SQL: MySQL використовує SQL для взаємодії з базами даних. SQL дозволяє виконувати операції, такі як вибірка, вставка, оновлення та видалення даних [6].

– Сервер-клієнтська модель: MySQL працює на основі моделі клієнт-сервер, де клієнти (наприклад, програми або веб-сайти) звертаються до сервера для доступу до даних [7].

– Масштабованість: MySQL може бути використаний для невеликих проєктів, а також для великих підприємств з великим обсягом даних, завдяки своїй масштабованості та продуктивності.

– Підтримка різноманітних типів даних: MySQL підтримує різноманітні типи даних, включаючи числа, рядки, дати, часи, зображення та інші.

2.3 Мова Java

Java – це високорівнева, об'єктно-орієнтована мова програмування. Вона пропонує простий інтерфейс для розробки різноманітних програм, надаючи зручність інструментів ООП. Java відома своєю переносимістю, завдяки віртуальній машині Java (JVM), яка дозволяє виконувати код на різних платформах [8].

Мова підтримує багатопоточність, що полегшує роботу з паралельним виконанням завдань. Велика стандартна бібліотека спрощує розв'язання різних задач, а система безпеки дозволяє контролювати доступ до ресурсів. Java знаходить застосування в розробці веб-додатків, мобільних додатків та корпоративних систем.

Java володіє простим синтаксисом, що полегшує розуміння та написання коду. Вона використовує систему управління пам'яттю, яка автоматично вивільняє ресурси, що сприяє попередженню багатьох типів помилок. Мова підтримує розробку мережевих додатків через розширені засоби роботи з мережами.

Java визначається своєю крос-платформенністю, що робить її ідеальним вибором для розробників, які прагнуть створювати програми, що працюють на різних операційних системах. Ця особливість дозволяє використовувати той же код на Windows, Linux або macOS, без необхідності його перекомпіляції.

2.4 Spring Boot

Spring Boot – це фреймворк для розробки додатків на мові програмування Java, який ґрунтується на ідей платформи Spring Framework. Однією з ключових особливостей Spring Boot є його підхід "конвенція перед конфігурацією", що дозволяє розробникам швидко створювати додатки з меншою кількістю налаштувань [9].

Цей фреймворк полегшує розгортання додатків, надаючи вбудовані сервери (такі як Tomcat, Jetty або Undertow) і стандартні конфігурації за замовчуванням. Він також вбудовує ряд важливих функцій, таких як автоматична конфігурація, управління залежностями, інтеграція з системами контролю версій, що спрощує розробку.

Spring Boot надає підтримку для вбудованих баз даних, що спрощує тестування і розробку, і спрямований на сприяння принципам мікросервісної архітектури. Він також інтегрується з іншими елементами екосистеми Spring, надаючи розробникам інструменти для створення високоякісних додатків на платформі Java.Spring Boot. Spring Boot вирізняється своєю гнучкістю та зручністю у використанні, надаючи розробникам можливість швидко втілювати ідеї та функціональність в коді. Він покликаний зменшити зайвий оверхед і спростити рутинні завдання, такі як конфігурація, щоб розробники могли більше уваги приділяти самому коду та бізнес-логіці [10].

Фреймворк активно підтримує розробку мікросервісів, створюючи умови для ефективного взаємодії окремих компонентів системи. Це особливо важливо в архітектурі розподілених систем.

Spring Boot також враховує принцип "програмуй згідно з домовленостями" (Convention over Configuration), спрощуючи роботу розробників та прискорюючи розробку. Це дозволяє стандартизувати підходи та забезпечити швидке втілення проєктів.

Узагальнюючи, Spring Boot є потужним інструментом для розробників, які бажають створювати додатки на Java з високою продуктивністю та ефективністю, особливо в умовах розподіленого середовища.

2.5 Model-View-Controller

MVC (Model View Controller) передбачає поділ додатка на три компоненти: контролер, відображення та модель.

Контролер (controller) представляє клас, що забезпечує зв'язок між користувачем і системою, відображенням і сховищем даних. Він отримує дані, що вводяться користувачем та обробляє їх, і в залежності від результатів обробки відправляє користувачеві певний висновок, наприклад, у вигляді відображення.

Відображення (view) — це власне візуальна частина або призначений для користувача інтерфейс програми. Як правило, HTML-сторінка, яку користувач бачить, зайшовши на сайт [11].

Модель (model) представляє клас, що описує логіку використовуваних даних. Модель — це частина програми, що відповідає за базову прикладну логіку, яку також називають бізнес-логікою. Об'єкти моделі зазвичай отримують дані з постійного сховища, наприклад MySQL Server, і виконують над ними бізнес-логіку. У кожної програми своя модель, тому платформа Spring MVC не обмежує створювані моделі [12].

Загальну схему взаємодії цих компонентів представлено на рисунку 2.1.

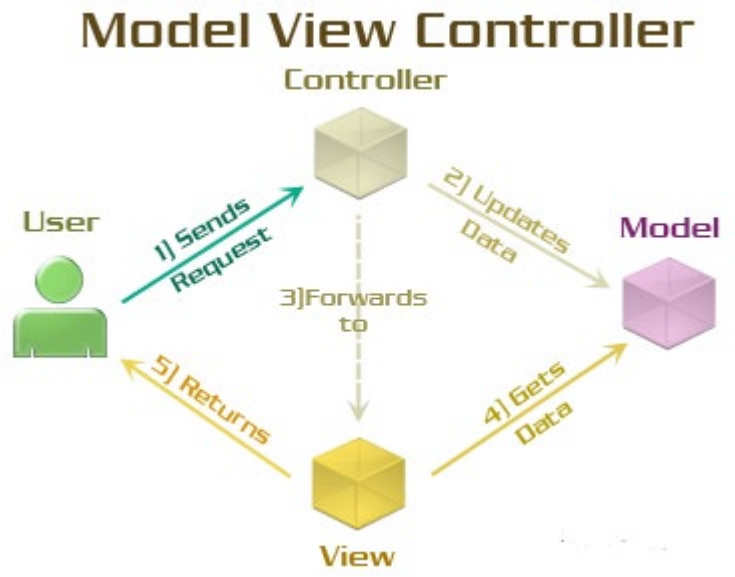

Рис. 2.1. Схема взаємодії компонентів Spring MVC

У цій схемі модель є незалежним компонентом — будь-які зміни контролера або відображення не зачіпають модель. Контролер і відображення є відносно незалежними компонентами, і нерідко їх можна змінювати незалежно один від одного [13].

Завдяки цьому реалізується концепція «поділ відповідальності», в зв'язку з чим легше будувати роботу над окремими компонентами. Крім того, внаслідок цього додаток зручніше тестувати.

Конкретні реалізації та визначення паттерну «поділ відповідальності» можуть відрізнятися, але в силу своєї гнучкості і простоти він став дуже популярним останнім часом.

2.6 Висновки

Застосування технологій IntelliJ IDEA, Java, MySQL, Spring Boot та патерну MVC у розробці програмного забезпечення виявляється дуже ефективним та потужним підходом. IntelliJ IDEA, як інтегроване середовище розробки, забезпечує зручність та продуктивність у процесі написання коду, тестування та налагодження додатків.

Мова програмування Java надає високий рівень абстракції та об'єднує в собі елементи ефективності та зручності у синтаксисі. Це дозволяє розробникам швидко писати якісний код, спрощує його розуміння та утримання. Крім того, Java підтримує об'єктно-орієнтований підхід, що сприяє структуруванню програм та полегшує їхнє розширення.

Застосування мови Java в поєднанні з іншими технологіями стало ключовим фактором у створенні високоякісного та функціонального програмного забезпечення. Ця комбінація не лише сприяє продуктивності та легкості розробки, а й відкриває додаткові можливості для розширення та покращення продукту в майбутньому.

MySQL, як легка та компактна система керування базами даних, є ідеальним вибором для проєктів різного роду, зокрема для мобільних додатків та проєктів з обмеженими ресурсами. Використання Spring Boot дозволяє створювати сучасні та естетично привабливі інтерфейси користувача для операційної системи Windows.

Патерн MVC в контексті розробки з IntelliJ IDEA, MySQL та Spring Boot дозволяє ефективно розділити бізнес-логіку, представлення та логіку взаємодії з користувачем, що полегшує тестування, управління кодом та робить програмний код більш підтримуваним.

Загальний висновок полягає в тому, що комбінація IntelliJ IDEA, MySQL, Spring Boot та MVC забезпечує потужний та гнучкий інструментарій для створення високоякісних, продуктивних та легко підтримуваних додатків для операційної системи Windows.

РОЗДІЛ 3
СТРУКТУРА ПРОГРАМИ, РЕАЛІЗАЦЯ ТА ПРИКЛАД ВИКОРИСТАННЯ

3.1 Опис проєкування

При проєктуванні була розроблена діаграма сценаріїв сайту, яка представлена на рисунку 3.1.

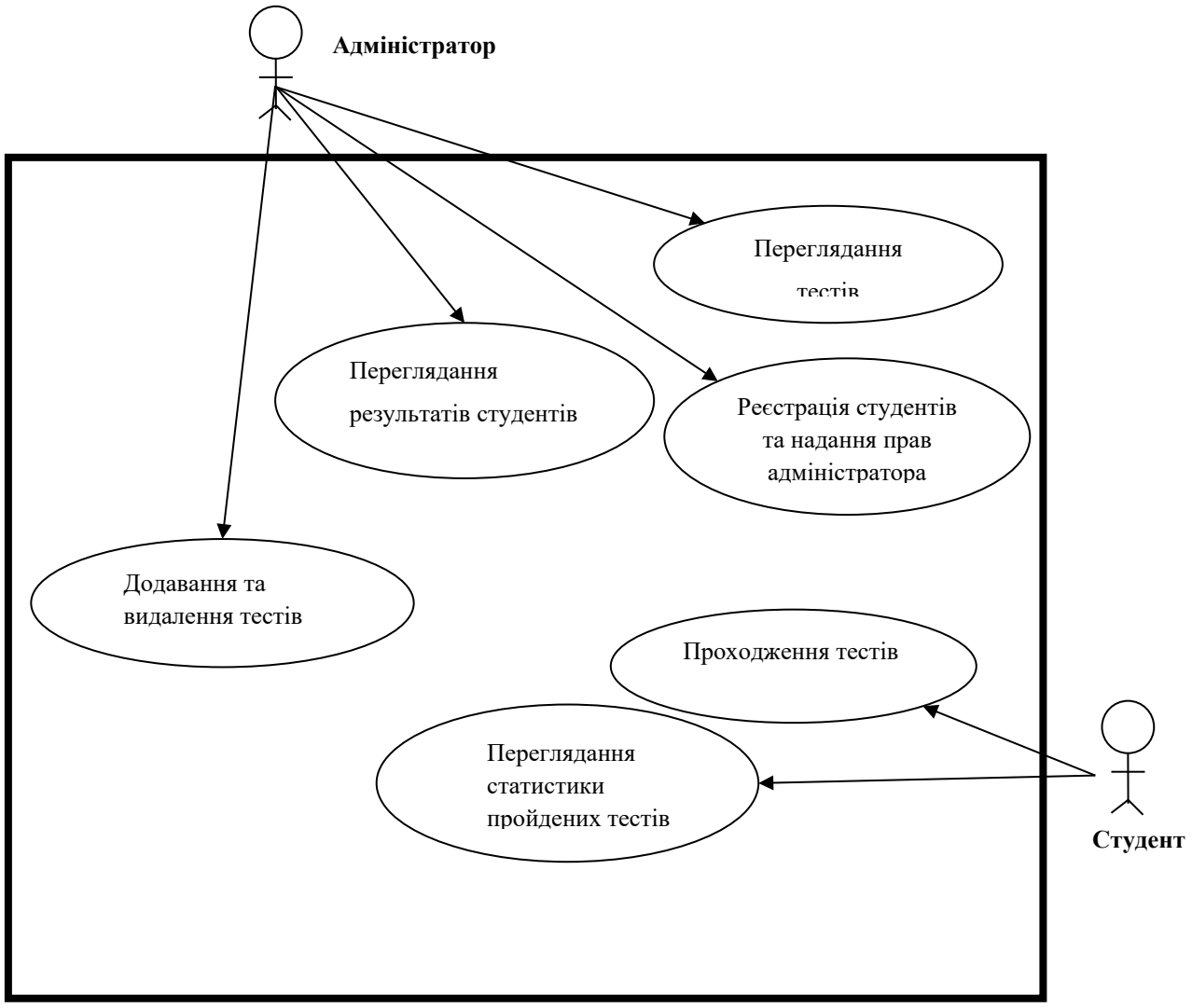

Рис. 3.1. Діаграма сценаріїв проєкту

Сценарій «Додавання та видалення тестів» передбачає:

− можливість адміністратором створювати тести, з питань які мають тип: одна правильна відповідь, декілька правильних відповідей, введення відповіді з клавіатури або встановлення відповідності;

− можливість налаштовувати тести: встановлювати ліміт часу, дозволяти проходити тести лише студентам з певної групи, кількість спроб для тесту;

− видаляти тести за назвою тесту.

Сценарій «Перегляду результатів студентів» передбачає: можливість адміністратором встановлювати фільтри по студентам: по групам та по кількості набраних балів.

Сценарій «Перегляду тестів» передбачає введення адміністратором параметрів пошуку: за назвою, за описом, за датою додавання.

Сценарій «Реєстрації студентів та надання викладачам прав адміністратора» передбачає:

− реєстрація студентів з вказанням: логіну, пароля, ПІБ студента, групи студента;

− додавання користувачам прав адміністратора, якщо це необхідно.

Сценарій «Проходження тестів» передбачає пошук тестів за датою та назвою. Можливість проходження тесту та можливість перездати тест, якщо не вдалося здати тест раніше.

Сценарій «Проходження тестів» передбачає пошук тестів за датою та назвою. Можливість проходження тесту та можливість перездати тест, якщо не вдалося здати тест раніше.

Сценарій «Перегляду статистики пройдених тестів» передбачає перегляд результатів минулих тестів, а також дізнатися такі показники: середній бал за всі тести, кількість невдало зданих тестів, середній час проходження тесту.

3.2 Схема і опис бази даних

Реалізація бази даних проєкту виконана засобами системи керування реляційними базами даних MySQL. База даних проєкту складається з 5 таблиць. На рисунку 3.2 приведена ER-діаграма веб-системи тестування.

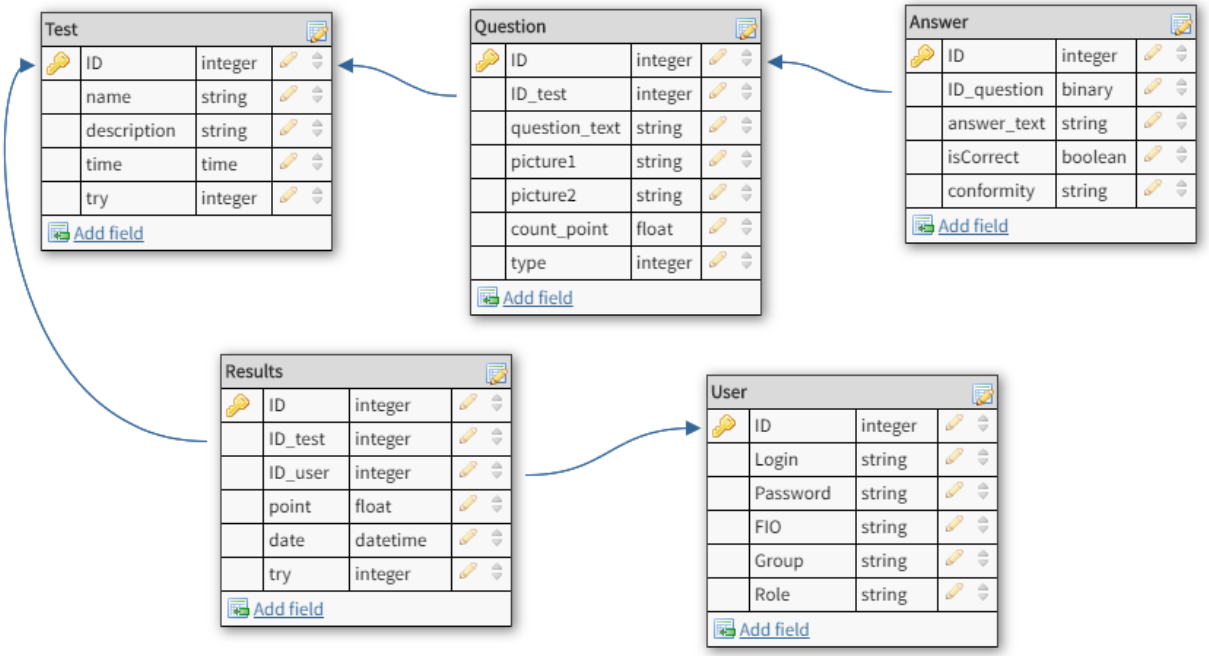

Рис 3.2. ER-діаграма контрольно-діагностичної системи

В таблиці Users, яка представлена в таблиці 3.1, знаходиться інформація про зареєстрованих користувачів.

Таблиця 3.1. Users

Назва поля	Опис
id	Код користувача, унікальний
login	Логін користувача
password	Пароль користувача
Fio	ПІБ користувача
Group	Група користувача
Role	Рівень доступу (Admin, User)

В таблиці Test, яка представлена в таблиці 3.2, знаходиться інформація про створені тести.

Таблиця 3.2. Test

Назва поля	Опис
id	Код тесту, унікальний
name	Тема тесту
description	Опис тесту
time	Ліміт часу на тест
try	Кількість спроб проходження тесту

В таблиці Result, яка представлена в таблиці 3.3, знаходиться статистика проходжень тестів студентами.

Таблиця 3.3. Result

Назва поля	Опис
id	Код результату, унікальний
user_id	Код користувача-студента
test_id	Код тесту
point	Отриманий бал
date	Дата проходження тесту
try	№ спроби здачі тесту

В таблиці Question, яка представлена в таблиці 3.4, знаходиться запитання до тестів.

Таблиця 3.4. Таблиця Question

Назва поля	Опис
id	Код запитання, унікальний
id_test	Код тесту
question_text	Запитання
picture1	Зображення до запитання
picture2	Зображення до запитання
count_point	Кількість балів за запитання
type	Тип тесту

В таблиці Answer, яка представлена в таблиці 3.5, знаходиться відповіді до запитань з тестів.

Таблиця 3.5. Таблиця Answer

Назва поля	Опис
id	Код відповідей, унікальний
id_question	Код запитання
answer_text	Значення
is_correct	Чи правильна це відповідь
conformity	Відповідність

3.3 Контролери програмної системи

Контролери у фреймворку MVC відповідають за обробку дій користувача, виклик методів моделей та передачу інформації користувачеві програмної системи.

У програмній системи реалізовано 7 контролерів:

– main — контролер, який відповідає за головну сторінку, сторінки перегляду тестів;

– user — контролер, який відповідає за сторінку авторизації;

– test — контролер, який відповідає за сторінку тесту;

– createTest — контролер, який відповідає за сторінку створення та редагування тесту;

– registration — контролер, який відповідає за сторінку реєстрації студента;

– result — контролер, який відповідає за сторінки перегляду результатів тесту.

3.4 Проєктування інтерфейсу та опис складових частин програми

За допомогою інтерфейсу, користувач може з легкістю взаємодіяти з програмним кодом. На кожній сторінці є різна кількість компонентів, які полегшують використання програмного додатка.

Інтерфейс програми складається з певної кількості сторінок:

– home — домашня сторінка;

– main — сторінка для перегляду всіх тестів з таблиць бази даних;

– create — сторінка додавання нових тестів у базу даних;

– test/id — сторінка проходження тестів;

– result — сторінка перегляду результатів тестів;

– user — сторінка перегляду зареєстрованих користувачів;

– registration — сторінка реєстрації користувачів;

– login — сторінка аунтефікації.

