Полтавський університет економіки і торгівлі
Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп’ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____Олена ОЛЬХОВСЬКА
(підпис)
«_____»_____2026 р.

# КВАЛІФІКАЦІЙНА РОБОТА

**на тему**
**«РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ТЕСТУВАННЯ СТУДЕНТІВ З ДИСЦИПЛІНИ «ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ»**

**зі спеціальності 122 «Комп’ютерні науки»**
**освітня програма «Комп’ютерні науки»**
**ступеня бакалавра**

**Виконавець роботи** Шкицький Валентин Володимирович
_____ «__.__.» __._____2026 р.
(підпис)

**Науковий керівник** к.ф.-м.н., доц. Олексійчук Юрій Федорович
_____ «__.__.» __._____2026 р.
(підпис)

**Рецензент**

**ПОЛТАВА 2026**

# РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи - 46 сторінок, 24 рисунка, 1 таблиця., 1 додаток., 23 джерел.

ТЕСТУВАННЯ СТУДЕНТІВ, БАЗА ДАНИХ, ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, ВЕБ-ЗАСТОСУНОК, JAVA, SPRING BOOT, MYSQL, MVC.

**Об'єкт дослідження** – процес оцінювання та контролю знань студентів з навчальних дисциплін.

**Предмет дослідження** – програмне забезпечення для автоматизації процесу тестування студентів з дисципліни «Об'єктно-орієнтоване програмування».

**Мета роботи** – розробка функціонального веб-застосунку для проведення тестування студентів, управління тестовими завданнями та зручного аналізу результатів викладачами.

**Методи дослідження** – об'єктно-орієнтований аналіз та проєктування, методи реляційного моделювання баз даних, архітектурний шаблон проєктування MVC.

У кваліфікаційній роботі проведено аналіз предметної області та обґрунтовано доцільність створення спеціалізованої системи для тестування знань з ООП. Розроблено клієнт-серверний веб-застосунок, що забезпечує автоматизацію процесу перевірки знань.

Серверна частина реалізована мовою програмування Java з використанням фреймворку Spring Boot. Для збереження даних (користувачів, тестів, питань та результатів) розроблено структуру реляційної бази даних під керуванням СКБД MySQL, взаємодія з якою здійснюється через технології Spring Data JPA та Hibernate (ORM). Клієнтська частина побудована за допомогою HTML, CSS та шаблонізатора Thymeleaf.

Система підтримує розмежування прав доступу (ролеву модель): Студент

(проходження тестів, перегляд власних результатів), Викладач (створення та управління тестами, перегляд статистики студентів) та Адміністратор (управління користувачами та налаштуваннями системи). Програмний продукт успішно пройшов тестування та готовий до впровадження у навчальний процес.

**ЗМІСТ**

# ВСТУП

У сучасних умовах розвитку цифрових технологій освіта зазнає суттєвих трансформацій, пов’язаних із широким впровадженням інформаційно-комунікаційних технологій у навчальний процес. Одним із найважливіших напрямів таких змін є автоматизація контролю знань студентів, що дозволяє підвищити ефективність навчання, об’єктивність оцінювання та зручність взаємодії між викладачем і студентом.

Традиційні форми перевірки знань (усне опитування, письмові контрольні роботи, тестування на папері) мають низку обмежень: вони вимагають значних часових витрат викладача, схильні до суб’єктивності оцінювання, ускладнюють збереження та аналіз результатів. У свою чергу, автоматизовані системи тестування усувають ці недоліки, забезпечуючи швидке оцінювання результатів, контроль цілісності даних і можливість статистичного аналізу успішності студентів

Особливої актуальності розробка таких систем набуває у контексті дисципліни **«Об’єктно-орієнтоване програмування (ООП)»**, яка є базовою для підготовки фахівців з комп’ютерних наук, інженерії програмного забезпечення та інформаційних технологій. Ця дисципліна вимагає не лише засвоєння теоретичних понять, а й практичного вміння застосовувати принципи об’єктно-орієнтованого підходу. Використання автоматизованого тестування дозволяє перевіряти як знання основних концепцій (класи, об’єкти, наслідування, поліморфізм), так і практичні навички застосування мови програмування.

Таким чином, створення програмного продукту, що забезпечує можливість автоматизованого тестування студентів із дисципліни «ООП», є актуальним завданням, яке поєднує освітню, технічну та наукову складові. Реалізація такого проєкту сприяє удосконаленню процесу навчання, підвищенню якості підготовки

майбутніх фахівців та формуванню цифрового освітнього середовища.

**Метою дипломної роботи є** розробка програмного забезпечення, що реалізує функції автоматизованого тестування студентів з дисципліни «Об’єктно-орієнтоване програмування», з використанням сучасних технологій програмування, баз даних і веб орієнтованих рішень .

Для досягнення поставленої мети необхідно виконати низку завдань, а саме:

1. Провести теоретичний аналіз існуючих підходів до створення систем автоматизованого тестування та визначити їхні переваги й недоліки.

2. Дослідити архітектурні підходи до побудови програмних систем і обґрунтувати вибір оптимальної архітектури для даного проєкту.

3. Визначити набір технологій, мов програмування, засобів розробки та систем управління базами даних, які забезпечать ефективну реалізацію системи.

4. Розробити логічну структуру системи та модель бази даних для збереження інформації про користувачів, тести, питання та результати.

5. Спроєктувати та реалізувати основні елементи програмного забезпечення, що забезпечують авторизацію, проходження тестів, збереження результатів і формування статистики.

**Об’єктом розробки є** розробка програмного забезпечення для тестування студентів з дисципліни «Об’єктно-орієнтоване програмування»

**Предметом розробки** є готове до використання програмне забезпечення для тестування студентів з дисципліни «Об’єктно-орієнтоване програмування»

**Методи, які були використанні при розробці**, – використання середовища розробки інтегрованого типу - IntelliJ IDEA [23], об'єктно-орієнтованої мови програмування Java, фреймворку Spring Boot [20] для побудови серверної частини застосунку, інструменту об’єктно-реляційного відображення Hibernate, системи управління базами даних MySQL [12], а також шаблонізатора Thymeleaf та веб-

технологій (HTML, CSS) для розробки інтерфейсу користувача.

**Практична значимість** роботи полягає в створені веб-орієнтованого застосунку для автоматизованого тестування знань студентів з дисципліни «Об’єктно-орієнтоване програмування». Розроблений програмний продукт рекомендовано використовувати в навчальному процесі студентами спеціальності 122 «Комп’ютерні науки» в ПУЕТ для проведення поточного та підсумкового контролю знань. Програмне забезпечення може бути впроваджене в дистанційні та змішані навчальні курси кафедри комп’ютерних наук та інформаційних технологій для автоматизації перевірки результатів, підвищення об'єктивності оцінювання та моніторингу успішності студентів.

**Пояснювальна записка** до дипломної роботи складається з чотирьох розділів. **Перший розділ** містить постановку задачі на розробку вебсистеми для автоматизованого тестування студентів з дисципліни «Об’єктно-орієнтоване програмування» та опис основних завдань проєкту.

**У другому розділі** проведено аналіз сучасних програмних засобів тестування, а також розглянуто технології розробки клієнт-серверних вебзастосунків.

**Третій розділ** містить обґрунтування вибору стеку технологій, де базовою системою зберігання виступає SQL Database , розроблено алгоритмізацію роботи застосунку та описано проєктування його логічної архітектури й моделі даних.

**У четвертому розділі** описано практичну програмну реалізацію системи, наведено результати тестування програмного продукту та представлено детальну інструкцію для користувачів.

# ПОСТАНОВКА ЗАДАЧІ

У межах кваліфікаційної роботи виконується розробка вебсистеми для автоматизованого тестування студентів, призначеної для ефективного контролю знань з дисципліни «Об’єктно-орієнтоване програмування» (ООП). Система має забезпечити зручне створення, редагування, проходження та перевірку тестів, а також формування результатів і статистики успішності студентів. Основна мета – підвищення якості навчального процесу за рахунок автоматизації оцінювання знань та зменшення суб’єктивного фактору під час перевірки.

Основні завдання проєкту включають:

- Розробку інформаційної моделі системи, яка міститиме опис користувачів, тестів, питань, варіантів відповідей і результатів.
- Вибір архітектури та технологій для реалізації клієнтської та серверної частин системи (Java, SpringBoot, Theamleaf, HTML, CSS, JavaScript, SQL Database).
- Проєктування інтерфейсу користувача, який має бути інтуїтивно зрозумілим, адаптивним і доступним з будь-якого пристрою.
- Реалізацію функціоналу автентифікації користувачів, що забезпечить вхід до системи з розмежуванням ролей – студента, викладача та адміністратора.
- Створення модулів для формування та проходження тестів, що дозволить викладачам створювати питання різних, а студентам – швидко проходити тестування.
- Розробку підсистеми збереження та обробки результатів, з автоматичним підрахунком балів і статистики успішності.

- Проведення тестування системи, перевірку коректності її роботи, швидкодії та стійкості до помилок.

Основна увага в роботі приділяється:

- створенню стабільного й масштабованого програмного середовища для тестування знань студентів;
- оптимізації процесу контролю знань викладачем через автоматизацію рутинних операцій;
- забезпеченню зручності для користувача та мінімізації часу на проведення тестування;
- впровадженню сучасних вебтехнологій і принципів безпеки.

У межах реалізації проєкту передбачається дослідження таких напрямів:

- аналіз існуючих систем тестування знань у навчальному процесі, їхніх переваг і недоліків;
- обґрунтування вибору архітектури клієнт-серверного типу;
- створення функціональних модулів для авторизації, тестування та обробки результатів;
- проведення тестування роботи системи в умовах реального використання.

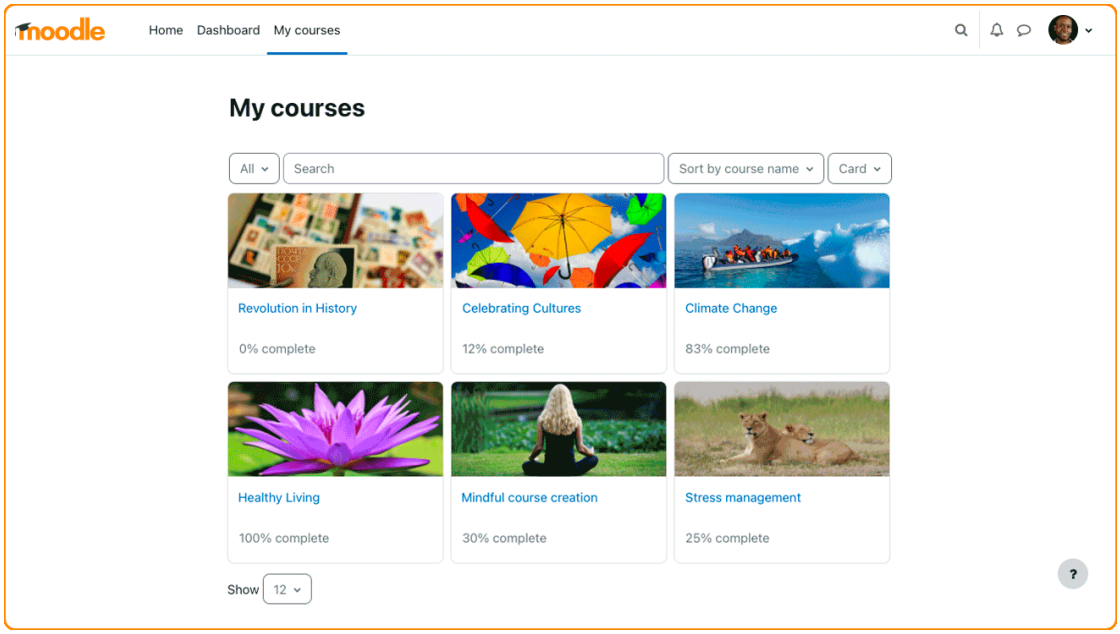
# 2. ІНФОРМАЦІЙНИЙ ОГЛЯД

## 2.1. Аналіз сучасних програмних засобів тестування

На сьогодні існує значна кількість програмних продуктів, призначених для автоматизованого контролю знань. Їх можна умовно поділити на комерційні, відкриті (open-source) та спеціалізовані навчальні системи, розроблені під конкретні заклади освіти.

До найбільш поширених належать такі платформи:

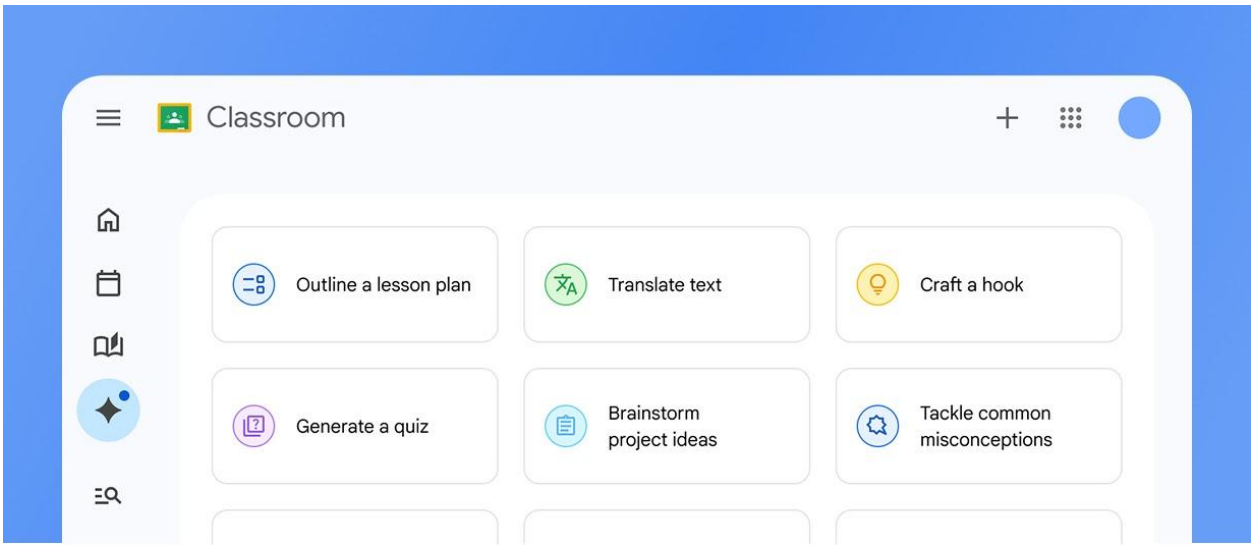
- **Moodle –** потужна система управління навчальним процесом (LMS), що підтримує створення курсів, тестів, облік результатів і статистику успішності. Її головна перевага – універсальність і розширюваність завдяки модульній структурі. Проте для невеликих курсів або окремих дисциплін вона може бути надмірно складною. Методологія розгортання та побудови систем електронного навчання на базі цієї платформи досліджується у [10], а технічні специфікації




представлені на офіційному сайті [15]. (рис.2.1)

Рисунок 2.1. – **Moodle** - платформа для організації електронного навчання.

- **Google Classroom** – зручна хмарна система, яка дозволяє викладачам роздавати завдання, проводити тести через Google Forms і збирати результати в автоматичному режимі. Основний недолік – обмежена функціональність у частині програмованих завдань і неможливість гнучкого налаштування логіки




тестування.(рис.2.2) [16]

Рисунок 2.2. **– Google Classroom** платформа для електронного навчання

- **ClassMarker** та **ProProfs Quiz Maker** – комерційні вебсервіси, орієнтовані на проведення онлайн-тестів, мають сучасний інтерфейс, але потребують підписки та не дозволяють повного контролю над даними.(рис. 2.3-2.4) [17-18]

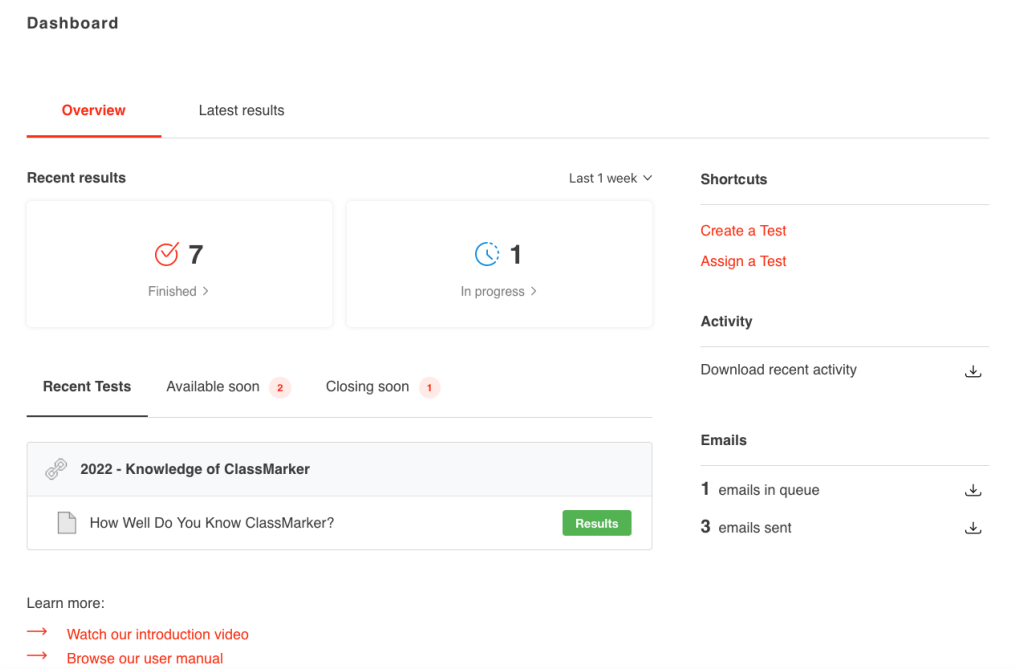



Рисунок 2.3. **– ClassMarker** – вебсервіс для тестування


Рисунок 2.4. **– ProProfs Quiz Maker** – вебсервіс для тестування

Порівняльний аналіз цих систем показує, що хоча більшість із них забезпечують базову функціональність тестування, вони або надто складні для впровадження у внутрішні навчальні процеси, або не дають можливості повної

адаптації під конкретні дисципліни. У контексті вищої школи впровадження таких інструментів потребує детального вивчення методології цифровізації освіти..Саме тому розробка власного, спеціалізованого рішення для дисципліни **«Об'єктно-орієнтоване програмування»** є доцільною. Таке програмне забезпечення дозволить реалізувати унікальні типи завдань, пов'язані з аналізом програмного коду, принципами спадкування та поліморфізму, що виходять за межі стандартних тестових систем.

## 2.2 Технології розробки веб-застосунків для тестування

Проєктування та розробка сучасної автоматизованої системи тестування вимагає комплексного підходу до вибору архітектурних рішень та програмного стека. Загальні класичні принципи інженерії програмного забезпечення та проектування подібних багаторівневих інформаційних систем детально викладено у посібнику [13]. Оскільки система повинна забезпечувати безперебійну роботу під час одночасного проходження тестування великою кількістю здобувачів освіти, гарантувати цілісність даних та мати зручні інструменти для управління навчальним процесом, її архітектура традиційно будується за багаторівневою клієнт-серверною моделлю.

Ця модель передбачає чіткий поділ системи на логічні рівні - рівень представлення (клієнтська частина або Frontend), рівень бізнес-логіки (серверна частина або Backend) та рівень зберігання даних (база даних).

### 2.2.1 Технології розробки серверної частини (Backend)

Серверна частина є програмним ядром застосунку, яке бере на себе виконання найважливіших завдань: обробку HTTP-запитів, реалізацію бізнес-логіки підрахунку результатів, маршрутизацію інформаційних потоків та забезпечення безпеки.

Для побудови високонавантажених та надійних освітніх платформ стандартом є використання об'єктно-орієнтованої мови програмування **Java**, архітектурні стандарти та практичні аспекти якої описані в офіційній документації компанії Oracle [11]. Оскільки розроблювана система орієнтована на контроль знань з дисципліни «Об'єктно-орієнтоване програмування», теоретичною та методологічною базою для проектування логічних модулів послужили концептуальні підходи, що досліджуються в [1], [2], [8] [9]. Розроблювана система призначена саме для контролю знань з парадигми ООП, ключове значення

має коректна реалізація базових концепцій мови (інкапсуляція, поліморфізм, спадкування та абстракція). Її переваги полягають у строгій типізації, потужних механізмах управління пам'яттю (Garbage Collection) та високій ефективності в багатопотоковому середовищі, що критично важливо для паралельної обробки відповідей студентів.

Основною технологічною базою для розробки серверної логіки доцільно обрати фреймворк **Spring Boot** [20]. Він надає потужний інструментарій для швидкого створення повноцінних RESTful веб-сервісів завдяки механізмам інверсії управління (IoC) та впровадження залежностей (Dependency Injection). Використання екосистеми Spring дозволяє ефективно реалізувати ключовий функціонал системи тестування:

- **Управління доступом та безпекою -** За допомогою модуля *Spring Security* реалізується надійна автентифікація та рольова модель авторизації. Це дозволяє чітко розмежувати права доступу: створити захищені панелі адміністратора (для глобального управління системою) та вчителя/викладача (для створення тестів, управління списками студентів та моніторингу успішності).
- **Об'єктно-реляційне відображення (ORM) -** Для взаємодії бізнес-логіки з базою даних використовується стандарт JPA (Java Persistence API), еталонною реалізацією якого є **Hibernate**. Hibernate автоматизує процес трансформації об'єктів мови Java у записи реляційних таблиць бази даних і навпаки. Це дозволяє розробнику оперувати складними зв'язаними сутностями (наприклад, система зв'язків між класами Студент, Тест, Питання, Варіант відповіді та Результат) на рівні коду, мінімізуючи написання рутинних SQL-запитів та забезпечуючи високу швидкість розробки.

### 2.2.2 Технології клієнтської частини (Frontend)

Інтерфейс користувача у системі тестування має бути максимально чуйним та інтуїтивно зрозумілим. Будь-які затримки або перезавантаження сторінок під

час проходження тесту можуть призвести до втрати відповідей або десинхронізації таймера.

Враховуючи архітектуру системи на базі Spring Boot, для розробки клієнтської частини було обрано серверний шаблонізатор Thymeleaf [19] у поєднанні з базовими веб-технологіями: HTML5[21], CSS3[22] та JavaScript. Загальні проектування інтерфейсів із використанням цих вебтехнологій та створення динамічних сторінок висвітлені у навчальному посібнику [3]. Замість використання громіздких SPA-фреймворків, необхідна динамічність та плавність інтерфейсу забезпечується за допомогою нативного JavaScript.

Зокрема, під час проходження тесту браузер завантажує сторінку з масивом питань лише один раз. Подальша взаємодія (перехід між питаннями) відбувається миттєво шляхом зміни видимості елементів у DOM-дереві за допомогою клієнтських скриптів без додаткових звернень до сервера. Це повністю унеможливлює втрату обраних відповідей та гарантує безперебійну і точну роботу клієнтського таймера. Крім того, для реалізації інтерактивних елементів системи (наприклад, функціонування вбудованого чату підтримки) використовується технологія асинхронних запитів (Fetch API) до серверної частини, що дозволяє оновлювати контент у фоновому режимі без перезавантаження активної сторінки.

### 2.2.3. Управління даними та їх зберігання

Система тестування генерує велику кількість структурованих даних, тому використання реляційних систем керування базами даних (СКБД) є найбільш обґрунтованим рішенням.

- **Організація підключень -** Одним із найвужчих місць у продуктивності веб-застосунків під час пікових навантажень (наприклад, під час модульного контролю) є процес встановлення з'єднання з базою даних. Для розв'язання цієї проблеми в сучасних Java-застосунках використовується пул з'єднань **HikariCP.** Це надзвичайно швидкий інструмент управління з'єднаннями, який підтримує

набір відкритих підключень до бази даних і ефективно розподіляє їх між потоками, унеможливлюючи перевантаження СКБД та суттєво зменшуючи час відгуку сервера.

- **Етап розробки та тестування -** На стадії активної розробки проєкту, проєктування архітектури бази даних та перевірки міграцій високоефективним є використання **H2 Database**. Це реляційна in-memory база даних, що підключається за допомогою H2 JDBC драйверів. Її головна перевага полягає в тому, що вона може працювати безпосередньо в оперативній пам'яті застосунку, не вимагаючи встановлення та конфігурації окремого сервера БД. Це значно прискорює процес розробки та автоматизованого тестування коду.
- **Продуктове середовище (Production) -** Для остаточного розгортання проєкту використовуються повноцінна СКБД, а саме MySQL [12], яка гарантує транзакційну цілісність даних та надійне збереження результатів тестувань та розгорнутої статистики.

# 3. ТЕОРЕТИЧНА ЧАСТИНА

## 3.1.Обґрунтування вибору стеку технологій

| Компонент | Технологія |
|---|---|
| Мова програмування | Java 17 |
| Основний фреймворк | Spring Boot 3.2.5 |
| Безпека та авторизація | Spring Security |
| Робота з даними | Hibernate 6.4.4 & Spring Data JPA |
| База даних | MySQL |
| Шаблонізатор | Thymeleaf |

Для реалізації автоматизованої системи тестування було обрано монолітну клієнт-серверну архітектуру на базі шаблону проєктування MVC (Model-View-Controller). Під час проєктування логічної моделі та зв'язків між сутностями використовувалися уніфіковані графічні патерни на основі методології UML, описаної у [14]. Такий підхід дозволяє централізовано управляти бізнес-логікою, даними та представленням, що є оптимальним для освітніх платформ із чітким розмежуванням прав доступу. При розробці архітектурних модулів також враховувався досвід створення сучасних вебсервісів кафедри комп'ютерних наук ПУЕТ, зокрема підходи до автоматизованої обробки документів та побудови інформаційних систем, висвітлені у наукових статтях [6, 7]. Вибір конкретних інструментів здійснювався з огляду на вимоги до надійності, безпеки та швидкості розробки.

**1. Мова програмування та серверна платформа (Backend)**

Основою серверної частини застосунку виступає мова **Java**, яка є індустріальним стандартом для побудови надійних корпоративних систем. Головним фреймворком обрано **Spring Boot**. Його використання обґрунтовується

такими перевагами:

- **Управління залежностями (Maven) -** Проєкт використовує pom.xml для автоматизованого збирання та управління підключеними бібліотеками, що унеможливлює конфлікти версій.
- **Spring Security -** Забезпечує надійну автентифікацію та авторизацію. У проєкті реалізовано SecurityConfig, що дозволяє розмежувати доступ до ендпоінтів на основі ролей (ADMIN, TEACHER, STUDENT). Це гарантує, що студент не зможе отримати доступ до панелі створення тестів або перегляду статистики.
- **Spring Data JPA (Hibernate) -** Забезпечує об'єктно-реляційне відображення (ORM). Використання репозиторіїв (наприклад, UserRepository, TestRepository) дозволяє виконувати CRUD-операції з базою даних без написання низькорівневих SQL-запитів, оперуючи виключно об'єктами Java.

**2. Клієнтська частина (Frontend)**

Для реалізації інтерфейсу користувача було відмовлено від складних SPA-фреймворків на користь серверного рендерингу (Server-Side Rendering) за допомогою шаблонізатора **Thymeleaf.**

- Thymeleaf (файли в директорії templates/, такі як admin-hub.html, test.html, student.html) дозволяє динамічно генерувати HTML-сторінки на стороні сервера, вбудовуючи в них дані з Java-контролерів.
- Цей підхід забезпечує високу швидкість завантаження сторінок, відсутність проблем із SEO-індексацією та ідеальну інтеграцію зі Spring Security (зокрема, використання тегів sec:authorize для відображення різних елементів меню залежно від ролі).
- Для стилізації використовується класичний CSS (static/style.css), що робить застосунок легким та незалежним від сторонніх UI-бібліотек.

**3. База даних**

Для надійного зберігання інформації у проєкті використовується повноцінна реляційна система управління базами даних (MySQL).

Вибір стандартної SQL-бази даних зумовлений її високою продуктивністю, надійністю та масштабованістю. На відміну від легковагових вбудованих рішень, використання повноцінної реляційної СКБД є оптимальним для реальних умов експлуатації в освітніх закладах. Вона гарантує строгу цілісність даних, безпечне зберігання облікових записів користувачів, структури тестів та результатів оцінювання, а також ефективно обробляє велику кількість одночасних підключень під час тестування цілих груп студентів.

Крім того, використання класичної SQL-бази дозволяє зручно адмініструвати систему, створювати резервні копії та перевіряти статистичні дані за допомогою спеціалізованих інструментів управління (наприклад, MySQL Workbench). Взаємодія серверної частини програми з базою даних здійснюється завдяки технології Spring Data JPA (через Hibernate). Це забезпечує гнучкість архітектури, дозволяє працювати з даними на рівні об'єктів Java без написання складних SQL-запитів вручну та суттєво спрощує подальшу підтримку і розширення функціоналу системи.

