

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**НАЦІОНАЛЬНИЙ ТРАНСПОРТНИЙ УНІВЕРСИТЕТ**  
Навчально-науковий інститут управління, технологій та правових наук  
Кафедра інформаційних технологій

**МАГІСТЕРСЬКА РОБОТА**

на тему:

**СИСТЕМА ДИСТАНЦІЙНОГО КОНТРОЛЮ ЗНАНЬ СТУДЕНТІВ  
«TESTIFY»**

Спеціальність: 121 «Інженерія програмного забезпечення»  
ОПП «Інженерія програмного забезпечення»

Розробив: Дмитрій ЗІНЧЕНКО  
(ім'я, ПРІЗВИЩЕ)

\_\_\_\_\_  
(підпис)

Керівник: к.ф.-м.н., доцент Олександр ТКАЧЕНКО\_  
(науковий ступінь, вчене звання) (ім'я, ПРІЗВИЩЕ)

\_\_\_\_\_  
(підпис)

Нормоконтроль: к.ф.-м.н., доцент Олександр ТКАЧЕНКО  
(науковий ступінь, вчене звання) (ім'я, ПРІЗВИЩЕ)

\_\_\_\_\_  
(підпис)

Допущений до захисту  
« \_\_\_\_ » \_\_\_\_\_ 2025р.

В.о. завідувача кафедри: д.т.н., професор Олег НІКОНОВ  
(науковий ступінь, вчене звання) (ім'я, ПРІЗВИЩЕ)

\_\_\_\_\_  
(підпис)

Рецензент: к.т.н., доцент Тетяна НОСЕНКО  
(науковий ступінь, вчене звання) (ім'я, ПРІЗВИЩЕ)

\_\_\_\_\_  
(підпис)

Київ – 2025

Міністерство освіти і науки України  
 Національний транспортний університет  
 Навчально-науковий інститут управління, технологій та правових наук  
 Кафедра інформаційних технологій

Рівень вищої освіти – другий (магістерський)  
 Галузь знань 12 «Інформаційні технології»  
 Спеціальність 121 «Інженерія програмного забезпечення»  
 ОПІ «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри  
 інформаційних технологій  
 д.т.н., проф. \_\_\_\_\_ Олег НІКОНОВ

«\_\_\_\_\_» \_\_\_\_\_ 2025 р.

**Завдання**  
**на магістерську роботу здобувачу вищої освіти**  
**Дмитрію ЗІНЧЕНКУ**

(ім'я, прізвище)

1. Тема роботи: Система дистанційного контролю знань студентів «Testify»

керівник роботи: к.ф.-м.н., доцент Олександр ТКАЧЕНКО

Затверджені наказом від «17» жовтня 2025 року № 1031

2. Строк подання здобувачем роботи 19 грудня 2025 року

3. Вихідні дані до роботи: веб-орієнтована система дистанційного навчання «Testify». База даних навчальних курсів, тестових завдань та облікових записів користувачів системи «Testify».

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз предметної області та існуючих програмних аналогів, розробка алгоритмів автоматизованого оцінювання знань. Побудова структурної та функціональної моделей системи, проєктування архітектури та бази даних. Програмна реалізація вебсервісу для дистанційного навчання на базі фреймворку Django, створення графічного інтерфейсу користувача для ролей викладача та студента.

5. Перелік графічного матеріалу: презентація магістерської роботи та скріншоти функціонування програмного продукту "Testify"

6. Консультанти роботи: к.ф.-м.н., доцент Олександр ТКАЧЕНКО

## 7. Календарний план виконання роботи

| № з/п | Найменування етапів магістерської роботи                   | Строк виконання етапів магістерської роботи | Примітки |
|-------|------------------------------------------------------------|---------------------------------------------|----------|
| 1     | Огляд та аналіз джерел тематичної області                  | 17.10.25 – 22.10.25                         |          |
| 2     | Огляд та аналіз шляхів розв’язання проблем роботи          | 23.10.25 – 28.10.25                         |          |
| 3     | Опис власного підходу до вирішення проблеми                | 29.10.25 – 04.11.25                         |          |
| 4     | Визначення вимог до програмного продукту та його структури | 05.11.25 – 07.11.25                         |          |
| 5     | Опис та реалізація основних функцій програмного продукту   | 08.11.25 – 28.11.25                         |          |
| 6     | Оформлення пояснювальної записки                           | 29.11.25 – 19.12.25                         |          |
| 7     | Захист                                                     |                                             |          |

## 8. Дата видачі завдання «17» жовтня 2025 року

Здобувач вищої освіти

\_\_\_\_\_

(підпис)

Зінченко Д.В.

(прізвище та ініціали)

Керівник роботи

\_\_\_\_\_

(підпис)

Ткаченко О.А.

(прізвище та ініціали)

## РЕФЕРАТ

Тема: «Система дистанційного контролю знань студентів «Testify»

Магістерська робота: 114 с., 29 рис., 37 літературних джерел, 1 додаток.

Магістерська робота присвячена розробці веб-орієнтованої інформаційної системи дистанційного контролю знань «Testify». Метою роботи є дослідження методів автоматизованого оцінювання, моделювання архітектури програмного забезпечення та створення ефективного інструменту для оптимізації процесів тестування в навчальних закладах.

Дослідження зосереджено на проектуванні та реалізації вебсервісу, що забезпечує повний цикл управління навчанням: від створення курсів та наповнення їх контентом до автоматичної перевірки відповідей студентів. Для реалізації функціоналу системи застосовано сучасні технології: мову програмування Python, вебфреймворк Django, систему управління базами даних PostgreSQL, а також HTML та CSS для побудови адаптивного інтерфейсу користувача.

У процесі розробки було проведено системний аналіз предметної області, сформульовано вимоги до безпеки та відмовостійкості, що дозволило забезпечити високу продуктивність системи при одночасній роботі багатьох користувачів.

Результатом роботи є впроваджена веб-орієнтована система «Testify», яка надає викладачам зручний конструктор тестів із різними варіантами відповідей, а студентам – інтуїтивний інтерфейс для проходження контролю знань із забезпеченням миттєвого зворотного зв'язку та збереженням історії успішності.

Ключові слова: веб-орієнтована система, дистанційне навчання, автоматизоване тестування, алгоритм оцінювання, Python, Django, база даних.

## ABSTRACT

Topic: "System for remote control of students' knowledge "Testify"

Master's thesis: 114 pages, 29 figures, 37 literary sources, 1 appendix.

The master's thesis is devoted to the development of a web-oriented information system for remote control of knowledge " Testify ". The purpose of the work is to study methods of automated assessment, modeling of software architecture and creating an effective tool for optimizing testing processes in educational institutions.

The research focuses on the design and implementation of a web service that provides a full cycle of learning management: from creating courses and filling them with content to automatically checking students' answers. To implement the functionality of the system, modern technologies were used: the Python programming language, the Django web framework, the PostgreSQL database management system, as well as HTML and CSS for building an adaptive user interface.

During the development process, a system analysis of the subject area was conducted, requirements for security and fault tolerance were formulated, which allowed to ensure high system performance with simultaneous work of many users. The result of the work is the implemented web-based system " Testify ", which provides teachers with a convenient test constructor with various answer options, and students with an intuitive interface for passing knowledge control with instant feedback and saving the history of success.

Keywords: web-based system, distance learning, automated testing, evaluation algorithm, Python, Django, database.

## ЗМІСТ

|                                                                                                                     |    |
|---------------------------------------------------------------------------------------------------------------------|----|
| ВСТУП                                                                                                               | 7  |
| 1 ОГЛЯД ТА АНАЛІЗ ТЕМАТИЧНОЇ ОБЛАСТІ – ДИСТАНЦІЙНОГО<br>КОНТРОЛЮ ЗНАНЬ СТУДЕНТІВ                                    | 10 |
| 1.1 Сутність дистанційного контролю знань студентів                                                                 | 10 |
| 1.2 Сучасний стан онлайн-тестування в освітньому процесі                                                            | 17 |
| 1.3 Сучасні системи дистанційного тестування знань студентів                                                        | 22 |
| 2 АНАЛІЗ ТА ШЛЯХИ РОЗВ’ЯЗАННЯ ПРОБЛЕМИ                                                                              | 32 |
| 2.1 Алгоритми обчислення результатів тестування                                                                     | 32 |
| 2.2 Структурна та функціональна моделі системи дистанційного<br>контролю знань студентів                            | 35 |
| 2.3 Стек технологій розробки системи дистанційного контролю знань<br>студентів                                      | 42 |
| 3 ОПИС ПРОГРАМНОГО ПРОДУКТУ – СИСТЕМИ<br>ДИСТАНЦІЙНОГО КОНТРОЛЮ ЗНАНЬ СТУДЕНТІВ «TESTIFY»                           | 53 |
| 3.1 Вимоги до програмного продукту та інтерфейсу користувача                                                        | 53 |
| 3.2 Структура інформаційної бази системи «Testify»                                                                  | 55 |
| 3.3 Основні функції та інтерфейс системи «Testify»                                                                  | 62 |
| 3.4 Опис функціонування системи «Testify» та особливостей реалізації<br>алгоритму обчислення результатів тестування | 77 |
| ВИСНОВКИ                                                                                                            | 78 |
| СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ                                                                                          | 80 |
| ДОДАТКИ                                                                                                             | 84 |

## ВСТУП

Сьогодення вимагає докорінних змін у підходах до перевірки знань. Традиційні методи оцінювання стають неефективними в умовах глобального переходу на дистанційний формат навчання. Викладачі та навчальні заклади потребують надійних інструментів, які дозволяють проводити тестування віддалено, зберігаючи при цьому об'єктивність результатів.

Однак, наявні на ринку рішення часто перевантажені зайвим функціоналом або є надто коштовними для впровадження в окремих навчальних групах. Саме тому актуальним стає розробка спеціалізованих веб-орієнтованих систем, таких як «Testify», що поєднують у собі легкість доступу, інтуїтивний інтерфейс та потужні механізми аналізу успішності студентів без необхідності встановлення складного програмного забезпечення.

**Актуальність проблеми**, що розглядається у даній магістерській роботі, полягає у тому, що зростає потреба у створенні адаптивних, легких у використанні та захищених інструментів для проведення онлайн-тестувань. Система «Testify» покликана вирішити проблему організації прозорого контролю знань в умовах віддаленого доступу, забезпечуючи при цьому зручний інтерфейс як для викладачів, так і для студентів.

**Науковою новизною** роботи є удосконалення методів організації дистанційного контролю знань шляхом розробки спеціалізованої інформаційної системи, яка, на відміну від існуючих аналогів, поєднує в собі гнучкість налаштування тестових завдань з сучасними механізмами авторизації та аналітики результатів.

**Метою роботи** є дослідження предметної області (дослідження дистанційного контролю знань студентів) на основі використання інформатизації та інтелектуалізації навчального процесу, що дозволить оптимізувати та підвищити ефективність, об'єктивність та індивідуалізацію процесу контролю знань студентів та розробити відповідну систему «Testify».

Для досягнення поставленої мети необхідно вирішити ряд завдань:

- зробити огляд існуючих систем дистанційного навчання та контролю знань, виявити їх переваги та недоліки;
- визначити основні підходи та методики проведення комп'ютерного тестування;
- проаналізувати сучасні технології веброзробки та обрати оптимальний стек технологій для реалізації системи;
- обґрунтувати архітектурні рішення та структуру бази даних системи;
- здійснити збір загальної інформації та експертних знань, необхідних для формування вимог до інтерфейсу користувача;
- розробити алгоритми роботи системи, включаючи механізми створення тестів, проходження їх студентами та автоматичного оцінювання;
- визначити вимоги до системи «Testify», її функції та основні компоненти (модуль авторизації, модуль тестування, модуль аналітики);
- розробити систему дистанційного контролю знань студентів «Testify» та провести її тестування.

**Об'єктом дослідження** в магістерській роботі є процес дистанційного контролю та оцінювання знань у вищих навчальних закладах.

**Предметом дослідження** в магістерській роботі є методи та програмні засоби створення системи дистанційного тестування знань «Testify».

**Практична цінність** магістерської роботи полягає у створенні працездатного програмного продукту, який дозволяє автоматизувати рутинну роботу викладача з перевірки робіт, зменшити вплив людського фактору на оцінку та забезпечити зручний доступ до навчальних матеріалів і тестів. Система «Testify» може бути впроваджена у навчальний процес університету.

Магістерська робота складається зі вступу, трьох основних розділів, висновків, списку використаних джерел та застосунку, в яких розглядаються теоретичні аспекти комп'ютерного тестування, проектування архітектури системи та її програмна реалізація.

*Особистий внесок магістра.* Магістерська робота є самостійним завершеним дослідженням автора.

*Публікації:* дослідження були апробовані:

- Ткаченко К.О., Зінченко Д.В. Вебзастосунок перевірки знань студентів з підтримкою автоматичних тестів та інтерактивних вправ. *Інформаційні технології в культурі, мистецтві, освіті, науці, економіці та бізнесі*: матеріали X міжнародної науково-практичної конференції (Київ, 24-25 квітня 2025 р.). Київ: КНУКіМ, 2025. С. 36-38. [36].
- результати дослідження були подані та прийняті до друку в науковій статті Ткаченко О.І., Зінченко Д.В. (2025). Онлайн-тестування – інструмент цифрової освіти: проблеми та перспективи. *IT Synergy*, № (1;2) (8;9) [37].

## 1 ОГЛЯД ТА АНАЛІЗ ТЕМАТИЧНОЇ ОБЛАСТІ – ДИСТАНЦІЙНОГО КОНТРОЛЮ ЗНАНЬ СТУДЕНТІВ

### 1.1 Сутність дистанційного контролю знань студентів

Сучасний етап розвитку вищої освіти характеризується фундаментальними трансформаційними процесами, що охоплюють усі складові навчально-виховного процесу, від методології викладання до інструментарію верифікації отриманих компетентностей. Глобалізація інформаційного простору, стрімка цифровізація суспільних відносин та вимушений перехід до дистанційних форм взаємодії, спричинений низкою кризових явищ глобального масштабу, актуалізували необхідність кардинального переосмислення сутності педагогічного контролю.

Традиційні підходи до оцінювання, що формувалися десятиліттями в умовах класичної класно-урочної системи, виявилися недостатньо гнучкими та адаптивними для забезпечення об'єктивності, надійності та валідності вимірювання знань у цифровому середовищі. Дистанційний контроль знань студентів, який тривалий час сприймався педагогічною спільнотою як допоміжний елемент заочної форми навчання або експериментальна практика, сьогодні трансформувався у самостійну педагогічну категорію та складний технологічний процес, що вимагає розробки спеціалізованих програмних рішень, таких як система «Testify».

Специфіка дистанційного навчання полягає у просторовому роз'єднанні суб'єктів навчальної діяльності, що докорінно змінює архітектуру педагогічної комунікації та вимагає впровадження нових, технологічно опосередкованих механізмів управління якістю освіти, які базуються на використанні сучасних інформаційно-комунікаційних технологій.

У загальнотеоретичному плані контроль знань розглядається як невід'ємна, органічна складова процесу навчання, що забезпечує критично

важливий зворотний зв'язок між викладачем і студентом, дозволяючи оцінити ступінь досягнення поставлених освітніх цілей та рівень сформованості професійних компетентностей.

Однак в умовах дистанційної освіти сутність контролю суттєво трансформується, набуваючи ознак кібернетичного процесу управління. Це вже не просто перевірка пам'яті чи здатності механічно відтворити засвоєний матеріал, а складна, багаторівнева система моніторингу, яка включає в себе діагностику поточного стану, прогнозування результатів, корекцію навчальної траєкторії та фінальну верифікацію досягнень.

На відміну від традиційного усного чи письмового іспиту в аудиторії, де значну роль відіграє безпосередній особистісний контакт, психологічна взаємодія та суб'єктивне сприйняття викладачем особистості студента, дистанційний контроль базується на формалізованих алгоритмах, стандартизованих процедурах та чітко визначених критеріях оцінювання.

Така технологізація процесу дозволяє мінімізувати вплив суб'єктивних факторів, забезпечуючи більш високий рівень об'єктивності, проте водночас висуває підвищені вимоги до якості контрольних-вимірювальних матеріалів та надійності програмно-апаратних комплексів [1].

Функціональне поле дистанційного контролю є надзвичайно широким і багатовекторним, включаючи в себе діагностичну, навчальну, виховну, розвивальну та оцінювальну функції, кожна з яких в умовах онлайн-навчання реалізується через специфічні цифрові інструменти. Діагностична функція, яка традиційно спрямована на виявлення рівня знань, умінь та навичок, у сучасних дистанційних системах реалізується через механізми вхідного тестування та безперервного моніторингу активності здобувачів освіти.

Особливістю реалізації цієї функції в цифровому середовищі є можливість миттєвого отримання та обробки результатів діагностики, що дозволяє студенту оперативно усвідомити прогалини у власних знаннях та своєчасно скоригувати зусилля для їх усунення. Навчальна функція дистанційного контролю заслуговує на особливу увагу, оскільки саме в умовах

електронного навчання межа між власне процесом здобуття знань та процедурою їх перевірки стає дедалі більш умовною та проникною. Використання інтерактивних тренувальних тестів з автоматизованими коментарями до правильних і неправильних відповідей перетворює процес оцінювання на потужний дидактичний інструмент самонавчання.

Правильно організований контроль сприяє поглибленню, розширенню та систематизації знань, і в дистанційному режимі це досягається шляхом багаторазового проходження тренувальних тестів та роботи над помилками без психологічного тиску з боку викладача, що дозволяє реалізувати навчальну функцію контролю на якісно новому рівні [2].

Важливим аспектом сутності дистанційного контролю є його орієнтація на формувальне оцінювання, яке націлене не стільки на фіксацію кінцевого результату, скільки на підтримку та спрямування самого процесу навчання. У науковій літературі наголошується, що ефективний онлайн-контроль повинен забезпечувати регулярний, змістовний та конструктивний зворотний зв'язок, який допомагає студенту відслідковувати власний прогрес та розвивати навички саморегуляції.

Системи дистанційного навчання дозволяють реалізувати це через автоматизовану перевірку завдань, надання розгорнутих пояснень до помилок, а також через інструменти взаємного оцінювання, коли студенти рецензують роботи один одного за визначеними критеріями. Такий підхід не лише знижує навантаження на викладача, але й сприяє розвитку критичного мислення, аналітичних здібностей та глибокого розуміння предметної області самими студентами. Впровадження елементів гейміфікації та рейтингових систем додатково стимулює пізнавальну активність та підтримує мотивацію до навчання в умовах фізичної ізоляції від академічної спільноти.

Сутність дистанційного контролю в цьому контексті полягає у переході від парадигми "оцінювання навчання" до парадигми "оцінювання для навчання", де головною метою стає прогрес здобувача [3].

Класифікація видів дистанційного контролю за часовим параметром та режимом взаємодії учасників освітнього процесу є ключовою для розуміння технологічних особливостей його організації. Розрізняють синхронний та асинхронний режими контролю, кожен з яких має свої переваги, недоліки та сферу застосування.

Синхронний контроль, що відбувається в режимі реального часу за допомогою засобів відеоконференцій, намагається максимально відтворити атмосферу та процедуру традиційного іспиту або заліку. Він дозволяє викладачу безпосередньо спостерігати за поведінкою та реакцією студента, ставити уточнюючі запитання та перевіряти глибину розуміння матеріалу в ході живої дискусії.

Проте синхронний режим має суттєві обмеження технічного та організаційного характеру: він вимагає стабільного високошвидкісного інтернет-з'єднання, наявності відповідного обладнання у всіх учасників, а також жорсткої прив'язки до розкладу, що може створювати нерівні умови для студентів, які перебувають у різних часових поясах або мають обмежений доступ до технологій.

Виклики, пов'язані з технічними збоями та нерівним доступом до технологій, є однією з головних перешкод для об'єктивного оцінювання в синхронному режимі, що змушує шукати альтернативні, більш надійні шляхи верифікації знань [4].

Асинхронний контроль, натомість, передбачає виконання контрольних завдань у зручний для студента час у межах визначеного викладачем терміну. Цей вид контролю є домінуючим у більшості сучасних систем управління навчанням та веб-орієнтованих платформ тестування, включаючи концепцію розробки системи «Testify».

Асинхронний підхід дозволяє автоматизувати перевірку знань через різноманітні тестові завдання, інтерактивні вправи та завантаження письмових робіт. Він забезпечує високу гнучкість навчального процесу, дозволяючи студентам ефективно поєднувати навчання з іншими видами діяльності та

працювати у власному темпі, що особливо актуально для дорослих здобувачів освіти та студентів заочної форми. Крім того, асинхронний режим знімає значну частину психологічного напруження, пов'язаного з необхідністю публічної відповіді, дозволяючи студенту зосередитися на змісті завдання.

Однак, незважаючи на технологічні переваги, асинхронний контроль несе в собі серйозні ризики, пов'язані з проблемою академічної доброчесності, оскільки відсутність безпосереднього нагляду створює спокусу для порушення правил проходження тестування [4].

Проблема академічної доброчесності в умовах дистанційного контролю набуває особливої гостроти та стає однією з центральних характеристик його сутності на сучасному етапі. Відсутність візуального контролю з боку викладача створює передумови для несанкціонованого використання допоміжних джерел, комунікації з іншими студентами під час тестування або навіть залучення сторонніх осіб для виконання завдань.

Це явище, відоме в міжнародній практиці як "contract cheating" [4], становить серйозну загрозу для репутації освітніх закладів та цінності виданих дипломів. Для протидії цим явищам розробники систем контролю впроваджують складні алгоритмічні рішення: обмеження часу на відповідь, випадкову генерацію варіантів із великого банку питань, блокування можливості копіювання тексту чи перемикання вікон браузера.

Однак суто технічні засоби не завжди є достатніми, тому питання формування культури академічної доброчесності залишається невід'ємною частиною організації дистанційного контролю.

Ситуація ускладнюється стрімким розвитком технологій штучного інтелекту, які здатні генерувати унікальні, логічно структуровані та граматично правильні тексти відповідей на відкриті запитання, розв'язувати математичні задачі та писати програмний код, що робить традиційні методи контролю, такі як есе, реферати чи домашні контрольні роботи, фактично беззахисними перед фальсифікаціями.