При переході на сайт користувач потрапляє на домашню сторінку, яка зображена на рисунку 3.3.

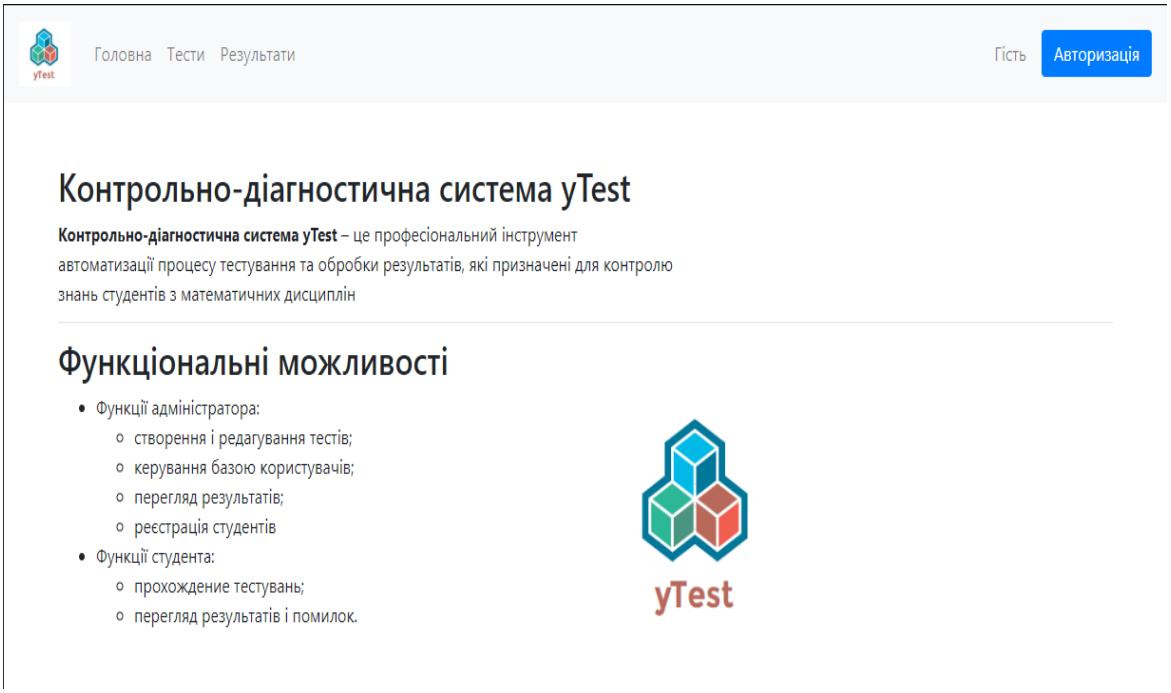

Рис. 3.3. Сторінка «home»

Сторінка «login» призначена для здійснення входу до облікового запису. На ній знаходяться наступні компоненти:

– navbar — навігаційна панель сторінки, в якій знаходяться посилання на інші сторінки сайту, логін аутентифікованого користувача та кнопка вихід, для завершення роботи;

– input-text1 — поле, для введення логіну користувача;

– input-text2 — поле, для введення пароля користувача;

– button — кнопка для здійснення аунтефікації.

Зовнішній вигляд сторінки подано на рисунку 3.4.

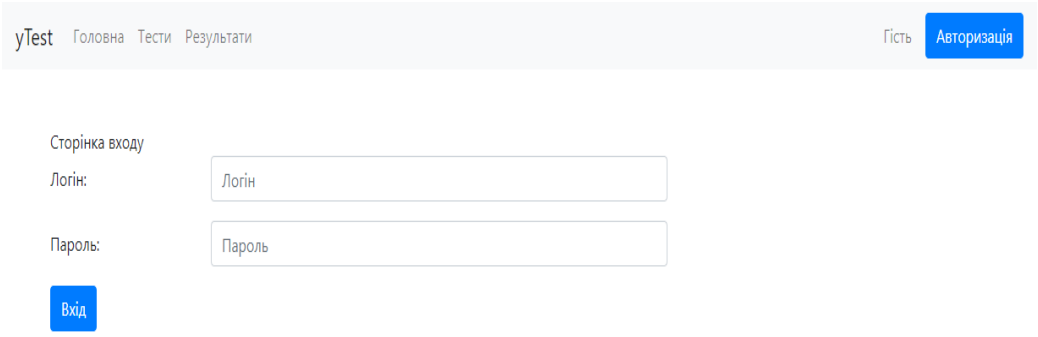

Рис. 3.4. Сторінка «login»

Сторінка «test/id» призначена для проходження тестів. На ній знаходяться наступні компоненти:

– navbar — навігаційна панель сторінки, в якій знаходяться посилання на інші сторінки сайту, логін аутентифікованого користувача та кнопка вихід, для завершення роботи;

– area — блок, який містить питання та елементи вибору відповіді;

– filter — поле для введення рядка по якому буде здійснюватися пошук;

– submit — кнопка для оновлення сторінки і здійснює пошук;

– pagination — для відображення поточної сторінки та переходу між сторінками тестів;

– pagination2 — для відображення поточної кількості тестів на одній сторінці;

– card — карточка для відображення всієї інформації про тест;

– card-header — поле, для відображення назви тесту;

– card-name1 — поле, для відображення опису тесту;

– card-name2 — поле, для відображення часу проходження тесту;

– card-name3 — поле, для відображення кількості спроб на прохождення тесту;

– card-name4 — поле, для відображення кількості питань в тесті;

– card-name5 — поле, для відображення кількості спроб, які залишилися;

– button2 — поле, для переходу на сторінку проходження тесту;

Зовнішній вигляд сторінки подано на рисунку 3.5.

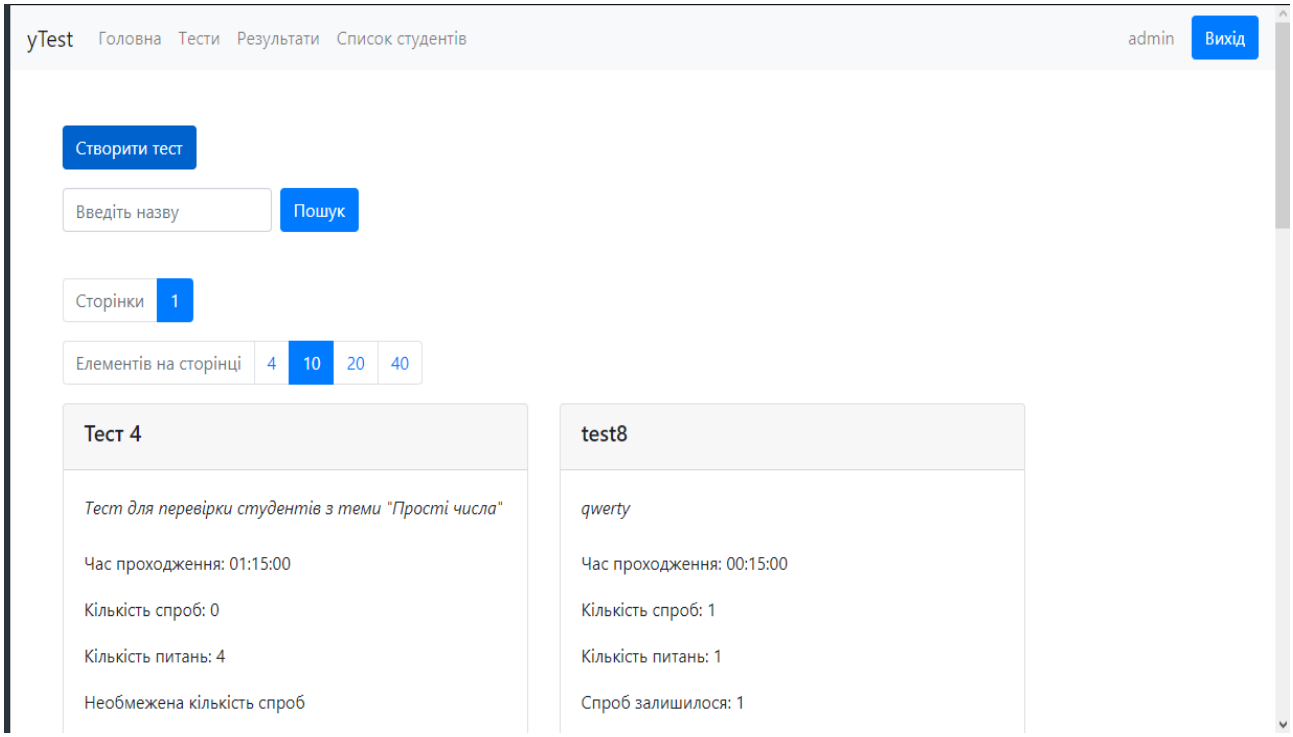

Рис. 3.5. Сторінка «main»

Сторінка «test/id» призначена для проходження тестів. На ній знаходяться наступні компоненти:

– navbar — навігаційна панель сторінки, в якій знаходяться посилання на інші сторінки сайту, логін аутентифікованого користувача та кнопка вихід, для завершення роботи;

– area — блок, в якому знаходятся питання та елементи вибору відповіді до питання

– submit — кнопка, яка відповідає за збереження результату в базі даних та виведення на екран користувача правильних та неправильних відповідей з оцінкою, яку отримав користувач

Зовнішній вигляд сторінки подано на рисунку 3.6.

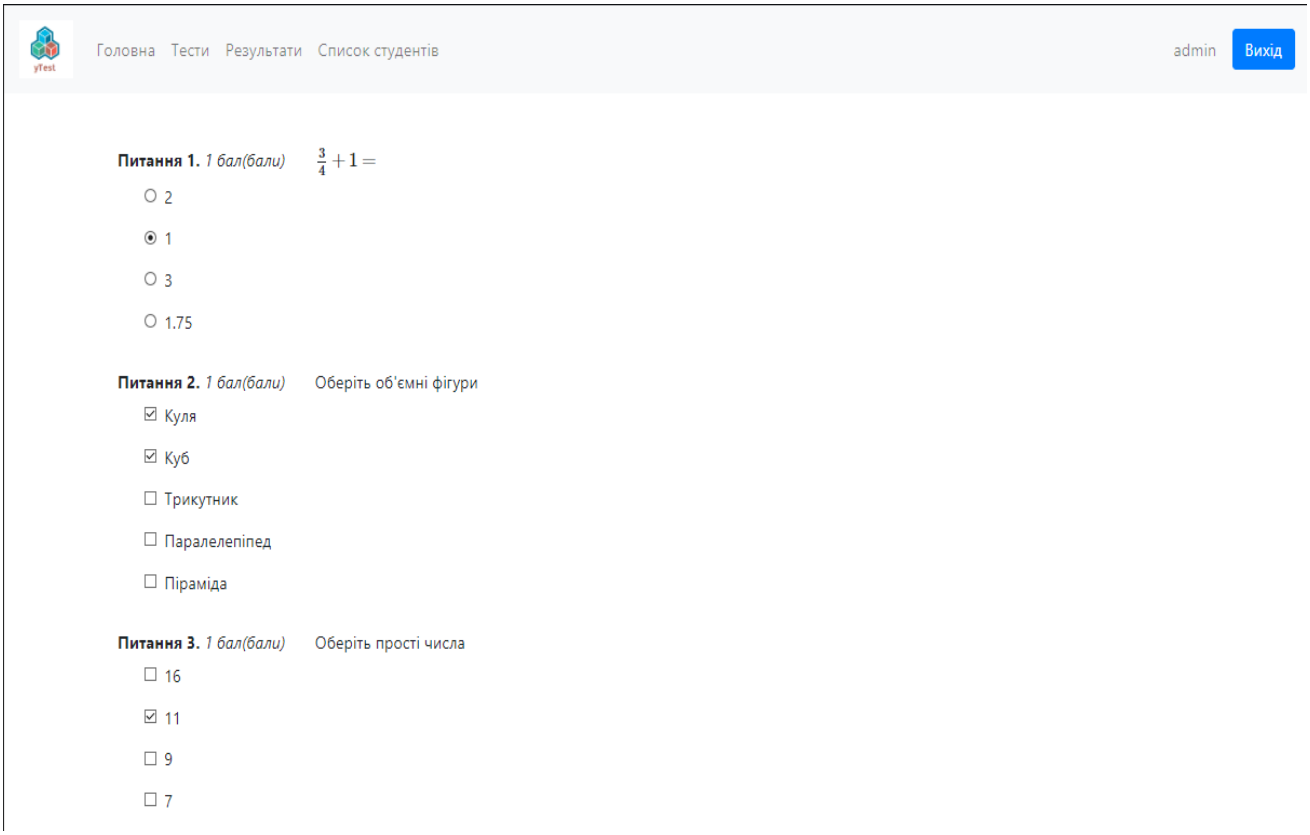

Рис. 3.6. Сторінка «test/id»

Сторінка «create» призначена для створення тестів. На ній знаходяться наступні компоненти:

– navbar — навігаційна панель сторінки, в якій знаходяться посилання на інші сторінки сайту, логін аутентифікованого користувача та кнопка вихід, для завершення роботи;

– href — посилання на сторінку з описом правил введення математичних формул;

– input-text1 — поле, для введення назви тесту;

– input-time — поле, для введення часу на проходження тесту;

– input-text2— поле, для введення опису тесту;

– input-number — поле, для введення кількості спроб на проходження тесту;

– buttonGroup — група кнопок для переходу між блоками питань та відповідей тесту;

– button — кнопка для додавання нового запитання;

– select — поле для вибору типу питання;

– input-text3 — поле для введення запитання;

– input-number2 — поле, для введення кількості балів за питання;

– filename1 — Завантаження зображення до питання тесту;

– filename2 — Завантаження другого зобраення до питання тесту;

– area — Блок, для вводу варіантів відповідей до питання;

– button2 — Кнопка для додавання ще однієї відповіді;

– button3 — Кнопка для видалення однієї відповіді;

– button4 — Кнопка для зберігання створеного тесту з усіма запитаннями в таблицях бази даних.

Зовнішній вигляд сторінки подано на рисунку 3.7.

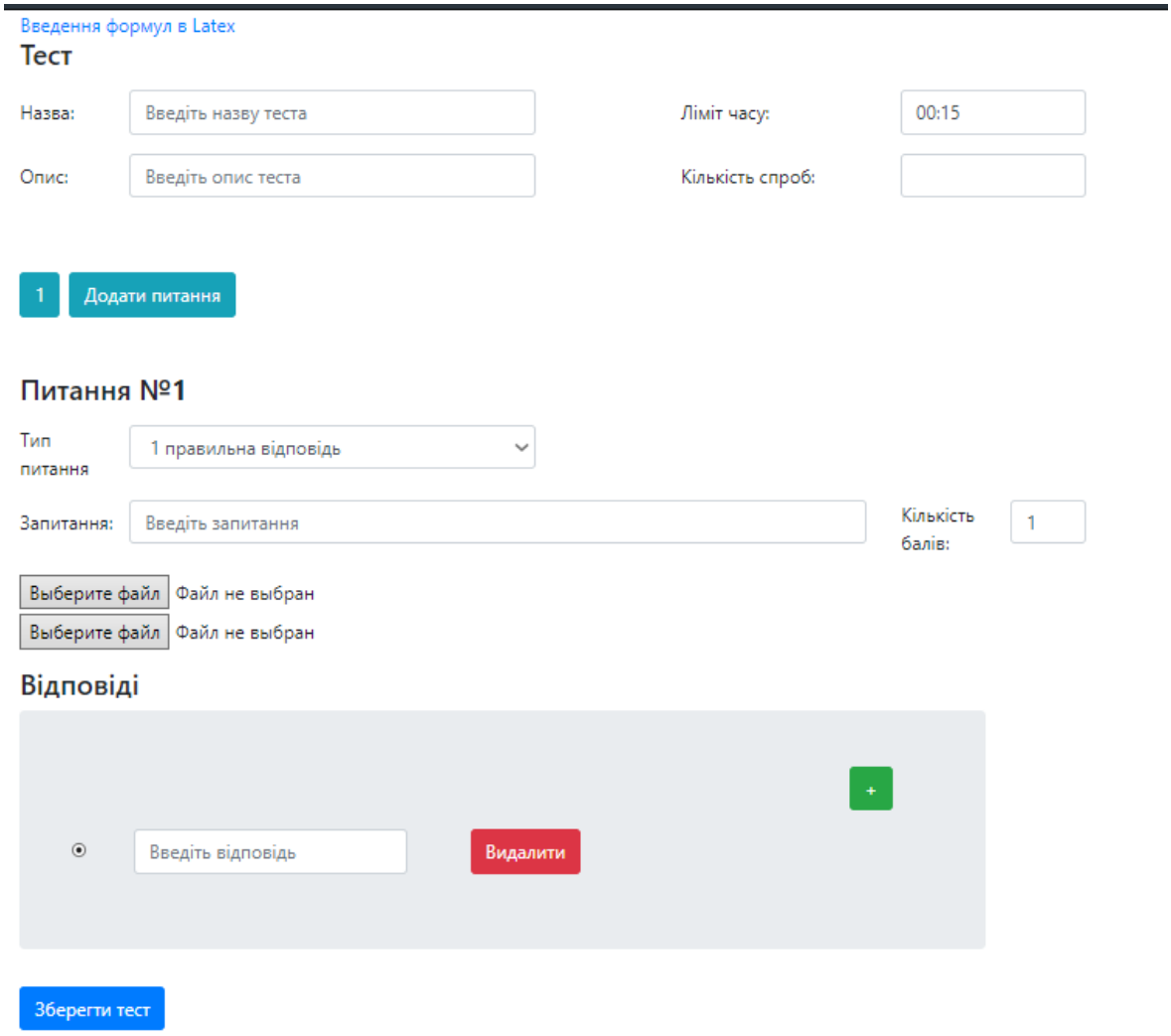

Рис. 3.7. Сторінка «create»

Сторінка «result» призначена для перегляду результатів тестів. На ній знаходяться наступні компоненти:

– navbar — навігаційна панель сторінки, в якій знаходяться посилання на інші сторінки сайту, логін аутентифікованого користувача та кнопка вихід, для завершення роботи;

– filter1 — поле для введення рядка назви тесту по якому буде здійснюватися пошук;

– filter2 — поле для введення рядка логіну студента по якому буде здійснюватися пошук;

– filter3 — поле для введення рядка групи студента по якому буде здійснюватися пошук;

– submit — кнопка для оновлення сторінки і здійснює пошук;

– table — таблиця для виведення даних з таблиці результатів тестів та оцінка за 100-бальною шкалою;

Зовнішній вигляд сторінки подано на рисунку 3.8.