**4. Додаткові технології**

У проєкті реалізовано спеціалізований модуль для читання та динамічної генерації тестових питань із зовнішнього файлу. Відповідно до архітектури системи, за цю логіку відповідає сервіс GeneratorService. Цей компонент зчитує масив даних із файлу questions.json (який розміщено в ресурсах проєкту) та здійснює їх десеріалізацію у відповідні об'єкти моделі Question за допомогою бібліотеки Jackson (ObjectMapper).

Розроблений алгоритм дозволяє автоматично фільтрувати питання за вказаною тематикою (категорією), доповнювати вибірку випадковими завданнями у разі їх нестачі та перемішувати результати перед видачею. Реалізація такого підходу свідчить про високу гнучкість системи до імпорту зовнішніх даних. Це дозволяє легко розширювати базу завдань, використовуючи масиви питань, експортовані зі сторонніх освітніх платформ, без необхідності змінювати

структуру бази даних чи модифікувати вихідний код програми.

### 3.2. Алгоритмізація роботи застосунку

Робота системи тестування базується на кількох ключових алгоритмах, які забезпечують життєвий цикл проходження контролю знань: від авторизації до підрахунку балів та формування статистики.

**1. Алгоритм автентифікації та маршрутизації користувачів**

Вхід у систему контролюється AuthController та фільтрами Spring Security(Рис. 3.1).

1. Користувач вводить облікові дані на сторінці login.html.
2. Система шукає об'єкт User у базі даних (через UserRepository).
3. У разі успішної перевірки пароля, алгоритм зчитує роль користувача (Role).
4. На основі ролі спрацьовує маршрутизатор (HomeController):
   - Якщо ADMIN -> перенаправлення на /admin (AdminController).
   - Якщо TEACHER -> перенаправлення на /teacher (TeacherController).
   - Якщо STUDENT -> перенаправлення на /student (StudentController).
5. Усі дії логуються за допомогою UserStatusInterceptor та зберігаються через AuditService у таблицю AuditLog для контролю безпеки.

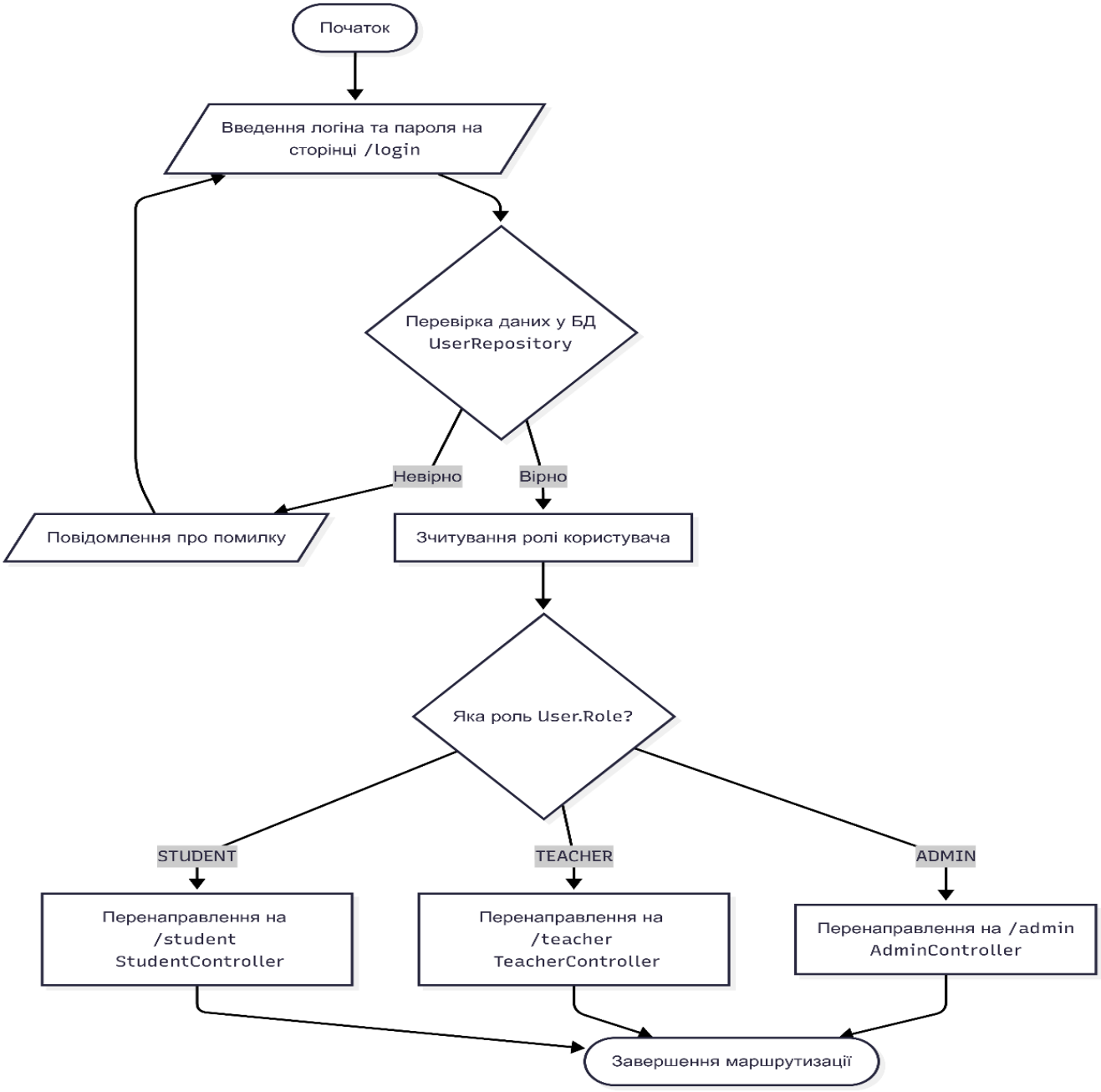



Рисунок 3.1 - Схема авторизації та маршрутизації користувача.

**2. Алгоритм проходження тесту та оцінювання (ResultService)**

Це ядро бізнес-логіки застосунку для студента. (Рис. 3.2).

1. Студент обирає доступний тест (Test). Система завантажує список питань (Question), пов'язаних із цим тестом.
2. Відображення питань реалізується на сторінці test.html.
3. Після вибору відповідей та натискання кнопки "Завершити", масив відповідей передається методом POST на сервер.
4. ResultService ініціалізує об'єкт Result.
5. Запускається цикл перевірки: кожна відповідь студента порівнюється з правильним варіантом (correct option), заданим у моделі Question.
6. Формується деталізація відповідей (Result.AnswerDetail), де фіксується статус кожної відповіді (правильно/неправильно).
7. Вираховується загальний бал у відсотковому співвідношенні або абсолютних величинах.
8. Об'єкт Result зберігається у базу даних, а студент перенаправляється на сторінку деталізації результату (result-details.html).

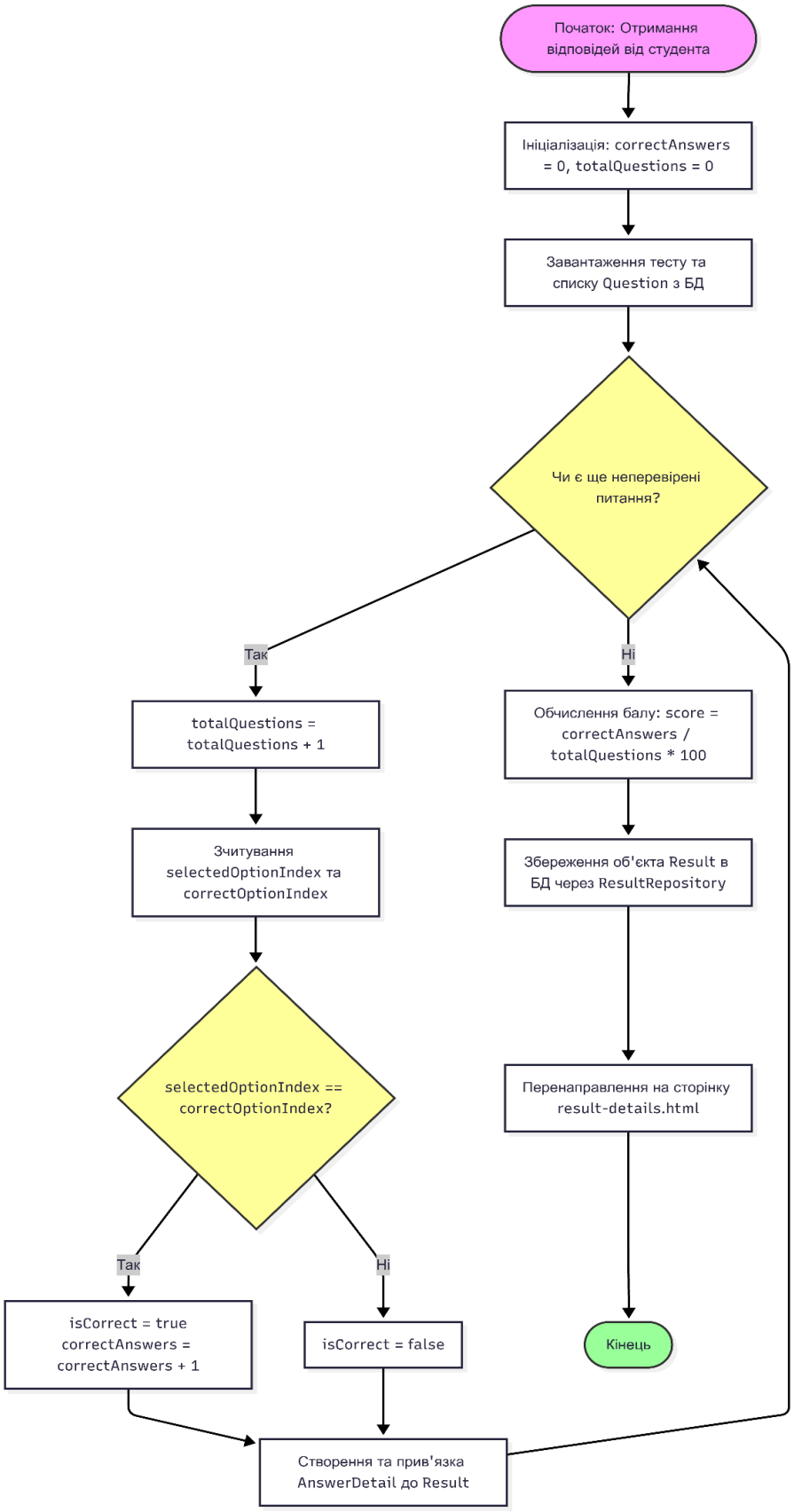



Рисунок 3.2 - Схема підрахунку балів за пройдений тест.

**3. Алгоритм генерації тестового контенту** Особливістю проєкту є алгоритм автоматизованого заповнення бази тестів GeneratorService. Алгоритм зчитує структуру з файлу questions.json, парсить JSON-об'єкти, конвертує їх у Java-сутності Question та прив'язує до відповідного об'єкта Test у базі даних. Це значно прискорює процес наповнення системи контентом для викладача.

### 3.3. Проєктування застосунку

На етапі проєктування було розроблено архітектуру, яка відповідає принципам SOLID, забезпечуючи високу модульність та легкість у підтримці коду.

#### 3.3.1. Логічна архітектура системи

Додаток структурно розділений на такі логічні шари (Рис. 3.3) :

1. **Шар представлення (View Layer)**

Набір HTML-шаблонів Thymeleaf (папка templates), які відповідають за графічний інтерфейс: панелі адміністратора (admin-content.html, admin-logs.html, admin-users.html), кабінети (teacher.html, student.html) та систему обміну повідомленнями (chat.html).

2. **Шар контролерів (Controller Layer)**

Класи-контролери (наприклад, AdminController, TeacherController, ChatController), які приймають HTTP-запити від браузера, передають дані на обробку в сервіси і повертають відповідний HTML-шаблон з оновленою моделлю даних.

3. **Шар бізнес-логіки (Service Layer)**

Класи з інструкцією @Service (TestService, AuthService, ChatService, AuditService). Тут інкапсульована вся складна логіка: валідація, підрахунок оцінок, управління сесіями користувачів та бізнес-правила системи.

4. **Шар доступу до даних (Repository Layer)**

Інтерфейси (наприклад, ResultRepository), що успадковують JpaRepository і забезпечують взаємодію з БД SQL.

5. **Шар моделей даних (Domain Layer)**

Java-класи (Entity), які є відображенням таблиць бази даних.

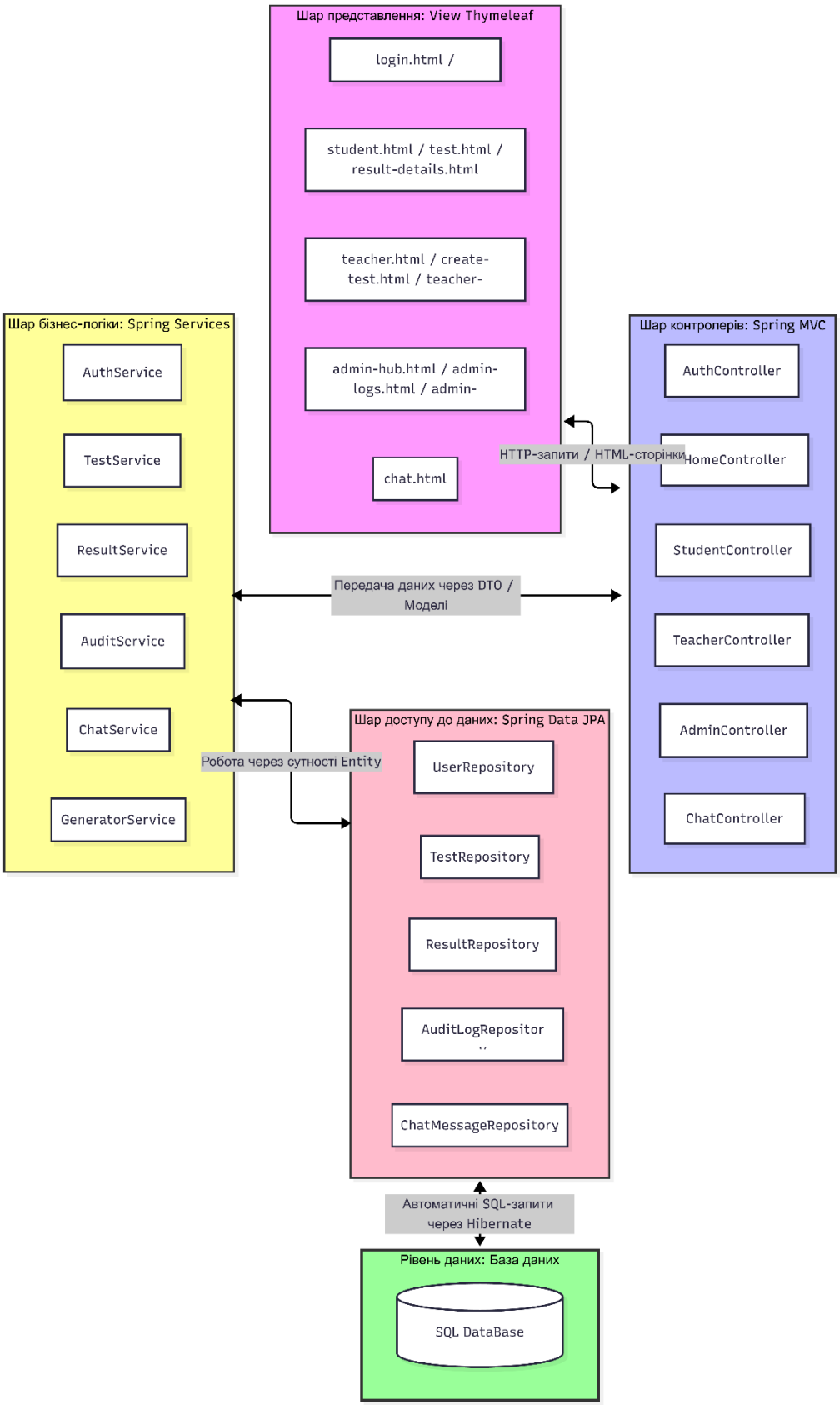



Рисунок 3.3 - Схема логічної архітектури вебзастосунку

### 3.3.2. Проєктування бази даних (Модель даних)

База даних системи побудована за реляційним принципом і нормалізована для уникнення дублювання інформації. В основі моделі лежать такі ключові сутності (Entities) та зв'язки між ними:

1. **Сутність User (Користувач):**

Базовий клас для всіх учасників системи. Зберігає облікові дані (username, password) та роль. Для специфічних потреб студентів реалізовано розширення моделі – StudentUser, яка може містити додаткові атрибути (група, спеціальність).

2. **Сутність Test (Тест):**

Центральний об'єкт навчального процесу. Містить назву, опис та посилання на автора-викладача (зв'язок Many-to-One з User).

3. **Сутність Question (Питання):**

Зберігає текст питання, варіанти відповідей та індекс правильної відповіді. Пов'язана з сутністю Test (зв'язок Many-to-One: багато питань належать одному тесту).

4. **Сутність Result (Результат):**

Сутність, що фіксує факт проходження тесту. Пов'язує студента (User) та конкретний іспит (Test). Зберігає підсумковий бал, дату проходження, а також деталізацію кожної відповіді (AnswerDetail).

5. **Допоміжні сутності:**

ChatMessage для реалізації внутрішнього чату системи між користувачами та AuditLog для зберігання історії дій (хто, коли і які зміни вносив у систему) для забезпечення прозорості та безпеки (SystemSettings).

# 4. ПРАКТИЧНА ЧАСТИНА

## 4.1. Програмна реалізація застосунку для тестування

Практична реалізація вебзастосунку виконана мовою програмування Java з використанням фреймворку Spring Boot. Проєкт має чітку пакетну структуру, яка відповідає архітектурному патерну MVC (Model-View-Controller) та принципам єдиної відповідальності (Single Responsibility).

Основна логіка застосунку розподілена між такими ключовими пакетами:

1. **Пакет config (Конфігурація системи)**

Клас SecurityConfig відповідає за налаштування безпеки (Spring Security). У ньому реалізовано шифрування паролів за допомогою BCryptPasswordEncoder та налаштовано ланцюжок фільтрів безпеки (SecurityFilterChain). Завдяки цьому екрани маршрутизуються залежно від ролі: доступ до URL /admin/ має лише роль ADMIN, до /teacher/ – TEACHER, а до /student/ – STUDENT.

Клас DataInitializer (імплементує CommandLineRunner) використовується для первинного наповнення бази даних базовими користувачами (адміністратором, викладачем) під час першого запуску застосунку.

2. **Пакет model (Об'єктна модель)**

Містить сутності JPA (Entity), які відображаються на таблиці в базі даних SQL. Наприклад, класи User, StudentUser, Test, Question, Result, AuditLog. Усі моделі містять відповідні анотації Hibernate (@Entity, @Table, @OneToMany, @ManyToOne) для автоматичної генерації реляційних зв'язків.

3. **Пакет repository (Доступ до даних)**

Містить інтерфейси (наприклад, TestRepository, ResultRepository), які успадковують JpaRepository. Вони забезпечують виконання базових операцій

(CRUD) та спеціалізованих запитів (наприклад, пошук результатів конкретного студента) без необхідності написання SQL-коду.

4. **Пакет service (Бізнес-логіка)**

ResultService: містить критичну логіку перевірки відповідей студента. Метод приймає масив обраних відповідей, порівнює їх з correctOptionIndex кожного питання, вираховує загальний бал та формує деталізацію AnswerDetail.

GeneratorService: реалізує парсинг файлу questions.json. Це дозволяє викладачу автоматично імпортувати питання безпосередньо в базу даних.

AuditService: фіксує дії користувачів та зберігає їх у систему логування.

5. **Пакет controller (Маршрутизація запитів)**

Контролери (AdminController, TeacherController, StudentController, AuthController) обробляють HTTP-запити від клієнта. Вони отримують дані з сервісів і передають їх у шаблони Thymeleaf (папка resources/templates/).

6. **Клієнтська частина (Frontend)**

Інтерфейс реалізовано за допомогою HTML-шаблонів Thymeleaf (наприклад, student.html, test.html). Для стилізації використовується єдиний файл style.css. Відображення елементів інтерфейсу є динамічним завдяки тегам sec:authorize (наприклад, кнопка "Адмін-панель" з'являється лише у користувачів з відповідними правами).

### 4.2. Тестування програмного продукту

Для забезпечення надійності та безперебійної роботи системи було проведено комплексне тестування розробленого програмного продукту. Методологічною основою для перевірки працездатності системи став посібник [5]. Оскільки застосунок орієнтований на навчальний процес, особлива увага приділялася функціональному тестуванню та тестуванню безпеки.

**1. Функціональне тестування (Functional Testing)** - Було розроблено та

успішно пройдено низку тест-кейсів (Test Cases), які імітують реальні дії користувачів.

- *Тест-кейс 1: Реєстрація та авторизація.* Перевірено успішне створення облікових записів адміністратором. Підтверджено, що при введенні неправильного логіна або пароля система видає відповідне попередження і не допускає користувача до панелі управління(Рис. 4.1,4.2).

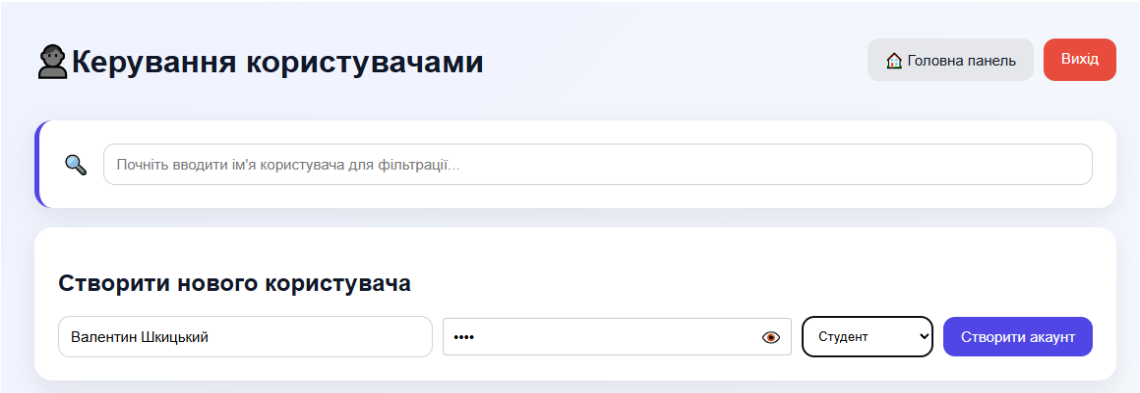



Рисунок 4.1 – створення користувача адміністратором.

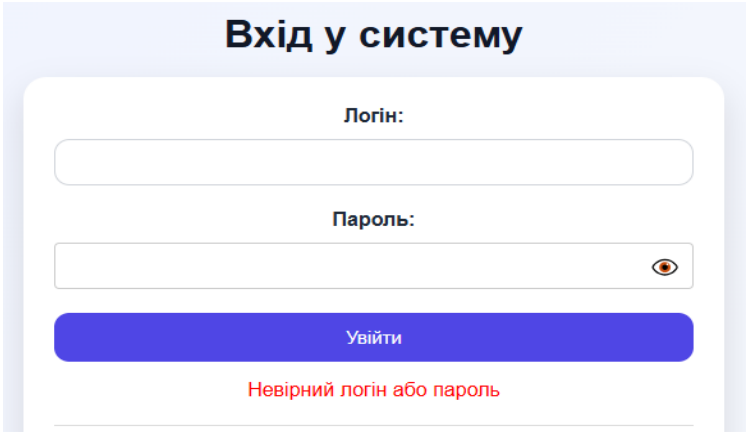



Рисунок 4.2 – Спроба введення не правильного пароля або логіну

- *Тест-кейс 2: Створення тесту викладачем.* Перевірено форму створення тесту (create-test.html). Система коректно зберігає назву тесту, питання та ідентифікатор правильної відповіді в базу даних SQL. Також успішно протестовано імпорт питань через GeneratorService(Рис. 4.3,4.4).

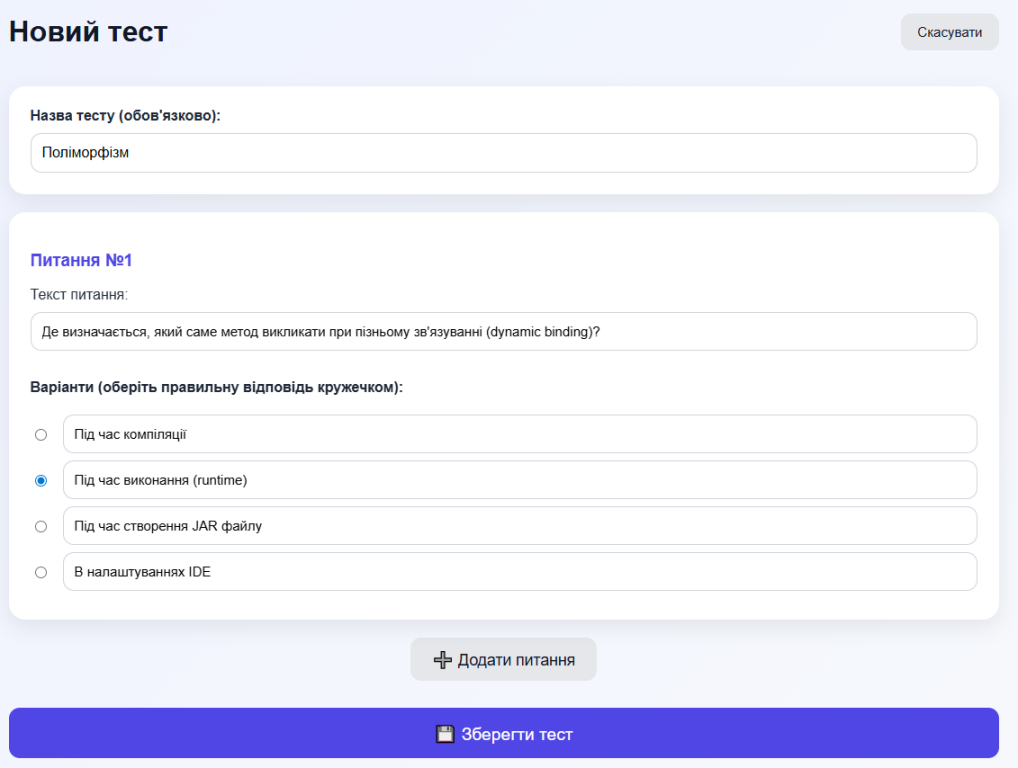



Рисунок 4.3 – Створення тесту Викладачем

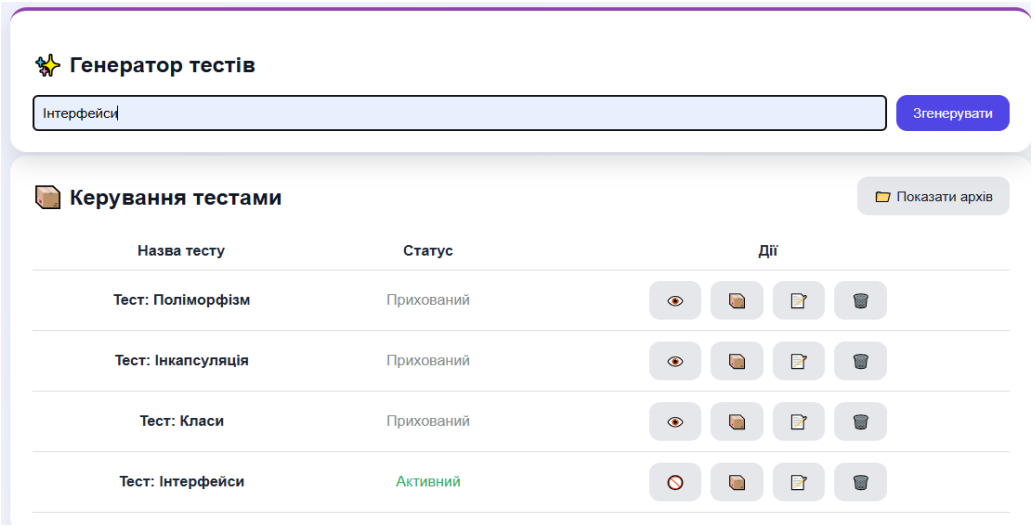



Рисунок 4.4 – Генерація тесту через GeneratorService

- *Тест-кейс 3: Проходження тесту студентом.* Студент обирає тест зі списку доступних. Перевірено коректність підрахунку балів (ResultService). Результати точно відповідають кількості правильних відповідей. Дані коректно зберігаються і відображаються на сторінці result-details.html(Рис. 4.5).