Це ставить перед системою освіти виклик необхідності трансформації самих підходів до формулювання завдань, зміщуючи акцент з репродуктивного відтворення інформації на перевірку творчого мислення, вміння аналізувати, синтезувати та оцінювати інформацію.

Сутність контролю зміщується в площину перевірки компетентностей, які важко зімітувати машині: критичного осмислення, особистісної рефлексії та роботи з нестандартними ситуаціями. Це вимагає інтеграції в платформи дистанційного навчання новітніх засобів верифікації авторства та використання прокторингових систем, що здійснюють автоматизований нагляд за процедурою іспиту [5].

Технологічна складова сутності дистанційного контролю полягає у необхідності створення надійної, безпечної та ергономічної програмної інфраструктури. Існуючі на ринку універсальні системи управління навчанням часто характеризуються надмірною складністю архітектури, перевантаженістю інтерфейсу та високою вартістю обслуговування, що створює бар'єри для їх ефективного впровадження в освітній процес.

Крім того, багато популярних платформ не мають достатньої гнучкості для реалізації специфічних методик оцінювання, притаманних різним дисциплінам. Це формує об'єктивний запит на розробку спеціалізованих, легковагових систем, які фокусуються саме на зручності створення тестів, генерації звітності та глибокій аналітиці успішності.

Проектована система «Testify» розглядається у цьому контексті як спроба створити оптимізований інструмент, що враховує недоліки існуючих платформ та специфічні потреби вітчизняних закладів освіти, забезпечуючи баланс між функціональністю та простотою використання.

Психологічний аспект дистанційного контролю також є важливою складовою його сутності, яку не можна ігнорувати при проектуванні освітніх систем. Зміна формату оцінювання суттєво впливає на емоційний стан студентів. З одного боку, комп'ютерне тестування в звичних домашніх умовах

може знижувати рівень стресу, пов'язаного з міжособистісною взаємодією, страхом перед авторитетом викладача чи публічним виступом.

З іншого боку, виникає специфічна "технологічна тривожність", зумовлена страхом технічного збою, втрати інтернет-з'єднання чи некоректного збереження відповідей. Крім того, відсутність емоційного контакту та емпатії з боку машини може створювати відчуття відчуженості та демотивації.

Тому проектування інтерфейсів систем контролю повинно базуватися на принципах людиноцентричності, забезпечуючи зрозумілу навігацію, чіткі інструкції та наявність механізмів технічної та методичної підтримки користувачів [4].

Економічна ефективність та оптимізація ресурсів є ще одним важливим виміром сутності автоматизованого контролю. Впровадження систем на кшталт «Testify» дозволяє суттєво скоротити часові витрати викладача на перевірку рутинних завдань, ведення журналів успішності та формування звітності. Автоматизація розрахунку балів та генерації відомостей мінімізує ризик механічних помилок та корупційних ризиків, роблячи процес оцінювання прозорим та підзвітним.

У довгостроковій перспективі це призводить до значної економії ресурсів навчального закладу та підвищення пропускної здатності системи оцінювання. Це дозволяє перерозподілити педагогічне навантаження, вивільняючи час викладачів для більш творчої методичної та наукової роботи, а також для індивідуального консультування студентів [1].

Таким чином, узагальнюючи викладене, можна стверджувати, що дистанційний контроль знань студентів є багатовимірним явищем, що поєднує в собі педагогічні, психологічні, технологічні, етичні та управлінські аспекти. Це не просто технічна трансляція завдань через мережу Інтернет, а нова парадигма оцінювання, яка вимагає переосмислення ролі викладача та студента в освітньому процесі.

Ефективна система дистанційного контролю повинна бути не лише інструментом верифікації знань, а й середовищем для розвитку, самовдосконалення та підтримки навчальної мотивації. Проектування системи «Testify» ґрунтується на глибокому розумінні цих складових, маючи на меті створення збалансованого, надійного та зручного інструменту, який відповідає сучасним вимогам до якості вищої освіти та забезпечує об'єктивність і прозорість контрольних заходів в умовах цифровізації суспільства.

Аналіз сутності дистанційного контролю доводить, що його подальший розвиток буде пов'язаний з впровадженням інтелектуальних алгоритмів, адаптивних технологій та комплексних систем безпеки, що дозволить зробити оцінювання ще більш точним, персоналізованим та справедливим [1].

## 1.2 Сучасний стан онлайн-тестування в освітньому процесі

Наукове обґрунтування процесів модернізації систем контролю знань у вищій школі на сучасному етапі базується на комплексному аналізі досягнень педагогічної теорії, прикладної математики, психометрики та інженерії програмного забезпечення.

Аналіз джерел та нормативно-технічної документації за період 2020–2024 років свідчить про те, що цифрова трансформація оцінювання перейшла від етапу простої комп'ютеризації паперових методик до створення складних інформаційних екосистем, що базуються на хмарних обчисленнях та просунутих статистичних методах обробки даних [6].

Сучасні дослідники наголошують, що ключовим трендом останніх років є відмова від лінійних, статичних форм контролю на користь динамічних моделей, які дозволяють враховувати індивідуальні особливості темпу навчання студентів та забезпечувати високу точність вимірювання компетентностей [6].

Розробка сучасних програмних комплексів для тестування вимагає глибокого розуміння теоретичних засад тестології, принципів захисту інформації та вимог до ергономіки інтерфейсів, що в сукупності визначає валідність та надійність отриманих результатів оцінювання [6].

Фундаментальною основою для побудови алгоритмів сучасних систем тестування залишається еволюція від класичної теорії тестів до сучасної теорії відповідей на завдання (Item Response Theory – IRT) [7].

В останні роки спостерігається інтенсивний розвиток математичного апарату IRT, зокрема розробка нових імовірнісних моделей, які дозволяють оцінювати не лише правильність відповіді, а й латентні параметри підготовленості студента з урахуванням складності завдання та його диференціюючої здатності [7].

На відміну від класичних підходів, де результати залежали від конкретного набору питань, використання параметричних моделей дозволяє створювати калібровані банки тестових завдань, де кожне питання має чітко визначену вагу та характеристичну криву. Це дає можливість порівнювати результати студентів, які виконували різні варіанти тестів, забезпечуючи математично обґрунтовану справедливість оцінювання.

Впровадження таких математичних моделей у програмне забезпечення дозволяє автоматизувати процес вирівнювання варіантів та підвищити точність вимірювання знань у великих вибірках респондентів, що є критичним для забезпечення якості освіти [7].

На основі теорії ймовірностей та математичної статистики активно розвивається методологія комп'ютерного адаптивного тестування. Принцип роботи таких алгоритмів полягає у динамічному виборі наступного завдання із банку питань залежно від результату відповіді на попереднє: у разі успіху система пропонує складніше питання, яке має вищу вагу, у разі невдачі – легше. Такий підхід дозволяє скоротити час тестування на 30–50% без втрати надійності, оскільки система швидко визначає рівень знань студента, не витрачаючи час на завдання, які є занадто легкими або занадто складними для

нього. Реалізація адаптивного тестування вимагає створення складних розгалужених сценаріїв та великих банків завдань, проте сучасні вебтехнології дозволяють ефективно вирішувати ці завдання, забезпечуючи миттєву обробку відповідей та перерахунок рейтингу студента в реальному часі без помітних затримок у роботі інтерфейсу [8].

Важливим вектором розвитку систем дистанційного контролю є забезпечення інтероперабельності – здатності різних освітніх платформ та баз даних взаємодіяти між собою без втрати інформації.

Необхідність використання уніфікованих форматів обміну даними (на основі XML або JSON), які дозволяють описувати структуру тестових завдань, логіку нарахування балів та метадані контенту розглядається в [9]. Це вирішує проблему прив'язки до конкретного розробника програмного забезпечення, дозволяючи навчальним закладам вільно експортувати та імпортувати банки питань між різними системами управління навчанням.

Сучасні підходи до архітектури освітніх систем передбачають модульність, де модуль тестування може бути інтегрований у будь-яке цифрове середовище університету через захищені протоколи передачі даних, забезпечуючи цілісність освітнього процесу та консолідацію звітності в єдиному центрі [9].

Особлива увага в сучасних дослідженнях приділяється питанням цифрової доступності та універсального дизайну інтерфейсів. Розробка програмного забезпечення для масового оцінювання повинна враховувати потреби користувачів з різними фізичними та когнітивними можливостями.

Це включає забезпечення контрастності візуальних елементів, можливість масштабування тексту, підтримку навігації за допомогою клавіатури та сумісність із допоміжними технологіями, такими як екранні зчитувачі.

Дослідження у сфері людино-машинної взаємодії показують, що ігнорування принципів інклюзивності може призводити до спотворення результатів тестування, коли низький бал зумовлений не відсутністю знань, а

неможливістю коректно сприйняти умову завдання через технічні бар'єри [10]. Тому проектування сучасних платформ передбачає створення гнучких, адаптивних інтерфейсів, які коректно відображаються на будь-яких пристроях, від настільних комп'ютерів до мобільних телефонів [10].

Паралельно з удосконаленням механізмів тестування відбувається розвиток методів збору та аналізу статистичних даних про процес навчання.

Сучасні архітектурні рішення дозволяють фіксувати найдрібніші деталі взаємодії студента з системою у вигляді структурованих логів подій: час, витрачений на кожне питання, послідовність переходів між сторінками, кількість змін відповіді перед остаточним підтвердженням.

Статистичний аналіз цих "сирих" даних методами кластеризації та регресійного аналізу дозволяє викладачам виявляти приховані закономірності, наприклад, діагностувати теми, які викликають масові труднощі, або ідентифікувати завдання з некоректними чи неоднозначними формулюваннями.

Глибока аналітика перетворює систему контролю з інструменту простої фіксації успішності на потужний діагностичний комплекс, що допомагає оптимізувати навчальний контент та методику викладання на основі об'єктивних даних [11].

Забезпечення академічної доброчесності в умовах дистанційного доступу залишається одним із найгостріших питань, яке вирішується комплексом організаційних та технічних заходів.

Сучасні підходи до вирішення цієї проблеми базуються на використанні спеціалізованих браузерних рішень, які обмежують функціонал операційної системи користувача під час іспиту, блокуючи можливість копіювання тексту, відкриття нових вкладок або запуск сторонніх програм.

Крім того, використовуються методи аналізу цифрового відбитка пристрою та мережевої активності для запобігання несанкціонованому доступу. Проблеми ефективності використання повної рандомізації – динамічного формування унікального варіанту тесту для кожного студента

шляхом випадкового вибору завдань із великого категоризованого банку, а також перемішування варіантів відповідей розглядаються в [12]. Комбінація цих технічних засобів дозволяє створити контрольоване середовище, яке мінімізує ризики маніпуляцій [12].

Хмарна архітектура сучасних систем тестування вимагає посиленої уваги до питань кібербезпеки та захисту персональних даних. Оскільки результати оцінювання є чутливою інформацією, розробка програмних платформ повинна відповідати суворим вимогам щодо конфіденційності.

Сучасні підходи передбачають використання наскрізного шифрування каналів передачі даних, регулярне створення розподілених резервних копій для запобігання втраті інформації та розмежування прав доступу на основі рольових моделей.

Дослідники також вказують на важливість забезпечення цілісності даних, щоб унеможливити будь-яке втручання в історію оцінювання або зміну балів "заднім числом" [12]. Забезпечення технічної надійності та стійкості до високих навантажень під час сесій є критичним фактором довіри до системи дистанційного контролю.

Важливим трендом, що знаходить відображення в наукових публікаціях, є впровадження мотиваційних механізмів та елементів ігрового дизайну в процес оцінювання.

Дослідження демонструють, що візуалізація прогресу, використання системи досягнень та миттєвий зворотний зв'язок дозволяють знизити рівень екзаменаційного стресу та підвищити залученість студентів [13]. Концепція формувального оцінювання передбачає, що контроль має бути інтегрований у процес навчання як інструмент самодіагностики, а не лише як фінальний вирок.

Такий підхід вимагає від розробників систем створення гнучких налаштувань, які дозволяють викладачам обирати різні режими тестування: від суворого екзаменаційного до тренувального з можливістю перегляду правильних відповідей та пояснень [13].

Методологія оцінювання також еволюціонує в бік використання комплексних форм контролю, таких як взаємне оцінювання. Сучасні платформи надають інструментарій для організації перехресної перевірки робіт студентами за чітко визначеними критеріями та рубриками. Це дозволяє не лише зняти частину навантаження з викладача при перевірці відкритих завдань, а й розвивати у студентів навички критичного мислення та аналізу.

Алгоритми таких систем автоматично розподіляють роботи між учасниками, забезпечують анонімність авторів та рецензентів, а також можуть статистично вираховувати узгодженість оцінок, відкидаючи аномальні результати. Інтеграція механізмів взаємного оцінювання у платформи тестування дозволяє значно розширити спектр типів завдань, включаючи туди есе, проекти та кейси[14].

Узагальнюючи огляд джерел та підходів, можна зробити висновок, що сучасна система дистанційного контролю знань – це складний програмно-методичний комплекс, що поєднує в собі досягнення математичної статистики, криптографії, веб-інженерії та педагогічного дизайну.

Основними векторами розвитку є підвищення точності вимірювань через імовірнісні алгоритми, забезпечення сумісності через уніфіковані протоколи обміну даними, гарантування безпеки та покращення користувацького досвіду через адаптивність та ергономіку.

Врахування цих аспектів при проектуванні нових програмних рішень дозволяє створити ефективні інструменти, що відповідають викликам цифрової епохи та забезпечують об'єктивність оцінювання навчальних досягнень студентів.

### 1.3. Сучасні системи дистанційного тестування знань студентів

Комплексний процес проектування архітектури та функціонального наповнення нової системи дистанційного контролю знань, що є об'єктом

даного дослідження, вимагає попереднього глибокого та всебічного аналізу існуючого ландшафту освітніх технологій.

Пошук та характеристика аналогів є критично важливим етапом, який дозволяє не лише виявити функціональні можливості конкурентних рішень, але й ідентифікувати їхні архітектурні обмеження, вразливості в підсистемах інформаційної безпеки та прогалини в методичному забезпеченні процесів оцінювання.

Сучасний глобальний ринок систем управління навчанням характеризується надзвичайно високим рівнем насиченості, фрагментації та конкуренції, пропонуючи споживачам широкий спектр інструментарію: від комплексних корпоративних платформ з надлишковим функціоналом до вузькоспеціалізованих хмарних сервісів, орієнтованих на мікронавчання або конкретні види тестування.

Існуючі системи дистанційного тестування знань студентів можна класифікувати за кількома ключовими критеріями: за типом ліцензування програмного забезпечення (відкриті вихідні коди проти пропрієтарних комерційних ліцензій), за моделлю розгортання інфраструктури (хмарні SaaS-рішення проти локальних on-premise інсталяцій) та за функціональним фокусом (універсальні системи управління навчальним процесом проти спеціалізованих платформ для комп'ютерного тестування).

Розуміння цієї класифікації та детальний розбір представників кожної групи є фундаментом для визначення ринкової ніші проєктованої системи та обґрунтування її унікальної ціннісної пропозиції для вітчизняних закладів вищої освіти, які стикаються зі специфічними викликами організації навчального процесу [15].

Безумовним лідером серед систем з відкритим вихідним кодом, який фактично став галузевим стандартом для академічного середовища в усьому світі, є платформа Moodle (*Modular Object-Oriented Dynamic Learning Environment*) [16].

Проводячи детальну характеристику цієї системи, дослідники та практики відзначають її виняткову гнучкість, адаптивність та масштабованість, що досягається завдяки глибоко продуманій модульній архітектурі ядра. Moodle дозволяє створювати складні, багаторівневі ієрархії навчальних курсів, керувати ролями користувачів з високим ступенем деталізації прав доступу (аж до окремих елементів курсу) та інтегрувати тисячі сторонніх плагінів, розроблених світовою спільнотою.

Особливо потужним є вбудований модуль тестування Moodle, який підтримує десятки типів питань, включаючи обчислювальні задачі зі змінними параметрами, завдання на перетягування об'єктів, роботу з інтерактивною графікою та вкладені відповіді. Система дозволяє налаштовувати складні сценарії поведінки тестів, такі як відкладений або негайний зворотний зв'язок, штрафні бали за неправильні спроби та адаптивний режим переходу між питаннями.

Проте, як свідчить аналіз досвіду масового впровадження цієї системи у період 2020–2023 років [16], її головними недоліками є надмірна складність інтерфейсу користувача, перевантаженість меню налаштувань та високий поріг входження для викладачів, які не мають глибоких технічних навичок. Адміністрування Moodle вимагає значних серверних ресурсів, оптимізації баз даних та наявності кваліфікованого інженерного персоналу для підтримки працездатності системи, встановлення оновлень безпеки та вирішення проблем сумісності плагінів.

Крім того, "важковагова" монолітна архітектура системи, написана на мові PHP, часто призводить до проблем із продуктивністю при високих пікових навантаженнях, наприклад, під час одночасного проведення підсумкових іспитів для великої кількості студентів, що є критичним фактором ризику для стабільності навчального процесу [16].

Альтернативою складним серверним рішенням, що вимагають власної інфраструктури, виступають хмарні сервіси від глобальних технологічних корпорацій, серед яких домінуюче становище в освітньому секторі займає

Google Classroom [17]. Ця платформа позиціонується розробниками не як повноцінна система управління навчанням у класичному розумінні, а як інструмент для організації ефективного документообігу, комунікації та колаборації в навчальному процесі.

Головною конкурентною перевагою Google Classroom є безшовна, нативна інтеграція з екосистемою хмарних офісних застосунків, віртуальним диском для зберігання файлів та сервісами відеоконференцій, що створює єдиний, інтуїтивно зрозумілий цифровий робочий простір.

Простота інтерфейсу, мінімалізм у налаштуваннях та відсутність необхідності в адмініструванні серверів зробили цю систему надзвичайно популярною під час екстреного переходу закладів освіти на дистанційне навчання. Однак, при детальному технічному аналізі функціоналу тестування виявляється його суттєва обмеженість для потреб вищої школи: система базується на простих формах опитування, які не підтримують складні логічні сценарії розгалуження, обмежені у типах питань (переважно вибір з варіантів та коротка текстова відповідь) та не надають інструментів для глибокої психометричної аналітики результатів.

Google Classroom не дозволяє створювати структуровані, калібровані банки питань з параметризацією складності та тематичними тегами, а механізми захисту від академічної недоброчесності є базовими і легко обходяться технічно грамотними студентами. Також серйозним питанням для багатьох університетів залишається політика обробки та зберігання персональних даних на закордонних серверах, яка часто вступає в конфлікт з вимогами національного законодавства щодо цифрового суверенітету та захисту інформації [17].

У сегменті професійних комерційних корпоративних рішень еталоном функціональності та надійності вважаються платформи Canvas (від компанії Instructure) та Blackboard Learn [18]. Ці системи пропонують винятковий рівень відмовостійкості, сучасний адаптивний дизайн інтерфейсу,

орієнтований на позитивний досвід користувача, та потужні вбудовані аналітичні інструменти для моніторингу успішності.

Canvas, зокрема, вирізняється своєю хмарною мікросервісною архітектурою, яка забезпечує автоматичне масштабування обчислювальних ресурсів залежно від поточного навантаження та дозволяє впроваджувати оновлення без зупинки сервісу [18].

Системи такого класу підтримують передові протоколи інтероперабельності, що дозволяє їм легко інтегруватися з університетськими інформаційними системами, реєстрами студентів та бібліотечними фондами через захищені канали передачі даних. Проте, головним бар'єром для масового впровадження цих висококласних аналогів у вітчизняних реаліях є їхня висока вартість володіння, яка включає значні щорічні ліцензійні платежі, плату за кожного активного користувача та витрати на сертифіковане навчання персоналу.

Крім того, пропрієтарна замкнена природа програмного коду робить практично неможливим внесення змін у бізнес-логіку роботи системи під специфічні потреби та регламенти конкретного навчального закладу, змушуючи університети адаптувати свої внутрішні процеси під логіку роботи програмного забезпечення, а не навпаки, що часто викликає опір академічної спільноти [18].

Окрему специфічну нішу на ринку займають платформи для гейміфікованого тестування та інтерактивного мікронавчання, такі як Kahoot!, Quizizz, Socrative або Mentimeter [13,19]. Ці сервіси фокусуються на залученні аудиторії та підвищенні емоційної залученості через використання ігрових механік, яскравої візуалізації, динамічних таймерів та змагальних елементів (рейтингові таблиці, бали за швидкість).

Вони ідеально підходять для проведення швидких бліц-опитувань, перевірки залишкових знань під час лекційних занять в режимі реального часу та підвищення мотивації студентів. Однак, як показує детальна

характеристика їхнього функціоналу, вони абсолютно непридатні для проведення серйозних підсумкових іспитів, заліків або державної атестації.

Жорсткі обмеження на довжину тексту запитань та відповідей, відсутність інструментів для введення математичних формул чи програмного коду, неможливість надійної ідентифікації особи студента та слабкий захист від технічних маніпуляцій роблять їх виключно допоміжним дидактичним інструментом.

Крім того, бізнес-модель цих сервісів часто передбачає суттєві обмеження функціоналу у безкоштовних версіях або агресивну монетизацію через показ реклами, що є неприйнятним для серйозного академічного середовища [19].

Важливим аспектом порівняльної характеристики аналогів є аналіз їхніх можливостей у сфері забезпечення академічної доброчесності та контролю за процедурою тестування. Більшість універсальних систем, включаючи Moodle та Canvas, покладаються на інтеграцію із зовнішніми спеціалізованими прокторинговими сервісами.

Ці сервіси забезпечують відеоспостереження через вебкамеру, запис екрану робочого столу, блокування функцій браузера та автоматичний аналіз поведінки студента на предмет підозрілої активності. Проте інтеграція таких зовнішніх рішень значно здорожчує вартість експлуатації системи тестування та створює додаткове суттєве навантаження на канали зв'язку та обладнання користувачів.