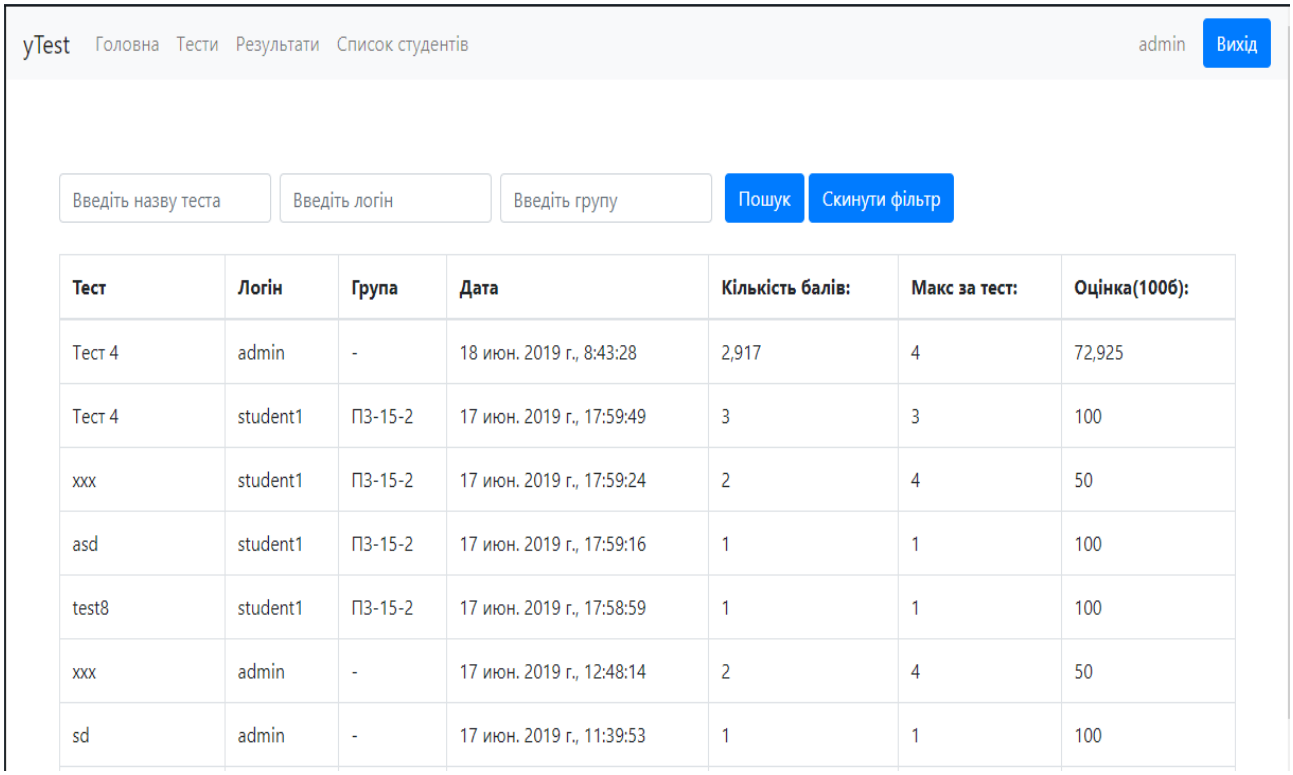

Рис. 3.8. Сторінка «result»

Сторінка «user» призначена для перегляду студентів та змінення прав доступу. Доступ до цієї сторінки мають тільки адміністратори. На ній знаходяться наступні компоненти:

– navbar — навігаційна панель сторінки, в якій знаходяться посилання на інші сторінки сайту, логін аутентифікованого користувача та кнопка вихід, для завершення роботи;

– button — кнопка для переходу на сторінку реєстрації студентів;

– table — таблиця для виведення даних про зареєстрованих користувачів;

– Edit — посилання на форму редагування прав доступу користувача.

Зовнішній вигляд сторінки подано на рисунку 3.9.

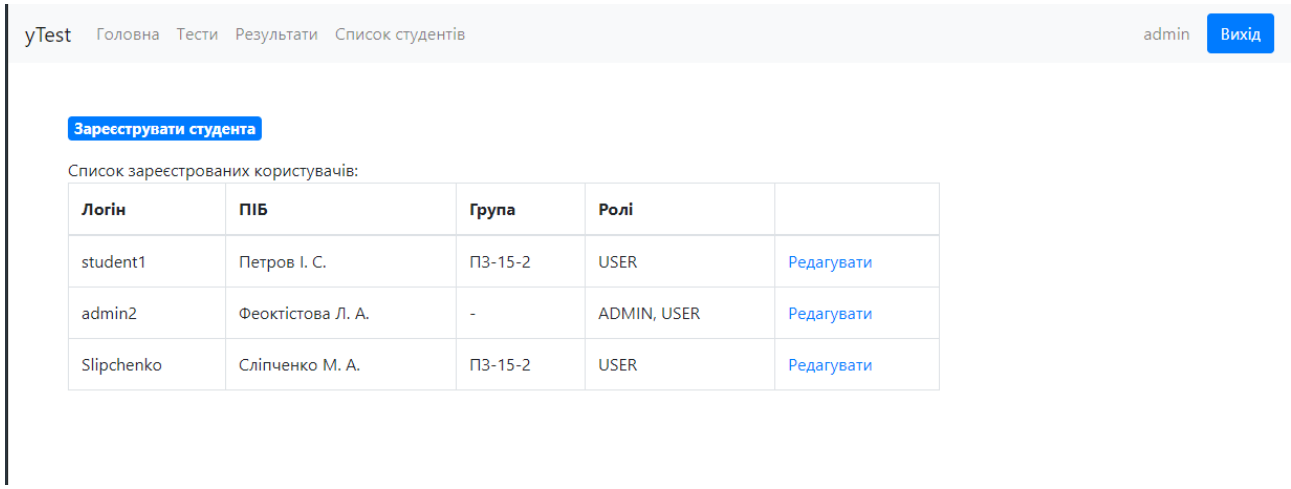

Рис. 3.9. Сторінка «user»

Форма «UserEdit» призначена для змінення прав доступу користувачам. Зовнішній вигляд форми подано на рисунку 3.10.

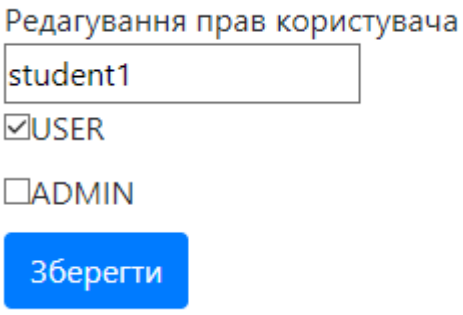

Рис. 3.10. Форма «UserEdit»

Сторінка «registration» призначена для реєстрації студентів. Доступ до цієї сторінки мають тільки адміністратори. На ній знаходяться наступні компоненти:

– navbar — навігаційна панель сторінки, в якій знаходяться посилання на інші сторінки сайту, логін аутентифікованого користувача та кнопка вихід, для завершення роботи;

– input-text1 — поле, для введення логіну користувача;

– input-text2 — поле, для введення пароля користувача;

– input-text3— поле, для введення ПІБ користувача;

– input-text4— поле, для введення групи користувача;

– button — кнопка для збереження користувача в таблиці користувачів.

Зовнішній вигляд сторінки подано на рисунку 3.11.

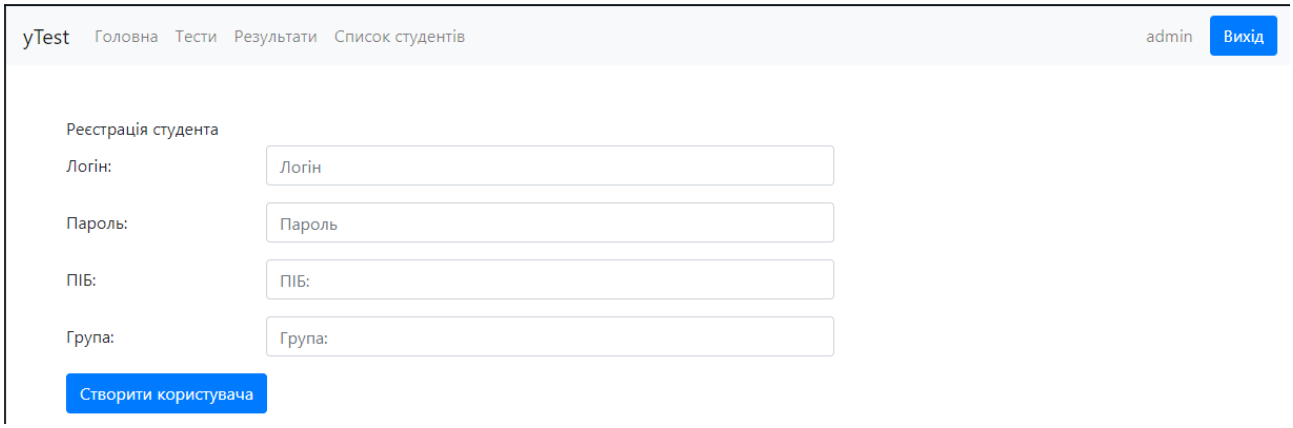

Рис. 3.11. Сторінка «registration»

3.5 Шаблони веб-сторінок додатку (вигляди)

Вигляди в архітектурному шаблоні MVC відображають форми та інформацію отриману від моделей(з бази даних) на веб-сторінках

Загальний вигляд веб-сторінок програмної системи створений мовою гіперрозмітки тексту HTML, оформлений таблицями стилів CSS та використанням JavaScript бібліотек:

jQuery — бібліотека JavaScript, що фокусується на взаємодії JavaScript і HTML. Бібліотека jQuery допомагає легко отримувати доступ до будь-якого елементу DOM, звертатися до атрибутів і вмісту елементів DOM, маніпулювати ними. Також бібліотека jQuery надає зручний API для роботи з AJAX. Зараз розробка jQuery ведеться командою jQuery на чолі з Джоном Резігом [14].

Для коректного відображення веб-сторінок на пристроях з різним розміром та розширенням екрану використовується веб-фреймворк Bootstrap.

Bootstrap (також відомий як Twitter Bootstrap) — вільний набір інструментів для створення сайтів і веб-додатків. Включає в себе HTML- і CSS-шаблони оформлення для типографіки, веб-форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсу, включаючи JavaScript-розширення. Bootstrap використовує сучасні напрацювання в області CSS і HTML, тому необхідно бути уважним при підтримці старих браузерів [15].

Деякі частовживані частини шаблону були винесені в окремі блоки(файли), що підключаються багаторазово — таким чином реалізується багатовживаність і перевикористання коду.

Загальні частини шаблонів веб-сторінок:

– common.ftl — шаблон для підключення скриптів;

– isAdmin.ftl — шаблон, який перевіряє чи є аунтефікований користувач адміністратором;

– login.ftl — шаблон авторизації;

– navbar.ftl — шаблон шапки сайту;

– pager.ftl — шаблон виведення даних по сторінкам;

– security.ftl — шаблон, який перевіряє дані користувача.

Всі шаблони написані на HTML за допомогою шаблонізатора FreeMarker template, з використанням каскадних таблиць стилів CSS для оформлення зовнішнього вигляду. FreeMarker — компілює обробник шаблонів, написаний на Java, один з інструментів, що дозволяють відокремити логіку і дані від подання в дусі концепції Model-view-controller. Використовується переважно при розробці web-додатків з використанням Java-сервлетов, також може

використовуватися для виведення тексту в інших випадках: генерація CSS, вихідного коду Java і т. д [15].

На відміну від JSP FreeMarker не є залежним від архітектури сервлету або від протоколу HTTP. Таким чином шаблонизатор може використовуватися не тільки в web-проєктах. FreeMarker є вільним програмним забезпеченням [16].

3.6 Способи використання

При запуску контрольно діагностичної системи відкривається головна сторінка.

Для авторизації, користувачу необхідно натиснути на кнопку «Увійти» у верхньому меню. Після натискання на кнопку відкриється форма входу, у яку необхідно ввести логін та пароль. Після введення даних необхідно натиснути на кнопку «Увійти» і якщо дані введені вірно, відкриється головна сторінка додатку. У верхньому меню буде відображатися напис з логіном авторизованого користувача та кнопка «Вихід», при натисканні на яку користувач виходить з облікового запису на рисунку 3.12 представлена форма авторизації користувача.

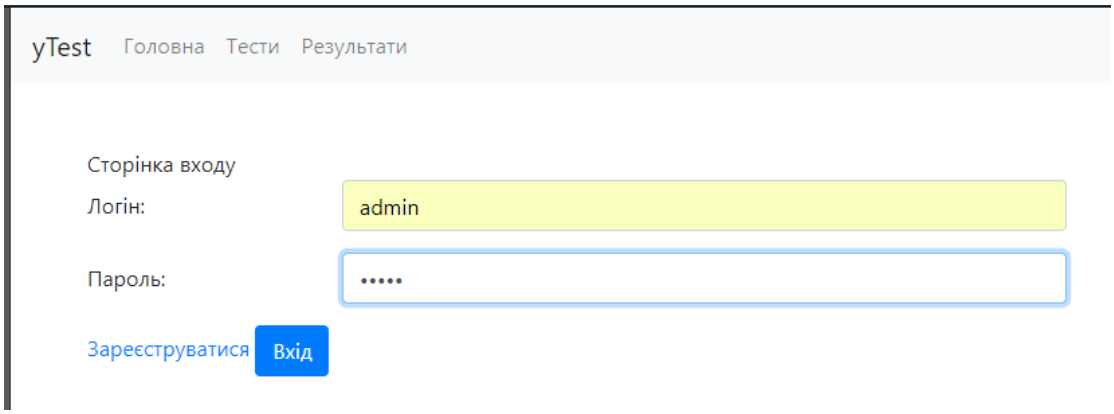

Рис. 3.12. Сторінка авторизації

Якщо на сайті авторизувався користувач з роллю «Адміністратор», він має можливість створення, редагування, та видалення тестів; можливість надавати права іншим зареєстрованим користувачам. Також адміністратор має доступ до перегляду результатів всіх студентів.

Якщо на сайті авторизувався користувач з роллю «Студент», він має можливість переглядати інформацію про, проходити тести та перегляди власні результати проходження тестів.

Для того, щоб переглянути список тестів, у головному меню потрібно обрати пункт «Тести».

На рисунку 3.13 наведено сторінку «Тести», яка відкрита користувачем з роллю «Адміністратор», тому вверху сторінки розташована кнопка створення тесту.

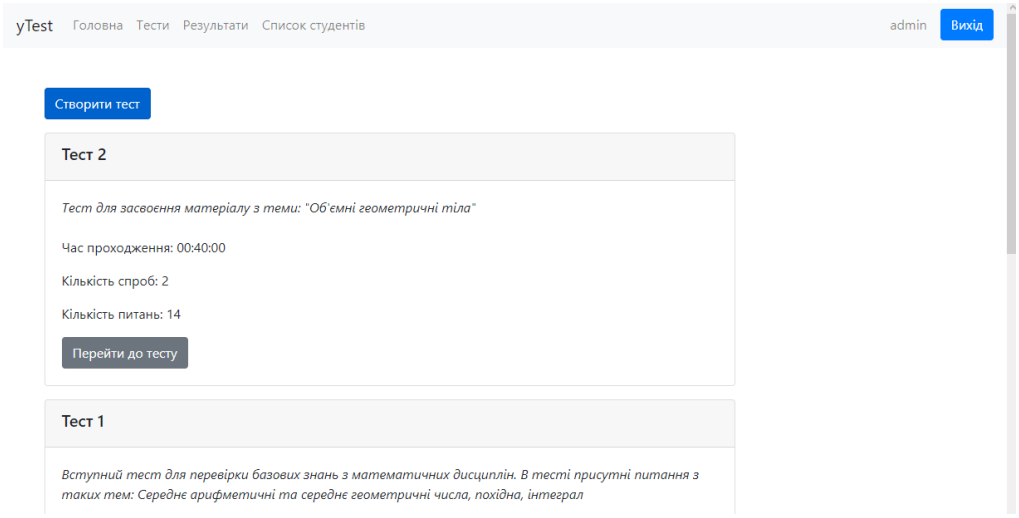

Рис. 3.13. Сторінка тестів

Для того, щоб створити новий тест необхідно натиснути на кнопку «Створити тест». Після натискання, відкриється форма створення нового тесту, в якій необхідно буде ввести інформацію про тест та ввести запитання і відповіді, які входять до тесту, як показано на рисунку 3.14.

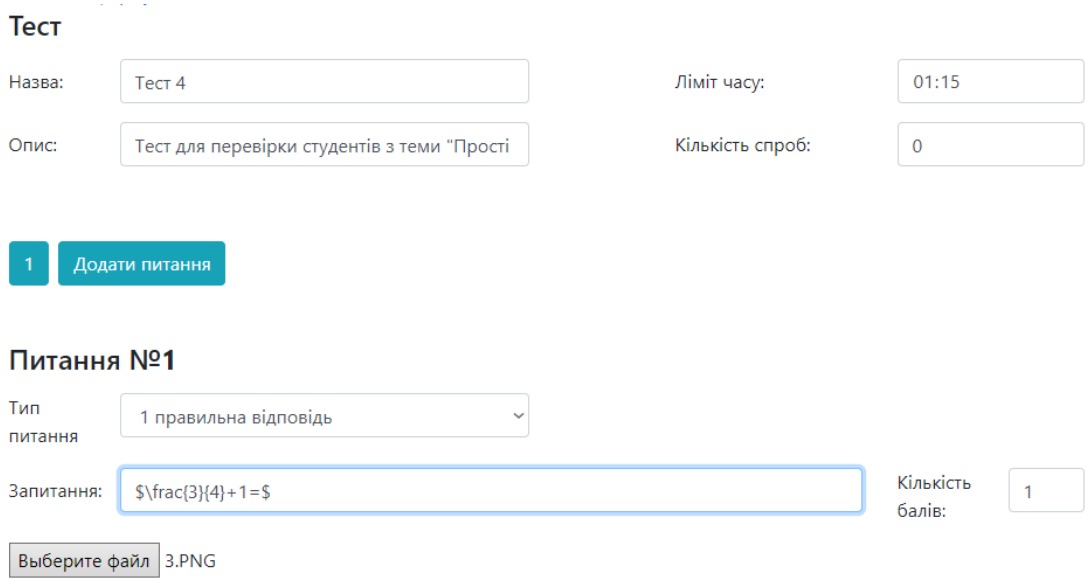

Рис. 3.14. Створення нового тесту

Якщо користувач захочу для тесту зробити необмежену кількість спроб проходження тесту, необхідно ввести число 0. При наведенні на поле буде виведена відповідна підказка, як зображено на рисунку 3.15

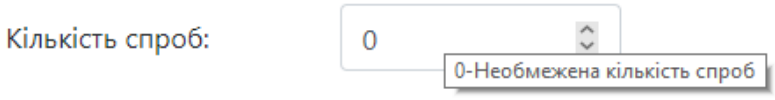

Рис. 3.15. Підказка при введенні кількості спроб

Питання тесту може бути одним з чотирьох типів:

- питання з однією правильною відповіддю;
- питання з декількома правильними відповіддями;
- питання з введенням правильної відповіді з клавіатури;
- питання на встановлення відповідності.

Обрати тип питання можна просто змінивши тип в полі зі списком, як показано на рисунку 3.16.

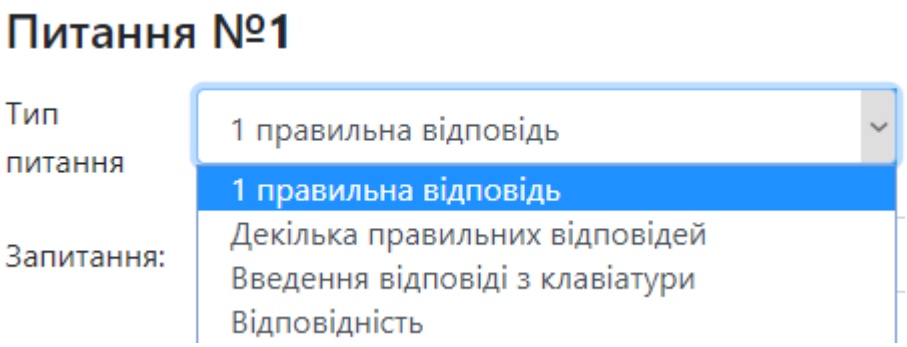

Рис. 3.16. Підказка при введенні кількості спроб

Після введення тексту питання і вибору типу питання необхідно ввести відповіді. Для цього реалізовані кнопки «+» та «Видалити» які додають та видаляють поля для вводу відповідей. Для питання с типом введення правильної відповіді з клавіатури цих кнопок немає. Максимальна кількість полей для введення — 10 для кожного питання, після створення десятого блоку для введення кнопка «+» більше не буде створювати поле.