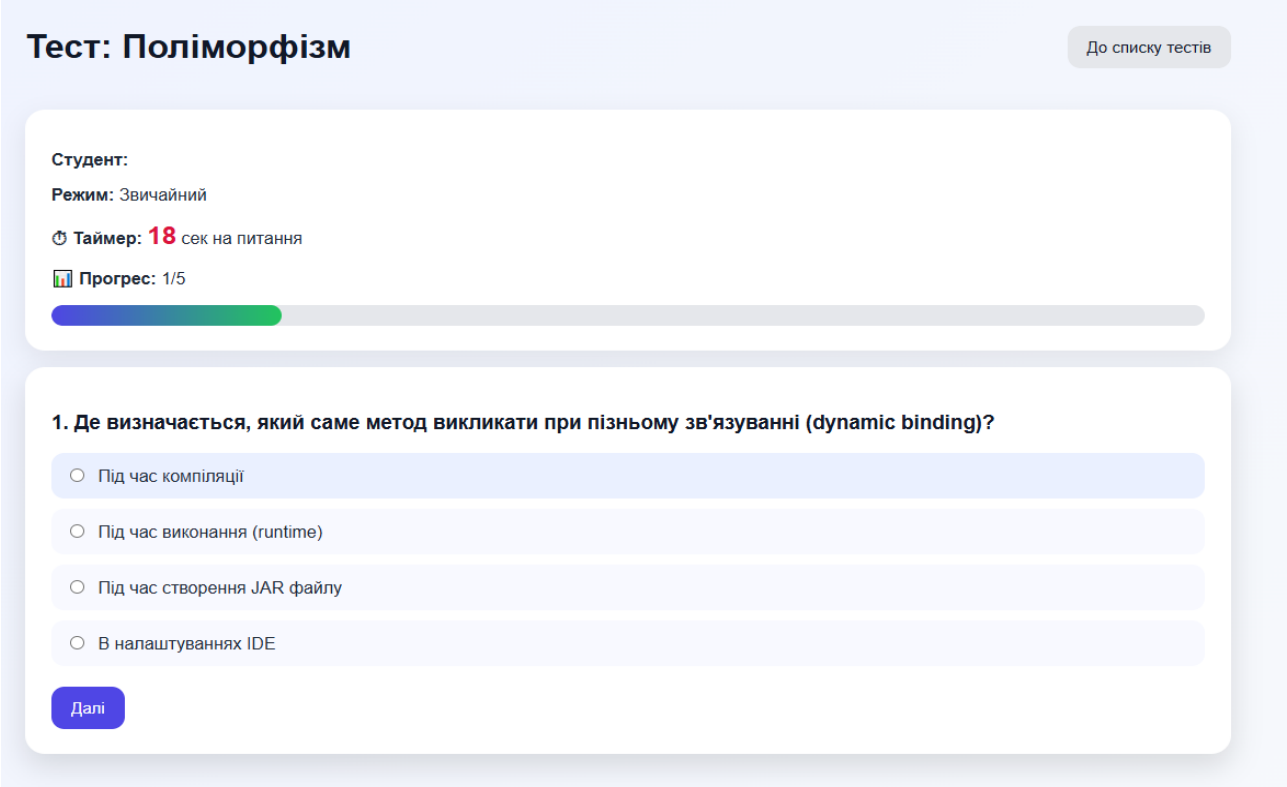



Рисунок 4.5 – Проходження тесту студентом

**2. Тестування безпеки та прав доступу (Security Testing)** - Захист системи тестувався шляхом спроб несанкціонованого доступу до сторінок.

- Спроба студента зайти за прямим посиланням /admin/stats або /teacher/students автоматично блокується механізмами SecurityConfig, після чого відбувається перенаправлення на сторінку автентифікації(login.html).

- Було перевірено роботу UserStatusInterceptor, який успішно перехоплює всі запити користувачів і передає дані в AuditLog для фіксації їхніх дій (Рис. 4.6).

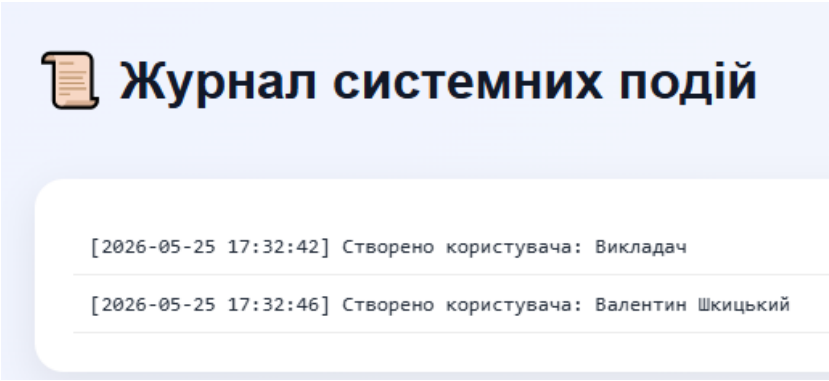



Рисунок 4.6 – Журнал системних подій

**3. Тестування користувацького інтерфейсу (UI/UX Testing)** - Перевірено коректне відображення шаблонів Thymeleaf. Переконано, що форми відправки даних працюють без помилок, а повідомлення (включаючи чат на базі ChatMessage) своєчасно оновлюються.

**Висновок до підрозділу.**

Всі етапи тестування завершилися успішно. Виявлені на етапі розробки дрібні недоліки були виправлені. Застосунок повністю відповідає початковим вимогам і готовий до експлуатації.

### 4.3. Інструкція для користувачів

Залежно від призначеної ролі (Студент, Викладач або Адміністратор), після авторизації користувач отримує доступ до відповідного функціоналу.

#### 4.3.1. Інструкція для Студента

1. **Авторизація** - На стартовій сторінці введіть свій логін та пароль, наданий адміністратором.
2. **Головний екран** - Після входу відкриється "Особистий кабінет студента"(Рис. 4.7.). Тут ви побачите список доступних для проходження тестів та розділ ваших попередніх результатів.

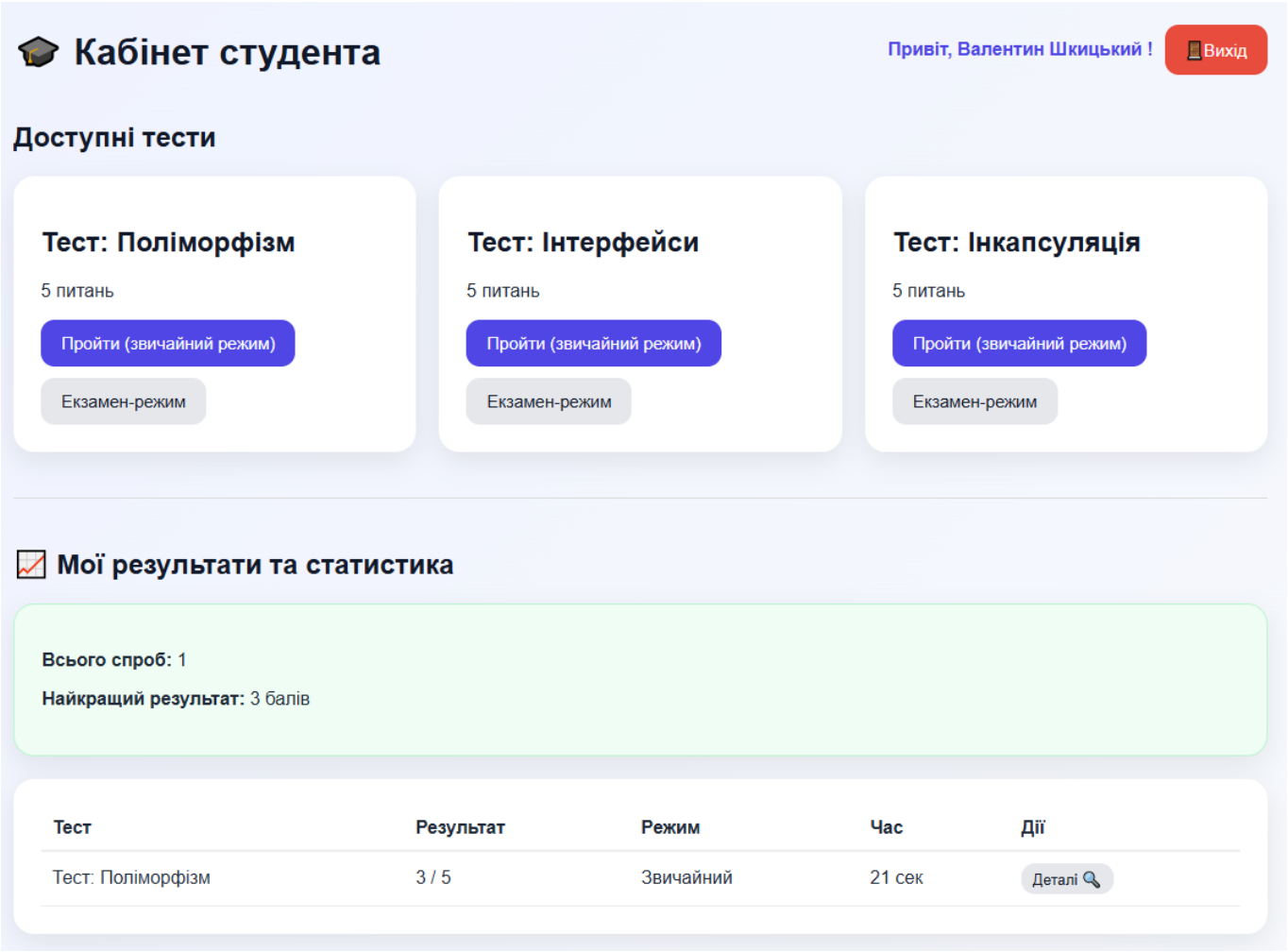



Рисунок 4.7 – Кабінет студента

**3. Проходження тесту -** Оберіть тест зі списку та натисніть "Почати". На екрані з'являться питання з варіантами відповідей. Уважно прочитайте кожне питання та оберіть один правильний варіант.

**4. Завершення -** Після відповіді на всі питання натисніть кнопку "Завершити тест". Система миттєво опрацює ваші відповіді та покаже ваш результат, а також дозволить переглянути, де саме ви допустили помилки (зеленим кольором підсвічуються правильні відповіді, червоним – неправильні)(Рис. 4.8).

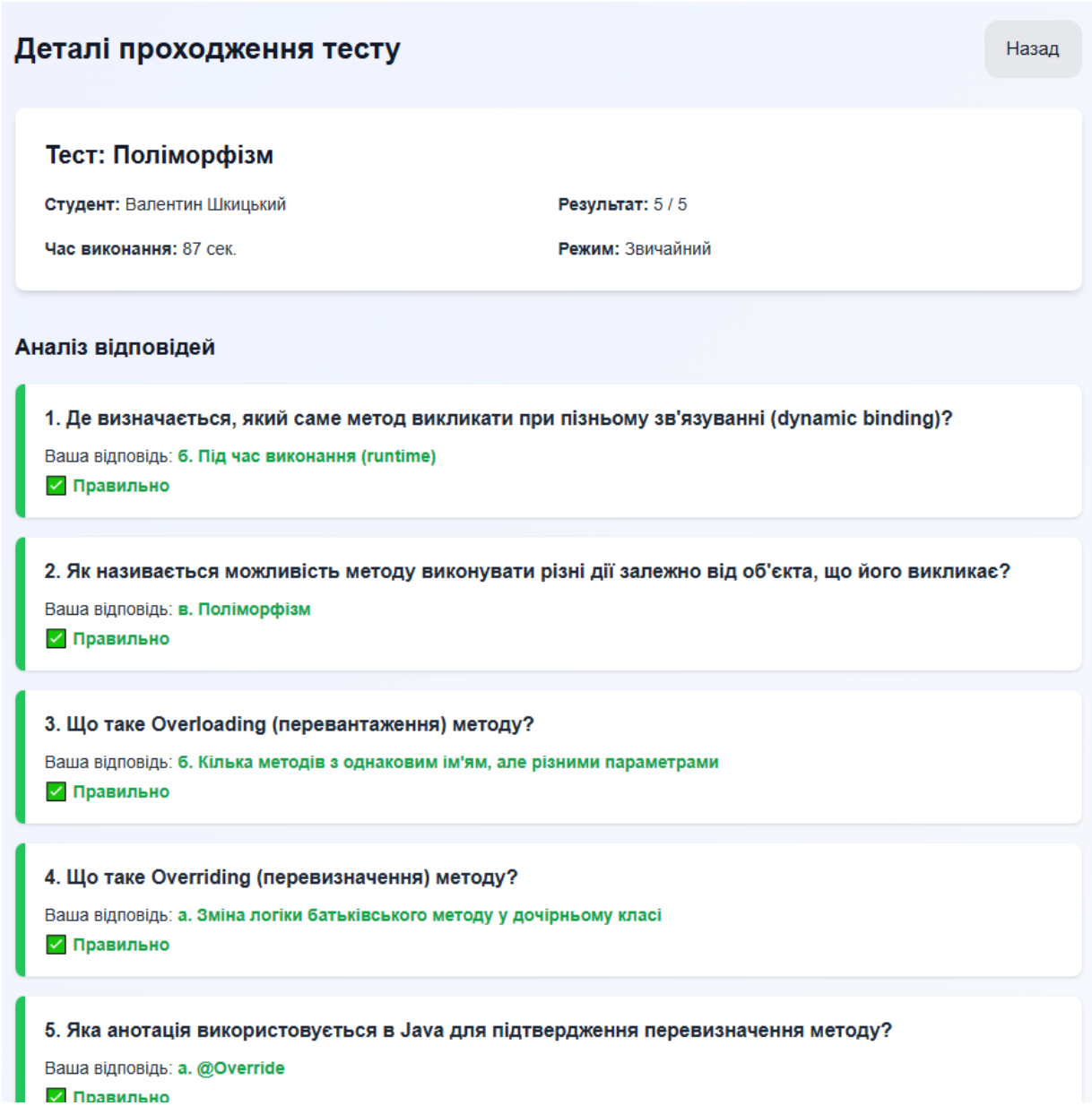



Рисунок 4.8 – Деталі проходження тесту.

### 4.3.2. Інструкція для Викладача

1. **Особистий кабінет -** Після авторизації викладач потрапляє до "Панелі викладача"(Рис. 4.9), де знаходяться інструменти для створення навчального контенту.

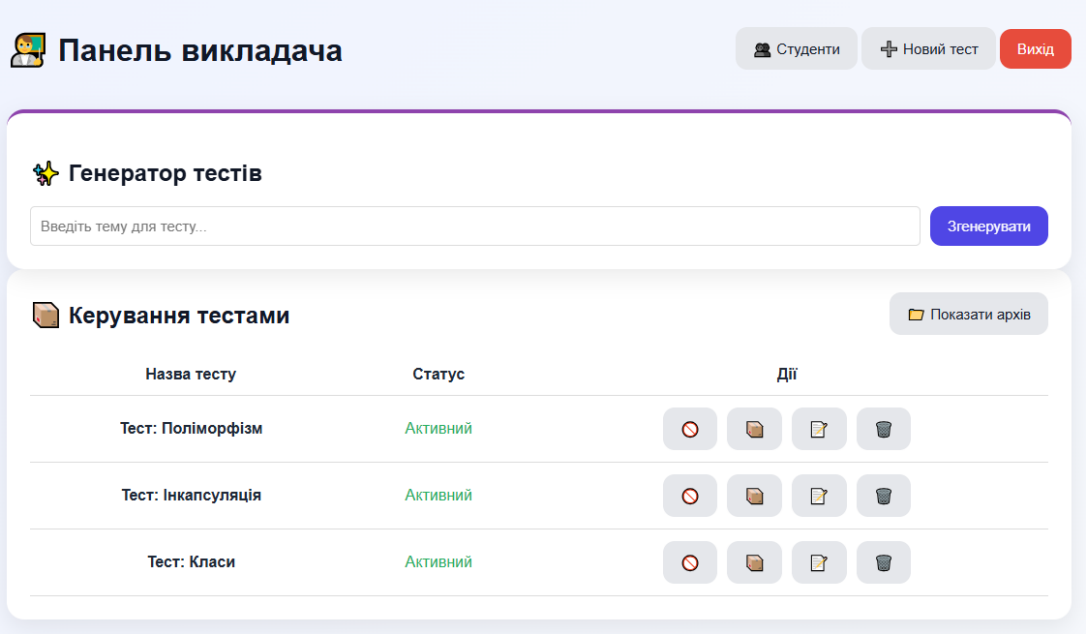



Рисунок 4.9 – Панель викладача

2. **Створення тестів -** Натисніть кнопку "Створити тест". Введіть назву, опис та додайте необхідну кількість питань із зазначенням правильних відповідей. Також доступна функція автоматичного генератора питань(Рис. 4.3).

3. **Аналіз успішності** - У розділі "Студенти" викладач має змогу переглядати список (Рис. 4.10) здобувачів освіти та детальну статистику проходження(Рис. 15-16) створених ним тестів кожним окремим студентом.

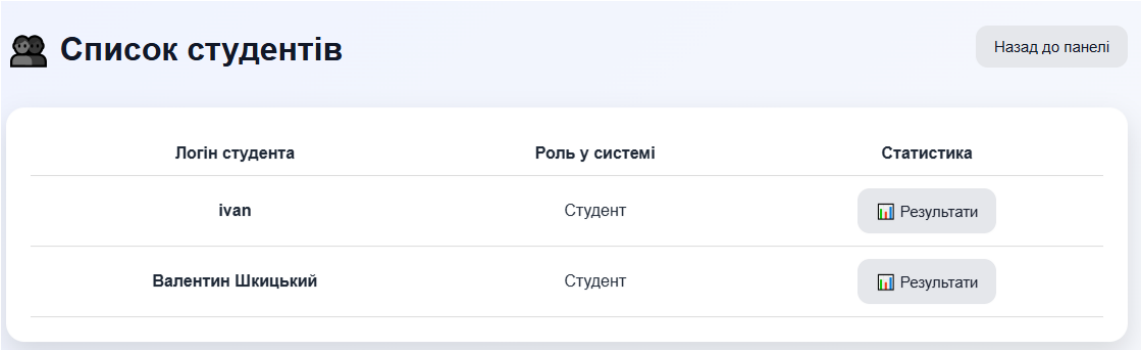



Рисунок 4.10 – Список студентів

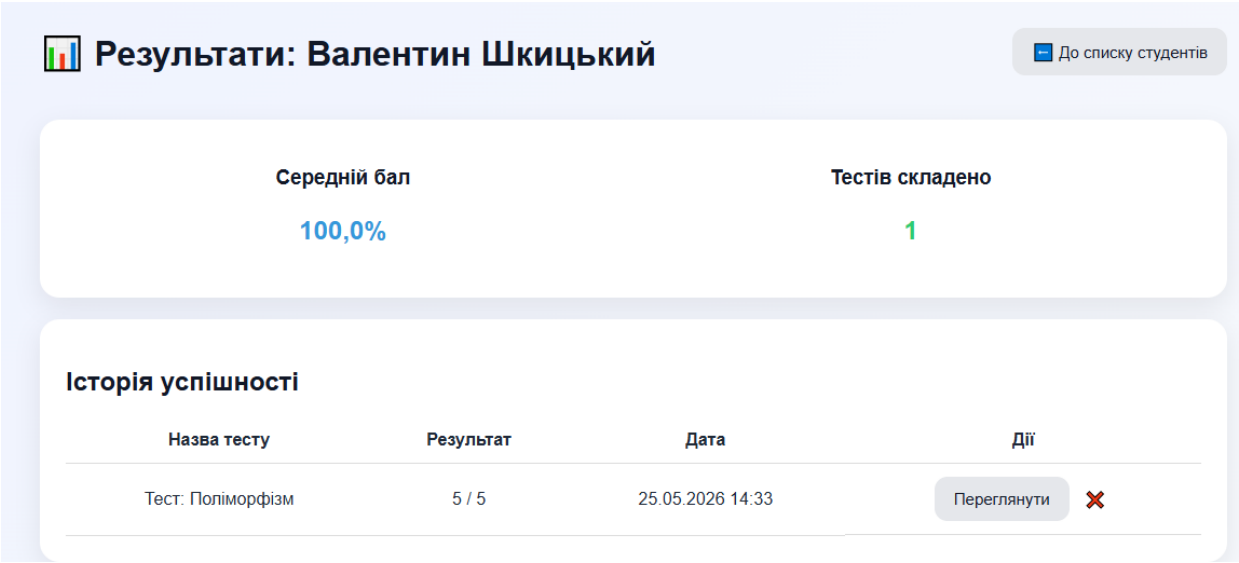



Рисунок 4.11 – Результати студента

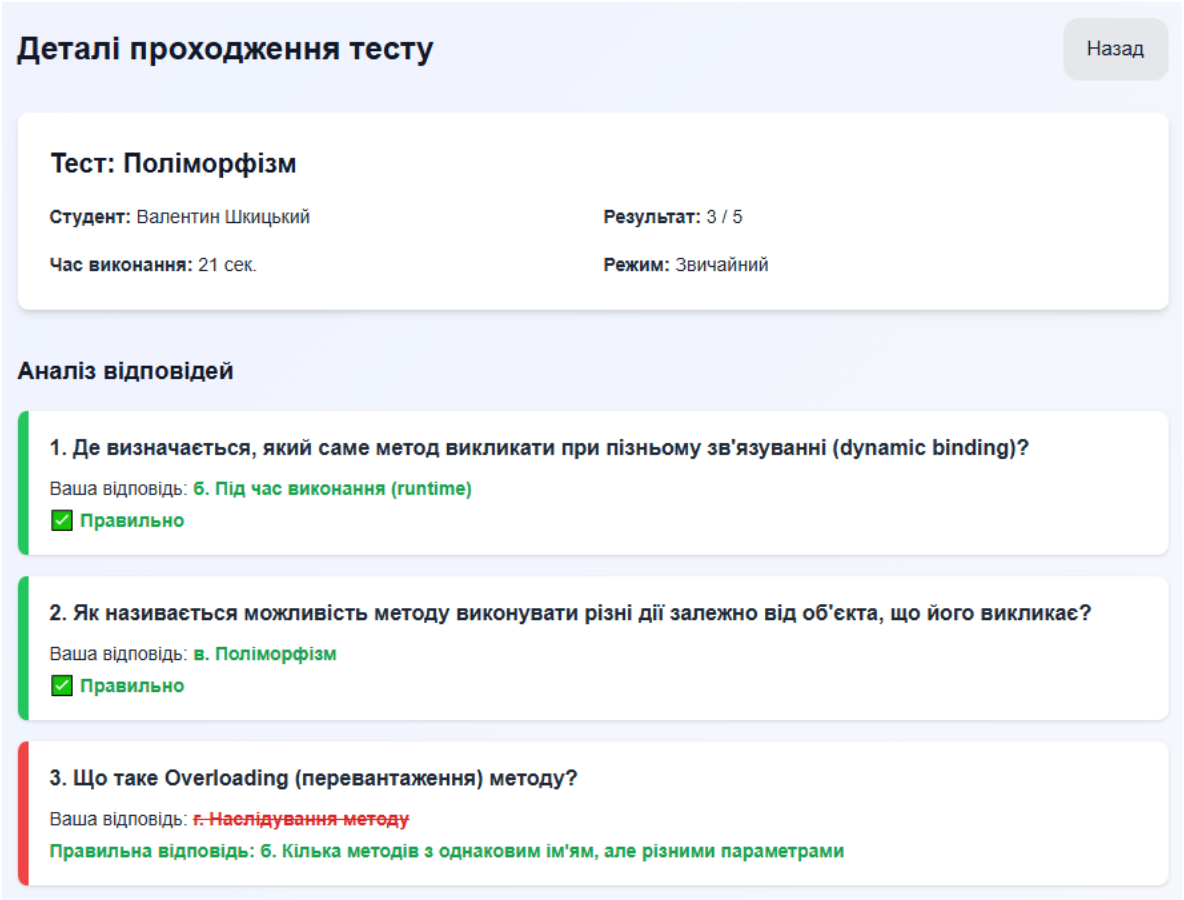



Рисунок 4.12 – Детальна статистика результатів студента

### 4.3.3. Інструкція для Адміністратора

Роль адміністратора призначена для здійснення технічної підтримки, конфігурування системних параметрів та глобального моніторингу працездатності платформи автоматизованого тестування. Після успішної автентифікації користувач із роллю ADMIN за допомогою механізмів маршрутизації автоматично перенаправляється на головну сторінку керування.

Нижче наведено детальний опис функціональних можливостей та інструкцію з використання кожного розділу панелі адміністратора:

1.**Панель адміністратора (Admin Hub) -** Головне вікно адміністратора що забезпечує доступ до глобальних налаштувань(Рис. 4.13).

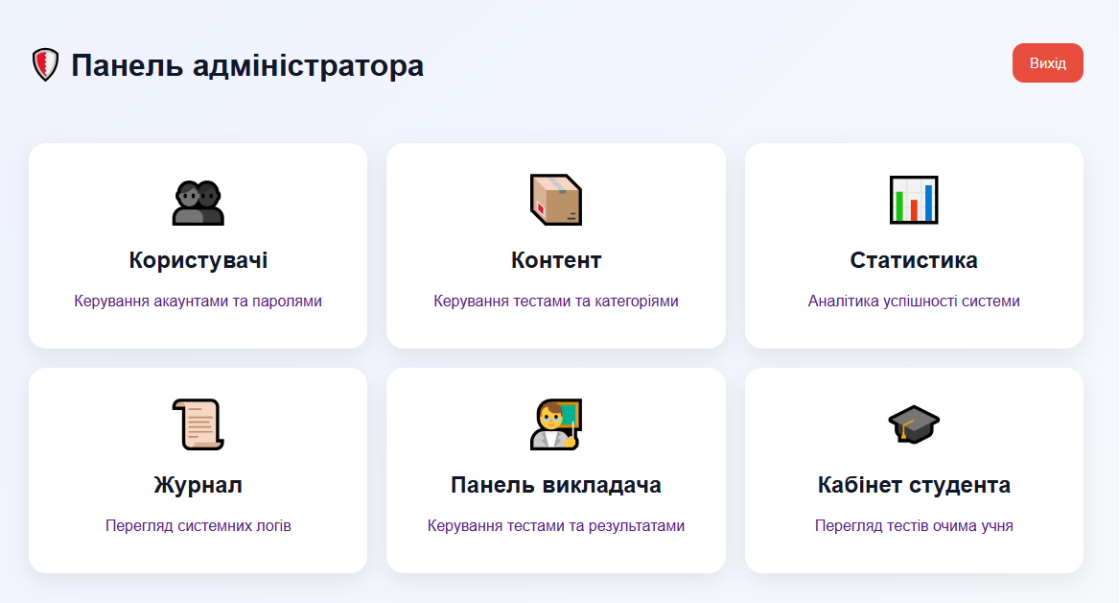



Рисунок 4.13 – Панель адміністратора

2.**Управління користувачами -** Повний список усіх зареєстрованих у базі даних осіб. Адміністратор може виконувати такі дії(Рис. 4.14):

1. Перегляд детальної інформації профілів користувачів.
2. Редагування облікових даних або надання/зміна ролі користувача.
3. Тимчасове блокування чи повне видалення облікових.