Крім того, існує проблема частих "хибних спрацьовувань", коли система блокує студента через технічні збої, обрив з'єднання або особливості освітлення приміщення. Це формує об'єктивний запит на розробку систем, які б мали вбудовані, легковагові механізми захисту, що не вимагають встановлення додаткового програмного забезпечення на комп'ютер користувача та базуються на аналізі цифрового сліду, мережесих ідентифікаторів та поведінкових патернів роботи з інтерфейсом, а не на

тотальному відеоспостереженні, яке часто сприймається як порушення приватності.

Аналіз архітектурних підходів до побудови розглянутих аналогів виявляє чітку тенденцію до технологічного переходу від монолітних систем до розподіленої мікросервісної архітектури. Традиційні платформи, розроблені 10-15 років тому, часто страждають від проблеми тісної зв'язаності компонентів, коли оновлення модуля тестування може порушити роботу журналу оцінок, форуму або системи авторизації.

Сучасні розробки намагаються розділити функціонал на незалежні, ізольовані сервіси, що взаємодіють через документовані програмні інтерфейси передачі даних. Проте більшість існуючих на ринку рішень, що реально використовуються в університетах, все ще базуються на застарілих технологічних стеках, що ускладнює їх технічну підтримку, масштабування та розвиток. Проектована система має врахувати цей негативний досвід та базуватися на сучасних фреймворках та мовах програмування, що забезпечують високу швидкість розробки, надійність, безпеку коду та ефективне використання обчислювальних ресурсів.

Окремої детальної уваги заслуговує питання аналітики та звітності в існуючих системах дистанційного навчання. Більшість розглянутих аналогів надають лише базову описову статистику: середній бал по групі, розподіл оцінок за шкалою, історію спроб проходження тесту. Однак викладачам-методистам та адміністрації навчальних закладів часто критично не вистачає глибокої психометричної аналітики, яка б дозволяла оцінити якість самих тестових завдань та валідність тесту в цілому.

У комерційних системах така розширена функціональність часто винесена в окремі дорогі модулі, а в безкоштовних рішеннях вона відсутня або реалізована на примітивному рівні. Це створює "аналітичний вакуум", коли величезні масиви даних про навчальну активність збираються, але ефективно не обробляються і не використовуються для покращення якості навчального контенту та оптимізації освітніх програм.

Характеристика аналогів підтверджує наявність незаповненої ніші для системи, яка б робила акцент саме на освітній аналітиці даних як на базовому, вбудованому інструменті, доступному кожному викладачу.

Також варто розглянути аспект мобільного навчання та кросплатформності. Сучасні студенти дедалі частіше використовують смартфони та планшети як основні пристрої для доступу до освітніх матеріалів та проходження поточного контролю.

Аналіз офіційних мобільних застосунків Moodle, Google Classroom та Canvas показує, що вони часто мають обмежений функціонал порівняно з повноцінними вебверсіями, особливо в частині проходження складних інтерактивних тестів.

Часто інтерфейси не адаптовані під сенсорне керування на екранах малої діагоналі, що призводить до помилкових натискань та розчарування користувачів. Проектована система повинна вирішувати цю проблему через використання концепції адаптивного вебдизайну або технології прогресивних вебзастосунків, які забезпечують однаковий, повноцінний користувацький досвід незалежно від типу пристрою та операційної системи, без необхідності встановлення окремих застосунків з магазинів.

Суттєвим недоліком багатьох зарубіжних аналогів є слабка локалізація та невідповідність національним стандартам діловодства та організації навчального процесу. Системи оцінювання в західних університетах часто відрізняються від вітчизняної практики, що вимагає складних, нетривіальних налаштувань формул розрахунку підсумкового рейтингового балу.

Крім того, звітна документація (екзаменаційні відомості, журнали обліку відвідування та успішності), яка автоматично генерується цими системами, часто не відповідає офіційно затвердженим формам, змушуючи викладачів та співробітників деканатів виконувати подвійну роботу, вручну переносячи дані з електронної системи в паперові або електронні документи потрібного формату.

Це підкреслює необхідність створення спеціалізованого вітчизняного продукту, який би від початку враховував специфіку нормативної бази, термінології та бюрократичних процедур української вищої школи, максимально автоматизуючи рутинну роботу з документами.

Окремого детального аналізу в контексті характеристики аналогів потребує ергономіка інтерфейсів з точки зору викладача-практика. Більшість розглянутих систем, як комерційних, так і вільно розповсюджуваних, історично проектувалися з фокусом на досвід студента (*Student-Centered Design*), залишаючи зручність роботи педагога на другому плані.

У таких платформах як Moodle процес створення нового тестового завдання або перевірки відкритої відповіді часто вимагає надмірної кількості кліків, переходів між вікнами та заповнення неочевидних полів налаштувань.

Це явище, відоме як "інтерфейсне тертя", призводить до суттєвого збільшення часових витрат на рутинні операції, що в умовах високого педагогічного навантаження стає критичним фактором неприйняття системи колективом. Натомість сучасні вимоги до програмного забезпечення диктують необхідність впровадження принципів оптимізованого робочого процесу (*Workflow Optimization*), де створення тесту має відбуватися в єдиному вікні за принципом "що бачиш, те й отримуєш", а перевірка робіт – у зручному інтерфейсі, що нагадує перегортання сторінок, без необхідності постійного перезавантаження вебсторінки.

Також важливим критерієм порівняння є економічна модель впровадження та сукупна вартість володіння (*Total Cost of Ownership – TCO*). Часто заклади освіти помилково вважають системи з відкритим кодом (*Open Source*) повністю безкоштовними.

Однак детальний економічний розрахунок показує, що витрати на оренду серверних потужностей, забезпечення безперебійного електроживлення, оплату праці системних адміністраторів та фахівців з кібербезпеки можуть зрівнятися або навіть перевищити вартість ліцензій комерційних хмарних рішень. Водночас, комерційні платформи часто

використовують моделі ціноутворення, прив'язані до валютних курсів, що створює бюджетні ризики для вітчизняних університетів. Це формує запит на розробку національного програмного продукту з прозорою та прогнозованою економічною моделлю, яка б враховувала реальні фінансові можливості українського освітнього сектору та дозволяла оптимізувати витрати за рахунок ефективного використання хмарних ресурсів лише у періоди пікових навантажень (наприклад, під час сесії) [20].

Узагальнюючи проведений пошук та характеристику аналогів, можна зробити висновок, що на поточному ринку освітніх технологій існує певна поляризація рішень. З одного боку знаходяться потужні, але надзвичайно складні, дорогі та ресурсоємні платформи корпоративного рівня, які вимагають значних інвестицій в інфраструктуру та персонал. З іншого боку – прості, зручні, але функціонально обмежені та методично слабкі сервіси, які не забезпечують необхідної глибини контролю, безпеки та аналітики.

Спостерігається очевидний дефіцит рішень "середнього рівня" – спеціалізованих систем дистанційного контролю, які б гармонійно поєднували професійний інструментарій тестування, сучасну глибоку аналітику, високий рівень інформаційної безпеки та простоту використання, не будучи при цьому перевантаженими зайвим функціоналом управління контентом курсів.

Саме на заповнення цієї технологічної та методичної прогалини спрямована розробка програмного комплексу, що є предметом даного дослідження. Воно покликане інтегрувати найкращі архітектурні та інтерфейсні практики існуючих рішень, нівелюючи їхні виявлені недоліки та пропонуючи оптимізований, адаптований під локальні потреби підхід до організації контролю знань.

## 2 АНАЛІЗ ТА ШЛЯХИ РОЗВ'ЯЗКУ ПРОБЛЕМИ

### 2.1 Алгоритми обчислення результатів тестування

Проектування та імплементація підсистеми автоматизованого контролю знань є одним із найбільш відповідальних етапів розробки веб-орієнтованого навчального середовища. В умовах дистанційного навчання, коли безпосередній контакт між викладачем та студентом є обмеженим, саме програмні алгоритми беруть на себе функцію арбітра, що визначає рівень засвоєння матеріалу.

Алгоритмічна база модуля тестування повинна відповідати суворим вимогам щодо детермінованості, об'єктивності. Основна мета розробленого алгоритму полягає в забезпеченні точного перетворення вхідного масиву даних, отриманого від користувача, у кількісний показник успішності, який підлягає подальшому зберіганню та аналізу. Важливою умовою є забезпечення повної автономності процесу перевірки, що виключає необхідність ручного втручання з боку адміністратора або викладача.

В основу розробленого програмного рішення покладено лінійний алгоритм порівняння дискретних величин. Оскільки система проектується для роботи з тестовими завданнями закритого типу, де множина правильних відповідей є заздалегідь визначеною та незмінною в межах однієї версії тесту, задача оцінювання зводиться до процедури верифікації вибору користувача.

Технічна реалізація даного підходу базується на серверній логіці обробки даних, що є критично важливим для забезпечення академічної доброчесності. Перенесення алгоритмів перевірки на бік клієнта є неприпустимим в сучасних вебдодатках, оскільки це створює критичні вразливості системи безпеки. Тому розроблений алгоритм функціонує в межах захищеного серверного середовища, використовуючи об'єктно-орієнтований підхід до маніпуляції даними [21].

Для формалізації задачі оцінювання доцільно використати математичний апарат теорії множин. Нехай навчальний тест  $T$  складається з впорядкованої послідовності питань, кількість яких позначимо як  $N$ . Кожному питанню  $q_i$  (де  $i$  змінюється від 1 до  $N$ ) у базі даних системи відповідає унікальний ідентифікатор правильної відповіді – ключ  $k_i$ . Процес проходження тесту студентом моделюється як формування вектора відповідей  $A = \{a_1, a_2, \dots, a_N\}$ , де кожен елемент  $a_i$  є ідентифікатором варіанту, обраного користувачем.

Алгоритм розрахунку підсумкового балу  $S$  базується на моделі рівновагової акумуляції. Це означає, що кожне тестове завдання має однакову вагу, а загальний результат є сумою успішних ітерацій перевірки.

Для математичного опису цього процесу введемо бінарну функцію індикатора істинності  $I(a_i, k_i)$ , яка повертає одиницю у випадку збігу відповіді з ключем, та нуль у протилежному випадку.

Формула розрахунку результату виглядає наступним чином:

$$S = \sum_{i=1}^N I(a_i, k_i), \quad \text{де } I(a_i, k_i) = \begin{cases} 1 & \text{якщо } a_i = k_i \\ 0 & \text{якщо } a_i \neq k_i \vee \emptyset \end{cases}$$

Блок-схема алгоритму приведена на рис. 2.1

Ядром алгоритму є ітераційний процес порівняння (Core Verification Loop). Програма виконує прохід по списку питань тесту. Для кожного питання алгоритм здійснює пошук відповіді, наданої користувачем.

Тут реалізовано захисний механізм обробки відсутніх даних: якщо студент не надав відповідь на певне питання (наприклад, пропустив його), система не генерує виняткову ситуацію, а коректно інтерпретує це як відсутність відповіді (*None*), що автоматично призводить до нарахування 0 балів за даний пункт.

Сама процедура порівняння є суворою.

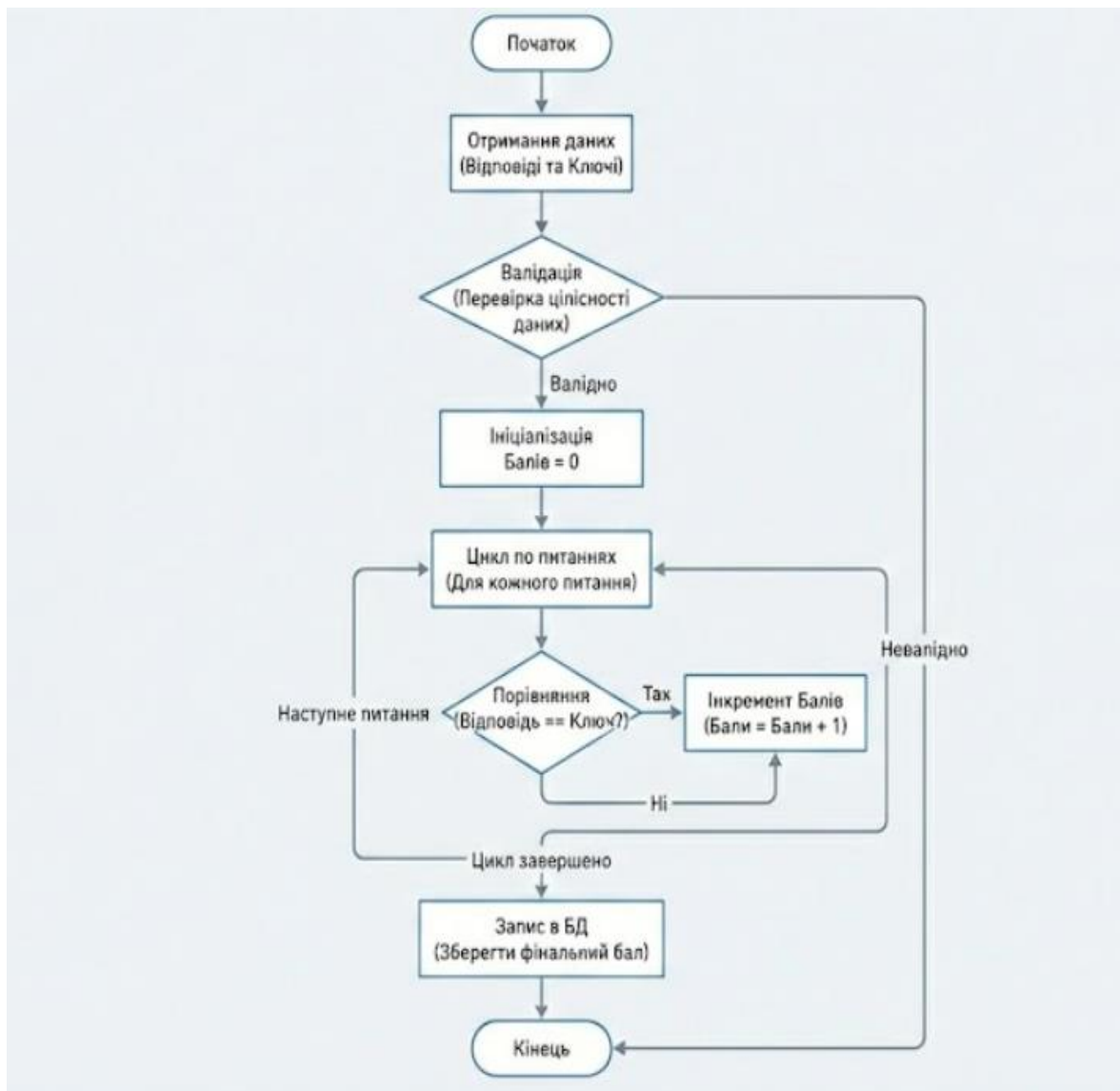


Рисунок 2.1 – Блок-схема алгоритму

Збіг ідентифікаторів повинен бути повним. Використання цілочисельних ідентифікаторів замість рядків тексту має низку переваг.

По-перше, це виключає помилки, пов'язані з кодуванням (наприклад, різниця між кириличною та латинською літерою "i").

По-друге, цілочисельні операції процесором виконуються значно швидше, ніж лексикографічне порівняння рядків.

Результатом роботи циклу є накопичена сума балів – абсолютний показник успішності.

Окремо слід розглянути питання обробки конкурентних запитів (Race Condition). Хоча ймовірність того, що один студент спробує здати один і той самий тест двічі одночасно (наприклад, з двох різних вкладок браузера), є низькою, алгоритм передбачає блокування запису для поточної сесії на момент початку обробки.

Це гарантує, що в базу даних потрапить лише один, перший успішно оброблений результат, що запобігає дублюванню даних та спотворенню статистики [22].

Підсумовуючи, можна стверджувати, що представлений алгоритм обчислення результатів тестування є збалансованим рішенням, яке поєднує математичну точність, програмну ефективність та високий рівень безпеки даних.

Використання серверної логіки обробки, транзакційних записів до бази даних (БД) та оптимізованих структур даних дозволяє створити надійний інструмент контролю знань, який повністю відповідає вимогам сучасних освітніх стандартів та забезпечує об'єктивність оцінювання навчальних досягнень студентів.

## 2.2 Структурна та функціональна моделі системи дистанційного контролю знань студентів

Розробка комплексних програмних систем освітнього призначення вимагає ретельного попереднього моделювання, що дозволяє формалізувати вимоги до системи, визначити її межі та спроектувати внутрішню архітектуру перед початком етапу кодування.

На етапі системного аналізу критично важливим є застосування підходів структурного та функціонального моделювання, які дозволяють розглядати проєктований об'єкт з різних точок зору. Структурна модель фокусується на статичній організації системи, визначаючи її компоненти та зв'язки між ними,

тоді як функціональна модель описує динамічні аспекти поведінки системи, процеси обробки інформації та взаємодію з зовнішніми користувачами.

Поєднання цих двох підходів дозволяє створити цілісне уявлення про проєктовану систему дистанційного контролю знань та гарантувати відповідність кінцевого продукту поставленим педагогічним та технічним завданням.

Функціональна модель системи дистанційного контролю знань призначена для визначення того, які завдання повинна виконувати система для задоволення потреб основних груп користувачів. Вона абстрагується від технічних деталей реалізації (таких як мова програмування чи тип БД) і зосереджується на бізнес-процесах та інформаційних потоках.

Основним інструментом для опису функціональності на високому рівні абстракції є діаграми прецедентів (*Use Case diagrams*) мови UML, які дозволяють ідентифікувати дійових осіб (акторів) та їхні цілі при взаємодії з системою [23].

У проєктованій системі виділено два основних типи акторів, що мають принципово різні права доступу та сценарії використання: «Викладач» та «Студент». Окремо може розглядатися актор «Адміністратор», але в контексті базової функціональності його роль часто зводиться до технічної підтримки, тому основна увага приділяється освітнім ролям.

Актор «Викладач» є ініціатором навчального процесу та генератором контенту. Його основні функціональні обов'язки включають створення навчальних курсів, управління списками студентів, розробку контрольних заходів (тестів, практичних завдань) та оцінювання результатів їх виконання. Актор «Студент» є споживачем освітнього контенту.

Його взаємодія з системою полягає у приєднанні до курсів, перегляді навчальних матеріалів, проходженні тестування та завантаженні файлів з виконаними завданнями, а також перегляді отриманих оцінок та коментарів викладача.

Ключовим аспектом функціональної моделі є розмежування зон відповідальності між акторами. Система повинна забезпечувати суворий контроль доступу, де викладач має повні права (CRUD – Create, Read, Update, Delete) на контент в межах своїх курсів, тоді як студент має переважно права на читання матеріалів та створення власних відповідей (Submissions).

Центральним функціональним блоком є підсистема тестування, яка повинна підтримувати автоматизовану перевірку для закритих типів питань та надавати інтерфейс для ручної перевірки відкритих питань викладачем. Важливою функцією є також забезпечення зворотного зв'язку, що дозволяє студенту бачити не лише фінальний бал, але й, за налаштуванням викладача, аналіз допущених помилок (рис. 2.2) [24].



Рисунок 2.2 – Діаграма прецедентів

Деталізацію процесів обробки інформації в системі доцільно представити за допомогою діаграми потоків даних (*Data Flow Diagram, DFD*). Ця модель дозволяє відстежити траєкторію руху інформаційних об'єктів від

моменту їх введення користувачем до моменту збереження в базі даних або виведення у вигляді звітів. На рівні контекстної діаграми система розглядається як єдиний процес «Система дистанційного контролю знань», що взаємодіє із зовнішніми сутностями (Викладач, Студент).

При декомпозиції системи на діаграмі DFD першого рівня виділяються основні функціональні процеси: «Автентифікація користувачів», «Управління курсами», «Проведення тестування» та «Оцінювання та звітність». Процес «Автентифікація» приймає облікові дані від зовнішніх сутностей та звіряє їх зі сховищем даних «Користувачі».

У разі успіху генерується токен доступу, який дозволяє переходити до інших процесів. Процес «Управління курсами» отримує навчальні матеріали та параметри завдань від Викладача і зберігає їх у сховищах «Матеріали курсів» та «Завдання».

Найбільш складним з точки зору інформаційних потоків є процес «Проведення тестування». Він ініціюється запитом Студента на початок тесту. Процес витягує необхідні питання зі сховища «Завдання», формує варіант тесту та передає його Студенту.

Відповіді Студента, у свою чергу, надходять назад до процесу, який може або направити їх на автоматичну перевірку (для тестів), або зберегти у сховище «Відповіді студентів» для подальшої ручної перевірки.

Фінальний процес «Оцінювання» консолідує результати автоматичної та ручної перевірки, записує підсумкові бали у сховище «Журнал успішності» та генерує інформаційний потік з результатами, який направляється Студенту.

Такий підхід до моделювання дозволяє чітко виявити вузькі місця в обробці даних та спроектувати оптимізовані алгоритми взаємодії між підсистемами (рис. 2.3).