Для того, щоб зберегти одне питання, необхідно натиснути на кнопку «Додати питання», яка знаходиться над блоком «Питання». Після натискання буде створена нова форма для введення наступного питання. Активне питання виділяється кольором, як наведено на рисунку 3.17

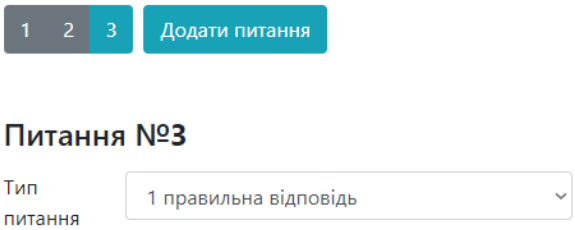

Рис. 3.17. Кнопка додати питання та виділення активного питання

Адміністратор може в будь-який момент переглянути вже створене питання, для цього необхідно клацнути на питання з бажаним номером. Після цього на формі будуть відображені дані бажаного питання, як зображено на рисунку 3.18.

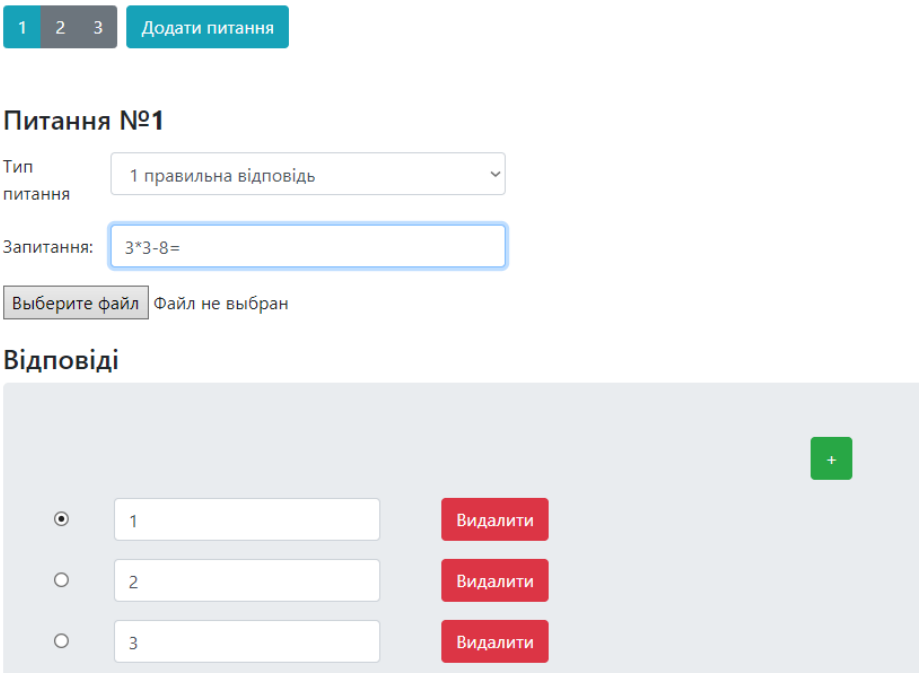

Рис. 3.18. Перехід до вже створеного питання

Якщо необхідно до тексту прикріпити зображення — необхідно натиснути кнопку «Оберіть файл», після чого відриється діалогове вікно, в якому адміністратору необхідно буде обрати файл. Після обрання файлу його ім’я буде зображено поряд з кнопкою, як показано на рисунках 3.19 – 3.20.

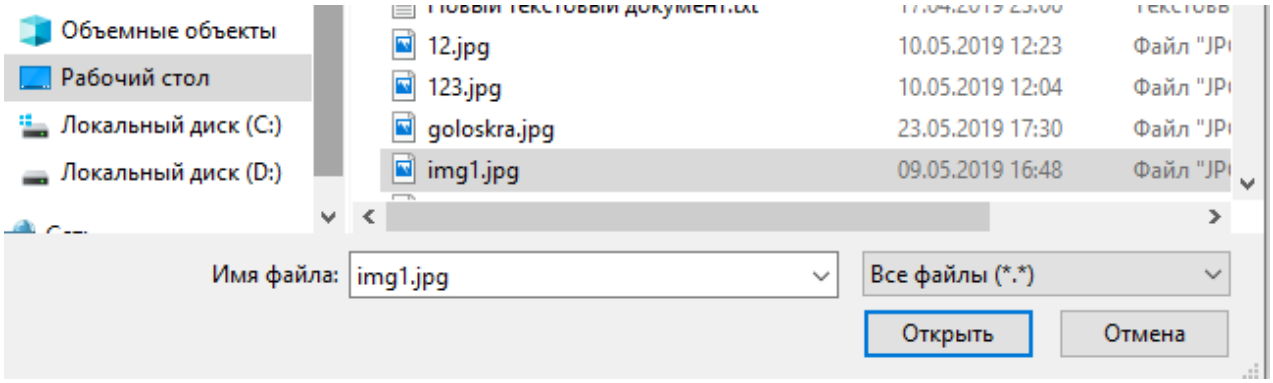

Рис. 3.19. Діалогове вікно завантаження зображення

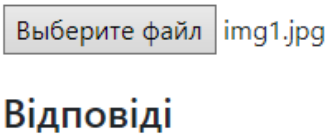

Рис. 3.20. Обраний файл «img1.jpg»

Для того, щоб занести всі питання до бази даних, необхідно внизу сторінки натиснути кнопку «Зберегти тест», після чого при успішному збереженні з’явиться повідомлення та адміністратора буде переправлено на сторінку з усіма тестами, як зображено на рисунках 3.21 – 3.23

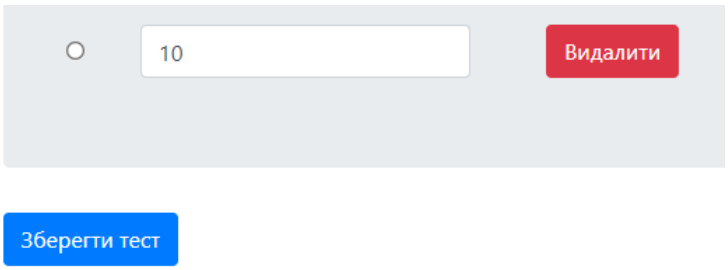

Рис. 3.21. Кнопка «Зберегти тест»

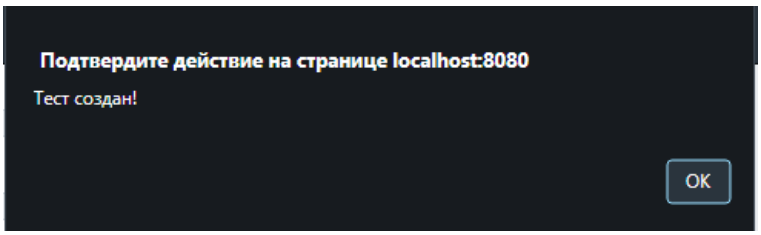

Рис. 3.22. Повідомлення про збереження тесту

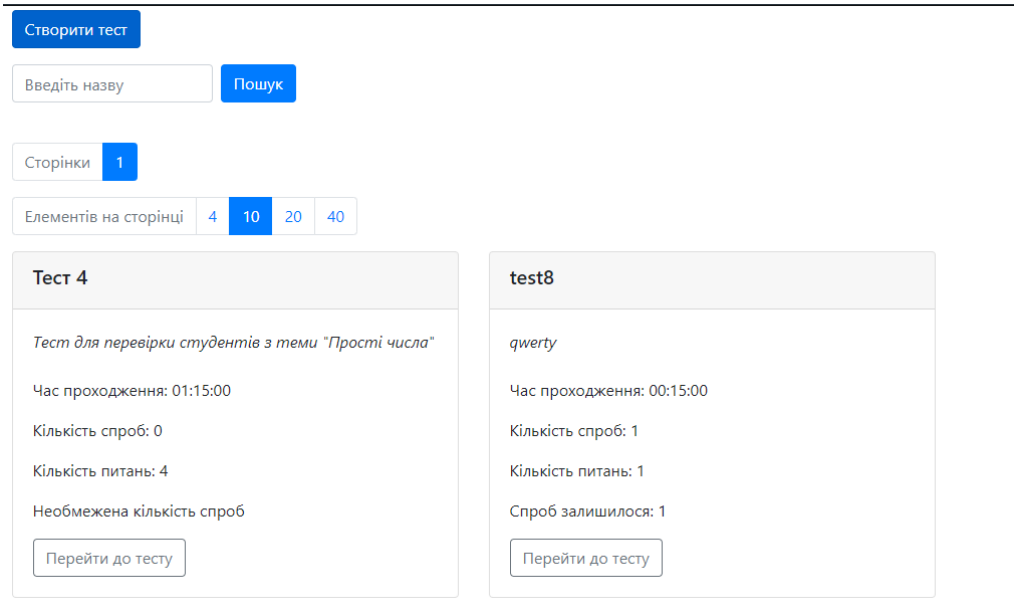

Рис. 3.13. Перегляд створеного тесту

На сторінці тестів реалізований пошук, та зручне розташування тестів у вигляді сторінок. Результат роботи пошуку зображено на рисунку 3.24.

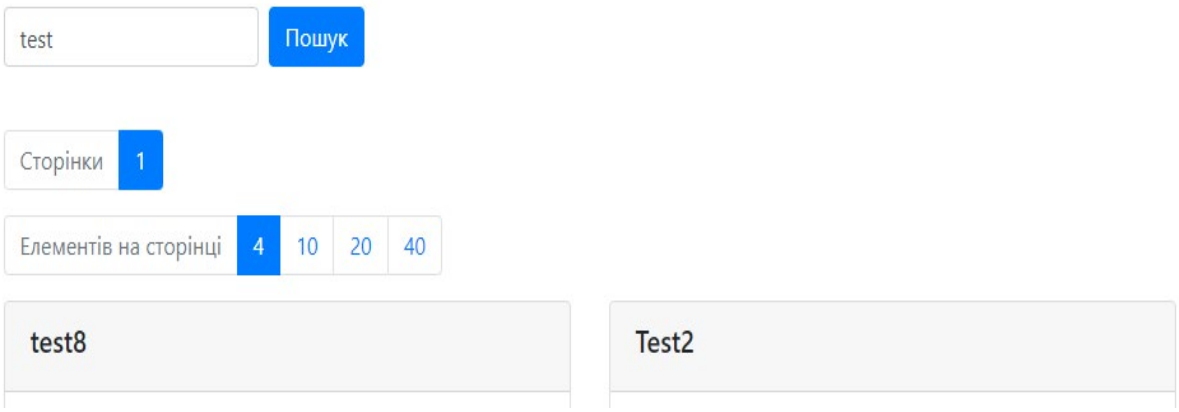

Рис. 3.24. Демонстрація пошуку тесту

Якщо користувач використав всі спроби проходження тесту, тоді в тесті з’явиться повідомлення, що всі спроби вичерпано не буде кнопки «Перейти до тесту». Ситуація, коли у користувача не залишилося спроб зображена на рисунку 3.25.

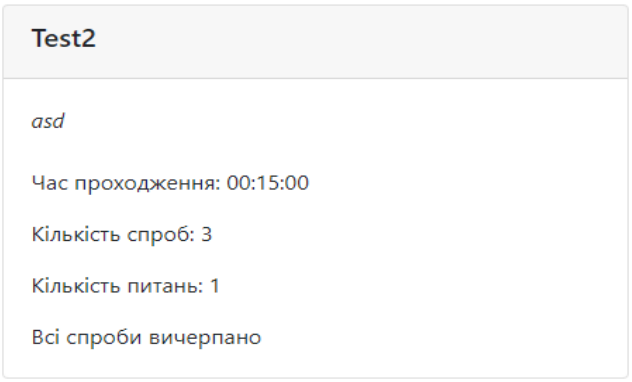

Рис. 3.25. Тест, в якому всі спроби проходження вичерпано

Для того, щоб перейти до проходження тесту необхідно натиснути на кнопку «Перейти до тесту». Після цього відкриється тест з питаннями. Після проходження тесту необхідно натиснути на кнопку «Зберегти результат». Після цього з’явиться напис з кількістю балів та правильні відповіді буде позначено зеленим кольором, неправильні — червоним. Проходження тесту зображено на рисунках 3.26 – 3.27.

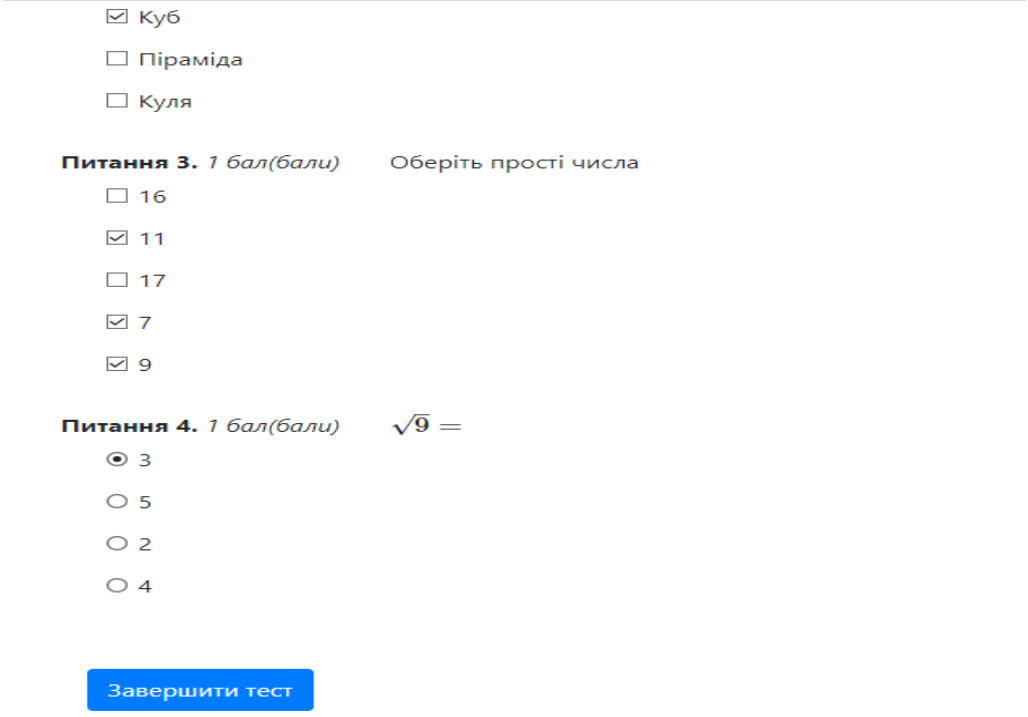

Рис. 3.26. Проходження тесту

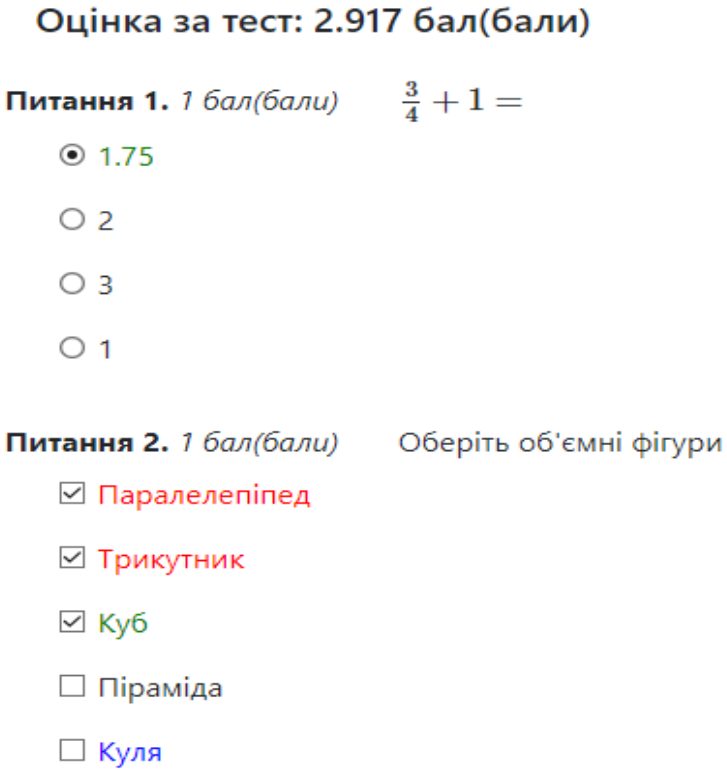

Рис. 3.27. Результат тесту

Для перегляду результатів тесту необхідно перейти на сторінку результатів. Студенту доступні тільки власні результати, а адміністратору

результати всіх студентів та, для зручності, реалізований пошук за трьома полями: Назва тесту, Логін, Група. Для перегляду всіх результатів студентів необхідно натиснути на кнопку «Скинути фільтр». Результат пошуку зображений на рисунку 3.28.

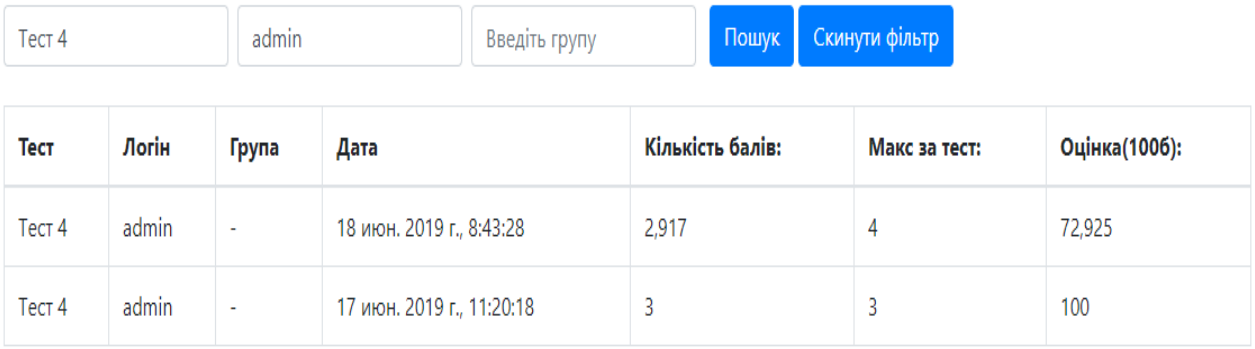

Рис. 3.28. Пошук по результатам

Реєструвати студентів може тільки адміністратор. Для цього необхідно перейти на сторінку зі списоком студентів та натиснути на клавішу «Зареєструвати студента». На цій сторінці адміністратор з логіном «admin» не відображається, для того, щоб другий адміністратор не зміг позбавити цього користувача прав адміністратор. Сторінка зі списком студентів зображена на рисунку 3.29.

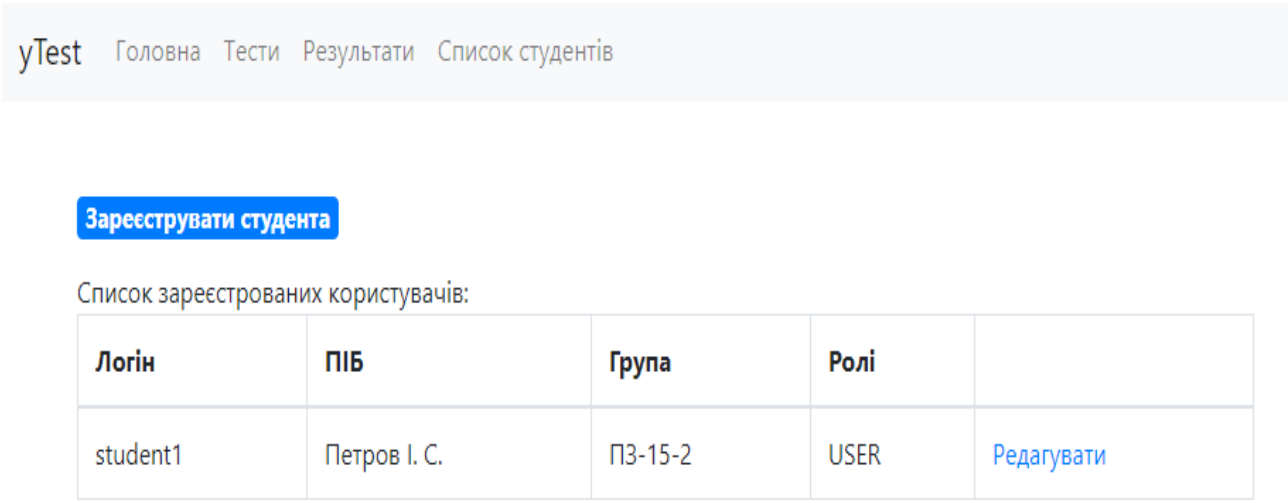

Рис. 3.29. Список зареєстрованих студентів

Після натиснення кнопки «Зареєструвати студента» відкриється форма реєстрації нового користувача, яка зображена на рисунку 3.30.