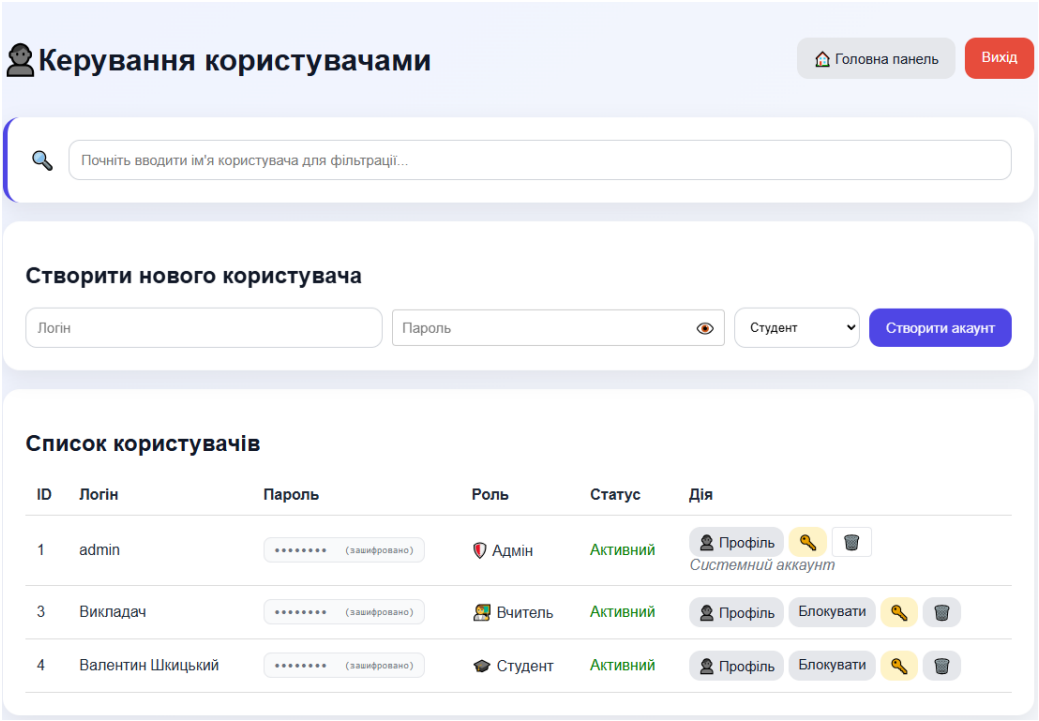



Рисунок 4.14 – Панель Управління користувачами

3.**Аудит системи** - У розділі "Журнал" адміністратор може переглядати історію дій у системі (час входу користувачів, створення нових тестів тощо), що допомагає виявляти підозрілу активність(Рис. 4.6).

4.**Контент** - У розділі "Контент" адміністратор має доступ до всіх навчальних матеріалів. Він може переглядати всі тести та питання, які створили викладачі. За потреби адміністратор може видалити застарілі тести(Рис. 4.15).

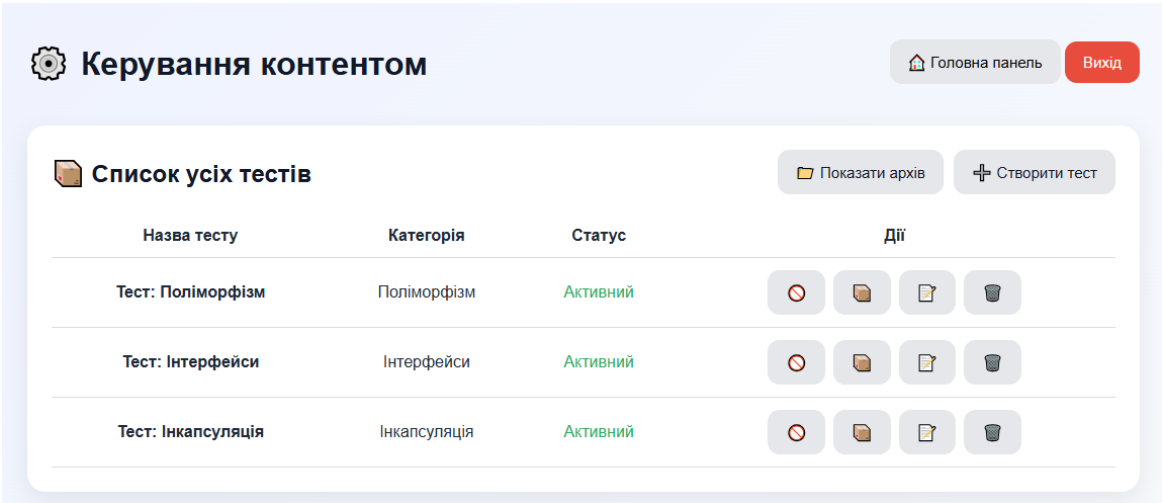



Рисунок 4.15 – Панель Керування контентом

5.**Статистика та інше** - У розділі "Статистика та інше" адміністратор може переглядати статистику проходження тестів студентами та їх успішність. а також має функції Системних налаштувань та Керування безпекою(Рис. 4.16).

**Системні налаштування:**

1. **"Режим технічних робіт"** - дозволяє закривати/відкривати доступ до сервісу для певних технічних маніпуляцій адміністратором.
2. **"Доступу до чату"** – закриває/відкриває функцію використання чату для користувачів.

**Керування безпекою:**

1. **"Очистити чат"** – повністю видаляє всі повідомлення в чаті.
2. **"Завершити всі сесії"** – всі сесії користувачів будуть завершені.

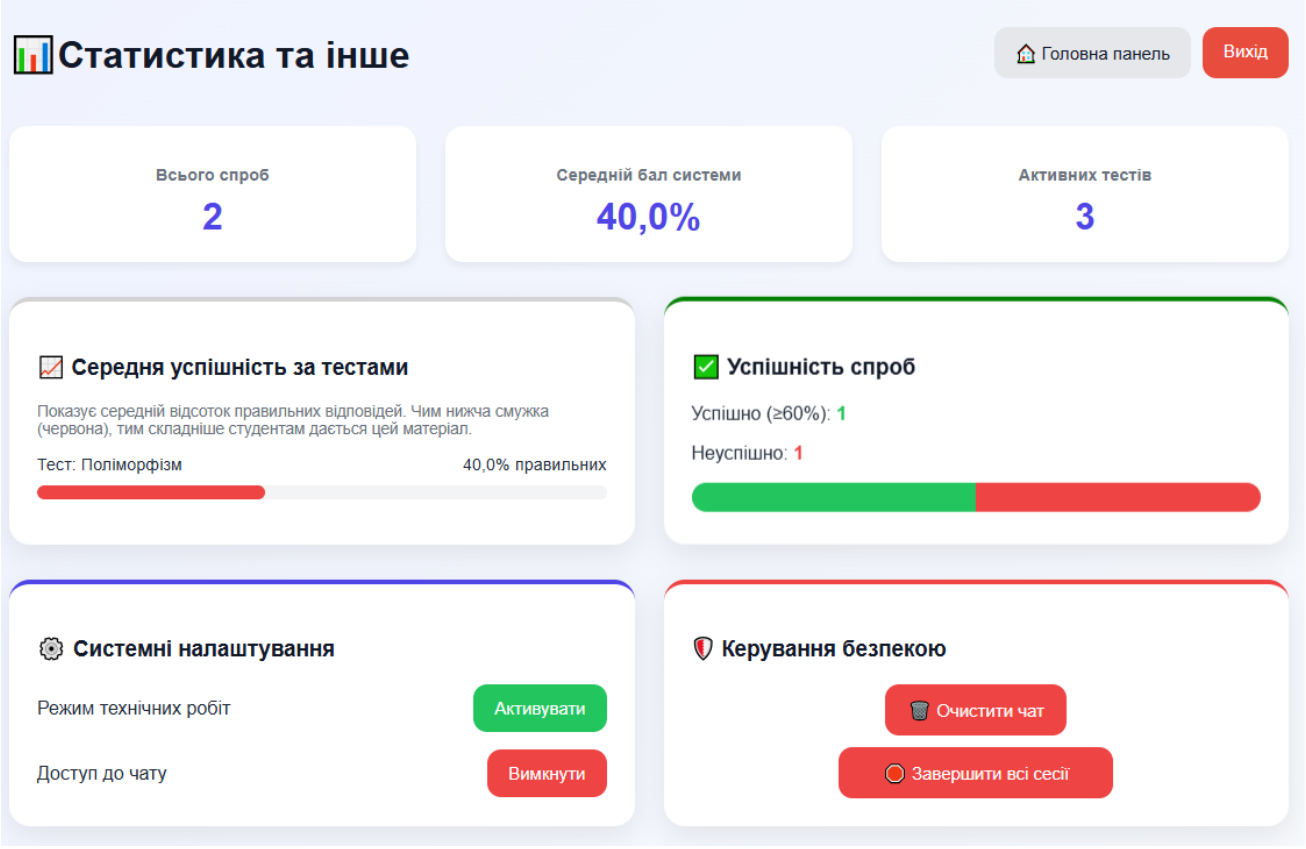



Рисунок 4.16 – Панель Статистика та інше

6.**Панель викладача -** Адміністратор має доступ до інтерфейсу викладачів, щоб оперативно допомагати їм у разі труднощів. Наприклад, якщо викладач не може розібратися, як згенерувати питання чи налаштувати тест, адміністратор може зайти і допомогти йому технічно.

7.**Кабінет студента -** Адміністратор також може бачити систему очима студентів, що допомагає вирішувати технічні проблеми. Наприклад, якщо під час тесту у студента зник інтернет або вимкнулося світло, адміністратор може перевірити його збережений результат і, за необхідності, надати технічну можливість перескласти тест.

Також для всіх користувачів системи доступний модуль "Чат підтримки" (chat.html), де можна обмінюватися повідомленнями щодо навчального процесу та повідомляти про помилки та несправності системи(Рис.4. 17).

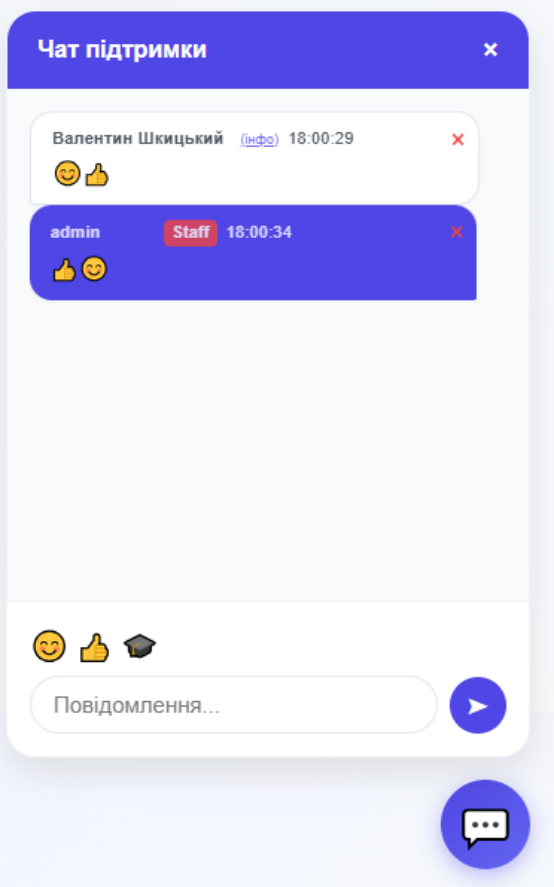



Рисунок 4.17 – Чат підтримки.

# ВИСНОВОК

У ході виконання кваліфікаційної роботи було розроблено програмне забезпечення для тестування студентів з дисципліни «Об’єктно-орієнтоване програмування». Створена система є повнофункціональним веб-застосунком, який автоматизує процес

перевірки знань, забезпечує об'єктивність оцінювання та спрощує взаємодію між викладачами та студентами.

**Під час роботи було вирішено наступні завдання:**

1. **Проведено** аналіз існуючих систем комп'ютерного тестування та обґрунтовано необхідність розробки спеціалізованого рішення для дисципліни ООП.

2. **Обрано** сучасний стек технологій на базі мови Java та фреймворку Spring Boot, що забезпечує високу продуктивність, масштабованість та безпеку застосунку. Для збереження даних використано об'єктно-реляційне відображення Hibernate та базу даних SQL.

3. **Спроектовано** архітектуру системи, що включає три основні ролі: студент (проходження тестів), викладач (керування контентом та перегляд статистики) та адміністратор (керування користувачами та системний аудит).

4. **Реалізовано** програмні модулі для автентифікації, динамічної генерації питань, обробки результатів тестування та візуалізації аналітичних даних.

5. **Проведено** тестування розробленого ПЗ, яке підтвердило стабільність роботи всіх функціональних блоків та відповідність вимогам технічного завдання.

6. **Розроблена** система дозволяє викладачам створювати тестові завдання різного рівня складності, а студентам – отримувати миттєвий зворотний зв’язок щодо рівня своїх знань.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Бублик В. В. Об'єктно-орієнтоване програмування : підручник. – Київ : ІТ-книга, 2015. – 624 с.
2. Алхімова С. М. Об'єктно-орієнтоване програмування. Частина 2. Об'єктно-орієнтований підхід до розробки програмного забезпечення : підручник. – Київ : КПІ ім. Ігоря Сікорського, 2019. – 192 с.
3. Каплун В. А. Основи Web-програмування : навчальний посібник. – Вінниця : ВНТУ, 2023. – 128 с.
4. Романюк О. Н., Савчук Т. О. Організація баз даних і знань : навчальний посібник. – Вінниця : ВНТУ, 2001. – 236 с.
5. Смагіна О. О., Переяславська С. О. Якість програмного забезпечення та тестування : навчальний посібник. – Старобільськ : ДЗ «ЛНУ імені Тараса Шевченка», 2021. – 134 с.
6. Олексійчук Ю. Ф., Ольховська О. В., Ольховський Д. М., Орлова Д. І. Проєктування та розробка web-сервісу для генерування та розсилки pdf-документів. Системи та технології. – 2023. – № 1 (65). – С. 39–45. URL: https://doi.org/10.32782/2521-6643-2023.1-65.5
7. Олексійчук Ю. Ф., Ольховський Д. М., Ольховська О. В., Андрушків О. М. Проєктування, розробка та тестування web-сервісу для вибору тем дипломних робіт. Системи та технології. – 2024. – № 1 (67). – С. 43–50. URL: https://doi.org/10.32782/2521-6643-2024-1-67.7
8. Кравець П. О. Об'єктно-орієнтоване програмування : навчальний посібник. – Львів : Видавництво Львівської політехніки, 2012. – 624 с.
9. Гришанович Т. О., Глинчук Л. Я. Основи об’єктно-орієнтованого

програмування : навчальне видання. – Луцьк : ВНУ імені Лесі Українки, 2022. – 120 с.

10. Триус Ю. В., Герасименко І. В., Франчук В. М. Система електронного навчання ВНЗ на базі MOODLE : методичний посібник. – Черкаси, 2012. – 220 с.

11. Oracle. Java Platform, Standard Edition Documentation. – URL: https://docs.oracle.com/javase/

12. Oracle. MySQL Reference Manual. – URL: https://dev.mysql.com/doc/

13. Sommerville I. Software Engineering. – 10th ed. – Pearson Education, 2015. – 816 p.

14. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. – 3rd ed. – Addison-Wesley Professional, 2003. – 208 p.

15. Moodle : офіційний сайт системи управління навчанням. – URL: https://moodle.org

16. Google Classroom : офіційна сторінка сервісу. – URL: https://classroom.google.com

17. ClassMarker : Online Quiz Maker. – URL: https://www.classmarker.com

18. ProProfs Quiz Maker : онлайн-сервіс для створення тестів. – URL: https://www.proprofs.com/quiz-school

19. Thymeleaf : Java-шаблонізатор для вебзастосунків. – URL: https://www.thymeleaf.org

20. Spring Boot : фреймворк для створення Java-застосунків. – URL: https://spring.io/projects/spring-boot

21. HTML Living Standard : специфікація мови розмітки HTML. – URL: https://html.spec.whatwg.org

22. CSS : Cascading Style Sheets. – URL: https://www.w3.org/Style/CSS

23. IntelliJ IDEA : інтегроване середовище розробки. – URL: https://www.jetbrains.com/idea

# Додаток А

Package config/DataInitializer

```
package com.example.oopp.config;

import com.example.oopp.model.User;
import com.example.oopp.repository.UserRepository;
import org.springframework.boot.CommandLineRunner;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
@Configuration
public class DataInitializer {
   @Bean
   CommandLineRunner initUsers(UserRepository userRepo, BCryptPasswordEncoder encoder) {
      return args -> {
         if (userRepo.count() == 0) {
            userRepo.save(new User(null, "admin", encoder.encode("admin123"), "ADMIN"));
            userRepo.save(new User(null, "ivan", encoder.encode("1234"), "STUDENT"));
            System.out.println("Користувачі створені з шифруванням BCrypt");

         }
      };

   }
}
```

class SecurityConfig

```
package com.example.oopp.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@Configuration
@EnableWebSecurity
public class SecurityConfig {
   @Bean
   public BCryptPasswordEncoder passwordEncoder() {
      return new BCryptPasswordEncoder();
   }
   @Bean
   public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
      http
            .csrf(csrf -> csrf.disable())
            .headers(headers -> headers.frameOptions(f -> f.disable()))
```

```
.authorizeHttpRequests(auth -> auth
        .requestMatchers( "/style.css", "/js/**").permitAll()
        .requestMatchers("/student/**", "/teacher/**", "/admin/**").permitAll()
        .anyRequest().permitAll()
```

```
        return http.build();
    }
}
class WebConfig
package com.example.oopp.config;

import com.example.oopp.service.UserStatusInterceptor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class WebConfig implements WebMvcConfigurer {
    @Autowired
    private UserStatusInterceptor userStatusInterceptor;
    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(userStatusInterceptor)
                .addPathPatterns("/**")
                .excludePathPatterns("/login", "/register", "/style.css", "/logout");
    }
}
package Controller
class AdminController
package com.example.oopp.controller;

import com.example.oopp.model.Result;
import com.example.oopp.model.SystemSettings;
import com.example.oopp.model.Test;
import com.example.oopp.model.User;
import com.example.oopp.service.*;
import jakarta.servlet.http.HttpSession;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.bind.annotation.ModelAttribute;

import java.util.List;
import java.util.stream.Collectors;

@Controller
@RequestMapping("/admin")
public class AdminController {
    private final ResultService resultService;
    private final TestService testService;
    private final AuthService authService;
    private final AuditService auditService;
    private final ChatService chatService;

    public AdminController(ResultService resultService, TestService testService,
                           AuthService authService, AuditService auditService,
                           ChatService chatService) {
```

```
    this.resultService = resultService;
    this.testService = testService;
    this.authService = authService;
    this.auditService = auditService;
    this.chatService = chatService;
}
@GetMapping("/stats")
public String stats(HttpSession session, Model model) {
    if (session.getAttribute("admin") == null) return "redirect:/login";
    model.addAttribute("totalAttempts", resultService.getTotalAttempts());
    model.addAttribute("averageScore", String.format("%.1f", resultService.getAverageScorePercent()));
    model.addAttribute("testsCount", testService.findAll().size());
    model.addAttribute("attemptsByTest", resultService.getAttemptsByTest());
    model.addAttribute("successStats", resultService.getGlobalSuccessStats());
    model.addAttribute("difficultyRating", resultService.getDifficultyRating());
    return "admin-stats";
}
@GetMapping("/users")
public String manageUsers(HttpSession session, Model model) {
    if (session.getAttribute("admin") == null) return "redirect:/login";
    model.addAttribute("users", authService.findAll());
    return "admin-users";
}
@PostMapping("/create-user")
public String createUser(@RequestParam String username, @RequestParam String password,
                         @RequestParam String role, HttpSession session) {
    if (session.getAttribute("admin") == null) return "redirect:/login";
    authService.register(username, password, role);
    auditService.log("Створено користувача: " + username);
    return "redirect:/admin/users";
}
@GetMapping("/block-user/{id}")
public String blockUser(@PathVariable Long id, HttpSession session) {
    if (session.getAttribute("admin") == null) return "redirect:/login";
    authService.toggleBlockUser(id);
    return "redirect:/admin/users";
}
@GetMapping("/delete-user/{id}")
public String deleteUser(@PathVariable Long id, HttpSession session) {
    if (session.getAttribute("admin") == null) return "redirect:/login";
    authService.deleteUser(id);
    return "redirect:/admin/users";
}
@GetMapping("/logs")
public String viewLogs(HttpSession session, Model model) {
    if (session.getAttribute("admin") == null) return "redirect:/login";
    model.addAttribute("logs", auditService.getLogs());
    return "admin-logs";
}
@GetMapping("/content")
public String manageContent(@RequestParam(defaultValue = "false") boolean showArchived,
                            Model model, HttpSession session) {
    if (session.getAttribute("admin") == null) return "redirect:/login";
```

```
        List<Test> allTests = testService.findAll();
        model.addAttribute("tests", allTests);
        List<Test> studentViewTests = allTests.stream()
                .filter(t -> !t.isArchived() && !t.isHidden())
                .collect(Collectors.toList());
        model.addAttribute("studentTests", studentViewTests);
        model.addAttribute("showArchived", showArchived);
        return "admin-content";
    }
    @GetMapping("/reset-password/{id}")
    public String resetPassword(@PathVariable Long id, HttpSession session) {
        if (session.getAttribute("admin") == null) return "redirect:/login";
        authService.resetPassword(id);
        auditService.log("Скинуто пароль для користувача з ID: " + id);
        return "redirect:/admin/users";
    }
    @GetMapping("/test/archive/{id}")
    public String archiveTest(@PathVariable Long id, HttpSession session) {
        if (session.getAttribute("admin") == null) return "redirect:/login";
        testService.findById(id).ifPresent(test -> {
            test.setArchived(!test.isArchived());
            testService.save(test);
            String status = test.isArchived() ? "архівовано" : "активовано";
            auditService.log("Тест '" + test.getTitle() + "' " + status);
        });
        return "redirect:/admin/content";
    }
    @PostMapping("/test/set-category")
    public String setCategory(@RequestParam Long testId, @RequestParam String category, HttpSession
session) {
        if (session.getAttribute("admin") == null) return "redirect:/login";
        testService.findById(testId).ifPresent(test -> {
            test.setCategory(category);
            testService.save(test);
            auditService.log("Змінено категорію тесту '" + test.getTitle() + "' на: " + category);
        });
        return "redirect:/admin/content";
    }
    @GetMapping("/settings/toggle-chat")
    public String toggleChat(HttpSession session) {
        if (session.getAttribute("admin") == null) return "redirect:/login";
        SystemSettings.chatEnabled = !SystemSettings.chatEnabled;
        auditService.log("Статус чату змінено. Тепер: " + (SystemSettings.chatEnabled ? "Активний" :
"Вимкнено"));
        return "redirect:/admin/stats";
    }
    @GetMapping("/settings/maintenance")
    public String toggleMaintenance(HttpSession session) {
        if (session.getAttribute("admin") == null) return "redirect:/login";
        SystemSettings.maintenanceMode = !SystemSettings.maintenanceMode;
        auditService.log("Режим технічних робіт змінено. Тепер: " + (SystemSettings.maintenanceMode ?
"Увімкнено" : "Вимкнено"));
        return "redirect:/admin/stats";
```

```
    }
    @GetMapping("/sessions/terminate-all")
    public String terminateAllSessions(HttpSession session) {
        if (session.getAttribute("admin") == null) return "redirect:/login";
        authService.invalidateAllSessions();
        auditService.log("Всі поточні сесії користувачів були примусово завершені.");
        return "redirect:/admin/stats";
    }
    @GetMapping("/chat/clear")
    public String clearChat(HttpSession session) {
        if (session.getAttribute("admin") == null) return "redirect:/login";
        chatService.clearAllMessages();
        auditService.log("Історія загального чату була повністю очищена.");
        return "redirect:/admin/stats";
    }
    @GetMapping("")
    public String adminHub(HttpSession session) {
        if (session.getAttribute("admin") == null) return "redirect:/login";
        return "admin-hub";
    }
    @ModelAttribute("currentUri")
    public String getCurrentUri(jakarta.servlet.http.HttpServletRequest request) {
        return request.getRequestURI();
    }
    @GetMapping("/users/{username}")
    public String viewUserDetailsAdmin(@PathVariable("username") String username, HttpSession session,
Model model) {
        if (session.getAttribute("admin") == null) return "redirect:/login";
        User user = authService.findAll().stream()
                .filter(u -> u.getUsername().equals(username))
                .findFirst().orElse(null);
        if (user == null) return "redirect:/admin/users";
        List<Result> userResults = resultService.getResultsByStudent(username);
        model.addAttribute("targetUser", user);
        model.addAttribute("results", userResults);
        double avg = userResults.stream()
                .filter(r -> r.getTotalQuestions() > 0)
                .mapToDouble(r -> (double) r.getScore() / r.getTotalQuestions() * 100)
                .average().orElse(0);
        model.addAttribute("averageScore", String.format("%.1f", avg));

        return "user-details";
    }
    @PostMapping("/users/{username}/delete-result/{resultId}")
    public String deleteResultAdmin(@PathVariable("username") String username,
                                    @PathVariable("resultId") Long resultId,
                                    HttpSession session) {
        if (session.getAttribute("admin") == null) return "redirect:/login";
        resultService.deleteResultById(resultId);
        auditService.log("Адміністратор видалив результат тесту (ID: " + resultId + ") студента " + username);
        return "redirect:/admin/users/" + username;
    }
    @PostMapping("/delete-admin/{id}")
```

```
    public String deleteAdminSecure(@PathVariable Long id,
                                    @RequestParam("adminPassword") String adminPassword,
                                    HttpSession session) {
        if (session.getAttribute("admin") == null) return "redirect:/login";
        String currentUsername = (String) session.getAttribute("username");
        User verifiedAdmin = authService.login(currentUsername, adminPassword);
        if (verifiedAdmin == null) {
            return "redirect:/admin/users?error=wrong_password";
        }
        authService.deleteUser(id);
        auditService.log("Захищене видалення: видалено адміністратора з ID " + id);
        return "redirect:/admin/users";
    }
}
class AuthController
package com.example.oopp.controller;

import com.example.oopp.model.SystemSettings;
import com.example.oopp.model.User;
import com.example.oopp.service.AuthService;
import jakarta.servlet.http.Cookie;
import jakarta.servlet.http.HttpSession;
import jakarta.servlet.http.HttpServletResponse;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

@Controller
public class AuthController {
    private final AuthService authService;
    public AuthController(AuthService authService) {
        this.authService = authService;
    }
    @GetMapping("/login")
    public String loginPage(@RequestParam(value = "error", required = false) String error, Model model) {
        model.addAttribute("isMaintenance", SystemSettings.maintenanceMode);
        if ("blocked".equals(error)) {
            model.addAttribute("errorText", "Ваш доступ було припинено або акаунт видалено.");
        } else if ("maintenance".equals(error)) {
            model.addAttribute("errorText", "На сайті проводяться технічні роботи.");
        }
        return "login";
    }
    @PostMapping("/login")
    public String doLogin(@RequestParam String username,
                          @RequestParam String password,
                          @RequestParam(required = false) String remember,
                          HttpServletResponse response,
                          HttpSession session,
                          Model model) {
        session.removeAttribute("admin");
        session.removeAttribute("teacher");
        session.removeAttribute("student");
```

```
        session.removeAttribute("username");
        User user = authService.login(username, password);
        if (user != null) {
            if (SystemSettings.maintenanceMode && !"ADMIN".equals(user.getRole())) {
                return "redirect:/login?error=maintenance";
            }
            if (user.isBlocked()) {
                model.addAttribute("error", "Ваш акаунт заблоковано адміністратором.");
                return "login";
            }
            session.setAttribute("username", username);
            session.setAttribute("sessionVersion", user.getSessionVersion());
            if ("on".equals(remember)) {
                Cookie cookie = new Cookie("remember_user", username);
                cookie.setMaxAge(30 * 24 * 60 * 60);
                cookie.setPath("/");
                response.addCookie(cookie);
            }
            if ("ADMIN".equals(user.getRole())) {
                session.setAttribute("admin", true);
                session.setAttribute("teacher", true);
                return "redirect:/admin";
            } else if ("TEACHER".equals(user.getRole())) {
                session.setAttribute("teacher", true);
                return "redirect:/teacher";
            } else {
                session.setAttribute("student", true);
                return "redirect:/student";
            }
        }
        model.addAttribute("error", "Невірний логін або пароль");
        return "login";
    }
    @GetMapping("/logout")
    public String logout(HttpSession session, HttpServletResponse response) {
        session.invalidate();
        Cookie cookie = new Cookie("remember_user", "");
        cookie.setMaxAge(0);
        cookie.setPath("/");
        response.addCookie(cookie);
        return "redirect:/";
    }
}
class ChatController
package com.example.oopp.controller;

import com.example.oopp.service.ChatService;
import jakarta.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
```

```
@Controller
@RequestMapping("/chat")
public class ChatController {
   @Autowired private ChatService chatService;
   @PostMapping("/send")
   public String sendMessage(@RequestParam String content, HttpSession session) {
      String username = (String) session.getAttribute("username");
      if (username == null) return "redirect:/login";
      String role = "STUDENT";
      if (session.getAttribute("admin") != null) role = "ADMIN";
      else if (session.getAttribute("teacher") != null) role = "TEACHER";
      chatService.addMessage(username, content, role);
      String referer = (String) session.getAttribute("lastPage");
      return "redirect:" + (referer != null ? referer : "/student");
   }
   @GetMapping("/delete/{id}")
   @ResponseBody
   public org.springframework.http.ResponseEntity<String> deleteMessage(@PathVariable Long id,
HttpSession session) {
      if (session.getAttribute("admin") != null || session.getAttribute("teacher") != null) {
         chatService.deleteMessage(id);
         return org.springframework.http.ResponseEntity.ok("Deleted");
      }
      return org.springframework.http.ResponseEntity.status(403).body("Error");
   }
   @GetMapping("/updates")
   public String getChatUpdates(Model model) {
      model.addAttribute("messages", chatService.getMessages());
      return "fragments :: messageList";
   }
}
class HomeController
package com.example.oopp.controller;

import com.example.oopp.model.User;
import com.example.oopp.service.AuthService;
import jakarta.servlet.http.Cookie;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpSession;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class HomeController {

   private final AuthService authService;
   public HomeController(AuthService authService) {
      this.authService = authService;
   }
   @GetMapping("/")
   public String home(HttpServletRequest request, HttpSession session) {
      if (session.getAttribute("username") != null) {
         return getRedirectByRole(session);
```

```
        }
        Cookie[] cookies = request.getCookies();
        if (cookies != null) {
            for (Cookie c : cookies) {
                if ("remember_user".equals(c.getName())) {
                    String username = c.getValue();
                    User user = authService.findAll().stream()
                            .filter(u -> u.getUsername().equals(username))
                            .findFirst().orElse(null);
                    if (user != null) {
                        session.setAttribute("username", user.getUsername());
                        if ("TEACHER".equals(user.getRole())) session.setAttribute("teacher", true);
                        else session.setAttribute("student", true);
                        return getRedirectByRole(session);
                    }
                }
            }
        }
        return "index";
    }
    private String getRedirectByRole(HttpSession session) {
        if (session.getAttribute("admin") != null) return "redirect:/admin/users";
        if (session.getAttribute("teacher") != null) return "redirect:/teacher";
        return "redirect:/student";
    }
}
class StudentController
package com.example.oopp.controller;

import com.example.oopp.model.*;
import com.example.oopp.service.*;
import jakarta.servlet.http.HttpSession;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import java.util.*;
import java.util.stream.Collectors;

@Controller
@RequestMapping("/student")
@SessionAttributes({"sessionQuestions", "testStartTime"})
public class StudentController {
    private final TestService testService;
    private final ResultService resultService;
    private final AuditService auditService;

    public StudentController(TestService testService, ResultService resultService, AuditService auditService) {
        this.testService = testService;
        this.resultService = resultService;
        this.auditService = auditService;
    }
    @GetMapping
    public String studentDashboard(HttpSession session, Model model) {
```

```
    if (session.getAttribute("username") == null) return "redirect:/login";
    String username = (String) session.getAttribute("username");

    List<Test> activeTests = testService.findAll().stream()
            .filter(t -> !t.isArchived() && !t.isHidden())
            .collect(Collectors.toList());
    List<Result> myResults = resultService.getResultsByStudent(username);
    int maxScore = myResults.stream()
            .mapToInt(Result::getScore)
            .max()
            .orElse(0);
    model.addAttribute("tests", activeTests);
    model.addAttribute("myResults", myResults);
    model.addAttribute("maxScore", maxScore);

    return "student";
}
@GetMapping("/test/{id}")
public String takeTest(@PathVariable Long id, HttpSession session, Model model) {
    if (session.getAttribute("username") == null) return "redirect:/login";
    Test test = testService.findById(id).orElseThrow();
    model.addAttribute("test", test);
    model.addAttribute("sessionQuestions", test.getQuestions());
    model.addAttribute("testStartTime", System.currentTimeMillis());
    return "test";
}
@PostMapping("/submit/{id}")
public String submitTest(@PathVariable Long id,
                         @RequestParam Map<String, String> answers,
                         @ModelAttribute("sessionQuestions") List<Question> sessionQuestions,
                         @ModelAttribute("testStartTime") Long testStartTime,
                         HttpSession session, Model model) {
    String username = (String) session.getAttribute("username");
    Test test = testService.findById(id).orElseThrow();
    List<Result.AnswerDetail> details = new ArrayList<>();
    int score = 0;
    String[] letters = {"а. ", "б. ", "в. ", "г. ", "д. "};
    for (int i = 0; i < sessionQuestions.size(); i++) {
        Question q = sessionQuestions.get(i);
        String rawAnswer = answers.getOrDefault("q" + i, "-1");
        int userAnswerIdx = Integer.parseInt(rawAnswer);
        int correctIdx = q.getCorrectAnswerIndex();
        boolean isCorrect = (userAnswerIdx == correctIdx);
        if (isCorrect) score++;
        String formattedCorrect = letters[correctIdx] + q.getOptions().get(correctIdx);
        String studentDisplay = (userAnswerIdx != -1)
                ? letters[userAnswerIdx] + q.getOptions().get(userAnswerIdx)
                : "Немає відповіді";
        details.add(new Result.AnswerDetail(
                q.getText(),
                studentDisplay,
                formattedCorrect,
                isCorrect
```

```
            ));
        }
        Result result = new Result(username, test.getTitle(), score, sessionQuestions.size());
        result.setDetails(details);
        result.setTimeSpentSeconds((System.currentTimeMillis() - testStartTime) / 1000);
        result.setMode("standard");
        resultService.save(result);
        model.addAttribute("result", result);
        return "results";
    }
    @GetMapping("/result/{id}")
    public String showResultDetails(@PathVariable Long id, Model model, HttpSession session) {
        if (session.getAttribute("username") == null) return "redirect:/login";
        Result result = resultService.getResultById(id);
        if (result == null) return "redirect:/student";
        model.addAttribute("result", result);
        return "result-details";
    }
    @PostMapping("/result/delete/{id}")
    public String deleteResult(@PathVariable("id") Long id, HttpSession session) {
        if (session.getAttribute("admin") == null) {
            return "redirect:/student";
        }

        resultService.deleteResultById(id);
        return "redirect:/student";
    }
}
class TeacherController
package com.example.oopp.controller;

import com.example.oopp.model.*;
import com.example.oopp.repository.UserRepository;
import com.example.oopp.service.*;
import jakarta.servlet.http.HttpSession;
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.List;
import java.util.stream.Collectors;

@Controller
@RequestMapping("/teacher")
public class TeacherController {
    private final TestService testService;
    private final ResultService resultService;
    private final UserRepository userRepository;

    public TeacherController(TestService testService, ResultService resultService, UserRepository
userRepository) {
        this.testService = testService;
        this.resultService = resultService;
```

```
    this.userRepository = userRepository;
}
private boolean isTeacher(HttpSession session) {
    return session.getAttribute("teacher") != null || session.getAttribute("admin") != null;
}

@GetMapping
public String teacherDashboard(@RequestParam(defaultValue = "false") boolean showArchived,
                               HttpSession session, Model model) {
    if (!isTeacher(session)) return "redirect:/login";
    List<Test> tests = testService.findAll().stream()
            .filter(t -> showArchived || !t.isArchived())
            .collect(Collectors.toList());
    model.addAttribute("tests", tests);
    model.addAttribute("showArchived", showArchived);
    model.addAttribute("results", resultService.findAll());
    return "teacher";
}
@PostMapping("/test/archive/{id}")
public String archiveTest(@PathVariable Long id, HttpSession session) {
    if (!isTeacher(session)) return "redirect:/login";
    testService.findById(id).ifPresent(t -> {
        t.setArchived(true);
        testService.save(t);
    });
    return "redirect:/teacher";
}
@PostMapping("/test/unarchive/{id}")
public String unarchiveTest(@PathVariable Long id, HttpSession session) {
    if (!isTeacher(session)) return "redirect:/login";
    testService.findById(id).ifPresent(t -> {
        t.setArchived(false);
        testService.save(t);
    });
    return "redirect:/teacher?showArchived=true";
}
@PostMapping("/test/toggle-visibility/{id}")
public String toggleVisibility(@PathVariable Long id, HttpSession session) {
    if (!isTeacher(session)) return "redirect:/login";
    testService.findById(id).ifPresent(t -> {
        t.setHidden(!t.isHidden());
        testService.save(t);
    });
    return "redirect:/teacher";
}
@GetMapping("/test/new")
public String showCreateForm(HttpSession session, Model model) {
    if (!isTeacher(session)) return "redirect:/login";
    model.addAttribute("test", new Test());
    return "create-test";
}
@GetMapping("/test/edit/{id}")
public String showEditForm(@PathVariable Long id, HttpSession session, Model model) {
```

```
    if (!isTeacher(session)) return "redirect:/login";
    testService.findById(id).ifPresent(t -> model.addAttribute("test", t));
    return "create-test";
}
@PostMapping("/test/save")
public String saveTest(HttpSession session, @ModelAttribute Test test) {
    if (!isTeacher(session)) return "redirect:/login";
    testService.save(test);
    return "redirect:/teacher";
}
@GetMapping("/test/delete/{id}")
public String deleteTest(HttpSession session, @PathVariable Long id) {
    if (!isTeacher(session)) return "redirect:/login";
    testService.deleteById(id);
    return "redirect:/teacher";
}
@GetMapping("/user-stats/{username}")
public String viewUserStats(@PathVariable String username, HttpSession session, Model model) {
    if (!isTeacher(session)) return "redirect:/login";
    User user = userRepository.findByUsername(username).orElse(null);
    if (user == null) return "redirect:/teacher?error=not_found";
    List<Result> userResults = resultService.getResultsByStudent(username);
    model.addAttribute("targetUser", user);
    model.addAttribute("results", userResults);
    double avg = userResults.stream()
            .filter(r -> r.getTotalQuestions() > 0)
            .mapToDouble(r -> (double) r.getScore() / r.getTotalQuestions() * 100)
            .average().orElse(0);
    model.addAttribute("averageScore", String.format("%.1f", avg));
    return "user-details";
}
@GetMapping("/result/{id}")
public String viewResultDetails(@PathVariable Long id, Model model, HttpSession session) {
    if (!isTeacher(session)) return "redirect:/login";
    Result result = resultService.getResultById(id);
    if (result == null) return "redirect:/teacher";
    model.addAttribute("result", result);
    return "result-details";
}
@Autowired
private GeneratorService aiService;
@GetMapping("/test/generate-ai")
public String generateAiTest(@RequestParam String topic, HttpSession session) {
    if (session.getAttribute("teacher") == null) return "redirect:/login";
    try {
        List<Question> aiQuestions = aiService.generateQuestions(topic, 5);
        Test newTest = new Test();
        newTest.setTitle("Тест: " + topic);
        newTest.setCategory(topic);
        newTest.setQuestions(aiQuestions);
        testService.save(newTest);
    } catch (Exception e) {
        return "redirect:/teacher?error=ai_failed";
```

```
        }
        return "redirect:/teacher";
    }
    @GetMapping("/students")
    public String listStudents(HttpSession session, Model model) {
        if (session.getAttribute("teacher") == null && session.getAttribute("admin") == null) {
            return "redirect:/login";
        }
        List<User> students = userRepository.findAll().stream()
                .filter(u -> "STUDENT".equals(u.getRole()))
                .collect(Collectors.toList());
        model.addAttribute("students", students);
        return "teacher-students";
    }
    @PostMapping("/user-stats/{username}/delete-result/{resultId}")
    public String deleteResultTeacher(@PathVariable("username") String username,
                                      @PathVariable("resultId") Long resultId,
                                      HttpSession session) {
        if (!isTeacher(session)) return "redirect:/login";

        resultService.deleteResultById(resultId);
        return "redirect:/teacher/user-stats/" + username;
    }
}
package model
class AuditLog
package com.example.oopp.model;

import jakarta.persistence.*;

@Entity
@Table(name = "audit_logs")
public class AuditLog {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String action;
    private String timestamp;

    public AuditLog() {}

    public AuditLog(String action, String timestamp) {
        this.action = action;
        this.timestamp = timestamp;
    }
    public Long getId() { return id; }
    public String getAction() { return action; }
    public String getTimestamp() { return timestamp; }

    public void setAction(String action) { this.action = action; }
    public void setTimestamp(String timestamp) { this.timestamp = timestamp; }
}
```