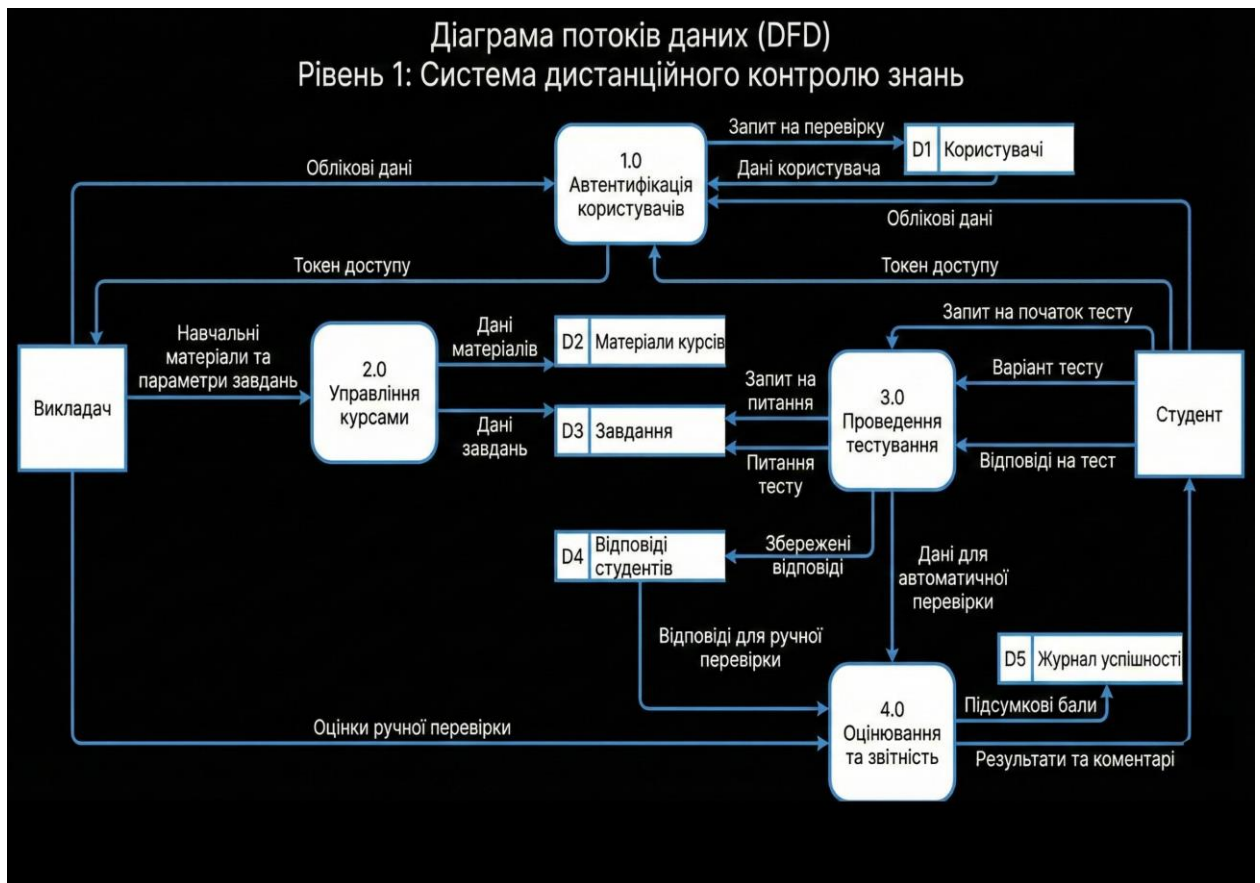


Рисунок 2.3 – Діаграма потоків даних DFD

Структурне моделювання визначає архітектуру програмного комплексу, тобто його організацію у вигляді сукупності взаємопов'язаних компонентів. Для проектованої системи дистанційного контролю знань обрано класичну трирівневу архітектуру (*Three-Tier Architecture*), яка є стандартом де-факто для сучасних вебзастосунків.

Ця архітектура передбачає логічне розділення системи на три шари: шар представлення (*Presentation Layer*), шар бізнес-логіки (*Business Logic Layer / Application Layer*) та шар даних (*Data Access Layer*).

Шар представлення відповідає за взаємодію з користувачем. Він реалізується у вигляді клієнтського веб-інтерфейсу (Frontend), який відображається у браузері. Його основна функція – прийом команд від користувача, первинна валідація ввідних даних та візуалізація інформації, отриманої від сервера. Цей шар повинен бути адаптивним, забезпечуючи

коректне відображення як на стаціонарних комп'ютерах, так і на мобільних пристроях [25].

Шар бізнес-логіки є ядром системи, що реалізується на стороні сервера (Backend). Саме тут зосереджені всі алгоритми обробки даних, правила доступу, механізми оцінювання тестів та логіка взаємодії між компонентами. Структурно цей шар доцільно представити у вигляді діаграми компонентів, яка демонструє модульну організацію сервера (рис. 2.4).

Основними модулями є:

1. Модуль безпеки та автентифікації: відповідає за реєстрацію користувачів, перевірку паролів, генерацію сесій та розмежування прав доступу до ресурсів на основі ролей.

2. Модуль управління курсами (CMS): забезпечує CRUD-операції для навчальних матеріалів, структурування контенту по темах, управління підписками студентів на курси.

3. Модуль тестування (Assessment Engine): центральний компонент, що реалізує логіку створення тестів, їх проведення (з урахуванням таймінгів та спроб) та автоматизованого оцінювання згідно з алгоритмами, описаними в попередніх розділах.

4. Модуль журналу та аналітики: агрегує дані про успішність, формує зведені відомості для викладачів та індивідуальні звіти для студентів.

Фундаментом структурної моделі будь-якої інформаційної системи є модель даних. Коректно спроектована схема бази даних забезпечує цілісність інформації, відсутність надлишковості та високу швидкість виконання запитів.

Для опису концептуальної схеми даних використовується діаграма «сутність-зв'язок» (ER-діаграма). Вона визначає основні інформаційні об'єкти предметної області та відношення між ними.

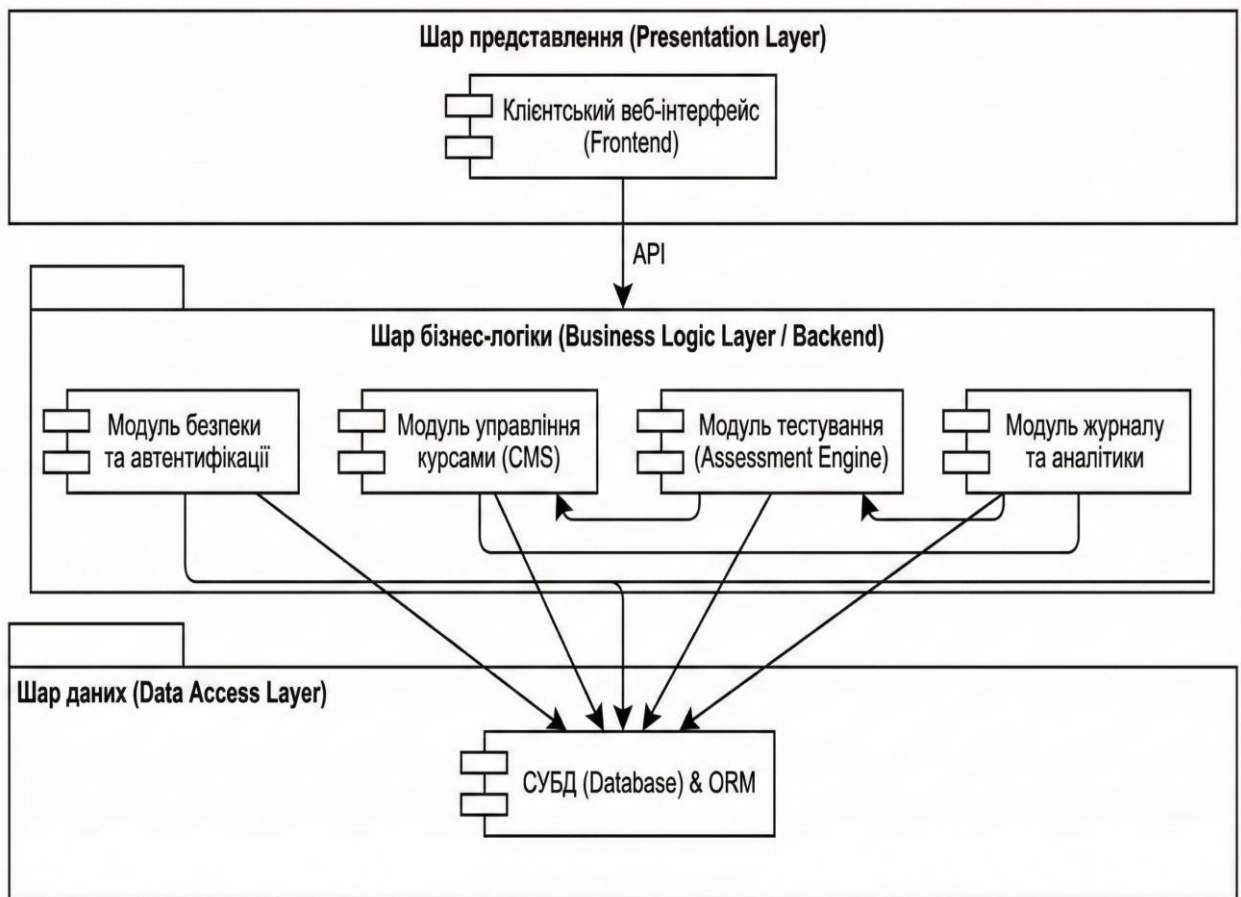


Рисунок 2.4 – Діаграма компонентів

Центральною сутністю системи є абстрактна сутність «Користувач» (User), яка містить базові атрибути: ідентифікатор, ім'я, електронну пошту та хеш пароля. Від неї можуть наслідуватися або бути пов'язані з нею через рольову модель конкретні типи користувачів – «Студент» та «Викладач». Іншою ключовою сутністю є «Курс» (Course), що містить метадані про навчальну дисципліну (назва, опис, автор).

Між сутностями «Користувач» (в ролі студента) та «Курс» існує зв'язок «багато-до-багатьох» (M:N), оскільки один студент може вивчати багато курсів, а на одному курсі навчається багато студентів. Цей зв'язок реалізується через проміжну сутність «Підписка» (Enrollment), яка також може зберігати дату приєднання та поточний статус студента на курсі.

Представлені функціональна та структурна моделі створюють цілісний проектний базис системи дистанційного контролю знань. Функціональне моделювання за допомогою діаграм прецедентів та потоків даних дозволило

чітко окреслити вимоги користувачів та логіку руху інформації. Структурне моделювання через трирівневу архітектуру, діаграму компонентів та ER-діаграму визначило надійний технічний фундамент для реалізації цих вимог.

Такий комплексний підхід до проектування гарантує створення масштабованої, безпечної та зручної у використанні системи, що відповідає сучасним стандартам розробки освітнього програмного забезпечення.

### 2.3 Стек технологій розробки системи дистанційного контролю знань студентів

Проектування архітектури сучасних програмних комплексів освітнього призначення вимагає виваженого підходу до вибору інструментальних засобів реалізації, оскільки цей вибір закладає фундамент для подальшої життєздатності, безпеки та масштабованості системи. У контексті розробки системи дистанційного контролю знань «Testify», ключовими вимогами до технологічного стеку були визначені: висока швидкість розробки, надійність обробки транзакційних даних, можливість горизонтального масштабування та кросплатформність.

На основі порівняльного аналізу існуючих рішень було сформовано стек технологій, що базується на використанні мови програмування Python, вебфреймворку Django, системи управління базами даних MySQL та клієнтських вебтехнологій HTML5/CSS3 з використанням бібліотеки Bootstrap. Інструментальний супровід розробки забезпечується інтегрованим середовищем PyCharm та CASE-засобом MySQL Workbench.

У якості базової мови програмування для серверної частини (back-end) системи обрано Python. Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня із суворою динамічною типізацією. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також

як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду [26].

Вибір Python обумовлений низкою факторів, які є критичними для освітніх систем. По-перше, синтаксис мови, орієнтований на читабельність коду, дозволяє мінімізувати кількість помилок на етапі розробки та суттєво спрощує подальшу підтримку та масштабування проєкту. По-друге, Python має одну з найпотужніших стандартних бібліотек, що дозволяє вирішувати типові задачі (робота з датою та часом, обробка текстових даних, мережева взаємодія) без залучення сторонніх залежностей.

Важливою перевагою Python у контексті розробки системи тестування є його ефективність у роботі з даними. Оскільки система передбачає обробку великої кількості результатів тестування та їх статистичний аналіз, наявність в екосистемі Python потужних математичних інструментів є суттєвим плюсом.

Крім того, кросплатформність інтерпретатора Python дозволяє розгорнути серверну частину системи на будь-якій операційній системі (Linux, Windows, macOS), що забезпечує гнучкість у виборі хостингу. Механізм автоматичного керування пам'яттю (Garbage Collection) звільняє розробника від необхідності ручного виділення та звільнення ресурсів, що підвищує стабільність роботи серверного застосунку під навантаженням.

Реалізація вебзастосунку здійснюється на базі фреймворку Django, який є де-факто стандартом для розробки складних вебсистем на Python. Django - безкоштовний і відкритий фреймворк для створення веб застосунків, написаний мовою програмування Python. Веб фреймворк - набір компонентів, які допомагають розробляти вебсайти швидше і простіше.

При побудові вебсайту, ви завжди потребуєте схожого набору компонентів: спосіб управління авторизацією користувача (створення акаунту, вхід, вихід), панель управління, форми, спосіб завантажити файли, тощо [27].

Архітектура Django базується на патерні MTV (Model-Template-View), який забезпечує чітке розділення логіки роботи з даними, логіки обробки

запитів та логіки відображення інтерфейсу. Такий підхід, відомий як Separation of Concerns (розділення відповідальності), дозволяє незалежно змінювати різні частини системи. Наприклад, зміна дизайну сторінки тестування (Template) не вимагає втручання в алгоритм нарахування балів (View) або структуру бази даних (Model) (рис. 2.5).

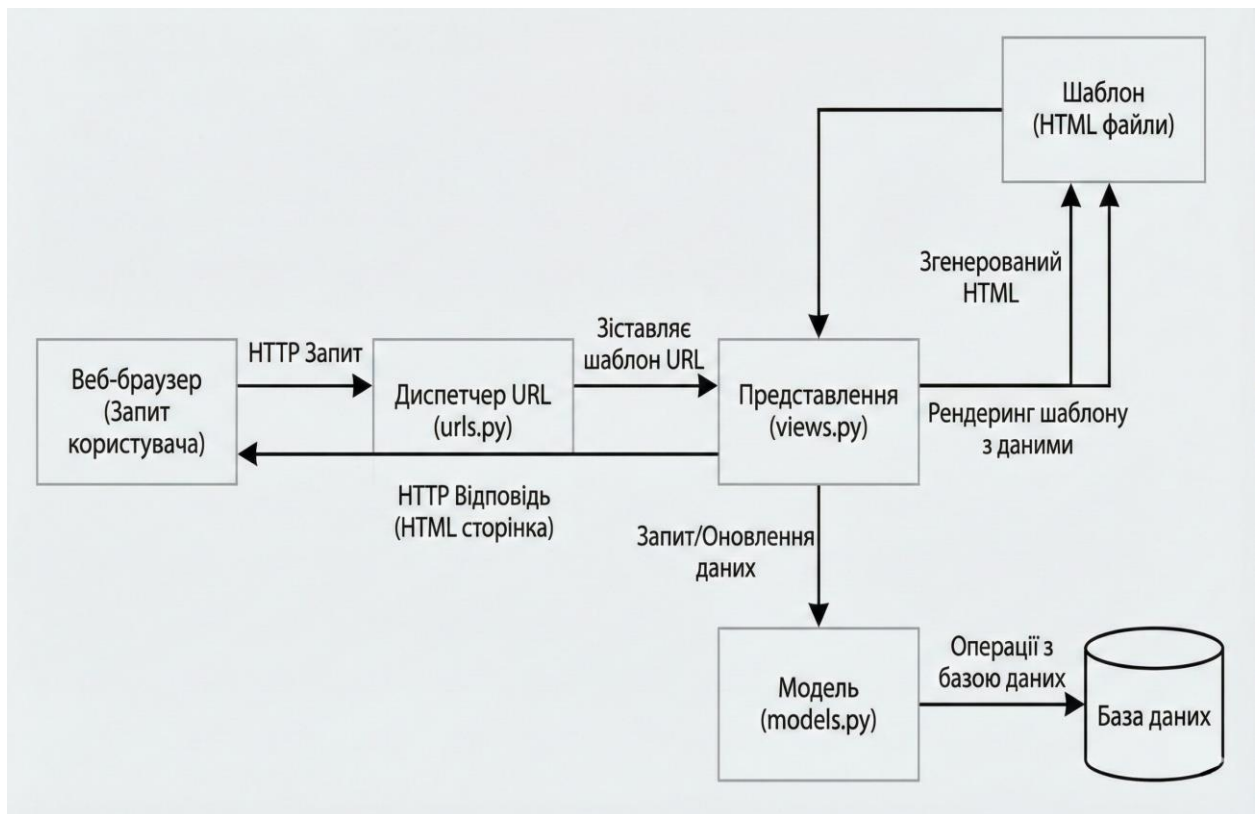


Рисунок 2.5 – Архітектурний патерн MTV у фреймворку Django

Центральним елементом Django, який значно спрощує розробку системи тестування, є Object-Relational Mapping (ORM). Ця технологія дозволяє працювати з таблицями бази даних як з класами Python. Замість написання складних SQL-запитів вручну, розробник оперує об'єктами. Наприклад, щоб знайти всі правильні відповіді студента, достатньо використати конструкцію *Answer.objects.filter(is\_correct=True)*, яку ORM автоматично трансліює в оптимізований SQL-запит для конкретної СУБД.

Це не лише прискорює розробку, а й забезпечує переносимість коду: систему можна легко переключити з MySQL на PostgreSQL або Oracle,

змінивши лише один рядок у конфігураційному файлі. Крім того, Django ORM включає механізм міграцій – систему контролю версій для схеми бази даних. Це дозволяє вносити зміни в структуру таблиць (наприклад, додати нове поле «Час проходження») без втрати вже накопичених даних, що є критично важливим на етапі еволюційного розвитку програмного продукту [28].

Окремої уваги заслуговує система безпеки Django. Фреймворк містить набір компонентів проміжного шару (Middleware), які автоматично фільтрують вхідні запити. Система захищена від атак типу Cross-Site Request Forgery (CSRF) шляхом використання криптографічних токенів у кожній формі введення даних.

Захист від SQL-ін'єкцій реалізовано на рівні ORM через параметризацію запитів. Також Django надає готову підсистему автентифікації користувачів, яка використовує надійні алгоритми хешування паролів (PBKDF2 за замовчуванням), що робить неможливим відновлення паролів навіть у випадку викрадення бази даних. Для освітньої системи, де зберігаються персональні дані студентів, такий рівень вбудованого захисту є обов'язковим [29].

Для персистентного зберігання даних обрано систему управління базами даних MySQL. Це реляційна СУБД, що базується на мові структурованих запитів SQL. Вибір реляційної моделі обумовлений суворою структурованістю даних у предметній області тестування.

Сутності «Студент», «Група», «Дисципліна», «Тест», «Питання» мають чіткі взаємозв'язки, які вимагають забезпечення посилальної цілісності (Referential Integrity). MySQL дозволяє реалізувати ці обмеження на рівні схеми бази даних за допомогою механізму зовнішніх ключів (Foreign Keys). Це гарантує, наприклад, що неможливо видалити запис про студента, поки в базі існують його результати тестування, або неможливо створити питання, прив'язане до неіснуючого тесту.

Ключовою перевагою MySQL є використання транзакційного рушія зберігання InnoDB. Транзакційність (відповідність принципам ACID) є критичною для системи оцінювання. Розглянемо сценарій завершення тесту:

система повинна одночасно зберегти відповіді, розрахувати бал, оновити статус спроби та записати час завершення.

Якщо на будь-якому з цих етапів станеться збій (наприклад, помилка мережі), всі зміни мають бути скасовані, щоб база даних не залишилася у неузгодженому стані. InnoDB забезпечує таку поведінку, гарантуючи надійність даних. Крім того, MySQL демонструє високу продуктивність при виконанні складних вибірок даних завдяки використанню B-Tree індексів, що дозволяє швидко формувати звіти успішності навіть при наявності мільйонів записів у таблиці відповідей [30].

Процес розробки структури бази даних здійснюється за допомогою візуального середовища MySQL Workbench. MySQL Workbench – інструмент для візуального проектування баз даних, що інтегрує проектування, моделювання, створення й експлуатацію БД в єдине безкоштовне оточення для системи баз даних MySQL. Є наступником DBDesigner 4 з FabForce [31].

Цей інструмент дозволяє застосовувати підхід EER-моделювання (Enhanced Entity-Relationship), створюючи графічні діаграми сутностей та зв'язків. Використання Workbench дозволяє візуалізувати складну структуру бази даних, контролювати типи даних (наприклад, використання *DECIMAL* для точного зберігання балів замість *FLOAT*), налаштовувати індекси та обмеження.

Функція прямого інжинірингу (Forward Engineering) автоматично генерує SQL-скрипти для створення фізичної бази даних на основі діаграми, що виключає синтаксичні помилки.

Також інструмент надає засоби для моніторингу продуктивності сервера («Dashboard»), що дозволяє виявляти «вузькі місця» та оптимізувати повільні SQL-запити шляхом аналізу планів їх виконання (Query Execution Plan).

Інтерфейс користувача (Front-end) реалізовано з використанням стандартизованого стеку вебтехнологій HTML5 та CSS3, посиленого фреймворком Bootstrap. HTML5 використовується для розмітки семантичної структури документів. Важливою особливістю HTML5 є наявність

спеціалізованих тегів для форм, які дозволяють виконувати валідацію даних на боці клієнта (наприклад, перевірка формату email або обов'язковості заповнення поля) ще до відправки запиту на сервер, що знижує навантаження на бек-енд. CSS3 відповідає за візуальне представлення елементів, анімацію переходів та адаптивність дизайну.

Особливу роль відіграє бібліотека Bootstrap 5 – це фронтенд-фреймворк, який надає набір інструментів для створення сайтів та вебзастосунків. До таких інструментів входять HTML- і CSS-шаблони оформлення для вебформ, типографіки, кнопок, міток, блоків навігації та інших компонентів вебінтерфейсу, включаючи JavaScript-розширення [32].

Головною причиною вибору Bootstrap є його вбудована система адаптивної сітки (Grid System). Ця система базується на використанні контейнерів, рядків та колонок (всього 12 колонок), які автоматично перебудовуються залежно від ширини екрану пристрою.

Це дозволяє реалізувати концепцію Mobile-First: інтерфейс системи «Testify» буде однаково зручним та функціональним як на великому моніторі викладача, так і на екрані смартфона студента (рис. 2.6).