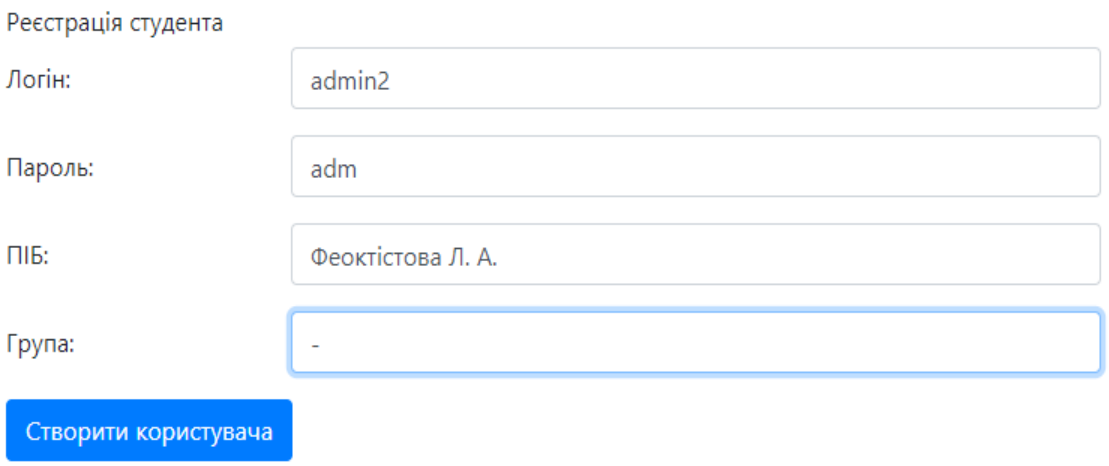

Рис. 3.30. Форма реєстрації користувача

Щоб змінити користувачу права, необхідно натиснути на напис редагувати в рядку с необхідним користувачем. Після цього відкриється форма редагування прав користувача, в якій можна надати користувачу права адміністратора. Демонстрація додавання прав адміністратора зображена на рисунках 3.31 – 3.33.

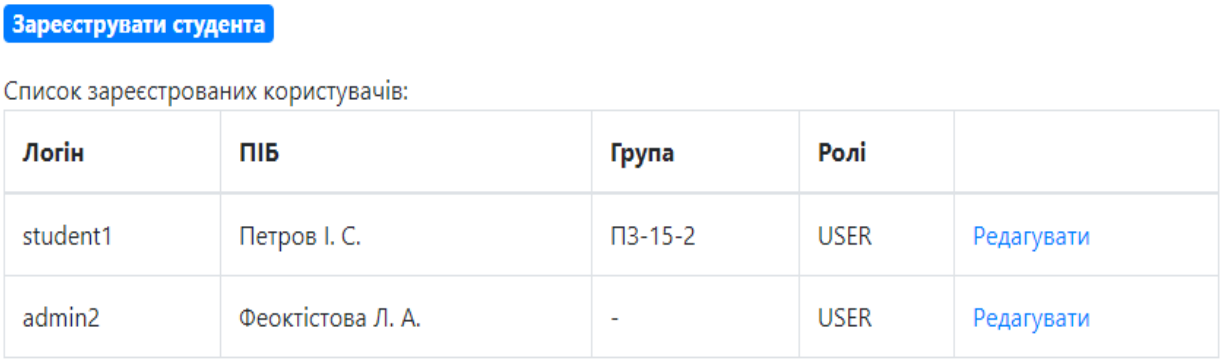

Рис. 3.31. Список користувачів

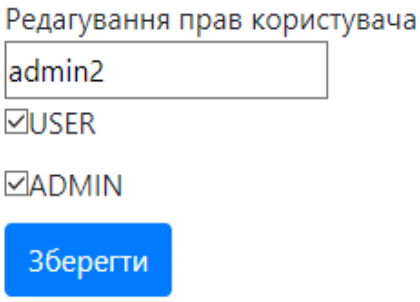

Рис. 3.32. Форма змінення прав користувача

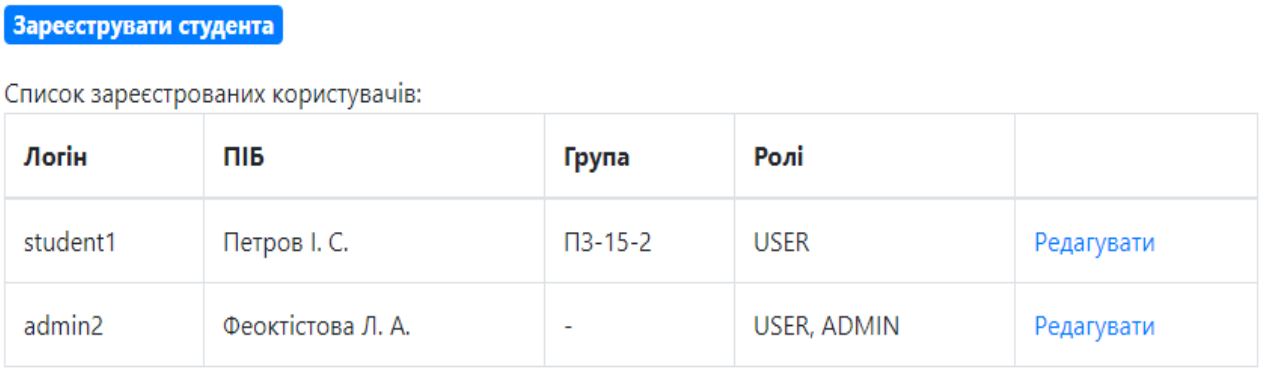

Рис. 3.33. Список користувачів після редагування

Після завершення необхідно роботи необхідно вийти з облікового запису, клацнувши на кнопку «Вихід», в правому верхньому куті, яка зображена на рисунку 3.34.

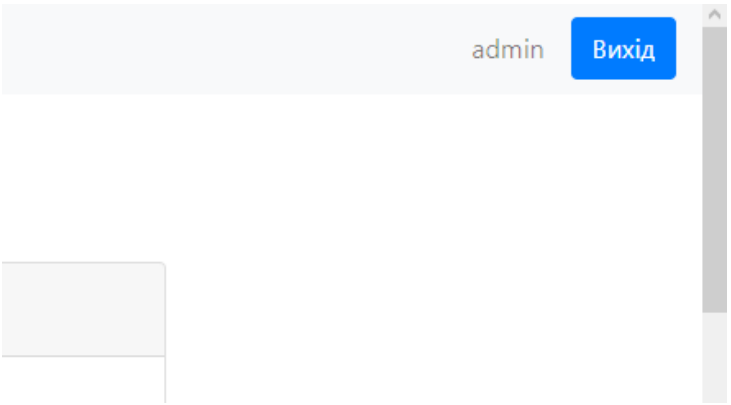

Рис. 3.34. Вихід з облікового запису

ВИСНОВКИ

Під час роботи над магістерським проєктом «Розробка та дослідження ефективності застосування контрольно-діагностичної системи для перевірки знань з математичних дисциплін» засобами середовища IntelliJ IDEA 2019 Spring MVC було створено контрольну-діагностичну систему, яка надає можливість для перевірки знань студентів.

Контрольно-діагностична система забезпечує роботу з моделями бази даних «Тести», «Запитання», «Відповіді» та «Результати». Для кожної моделі контрольно-діагностична система забезпечує можливість перегляду, додавання даних, редагування та видалення.

Розроблена контрольно-діагностична система має інтуїтивно-зрозумілий графічний інтерфейс, а користувач потребує лише базових знань роботи з комп’ютером.

Розроблена пояснювальна записка акцентує увагу на інтерфейсі контрольно-діагностичної системи та інструкції користувача, що спрощує його використання.

Контрольно-діагностична система може бути корисною та актуальною для використання її студентами та викладачами у навчальному процесі, адже дозволяє спростити перевірку знань студентів з математичних дисциплін, що пришвидшує роботу викладачів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 МЕТОДИЧНІ РЕКОМЕНДАЦІЇ до виконання кваліфікаційних робіт здобувачами другого (магістерського) рівня вищої освіти спеціальностей 121 «Інженерія програмного забезпечення» / / Б.І. Мороз, О.В. Іванченко, О.В. Реута, О.С. Шевцова; М-во освіти і науки України, Нац. техн. ун-т “Дніпровська політехніка”. – Дніпро : НТУ «ДП», 2021. – 56 с.

2 The IntelliJ IDEA Blog [Електронний ресурс] / The IntelliJ IDEA Blog 2023 – Режим доступа: https://blog.jetbrains.com/ru/idea/ – Назва з екрану.

3 Знайомлення з IntelliJ IDEA [Електронний ресурс] / Знайомлення з IntelliJ IDEA 2023 – Режим доступа: https://ravesli.com/getting-started-with-intellij-idea/ – Назва з екрану.

4 Налаштування IntelliJ IDEA Jet Brains [Електронний ресурс] / Налаштування IntelliJ IDEA Jet Brains 2023 – Режим доступа: https://docs.spongepowered.org/5.1.0/ru/plugin/workspace/idea.html – Назва з екрану.

5 Огляд типів індексів Oracle, MySQL, PostgreSQL, MS SQL [Електронний ресурс] / Огляд типів індексів Oracle, MySQL, PostgreSQL, MS SQL, 2010. – Режим доступа: https://habr.com/ru/post/102785/. – Назва з екрану.

6 MySQL [Електронний ресурс] / MySQL, 2023. – Режим доступа: https://en.wikipedia.org/wiki/MySQL. – Назва з екрану.

7 MySQL Server [Електронний ресурс] / MySQL Server, 2023. – Режим доступа: https://mysql.com/products/server. – Назва з екрану.

8 OWIN і Katana [Електронний ресурс] // metanit.com. Режим доступу: https://metanit.com/java/mvc5/11.1.php – Назва з екрану.

9 Введення у Spring Boot MVC [Електронний ресурс] // metanit.com/spring/mvc. Режим доступа: https://metanit.com/sharp/mvc/1.1.php – Назва з екрану.

10 Моделі Spring Boot MVC [Електронний ресурс] // msdn.microsoft.com. Режим доступу: https://msdn.microsoft.com/ru-ru/library/dd410405(v=vs.98).aspx – Назва з екрану.

11 Model-View-Controller (MVC) [Електронний ресурс] / Model-View-Controller (MVC), 2023. – Режим доступа: https://www.techtarget.com/whatis/definition/Model-View-Controller#:~:text=Model%2DView%2DController%20(MVVM)%20is%20a%20software%20design,logic%20and%20user%20interface%20controls. – Назва з екрану.

12 Model–view–Controller [Електронний ресурс] / Model–view–controller, 2023. – Режим доступа: https://en.wikipedia.org/wiki-Model%E293view%E2controller. – Назва з екрану.

13 Model-View-Controller (MVC) [Електронний ресурс] / Model-View-Controller (MVC), 2023. – Режим доступа: https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm. – Назва з екрану.

14 Including jQuery [Електронний ресурс] / jQuery https://www.npmjs.com/package/jquery – Назва з екрану.

15 Bootstrap [Електронний ресурс] // Вікіпедія: вільна енциклопедія. Режим доступу: https://ru.wikipedia.org/wiki/Bootstrap_(фреймворк) – Назва з екрану.

16 JavaServer Pages [Електронний ресурс] // Вікіпедія: вільна енциклопедія. Режим доступу: https://ua.wikipedia.org/wiki/JavaServer_Pages – Назва з екрану.

17 Apache FreeMarker [Електронний ресурс] // Вікіпедія: вільна енциклопедія. Режим дос https://ru.wikipedia.org/wiki/Apache_FreeMarker Pages – Назва з екрану.

18 GitHub Copilot [Електронний ресурс] / GitHub Copilot– Режим доступа: https://visualstudio.microsoft.com/vs/. – Назва з екрану.

19 Windows technical documentation for developers and IT pros [Електронний ресурс] / Windows technical documentation for developers and IT

pros, 2023. – Режим доступа: https://learn.microsoft.com/en-us/windows/. – Назва з екрану.

20 Складання списку літератури в навчальних виданнях : посіб. для наук.-пед. працівників / В.О. Салов, О.Н. Нефедова, О.Н. Ільченко, В.В. Панченко, Т.О. Недайвода, В.Г. Римар ; М-во освіти і науки України, Нац. гірн. ун-т. – Д. : НГУ, 2013. – 39 с

Додаток А

КОД ПРОГРАМИ

```
package com.yanyshivskyi.yMathTest;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

}


// Файл конфігурації MvcConfig

package com.yanyshivskyi.yMathTest.config;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class MvcConfig implements WebMvcConfigurer {
    @Value("${upload.path}")
    private String uploadPath;

    public void addViewControllers(ViewControllerRegistry registry) {
        registry.addViewController("/login").setViewName("login");
    }

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("/img/**")
                .addResourceLocations("file:///" + uploadPath + "/");
        registry.addResourceHandler("/static/**")
                .addResourceLocations("classpath:/static/");
    }

}

// Файл конфігурації WebSecurityConfig

package com.yanyshivskyi.yMathTest.config;

import com.yanyshivskyi.yMathTest.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationManager
Builder;
```

```
import
org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecuri
ty;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapte
r;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    private UserService userService;
    @Autowired
    private PasswordEncoder passwordEncoder;

    @Bean
    public PasswordEncoder getPasswordEncoder(){
        return new BCryptPasswordEncoder(8);
    }
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
                    .authorizeRequests()
                    .antMatchers("/", "/registration", "/static/**").permitAll()
                    .anyRequest().authenticated()
                .and()
                    .formLogin()
                    .loginPage("/login")
                    .permitAll()
                    .defaultSuccessUrl("/main")
                .and()
                    .rememberMe()
                .and()
                    .logout()
                    .permitAll();
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userService)
                .passwordEncoder(passwordEncoder);
    }
}

//Контролер MainControler

package com.yanyshivskyi.yMathTest.controller;

import com.yanyshivskyi.yMathTest.domain.Test;
import com.yanyshivskyi.yMathTest.domain.User;
import com.yanyshivskyi.yMathTest.service.TestService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.web.PageableDefault;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
```

```
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class MainController {
    @Autowired
    private TestService testService;

    @Value("${upload.path}")
    private String uploadPath;

    @GetMapping("/")
    public String greeting(Model model) {
        return "greeting";
    }



    @GetMapping("/main")
    public String main(@AuthenticationPrincipal final User user,
                       @RequestParam(required=false, defaultValue = "") String filter, Model
model,
                       @PageableDefault(sort = { "id" }, direction = Sort.Direction.DESC)
Pageable pageable) {


        Page<Test> page=testService.findAll(pageable);
        if (filter!=null && !filter.isEmpty()) {
            page = testService.findTest(filter, pageable);
        }
        else {
            page=testService.findAll(pageable);
        }
        model.addAttribute("page", page);
        model.addAttribute("filter", filter);
        int[] count_qq=new int[page.getSize()];
        int[] count_gg=new int[page.getSize()];
        int j=0;
        for(Test i:page){
            count_qq[j] = testService.countQuestion(i.getId());
            count_gg[j] = testService.countTry(i, user);
            j++;
        }
        model.addAttribute("count_q", count_qq);
        model.addAttribute("url", "/main");
        model.addAttribute("count_g", count_gg);

        return "main";
    }

}

// Контролер CompleteTestControler
package com.yanyshivskyi.yMathTest.controller;

import com.yanyshivskyi.yMathTest.domain.Answer;
import com.yanyshivskyi.yMathTest.domain.Question;
import com.yanyshivskyi.yMathTest.domain.Result;
import com.yanyshivskyi.yMathTest.service.CompleteTestService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
```

```
import java.util.*;


@Controller
public class CompleteTestController {
    @Autowired
    private CompleteTestService cTestService;

    private Long id_test;


    @GetMapping("test/{id}")
    public String test_idd(@PathVariable Long id, Model model) {
        List<Question> list =cTestService.findQuests(id);
        List<List<Answer>> all = new ArrayList<>();
        List<Answer> answers=cTestService.findAnswer(list.get(0));
        for(Question qst:list) {
            List<Answer> answer = cTestService.findAnswer(qst);
            Collections.shuffle(answer);
            all.add(answer);
        }

        model.addAttribute("quests", list);
        model.addAttribute("answers", all);
        id_test=id;

        return "testId";
    }


    @PostMapping("/test/{id}")
    public String compllTes1(@RequestParam String mystr, @RequestParam Float fpoint) {
        System.out.println(id_test);
        System.out.println(fpoint);
        cTestService.saveResult(id_test, mystr, fpoint);
        return "redirect:/main";
    }

    @GetMapping("/result")
    public String test(@RequestParam(required = false, defaultValue = "") String filter1,
                       @RequestParam(required = false, defaultValue = "") String filter2,
                       @RequestParam(required = false, defaultValue = "") String filter3,
                       Model model) {
        List<Result> listRes = cTestService.findResults();

        if ( (filter1!=null && !filter1.isEmpty()) &&
             (filter2==null ||  filter2.isEmpty()) &&
             (filter3==null || filter3.isEmpty())){
                listRes = cTestService.findResultsT(filter1);
        } else if ( (filter2!=null && !filter2.isEmpty()) &&
                    (filter1==null ||  filter1.isEmpty()) &&
                    (filter3==null || filter3.isEmpty())) {
                        listRes = cTestService.findResultsU(filter2);
        } else if ( (filter3!=null && !filter3.isEmpty()) &&
                    (filter1==null ||  filter1.isEmpty()) &&
                    (filter2==null || filter2.isEmpty())) {
                        listRes = cTestService.findResultsG(filter3);
        } else if ( (filter1!=null && !filter1.isEmpty()) &&
                    (filter2!=null &&  !filter2.isEmpty()) &&
                    (filter3==null || filter3.isEmpty())) {
                        listRes=cTestService.findResultsTU(filter1, filter2);
        } else if ( (filter1!=null && !filter1.isEmpty()) &&
                    (filter3!=null &&  !filter3.isEmpty()) &&
```

```
                    (filter2==null || filter2.isEmpty())) {
                        listRes=cTestService.findResultsTG(filter1, filter3);
        } else if ( (filter2!=null && !filter2.isEmpty()) &&
                    (filter3!=null &&  !filter3.isEmpty()) &&
                    (filter1==null || filter1.isEmpty())) {
                        listRes=cTestService.findResultsUG(filter2, filter3);
        } else if ( (filter1!=null && !filter1.isEmpty()) &&
                    (filter2!=null &&  !filter2.isEmpty()) &&
                    (filter3!=null && !filter3.isEmpty())) {
                        listRes=cTestService.findResultsTUG(filter1,filter2, filter3);
        } else {        listRes=cTestService.findResults();
        }

        model.addAttribute("filter1", filter1);
        model.addAttribute("filter2", filter2);
        model.addAttribute("filter3", filter3);

        model.addAttribute("results", listRes);


        List<Float> maxPoint=new ArrayList<>();

        for(Result res:listRes){
            maxPoint.add(cTestService.getPoint(res.getTest()));
        }
        model.addAttribute("maxPoint", maxPoint);

        return "result";
    }

}
```