class ChatMessage

```
package com.example.oopp.model;

import jakarta.persistence.*;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

@Entity
@Table(name = "chat_messages")
public class ChatMessage {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String sender;
    private String content;
    private String role;
    private String timestamp;

    public ChatMessage() {}

    public ChatMessage(String sender, String content, String role) {
        this.sender = sender;
        this.content = content;
        this.role = role;
        this.timestamp = LocalDateTime.now().format(DateTimeFormatter.ofPattern("HH:mm:ss"));
    }
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public String getSender() { return sender; }
    public void setSender(String sender) { this.sender = sender; }
    public String getContent() { return content; }
    public void setContent(String content) { this.content = content; }
    public String getRole() { return role; }
    public void setRole(String role) { this.role = role; }
    public String getTimestamp() { return timestamp; }
}
```

class Question

```
package com.example.oopp.model;

import jakarta.persistence.*;
import java.util.ArrayList;
import java.util.List;

@Entity
@Table(name = "questions")
public class Question {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String text;
    private int correctAnswerIndex;

    @ElementCollection(fetch = FetchType.EAGER)
```

```
    @CollectionTable(name = "question_options", joinColumns = @JoinColumn(name = "question_id"))
    @Column(name = "option_text")
    private List<String> options = new ArrayList<>();

    public Question() {}
    public Question(String text, List<String> options, int correctAnswerIndex) {
        this.text = text;
        this.options = options;
        this.correctAnswerIndex = correctAnswerIndex;
    }
    public String getText() { return text; }
    public void setText(String text) { this.text = text; }
    public List<String> getOptions() { return options; }
    public void setOptions(List<String> options) { this.options = options; }
    public int getCorrectAnswerIndex() { return correctAnswerIndex; }
    public void setCorrectAnswerIndex(int correctAnswerIndex) { this.correctAnswerIndex =
correctAnswerIndex; }
}
class Result
package com.example.oopp.model;

import jakarta.persistence.*;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

@Entity
@Table(name = "results")
public class Result {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String studentName;
    private String testTitle;
    private int score;
    private int totalQuestions;
    private int streak;
    private boolean perfectRun;

    private Long timeSpentSeconds;
    private String mode;
    private LocalDateTime dateTime = LocalDateTime.now();

    @ElementCollection(fetch = FetchType.EAGER)
    @CollectionTable(name = "result_details", joinColumns = @JoinColumn(name = "result_id"))
    private List<AnswerDetail> details = new ArrayList<>();

    public Result() {}

    public Result(String studentName, String testTitle, int score, int totalQuestions) {
        this.studentName = studentName;
        this.testTitle = testTitle;
```

```
        this.score = score;
        this.totalQuestions = totalQuestions;
        this.dateTime = LocalDateTime.now();
    }

    @Embeddable
    public static class AnswerDetail {
        private String questionText;
        private String studentAnswer;
        private String correctAnswer;
        private boolean isCorrect;

        public AnswerDetail() {}

        public AnswerDetail(String questionText, String studentAnswer, String correctAnswer, boolean isCorrect)
{
            this.questionText = questionText;
            this.studentAnswer = studentAnswer;
            this.correctAnswer = correctAnswer;
            this.isCorrect = isCorrect;
        }
        public String getQuestionText() { return questionText; }
        public String getStudentAnswer() { return studentAnswer; }
        public String getCorrectAnswer() { return correctAnswer; }
        public boolean isCorrect() { return isCorrect; }
    }
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public String getStudentName() { return studentName; }
    public void setStudentName(String studentName) { this.studentName = studentName; }
    public String getTestTitle() { return testTitle; }
    public void setTestTitle(String testTitle) { this.testTitle = testTitle; }
    public int getScore() { return score; }
    public void setScore(int score) { this.score = score; }
    public int getTotalQuestions() { return totalQuestions; }
    public void setTotalQuestions(int totalQuestions) { this.totalQuestions = totalQuestions; }
    public int getStreak() { return streak; }
    public void setStreak(int streak) { this.streak = streak; }
    public boolean isPerfectRun() { return perfectRun; }
    public void setPerfectRun(boolean perfectRun) { this.perfectRun = perfectRun; }

    public Long getTimeSpentSeconds() { return timeSpentSeconds; }
    public void setTimeSpentSeconds(Long timeSpentSeconds) { this.timeSpentSeconds = timeSpentSeconds; }

    public String getMode() { return mode; }
    public void setMode(String mode) { this.mode = mode; }

    public LocalDateTime getDateTime() { return dateTime; }
    public void setDateTime(LocalDateTime dateTime) { this.dateTime = dateTime; }

    public List<AnswerDetail> getDetails() { return details; }
    public void setDetails(List<AnswerDetail> details) { this.details = details; }
}
```

class StudentUser

```
package com.example.oopp.model;

public class StudentUser {
    private String username;
    private String password;

    public StudentUser(String username, String password) {
        this.username = username;
        this.password = password;
    }
    public String getUsername() {
        return username;
    }

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

    public String getPassword() {
        return password;
    }

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

class SystemSettings

```
package com.example.oopp.model;

import jakarta.persistence.*;

@Entity
@Table(name = "system_settings")
public class SystemSettings {
    @Id
    private String id = "SETTINGS";

    public static boolean maintenanceMode = false;
    public static boolean chatEnabled = true;

    public SystemSettings() {}
    public String getId() { return id; }
    public boolean isChatEnabled() { return chatEnabled; }
    public void setChatEnabled(boolean chatEnabled) { this.chatEnabled = chatEnabled; }
    public boolean isMaintenanceMode() { return maintenanceMode; }
    public void setMaintenanceMode(boolean maintenanceMode) { this.maintenanceMode = maintenanceMode;
}
}
```

class Test

```
package com.example.oopp.model;

import jakarta.persistence.*;
```

```
import java.util.ArrayList;
import java.util.List;

@Entity
@Table(name = "tests")
public class Test {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String title;
    private String category = "Загальне";
    private boolean archived = false;
    private boolean hidden = false;

    @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    @JoinColumn(name = "test_id")
    private List<Question> questions = new ArrayList<>();

    public Test() {}

    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }
    public List<Question> getQuestions() { return questions; }
    public void setQuestions(List<Question> questions) { this.questions = questions; }
    public String getCategory() { return category; }
    public void setCategory(String category) { this.category = category; }

    public boolean isArchived() { return archived; }
    public void setArchived(boolean archived) { this.archived = archived; }

    public boolean isHidden() { return hidden; }
    public void setHidden(boolean hidden) { this.hidden = hidden; }
}
class User
package com.example.oopp.model;

import jakarta.persistence.*;

@Entity
@Table(name = "users")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String username;
    private String password;
    private String role;
    private boolean blocked;
    private int sessionVersion = 0;
```

```
    public User() {}

    public User(Long id, String username, String password, String role) {
        this.id = id;
        this.username = username;
        this.password = password;
        this.role = role;
        this.blocked = false;
        this.sessionVersion = 0;
    }
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }

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

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

    public String getRole() { return role; }
    public void setRole(String role) { this.role = role; }

    public boolean isBlocked() { return blocked; }
    public void setBlocked(boolean blocked) { this.blocked = blocked; }

    public int getSessionVersion() { return sessionVersion; }
    public void setSessionVersion(int sessionVersion) { this.sessionVersion = sessionVersion; }
}
package/repository
class AuditLogRepository
package com.example.oopp.repository;

import com.example.oopp.model.AuditLog;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface AuditLogRepository extends JpaRepository<AuditLog, Long> {
}
class ChatMessageRepository
package com.example.oopp.repository;

import com.example.oopp.model.ChatMessage;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface ChatMessageRepository extends JpaRepository<ChatMessage, Long> {
}
class ResultRepository
package com.example.oopp.repository;

import com.example.oopp.model.Result;
```

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import java.util.List;

@Repository
public interface ResultRepository extends JpaRepository<Result, Long> {
   List<Result> findByStudentName(String studentName);
}
class TestRepository
package com.example.oopp.repository;

import com.example.oopp.model.Test;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface TestRepository extends JpaRepository<Test, Long> {
}
class UserRepository
package com.example.oopp.repository;

import com.example.oopp.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import java.util.Optional;

@Repository
public interface UserRepository extends JpaRepository<User, Long> {
   Optional<User> findByUsername(String username);
}
package/service
class AuditService
package com.example.oopp.service;

import com.example.oopp.model.AuditLog;
import com.example.oopp.repository.AuditLogRepository;
import org.springframework.stereotype.Service;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.List;
import java.util.stream.Collectors;

@Service
public class AuditService {
   private final AuditLogRepository auditLogRepository;

   public AuditService(AuditLogRepository auditLogRepository) {
      this.auditLogRepository = auditLogRepository;
   }
   public void log(String action) {
      String timestamp = LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss"));
      AuditLog newEntry = new AuditLog(action, timestamp);
```

```
        auditLogRepository.save(newEntry);
    }
    public List<String> getLogs() {
        return auditLogRepository.findAll().stream()
                .map(log -> "[" + log.getTimestamp() + "] " + log.getAction())
                .collect(Collectors.toList());
    }
}
class AuthService
package com.example.oopp.service;

import com.example.oopp.model.User;
import com.example.oopp.repository.UserRepository;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;
import java.util.List;

@Service
public class AuthService {
    private final UserRepository userRepository;
    private final BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();
    private int globalSessionVersion = 0;

    public AuthService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    public User login(String username, String password) {
        return userRepository.findByUsername(username)
                .filter(u -> passwordEncoder.matches(password, u.getPassword()))
                .orElse(null);
    }
    public boolean register(String username, String password, String role) {
        if (userRepository.findByUsername(username).isPresent()) return false;

        User newUser = new User();
        newUser.setUsername(username);
        newUser.setPassword(passwordEncoder.encode(password));
        newUser.setRole(role);
        userRepository.save(newUser);
        return true;
    }

    public List<User> findAll() {
        return userRepository.findAll();
    }

    public void deleteUser(Long id) {
        userRepository.deleteById(id);
    }

    public void toggleBlockUser(Long id) {
        userRepository.findById(id).ifPresent(user -> {
```

```
            if (!"ADMIN".equals(user.getRole())) {
                user.setBlocked(!user.isBlocked());
                userRepository.save(user);
            }
        });
    }
    public void resetPassword(Long userId) {
        userRepository.findById(userId).ifPresent(user -> {
            user.setPassword(passwordEncoder.encode("1234"));
            userRepository.save(user);
        });
    }
    public void invalidateAllSessions() {
        List<User> users = userRepository.findAll();
        for (User u : users) {
            u.setSessionVersion(u.getSessionVersion() + 1);
        }
        userRepository.saveAll(users);
    }
    public int getGlobalSessionVersion() {
        return globalSessionVersion;
    }
}
class ChatService
package com.example.oopp.service;

import com.example.oopp.model.ChatMessage;
import com.example.oopp.repository.ChatMessageRepository;
import org.springframework.stereotype.Service;
import java.util.List;

@Service
public class ChatService {
    private final ChatMessageRepository chatMessageRepository;

    public ChatService(ChatMessageRepository chatMessageRepository) {
        this.chatMessageRepository = chatMessageRepository;
    }
    public List<ChatMessage> getMessages() {
        return chatMessageRepository.findAll();
    }
    public void addMessage(String sender, String content, String role) {
        chatMessageRepository.save(new ChatMessage(sender, content, role));
    }
    public void deleteMessage(Long id) {
        chatMessageRepository.deleteById(id);
    }
    public void clearAllMessages() {
        chatMessageRepository.deleteAll();
    }
}
class GeneratorService
package com.example.oopp.service;
```

```
import com.example.oopp.model.Question;
import com.fasterxml.jackson.core.type.TypeReference;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.springframework.core.io.ClassPathResource;
import org.springframework.stereotype.Service;

import java.io.InputStream;
import java.util.*;
import java.util.stream.Collectors;

@Service
public class GeneratorService {

    private final ObjectMapper objectMapper = new ObjectMapper();

    public List<Question> generateQuestions(String topic, int count) throws Exception {
        ClassPathResource resource = new ClassPathResource("questions.json");
        InputStream inputStream = resource.getInputStream();
        List<Map<String, Object>> allData = objectMapper.readValue(inputStream,
                new TypeReference<List<Map<String, Object>>>() {});
        List<Map<String, Object>> topicQuestions = allData.stream()
                .filter(q -> q.get("category").toString().toLowerCase().contains(topic.toLowerCase()))
                .collect(Collectors.toList());
        List<Map<String, Object>> finalSelection = new ArrayList<>(topicQuestions);
        if (finalSelection.size() < count) {
            List<Map<String, Object>> remaining = new ArrayList<>(allData);
            remaining.removeAll(topicQuestions);
            Collections.shuffle(remaining);
            finalSelection.addAll(remaining.stream().limit(count - finalSelection.size()).collect(Collectors.toList()));
        }
        Collections.shuffle(finalSelection);
        List<Map<String, Object>> limitedSelection = finalSelection.stream()
                .limit(count)
                .collect(Collectors.toList());
        return limitedSelection.stream().map(data -> {
            Question q = new Question();
            q.setText((String) data.get("text"));
            q.setOptions((List<String>) data.get("options"));
            q.setCorrectAnswerIndex((Integer) data.get("correctAnswerIndex"));
            return q;
        }).collect(Collectors.toList());
    }
}
class ResultService
package com.example.oopp.service;

import com.example.oopp.model.Result;
import com.example.oopp.repository.ResultRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.*;
```

```
import java.util.stream.Collectors;

@Service
public class ResultService {

    @Autowired
    private ResultRepository resultRepository;

    public List<Result> findAll() {
        return resultRepository.findAll();
    }

    public void save(Result result) {
        resultRepository.save(result);
    }

    public List<Result> findByUser(com.example.oopp.model.User user) {
        return resultRepository.findByStudentName(user.getUsername());
    }

    public List<Result> getResultsByStudent(String username) {
        return resultRepository.findByStudentName(username);
    }

    public Result getResultById(Long id) {
        return resultRepository.findById(id).orElse(null);
    }
    public void deleteResultById(Long id) {
        resultRepository.deleteById(id);
    }

    public int getTotalAttempts() {
        return (int) resultRepository.count();
    }

    public double getAverageScorePercent() {
        List<Result> allResults = resultRepository.findAll();
        if (allResults.isEmpty()) return 0;

        double totalPercent = allResults.stream()
                .filter(r -> r.getTotalQuestions() > 0)
                .mapToDouble(r -> ((double) r.getScore() / r.getTotalQuestions()) * 100.0)
                .sum();
        return totalPercent / allResults.size();
    }

    public Map<String, Long> getAttemptsByTest() {
        return resultRepository.findAll().stream()
                .collect(Collectors.groupingBy(Result::getTestTitle, Collectors.counting()));
    }

    public Map<String, Long> getGlobalSuccessStats() {
        List<Result> allResults = resultRepository.findAll();
```

```
        long successful = allResults.stream()
                .filter(r -> r.getTotalQuestions() > 0 && ((double) r.getScore() / r.getTotalQuestions()) >= 0.6)
                .count();
        long failed = allResults.size() - successful;

        Map<String, Long> stats = new HashMap<>();
        stats.put("Успішно", successful);
        stats.put("Незадано", failed);
        return stats;
    }
    public Map<String, Double> getDifficultyRating() {
        List<Result> allResults = resultRepository.findAll();

        return allResults.stream()
                .filter(r -> r.getTotalQuestions() > 0)
                .collect(Collectors.groupingBy(
                        Result::getTestTitle,
                        Collectors.averagingDouble(r -> ((double) r.getScore() / r.getTotalQuestions()) * 100.0)
                ));
    }
}
class TestService
package com.example.oopp.service;

import com.example.oopp.model.Test;
import com.example.oopp.repository.TestRepository;
import org.springframework.stereotype.Service;
import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;

@Service
public class TestService {
    private final TestRepository testRepository;

    public TestService(TestRepository testRepository) {
        this.testRepository = testRepository;
    }

    public List<Test> findAll() {
        return testRepository.findAll();
    }

    public Optional<Test> findById(Long id) {
        return testRepository.findById(id);
    }

    public void save(Test test) {
        testRepository.save(test);
    }

    public void deleteById(Long id) {
        testRepository.deleteById(id);
```

```
    }
    public List<Test> findAllActive() {
        return testRepository.findAll().stream()
                .filter(t -> !t.isArchived())
                .collect(Collectors.toList());
    }
}
class UserStatusInterceptor
package com.example.oopp.service;

import com.example.oopp.model.SystemSettings;
import com.example.oopp.model.User;
import com.example.oopp.repository.UserRepository;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import org.springframework.web.servlet.HandlerInterceptor;

@Component
public class UserStatusInterceptor implements HandlerInterceptor {
    @Autowired
    private UserRepository userRepository;

    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws
Exception {
        HttpSession session = request.getSession();
        if (SystemSettings.maintenanceMode && session.getAttribute("admin") == null) {
            String uri = request.getRequestURI();
            if (!uri.equals("/login") && !uri.startsWith("/css") && !uri.startsWith("/js")) {
                response.sendRedirect("/login?error=maintenance");
                return false;
            }
        }
        String username = (String) session.getAttribute("username");
        Integer sessionVersion = (Integer) session.getAttribute("sessionVersion");
        if (username != null) {
            User user = userRepository.findByUsername(username).orElse(null);
            if (user == null || user.isBlocked()) {
                session.invalidate();
                response.sendRedirect("/login?error=blocked");
                return false;
            }
            if (sessionVersion != null && sessionVersion < user.getSessionVersion()) {
                session.invalidate();
                response.sendRedirect("/login?error=session_expired");
                return false;
            }
        }
        return true;
    }
```

```
}
class OoppApplication
package com.example.oopp;

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

@SpringBootApplication
public class OoppApplication {

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

}
package/resources/static/style.css
body {
    font-family: Arial, sans-serif;
    background: linear-gradient(135deg, #eef2ff, #f8fafc);
    margin: 0;
    padding: 0;
    color: #1f2937;
}

.container {
    width: 90%;
    max-width: 1100px;
    margin: 40px auto;
}

.welcome {
    text-align: center;
    margin-top: 60px;
}

h1, h2, h3 {
    color: #111827;
}

.header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 20px;
}

.role-selection {
    display: flex;
    justify-content: center;
    gap: 20px;
    margin-top: 30px;
    flex-wrap: wrap;
}
```

```
.card-list {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));
    gap: 20px;
}

.card {
    background: white;
    padding: 24px;
    border-radius: 18px;
    box-shadow: 0 8px 24px rgba(0, 0, 0, 0.08);
    transition: transform 0.2s ease, box-shadow 0.2s ease;

}

.card:hover {
    transform: translateY(-4px);
    box-shadow: 0 12px 28px rgba(0, 0, 0, 0.12);
}

.button, .button-secondary, button {
    display: inline-block;
    text-decoration: none;
    border: none;
    padding: 12px 18px;
    border-radius: 12px;
    cursor: pointer;
    font-size: 15px;
    transition: 0.2s ease;
}

.button {
    background: #4f46e5;
    color: white;
}

.button:hover {
    background: #4338ca;
}

.button-secondary {
    background: #e5e7eb;
    color: #111827;
}

.button-secondary:hover {
    background: #d1d5db;
}

input[type="text"],
input[type="password"] {
    width: 100%;
```

```
    padding: 12px;
    border: 1px solid #d1d5db;
    border-radius: 12px;
    font-size: 15px;
    box-sizing: border-box;
}

.result-score {
    font-size: 22px;
    margin: 20px 0;
    color: #4f46e5;
}

.progress-container {
    width: 100%;
    height: 20px;
    background: #e5e7eb;
    border-radius: 999px;
    overflow: hidden;
    margin-top: 10px;
}

.progress-bar {
    height: 100%;
    width: 0%;
    background: linear-gradient(90deg, #4f46e5, #22c55e);
    transition: width 0.3s ease-in-out;
    border-radius: 999px;
}

table {
    width: 100%;
    border-collapse: collapse;
}

table th, table td {
    border-bottom: 1px solid #ddd;
    padding: 12px;
}

.question-card {
    animation: fadeIn 0.3s ease-in-out;
}

.option {
    margin: 10px 0;
    padding: 12px;
    border-radius: 12px;
    background: #f8f9ff;
    transition: 0.2s;
}

.option:hover {
```

```
    background: #eaf0ff;
}

input[type="radio"] {
    margin-right: 8px;
}

#timer {
    color: crimson;
    font-size: 24px;
    font-weight: bold;
}

ul {
    padding-left: 20px;
}

li {
    margin: 8px 0;
}

hr {
    border: none;
    height: 1px;
    background: #ddd;
}

@keyframes fadeIn {
    from {
        opacity: 0;
        transform: translateY(12px);
    }
    to {
        opacity: 1;
        transform: translateY(0);

        .chat-bubble {
            position: fixed;
            bottom: 20px;
            left: 20px;
            width: 60px;
            height: 60px;
            background: #4f46e5;
            border-radius: 50%;
            display: flex;
            align-items: center;
            justify-content: center;
            font-size: 30px;
            color: white;
            cursor: pointer;
            box-shadow: 0 4px 12px rgba(0, 0, 0, 0.2);
            z-index: 1000;
            transition: transform 0.2s;
```

```
}

.chat-bubble:hover {
    transform: scale(1.1);
}
.chat-window {
    position: fixed;
    bottom: 90px;
    left: 20px;
    width: 350px;
    height: 500px;
    background: white;
    border-radius: 15px;
    box-shadow: 0 8px 24px rgba(0, 0, 0, 0.15);
    display: none;
    flex-direction: column;
    z-index: 1000;
    overflow: hidden;
    border: 1px solid #e5e7eb;
}

.chat-header {
    background: #4f46e5;
    color: white;
    padding: 15px;
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.chat-messages {
    flex: 1;
    padding: 15px;
    overflow-y: auto;
    background: #f9fafb;
}

.chat-footer {
    padding: 10px;
    border-top: 1px solid #eee;
}

.message-item {
    background: white;
    padding: 8px 12px;
    border-radius: 10px;
    margin-bottom: 10px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.05);
    position: relative;
}

.admin-badge {
    background: #fee2e2;
```

```
            color: #dc2626;
            font-size: 0.7em;
            padding: 2px 5px;
            border-radius: 4px;
        }

        .teacher-badge {
            background: #e0e7ff;
            color: #4338ca;
            font-size: 0.7em;
            padding: 2px 5px;
            border-radius: 4px;
        }

        #chatInput {
            width: 100%;
            margin-bottom: 5px;
        }
        .admin-btn {
            background-color: #f3f4f6;
            color: #374151;
            border: 1px solid #d1d5db;
            font-weight: 600;
            display: flex;
            align-items: center;
            gap: 8px;
            padding: 8px 16px;
        }

        .admin-btn:hover {
            background-color: #e5e7eb;
            border-color: #9ca3af;
            transform: translateY(-2px);
            box-shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.1);
        }
        .logout-btn {
            border: 1px solid #fee2e2;
            color: #dc2626 !important;
        }

        .logout-btn:hover {
            background-color: #fef2f2;
            border-color: #ef4444;
        }
        .nav-link.active, .button-secondary.active {
            background-color: #4f46e5 !important;
            color: white !important;
            border-color: #4338ca !important;
        }
    }
}
```