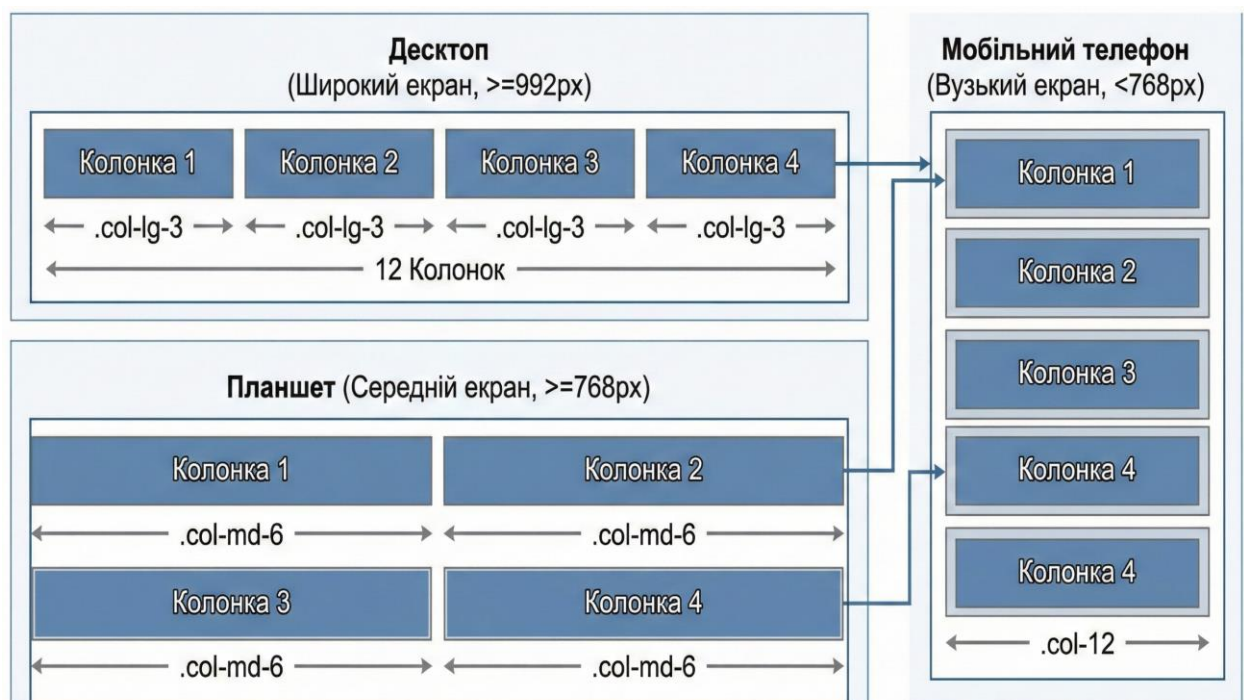


Рисунок 2.6 – Принцип роботи адаптивної сітки Bootstrap

Для написання програмного коду використовується інтегроване середовище розробки (IDE) JetBrains PyCharm – це кросплатформне, потужне і розумне інтегроване середовище розробки, створене компанією JetBrains. Цей інструмент надає все, що вам може знадобитися для комфортної роботи з мовою Python, починаючи від автодоповнення коду і закінчуючи потужним відладчиком та інструментами рефакторингу [33].

PyCharm забезпечує інтелектуальний аналіз коду «на льоту», підсвічуючи синтаксичні помилки, невідповідність стандартам кодування PEP 8 та потенційні логічні проблеми. Вбудований налагоджувач (Debugger) дозволяє виконувати код покроково, інспектувати значення змінних у оперативній пам'яті та відслідковувати стек викликів функцій, що є незамінним при пошуку помилок у складних алгоритмах генерації тестів.

Інтеграція з системою контролю версій Git дозволяє керувати історією змін коду безпосередньо з інтерфейсу IDE, а вбудований термінал та клієнт бази даних створюють єдиний робочий простір, де розробник має доступ до всіх необхідних інструментів без необхідності перемикавання між вікнами.

Розробка складних програмних систем на Python вимагає суворого контролю за версіями використовуваних бібліотек. Проблема конфлікту залежностей (Dependency Hell) виникає, коли різні компоненти системи потребують різних версій однієї й тієї ж бібліотеки.

Для вирішення цієї проблеми у проєкті використовується інструмент `venv` (Virtual Environment), який входить до стандартної бібліотеки Python. Віртуальне оточення створює ізольовану директорію для проєкту, де зберігаються специфічні версії інтерпретатора Python та всіх встановлених пакетів. Це гарантує, що оновлення системних бібліотек операційної системи сервера не призведе до порушення працездатності системи тестування.

Для фіксації списку залежностей використовується стандартний файл `requirements.txt` або більш сучасний інструмент `Poetry`, який дозволяє не лише фіксувати версії прямих залежностей (наприклад, `Django==4.2`), але й

контролювати версії транзитивних залежностей (бібліотек, від яких залежить сам Django).

Це забезпечує відтворюваність середовища розробки (*Reproducibility*): будь-який розробник або системний адміністратор, розгорнувши проєкт на новому сервері, отримає ідентичний набір інструментів, що мінімізує ризик виникнення помилок, пов'язаних з розбіжністю версій програмного забезпечення.

При проектуванні програмної системи критично важливо розрізняти середовище розробки та середовище промислової експлуатації (Production). Вбудований у Django вебсервер, який використовується на етапі написання коду, не призначений для обробки реального трафіку через однопотокову природу та відсутність механізмів захисту від перевантажень.

Тому технологічний стек системи «Testify» включає компоненти для організації надійного серверного середовища. Ключову роль тут відіграє протокол WSGI (Web Server Gateway Interface), який є стандартом взаємодії між вебсерверами та вебзастосунками на Python.

Організація процесу розробки базується на використанні розподіленої системи контролю версій Git. Це дозволяє зберігати повну історію змін програмного коду, повертатися до попередніх версій у разі виявлення помилок та вести паралельну розробку нового функціоналу в окремих гілках (branches).

Для проєкту обрано стратегію розробки GitHub Flow або схожу спрощену модель, де існує основна стабільна гілка main та гілки для розробки окремих функцій (feature branches). Використання файлу .gitignore дозволяє виключити з репозиторію тимчасові файли, налаштування локального середовища, віртуальне оточення та конфіденційні дані (паролі, ключі API), що є важливою вимогою безпеки.

Інтеграція Git з середовищем PyCharm дозволяє виконувати всі операції з версіями (commit, push, pull, merge) через зручний графічний інтерфейс, що

підвищує ефективність роботи розробника та забезпечує цілісність кодової бази проєкту [34].

Для забезпечення повної ізоляції компонентів системи та гарантування ідентичності поведінки програмного забезпечення на етапах розробки, тестування та промислової експлуатації, у технологічний стек проєкту введено технологію контейнеризації Docker. На відміну від класичних віртуальних машин, які емулюють повний набір апаратного забезпечення та потребують встановлення окремої операційної системи, Docker використовує віртуалізацію на рівні ядра операційної системи хоста.

Це дозволяє упаковувати застосунок разом з усіма його залежностями (інтерпретатором Python, системними бібліотеками, конфігураційними файлами) у легковаговий контейнер.

Такий підхід вирішує класичну проблему розробки "it works on my machine", оскільки контейнер, створений на локальній машині розробника, гарантовано запуститься ідентично на будь-якому сервері, що підтримує Docker Engine, незалежно від версії ОС сервера [35].

Вся інфраструктура проєкту описується у декларативному файлі `docker-compose.yml`, де визначаються образи контейнерів, налаштування мережевої взаємодії між ними, змінні оточення та правила монтування томів (Volumes) для збереження персистентних даних.

Це дозволяє розгорнути повний стек системи однією командою консолі, що суттєво спрощує процес впровадження системи у нових навчальних закладах та налаштування конвеєрів безперервної інтеграції (CI/CD).

Використання Docker також підвищує безпеку системи, оскільки кожен сервіс працює в ізольованому просторі імен і має доступ лише до тих ресурсів, які явно дозволені конфігурацією, що мінімізує наслідки потенційного злому одного з компонентів (рис. 2.7).

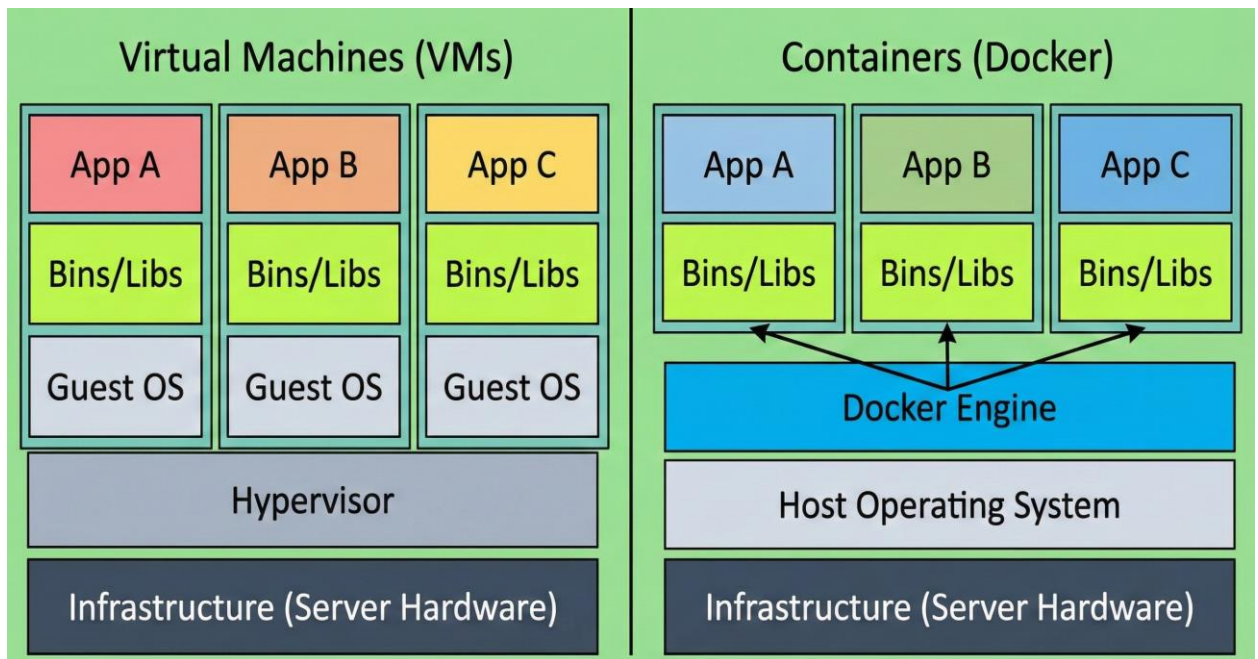


Рисунок 2.7 – Порівняння архітектури віртуальних машин та контейнерів Docker

Використання Docker у розробці системи дистанційного контролю забезпечує наступні архітектурні переваги:

1. Ізоляція ресурсів: кожен компонент системи (вебсервер Django, база даних MySQL, кеш-сервер) працює у власному ізольованому контейнері. Це гарантує, що процеси одного сервісу не зможуть вплинути на роботу інших або отримати несанкціонований доступ до їхніх файлів. Така архітектура підвищує загальний рівень безпеки системи, мінімізуючи "поверхню атаки".

2. Передбачуваність середовища: образ контейнера (Docker Image) створюється на основі декларативного файлу Dockerfile. У цьому файлі чітко прописуються всі кроки складання середовища: вибір базового образу (наприклад, `python:3.10-slim`), встановлення залежностей через `pip`, копіювання вихідного коду та налаштування змінних оточення. Цей підхід, відомий як "Інфраструктура як код" (Infrastructure as Code), гарантує бінарну ідентичність середовища на будь-якому сервері, що підтримує Docker Engine.

3. Ефективність використання ресурсів: завдяки відсутності гіпервізора та гостьової ОС, контейнери запускаються за лічені секунди та споживають значно менше оперативної пам'яті. Це дозволяє розмістити

систему на більш економічних віртуальних серверах (VPS), знижуючи експлуатаційні витрати навчального закладу.

Обраний стек технологій (Python, Django, MySQL, Bootstrap) являє собою збалансоване, перевірене часом рішення, яке поєднує високу продуктивність, безпеку та швидкість розробки.

Використання відкритого програмного забезпечення (Open Source) дозволяє мінімізувати вартість впровадження системи та уникнути ліцензійних обмежень.

Модульна архітектура Django та реляційна структура MySQL створюють надійний фундамент для подальшого масштабування системи та додавання нового функціоналу у майбутньому.

## 3 ОПИС ПРОГРАМНОГО ПРОДУКТУ – СИСТЕМИ ДИСТАНЦІЙНОГО КОНТРОЛЮ ЗНАНЬ СТУДЕНТІВ «TESTIFY»

### 3.1 Вимоги до програмного продукту та інтерфейсу користувача

Формування чітких та несуперечливих вимог до програмного продукту є фундаментальним етапом життєвого циклу розробки будь-якої інформаційної системи, особливо коли йдеться про освітні платформи. Система дистанційного навчання, що розробляється, повинна забезпечувати надійність зберігання даних та надавати зручний інструментарій для взаємодії.

Функціональні вимоги визначають поведінку системи за певних умов та описують набір функцій, які повинні бути реалізовані для досягнення цілей користувачів.

Головною вимогою до розроблюваної системи є забезпечення повного циклу управління навчальним процесом: від створення курсу до виставлення фінальної оцінки. Система повинна підтримувати рольову модель доступу, де базовими ролями є «Викладач» та «Студент».

Для ролі викладача система повинна надавати розширений набір інструментів для управління контентом. Першочерговою вимогою є можливість створення навчальних курсів.

Викладач повинен мати змогу налаштовувати параметри курсу, додавати опис, завантажувати методичні матеріали та структурувати їх.

Критично важливою функціональною вимогою є наявність гнучкого конструктора тестів. Система повинна дозволяти створювати тестові завдання різних типів, зокрема з вибором однієї правильної відповіді. Для кожного тесту викладач повинен мати можливість встановити часові обмеження (дедлайни).

Для ролі студента функціональні вимоги фокусуються на зручності споживання контенту та проходження контрольних заходів. Система повинна забезпечувати простий механізм реєстрації та приєднання до курсів за

допомогою унікального коду доступу або запрошення. Студент повинен мати доступ до особистого кабінету, де відображається перелік активних курсів та список завдань, що потребують виконання.

Після завершення тестування, відповідно до налаштувань викладача, студент повинен отримати миттєвий зворотний зв'язок у вигляді набраних балів та, за необхідності, деталізації помилок.

Враховуючи, що система оперує персональними даними користувачів та результатами їхньої інтелектуальної праці, вимоги до безпеки є пріоритетними. Розроблюваний програмний продукт повинен забезпечувати конфіденційність, цілісність та доступність інформації.

Однією з ключових вимог є реалізація надійного механізму автентифікації та авторизації. Паролі користувачів не повинні зберігатися у відкритому вигляді; обов'язковим є використання надійних алгоритмів хешування.

Масштабованість є ще однією важливою вимогою, яка диктує вибір технологічного стеку. Програмний продукт повинен будуватися за модульним принципом, що дозволить у майбутньому додавати нові функціональні блоки (наприклад, модуль відеоконференцій або чат) без необхідності повної переробки ядра системи. Використання фреймворку Django повністю задовольняє цю вимогу завдяки своїй архітектурі застосунків (Apps), які можна підключати та відключати незалежно один від одного.

Інтерфейс користувача є "обличчям" системи, і від його якості безпосередньо залежить ефективність навчання. Основною вимогою до UI є інтуїтивна зрозумілість та мінімалізм. Користувач не повинен витратити час на вивчення інструкцій по роботі з системою; навігація повинна бути очевидною, а розташування елементів – звичним. Концепція дизайну повинна базуватися на принципах Material Design або аналогічних сучасних дизайн-системах.

Враховуючи сучасні тенденції споживання контенту, критичною вимогою є адаптивність інтерфейсу (Responsive Design). Значна частина

студентів використовує для навчання мобільні пристрої та планшети. Інтерфейс системи повинен автоматично підлаштовуватися під розмір екрану пристрою, забезпечуючи читабельність тексту та зручність натискання кнопок без необхідності масштабування сторінки.

Всі елементи управління (кнопки, поля вводу) повинні мати достатній розмір для комфортної взаємодії через сенсорний екран.

З точки зору UX (User Experience), система повинна мінімізувати когнітивне навантаження на користувача. Наприклад, при створенні тесту форма не повинна бути перевантажена зайвими полями; складні налаштування краще приховати під спойлером "Додатково".

Важливою є візуалізація статусу системи: користувач повинен завжди розуміти, що відбувається (наприклад, індикатор завантаження при відправці файлу або повідомлення про успішне збереження змін).

Підсумовуючи, визначені у цьому розділі вимоги створюють цілісну картину майбутнього програмного продукту. Функціональні вимоги гарантують, що система буде вирішувати поставлені педагогічні завдання.

Вимоги до безпеки та надійності забезпечують захист даних та стабільність роботи. Вимоги до інтерфейсу фокусуються на зручності кінцевого користувача, а технічні вимоги визначають технологічний стек, необхідний для реалізації проекту. Дотримання цього комплексу вимог є запорукою створення якісної системи дистанційного контролю знань, яка зможе ефективно конкурувати з існуючими аналогами.

### 3.2 Структура інформаційної бази системи «Testify»

Проектування ефективної структури інформаційної бази є фундаментом для створення надійної, масштабованої та швидкодіючої інформаційної системи.

У контексті розробки системи дистанційного навчання «TESTIFY», яка оперує великими обсягами пов'язаних даних (користувачі, навчальні курси, тестові завдання, результати оцінювання, файли студентських робіт), вибір моделі даних та її коректна реалізація визначають стабільність роботи всього програмного комплексу.

Інформаційна база системи побудована на основі реляційної моделі даних. Вибір реляційної моделі обумовлений чіткою структурованістю навчальних даних та наявністю стійких зв'язків між сутностями (наприклад, один курс має багато студентів, один тест складається з багатьох питань тощо).

Фізична реалізація бази даних виконана з використанням системи управління базами даних (СУБД) PostgreSQL, яка є стандартом індустрії для високонавантажених вебсистем на базі Python/Django завдяки підтримці ACID-транзакцій, складних типів даних та високій надійності.

Схема бази даних була спроектована з дотриманням принципів нормалізації.

Більшість відношень (таблиць) знаходяться у третій нормальній формі (3NF), що забезпечує усунення надлишковості даних, запобігання аномаліям при вставці, оновленні та видаленні інформації, а також забезпечує логічну цілісність бази.

Взаємодія програмного коду з базою даних здійснюється через механізм ORM (Object-Relational Mapping) фреймворку Django, що дозволяє працювати з таблицями як з класами мови Python, автоматично генеруючи безпечні SQL-запити (рис.3.1).

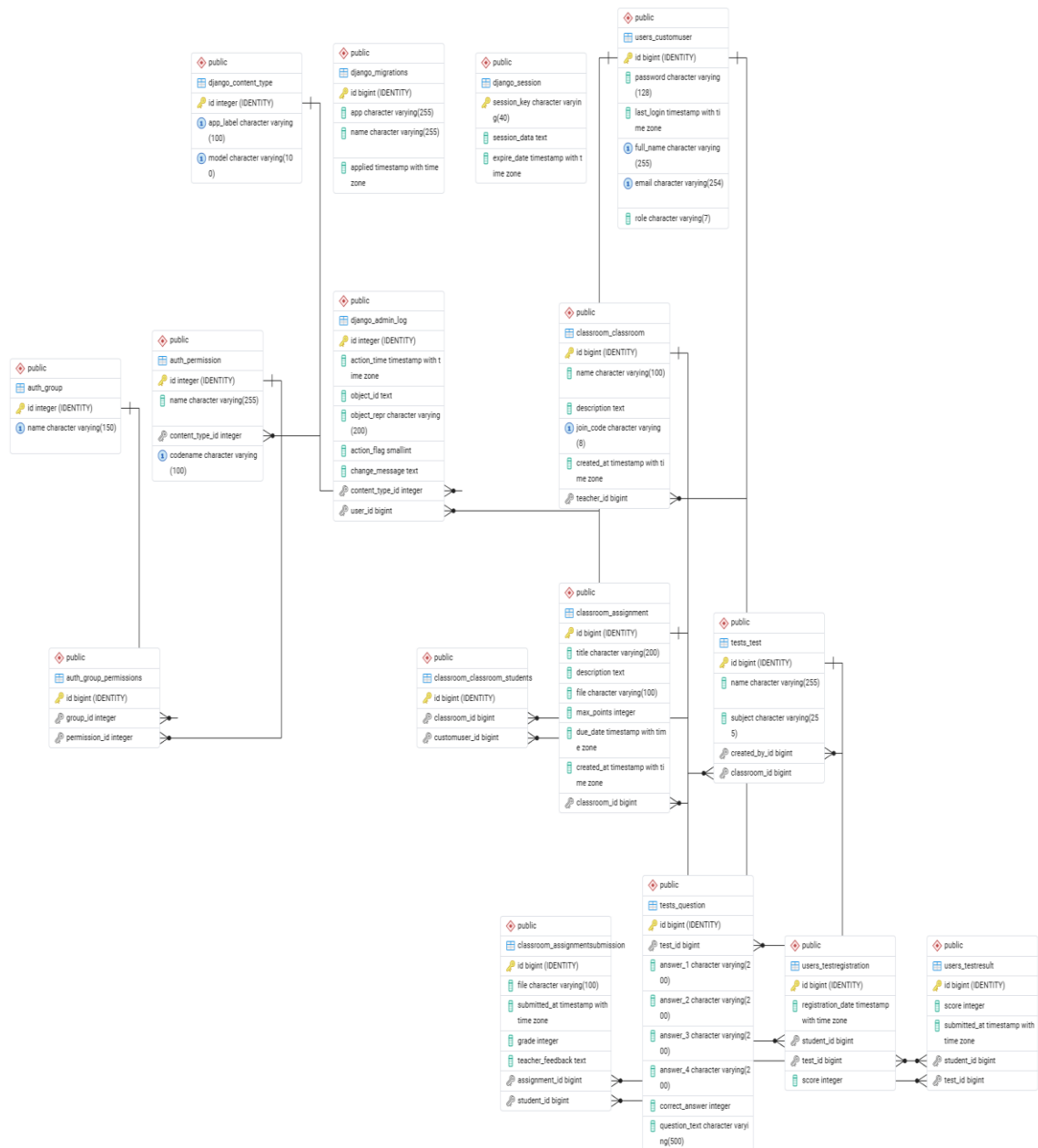


Рисунок 3.1 – ERD діаграма бази даних проєкту

Загальну структуру інформаційної бази, представлену на ER-діаграмі (Entity-Relationship Diagram), можна логічно розділити на три функціональні кластери:

1. Системний кластер та автентифікація (таблиці ядра Django та користувачів).
2. Кластер навчального процесу (класи, завдання).
3. Кластер тестування та оцінювання (тести, питання, результати).