// Контролер ControlerUtils

```
package com.yanyshivskyi.yMathTest.controller;

import org.springframework.validation.BindingResult;
import org.springframework.validation.FieldError;

import java.util.Map;
import java.util.stream.Collector;
import java.util.stream.Collectors;

public class ControllerUtils {
    static Map<String, String> getErrors(BindingResult bindingResult) {
        Collector<FieldError, ?, Map<String, String>> collector = Collectors.toMap(
                fieldError -> fieldError.getField() + "Error",
                FieldError::getDefaultMessage
        );
        return bindingResult.getFieldErrors().stream().collect(collector);
    }
}
```

// Контролер RegistrationControler

```
package com.yanyshivskyi.yMathTest.controller;

import com.yanyshivskyi.yMathTest.domain.Role;
import com.yanyshivskyi.yMathTest.domain.User;
import com.yanyshivskyi.yMathTest.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.stereotype.Controller;
```

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;

import java.util.Collections;
import java.util.Map;


@Controller
@PreAuthorize("hasAuthority('ADMIN')")
public class RegistrationController {
    @Autowired
    private UserService userService;

    @GetMapping("/registration")
    public String registration(){
        return "registration";
    }

    @PostMapping("/registration")
    public String addUser(User user, Map<String, Object> model){
        User userFromDb = userService.findByUsername(user.getUsername());

        if(userFromDb!=null) {
            model.put("message", "User exists");
            return "registration";
        }

        user.setActive(true);
        user.setRoles(Collections.singleton(Role.USER));
        System.out.println("error");
        userService.save(user);

        return "redirect:/user";
    }
}

// Контролер TestControler
package com.yanyshivskyi.yMathTest.controller;
import com.yanyshivskyi.yMathTest.domain.Test;
import com.yanyshivskyi.yMathTest.service.TestService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;
import javax.validation.Valid;
import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Map;

@Controller
@PreAuthorize("hasAuthority('ADMIN')")
public class TestController {
    @Autowired
    TestService tr;

    @GetMapping("/create")
    public String createTest(Model model) {
```

```
        return "createTest";
    }
    @PostMapping("create")
    public String saveTest(
            @RequestParam String[] cquest, @RequestParam String[] cquesttype,
            @RequestParam Float[] countPoint, @RequestParam(required = false)
MultipartFile[] filename1,
            @RequestParam(required = false) MultipartFile[] filename2, @RequestParam
String[] canswer,
            @RequestParam(required=false, name="canswercor") String[] canswercor,
            @RequestParam(required = false, name="canswer1") String canswer1,
            @Valid Test test, BindingResult bindingResult,
            Model model) throws ParseException, IOException {

        SimpleDateFormat dateParser = new SimpleDateFormat("HH:mm");
        Date date = dateParser.parse(test.getTime());
        SimpleDateFormat dateFormatter = new SimpleDateFormat("HH:mm:ss");
        test.setTime(dateFormatter.format(date));
        if(test.getCountTry()==null) test.setCountTry(0);

        if (bindingResult.hasErrors()) {
            Map<String, String> errorsMap = ControllerUtils.getErrors(bindingResult);
            model.mergeAttributes(errorsMap);
            model.addAttribute("test", test);
            return "createTest";
        } else {
            System.out.println(cquesttype[0]+"asss");
            tr.createTest(test, cquesttype, cquest, canswer, canswer1, canswercor,
filename1, filename2, countPoint);
            model.addAttribute("test", null);
        }
        return "redirect:/main";
    }
}

// Контролер UserControler
package com.yanyshivskyi.yMathTest.controller;
import com.yanyshivskyi.yMathTest.domain.Role;
import com.yanyshivskyi.yMathTest.domain.User;
import com.yanyshivskyi.yMathTest.repos.UserRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.util.Arrays;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;

@Controller
@RequestMapping("/user")
@PreAuthorize("hasAuthority('ADMIN')")
public class UserController {
    @Autowired
    private UserRepo userRepo;

    @GetMapping
    public String userList(Model model) {
        model.addAttribute("users", userRepo.findAll());
        return "userList";
    }
```

```
    @GetMapping("{user}")
    public String userEditForm(@PathVariable User user, Model model) {
        model.addAttribute("user", user);
        model.addAttribute("roles", Role.values());

        return "userEdit";
    }

    @PostMapping
    public String userSave(
            @RequestParam String username,
            @RequestParam Map<String, String> form,
            @RequestParam(value = "us_id") User user
    ) {
        user.setUsername(username);
        Set<String> roles = Arrays.stream(Role.values())
                .map(Role::name)
                .collect(Collectors.toSet());

        user.getRoles().clear();

        for (String key : form.keySet()) {
            if (roles.contains(key)) {
                user.getRoles().add(Role.valueOf(key));
            }
        }

        userRepo.save(user);

        return "redirect:/user";
    }
}

// Таблиця  Answer
package com.yanyshivskyi.yMathTest.domain;

import javax.persistence.*;

@Entity // This tells Hibernate to make a table out of this class
public class Answer {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Long id;

    public Answer() {
    }

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name="id_question")
    private Question question;

    @Column(name="answer_text")
    private String answerText;
    private Boolean isCorrect;
    private String conformity;

    public Answer(Question question, String answerText, Boolean isCorrect, String
conformity) {
        this.question = question;
        this.answerText = answerText;
        this.isCorrect = isCorrect;
        this.conformity = conformity;
    }
```

```
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public Question getId_question() {
        return question;
    }

    public void setId_question(Question question) {
        this.question = question;
    }

    public String getAnswerText() {
        return answerText;
    }

    public void setAnswerText(String answerText) {
        this.answerText = answerText;
    }

    public String getCorrect() {
        if(isCorrect) return "1";
        else return "0";
    }

    public void setCorrect(Boolean correct) {
        isCorrect = correct;
    }

    public String getConformity() {
        return conformity;
    }

    public void setConformity(String conformity) {
        this.conformity = conformity;
    }
}

// Таблиця  Question

package com.yanyshivskyi.yMathTest.domain;

        import javax.persistence.*;

@Entity // This tells Hibernate to make a table out of this class
public class Question {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Long id;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name="id_test")
    private Test idTest;

    @Column(name = "question_text")
    private String questionText;
    private String filename1;
    private String filename2;

    @Column(name = "count_point")
```

```
private Float countPoint;

private String type;

public Question() {

}
public Question(Test idTest, String questionText,
                String filename1, String filename2, Float count_point, String type) {
    this.idTest = idTest;
    this.questionText = questionText;
    this.filename1 = filename1;
    this.filename2 = filename2;
    this.countPoint = count_point;
    this.type = type;
}

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public Test getIdTest() {
    return idTest;
}

public void setIdTest(Test idTest) {
    this.idTest = idTest;
}

public String getQuestionText() {
    return questionText;
}

public void setQuestionText(String questionText) {
    this.questionText = questionText;
}

public String getFilename1() {
    return filename1;
}

public void setFilename1(String filename1) {
    this.filename1 = filename1;
}

public String getFilename2() {
    return filename2;
}

public void setFilename2(String filename2) {
    this.filename2 = filename2;
}

public Float getCountPoint() {
    return countPoint;
}

public void setCountPoint(Float countPoint) {
    this.countPoint = countPoint;
}
```

```
    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }
}

// Таблиця  Result

package com.yanyshivskyi.yMathTest.domain;

import javax.persistence.*;
import java.util.Date;

@Entity // This tells Hibernate to make a table out of this class
public class Result {
    @Id
    @GeneratedValue(strategy= GenerationType.AUTO)
    private Long id;

    public Result() {
    }

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name="id_user")
    private User user;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public Test getTest() {
        return test;
    }

    public void setTest(Test test) {
        this.test = test;
    }

    public Float getPoint() {
        return point;
    }

    public void setPoint(Float point) {
        this.point = point;
    }

    public Date getMyDate() {
        return myDate;
```

```
    }

    public void setMyDate(Date myDate) {
        this.myDate = myDate;
    }

    public Integer getNumberTry() {
        return numberTry;
    }

    public void setNumberTry(Integer numberTry) {
        this.numberTry = numberTry;
    }

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name="id_test")
    private Test test;
    private Float point;

    @Temporal(TemporalType.TIMESTAMP)
    private java.util.Date myDate;

    @Column(name="number_try")
    private Integer numberTry;

    public Result(User user, Test test, Float point, Date myDate, Integer numberTry) {
        this.user = user;
        this.test = test;
        this.point = point;
        this.myDate = myDate;
        this.numberTry = numberTry;
    }

// Перечислення Role
package com.yanyshivskyi.yMathTest.domain;

import org.springframework.security.core.GrantedAuthority;

public enum Role implements GrantedAuthority {
    USER, ADMIN;

    @Override
    public String getAuthority() {
        return name();
    }
}

// Таблиця Test
package com.yanyshivskyi.yMathTest.domain;

import javax.persistence.*;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotBlank;

@Entity
// This tells Hibernate to make a table out of this class
public class Test {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Long id;

    public Test() {

    }
```

```
    @NotBlank(message="Назва теста не може бути пустою!")
    private String name;
    private String description;
    private String time;

    @Min(value = 0, message = "Мінімальне значення - 0")
    @Column (name="count_try")
    private Integer countTry;

    public Test(String name, String description, String time, Integer count_try) {
        this.name = name;
        this.description = description;
        this.time = time;
        this.countTry = count_try;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getTime() {
        return time;
    }

    public void setTime(String time) {
        this.time = time;
    }

    public Integer getCountTry() {
        return countTry;
    }

    public void setCountTry(Integer countTry) {
        this.countTry = countTry;
    }
}

// Таблиця User
package com.yanyshivskyi.yMathTest.domain;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
```

```
import javax.persistence.*;
import java.util.Collection;
import java.util.Set;

@Entity
@Table(name = "usr")
public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String username;
    private String password;
    private String groupst;
    private String stfio;
    private boolean active;


    public String getStfio() {
        return stfio;
    }

    public void setStfio(String stfio) {
        this.stfio = stfio;
    }



    public String getGroupst() {
        return groupst;
    }

    public void setGroupst(String groupst) {
        this.groupst = groupst;
    }


    @ElementCollection(targetClass = Role.class, fetch = FetchType.EAGER)
    @CollectionTable(name = "user_role", joinColumns = @JoinColumn (name= "user_id"))
    @Enumerated(EnumType.STRING)
    private Set<Role> roles;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public boolean isAdmin() {
        return roles.contains(Role.ADMIN);
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
```

```
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return isActive();
    }

    public void setUsername(String username) {
        this.username = username;
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return getRoles();
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public boolean isActive() {
        return active;
    }

    public void setActive(boolean active) {
        this.active = active;
    }

    public Set<Role> getRoles() {
        return roles;
    }

    public void setRoles(Set<Role> roles) {
        this.roles = roles;
    }
}

// Таблиця User
package com.yanyshivskyi.yMathTest.domain;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import javax.persistence.*;
import java.util.Collection;
import java.util.Set;

@Entity
@Table(name = "usr")
public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String username;
```

```
private String password;
private String groupst;
private String stfio;
private boolean active;


public String getStfio() {
    return stfio;
}

public void setStfio(String stfio) {
    this.stfio = stfio;
}



public String getGroupst() {
    return groupst;
}

public void setGroupst(String groupst) {
    this.groupst = groupst;
}


@ElementCollection(targetClass = Role.class, fetch = FetchType.EAGER)
@CollectionTable(name = "user_role", joinColumns = @JoinColumn (name= "user_id"))
@Enumerated(EnumType.STRING)
private Set<Role> roles;

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getUsername() {
    return username;
}

public boolean isAdmin() {
    return roles.contains(Role.ADMIN);
}

@Override
public boolean isAccountNonExpired() {
    return true;
}

@Override
public boolean isAccountNonLocked() {
    return true;
}

@Override
public boolean isCredentialsNonExpired() {
    return true;
}

@Override
public boolean isEnabled() {
    return isActive();
```

```
    }

    public void setUsername(String username) {
        this.username = username;
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return getRoles();
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public boolean isActive() {
        return active;
    }

    public void setActive(boolean active) {
        this.active = active;
    }

    public Set<Role> getRoles() {
        return roles;
    }

    public void setRoles(Set<Role> roles) {
        this.roles = roles;
    }
}
```

```
// Репозиторій AnswerRepo
package com.yanyshivskyi.yMathTest.repos;

import com.yanyshivskyi.yMathTest.domain.Answer;
import com.yanyshivskyi.yMathTest.domain.Question;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface AnswerRepo extends JpaRepository<Answer, Long> {
    Answer findByAnswerText(String text);
    List<Answer> findByQuestion(Question m);
}
```

```
// Репозиторій QuestionRepo
package com.yanyshivskyi.yMathTest.repos;

import com.yanyshivskyi.yMathTest.domain.Question;
import com.yanyshivskyi.yMathTest.domain.Test;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;
import java.util.Optional;
```

```
public interface QuestionRepo extends JpaRepository<Question, Long> {
        Question findByQuestionText(String text);

        List<Question> findByIdTest(Optional<Test> all);
}


// Репозиторій ResultRepo
package com.yanyshivskyi.yMathTest.repos;

import com.yanyshivskyi.yMathTest.domain.Result;
import com.yanyshivskyi.yMathTest.domain.Test;
import com.yanyshivskyi.yMathTest.domain.User;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;
import java.util.Optional;

public interface ResultRepo extends JpaRepository<Result, Long> {
    Optional<Result> findByTestAndUser(Test test, User user);

    List<Result> findByTestInOrderByIdDesc(List<Test> ts);
    List<Result> findByUserInOrderByIdDesc(List<User> ts);
    List<Result> findByTestInAndUserInOrderByIdDesc(List<Test> ts, List<User> us);
    List<Result> findAllByOrderByIdDesc();
}

// Репозиторій TestRepo
package com.yanyshivskyi.yMathTest.repos;

import com.yanyshivskyi.yMathTest.domain.Test;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;


public interface TestRepo extends JpaRepository<Test, Long> {
    Page<Test> findAll(Pageable pageable);
    List<Test> findByNameContainingIgnoreCase(String name);
    Page<Test> findByNameContainingIgnoreCase(String name, Pageable pageable);
}

// Репозиторій UserRepo
package com.yanyshivskyi.yMathTest.repos;

import com.yanyshivskyi.yMathTest.domain.User;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface UserRepo extends JpaRepository<User, Long> {
    User findByUsername(String username);
    List<User> findByUsernameContainingIgnoreCase(String username);
    List<User> findByGroupstContainingIgnoreCase(String groupSt);
    List<User> findByUsernameContainingIgnoreCaseAndGroupstContainingIgnoreCase
(String username, String groupSt);
}

// Сервіс CompleteTestService
package com.yanyshivskyi.yMathTest.service;

import com.yanyshivskyi.yMathTest.domain.*;
```

```
import com.yanyshivskyi.yMathTest.repos.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;

import java.io.Serializable;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import java.util.Optional;

@Service
public class CompleteTestService implements Serializable {
    @Autowired
    private TestRepo testRepo;

    @Autowired
    private QuestionRepo questr;

    @Autowired
    private AnswerRepo answerr;

    @Autowired
    private ResultRepo resultr;

    @Autowired
    private UserRepo userr;


    public List<Test> findAll() {
        return testRepo.findAll(Sort.by(Sort.Direction.DESC, "id"));
    }

    public int countQuestion(Long id) {
        int count=0;
        Optional<Test> byId = testRepo.findById(id);
        List<Question> allquest = questr.findAll();
        for(Question i: allquest){
            if(i.getIdTest()==byId.get()) count++;
        }
        return count;
    }


    public List<Question> findQuests(Long id) {
        Optional<Test> all = testRepo.findById(id);

        List<Question> list_quest= questr.findByIdTest(all);

        return list_quest;
    }

    public List<Answer> findAnswer(Question qst) {
        return answerr.findByQuestion(qst);
    }


    public void saveResult(Long id_test, String mystr, Float fpoint) {
        Date date=new Date();
        SimpleDateFormat sformat = new SimpleDateFormat("yyyy.MM.dd hh:mm:ss");
        sformat.format(date);

        Optional<Result> prevRes=resultr.findByTestAndUser(testRepo.findById(id_test).get(),
userr.findByUsername(mystr));
```

```
        int countTry=0;
        if(prevRes.isPresent()) {
            countTry=prevRes.get().getNumberTry()-1;
            if(prevRes.get().getPoint() > fpoint) {
                fpoint=prevRes.get().getPoint();
            }
            if(prevRes.get().getNumberTry()==0) return;
            resultr.delete(prevRes.get());
        }
        else {
            countTry=testRepo.findById(id_test).get().getCountTry()-1;
        }


        Result a=new Result(userr.findByUsername(mystr), testRepo.findById(id_test).get(),
fpoint, date, countTry);
        resultr.save(a);
    }

    public List<Result> findResults() {
       return resultr.findAllByOrderByIdDesc();
    }

    public float getPoint(Test test) {
        float point= 0;
        List<Question> allquest = questr.findAll();
        for(Question i: allquest){
            if(i.getIdTest()==test) {
                point+=i.getCountPoint();
            }
        }
        return point;
    }

    public List<Result> findResultsT(String filter1) {
        List<Test> ts=testRepo.findByNameContainingIgnoreCase(filter1);
        return resultr.findByTestInOrderByIdDesc(ts);
    }

    public List<Result> findResultsU(String filter2) {
        List<User> us=userr.findByUsernameContainingIgnoreCase(filter2);
        return resultr.findByUserInOrderByIdDesc(us);
    }


    public List<Result> findResultsG(String filter3) {
        List<User> us=userr.findByGroupstContainingIgnoreCase(filter3);
        return resultr.findByUserInOrderByIdDesc(us);
    }


    public List<Result> findResultsTU(String filter1, String filter2) {
        List<Test> ts=testRepo.findByNameContainingIgnoreCase(filter1);
        List<User> us=userr.findByUsernameContainingIgnoreCase(filter2);
        return resultr.findByTestInAndUserInOrderByIdDesc(ts, us);
    }

    public List<Result> findResultsTG(String filter1, String filter3) {
        List<Test> ts=testRepo.findByNameContainingIgnoreCase(filter1);
        List<User> us=userr.findByGroupstContainingIgnoreCase(filter3);
        return resultr.findByTestInAndUserInOrderByIdDesc(ts, us);
    }

    public List<Result> findResultsUG(String filter2, String filter3) {
```

```
        List<User>
us=userr.findByUsernameContainingIgnoreCaseAndGroupstContainingIgnoreCase(filter2, filter3);
        return resultr.findByUserInOrderByIdDesc(us);
    }

    public List<Result> findResultsTUG(String filter1, String filter2, String filter3) {
        List<Test> ts=testRepo.findByNameContainingIgnoreCase(filter1);
        List<User>
us=userr.findByUsernameContainingIgnoreCaseAndGroupstContainingIgnoreCase(filter2, filter3);
        return resultr.findByTestInAndUserInOrderByIdDesc(ts, us);
    }
}

// Сервіс TestService
package com.yanyshivskyi.yMathTest.service;

import com.yanyshivskyi.yMathTest.domain.*;
import com.yanyshivskyi.yMathTest.repos.AnswerRepo;
import com.yanyshivskyi.yMathTest.repos.QuestionRepo;
import com.yanyshivskyi.yMathTest.repos.ResultRepo;
import com.yanyshivskyi.yMathTest.repos.TestRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

import java.io.File;
import java.io.IOException;
import java.text.ParseException;
import java.util.List;
import java.util.Optional;
import java.util.UUID;

@Service
public class TestService {
    @Autowired
    private TestRepo testRepo;

    @Autowired
    private QuestionRepo questr;

    @Autowired
    private AnswerRepo answerr;

    @Autowired
    private ResultRepo resr;

    @Value("${upload.path}")
    private String uploadPath;


    public Page<Test> findAll(Pageable pageable) {
        return testRepo.findAll(pageable);
    }

    public int countQuestion(Long id) {
        int count=0;
        Optional<Test> byId = testRepo.findById(id);
        List<Question> allquest = questr.findAll();
        for(Question i: allquest){
            if(i.getIdTest()==byId.get()) count++;
        }
```