package/resources/templates
admin-content.html

```
<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
   <meta charset="UTF-8">
   <title>Керування контентом</title>
   <link rel="stylesheet" th:href="@{/style.css}">
</head>
<body>
<div class="container">
   <div class="header">
      <h1>⚙ Керування контентом</h1>
      <div class="nav-buttons">
         <a href="/admin" class="button-secondary" style="background-color: #e5e7eb; color: black;">🏠
Головна панель </a>
         <a href="/logout" class="button-secondary" style="background-color: #e74c3c; color:
white;">Вихід</a>
      </div>
   </div>
   <div class="card">
      <div style="display: flex; justify-content: space-between; align-items: center; margin-bottom: 20px;">
         <h2 style="margin: 0;">📄 Список усіх тестів</h2>
         <div style="display: flex; gap: 10px;">
            <a th:href="@{/admin/content(showArchived=${!showArchived})}" class="button-secondary">
               <span th:text="${showArchived} ? '📂 Сховати архів' : '📂 Показати архів'"></span>
            </a>
            <a href="/teacher/test/new" class="button-secondary">➕ Створити тест</a>
         </div>
      </div>
      <table>
         <thead>
         <tr>
            <th>Назва тесту</th>
            <th>Категорія</th>
            <th>Статус</th>
            <th style="text-align: center;">Дії</th>
         </tr>
         </thead>
         <tbody style="text-align: center;" >
         <tr th:each="t : ${tests}" th:style="${t.archived} ? 'opacity: 0.6; background: #f9f9f9;' : ''">
            <td>
               <strong th:text="${t.title}"></strong>
            </td>
            <td th:text="${t.category}"></td>
            <td>
               <span th:if="${t.archived}" style="color: #e74c3c; font-weight: bold;">Архів</span>
               <span th:if="${!t.archived && t.hidden}" style="color: #7f8c8d;">Прихований</span>
               <span th:if="${!t.archived && !t.hidden}" style="color: #27ae60;">Активний</span>
            </td>
            <td>
               <div style="display: flex; justify-content: center; gap: 8px;">
                  <form th:if="${!t.archived}" th:action="@{/teacher/test/toggle-visibility/{id}(id=${t.id})}"
method="post">
```

```
                    <button type="submit" class="button-secondary" th:text="${t.hidden ? '👁' : '🚫'}"
title="Видимість"></button>
                </form>
                <form th:if="${!t.archived}" th:action="@{/teacher/test/archive/{id}(id=${t.id})}"
method="post">
                    <button type="submit" class="button-secondary" title="В архів">📁</button>
                </form>
                <form th:if="${t.archived}" th:action="@{/teacher/test/unarchive/{id}(id=${t.id})}"
method="post">
                    <button type="submit" class="button-secondary" title="Відновити">🔄</button>
                </form>
                <a th:if="${!t.archived}" th:href="@{/teacher/test/edit/{id}(id=${t.id})}" class="button-
secondary">📝</a>
                <a th:href="@{/teacher/test/delete/{id}(id=${t.id})}" class="button-secondary"
                   style="color: #e74c3c; border-color: #e74c3c;" onclick="return confirm('Видалити
тест?')">🗑</a>
            </div>
        </td>
    </tr>
    <tr th:if="${#lists.isEmpty(tests)}">
        <td colspan="4" style="text-align: center; padding: 20px; color: #7f8c8d;">Тестів не знайдено</td>
    </tr>
    </tbody>
</table>
</div>
</div>
<div th:replace="~{fragments :: chat}"></div>
</body>
</html>
```

Admin-hub.html

```
<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Панель адміністратора</title>
    <link rel="stylesheet" th:href="@{/style.css}">
</head>
<body>
<div class="container">
    <div class="header">
        <h1>🛡 Панель адміністратора</h1>
        <a href="/logout" class="button-secondary" style="background-color: #e74c3c; color: white;">Вихід</a>
    </div>
    <div class="card-list" style="margin-top: 40px;">
        <a href="/admin/users" class="card" style="text-decoration: none; text-align: center;">
            <span style="font-size: 3em;">👥</span>
            <h2>Користувачі</h2>
            <p>Керування акаунтами та паролями</p>
        </a>
        <a href="/admin/content" class="card" style="text-decoration: none; text-align: center;">
```

```
        <span style="font-size: 3em;">📄</span>
        <h2>Контент</h2>
        <p>Керування тестами та категоріями</p>
      </a>
      <a href="/admin/stats" class="card" style="text-decoration: none; text-align: center;">
        <span style="font-size: 3em;">📊</span>
        <h2>Статистика</h2>
        <p>Аналітика успішності системи</p>
      </a>
      <a href="/admin/logs" class="card" style="text-decoration: none; text-align: center;">
        <span style="font-size: 3em;">📜</span>
        <h2>Журнал</h2>
        <p>Перегляд системних логів</p>
      </a>
      <a href="/teacher" class="card" style="text-decoration: none; text-align: center;">
        <span style="font-size: 3em;">👨🏫</span>
        <h2>Панель викладача</h2>
        <p>Керування тестами та результатами</p>
      </a>
        <a href="/student" class="card" style="text-decoration: none; text-align: center;">
          <span style="font-size: 3em;">🎓</span>
          <h2>Кабінет студента</h2>
          <p>Перегляд тестів очима учня</p>
        </a>
      </div>
    </div>
</div>
</body>
</html>
```

Admin-logs.html

```
<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Журнал подій</title>
    <link rel="stylesheet" th:href="@{/style.css}">
</head>
<body>
<div class="container">
    <div class="header">
        <h1>📜 Журнал системних подій</h1> <div style="display:flex; gap:10px;">
        <a href="/admin" class="button-secondary" style="display: flex; align-items: center; gap: 8px;">
            🏠 Головна панель
        </a>
        <a href="/logout" class="button-secondary" style="background-color: #e74c3c; color: white;">Вихід</a>
    </div>
    </div>
    <div class="card">
        <div th:each="entry : ${logs}" style="padding: 10px; border-bottom: 1px solid #eee; font-family:
monospace;">
            <span th:text="${entry}"></span>
        </div>
```

```
        <p th:if="${logs.isEmpty()}" style="text-align:center; color:#666;">Подій ще не зафіксовано.</p>
    </div>
</div>
<div th:replace="~{fragments :: chat}"></div>
</body>
</html>
```

Admin-stats.html

```
<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org" xmlns="http://www.w3.org/1999/html">
<head>
    <meta charset="UTF-8">
    <title>Панель адміністратора | Статистика</title>
    <link rel="stylesheet" th:href="@{/style.css}">
    <style>
        .stats-grid {
            display: grid;
            grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
            gap: 25px;
            margin-bottom: 30px;
        }
        .mini-card {
            text-align: center;
            padding: 20px;
            background: white;
            border-radius: 15px;
            box-shadow: 0 4px 6px rgba(0,0,0,0.05);
        }
        .mini-card h3 { font-size: 0.9em; color: #6b7280; margin-bottom: 10px; }
        .mini-card .value { font-size: 2em; font-weight: bold; color: #4f46e5; }
        .action-bar {
            display: flex;
            gap: 15px;
            flex-wrap: wrap;
            margin-bottom: 30px;
        }
    </style>
</head>
<body>
<div class="container">
    <div class="header">
        <h1>📊Статистика та інше</h1> <div style="display:flex; gap:10px;">
        <a href="/admin" class="button-secondary" style="display: flex; align-items: center; gap: 8px;">
            🏠 Головна панель
        </a>
        <a href="/logout" class="button-secondary" style="background-color: #e74c3c; color: white;">Вихід</a>

    </div>
</div>
    <div class="stats-grid">
        <div class="mini-card">
            <h3>Всього спроб</h3>
            <div class="value" th:text="${totalAttempts}">0</div>
        </div>
```

```
      <div class="mini-card">
        <h3>Середній бал системи</h3>
        <div class="value"><span th:text="${averageScore}">0.0</span>%</div>
      </div>
      <div class="mini-card">
        <h3>Активних тестів</h3>
        <div class="value" th:text="${testsCount}">0</div>
      </div>
    </div>
    <div style="display: grid; grid-template-columns: 1fr 1fr; gap: 25px; margin-bottom: 30px;">
      <div class="card" style="border-top: 4px solid #D3D3D3;">
        <h3>📈 Середня успішність за тестами</h3>
        <p style="font-size: 0.85em; color: #6b7280; margin-bottom: 15px;">
          Показує середній відсоток правильних відповідей. Чим нижча смужка (червона), тим складніше
студентам дається цей матеріал.
        </p>

        <div th:if="${difficultyRating != null && !difficultyRating.isEmpty()}">
          <div th:each="entry : ${difficultyRating}" style="margin-bottom: 15px;">
            <div style="display: flex; justify-content: space-between; font-size: 0.9em; margin-bottom: 5px;">
              <span style="font-weight: 500;" th:text="${entry.key}">Назва тесту</span>
              <span th:text="${#numbers.formatDecimal(entry.value, 1, 1)} + '% правильних'">0%</span>
            </div>

            <div class="progress-container" style="height: 12px; background-color: #f3f4f6; border-radius:
6px;">
              <div class="progress-bar"
                th:style="'width:' + ${entry.value} + '%; height: 100%; border-radius: 6px; background: ' +
                  (${entry.value} < 50.0 ? '#ef4444' : (${entry.value} < 75.0 ? '#f59e0b' : '#22c55e')) + ';'">
              </div>
            </div>
          </div>
        </div>

        <div th:if="${difficultyRating == null || difficultyRating.isEmpty()}">
          <p style="color: #6b7280; font-size: 0.9em; text-align: center; padding: 10px;">
            Статистика складності поки що відсутня.
          </p>
        </div>
      </div>
      <div class="card" style="border-top: 4px solid #008000;">
        <h3>✅ Успішність спроб</h3>
        <div style="margin-top: 20px;">
          <p>Успішно (≥60%): <strong style="color: #22c55e;"
th:text="${successStats['Успішно']}">0</strong></p>
          <p>Неуспішно: <strong style="color: #ef4444;"
th:text="${successStats['Нездано']}">0</strong></p>
          <div class="progress-container" style="height: 25px; display: flex; margin-top: 15px;">
            <div th:if="${totalAttempts > 0}"
              th:style="'width:' + (${successStats['Успішно']} * 100 / ${totalAttempts}) + '%; background:
#22c55e;'"></div>
            <div th:if="${totalAttempts > 0}"
              th:style="'width:' + (${successStats['Нездано']} * 100 / ${totalAttempts}) + '%; background:
```

```
#ef4444;'"></div>
            </div>
          </div>
        </div>
    </div>
    <div style="display: grid; grid-template-columns: 1fr 1fr; gap: 25px;">
        <div class="card" style="border-top: 4px solid #4f46e5;">
          <h3>⚙️ Системні налаштування</h3>
          <div style="display: flex; flex-direction: column; gap: 15px; margin-top: 15px;">
            <div style="display: flex; justify-content: space-between; align-items: center;">
              <span>Режим технічних робіт</span>
              <a th:href="@{/admin/settings/maintenance}" class="button"
                th:style="${T(com.example.oopp.model.SystemSettings).maintenanceMode} ? 'background:
#ef4444' : 'background: #22c55e'">
                <span th:text="${T(com.example.oopp.model.SystemSettings).maintenanceMode} ? 'Вимкнути'
: 'Активувати'"></span>
              </a>
            </div>
            <div style="display: flex; justify-content: space-between; align-items: center;">
              <span>Доступ до чату</span>
              <a th:href="@{/admin/settings/toggle-chat}" class="button"
                th:style="${T(com.example.oopp.model.SystemSettings).chatEnabled} ? 'background: #ef4444' :
'background: #22c55e'">
                <span th:text="${T(com.example.oopp.model.SystemSettings).chatEnabled} ? 'Вимкнути' :
'Активувати'"></span>
              </a>
            </div>
          </div>
        </div>
        <div class="card" style="border-top: 4px solid #ef4444;">
          <h3>🛡️ Керування безпекою</h3>
          <div style="display: flex; flex-direction: column; gap: 10px;align-items: center;">
            <a th:href="@{/admin/chat/clear}" class="button"  style="width:120px; background: #ef4444; text-
align: center; "
               onclick="return confirm('Видалити історію чату?')">🗑️ Очистити чат</a>
            <a href="/admin/sessions/terminate-all" class="button" style="width:200px; background: #ef4444;
text-align: center;"
               onclick="return confirm('Вийти з усіх акаунтів?')">🛑 Завершити всі сесії</a>

          </div>
        </div>
    </div>
</div>
<div th:replace="~{fragments :: chat}"></div>
</body>
</html>
```

Admin-users.html

```
<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Керування користувачами</title>
    <link rel="stylesheet" th:href="@{/style.css}">
```

```
    <style>
        .password-cell {
            display: flex;
            align-items: center;
            gap: 10px;
            font-family: monospace;
            background: #f9fafb;
            padding: 5px 10px;
            border-radius: 8px;
            border: 1px solid #e5e7eb;
            min-width: 120px;
            width: 150px;

            width: 150px;
            justify-content: space-between;
        }
        .hidden-pass {
            -webkit-text-security: disc;
            color: #6b7280;
        }
        .visible-pass {
            -webkit-text-security: none;
            color: #111827;
            font-weight: bold;
        }
        .eye-btn {
            cursor: pointer;
            user-select: none;
            font-size: 1.1em;
            transition: transform 0.1s;
        }
        .eye-btn:active {
            transform: scale(0.9);
        }
    </style>
</head>
<body>
<div class="container">
    <div class="header">
        <h1>Керування користувачами</h1> <div style="display:flex; gap:10px;">
        <a href="/admin" class="button-secondary" style="display: flex; align-items: center; gap: 8px;">
            Головна панель
        </a>
        <a href="/logout" class="button-secondary" style="background-color: #e74c3c; color: white;">Вихід</a>

    </div>
    </div>
    <div class="card" style="margin-bottom: 20px; border-left: 5px solid #4f46e5;">
        <div style="display: flex; align-items: center; gap: 15px;">
            <span style="font-size: 1.2em;">🔍</span>
            <input type="text" id="userSearch"
                   placeholder="Почніть вводити ім'я користувача для фільтрації..."
                   onkeyup="filterUsers()"
```

```
            style="width: 100%; padding: 12px; border-radius: 12px; border: 1px solid #d1d5db; font-size:
15px;">
        </div>
    </div>
    <script>
        function filterUsers() {
            let input = document.getElementById('userSearch').value.toLowerCase();
            let rows = document.querySelectorAll('table tbody tr');
            rows.forEach(row => {
                let username = row.cells[1].innerText.toLowerCase();
                if (username.includes(input)) {
                    row.style.display = "";
                } else {
                    row.style.display = "none";
                }
            });
        }
    </script>
    <div class="card">
        <h2>Створити нового користувача</h2>
        <form th:action="@{/admin/create-user}" method="post" style="display:flex; gap:10px; flex-wrap:wrap;
align-items:center;">
            <input type="text" name="username" placeholder="Логін" required style="flex:1; min-width:150px;">
            <div style="position: relative; flex:1; min-width:150px;">
                <input type="password" id="adminPassword" name="password" placeholder="Пароль" required
style="width:100%; padding:10px; padding-right: 35px; box-sizing: border-box; border-radius: 4px; border: 1px
solid #ccc;">
                <span onclick="togglePasswordInput('adminPassword', this)" style="position: absolute; right: 10px;
top: 50%; transform: translateY(-50%); cursor: pointer; user-select: none; font-size: 16px;">👁 </span>
            </div>
            <select name="role" required style="padding:12px; border-radius:12px; border:1px solid #d1d5db;">
                <option value="STUDENT">Студент</option>
                <option value="TEACHER">Викладач</option>
                <option value="ADMIN">Адміністратор</option>
            </select>
            <button type="submit" class="button">Створити акаунт</button>
        </form>
    </div>
    <div th:if="${param.error != null and param.error[0] == 'wrong_password'}"
         style="background-color: #fee2e2; border-left: 5px solid #ef4444; color: #991b1b; padding: 15px; margin-
bottom: 20px; border-radius: 4px;">
        ⚠ <strong>Помилка:</strong> Введено неправильний пароль! Видалення адміністратора
скасовано.
    </div>
    <div class="card" style="margin-top: 20px;">
        <h2>Список користувачів</h2>
        <table>
            <thead>
            <tr>
                <th style="text-align:left;">ID</th>
                <th style="text-align:left;">Логін</th>
                <th style="text-align:left;">Пароль</th>
                <th style="text-align:left;">Роль</th>
```

```
    <th style="text-align:left;">Статус</th>
    <th style="text-align:left;">Дія</th>
</tr>
</thead>
<tbody>
<tr th:each="user : ${users}" th:if="${user != null}">
    <td th:text="${user?.id}">1</td>
    <td th:text="${user?.username}">ivan</td>
    <td>
        <div class="password-cell">
            <span class="hidden-pass">********</span>
            <span style="font-size: 0.8em; color: #6b7280;">(зашифровано)</span>
        </div>
    </td>
    <td>
        <span th:if="${user?.role == 'STUDENT'}">🎓 Студент</span>
        <span th:if="${user?.role == 'TEACHER'}">👨‍🏫 Вчитель</span>
        <span th:if="${user?.role == 'ADMIN'}">🛡️ Адмін</span>
    </td>
    <td>
        <span th:if="${user?.blocked}" style="color: crimson; font-weight: bold;">Заблокований</span>
        <span th:if="${!user?.blocked}" style="color: green;">Активний</span>
    </td>
    <td>
        <div style="display:flex; gap:5px;">
            <a th:href="@{/admin/users/{username}(username=${user.username})}"
               class="button-secondary"
               style="padding: 6px 10px;"
               title="Переглянути профіль">
                👤 Профіль
            </a>
            <a th:if="${user?.role != 'ADMIN'}"
               th:href="@{'/admin/block-user/' + ${user?.id}}"
               class="button-secondary"
               th:text="${user?.blocked ? 'Розблокувати' : 'Блокувати'}"
               style="padding: 6px 10px;"></a>
            <a th:href="@{'/admin/reset-password/' + ${user?.id}}"
               class="button-secondary"
               title="Скинути пароль до 1234"
               onclick="return confirm('Скинути пароль цього користувача до 1234?')"
               style="padding: 6px 10px; background: #fef3c7; color: #92400e;">🔑</a>
            <a th:if="${user.role != 'ADMIN'}"
               th:href="@{'/admin/delete-user/' + ${user.id}}"
               class="button-secondary"
               onclick="return confirm('Ви впевнені, що хочете видалити цього користувача?');"
               style="color:crimson; padding: 6px 10px;">🗑️ </a>
            <form th:if="${user.role == 'ADMIN'}"
                  th:action="@{'/admin/delete-admin/' + ${user.id}}"
                  method="post"
                  style="margin: 0; display: inline-flex;"
                  onsubmit="return confirmAdminDelete(this);">
                <input type="hidden" name="adminPassword" class="admin-password-input">
```

```
                    <button type="submit" class="button-secondary"
                            style="color:crimson; padding: 5px 10px; background: none; border: 1px solid #e5e7eb;
cursor: pointer; border-radius: 4px;"
                            title="Видалити адміністратора">
                        🗑
                    </button>
                </form>
            </div>
            <span th:if="${user?.role == 'ADMIN'}" style="color: #6b7280; font-style: italic;">Системний
аккаунт</span>
        </td>
    </tr>
    </tbody>
</table>
</div>
</div>
<script>
// Функція для перемикання видимості пароля в таблиці
function toggleCellPassword(btn) {
    const passText = btn.previousElementSibling;
    if (passText.classList.contains('hidden-pass')) {
        passText.classList.remove('hidden-pass');
        passText.classList.add('visible-pass');
        btn.textContent = '🙈';
    } else {
        passText.classList.remove('visible-pass');
        passText.classList.add('hidden-pass');
        btn.textContent = '👁 ';
    }
}
// Функція для поля вводу (форма зверху)
function togglePasswordInput(inputId, icon) {
    const input = document.getElementById(inputId);
    if (input.type === "password") {
        input.type = "text";
        icon.textContent = "🙈";
    } else {
        input.type = "password";
        icon.textContent = "👁 ";
    }
}// Функція для підтвердження видалення адміністратора
function confirmAdminDelete(form) {
    let pwd = prompt("⚠ УВАГА! Видалення адміністратора.\nДля підтвердження дії введіть ВАШ
поточний пароль:");
    if (pwd === null || pwd.trim() === "") {
        return false;
    }
    form.querySelector('.admin-password-input').value = pwd;
    return true;
}
</script>
</body>
```

```
<div th:replace="~{fragments :: chat}"></div>
</html>
```