Центральним елементом будь-якої багатокористувацької системи є таблиця зберігання облікових записів. У системі «TESTIFY» для цього використовується кастомізована сутність **users\_customuser**.

Таблиця **users\_customuser** Ця таблиця є розширенням стандартної моделі користувача Django і містить критично важливі атрибути для ідентифікації та авторизації учасників навчального процесу (рис. 3.2).

- **id (bigint, PK):** Унікальний ідентифікатор користувача. Використання типу даних bigint (64-бітне ціле число) забезпечує можливість реєстрації практично необмеженої кількості користувачів без ризику переповнення лічильника.
- **password (varchar 128):** Зберігає хеш пароля користувача. З міркувань безпеки паролі ніколи не зберігаються у відкритому вигляді. Django використовує алгоритм PBKDF2 з сіллю для хешування.
- **last\_login (timestamp with time zone):** Часова мітка останньої успішної авторизації. Використовується для аналітики активності та політик безпеки.
- **full\_name (varchar 255):** Поле для зберігання реального імені та прізвища користувача, що використовується для відображення у відомостях та інтерфейсі.
- **email (varchar 254):** Електронна пошта, яка виступає унікальним логіном у системі. Це поле має обмеження UNIQUE на рівні СУБД.
- **role (varchar 7):** Ключове поле, що визначає тип доступу («student» або «teacher»). На основі цього атрибута будується бізнес-логіка розмежування прав доступу в інтерфейсі.

The screenshot shows the pgAdmin 4 interface. On the left is the Object Explorer showing a database structure with tables like 'auth\_group', 'classroom\_assignment', and 'users\_customuser'. The main pane shows a SQL query: `SELECT * FROM public.users_customuser ORDER BY id ASC`. Below the query, the 'Data Output' tab displays the results of the query as a table with 7 rows and 5 columns: `id`, `last_login`, `full_name`, `email`, and `role`.

| id | last_login                     | full_name           | email             | role    |
|----|--------------------------------|---------------------|-------------------|---------|
| 1  | 2025-12-05 17:18:05.968395+... | TEST                | test@gmail.com    | student |
| 2  | 2025-12-05 23:23:50.319042+... | ivan                | ivan@gmail.com    | teacher |
| 3  | 2025-12-11 23:38:38.111579+... | pityx               | pityx@gmail.com   | teacher |
| 4  | 2025-12-11 23:50:59.871847+... | bb                  | bb@gmail.com      | student |
| 5  | 2025-12-10 20:32:50.33812+00   | student             | student@gmail.com | student |
| 6  | 2025-12-12 11:21:53.561974+... | Балацький Данило    | test1@gmail.com   | student |
| 7  | 2025-12-12 11:20:44.958659+... | Балацька Олександра | test2@gmail.com   | teacher |

At the bottom of the Data Output pane, it says 'Total rows: 7' and 'Query complete 00:00:00.643'.

Рисунок 3.2 – Таблиця бази даних користувачів

Таблиця **classroom\_classroom** Сутність, що репрезентує навчальний курс або дисципліну.

- **id (bigint, PK):** Первинний ключ.
- **name (varchar 100):** Назва курсу.
- **description (text):** Детальний опис курсу, програмні вимоги або методичні вказівки. Тип `text` дозволяє зберігати необмежений обсяг інформації.
- **join\_code (varchar 8):** Унікальний код приєднання. Це короткий рядок, який генерується системою і передається викладачем студентам для самостійної реєстрації на курсі.
- **teacher\_id (bigint, FK):** Зовнішній ключ, що посилається на таблицю `users_customuser`. Реалізує зв'язок «Один-до-багатьох» (One-to-Many): один

викладач може створити багато курсів, але кожен курс має одного власника-викладача.

- **created\_at:** Час створення курсу.

Таблиця **classroom\_assignment** Зберігає інформацію про практичні завдання, створені викладачем.

- **title, description:** Назва та зміст завдання.
- **file (varchar 100):** Посилання на файл (методичку або приклад), завантажений викладачем. Зберігається лише шлях до файлу на диску сервера.
- **max\_points (integer):** Максимальна оцінка за завдання.
- **due\_date:** Дедлайн (термін здачі).
- **classroom\_id (FK):** Прив'язка завдання до конкретного курсу.

Таблиця **classroom\_assignmentsubmission** Репрезентує відповідь студента на завдання.

- **file:** Файл з виконаною роботою, завантажений студентом.
- **submitted\_at:** Час фактичної здачі. Порівнюється з **due\_date** для визначення своєчасності.
- **grade (integer):** Оцінка, виставлена викладачем.
- **teacher\_feedback (text):** Текстовий коментар викладача до роботи.
- **assignment\_id (FK)** та **student\_id (FK):** Зв'язки, що вказують, хто і на яке завдання надав відповідь.

Найбільш складною частиною схеми є блок таблиць, що забезпечують функціонування конструктора тестів та збереження результатів.

Таблиця **tests\_test** Заголовок тесту.

- **name (varchar 255):** Назва тесту.
- **subject (varchar 255):** Предмет або тема тестування.
- **classroom\_id (FK):** Прив'язка тесту до курсу. Це означає, що тести є частиною навчальних матеріалів курсу.

- **created\_by\_id (FK):** Посилання на викладача, який створив тест.

Таблиця **tests\_question** Зберігає окремі питання, що входять до складу тестів.

- **test\_id (bigint, FK):** Зовнішній ключ на таблицю tests\_test. Реалізує зв'язок «Один-до-багатьох» (один тест складається з багатьох питань).
- **question\_text (varchar 500):** Текст запитання.
- **answer\_1, answer\_2, answer\_3, answer\_4 (varchar 200):** Тексти варіантів відповідей. У поточній реалізації схема оптимізована під фіксовану кількість варіантів (чотири), що спрощує структуру, хоча і накладає певні обмеження.
- **correct\_answer (integer):** Числовий індекс (наприклад, 1, 2, 3 або 4), що вказує, який з варіантів є правильним. Це поле є ключовим для роботи алгоритму автоматичної перевірки.

Таблиця **users\_testresult** Зберігає фінальні результати проходження тесту.

- **score (integer):** Кількість набраних балів.
- **submitted\_at (timestamp):** Час завершення тестування.
- **student\_id (FK)** та **test\_id (FK):** Вказують, хто саме і який тест проходив. Ця таблиця є джерелом даних для журналу успішності.

Таблиця **users\_testregistration** Допоміжна таблиця для контролю процесу тестування.

- **registration\_date:** Фіксує час, коли студент відкрив тест або натиснув кнопку "Почати". Це дозволяє реалізувати логіку таймера (обмеження часу на проходження), порівнюючи час початку з поточним часом.

Важливим аспектом проектування схеми є використання обмежень цілісності (Integrity Constraints), які відображені на ER-діаграмі у вигляді зв'язків.

1. **Первинні ключі (Primary Keys):** Кожна таблиця має поле id типу bigint IDENTITY, яке гарантує унікальність кожного запису. Це вимога першої нормальної форми.

2. **Зовнішні ключі (Foreign Keys):** Зв'язки між таблицями (лінії на діаграмі) реалізовані через зовнішні ключі. Наприклад, поле teacher\_id у

таблиці `classroom_classroom` гарантує, що курс не може існувати без прив'язки до існуючого користувача.

**3. Каскадне видалення (Cascading Delete):** У системі налаштовано поведінку `ON DELETE CASCADE` для більшості залежних сутностей. Наприклад, якщо викладач видаляє курс (`classroom_classroom`), СУБД автоматично видаляє всі пов'язані з ним завдання (`classroom_assignment`), підписки студентів (`classroom_classroom_students`) та тести (`tests_test`). Це запобігає появі "сирітських" записів у базі даних, які б засмічували пам'ять.

Розроблена структура інформаційної бази системи «TESTIFY» є повністю нормалізованою та відповідає всім функціональним вимогам, визначеним у технічному завданні. Використання окремих таблиць для сутностей навчального процесу та їх зв'язування через зовнішні ключі забезпечує гнучкість архітектури. Розділення таблиць `tests_test` та `tests_question` дозволяє створювати тести, а таблиця проміжних зв'язків `classroom_students` дозволяє ефективно керувати зарахуванням студентів.

Схема БД спроектована з урахуванням можливостей PostgreSQL та Django ORM, що гарантує надійність зберігання даних, цілісність посилань та високу швидкість обробки запитів користувачів.

### 3.3 Основні функції та інтерфейс системи «Testify»

Розроблена система дистанційного контролю знань «Testify» являє собою веб-орієнтований програмний комплекс, призначений для автоматизації навчального процесу та контролю знань студентів. Функціональна архітектура системи спроектована з урахуванням потреб двох ключових груп користувачів – викладачів та студентів, забезпечуючи для кожної з них спеціалізований набір інструментів та інтерфейсів.

Реалізація клієнтської частини системи базується на принципах інтуїтивності (Usability) та адаптивності (Responsive Design), що дозволяє

ефективно використовувати програмний продукт як на стаціонарних комп'ютерах, так і на мобільних пристроях без необхідності встановлення додаткового програмного забезпечення.

Взаємодія користувача з системою відбувається через графічний веб-інтерфейс (GUI), який візуалізує бізнес-логіку застосунку та забезпечує доступ до основних функціональних модулів: автентифікації, управління курсами, конструктора тестів та журналу успішності.

Перша сторінка, з якою буде взаємодіяти користувач – це сторінка реєстрації вона є точкою входу нових користувачів до навчальної вебсистеми.

Інтерфейс спроектовано таким чином, щоб насамперед надати користувачеві можливість ідентифікувати свою роль у навчальному процесі.

У верхній частині форми розташовані перемикачі «Студент» та «Викладач», що дозволяє системі одразу розмежувати права доступу та функціонал майбутнього кабінету. (рис. 3.3).

The image shows a web registration form titled "Регістрація" (Registration). At the top, there are two buttons for role selection: "Студент" (Student) and "Викладач" (Lecturer). Below these are four input fields: "ПІБ" (Full Name) with the placeholder "Введіть ваш ПІБ", "Пошта" (Email) with the placeholder "example@gmail.com", "Пароль" (Password) with the placeholder "Введіть пароль", and "Підтвердження паролю" (Confirm Password) with the placeholder "Повторіть пароль". At the bottom of the form is a large blue button labeled "Створити акаунт" (Create Account). Below this button is a link that says "Вже є акаунт? Увійти" (Already have an account? Log in).

Рисунок 3.3 – Сторінка реєстрації користувача

Далі зображено головну сторінку вебсистеми – «Дашборд», на яку користувач потрапляє відразу після успішної реєстрації або авторизації в ролі студента. Ця сторінка виступає центральним вузлом навігації та управління навчальним процесом.

Оскільки основним завданням студента є проходження курсів, центральне місце займає функціонал для додавання нових дисциплін. Користувач бачить поле введення з підказкою (placeholder) «Наприклад: G7H2K».

Сценарій роботи передбачає, що студент вводить у це поле унікальний код-ідентифікатор, який йому надав викладач, і натискає кнопку «Приєднатися».

Це ініціює запит на додавання студента до відповідної навчальної групи. У нижній частині інтерфейсу передбачено область для відображення переліку курсів, до яких користувач вже має доступ (рис.3.4).

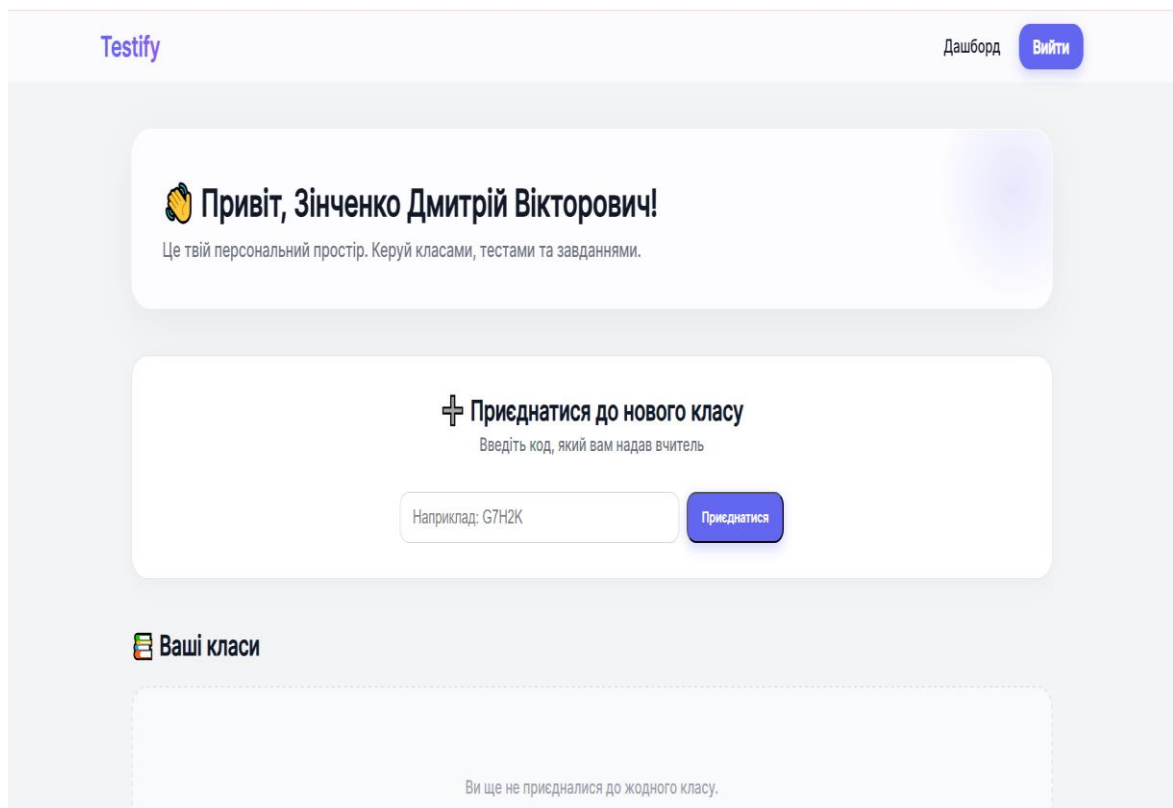


Рисунок 3.4 – Головна сторінка студента

Далі продемонстровано головну сторінку адміністративної панелі викладача. Цей інтерфейс відкривається користувачеві після авторизації з відповідними правами доступу та слугує центром управління навчальним контентом.

Ключовою відмінністю від інтерфейсу студента є наявність акцентної кнопки «+ Створити новий клас», виконаної у зеленому кольорі. Вона розташована на видному місці та є основним заклик до дії (Call-to-Action). Натискання на цю кнопку ініціює сценарій створення нового навчального простору (класу), куди згодом зможуть долучатися студенти. (рис. 3.5).

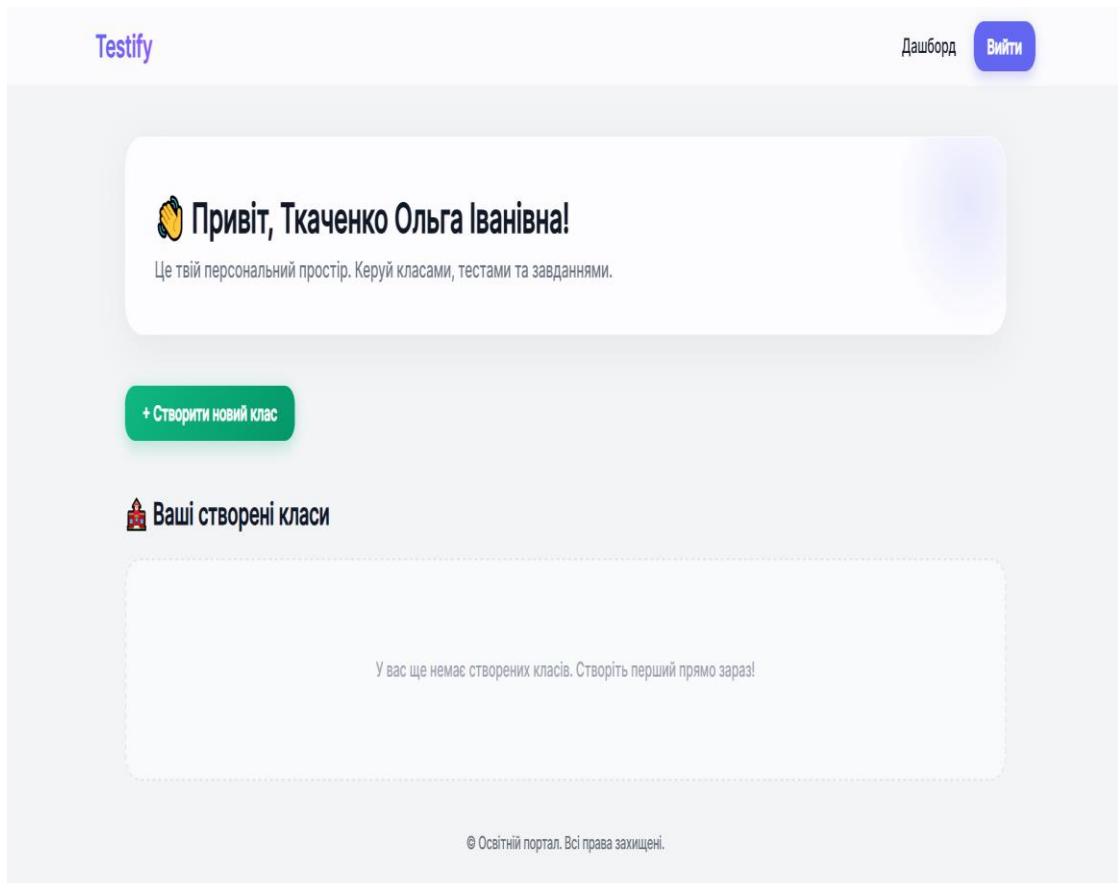


Рисунок 3.5 - Головна сторінка викладача

Наступним зображено інтерфейс сторінки створення класу, яка реалізована як окремий розділ вебсистеми. Таке архітектурне рішення дозволяє користувачеві зосередитися на налаштуваннях нової дисципліни, не відволікаючись на інші елементи інтерфейсу.

Центральну частину екрану займає форма введення даних, яка починається з заголовка та короткої інструкції для викладача.

Основним елементом взаємодії є поле для введення назви класу, яке оснащено текстовою підказкою для полегшення заповнення. Під ним розташовано область для детального опису курсу, яка є необов'язковою до заповнення. У нижній частині форми розміщено функціональні клавiші: основна кнопка для ініціації створення класу в базі даних та кнопка скасування, що дозволяє повернутися до головної панелі управління без внесення змін (рис. 3.6).

The screenshot shows a web interface for creating a class. At the top left is the 'Testify' logo. At the top right are links for 'Дашборд' (Dashboard) and 'Вийти' (Logout). The main content area features a white card titled 'Створення класу' (Create Class) with the instruction 'Заповніть форму, щоб створити новий простір для навчання.' (Fill out the form to create a new learning space). The form includes a 'Назва класу:' (Class Name) field with a placeholder 'Наприклад: Інженерія програмного забезпечення' (Example: Software Engineering), an 'Опис класу (Необов'язково)' (Class Description (Optional)) text area with a placeholder 'Короткий опис класу' (Brief description of the class), a blue 'Створити клас' (Create Class) button, and a 'Скасувати' (Cancel) link. The footer contains the text '© Освітній портал. Всі права захищені.' (© Educational portal. All rights reserved.).

Рисунок 3.6 – Сторінка створення класу

Далі представлено інтерфейс сторінки управління навчальним класом. Верхня інформаційна панель містить назву курсу, дані викладача та унікальний код доступу для реєстрації студентів. Основна робоча область розділена на два функціональні блоки – «Тести» та «Завдання», кожен з яких оснащений відповідними кнопками для додавання нового контенту.

Оскільки клас щойно створено, ці секції демонструють стан відсутності даних, підказуючи необхідність наповнення курсу. У нижній частині екрану розміщено кнопку навігації для повернення до загального дашборду (рис. 3.7).