```
        return count;
    }

    public void createTest(Test test,
                           String[] cquesttype, String[] cquest, String[] canswer, String
canswer1,
                           String[] canswercor, MultipartFile[] filename1,MultipartFile[]
filename2,
                           Float[] countPoint) throws ParseException, IOException {


        testRepo.save(test);
        System.out.println("QWERTY" + cquesttype[0]);
        int j=0; // счетчик 1 правильный ответ
        int curj=0; //текущий правильный ответ
        int questTypeTrue=0; //номер вопроса с 1м правильным ответом
        int idTrueQuest=0; //множественный варианты счетчик
        Answer answ;
        Question quest;
        String resultFileName1="";
        String resultFileName2="";

        for(int i=0; i<cquesttype.length;i++) {
            if (filename1[i] !=null && !filename1[i].getOriginalFilename().isEmpty()) {
                File uploadDir = new File(uploadPath);
                if (!uploadDir.exists()) {
                    uploadDir.mkdir();
                }

                String uuidFile = UUID.randomUUID().toString();
                resultFileName1 = uuidFile + "." + filename1[i].getOriginalFilename();
                filename1[i].transferTo(new File(uploadPath + "/" + resultFileName1));
            }

                if (filename2[i] !=null && !filename2[i].getOriginalFilename().isEmpty()) {
                    File uploadDir2 = new File(uploadPath);
                    if (!uploadDir2.exists()) {
                        uploadDir2.mkdir();
                    }

                    String uuidFile2 = UUID.randomUUID().toString();
                    resultFileName2 = uuidFile2 + "." + filename2[i].getOriginalFilename();
                    filename2[i].transferTo(new File(uploadPath + "/" + resultFileName2));
                }

            if(countPoint[i]<0 ||countPoint[i].isNaN() || countPoint[i]==null)
countPoint[i]= Float.valueOf(1);

            quest = new Question(test, cquest[i], resultFileName1, resultFileName2,
countPoint[i], cquesttype[i]);
            questr.save(quest);
            if (cquesttype[i].equals("0")) {
                curj = 0;
                System.out.println("q");
                while (!canswer[j].equals("**/**")) {
                    System.out.println("w");
                    if (canswer1.charAt(questTypeTrue)-48 == curj) {
                        answ = new Answer(quest, canswer[j], true, null);
                        System.out.println("trr");
                    } else {
                        answ = new Answer(quest, canswer[j], false, null);
                    }
                    answerr.save(answ);
                    j++;
```

```
                    curj++;
                }
                j++;
                questTypeTrue++;
            }

            if (cquesttype[i].equals("1")) {
                curj=0;

                while (!canswer[j].equals("**/**")) {
                    if (canswercor[idTrueQuest].charAt(0)-48 == curj) {
                        answ = new Answer(quest, canswer[j], true, null);
                        idTrueQuest++;
                    } else {
                        answ = new Answer(quest, canswer[j], false, null);
                    }
                    answerr.save(answ);
                    j++;
                    curj++;
                }
                j++;
            }

            if (cquesttype[i].equals("2")) {
                answ = new Answer(quest, canswer[j], true, null);
                answerr.save(answ);
                j += 2; // скипаем **/**
            }

            if (cquesttype[i].equals("3")) {
                while (!canswer[j].equals("**/**")) {
                    answ = new Answer(quest, canswer[j], true, canswer[j + 1]);
                    answerr.save(answ);
                    j += 2; // скипаем **/**
                }
                j++; //last
            }
            if(canswercor[idTrueQuest].equals("**/**")) idTrueQuest++;
        }

    }

    public Page<Test> findTest(String filter, Pageable pageable) {
        return testRepo.findByNameContainingIgnoreCase(filter, pageable);
    }

    public int countTry(Test test, User user) {
        int count=0;
        if (test.getCountTry()==0) return -1;
        Optional<Result> rs = resr.findByTestAndUser(test, user);
        if(rs.isPresent()) {
            count = rs.get().getNumberTry();
        }
        else count = test.getCountTry();
        return count;
    }
}

// Сервіс UserService
package com.yanyshivskyi.yMathTest.service;

import com.yanyshivskyi.yMathTest.domain.Role;
import com.yanyshivskyi.yMathTest.domain.User;
import com.yanyshivskyi.yMathTest.repos.UserRepo;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

import java.util.Collections;

@Service
public class UserService implements UserDetailsService {
    @Autowired
    private UserRepo userRepo;

    @Autowired
    private PasswordEncoder passwordEncoder;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException
{
        return userRepo.findByUsername(username);
    }

    public User findByUsername(String username) {
        return userRepo.findByUsername(username);
    }

    public void save(User user) {
        user.setActive(true);
        user.setRoles(Collections.singleton(Role.USER));
        user.setGroupst(user.getGroupst());
        user.setStfio(user.getStfio());
        user.setPassword(passwordEncoder.encode(user.getPassword()));
        userRepo.save(user);
    }
}

//Файл міграції V1_init_db.sql
create table hibernate_sequence (next_val bigint primary key) engine=MyISAM;

insert into hibernate_sequence values ( 1 );

create table answer (
    id bigint not null,
    answer_text varchar(255),
    conformity varchar(255),
    is_correct bit,
    id_question bigint,
    primary key (id))  engine=MyISAM;

create table question (
    id bigint not null,
    count_point float, filename1 varchar(255),
    filename2 varchar(255),
    question_text varchar(255),
    type varchar(255),
    id_test bigint,
    primary key (id)) engine=MyISAM;

create table result (
    id bigint not null,
    my_date datetime,
    number_try integer,
    point float,
```

```
    id_test bigint,
    id_user bigint,
    primary key (id)) engine=MyISAM;

create table test (
    id bigint not null,
    count_try integer,
    description varchar(255),
    name varchar(255),
    time varchar(255),
    primary key (id)) engine=MyISAM;

create table user_role (
    user_id bigint not null,
    roles varchar(255)) engine=MyISAM;

create table usr (
    id bigint not null,
    active bit not null,
    password varchar(255),
    username varchar(255),
    groupst varchar(255),
    stfio varchar(255),
    primary key (id)) engine=MyISAM;

alter table answer add constraint id_quest_fk foreign key (id_question) references question
(id);

alter table question add constraint id_test_fk foreign key (id_test) references test (id);

alter table result add constraint id_test_fk foreign key (id_test) references test (id);

alter table result add constraint id_user_fk foreign key (id_user) references usr (id);

alter table user_role add constraint user_id_fk foreign key (user_id) references usr (id);

// Файл міграції V2_add_admin.sql

insert into usr(id, username, password,groupst, stfio, active) values (1, 'admin',
'$2a$08$pKCfk3/wB2mhPTCRk4OsI.x6U.FCkKhQRudTlTBEo3YkXbG4r4/Be','-','-',true);

insert into user_role(user_id, roles) values (1, 'USER'),
(1, 'ADMIN');

// Скрипт complete.js
$('#endtest').on('submit', function(e){
        var curanswer=1;
        var countquest=0;
        var a='';
        var b='';
        var curtype="0";
        var pointType1=0;
        var countTrueType1=0;
        var point = 0.0;
        var fullpoint=0.0;
        var koefPoint=1.0;
        var formatpoint=0.0;
        for (var i=0;i<1000;i++){
            a=$(`#correct_${i}_0`).val();
            if(typeof a =='undefined') break;
            curtype=$(`#typeq_${i}`).val();
            koefPoint=$(`#point_${i}`).val();
            if(curtype=="1") {
                pointType1=0;
                countTrueType1=0;
```

```
            }
            point=0.0;
            countquest++;

            for (var j=0; j<1000; j++){
                   b=$(`#correct_${i}_${j}`).val();
                   if(typeof b=='undefined')  break;

                   switch (curtype) {
                        case "0" : {
                            if(b=="1") {
                            $(`#text0ans-${i}-${j}`).css('color', 'green');
                             //   alert("Перекрасить в зеленый: " + i + j);
//input[name="rg-${i}"]
                            }
                            if($(`input[name="rg-${i}"]:checked`).val() == j && b=="1") {
                                point++;
                            }
                            if($(`input[name="rg-${i}"]:checked`).val() == j && b=="0") {
                                $(`#text0ans-${i}-${j}`).css('color', 'red');
                            }
                            break;
                        }

                        case "1" :{
                            if(b==1) countTrueType1++;

                            if($(`#c1h-${i}-${j}`).prop('checked') && b==1) {
                               pointType1++;
                              $(`#tc1h-${i}-${j}`).css('color', 'green');
                            }

                            if(!$(`#c1h-${i}-${j}`).prop('checked') && b==1) {
                              $(`#tc1h-${i}-${j}`).css('color', 'blue');
                            }

                            if($(`#c1h-${i}-${j}`).prop('checked') && b==0) {
                                $(`#tc1h-${i}-${j}`).css('color', 'red');
                                countTrueType1++;
                            }
                            break;
                        }

                        case "2" :{
                            if($(`#onl-${i}`).val() == b){
                                point++;
                                $(`#onl-${i}`).css('color', 'green');
                            }
                            else {
                            $(`#onl-${i}`).css('color', 'red');
                            ///    alert("Перекрасить в красный + вывести справа b(лучше
снизу)" + i + j + b);
                            }
                            break;
                        }

                   }

            }

            if(curtype==1) {
                point= pointType1/countTrueType1;
            }
            fullpoint+=point*koefPoint;
```

```
        }
       // alert(': '+fullpoint);
        fullpoint=fullpoint.toFixed(3);
        $(`#flpoint`).val(fullpoint);
        $(`#flpoint`).val(fullpoint);

        $(`#enterTest`).hide();
        let block = `
                        <div class="row my-3 justify-content-start ml-3">
                            <h5> Оцінка за тест: ${fullpoint} бал(бали)</h5>
                            <br>
                        </div>`;
         $(`#myclass2`).prepend(block);
    });

//Скрипт create.js
(() => {
            //скрытие невыбранных типов ответов
            $('#answerTest-1').hide();
        $('#answerTest-2').hide();
        $('#answerTest-3').hide();
        //номер текущего вопроса
        let curAsk = 0; //
        //выбранный тип ответов
        let curAnsType = 0; //
        //количество ответов
        let curAnsCount = 1;
        //нумерация ответов (сделано, чтобы не было пролем с ид-инпутов при удалении и
добавлении)
        let curAnsNumb = 0;
        //общее к-во вопросов
        let allAskCount = 1;
        //класс, ответственный за запоминание общих данных вопросов (для перехода по
вкладкам)


        function Ask() {
            let askArr = [];
            this.pushAsk = function (id, type = 0, ansCount = 1, ansNum=0) {
                //если записан такой вопрос
                if(askArr[id]){
                    //обновляем данные
                    askArr[id].type = type;
                    askArr[id].ansCount = ansCount;
                    askArr[id].ansNum = ansNum;
                } else {
                    //создаём новый элемент
                    askArr.push({type: type, ansCount: ansCount, ansNum: ansNum});
                }
            };
            //получаем данные по впоросу с заданым номером
            this.getAsk = function (num) {
                return askArr[num];
            }
        }
        //объект для работы со списком вопросов
        askArr = new Ask();
        // создание нового блока для ответа
        const createAskBlock = function (typeAns, reset=false) {
            let block = '';
            if(!reset && typeAns!==2){
                curAnsNumb++;
            }
```

```
            switch (typeAns) {
                case 0: {
                    block = `
                <div class="row my-3 justify-content-start">
                    <label class="col col-1 col-form-label pl-5">
                          <input class="form-check-input" type="radio" id="r-${curAsk}-
${curAnsNumb}" name="rg-${curAsk}" value="${curAnsNumb}" ${!curAnsNumb && 'checked' } >

                    </label>
                    <div class="col col-4">
                        <input type="text" class="form-control" id="d1-${curAsk}-
${curAnsNumb}" name="canswer" placeholder="Введіть відповідь" required>
                    </div>
                    <div class="col col-1 ml-4">
                        <button type="button" class="btn btn-danger del-
ans">Видалити</button>
                    </div>
                </div>`;
                    break;
                }
                case 1: {
                    block = `
                <div class="row my-3 justify-content-start">
                    <label class="col col-1 col-form-label pl-5">
                        <input class="form-check-input" type="checkbox" name="canswercor"
id="ch-${curAsk}-${curAnsNumb}" value="${curAnsNumb}" ${curAnsNumb==0 ? 'checked' :''} >
                    </label>
                    <div class="col col-4">
                        <input type="text" class="form-control" id="d2-${curAsk}-
${curAnsNumb}" name="canswer" placeholder="Введіть відповідь" required>
                    </div>
                    <div class="col col-1 ml-4">
                        <button type="button" class="btn btn-danger del-
ans">Видалити</button>
                    </div>
                </div>`;
                    break;
                }
                case 2: {
                    block = `
                <div class="row my-3 justify-content-start">
                        <div class="col col-8">
                            <input type="text" class="form-control" id="onl-${curAsk}" name
="canswer" placeholder="Введіть відповідь" required>
                        </div>
                    </div>`;
                    break;
                }
                case 3: {
                    block = `<div class="row my-3 justify-content-start">
                        <div class="col col-4">
                            <input type="text" class="form-control" id="p1-${curAsk}-
${curAnsNumb}"  name ="canswer" placeholder="Введіть відповідь">
                        </div>
                        <div class="col col-4">
                            <input type="text" class="form-control" id="p2-${curAsk}-
${curAnsNumb}"  name ="canswer" placeholder="Введіть відповідь">
                        </div>
                        <div class="col col-1 ml-4">
                            <button type="button" class="btn btn-danger del-
ans">Видалити</button>
                        </div>
```

```
                </div>`;
                break;
            }

        }
        if(!reset && typeAns!==2){
            curAnsCount++;
        }
        return block;
    };


    // сброс блока ответов к начальному состоянию
    const resetAnsBlock = function (ansType) {
        $(`#ask-${curAsk} .ans-area-${ansType} .row`).remove();

    };
    // прослушка события изменения типа события
    $('.chg-area').on('change', '#typeAnswer', function () {
        $(`#ask-${curAsk} #answerTest-${curAnsType}`).hide();
        //сброс данных блока
        curAnsNumb = 0;
        resetAnsBlock(curAnsType);
        //Можно добавить проверку на наличие данных в блоке и вывести предупреждение,
что данные будут стерты
        curAnsType = $(this).find(":selected").index();
        let currentAnsBlock = $(`#ask-${curAsk} #answerTest-${curAnsType}`);
        $(`#ask-${curAsk} .ans-area-${curAnsType}`).append(createAskBlock(curAnsType,
true));
        currentAnsBlock.show();
        curAnsCount = currentAnsBlock.find('.row').length;


    });
    // прослушка события добавления ответа
    $('.chg-area').on('click', '.add-ans', function () {
        if (curAnsCount !== 10) {
            $(`#ask-${curAsk} .ans-area-
${curAnsType}`).append(createAskBlock(curAnsType));
        }
    });
    // прослушка события удаления ответа
    $('.chg-area').on('click', '.del-ans', function () {
        curAnsCount--;
        this.closest(".row").remove();
    });
    // прослушка события изменения вкладки вопроса
    $('.ask-btn-group').on('click', 'button', function () {
        if(curAsk === allAskCount-1){
            askArr.pushAsk(curAsk, curAnsType, curAnsCount, curAnsNumb)
        }
        let selId = +this.id.split('-')[1];
        if (curAsk !== selId) {
            $(`#btn-${curAsk}`).removeClass('btn-info');
            $(`#btn-${selId}`).addClass('btn-info');
            let ask = askArr.getAsk(selId);
            $(`fieldset#ask-${selId}`).show();
            $(`fieldset#ask-${curAsk}`).hide();
            curAsk = selId;
            curAnsType = ask.type;
            curAnsCount = ask.ansCount;
            curAnsNumb = ask.ansNum;
        }
    });
```

```
        // добавление вопроса
        $('#add-ask').on('click', function () {
            let nameInput = $('#testnametc');
            let typeInput = $('#typeAnswer');
            let prevAsk = curAsk;
            askArr.pushAsk(curAsk, curAnsType, curAnsCount, curAnsNumb);
            curAsk = allAskCount;
            allAskCount++;
            curAnsType = 0;
            curAnsCount = 0;
            curAnsNumb = 0;
            // create new button with ask number
            let btnPrev = $(`#btn-${prevAsk}`);
            let btnCopy = btnPrev.clone();
            btnCopy.text(curAsk + 1);
            btnCopy.attr('id', 'btn-'+curAsk);
            $('.ask-list').append(btnCopy);
            btnPrev.removeClass('btn-info');


            setTimeout(function (){
                                    let fieldset = $(`.chg-area #ask-${prevAsk}`).clone();
                                    fieldset.find('input').each(function() {
                                        this.name= this.name.replace('-', '_');
                                    });
                                    fieldset.attr('id', 'ask-' + curAsk);
                                    fieldset.appendTo('.chg-area');
                       $(`#ask-${curAsk} #testnametc`).val("");
                       $(`#ask-${curAsk} #countpp`).val("1");
                      }, 0);

            setTimeout(function () {
                        resetAnsBlock(0);
                        resetAnsBlock(1);
                        resetAnsBlock(2);
                        resetAnsBlock(3);
                        curAnsCount++;

                        let currentAnsBlock = $(`#ask-${curAsk} #answerTest-${curAnsType}`);
                        $(`#ask-${curAsk} .ans-area-
${curAnsType}`).append(createAskBlock(curAnsType, true));
                        currentAnsBlock.show();
                        curAnsCount = currentAnsBlock.find('.row').length;
             }, 8);

            setTimeout(function () {
                $(`#ask-${curAsk} #answerTest-1`).hide();
                $(`#ask-${curAsk} #answerTest-2`).hide();
                $(`#ask-${curAsk} #answerTest-3`).hide();
                $(`#ask-${curAsk} .ask-num`).text(curAsk + 1);
                $(`fieldset#ask-${prevAsk}`).hide();

             }, 4);

        });

        $( "#nameTest" ).submit(function( event ) {
          var str="";
          for(var i=0; i<allAskCount; i++){
          if($(`input[name=rg-${i}]:checked`).val()) str+=$(`input[name=rg-
${i}]:checked`).val();
          }
          if(str!=="") {let block = `<input type="hidden" name="canswer1" value="${str}"
/>`;
```

```
          $(`#nameTest`).append(block);
          }

        //  alert("Тест создан!");
        });
        })();

// Скрипт res.js
   $('#res').bind('click', function(){
     $(`#filter1`).val(null);
     $(`#filter2`).val(null);
     $(`#filter3`).val(null);
   });

// Шаблон common.ftl
   <#macro page>
<!DOCTYPE HTML>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>yTest</title>
    <link rel="stylesheet" href="/static/style.css">

    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
          integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
          crossorigin="anonymous">

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <script src =
'http://ajax.googleapis.com/ajax/libs/jquery/2.0.3/jquery.min.js'></script>

    <script type="text/javascript" async
src="http://cdn.mathjax.org/mathjax/latest/MathJax.js?config=TeX-AMS_HTML-full"></script>
    <script type="text/x-mathjax-config">
        MathJax.Hub.Config({
        tex2jax: {inlineMath: [['$','$'], ['\\(','\\)']]}
    });
    </script>

    <script defer src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
    <script defer
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous"></script>
    <script type='text/javascript' src='http://code.jquery.com/jquery-
latest.min.js'></script>



</head>
<body>
<#include "navbar.ftl">
<div class="container mt-5">
<#nested />
</div>
```

```
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous"></script>
</body>
</html>
</#macro>

// Шаблон isAdmin.ftl
   <#include "security.ftl">
<#macro adm>
    <#if isAdmin>
        <#nested />
    </#if>
</#macro>

// Шаблон login.ftl
<#macro login path>
<form action="${path}" method="post">
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Логін:</label>
        <div class="col-sm-6">
            <input type="text" name="username" class="form-control" placeholder="Логін" />
        </div>
    </div>
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Пароль:</label>
        <div class="col-sm-6">
            <input type="password" name="password" class="form-control" placeholder="Пароль"
/>
        </div>
    </div>
    <input type="hidden" name="_csrf" value="${_csrf.token}" />
<button class="btn btn-primary" type="submit">Вхід</button>
</form>
</#macro>