Chat.html

```
<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Чат</title>
    <link rel="stylesheet" th:href="@{/style.css}">
    <style>
        /* Кнопка-хмаринка в куті */
        .chat-bubble {
            position: fixed;
            bottom: 20px;
            left: 20px;
            width: 60px;
            height: 60px;
            background: #4f46e5;
            border-radius: 50%;
            display: flex;
            align-items: center;
            justify-content: center;
            font-size: 30px;
            color: white;
            cursor: pointer;
            box-shadow: 0 4px 12px rgba(0,0,0,0.2);
            z-index: 1000;
            transition: transform 0.2s;
        }
        .chat-bubble:hover { transform: scale(1.1); }
        .chat-window {
            position: fixed;
            bottom: 90px;
            left: 20px;
            width: 350px;
            height: 500px;
            background: white;
            border-radius: 15px;
            box-shadow: 0 8px 24px rgba(0,0,0,0.15);
            display: none;
            flex-direction: column;
            z-index: 1000;
            overflow: hidden;
            border: 1px solid #e5e7eb;
        }

        .chat-header {
            background: #4f46e5;
            color: white;
            padding: 15px;
            display: flex;
            justify-content: space-between;
            align-items: center;
```

```
        }

        .chat-messages {
            flex: 1;
            padding: 15px;
            overflow-y: auto;
            background: #f9fafb;
        }

        .chat-footer {
            padding: 10px;
            border-top: 1px solid #eee;
        }

        .message-item {
            background: white;
            padding: 8px 12px;
            border-radius: 10px;
            margin-bottom: 10px;
            box-shadow: 0 2px 4px rgba(0,0,0,0.05);
            position: relative;
        }

        .admin-badge { background: #fee2e2; color: #dc2626; font-size: 0.7em; padding: 2px 5px; border-radius:
4px; }
        .teacher-badge { background: #e0e7ff; color: #4338ca; font-size: 0.7em; padding: 2px 5px; border-radius:
4px; }

        #chatInput { width: 100%; margin-bottom: 5px; }
    </style>
</head>
<body>
<div class="chat-bubble" onclick="toggleChat()">💬</div>
<div class="chat-window" id="chatWindow">
    <div class="chat-header">
        <span>Чат підтримки</span>
        <button onclick="toggleChat()" style="background:none; border:none; color:white; cursor:pointer; font-
size:20px;">×</button>
    </div>
    <div class="chat-messages" id="messageArea">
        <div th:each="msg : ${messages}" class="message-item">
            <strong th:text="${msg.sender}"></strong>
            <span th:if="${msg.role == 'ADMIN'}" class="admin-badge">🛡️ АДМІН</span>
            <span th:if="${msg.role == 'TEACHER'}" class="teacher-badge">👨‍🏫 ВЧИТЕЛЬ</span>
            <small th:text="${msg.timestamp}" style="color:#999; font-size: 0.7em;"></small>
            <a th:if="${session.admin != null || session.teacher != null}"
               th:href="@{/chat/delete/{id}(id=${msg.id})}"
               style="color:red; text-decoration:none; position:absolute; right:10px; top:10px;">×</a>
            <p th:text="${msg.content}" style="margin: 5px 0 0 0; font-size: 0.9em;"></p>
        </div>
    </div>
    <div class="chat-footer">
        <form th:action="@{/chat/send}" method="post">
```

```
        <input type="text" name="content" id="chatInput" placeholder="Ваше повідомлення..." required
autocomplete="off">
        <div style="margin-bottom: 5px;">
          <span onclick="addEmoji('😊')" style="cursor:pointer">😊</span>
          <span onclick="addEmoji('👍')" style="cursor:pointer">👍</span>
          <span onclick="addEmoji('❓')" style="cursor:pointer">❓</span>
          <span onclick="addEmoji('🎓')" style="cursor:pointer">🎓</span>
        </div>
        <button type="submit" class="button" style="width:100%; padding: 8px;">Надіслати</button>
      </form>
    </div>
</div>
<script>
    function toggleChat() {
      const chat = document.getElementById('chatWindow');
      if (chat.style.display === 'none' || chat.style.display === '') {
        chat.style.display = 'flex';
        // Прокрутка вниз при відкритті
        const area = document.getElementById('messageArea');
        area.scrollTop = area.scrollHeight;
      } else {
        chat.style.display = 'none';
      }
    }
    function addEmoji(emoji) {
      document.getElementById('chatInput').value += emoji;
    }
</script>
<div th:replace="~{fragments :: chat}"></div>
</body>
</html>
Create-test.html
<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Налаштування тесту</title>
    <link rel="stylesheet" th:href="@{/style.css}">
</head>
<body>
<div class="container">
    <div class="header">
      <h1 th:text="${test.id == null ? 'Новий тест' : 'Редагування тесту'}"></h1>
      <a href="/teacher" class="button-secondary">Скасувати</a>
    </div>
    <form th:action="@{/teacher/test/save}" th:object="${test}" method="post">
      <input type="hidden" th:field="*{id}" />
      <div class="card">
        <label><strong>Назва тесту (обов'язково):</strong></label>
        <input type="text" th:field="*{title}" placeholder="Наприклад: Основи Java" required style="margin-
top:10px; font-size: 16px;" />
      </div>
```

```
    <div id="questions-container">
      <div th:each="question, stat : *{questions}" class="card question-block" style="margin-top:20px;">
        <h3 style="color: #4f46e5;">Питання №<span class="question-number" th:text="${stat.index + 1}"></span></h3>
        <label>Текст питання:</label>
        <input type="text" th:field="*{questions[__${stat.index}__].text}" placeholder="Введіть запитання..." style="margin:10px 0;" required/>
        <h4>Варіанти (оберіть правильну відповідь кружечком):</h4>
        <div th:each="option, optStat : *{questions[__${stat.index}__].options}">
          <div style="display:flex; align-items:center; gap:10px; margin-bottom:8px;">
            <input type="radio"
                th:field="*{questions[__${stat.index}__].correctAnswerIndex}"
                th:value="${optStat.index}"
                required />
            <input type="text"
                th:field="*{questions[__${stat.index}__].options[__${optStat.index}__]}"
                placeholder="Варіант відповіді"
                style="flex:1;" required/>
          </div>
        </div>
      </div>
    </div>
    <div style="margin-top:20px; text-align: center;">
      <button type="button" class="button-secondary" onclick="addQuestion()" style="font-size: 1.1em; padding: 12px 24px;">➕ Додати питання</button>
    </div>
    <div style="margin-top:30px;">
      <button type="submit" class="button" style="width:100%; font-size: 1.2em; padding: 15px;">💾 Зберегти тест</button>
    </div>
  </form>
</div>
<script>
  function addQuestion() {
    const container = document.getElementById('questions-container');
    const questionBlocks = container.getElementsByClassName('question-block');
    const index = questionBlocks.length;
    const newBlock = document.createElement('div');
    newBlock.className = 'card question-block';
    newBlock.style.marginTop = '20px';

    newBlock.innerHTML = `
      <h3 style="color: #4f46e5;">Питання №<span class="question-number">${index + 1}</span></h3>

      <label>Текст питання:</label>
      <input type="text" name="questions[${index}].text" placeholder="Введіть запитання..." style="margin:10px 0; width: 100%; padding: 12px; border: 1px solid #d1d5db; border-radius: 12px; font-size: 15px; box-sizing: border-box;" required />

      <h4>Варіанти (оберіть правильну відповідь кружечком):</h4>

      <div style="display:flex; align-items:center; gap:10px; margin-bottom:8px;">
        <input type="radio" name="questions[${index}].correctAnswerIndex" value="0" required />
```

```
            <input type="text" name="questions[${index}].options[0]" placeholder="Варіант відповіді"
style="flex:1; width: 100%; padding: 12px; border: 1px solid #d1d5db; border-radius: 12px; font-size: 15px;
box-sizing: border-box;" required />
        </div>
        <div style="display:flex; align-items:center; gap:10px; margin-bottom:8px;">
            <input type="radio" name="questions[${index}].correctAnswerIndex" value="1" required />
            <input type="text" name="questions[${index}].options[1]" placeholder="Варіант відповіді"
style="flex:1; width: 100%; padding: 12px; border: 1px solid #d1d5db; border-radius: 12px; font-size: 15px;
box-sizing: border-box;" required />
        </div>
        <div style="display:flex; align-items:center; gap:10px; margin-bottom:8px;">
            <input type="radio" name="questions[${index}].correctAnswerIndex" value="2" required />
            <input type="text" name="questions[${index}].options[2]" placeholder="Варіант відповіді"
style="flex:1; width: 100%; padding: 12px; border: 1px solid #d1d5db; border-radius: 12px; font-size: 15px;
box-sizing: border-box;" required />
        </div>
        <div style="display:flex; align-items:center; gap:10px; margin-bottom:8px;">
            <input type="radio" name="questions[${index}].correctAnswerIndex" value="3" required />
            <input type="text" name="questions[${index}].options[3]" placeholder="Варіант відповіді"
style="flex:1; width: 100%; padding: 12px; border: 1px solid #d1d5db; border-radius: 12px; font-size: 15px;
box-sizing: border-box;" required />
        </div>
    `;
    container.appendChild(newBlock);
  }
</script>
<div th:replace="~{fragments :: chat}"></div>
</body>
</html>
Fragments.html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<body>
<div th:fragment="chat" th:if="${T(com.example.oopp.model.SystemSettings).chatEnabled || session.admin !=
null}">
  <link rel="stylesheet" th:href="@{/style.css}">
  <div th:fragment="chat">
    <style>
      /* Хмаринка-кнопка */
      .chat-bubble {
        position: fixed;
        bottom: 25px;
        right: 25px;
        width: 60px;
        height: 60px;
        background: linear-gradient(135deg, #4f46e5, #6366f1);
        border-radius: 50%;
        display: flex;
        align-items: center;
        justify-content: center;
        font-size: 28px;
        color: white;
        cursor: pointer;
```

```
    z-index: 9999;
    box-shadow: 0 4px 15px rgba(79, 70, 229, 0.4);
    transition: all 0.3s cubic-bezier(0.4, 0, 0.2, 1);
}
.chat-bubble:hover { transform: scale(1.1) rotate(5deg); }

@keyframes pulse-red {
    0% { box-shadow: 0 0 0 0 rgba(239, 68, 68, 0.7); transform: scale(1); }
    70% { box-shadow: 0 0 0 15px rgba(239, 68, 68, 0); transform: scale(1.05); }
    100% { box-shadow: 0 0 0 0 rgba(239, 68, 68, 0); transform: scale(1); }
}

.new-mention {
    animation: pulse-red 2s infinite !important;
    background: #ef4444 !important;
}

.chat-window {
    position: fixed;
    bottom: 100px;
    right: 25px;
    width: 350px;
    height: 500px;
    background: white;
    border-radius: 20px;
    box-shadow: 0 10px 25px rgba(0,0,0,0.1);
    display: none;
    flex-direction: column;
    z-index: 9999;
    border: 1px solid #f1f5f9;
    overflow: hidden;
    animation: slideIn 0.3s ease-out;
}

@keyframes slideIn {
    from { opacity: 0; transform: translateY(20px); }
    to { opacity: 1; transform: translateY(0); }
}

.chat-header {
    background: #4f46e5;
    color: white;
    padding: 15px 20px;
    font-weight: 600;
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.chat-messages {
    flex: 1;
    padding: 15px;
    overflow-y: auto;
```

```
            background: #f8fafc;
            display: flex;
            flex-direction: column;
            gap: 12px;
        }

        .message-item {
            max-width: 85%;
            padding: 10px 14px;
            border-radius: 15px;
            font-size: 0.95em;
            line-height: 1.4;
            position: relative;
            box-shadow: 0 2px 4px rgba(0,0,0,0.02);
        }

        .msg-other { align-self: flex-start; background: white; border-bottom-left-radius: 2px; border: 1px solid #e2e8f0; }
        .msg-me { align-self: flex-end; background: #4f46e5; color: white; border-bottom-right-radius: 2px; }

        .msg-info { display: flex; align-items: center; gap: 6px; margin-bottom: 4px; font-size: 0.75em; opacity: 0.8; }
        .admin-badge { background: #ef4444; color: white; padding: 1px 5px; border-radius: 4px; font-weight: bold; }

        .chat-footer { padding: 15px; background: white; border-top: 1px solid #f1f5f9; }
        .input-group { display: flex; gap: 8px; align-items: center; }
        #chatInput { flex: 1; padding: 10px 15px; border: 1px solid #e2e8f0; border-radius: 25px; outline: none; }

        .send-btn {
            background: #4f46e5; color: white; border: none; width: 38px; height: 38px;
            border-radius: 50%; cursor: pointer; display: flex; align-items: center; justify-content: center;
        }

        .mention { background: #fbbf24; color: #000; padding: 0 4px; border-radius: 4px; font-weight: bold; }
        .tag-link { cursor: pointer; color: inherit; text-decoration: none; }

        .test-mode-chat { filter: grayscale(100%); opacity: 0.4; cursor: not-allowed !important; }
    </style>

    <script th:inline="javascript">
        const notificationSound = new Audio('https://raw.githubusercontent.com/rafael-pernil-guillen/custom-sounds-with-notifications/master/notification.mp3');
        const myUsername = /*[[${session.username}]]*/ 'guest';
        let lastMessageCount = -1;

        function playNotification() {
            notificationSound.play().catch(e => console.warn("Звук чекає на клік по сторінці"));

            const bubble = document.querySelector('.chat-bubble');
            if (bubble) {
                bubble.classList.add('new-mention');
```

```
        setTimeout(() => {
            if (document.getElementById('chatWindow').style.display !== 'flex') {
                bubble.classList.remove('new-mention');
            }
        }, 15000);
    }
}

function formatMessage(text) {
    return text.replace(/@(\w+)/g, '<span class="mention">@$1</span>');
}

function refreshChat() {
    const win = document.getElementById('chatWindow');

    fetch('/chat/updates')
        .then(res => res.text())
        .then(html => {
            const area = document.getElementById('msgArea');
            const tempDiv = document.createElement('div');
            tempDiv.innerHTML = html;

            const newMessages = tempDiv.querySelectorAll('.message-item');
            const currentCount = newMessages.length;

            // Логіка перевірки нових тегів
            if (lastMessageCount !== -1 && currentCount > lastMessageCount) {
                for (let i = lastMessageCount; i < currentCount; i++) {
                    const msgText = newMessages[i].querySelector('.msg-text').innerText;
                    const senderElement = newMessages[i].querySelector('b');
                    const sender = senderElement ? senderElement.innerText : "";

                    if (msgText.includes('@' + myUsername) && sender !== myUsername) {
                        playNotification();
                        break;
                    }
                }
            }

            area.innerHTML = html;

            // Форматування тегів після оновлення HTML
            document.querySelectorAll('.msg-text').forEach(p => {
                if (!p.dataset.formatted) {
                    p.innerHTML = formatMessage(p.innerText);
                    p.dataset.formatted = "true";
                }
            });

            lastMessageCount = currentCount;
            if (win.style.display === 'flex') scrollChat();
        });
}
```

```
        function tagUser(username) {
            const input = document.getElementById('chatInput');
            input.value = '@' + username + ' ' + input.value;
            input.focus();
        }

        function toggleChat() {
            const win = document.getElementById('chatWindow');
            const bubble = document.querySelector('.chat-bubble');
            const isOpening = win.style.display === 'none' || win.style.display === '';

            win.style.display = isOpening ? 'flex' : 'none';
            if (isOpening) {
                bubble.classList.remove('new-mention');
                scrollChat();
            }
        }

        function scrollChat() {
            const area = document.getElementById('msgArea');
            area.scrollTop = area.scrollHeight;
        }

        function addEmoji(emoji) {
            document.getElementById('chatInput').value += emoji;
        }

        function sendMessage(event) {
            event.preventDefault();
            const input = document.getElementById('chatInput');
            const content = input.value;
            if (!content.trim()) return;

            const formData = new FormData();
            formData.append('content', content);

            fetch('/chat/send', { method: 'POST', body: formData })
                .then(() => {
                    input.value = '';
                    refreshChat();
                });
        }
        function deleteMsg(id) {
    if (confirm("Видалити це повідомлення?")) {
        fetch('/chat/delete/' + id)
            .then(response => {
                if (response.ok) {
                    refreshChat();
                }
            })
            .catch(error => console.error('Помилка видалення:', error));
    }
```

```
}
        setInterval(refreshChat, 3000);
    </script>
    <div class="chat-bubble"
         th:classappend="${test != null and sessionQuestions != null} ? 'test-mode-chat' : ''"
         onclick="if(!this.classList.contains('test-mode-chat')) toggleChat()">💬</div>
    <div class="chat-window" id="chatWindow">
        <div class="chat-header">
            <span>Чат підтримки</span>
            <span onclick="toggleChat()" style="cursor:pointer; font-size: 1.2em;">&times;</span>
        </div>
        <div class="chat-messages" id="msgArea">
            <div th:fragment="messageList">
                <div th:each="msg : ${@chatService.messages}"
                     class="message-item"
                     th:classappend="${msg.sender == session.username} ? 'msg-me' : 'msg-other'">
                    <div class="msg-info">
                    <span th:data-username="${msg.sender}" onclick="tagUser(this.getAttribute('data-username'))"
class="tag-link">
                        <b th:text="${msg.sender}"></b>
                    </span>
                        <span th:if="${session.admin != null || session.teacher != null}">
                        <a th:href="@{/teacher/user-stats/{username}(username=${msg.sender})}" style="font-size:
0.8em; margin-left: 5px; color: #4f46e5;">(інфо)</a>
                    </span>
                        <span th:if="${msg.role != 'STUDENT'}" class="admin-badge">Staff</span>
                        <span th:text="${msg.timestamp}"></span>
                    </div>
                    <p class="msg-text" th:text="${msg.content}" style="margin:0;"></p>
                    <span th:if="${session.admin != null || session.teacher != null}"
                          th:onclick="'deleteMsg(' + ${msg.id} + ')'"
                          style="font-size: 1.2em; color: #ef4444; cursor: pointer; position: absolute; top: 5px; right:
8px;">&times;</span>
                </div>
            </div>
        </div>

        <div class="chat-footer">
            <div th:if="${test != null and sessionQuestions != null}" style="text-align: center; color: #64748b;
font-size: 0.85em;">
                ⚠ Чат вимкнено під час тесту
            </div>
            <form th:unless="${test != null and sessionQuestions != null}" id="chatForm"
onsubmit="sendMessage(event)">
                <div class="emoji-bar" style="margin-bottom: 8px; font-size: 1.2em;">
                    <span onclick="addEmoji('😊')" style="cursor:pointer">😊</span>
                    <span onclick="addEmoji('👍')" style="cursor:pointer">👍</span>
                    <span onclick="addEmoji('🎓')" style="cursor:pointer">🎓</span>
                </div>
                <div class="input-group">
                    <input type="text" id="chatInput" placeholder="Повідомлення..." required autocomplete="off">
                    <button type="submit" class="send-btn">➤</button>
```

```
            </div>
          </form>
        </div>
      </div>
    </div>
</div>
</body>
</html>
```

Index.html

```
<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Система тестування ООП</title>
    <link rel="stylesheet" th:href="@{/style.css}">
</head>
<body>
<div class="container welcome">
    <div th:if="${maintenanceMode}"
         class="card"
         style="background: #fffbeb; border: 2px solid #fbbf24; margin-bottom: 30px; animation: slideIn 0.5s ease-out;">
        <div style="display: flex; align-items: center; justify-content: center; gap: 15px;">
            <span style="font-size: 2em;">⚠</span>
            <div style="text-align: left;">
                <h2 style="color: #92400e; margin: 0;">Технічні роботи</h2>
                <p style="color: #b45309; margin: 5px 0 0 0;">
                    Зараз проводяться планові оновлення системи. Доступ до тестів для студентів тимчасово обмежений.
                </p>
            </div>
        </div>
    </div>
    <div class="card" style="padding: 50px;">
        <h1>🚀 Система тестування Java OOP</h1>
        <p style="font-size: 1.2em; color: #666;">Ласкаво просимо! Перевірте свої знання або керуйте навчальним процесом.</p>
        <div class="role-selection" style="margin-top: 40px; display: flex; flex-direction: column; align-items: center; gap: 15px;">
            <a href="/login" class="button" style="padding: 20px 40px; font-size: 1.2em; width: 100%; max-width: 300px;">Увійти в кабінет</a>
            <span style="color: #6b7280; font-size: 0.9em;">* Логін та пароль надає викладач або адміністратор</span>
        </div>
        <div style="margin-top: 50px; display: flex; justify-content: center; gap: 30px; flex-wrap: wrap;">
            <div class="card" style="flex: 1; max-width: 300px;">
                <h3>📖 Тести</h3>
                <p>Багато тем з об'єктно-орієнтованого програмування.</p>
            </div>

            <div class="card" style="flex: 1; max-width: 300px;">
                <h3>📊 Аналітика</h3>
                <p>Вчителі можуть бачити детальний прогрес кожного.</p>
```

```
        </div>
      </div>
   </div>
</div>
</body>
</html>
```

Login.html

```
<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
   <meta charset="UTF-8">
   <title>Вхід</title>
   <link rel="stylesheet" th:href="@{/style.css}">
</head>
<body>
<div class="container welcome">
   <h1>Вхід у систему</h1>
   <div class="card" style="max-width:500px; margin:0 auto;">
      <form th:action="@{/login}" method="post">
         <label><strong>Логін:</strong></label>
         <input type="text" name="username" required style="width:100%; padding:10px; margin:10px 0
20px;">
         <label><strong>Пароль:</strong></label>
         <div style="position: relative; width: 100%; margin: 10px 0 20px;">
            <input type="password" id="loginPassword" name="password" required style="width:100%;
padding:10px; padding-right: 40px; box-sizing: border-box; border-radius: 4px; border: 1px solid #ccc;">
            <span onclick="togglePassword('loginPassword', this)" style="position: absolute; right: 10px; top:
50%; transform: translateY(-50%); cursor: pointer; user-select: none; font-size: 18px;">👁      </span>
         </div>
         <button type="submit" class="button" style="width:100%;">Увійти</button>
      </form>
      <p th:if="${error}" style="color:red; margin-top:15px;" th:text="${error}"></p>
      <p th:if="${success}" style="color:green; margin-top:15px;" th:text="${success}"></p>
      <hr style="margin:20px 0;">
      <p style="color: #4b5563;">Немає акаунта? <strong>Зверніться до адміністратора</strong></p>
      <div style="margin: 10px 0;">
         <input type="checkbox" name="remember" id="remember">
         <label for="remember">Запам'ятати мене</label>
      </div>
      <div th:if="${param.error != null and param.error[0] == 'maintenance' and isMaintenance}"
          style="background-color: #fff3cd; color: #856404; padding: 10px; border-radius: 5px;">
         <i class="fas fa-tools"></i>
         <strong>Увага!</strong> На сайті проводяться техроботи.
      </div>
      <p style="margin-top:20px; font-size: 0.9em; color: #6b7280;"><strong>Підказка для
перевірки:</strong> admin/admin123</p>
   </div>
</div>
<script>
   function togglePassword(inputId, icon) {
      const input = document.getElementById(inputId);
      if (input.type === "password") {
         input.type = "text";
         icon.textContent = "🙈";
```

```
        } else {
            input.type = "password";
            icon.textContent = "👁";
        }
    }
</script>
</body>
</html>
```

Register.html

```
<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Реєстрація</title>
    <link rel="stylesheet" th:href="@{/style.css}">
</head>
<body>
<div class="container welcome">
    <h1>Реєстрація користувача</h1>
    <div class="card" style="max-width:500px; margin:0 auto;">
        <form th:action="@{/register}" method="post">
            <label><strong>Логін:</strong></label>
            <input type="text" name="username" required style="width:100%; padding:10px; margin:10px 0
20px;">
            <label><strong>Пароль:</strong></label>
            <div style="position: relative; width: 100%; margin: 10px 0 20px;">
                <input type="password" id="regPassword" name="password" required style="width:100%;
padding:10px; padding-right: 40px; box-sizing: border-box; border-radius: 4px; border: 1px solid #ccc;">
                <span onclick="togglePassword('regPassword', this)" style="position: absolute; right: 10px; top: 50%;
transform: translateY(-50%); cursor: pointer; user-select: none; font-size: 18px;">👁</span>
            </div>
            <label><strong>Хто ви?:</strong></label>
            <select name="role" required style="width:100%; padding:12px; margin:10px 0 20px; border-
radius:12px; border:1px solid #d1d5db;">
                <option value="STUDENT">Учень</option>
                <option value="TEACHER">Вчитель</option>
            </select>
            <button type="submit" class="button" style="width:100%;">Зареєструватися</button>
        </form>
        <p th:if="${error}" style="color:red; margin-top:15px;" th:text="${error}"></p>
        <hr style="margin:20px 0;">
        <p>Вже є акаунт? <a href="/login">Увійти</a></p>
    </div>
</div>
<script>
    function togglePassword(inputId, icon) {
        const input = document.getElementById(inputId);
        if (input.type === "password") {
            input.type = "text";
            icon.textContent = "🙈";
        } else {
            input.type = "password";
            icon.textContent = "👁";
```

```
        }
    }
</script>
<div th:replace="~{fragments :: chat}"></div>
</body>
</html>
```