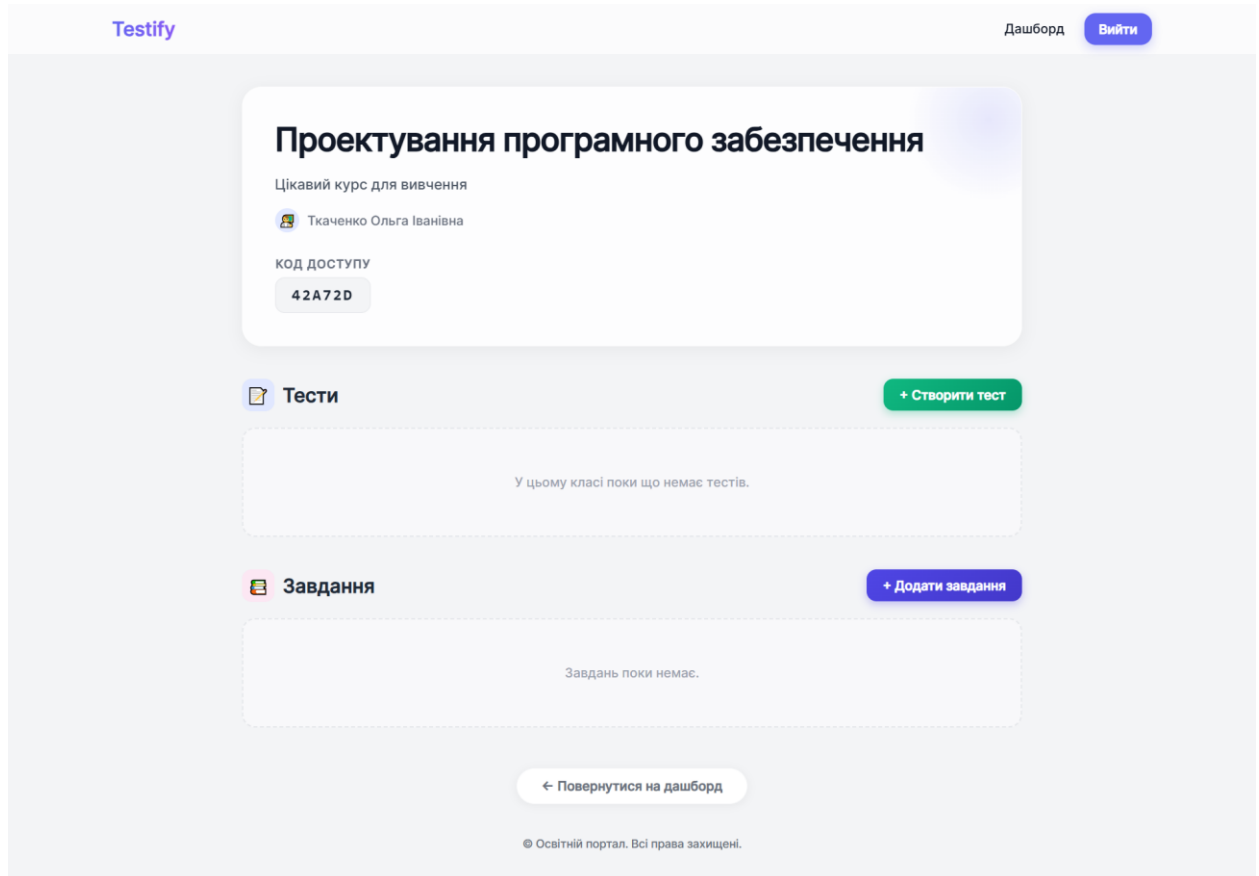


Рисунок 3.7 – Сторінка створеного класу

Після натискання кнопки «Створити тест» представлено інтерфейс конструктора тестів. Верхня панель містить назву тесту, дисципліну та лічильник питань.

Центральну частину займає форма створення нового питання, яка включає поля для тексту завдання, варіантів відповідей та механізм вибору правильного варіанту для системи автоматичного оцінювання.

Збережені питання додаються до списку у нижній частині сторінки, формуючи структуру тесту. Інтерфейс завершується кнопкою навігації для повернення до сторінки класу (рис. 3.8).

Testify

Дашборд Вийти

ТЕСТ Проектування програмного забезпечення

### Тестування номер 1

Кількість питань: 0

### Результати учнів

Ще ніхто не склав цей тест.

### Додати нове питання

Текст питання

Що є основною метою проектування програмного забезпечення?

Варіант 1: Написати якнайбільше коду

Варіант 2: Створити зрозумілу структуру системи перед реалізацією

Варіант 3: Перевірити програму на помилки

Варіант 4: Зменшити кількість учасників команди

Оберіть правильну відповідь:

☐ Варіант 1 ☒ Варіант 2 ☐ Варіант 3 ☐ Варіант 4

Зберегти питання

### Існуючі питання

Питань ще немає.

← Повернутися до класу

© Освітній портал. Всі права захищені.

Рисунок 3.8 – Меню створення тестів

Після додавання запитання воно з'являється в нижній частині сторінки та показує правильну відповідь для викладача (рис. 3.9)

### Існуючі питання

1. Що є основною метою проектування програмного забезпечення?

1. Написати якнайбільше коду

2. Створити зрозумілу структуру системи перед реалізацією

3. Перевірити програму на помилки

4. Зменшити кількість учасників команди

← Повернутися до класу

Рисунок 3.9 – Додане питання для тесту

Після натискання кнопки «Створити завдання» представлено інтерфейс сторінки створення завдання.

Форма включає поля для введення назви роботи, текстового опису умов та модуль завантаження допоміжних файлів.

Для контролю навчального процесу викладач встановлює максимальний бал та дедлайн виконання. Завершується створення натисканням кнопки публікації, що робить завдання доступним для студентів (рис 3.10).

The screenshot shows the 'Створити завдання' (Create Task) form in the Testify application. The form is titled 'Створити завдання' and is for the class 'Проектування програмного забезпечення'. It contains the following fields and controls:

- Назва:** Text input field with the value 'Практична 1'.
- Опис:** Text input field with the value 'Виконати завдання з файлу'.
- Прикріпити файл (необов'язково):** A button labeled 'Вибрати файл' and a text input field with the value 'Практична 1.docx'.
- Бали:** Text input field with the value '25'.
- Дедлайн:** Text input field with the value '20.12.2025 16:30' and a calendar icon.
- Buttons:** A large blue button labeled 'Опублікувати завдання' and a smaller button labeled 'Скасувати'.

The form is set against a light blue background with a subtle gradient. The Testify logo is in the top left corner, and the user's name 'Дашборд' and a 'Вийти' button are in the top right corner.

Рисунок 3.10 – Меню створення завдання

Після створення тесту та завдання вони з'являються на головній сторінці класу де викладач може їх переглядати (рис. 3.11).

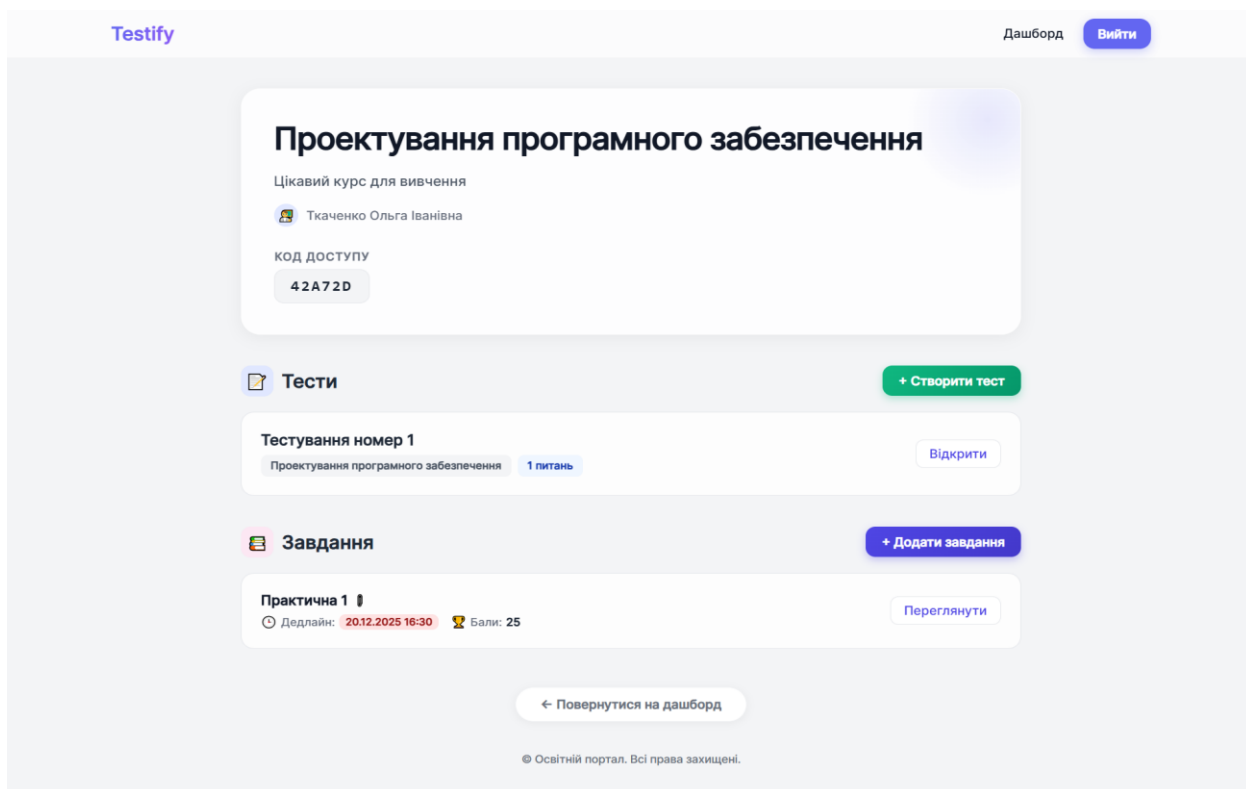


Рисунок 3.11 – Головна сторінка класу зі створеними завданнями

Тепер повернемося до сторінки студента для того, щоб здати всі завдання. Для цього спочатку потрібно пройти авторизацію на сторінці «Вхід» (рис. 3.12).

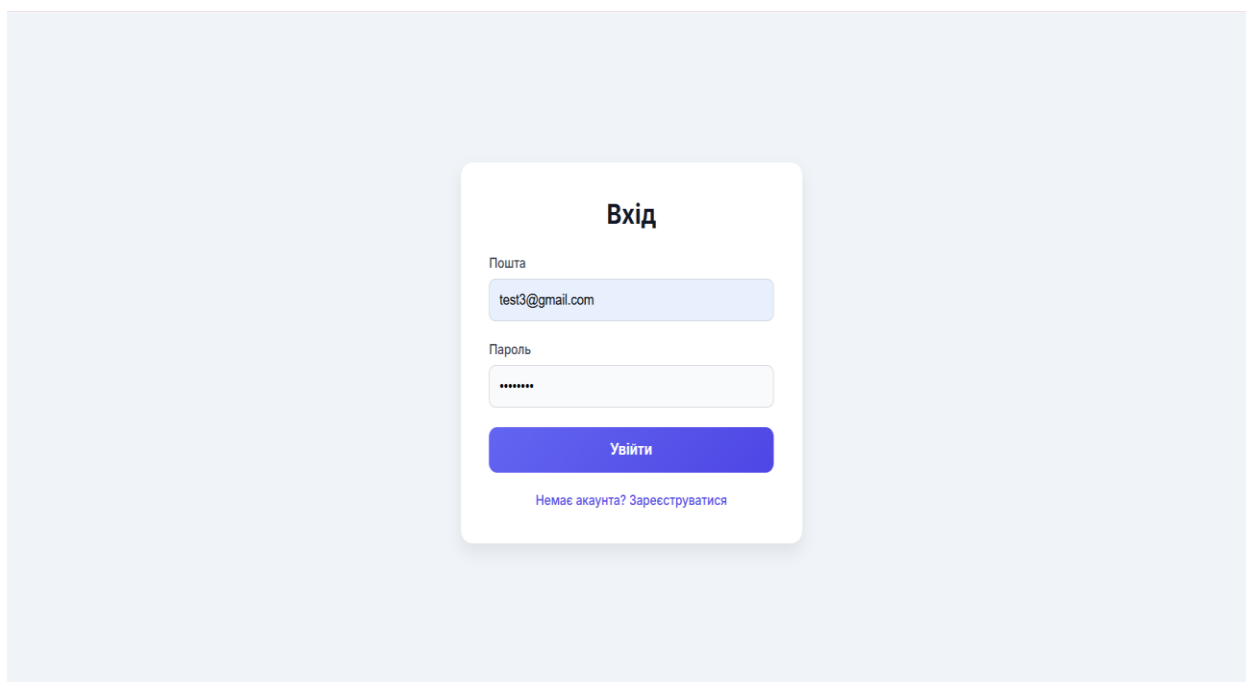


Рисунок 3.12 – Сторінка вхід

Далі приєднаємося до класу за кодом доступу, який надає викладач в нашому випадку – це 42A72D (рис. 3.13).

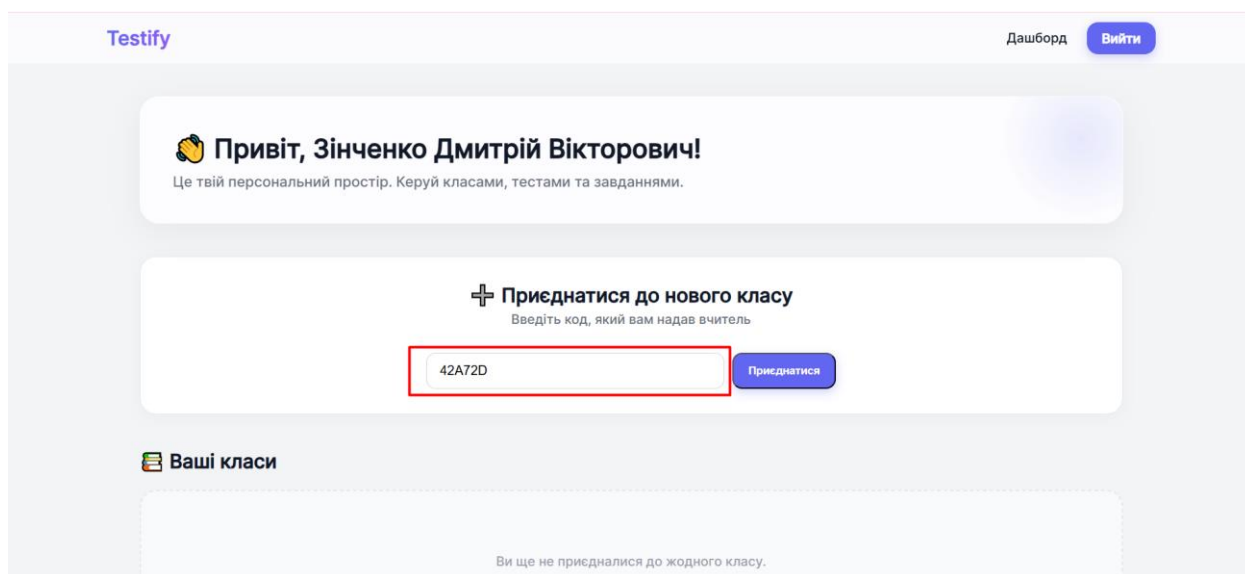


Рисунок 3.13 – Приєднання до класу за кодом

Після приєднання потрапляємо на головну сторінку класу де можемо здавати завдання (рис. 3.14).

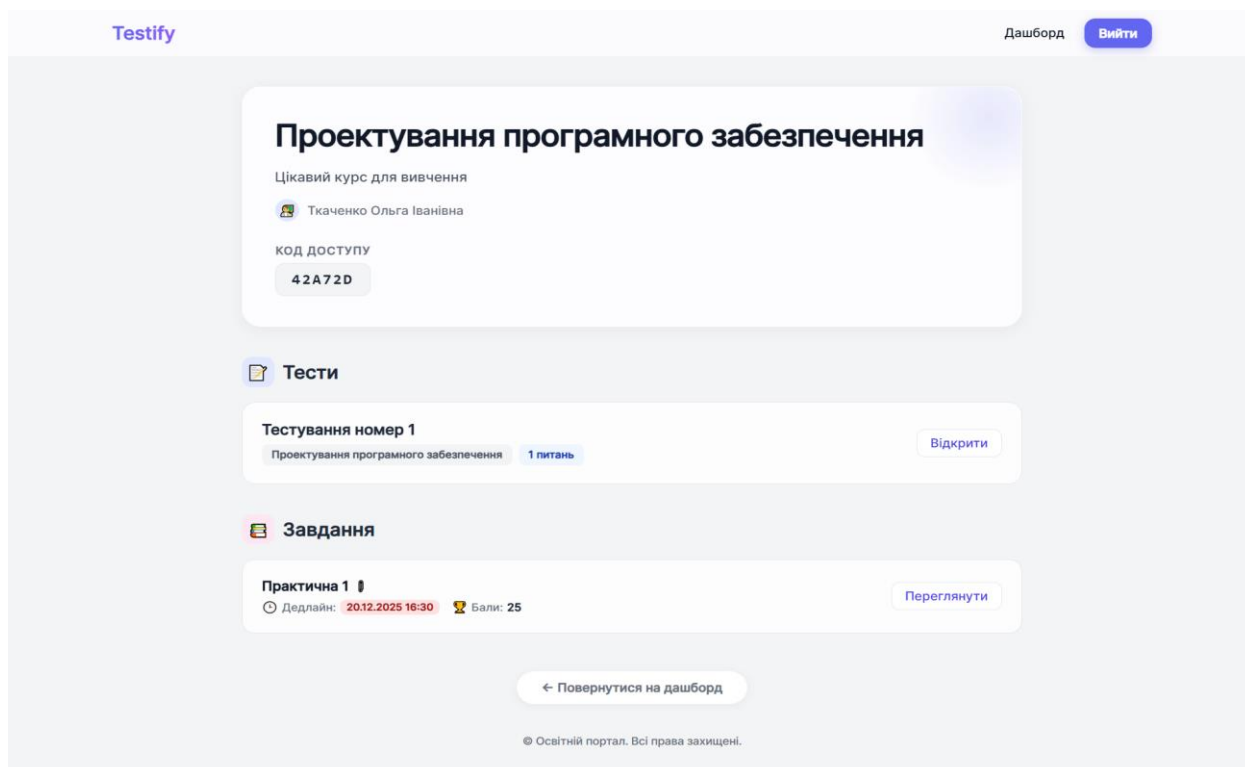


Рисунок 3.14 – Головна сторінка класу

Після цього обираємо тести та починаємо їх виконувати. Де відображено покроковий інтерфейс складання тесту студентом. Перший екран є стартовою сторінкою, яка містить назву дисципліни, номер тестування та кнопку для початку роботи.

Наступний екран демонструє безпосередній процес відповіді на запитання: користувач бачить сформульоване завдання та обирає варіант відповіді зі списку за допомогою радіо-кнопок.

Інтерфейс завершується кнопкою, що дозволяє фіналізувати спробу та відправити результати на автоматичну перевірку (рис. 3.15, рис. 3.16).

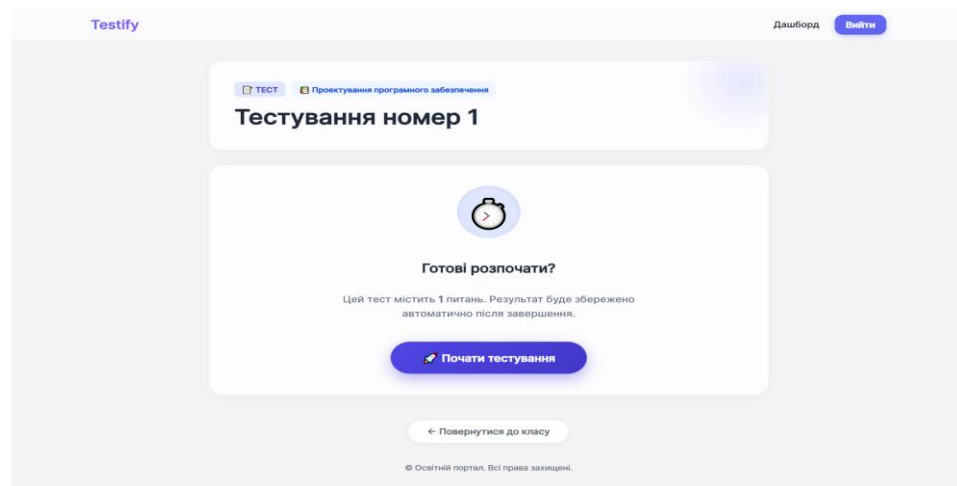
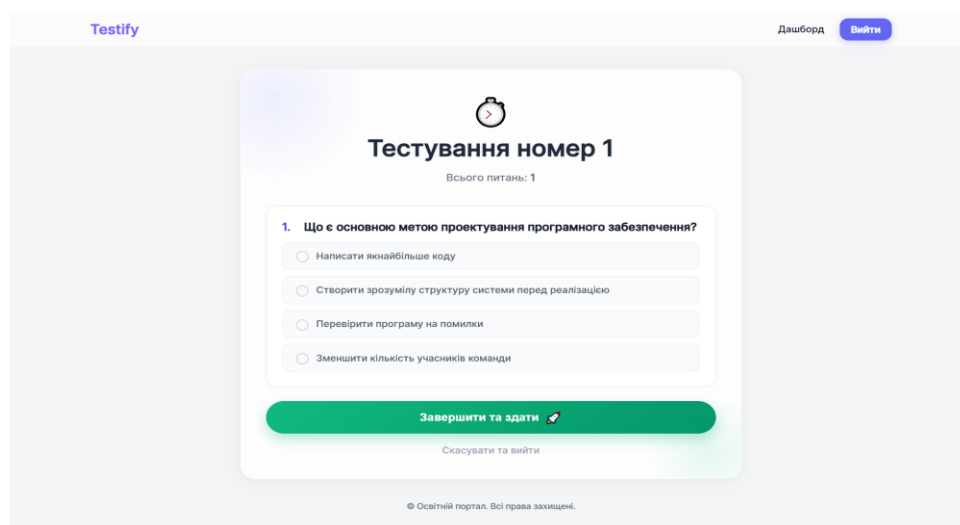


Рисунок 3.15 Виконання завдань тесту



Рисунки 3.16 Виконання завдань тесту

Нижче продемонстровано фінальну сторінку модуля тестування, яка відображається студенту відразу після відправки відповідей на перевірку.

Центральним елементом інтерфейсу є картка результатів, що містить повідомлення про завершення тесту та блок з отриманою оцінкою, який автоматично розраховується системою. Нижній частині сторінки розміщено навігаційний елемент, який дозволяє користувачеві повернутися до переліку матеріалів навчальної дисципліни, завершуючи сценарій проходження контрольного заходу (рис. 3.17).