<#macro registration path>
<form action="${path}" method="post">
    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Логін:</label>
        <div class="col-sm-6">
            <input type="text" name="username" class="form-control" placeholder="Логін" />
        </div>
    </div>

    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Пароль:</label>
        <div class="col-sm-6">
            <input type="text" name="password" class="form-control" placeholder="Пароль" />
        </div>
    </div>

    <div class="form-group row">
        <label class="col-sm-2 col-form-label">ПІБ:</label>
        <div class="col-sm-6">
            <input type="text" name="stfio" class="form-control" placeholder="ПІБ:" />
        </div>
    </div>

    <div class="form-group row">
        <label class="col-sm-2 col-form-label">Група:</label>
        <div class="col-sm-6">
            <input type="text" name="groupst" class="form-control" placeholder="Група:" />
        </div>
```

```
    </div>

    <input type="hidden" name="_csrf" value="${_csrf.token}" />
    <button class="btn btn-primary" type="submit">Створити користувача</button>
</form>
</#macro>

<#macro logout>
<form action="/logout" method="post">
    <input type="hidden" name="_csrf" value="${_csrf.token}" />
    <button class="btn btn-primary" type="submit">Вихід</button>
</form>
</#macro>

<#macro sign>
<form action="/login" method="post">
    <input type="hidden" name="_csrf" value="${_csrf.token}" />
    <button class="btn btn-primary" type="submit">Авторизація</button>
</form>
</#macro>

// Шаблон navbar.ftl
<#include "security.ftl">
<#import "login.ftl" as l>
<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="/">yTest</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item">
                <a class="nav-link" href="/">Головна</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="/main">Тести</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="/result">Результати</a>
            </li>
            <#if isAdmin>
            <li class="nav-item">
                <a class="nav-link" href="/user">Список студентів</a>
            </li>
            </#if>
        </ul>

        <div class="navbar-text mr-3">${name}</div>
        <#if isSign>
            <@l.logout />
        <#else> <@l.sign />
        </#if>
    </div>



// Шаблон pager.ftl
<#macro pager url page>
        <#if page.getTotalPages() gt 7>
        <#assign
        totalPages = page.getTotalPages()
        pageNumber = page.getNumber() + 1
```

```
        head = (pageNumber > 4)?then([1, -1], [1, 2, 3])
        tail = (pageNumber < totalPages - 3)?then([-1, totalPages], [totalPages - 2,
totalPages - 1, totalPages])
        bodyBefore = (pageNumber > 4 && pageNumber < totalPages - 1)?then([pageNumber - 2,
pageNumber - 1], [])
        bodyAfter = (pageNumber > 2 && pageNumber < totalPages - 3)?then([pageNumber + 1,
pageNumber + 2], [])

        body = head + bodyBefore + (pageNumber > 3 && pageNumber < totalPages -
2)?then([pageNumber], []) + bodyAfter + tail
        >
        <#else>
        <#assign body = 1..page.getTotalPages()>
        </#if>
        <div class="mt-3">
            <ul class="pagination">
                <li class="page-item disabled">
                    <a class="page-link" href="#" tabindex="-1">Сторінки</a>
                </li>
                <#list body as p>
                <#if (p - 1) == page.getNumber()>
                <li class="page-item active">
                    <a class="page-link" href="#" tabindex="-1">${p}</a>
                </li>
                <#elseif p == -1>
                <li class="page-item disabled">
                    <a class="page-link" href="#" tabindex="-1">...</a>
                </li>
                <#else>
                <li class="page-item">
                    <a class="page-link" href="${url}?page=${p -
1}&size=${page.getSize()}&filter=${filter}" tabindex="-1">${p}</a>
                </li>
            </#if>
        </#list>
        </ul>

        <ul class="pagination">
            <li class="page-item disabled">
                <a class="page-link" href="#" tabindex="-1">Елементів на сторінці</a>
            </li>
            <#list [4, 10, 20, 40] as c>
            <#if c == page.getSize()>
            <li class="page-item active">
                <a class="page-link" href="#" tabindex="-1">${c}</a>
            </li>
            <#else>
            <li class="page-item">
                <a class="page-link"
href="${url}?page=${page.getNumber()}&size=${c}&filter=${filter}" tabindex="-1">${c}</a>
            </li>
        </#if>
        </#list>
        </ul>
        </div>
</#macro>

// Шаблон security.ftl
<#assign
    known = Session.SPRING_SECURITY_CONTEXT??
    >
<#if known>
    <#assign
```

```
        user = Session.SPRING_SECURITY_CONTEXT.authentication.principal
        name = user.getUsername()
        isAdmin = user.isAdmin()
        isSign=true
    >
<#else>
    <#assign
        name = "Гість"
        isSign=false
        isAdmin=false
    >
</#if>


// Сторінка createTest.ftl
<#import "parts/common.ftl" as c>

<@c.page>
<form id="nameTest" action="/create" role="form" enctype="multipart/form-data"
method="post">

    <a href="http://hijos.ru/nabor-formul-v-latex/">Введення формул в Latex</a>
    <h4>Тест</h4>

    <div class="row my-3 justify-content-start">
        <div class="col col-1 col-form-label">Назва:</div>

        <div class="col col-4">
            <input type="text" class="form-control ${(nameError??)?string('is-invalid',
'')}"
                   id="testname" name="name" placeholder="Введіть назву теста"
                   value="<#if test??>${test.name}</#if>">
            <#if nameError??>
            <div class="invalid-feedback">
                ${nameError}
            </div>
            </#if>
        </div>

        <div class="col col-2 col-form-label offset-md-1">Ліміт часу:</div>
        <div class="col col-2"><input type="time" value="00:15" min="00:00" max="03:00"
class="form-control" name="time" id="limit_time"></div>
        <div class="w-100 mb-3"></div>
        <div class="col col-1 col-form-label">Опис:</div>
        <div class="col col-4"><input type="text" class="form-control" id="description"
name="description"
                                      placeholder="Введіть опис теста"></div>
        <div class="col col-2 col-form-label offset-md-1">Кількість спроб:</div>
        <div class="col col-2">
            <input type="number" class="form-control ${(countTryError??)?string('is-
invalid', '')}" name="countTry" min="0" id="count_test" data-toggle="tooltip" data-
placement="top" title="0-Необмежена кількість спроб" required>
        </div>

        <div class="w-100 mb-5"></div>
    </div>

    <div class="btn-toolbar mb-5" role="toolbar" aria-label="Toolbar with button groups">
        <div class="btn-group mr-2 ask-list ask-btn-group" role="group" aria-label="First
group">
            <button type="button" class="btn btn-secondary btn-info" id="btn-0">1</button>
        </div>
        <button type="button" class="btn btn-info" id="add-ask">Додати питання</button>
    </div>
```

```
    <div class="chg-area">
        <fieldset id="ask-0">
            <h4>Питання №<b class="ask-num">1</b></h4>
            <div class="row my-3 justify-content-start form-group">
                <label class="control-label col-1" for="typeAnswer">Тип питання</label>
                <div class="col col-md-4">
                    <select id="typeAnswer" name="cquesttype" class="form-control">
                        <option value="0" selected>1 правильна відповідь</option>
                        <option value="1">Декілька правильних відповідей</option>
                        <option value="2">Введення відповіді з клавіатури</option>
                        <option value="3">Відповідність</option>
                    </select>
                </div>
                <div class="w-100 mb-2"></div>
                <div class="col col-1 col-form-label">Запитання:</div>
                <div class="col col-7"><input type="text" name="cquest" class="form-control"
id="testnametc" placeholder="Введіть запитання"></div>
                <div class="col col-1 ">Кількість балів:</div>
                <div class="col col-1"><input type="text" name="countPoint" class="form-
control" id="countpp" placeholder="Кількість балів" value="1"></div>
                <div class="w-100 mb-3"></div>
                <div class="col col-4" id="img1" ><input type="file" name="filename1"></div>
                <div class="w-100 mb-1"></div>
                <div class="col col-4" id="img2" ><input type="file" name="filename2"
id="file"></div>
            </div>

            <h4>Відповіді </h4>
            <div class="jumbotron w-75 py-5">
                <div id="answerTest-0" class="ans-area ans-area-0">
                    <div class="d-flex flex-row-reverse mr-5">
                        <button type="button" class="btn btn-success add-ans">+</button>
                    </div>

                    <div class="row my-3 justify-content-start">
                        <label class="col col-1 col-form-label pl-5">
                            <input class="form-check-input" type="radio" id="r-0-0"
name="rg-0" value="0" checked>
                        </label>
                        <div class="col col-4">
                            <input type="text" class="form-control" id="d1-0-0"
name="canswer" placeholder="Введіть відповідь" required>
                        </div>
                        <div class="col col-1 ml-4">
                            <button type="button" class="btn btn-danger del-
ans">Видалити</button>
                        </div>
                    </div>
                </div>

                <div id="answerTest-1" class="ans-area ans-area-1">
                    <div class="d-flex flex-row-reverse mr-5">
                        <button type="button" class="btn btn-success add-ans">+</button>
                    </div>
                </div>

                <div id="answerTest-2" class="ans-area ans-area-2">
                </div>

                <div id="answerTest-3" class="ans-area ans-area-3">
                    <div class="d-flex flex-row-reverse mr-5">
                        <button type="button" class="btn btn-success add-ans">+</button>
                    </div>
```

```
                </div>
            </div>
            <input type="hidden" name="canswer" value="**/**" />
            <input type="hidden" name="canswercor" value="**/**" />
        </fieldset>
    </div>

        <button class="btn btn-primary mb-3" type="submit">Зберегти тест</button>
    <input type="hidden" name="_csrf" value="${_csrf.token}"/>

</form>
<script src="/static/create.js"></script>
/@c.page

// Сторінка login.ftl
<#import "parts/common.ftl" as c>
<#import "parts/login.ftl" as l>

<@c.page>
Сторінка входу
<@l.login "/login"/>
/@c.page

// Сторінка main.ftl
<#import "parts/common.ftl" as c>
<#import "parts/isAdmin.ftl" as a>
<#import "parts/pager.ftl" as p>

<@c.page>

    <@a.adm>
        <a href="/create" class="btn btn-primary active mb-3" role="button" aria-
pressed="true">Створити тест</a>
    </@a.adm>
<br>
<form method="get" action="/main" class="form-inline">
    <input type="text" name="filter" class="form-control" value="${filter?ifExists}"
placeholder="Введіть назву">
    <button type="submit" class="btn btn-primary ml-2">Пошук</button>
</form>
<br>

<@p.pager url page/>

<div class="row my-3 justify-content-start">
<#list page.content as test>

<div class="col col-5">
    <div class="card w-100 mb-3">
        <div class="card-header">
            <h5>  ${test.name}</h5>
        </div>
        <div class="card-body">
            <div class="card-title mb-4">
                <i> ${test.description}</i>
            </div>
            <p class="card-text">Час проходження: ${test.time}</p>
            <p class="card-text">Кількість спроб: ${test.countTry}</p>
            <p class="card-text">Кількість питань: ${count_q[test?index]}</p>

            <#if (count_g[test?index]>0) >
            <p class="card-text">Спроб залишилося: ${count_g[test?index]}</p>
            <a href="/test/${test.id}" class="btn btn-outline-secondary">Перейти до
тесту</a>
```

```
            </#if>
                <#if count_g[test?index]<0>
                    <p class="card-text">Необмежена кількість спроб</p>
                    <a href="/test/${test.id}" class="btn btn-outline-secondary">Перейти до
тесту</a>
             </#if>
    <#if count_g[test?index]==0>
    <p class="card-text">Всі спроби вичерпано</p>
    </#if>
    </div>
</div>
</div>
<#else>
    Тестів не знайдено
</#list>
</div>
</@p.pager url page />

</@c.page>


// Сторінка registration.ftl
<#import "parts/common.ftl" as c>
<#import "parts/login.ftl" as l>

<@c.page>
    <div class="mb-1">Реєстрація студента</div>
    ${message?ifExists}
    <@l.registration "/registration" />
/@c.page

// Сторінка result.ftl
<#import "parts/common.ftl" as c>
<#import "parts/security.ftl" as sc>

<@c.page>
<br>

<#if sc.isAdmin>
<form method="get" action="/result" class="form-inline">
    <input type="text" id="filter1" name="filter1" class="form-control mr-2"
value="${filter1?ifExists}" placeholder="Введіть назву теста">
    <input type="text" id="filter2" name="filter2" class="form-control mr-2"
value="${filter2?ifExists}" placeholder="Введіть логін">
    <input type="text" id="filter3" name="filter3" class="form-control mr-2"
value="${filter3?ifExists}" placeholder="Введіть групу">
    <button type="submit" class="btn btn-primary ml-1">Пошук</button>
    <button id="res" class="btn btn-primary ml-1">Скинути фільтр</button>
</form>
<br>
</#if>

<table class="table table-bordered w-100">
    <thead>
    <tr>
        <th>Тест</th>
        <th>Логін</th>
        <th>Група</th>
        <th>Дата</th>
        <th>Кількість балів:</th>
        <th>Макс за тест:</th>
        <th>Оцінка(100б):</th>
    </tr>
    </thead>
```

```
    <tbody>
    <#list results as result>
    <#if sc.isAdmin==false && result.user.username==sc.name>
    <tr>
        <td>${result.test.name}</td>
        <td>${result.user.username}</td>
        <td>${result.user.groupst}</td>
        <td>${result.myDate}</td>
        <td>${result.point}</td>
        <td>${maxPoint[result?index]}</td>
        <td>${result.point/maxPoint[result?index]*100}</td>
    </tr>
    </#if>
    <#if sc.isAdmin>
    <tr>
        <td>${result.test.name}</td>
        <td>${result.user.username}</td>
        <td>${result.user.groupst}</td>
        <td>${result.myDate}</td>
        <td>${result.point}</td>
        <td>${maxPoint[result?index]}</td>
        <td>${result.point / maxPoint[result?index] *100 }</td>
    </tr>
    </#if>
<#else>
<td colspan="7">Жодного тесту не було знайдено</td>
</#list>
</tbody>
</table>
<script type="text/javascript" src="/static/res.js"></script>
</@c.page>

// Сторінка testId.ftl
<#import "parts/security.ftl" as sc>
<#import "parts/common.ftl" as c>


<@c.page>
    <div id="myclass2">
    <#list answers as answer>
        <b>Питання ${answer?index + 1}.</b> <em> ${quests[answer?index].countPoint}
бал(бали) </em>       ${quests[answer?index].questionText}
        <#if quests[answer?index].filename1!="">
        <img src="/img/${quests[answer?index].filename1}" class="rounded float-left"
alt="Адаптивные изображения" width="245px">
        </#if>
        <#if quests[answer?index].filename2!="">
        <img src="/img/${quests[answer?index].filename2}" id="main-img" class="rounded
float-rigth" alt="Адаптивные изображения" width="245px" ">
        </#if>
        <br>
        <input type="hidden" class="myclass1" form="endtest" id="typeq_${answer?index}"
value="${quests[answer?index].getType()}" />
        <input type="hidden" class="myclass1" form="endtest" id="point_${answer?index}"
value="${quests[answer?index].countPoint}" />
            <#list answer as ans>


                <#if quests[answer?index].type=="0">
                    <input type="hidden" class="myclass1" form="endtest"
id="correct_${answer?index}_${ans?index}" value="${ans.getCorrect()}" />
                    <label class="col col-1 col-form-label pl-5" id="text0ans-
${answer?index}-${ans?index}">
```

```
                        <input class="form-check-input" type="radio" id="r-${answer?index}-
${ans?index}" name="rg-${answer?index}" value="${ans?index}">
                        ${ans.answerText}
                    </label>
                    <br>
                </#if>
                <#if quests[answer?index].type=="1">
<input type="hidden" class="myclass1" form="endtest"
id="correct_${answer?index}_${ans?index}" value="${ans.getCorrect()}" />
                    <label class="col col-1 col-form-label pl-5" id="tc1h-${answer?index}-
${ans?index}">
                        <input class="form-check-input" type="checkbox" id="c1h-
${answer?index}-${ans?index}" name="chh-${answer?index}" value="${answer?index}"} >
                        ${ans.answerText}
                    </label>
                <br>
                </#if>

                <#if quests[answer?index].type=="2">
<input type="hidden" class="myclass1" form="endtest" id="correct_${answer?index}_${0}"
value="${ans.answerText}" />
                    <div class="col col-8 mt-3">
                        <input type="text" value="" class="form-control" id="onl-
${answer?index}" name ="canswer" placeholder="Введіть відповідь">
                    </div>
                <br>
                </#if>
            <#else>
                Вопросов нет!
            </#list>
    <br>
    <#else>
        Тест пуст
    </#list>

<label id="ff" value="">
</label>



<form id="endtest" action="/test/{id}" role="form" enctype="multipart/form-data"
target="iframe1" method="post">
    <input type="hidden" name="_csrf" value="${_csrf.token}"/>
    <input type="hidden" name="mystr" id="mystrr" value="${sc.name}"/>
    <input type="hidden" name="fpoint" id="flpoint" value=""/>
    <nav class="navbar">
        <button class="btn btn-primary mb-3" id="enterTest" type="submit">Завершити
тест</button>
    </nav>


</form>

<iframe name="iframe1" style="position: absolute; display:none"></iframe>
</div>
<script type="text/javascript" src="/static/complete.js"></script>
<script type="text/javascript" src="/static/jj.js"></script>
</@c.page>


// Сторінка userEdit.ftl
<#import "parts/common.ftl" as c>
```

```
<@c.page>
Редагування прав користувача

<form action="/user" method="post">
    <input type="text" name="username" value="${user.username}" readonly>
    <#list roles as role>
    <div>
        <label><input type="checkbox" name="${role}"
${user.roles?seq_contains(role)?string("checked", "")}>${role}</label>
    </div>
</#list>
<input type="hidden" value="${user.id}" name="us_id">
<input type="hidden" value="${_csrf.token}" name="_csrf">
<button class="btn btn-primary" type="submit">Зберегти</button>
</form>
/@c.page

// Сторінка userList.ftl
<#import "parts/common.ftl" as c>

<@c.page>
<div class ="mb-3">
<h5> <a href="/registration" class="badge badge-primary">Зареєструвати студента</a> </h5>
</div>

Список зареєстрованих користувачів:

<table class="table table-bordered w-75">
    <thead>
    <tr>
        <th>Логін</th>
        <th>ПІБ</th>
        <th>Група</th>
        <th>Ролі</th>
        <th></th>
    </tr>
    </thead>
    <tbody>
    <#list users as user>
    <#if user.username!='admin'>
    <tr>
        <td>${user.username}</td>
        <td>${user.stfio}</td>
        <td>${user.groupst}</td>
        <td><#list user.roles as role>${role}<#sep>, </#list></td>
        <td><a href="/user/${user.id}">Редагувати</a></td>
    </tr>
    </#if>
    </#list>
    </tbody>
</table>
</@c.page>

//Налаштування application.properties

spring.datasource.url=jdbc:mysql://localhost:3306/ytest_db?useUnicode=true&useJDBCCompliantT
imezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC&allowPublicKeyRetrieval=tru
e
spring.datasource.username=root
spring.datasource.password=ytest123
spring.jpa.generate-ddl=true

spring.flyway.baseline-on-migrate=true

spring.freemarker.expose-request-attributes=true
```

```
upload.path= /D:/Programming/upload_file
spring.session.jdbc.initialize-schema=always
spring.session.jdbc.table-name=SPRING_SESSION

spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=validate
```

Додаток Б

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
ПЗ Янишівський В.В.docx	Пояснювальна записка роботи. Документ Word.
ПЗ Янишівський В.В.pdf	Пояснювальна записка роботи. Документ PDF.
Програма	
YTest.zip	Архів. Містить код програми.
Презентація	
Презентація Янишівський.pptx	Презентація магістерської роботи.