Result-details.html

```
<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Деталі проходження</title>
    <link rel="stylesheet" th:href="@{/style.css}">
    <script src="https://cdn.tailwindcss.com"></script>
</head>
<body class="bg-gray-100 py-10">
<div class="max-w-4xl mx-auto px-4">
    <div class="flex justify-between items-center mb-6">
        <h1 class="text-2xl font-bold">Деталі проходження тесту</h1>
        <a th:href="${session.teacher != null ? '/teacher' : '/student'}" class="button-secondary" style="display:
flex; align-items: center; gap: 8px;">
        Назад
        </a>
    </div>
    <div class="bg-white p-6 rounded-lg shadow-md mb-8">
        <h2 class="text-xl font-bold mb-4" th:text="${result.testTitle}">Назва тесту</h2>
        <div class="grid grid-cols-2 gap-4 text-sm">
            <p><strong>Студент:</strong> <span th:text="${result.studentName}"></span></p>
            <p><strong>Результат:</strong> <span th:text="${result.score} + ' / ' +
${result.totalQuestions}"></span></p>
            <p><strong>Час виконання:</strong> <span th:text="${result.timeSpentSeconds} + '
сек.'"></span></p>
            <p><strong>Режим:</strong> <span th:text="${result.mode == 'exam' ? 'Екзамен' :
'Звичайний'}"></span></p>
        </div>
    </div>
    <h3 class="text-lg font-bold mb-4">Аналіз відповідей</h3>
    <div th:each="detail, stat : ${result.details}" class="bg-white p-4 rounded-lg shadow mb-4 border-l-8"
         th:styleappend="${detail.isCorrect} ? 'border-color: #22c55e;' : 'border-color: #ef4444;'">
        <p class="font-bold mb-2" th:text="${stat.index + 1} + '. ' + ${detail.questionText}"></p>
        <p class="text-sm">Ваша відповідь:
            <span th:text="${detail.studentAnswer}"
                  th:class="${detail.isCorrect ? 'text-green-600 font-bold' : 'text-red-600 font-bold line-
through'}"></span>
        </p>
        <p th:if="${!detail.isCorrect}" class="text-sm mt-1">
            <strong class="text-green-600">Правильна відповідь:</strong>
            <span class="text-green-600 font-bold" th:text="${detail.correctAnswer}"></span>
        </p>
        <p th:if="${detail.isCorrect}" class="text-green-600 text-sm font-bold mt-1">✅ Правильно</p>
    </div>
</div>
<div th:replace="~{fragments :: chat}"></div>
```

```
</body>
</html>
```
Results.html
```
<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Результат тесту</title>
    <link rel="stylesheet" th:href="@{/style.css}">
</head>
<body>
<div class="container welcome">
    <h1>Ваш результат</h1>
    <div class="card">
        <h2 th:text="'Тест: ' + ${result.testTitle}">Назва тесту</h2>
        <p><strong>Студент:</strong> <span th:text="${result.studentName}">Іван</span></p>
        <p class="result-score">
            Ви набрали <strong th:text="${result.score}">2</strong> з
            <strong th:text="${result.totalQuestions}">3</strong>
        </p>
        <p><strong>Макс. серія:</strong> <span th:text="${result.streak}">3</span> правильних підряд</p>
        <p><strong>Час:</strong> <span th:text="${result.timeSpentSeconds}">20</span> сек</p>
        <p><strong>Режим:</strong> <span th:text="${result.mode == 'exam' ? 'Екзамен' :
'Звичайний'}">Звичайний</span></p>
    </div>
    <a href="/student" class="button" style="margin-top:20px;">Повернутись до списку тестів</a>
</div>
<div th:replace="~{fragments :: chat}"></div>
</body>
</html>
```
Student.html
```
<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Панель учня</title>
    <link rel="stylesheet" th:href="@{/style.css}">
</head>
<body>
<div class="container">
    <div class="header">
        <h1>🎓 Кабінет студента</h1>
        <div style="display:flex; gap:10px; align-items: center;">
            <span style="font-weight: bold; color: #4f46e5;">Привіт, <span
th:text="${session.username}">Студент</span>!</span>
            <a th:if="${session.admin != null}"
               href="/admin"
               class="button-secondary"
               style="background-color: #e5e7eb; color: black; margin-right: 10px;">
                🏠 Головна панель
            </a>
            <a href="/logout" class="button-secondary" style="background-color: #e74c3c; color:
white;">🚪Вихід</a>
```

```
    </div>
  </div>
  <h2>Доступні тести</h2>
  <div class="card-list">
    <div class="card" th:each="test : ${tests}">
      <h2 th:text="${test.title}">Назва тесту</h2>
      <p th:text="${#lists.size(test.questions)} + ' питань'">Кількість питань</p>

      <div style="display:flex; gap:10px; flex-wrap:wrap;">
        <a th:href="@{/student/test/{id}(id=${test.id}, mode='normal')}" class="button">
          Пройти (звичайний режим)
        </a>
        <a th:href="@{/student/test/{id}(id=${test.id}, mode='exam')}" class="button-secondary">
          Екзамен-режим
        </a>
      </div>
    </div>
    <div th:if="${#lists.isEmpty(tests)}">
      <p>Наразі немає доступних тестів.</p>
    </div>
  </div>
  <hr style="margin:40px 0;">
  <h2>☑ Мої результати та статистика</h2>
  <div class="card" style="margin-bottom: 20px; background: #f0fdf4; border: 1px solid #bbf7d0;">
    <p><strong>Всього спроб:</strong> <span th:text="${#lists.size(myResults)}">0</span></p>
    <p><strong>Найкращий результат:</strong> <span th:text="${maxScore} + ' балів'">0
балів</span></p>
  </div>
  <div class="card">
    <table style="width:100%; border-collapse: collapse;" th:if="${!#lists.isEmpty(myResults)}">
      <thead>
      <tr>
        <th style="text-align:left; padding:10px; border-bottom: 2px solid #eee;">Тест</th>
        <th style="text-align:left; padding:10px; border-bottom: 2px solid #eee;">Результат</th>
        <th style="text-align:left; padding:10px; border-bottom: 2px solid #eee;">Режим</th>
        <th style="text-align:left; padding:10px; border-bottom: 2px solid #eee;">Час</th>
        <th style="text-align:left; padding:10px; border-bottom: 2px solid #eee;">Дії</th>
      </tr>
      </thead>
      <tbody>
      <tr th:each="r : ${myResults}">
        <td style="padding:10px; border-bottom: 1px solid #eee;" th:text="${r.testTitle}">Назва тесту</td>
        <td style="padding:10px; border-bottom: 1px solid #eee;">
          <span th:text="${r.score} + ' / ' + ${r.totalQuestions}"
              th:style="${r.score == r.totalQuestions} ? 'color: green; font-weight: bold;' : ''"></span>
        </td>
        <td style="padding:10px; border-bottom: 1px solid #eee;">
          <span th:text="${r.mode == 'exam' ? 'Екзамен' : 'Звичайний'}"
              th:class="${r.mode == 'exam' ? 'exam-badge' : ''}"></span>
        </td>
        <td style="padding:10px; border-bottom: 1px solid #eee;" th:text="${r.timeSpentSeconds} + '
сек'">30 сек</td>
        <td style="padding:10px; border-bottom: 1px solid #eee; display: flex; gap: 8px; align-items: center;">
```

```
                <a th:href="@{/student/result/{id}(id=${r.id})}" class="button-secondary" style="font-size: 13px;
padding: 5px 10px;">Деталі 🔍</a>

                <form th:if="${session.admin != null}" th:action="@{/student/result/delete/{id}(id=${r.id})}"
method="post" style="margin: 0; display: inline-flex;">
                    <button type="submit"
                            onclick="return confirm('Ви впевнені, що хочете видалити цей результат?');"
                            style="background: none; border: none; font-size: 16px; cursor: pointer; padding: 0 5px;"
                            title="Видалити результат">
                        ✕
                    </button>
                </form>
            </td>
        </tr>
        </tbody>
    </table>
    <div th:if="${#lists.isEmpty(myResults)}" style="text-align: center; padding: 20px; color: #666;">
        <p>Ви ще не проходили жодного тесту. Ваші результати з'являться тут.</p>
    </div>
  </div>
</div>
<div th:replace="~{fragments :: chat}"></div>
</body>
</html>
Teacher.html
<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Панель викладача</title>
  <link rel="stylesheet" th:href="@{/style.css}">
</head>
<body>
<div class="container">
  <div class="header">
    <h1>👨‍🏫 Панель викладача</h1>
    <div class="nav-buttons">
        <a th:if="${session.admin != null}" href="/admin" class="button-secondary" style="background-color:
#2c3e50; color: white;">
            🛡️ Адмін Панель
        </a>
        <a href="/teacher/students" class="button-secondary">👥 Студенти</a>
        <a href="/teacher/test/new" class="button-secondary">➕ Новий тест</a>
        <a href="/logout" class="button-secondary" style="background-color: #e74c3c; color:
white;">Вихід</a>

    </div>
  </div>
  <div class="card" style="border-top: 4px solid #8e44ad;">
    <h2>✨ Генератор тестів</h2>
    <form action="/teacher/test/generate-ai" method="get" style="display: flex; gap: 10px; margin-top: 15px;">
        <input type="text" name="topic" placeholder="Введіть тему для тесту..." required
```

```
            style="flex: 1; padding: 10px; border: 1px solid #ddd; border-radius: 5px;">
        <button type="submit" class="button-primary" style="background-color: #4f46e5;color:
white;">Згенерувати</button>
      </form>
    </div>
    <div class="card">
      <div style="display: flex; justify-content: space-between; align-items: center; margin-bottom: 20px;">
        <h2 style="margin: 0;">📄 Керування тестами</h2>
        <a th:href="@{/teacher(showArchived=${!showArchived})}" class="button-secondary">
          <span th:text="${showArchived} ? '🗀 Сховати архів' : '🗀 Показати архів'"></span>
        </a>
      </div>
      <table>
        <thead>
        <tr>
          <th style="text-align: center;">Назва тесту</th>
          <th style="text-align: center;">Статус</th>
          <th style="text-align: center;">Дії</th>
        </tr>
        </thead>
        <tbody style="text-align: center;">
        <tr th:each="t : ${tests}" th:if="${t != null}" th:style="${t.archived} ? 'opacity: 0.6; background:
#f9f9f9;' : ''">
          <td>
            <strong th:text="${t.title}"></strong>
            <span th:if="${t.archived}" style="color: #e74c3c; font-weight: bold;"> (АРХІВ)</span>
          </td>
          <td>
            <span th:if="${t.archived}" style="color: #e74c3c;">Архів</span>
            <span th:if="${!t.archived && t.hidden}" style="color: #7f8c8d;">Прихований</span>
            <span th:if="${!t.archived && !t.hidden}" style="color: #27ae60;">Активний</span>
          </td>
          <td>
            <div style="display: flex; justify-content: center; gap: 10px;">
              <form th:if="${!t.archived}" th:action="@{/teacher/test/toggle-visibility/{id}(id=${t.id})}"
method="post">
                <button type="submit" class="button-secondary" th:text="${t.hidden ? '👁 ' : '🚫'}"
title="Змінити видимість"></button>
              </form>

              <form th:if="${!t.archived}" th:action="@{/teacher/test/archive/{id}(id=${t.id})}"
method="post">
                <button type="submit" class="button-secondary" title="В архів">📄</button>
              </form>
              <form th:if="${t.archived}" th:action="@{/teacher/test/unarchive/{id}(id=${t.id})}"
method="post">
                <button type="submit" class="button-secondary" title="Відновити">🔄</button>
              </form>

              <a th:if="${!t.archived}" th:href="@{/teacher/test/edit/{id}(id=${t.id})}" title="Редагувати
тест" class="button-secondary">📝</a>
              <a th:href="@{/teacher/test/delete/{id}(id=${t.id})}" class="button-secondary"
```

```
                    style="color: #e74c3c; border-color: #e74c3c;" title="Видалити" onclick="return
confirm('Видалити тест?')">🗑      </a>
                </div>
            </td>
          </tr>
          </tbody>
        </table>
    </div>
</div>
<div th:replace="~{fragments :: chat}"></div>
</body>
</html>
```

Teacher-students.html

```
<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Список студентів</title>
    <link rel="stylesheet" th:href="@{/style.css}">
</head>
<body>
<div class="container">
    <div class="header">
        <h1>👥 Список студентів</h1>
        <a href="/teacher" class="button-secondary">Назад до панелі</a>
    </div>
    <div class="card">
        <table>
            <thead>
            <tr>
                <th>Логін студента</th>
                <th>Роль у системі</th>
                <th style="text-align: center;">Статистика</th>
            </tr>
            </thead>
            <tbody style="text-align: center;">
            <tr th:each="student : ${students}">
                <td>
                    <strong th:text="${student.username}"></strong>
                </td>
                <td>Студент</td>
                <td style="text-align: center;">
                    <a th:href="@{/teacher/user-stats/{username}(username=${student.username})}" class="button-
secondary">
                        📊 Результати
                    </a>
                </td>
            </tr>
            <tr th:if="${#lists.isEmpty(students)}">
```

```
            <td colspan="3" style="text-align: center; padding: 20px; color: #7f8c8d;">Студентів не
знайдено</td>
          </tr>
          </tbody>
        </table>
    </div>
</div>
<div th:replace="~{fragments :: chat}"></div>
</body>
</html>
Test.html
<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title th:text="${test.title}">Проходження тесту</title>
    <link rel="stylesheet" th:href="@{/style.css}">
</head>
<body>
<div class="container">
    <div class="header">
        <h1 th:text="${test.title}">Назва тесту</h1>
        <a href="/student" class="button-secondary">До списку тестів</a>
    </div>
    <div class="card" style="margin-bottom:20px;">
        <p><strong>Студент:</strong> <span th:text="${username}">student</span></p>
        <p><strong>Режим:</strong>
            <span th:text="${mode == 'exam' ? 'Екзамен' : 'Звичайний'}">Звичайний</span>
        </p>
        <p><strong>□ Таймер:</strong> <span id="timer">10</span> сек на питання</p>
        <p><strong>📊 Прогрес:</strong> <span id="progressText">1</span>/<span
th:text="${sessionQuestions.size()}">5</span></p>
        <div class="progress-container">
            <div class="progress-bar" id="progressBar"></div>
        </div>
        <p th:if="${mode == 'exam'}" style="color:crimson;">
            ⚠ Екзамен-режим: без повернення назад, час обмежений.
        </p>
    </div>
    <form id="testForm" th:action="@{/student/submit/{id}(id=${test.id})}" method="post">
        <input type="hidden" name="mode" th:value="${mode}">
        <div th:each="question, stat : ${sessionQuestions}" class="card question-card question-block"
             th:attr="data-index=${stat.index}"
             th:style="${stat.index == 0} ? 'display:block;' : 'display:none;'">
            <h3 th:text="${(stat.index + 1)} + '. ' + ${question.text}">Текст питання</h3>
            <div th:each="option, optStat : ${question.options}" class="option">
                <input type="radio"
                       th:name="'q' + ${stat.index}"
                       th:id="'q' + ${stat.index} + 'o' + ${optStat.index}"
                       th:value="${optStat.index}" />
                <label th:for="'q' + ${stat.index} + 'o' + ${optStat.index}" th:text="${option}">
                    Варіант відповіді
                </label>
```

```
            </div>
            <div style="margin-top:20px; display:flex; gap:10px;">
                <button type="button" class="button-secondary prevBtn" th:if="${stat.index > 0}">Назад</button>
                <button type="button" class="button nextBtn"
                        th:text="${stat.index == sessionQuestions.size() - 1 ? 'Завершити' : 'Далі'}">
                    Далі
                </button>
            </div>
        </div>
    </form>
</div>
<script th:inline="javascript">
    const mode = /*[[${mode}]]*/ 'normal';
    const totalQuestions = /*[[${sessionQuestions.size()}]]*/ 1;
    const questionBlocks = document.querySelectorAll('.question-block');
    const timerEl = document.getElementById('timer');
    const form = document.getElementById('testForm');
    const progressBar = document.getElementById('progressBar');
    const progressText = document.getElementById('progressText');

    let current = 0;
    let timeLeft = 20;
    let timer;

    function updateProgress() {
        const percent = ((current + 1) / totalQuestions) * 100;
        progressBar.style.width = percent + '%';
        progressText.textContent = current + 1;
    }

    function showQuestion(index) {
        questionBlocks.forEach((q, i) => {
            q.style.display = i === index ? 'block' : 'none';
        });
        updateProgress();
        resetTimer();
    }

    function resetTimer() {
        clearInterval(timer);
        timeLeft = 20;
        timerEl.textContent = timeLeft;

        timer = setInterval(() => {
            timeLeft--;
            timerEl.textContent = timeLeft;

            if (timeLeft <= 0) {
                clearInterval(timer);
                goNextAutomatically();
            }
        }, 1000);
    }
```

```
// НОВА ФУНКЦІЯ: Перевірка, чи обрано відповідь
function validateCurrentQuestion() {
    const currentBlock = questionBlocks[current];
    const isChecked = currentBlock.querySelector('input[type="radio"]:checked');

    if (!isChecked) {
        alert('Будь ласка, оберіть варіант відповіді перед тим, як продовжити!');
        return false;
    }
    return true;
}

function goNextAutomatically() {
    // Якщо час вийшов, але відповідь не обрана - зупиняємо тест і чекаємо вибору
    if (!validateCurrentQuestion()) {
        timerEl.textContent = "0 (Оберіть відповідь!)";
        return;
    }

    if (current < questionBlocks.length - 1) {
        current++;
        showQuestion(current);
    } else {
        form.submit();
    }
}

document.querySelectorAll('.nextBtn').forEach((btn) => {
    btn.addEventListener('click', () => {
        // Перевіряємо перед переходом
        if (!validateCurrentQuestion()) {
            return; // Зупиняємо дію, якщо нічого не обрано
        }

        if (current < questionBlocks.length - 1) {
            current++;
            showQuestion(current);
        } else {
            form.submit();
        }
    });
});

document.querySelectorAll('.prevBtn').forEach((btn) => {
    btn.addEventListener('click', () => {
        if (mode === 'exam') {
            alert('У екзамен-режимі не можна повертатися назад!');
            return;
        }

        if (current > 0) {
            current--;
```

```
            showQuestion(current);
        }
    });
});

document.addEventListener('copy', function(e) {
    e.preventDefault();
    alert('Копіювання під час тесту заборонено!');
});

document.addEventListener('contextmenu', function(e) {
    e.preventDefault();
});

window.onbeforeunload = function () {
    return "Ви впевнені, що хочете покинути тест?";
};

showQuestion(0);
</script>
</body>
</html>
```

User.html

```
<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Статистика студента</title>
    <link rel="stylesheet" th:href="@{/style.css}">
</head>
<body>
<div class="container">
    <div class="header">
        <h1>📊 Результати: <span th:text="${targetUser?.username}">Студент</span></h1>

        <a th:if="${session.admin != null}" href="/admin/users" class="button-secondary">← До списку користувачів</a>
        <a th:if="${session.admin == null}" href="/teacher/students" class="button-secondary">← До списку студентів</a>
    </div>

    <div class="card" style="display: flex; justify-content: space-around; text-align: center; margin-bottom: 20px;">
        <div>
            <h3 style="margin-bottom: 5px;">Середній бал</h3>
            <p style="font-size: 1.5em; font-weight: bold; color: #3498db;"
               th:text="${averageScore != null ? averageScore + '%' : '–'}"></p>
        </div>
        <div>
            <h3 style="margin-bottom: 5px;">Тестів складено</h3>
            <p style="font-size: 1.5em; font-weight: bold; color: #2ecc71;"
               th:text="${results != null ? #lists.size(results) : 0}"></p>
        </div>
```

```
    </div>

    <div class="card">
        <h2>Історія успішності</h2>

        <table th:if="${results != null and !#lists.isEmpty(results)}">
            <thead>
            <tr>
                <th>Назва тесту</th>
                <th>Результат</th>
                <th>Дата</th>
                <th>Дії</th>
            </tr>
            </thead>
            <tbody style="text-align: center;">
            <tr th:each="r : ${results}">
                <td th:text="${r.testTitle}"></td>
                <td>
                    <span th:text="${r.score} + ' / ' + ${r.totalQuestions}"></span>
                </td>
                <td th:text="${r.dateTime != null ? #temporals.format(r.dateTime, 'dd.MM.yyyy HH:mm') : '–
'}"></td>
                <td style="display: flex; justify-content: center; gap: 8px; align-items: center;">
                    <a th:href="@{/teacher/result/{id}(id=${r.id})}" class="button-secondary">Переглянути</a>

                    <form th:if="${session.admin != null}"
                          th:action="@{/admin/users/{username}/delete-
result/{resultId}(username=${targetUser.username}, resultId=${r.id})}"
                          method="post" style="margin: 0; display: inline-flex;">
                        <button type="submit" onclick="return confirm('Видалити цей результат?');"
style="background: none; border: none; font-size: 16px; cursor: pointer; padding: 5px;">✕</button>
                    </form>

                    <form th:if="${session.admin == null and session.teacher != null}"
                          th:action="@{/teacher/user-stats/{username}/delete-
result/{resultId}(username=${targetUser.username}, resultId=${r.id})}"
                          method="post" style="margin: 0; display: inline-flex;">
                        <button type="submit" onclick="return confirm('Видалити цей результат?');"
style="background: none; border: none; font-size: 16px; cursor: pointer; padding: 5px;">✕</button>
                    </form>
                </td>
            </tr>
            </tbody>
        </table>

        <div th:if="${results == null or #lists.isEmpty(results)}" style="text-align: center; padding: 20px; color:
#666;">
            <p>Студент ще не проходив жодного тесту.</p>
        </div>
    </div>
</div>
</body>
</html>
```

Application.properties

```
server.port=8080

# MySQL
spring.datasource.url=jdbc:mysql://localhost:3306/oop_test_db?createDatabaseIfNotExist=true&serverTimezone=UTC&characterEncoding=UTF-8
spring.datasource.username=root
spring.datasource.password=1234
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# Настройки Hibernate/JPA
spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

question.json

```
[
  {
    "category": "Основи ООП",
    "text": "Який принцип ООП відповідає за приховування внутрішньої реалізації класу?",
    "options": ["Інкапсуляція", "Поліморфізм", "Наслідування", "Абстракція"],
    "correctAnswerIndex": 0
  },
  {
    "category": "Основи ООП",
    "text": "Що таке об'єкт у контексті ООП?",
    "options": ["Клас", "Екземпляр класу", "Тип даних", "Функція"],
    "correctAnswerIndex": 1
  },
  {
    "category": "Інкапсуляція",
    "text": "Який модифікатор доступу робить поле видимим лише всередині поточного класу?",
    "options": ["public", "protected", "default", "private"],
    "correctAnswerIndex": 3
  },
  {
    "category": "Наслідування",
    "text": "Яке ключове слово використовується в Java для наслідування класу?",
    "options": ["implements", "extends", "inherits", "this"],
    "correctAnswerIndex": 1
  },
  {
    "category": "Поліморфізм",
    "text": "Як називається можливість методу виконувати різні дії залежно від об'єкта, що його викликає?",
    "options": ["Перевантаження", "Інкапсуляція", "Поліморфізм", "Наслідування"],
    "correctAnswerIndex": 2
  },
  {
    "category": "Абстракція",
    "text": "Чи можна створити екземпляр абстрактного класу за допомогою оператора new?",
    "options": ["Так", "Ні", "Тільки якщо в класі немає методів", "Тільки в пакеті java.lang"],
    "correctAnswerIndex": 1
  },
```

```
  {
    "category": "Інтерфейси",
    "text": "Яке ключове слово використовується для реалізації інтерфейсу в Java?",
    "options": ["extends", "implements", "instanceof", "interface"],
    "correctAnswerIndex": 1
  },
  {
    "category": "Конструктори",
    "text": "Яке основне призначення конструктора класу?",
    "options": ["Знищення об'єкта", "Створення копії класу", "Ініціалізація початкового стану об'єкта",
"Виклик статичних методів"],
    "correctAnswerIndex": 2
  },
  {
    "category": "Поліморфізм",
    "text": "Що таке Overriding (перевизначення) методу?",
    "options": ["Зміна логіки батьківського методу у дочірньому класі", "Створення методу з тим же ім'ям,
але іншими параметрами", "Видалення методу", "Зміна модифікатора доступу на private"],
    "correctAnswerIndex": 0
  },
  {
    "category": "Поліморфізм",
    "text": "Що таке Overloading (перевантаження) методу?",
    "options": ["Зміна назви методу", "Кілька методів з однаковим ім'ям, але різними параметрами",
"Виклик методу самого себе", "Наслідування методу"],
    "correctAnswerIndex": 1
  },
  {
    "category": "Java Keywords",
    "text": "Що означає ключове слово final для класу?",
    "options": ["Клас не може мати конструкторів", "Від класу не можна наслідуватися", "Всі методи класу
мають бути статичними", "Клас автоматично стає інтерфейсом"],
    "correctAnswerIndex": 1
  },
  {
    "category": "Java Keywords",
    "text": "Для чого використовується ключове слово super?",
    "options": ["Для виклику методів поточного класу", "Для звернення до членів батьківського класу",
"Для створення нового об'єкта", "Для завершення програми"],
    "correctAnswerIndex": 1
  },
  {
    "category": "Java Keywords",
    "text": "Що означає ключове слово static для методу?",
    "options": ["Метод належить об'єкту", "Метод належить класу і викликається без створення об'єкта",
"Метод не може повертати значення", "Метод виконується в окремому потоці"],
    "correctAnswerIndex": 1
  },
  {
    "category": "Java Keywords",
    "text": "Яка роль ключового слова this?",
    "options": ["Посилання на батьківський клас", "Посилання на поточний екземпляр об'єкта", "Видалення
поточного об'єкта", "Створення статичної змінної"],
```

```
  "correctAnswerIndex": 1
 },
 {
  "category": "Абстракція",
  "text": "Чим відрізняється інтерфейс від абстрактного класу в Java 8+?",
  "options": ["В інтерфейсі не можна мати default методи", "Абстрактний клас може мати стан (поля), інтерфейс – ні", "Інтерфейс не може мати констант", "Немає ніякої різниці"],
  "correctAnswerIndex": 1
 },
 {
  "category": "Основи ООП",
  "text": "Від якого класу неявно наслідуються всі класи в Java?",
  "options": ["Main", "Class", "Object", "Base"],
  "correctAnswerIndex": 2
 },
 {
  "category": "Наслідування",
  "text": "Чи підтримує Java множинне наслідування класів?",
  "options": ["Так", "Ні", "Тільки для абстрактних класів", "Тільки через ключове слово multiextends"],
  "correctAnswerIndex": 1
 },
 {
  "category": "Інтерфейси",
  "text": "Чи може клас реалізовувати декілька інтерфейсів одночасно?",
  "options": ["Так", "Ні", "Тільки якщо вони в одному пакеті", "Тільки якщо вони порожні"],
  "correctAnswerIndex": 0
 },
 {
  "category": "Інкапсуляція",
  "text": "Який стандартний спосіб доступу до private полів класу?",
  "options": ["Через оператор точка", "Через методи геттери та сеттери", "Доступ неможливий", "Через рефлексію лише"],
  "correctAnswerIndex": 1
 },
 {
  "category": "Основи ООП",
  "text": "Який принцип ООП допомагає виділити лише значущі характеристики об'єкта?",
  "options": ["Абстракція", "Інкапсуляція", "Поліморфізм", "Композиція"],
  "correctAnswerIndex": 0
 },
 {
  "category": "Java Keywords",
  "text": "Який модифікатор доступу дозволяє бачити члени класу лише в межах поточного пакету та підкласам?",
  "options": ["public", "private", "protected", "default"],
  "correctAnswerIndex": 2
 },
 {
  "category": "Конструктори",
  "text": "Що станеться, якщо не прописати конструктор у класі?",
  "options": ["Програма не скомпілюється", "Java автоматично створить порожній конструктор за замовчуванням", "Клас не зможе мати полів", "Об'єкт буде створено випадковим чином"],
  "correctAnswerIndex": 1
```

```
  },
  {
    "category": "Основи ООП",
    "text": "Що таке агрегація?",
    "options": ["Тип наслідування", "Форма асоціації, де об'єкти можуть існувати незалежно", "Приховування даних", "Виклик одного конструктора з іншого"],
    "correctAnswerIndex": 1
  },
  {
    "category": "Основи ООП",
    "text": "Що таке композиція?",
    "options": ["Наслідування інтерфейсів", "Форма асоціації, де життєвий цикл частин залежить від цілого", "Метод порівняння об'єктів", "Тип циклу у Java"],
    "correctAnswerIndex": 1
  },
  {
    "category": "Екземпляри",
    "text": "Який оператор перевіряє, чи належить об'єкт до певного класу?",
    "options": ["is-a", "instanceof", "typeof", "check"],
    "correctAnswerIndex": 1
  },
  {
    "category": "Java Keywords",
    "text": "Що робить модифікатор transient?",
    "options": ["Робить змінну статичною", "Виключає поле з процесу серіалізації", "Дозволяє змінювати final поле", "Прискорює доступ до змінної"],
    "correctAnswerIndex": 1
  },
  {
    "category": "Java Keywords",
    "text": "Яке значення має ключове слово volatile?",
    "options": ["Зупиняє потік", "Гарантує видимість змінної між різними потоками", "Шифрує дані", "Дозволяє видаляти об'єкти вручну"],
    "correctAnswerIndex": 1
  },
  {
    "category": "Класи",
    "text": "Що таке внутрішній (inner) клас?",
    "options": ["Клас у тому ж пакеті", "Клас, описаний всередині іншого класу", "Клас, що не має методів", "Клас, назва якого починається з символу _"],
    "correctAnswerIndex": 1
  },
  {
    "category": "Класи",
    "text": "Як називається клас, який не має імені та створюється 'на льоту'?",
    "options": ["Абстрактний клас", "Інтерфейс", "Анонімний клас", "Приватний клас"],
    "correctAnswerIndex": 2
  },
  {
    "category": "Перерахування",
    "text": "Який тип даних у Java використовується для представлення фіксованого набору констант?",
    "options": ["Class", "Interface", "Enum", "Array"],
    "correctAnswerIndex": 2
```

```
  },
  {
    "category": "Основи ООП",
    "text": "Яка головна перевага наслідування?",
    "options": ["Швидкість виконання коду", "Повторне використання коду", "Зменшення пам'яті",
"Можливість не писати конструктори"],
    "correctAnswerIndex": 1
  },
  {
    "category": "Поліморфізм",
    "text": "Де визначається, який саме метод викликати при пізньому зв'язуванні (dynamic binding)?",
    "options": ["Під час компіляції", "Під час виконання (runtime)", "Під час створення JAR файлу", "В
налаштуваннях IDE"],
    "correctAnswerIndex": 1
  },
  {
    "category": "Абстракція",
    "text": "Чи може абстрактний клас не мати жодного абстрактного методу?",
    "options": ["Так", "Ні", "Тільки якщо він порожній", "Тільки якщо він наслідує інший клас"],
    "correctAnswerIndex": 0
  },
  {
    "category": "Java Keywords",
    "text": "Що станеться, якщо спробувати змінити значення final змінної?",
    "options": ["Значення зміниться", "Виникне помилка компіляції", "Значення стане null", "Програма
проігнорує дію"],
    "correctAnswerIndex": 1
  },
  {
    "category": "Основи ООП",
    "text": "Який зв'язок описує наслідування (Inheritance)?",
    "options": ["HAS-A", "IS-A", "USES-A", "PART-OF"],
    "correctAnswerIndex": 1
  },
  {
    "category": "Основи ООП",
    "text": "Який зв'язок описує композиція/агрегація?",
    "options": ["IS-A", "HAS-A", "BELONGS-TO", "WORKS-WITH"],
    "correctAnswerIndex": 1
  },
  {
    "category": "Інтерфейси",
    "text": "Чи можуть методи інтерфейсу мати тіло в Java 17?",
    "options": ["Ні, ніколи", "Так, якщо це default або static методи", "Тільки якщо інтерфейс приватний",
"Тільки у підкласах"],
    "correctAnswerIndex": 1
  },
  {
    "category": "Класи",
    "text": "Що таке конструктор копіювання?",
    "options": ["Метод clone()", "Конструктор, який приймає об'єкт свого ж класу як параметр",
"Статичний метод factory", "Оператор ="],
    "correctAnswerIndex": 1
```

```
  },
  {
    "category": "Поліморфізм",
    "text": "Яка анотація використовується в Java для підтвердження перевизначення методу?",
    "options": ["@Override", "@Overwrite", "@NewMethod", "@Inherited"],
    "correctAnswerIndex": 0
  },
  {
    "category": "Основи ООП",
    "text": "Як називається процес створення об'єкта за описом класу?",
    "options": ["Ініціалізація", "Інстанціювання (Instantiation)", "Декларація", "Визначення"],
    "correctAnswerIndex": 1
  }
]
```

Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.2.5</version>
        <relativePath/> </parent>
    <groupId>com.example</groupId>
    <artifactId>oop-test</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>oop-test</name>
    <description>Online testing platform for OOP in Java</description>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
        <dependency>
```

```
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>com.squareup.okhttp3</groupId>
            <artifactId>okhttp</artifactId>
            <version>4.12.0</version>
        </dependency>
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>com.mysql</groupId>
            <artifactId>mysql-connector-j</artifactId>
            <scope>runtime</scope>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>17</source>
                    <target>17</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```