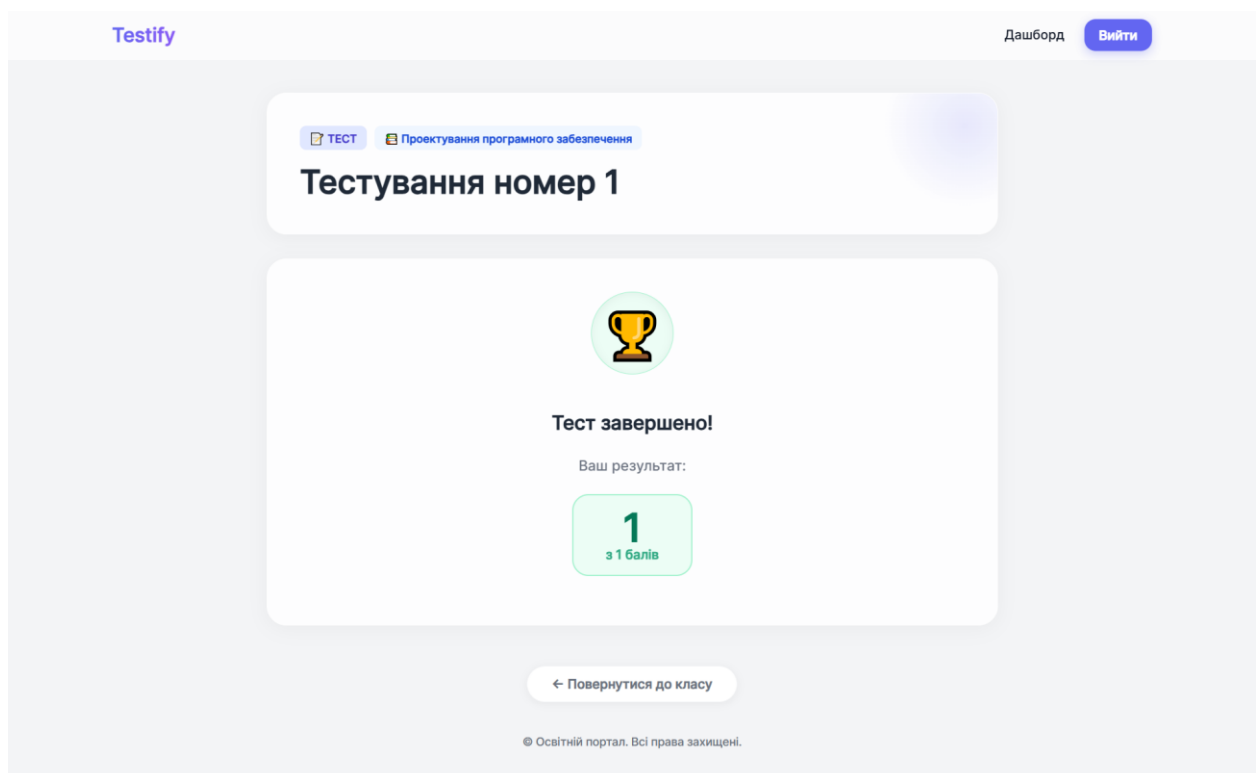


Рисунок 3.17 – Сторінка здачі тесту

Далі сторінка виконання практичного завдання з боку студента. Верхня інформаційна картка містить ключові параметри роботи: назву, максимальну оцінку та кінцевий термін здачі, який візуально виділено для акцентування уваги на часових обмеженнях.

Також у цьому блоці розміщено інструкцію та кнопку для завантаження супровідних матеріалів, наданих викладачем. Нижній блок, призначений

безпосередньо для звітування, включає модуль вибору файлу та попередження про неможливість внесення змін після відправки (рис. 3.18).

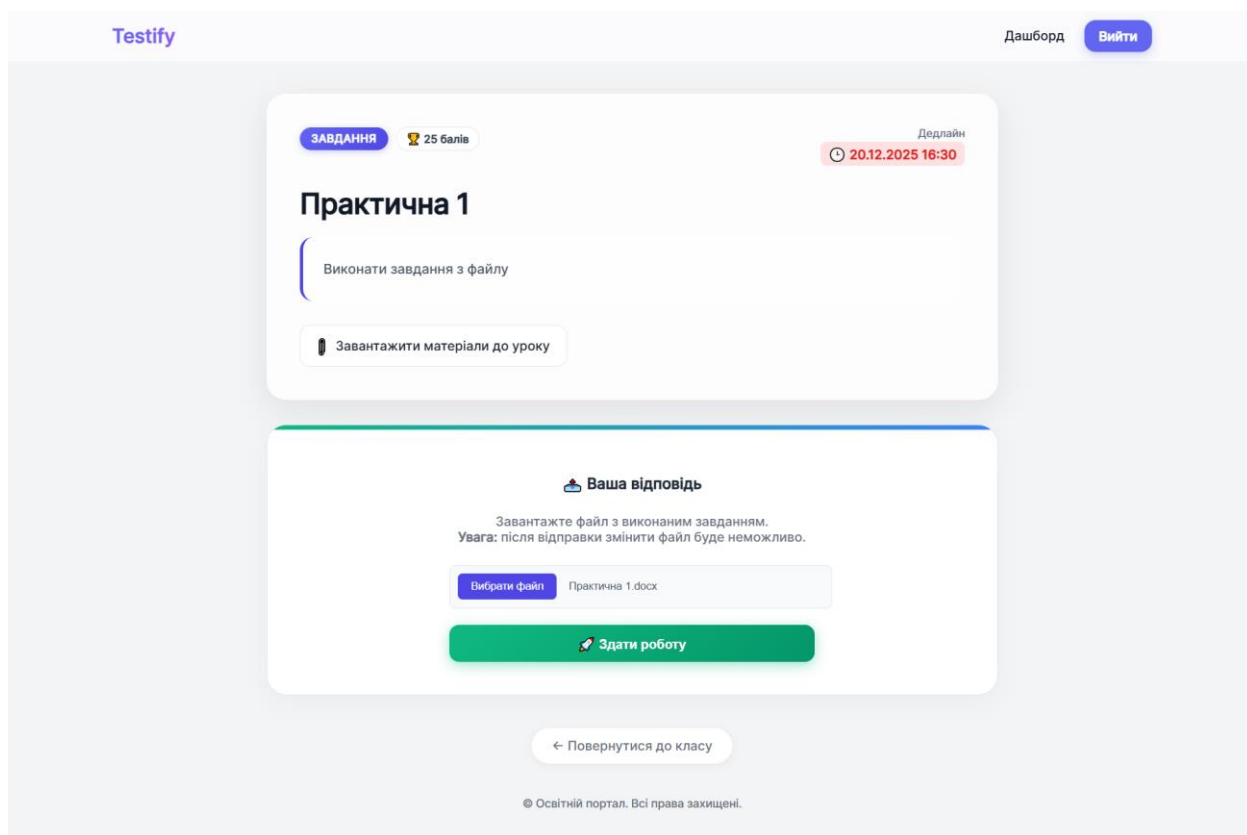


Рисунок 3.18 – Сторінка здачі завдання

Після виконання тесту (проходження тесту) бачимо інформацію про те, що робота надіслана та очікує перевірки викладачем (рис. 3.19).

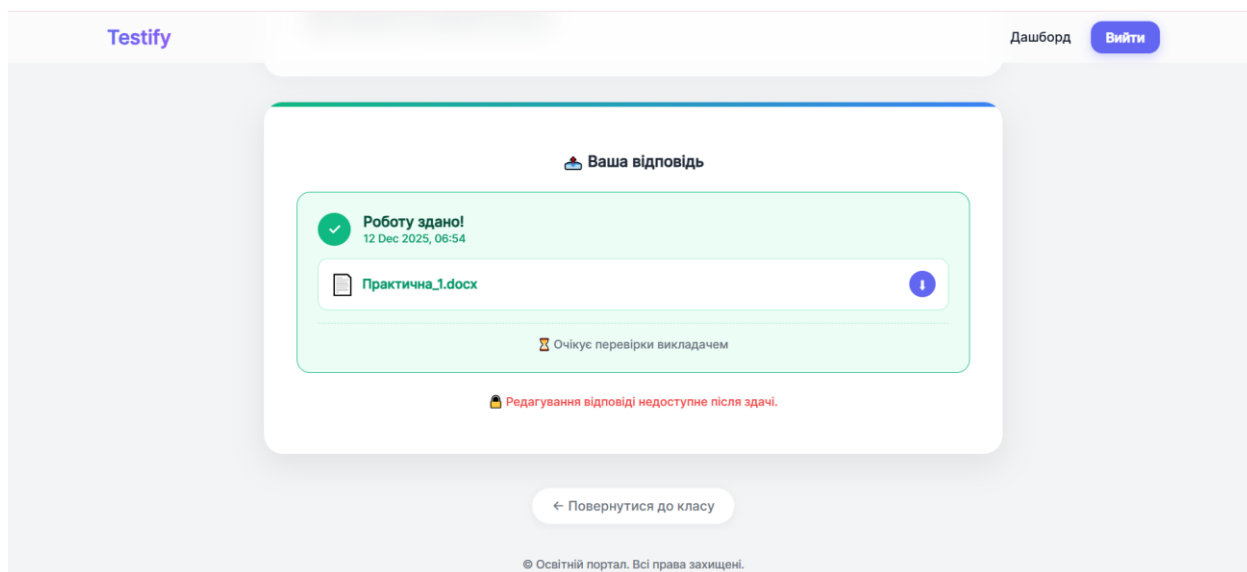


Рисунок 3.19 – Інформація про надіслану роботу

Після виконання завдання тесту викладач бачить надіслану роботу, може її скачати та яку залишається перевірити та виставити оцінку студенту (рис. 3.20).

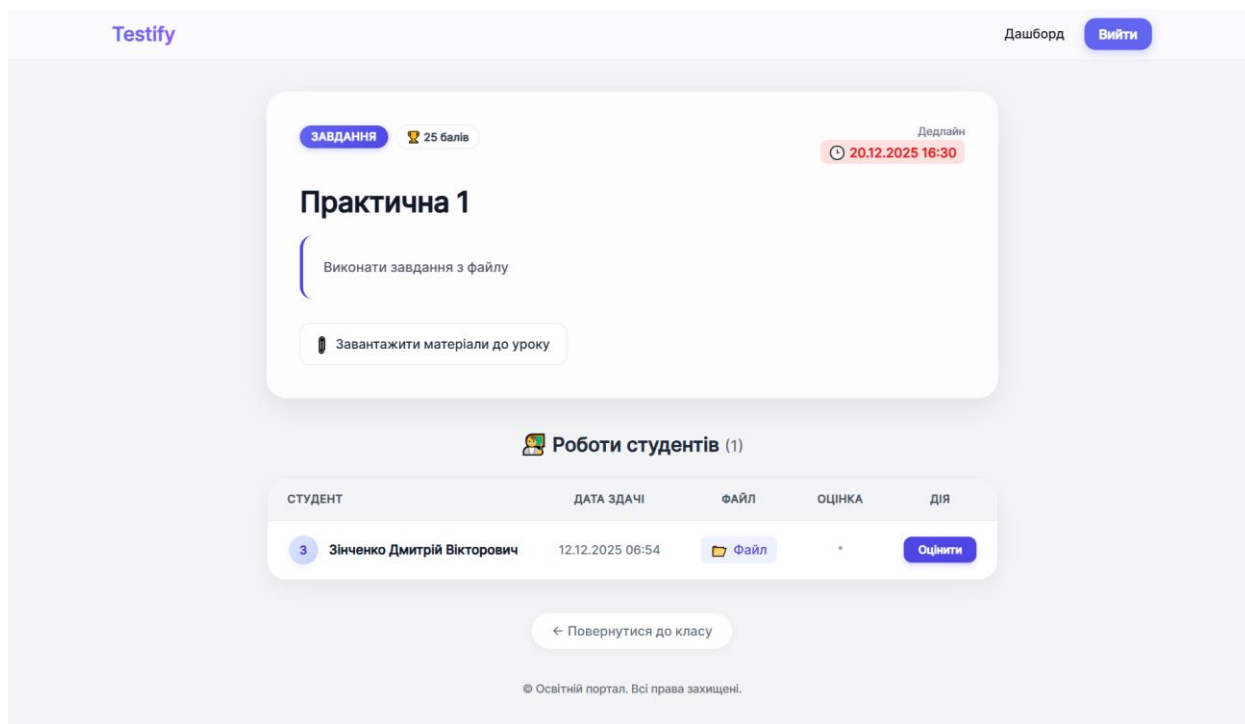


Рисунок 3.20 – Надіслана робота студентом

Далі зображено інтерфейс оцінювання практичної роботи. Картка містить дані студента, час здачі та посилання на файл виконаного завдання. Для перевірки передбачено поля введення балів з урахуванням максимуму та додавання коментаря викладача.

Завершується оцінювання знань студента при виконанні завдань тесту натисканням кнопки Зберегти оцінку в системі (рис. 3.21).

The screenshot shows a web interface for evaluating a student's work. At the top, the 'Testify' logo is on the left, and 'Дашборд' (Dashboard) and 'Вийти' (Logout) buttons are on the right. The main form is titled 'Оцінювання роботи' (Work Evaluation). It includes a student's name 'Зінченко Дмитрій Вікторович' and a submission time 'Здано: 12.12.2025 06:54'. Below this, the task is identified as 'ЗАВДАННЯ Практична 1' (Task Practical 1). A section for the student's file shows a document icon and the text 'Переглянути прикріплений файл' (View attached file). The evaluation section features a dropdown menu for the grade, currently set to '25', with a 'Максимум: 25' (Maximum: 25) label. A text area for the teacher's comment contains the text 'Молодець! Все правильно!' (Well done! Everything is correct!). At the bottom of the form are two buttons: 'Зберегти оцінку' (Save grade) in blue and 'Скасувати' (Cancel) in gray. A footer note at the bottom of the page reads '© Освітній портал. Всі права захищені.' (© Educational portal. All rights reserved.).

Рисунок 3.21 – Інтерфейс оцінювання роботи

Студент після оцінювання може побачити свою оцінку та коментар від викладача (рис. 3.22).

The screenshot shows the 'Ваша відповідь' (Your answer) view. At the top, the 'Testify' logo is on the left, and 'Дашборд' (Dashboard) and 'Вийти' (Logout) buttons are on the right. The main content area is titled 'Ваша відповідь' and features a green confirmation box with a checkmark icon, stating 'Роботу здано!' (Work submitted!) and '12 Dec 2025, 06:54'. Below this, a document icon is next to the filename 'Практична\_1.docx'. The evaluation results are displayed as 'Оцінка: 25 / 25'. A comment box shows the teacher's feedback: 'КОМЕНТАР ВЧИТЕЛЯ: "Молодець! Все правильно!"'. At the bottom, a red warning icon and text state 'Редагування відповіді недоступне після здачі.' (Editing the answer is unavailable after submission.).

Рисунок 3.22 – Перегляд оцінювання завдання

### 3.4 Опис функціонування системи «Testify» та особливостей реалізації алгоритму обчислення результатів тестування

Функціонування системи «TESTIFY» реалізовано засобами Django ORM, що дозволяє представити логічну структуру тесту через взаємопов'язані класи Python. Архітектура БД спроектована для забезпечення цілісності даних та швидкого доступу до ключів перевірки.

Центральним елементом підсистеми є модель Test, яка виступає контейнером для навчальних матеріалів. Зв'язок ForeignKey з моделлю Classroom дозволяє динамічно прив'язувати тести до віртуальних класів. Параметр on\_delete=models.CASCADE гарантує, що при видаленні курсу або викладача (created\_by) всі пов'язані тести автоматично видаляються, що запобігає появі некоректних записів у БД.

Зміст завдань реалізовано через модель Question. Особливістю є денормалізована структура зберігання варіантів відповідей в окремих текстових полях (answer\_1 ... answer\_4). Це рішення спрощує структуру бази та прискорює вибірку питань, усуваючи необхідність складних JOIN-запитів.

Алгоритм обчислення результатів базується на порівнянні цілочисельних індексів. У моделі Question правильна відповідь зберігається як числовий ідентифікатор у полі correct\_answer (IntegerChoices).

Логіка оцінювання на сервері полягає в тому, що система отримує набір пар «ID питання – обраний індекс відповіді», після чого виконується ітерація по об'єктах Question даного тесту.

Алгоритм здійснює пряме порівняння отриманого індексу та значення correct\_answer з БД. Цей підхід забезпечує алгоритмічну складність  $O(1)$  для перевірки одного питання, гарантуючи миттєвий розрахунок результату.

Система підтримує подвійний контур фіксації взаємодії. Модель TestRegistration фіксує початок тесту (registration\_date) та прив'язує студента до завдання для контролю доступу. Модель TestResult зберігає фінальні дані: набраний бал (score) та час завершення (submitted\_at).

## ВИСНОВКИ

У магістерській роботі досліджено актуальну науково-прикладну задачу проєктування та розроблено веб-орієнтовану інформаційну систему дистанційного контролю знань Testify. В ході виконання поставлених завдань було досягнуто основної мети, яка полягала у створенні ефективного, масштабованого та зручного інструменту для автоматизації процесів тестування в навчальних закладах, що є критично важливим в умовах сучасних тенденцій до цифровізації освіти.

За результатами проведеного системного аналізу предметної області було досліджено сучасні платформи дистанційного навчання та виявлено потребу у створенні спеціалізованих легковантажних систем, які фокусуються на простоті інтерфейсу та швидкодії.

Встановлено, що існуючі рішення часто перевантажені надлишковим функціоналом або мають високий поріг входження для користувачів, тому система TESTIFY була спроектована як рішення, що усуває ці недоліки та пропонує інтуїтивний інструментарій для створення та проходження тестів.

Важливим результатом роботи стала розробка алгоритмічного та математичного забезпечення системи. Сформульовано та обґрунтовано математичну модель оцінювання знань, що базується на детермінованому алгоритмі порівняння дискретних ідентифікаторів відповідей.

Запропонований підхід забезпечує об'єктивність оцінювання та високу алгоритмічну швидкість обробки результатів, що дозволяє системі працювати стабільно навіть при великих навантаженнях. Описані алгоритми виключають можливість неоднозначного трактування відповідей та мінімізують вплив людського фактору на процес перевірки робіт.

У рамках проєктування архітектури системи було розроблено структурну та функціональну моделі з використанням засобів UML-моделювання.

Побудована схема БД у третій нормальній формі забезпечує цілісність даних та відсутність їх надлишковості, а обране архітектурне рішення на базі трирівневої моделі дозволило чітко розмежувати логіку обробки даних, інтерфейс користувача та механізми управління базою даних. Це створює надійний фундамент для подальшого масштабування та підтримки програмного комплексу.

Програмна реалізація продукту виконана з використанням мови програмування Python та фреймворку Django, що забезпечило високий рівень інформаційної безпеки, зокрема захист від поширених вебвразливостей.

Реалізовано ключові функціональні модулі, серед яких модуль автентифікації з рольовою моделлю доступу, ядро управління навчальним контентом для створення курсів та підписок, а також модуль тестування з конструктором питань та механізмом автоматичної перевірки.

Надійність зберігання даних забезпечено використанням системи управління базами даних PostgreSQL.

Результати тестування розробленого програмного продукту підтвердили стабільну роботу системи та коректність виконання алгоритмів оцінювання. Інтерфейс користувача, розроблений з урахуванням принципів адаптивності, забезпечує коректне відображення на різних типах пристроїв, що підвищує доступність навчання.

Практична цінність роботи полягає у створенні готового до впровадження вебсервісу, який може бути використаний в освітньому процесі для проведення проміжних та підсумкових контролів знань.

Перспективи подальшого розвитку системи вбачаються у впровадженні інструментів аналітики на базі штучного інтелекту та розширенні спектру типів тестових завдань.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Suleiman A. Cybernetic Based Instruction: an Innovative Learning Model in Digital Age. URL: [https://www.researchgate.net/publication/374577395\\_Cybernetic\\_Based\\_Instruction\\_an\\_Innovative\\_Learning\\_Model\\_in\\_Digital\\_Age](https://www.researchgate.net/publication/374577395_Cybernetic_Based_Instruction_an_Innovative_Learning_Model_in_Digital_Age).
2. Butler-Henderson K., Crawford J. A systematic review of online examinations: A pedagogical innovation for scalable authentication and integrity. URL: [https://www.researchgate.net/publication/344348060\\_A\\_systematic\\_review\\_of\\_online\\_examinations\\_A\\_pedagogical\\_innovation\\_for\\_scalable\\_authentication\\_and\\_integrity](https://www.researchgate.net/publication/344348060_A_systematic_review_of_online_examinations_A_pedagogical_innovation_for_scalable_authentication_and_integrity).
3. M. Guangul F., H. Suhail A., I. Khalit M. Challenges of remote assessment in higher education in the context of COVID-19: a case study of Middle East College. URL: <https://link.springer.com/article/10.1007/s11092-020-09340-w>.
4. Неруцький Н. Академічна доброчесність: виклики дистанційної форми навчання. URL: <https://ur.knute.edu.ua/server/api/core/bitstreams/93f33616-7ac7-4658-97e9-fd5121f0f19c/content>.
5. Кубанов Р. Використання штучного інтелекту у науково-педагогічній діяльності: переваги та недоліки. URL: [https://cuesc.org.ua/images/informlist/advanced\\_training\\_EU\\_%D0%905\\_%D0%94%D0%A0%D0%A3%D0%9A\\_.pdf](https://cuesc.org.ua/images/informlist/advanced_training_EU_%D0%905_%D0%94%D0%A0%D0%A3%D0%9A_.pdf).
6. Bozkurt A., C Sharma R. Education in normal, new normal, and next normal: Observations from the past, insights from the present and projections for the future. URL: [https://www.researchgate.net/publication/347505955\\_Education\\_in\\_normal\\_new\\_normal\\_and\\_next\\_normal\\_Observations\\_from\\_the\\_past\\_insights\\_from\\_the\\_present\\_and\\_projections\\_for\\_the\\_future](https://www.researchgate.net/publication/347505955_Education_in_normal_new_normal_and_next_normal_Observations_from_the_past_insights_from_the_present_and_projections_for_the_future).
7. Hambleton R., Swaminathan H. Item Response Theory. URL: <https://link.springer.com/book/10.1007/978-94-017-1988-9>.
8. Ebenbeck N., Gebhardt M. Differential Performance of Computerized Adaptive Testing in Students With and Without Disabilities -A Simulation Study. URL: [https://www.researchgate.net/publication/378041123\\_Differential\\_Performance\\_of\\_](https://www.researchgate.net/publication/378041123_Differential_Performance_of_)

Computerized\_Adaptive\_Testing\_in\_Students\_With\_and\_Without\_Disabilities\_-  
A\_Simulation\_Study.

9. OECD Digital Education Outlook 2023. URL: [https://www.oecd.org/en/publications/oecd-digital-education-outlook-2023\\_c74f03de-en/full-report/interoperability-unifying-and-maximising-data-reuse-within-digital-education-ecosystems\\_660f8da1.html](https://www.oecd.org/en/publications/oecd-digital-education-outlook-2023_c74f03de-en/full-report/interoperability-unifying-and-maximising-data-reuse-within-digital-education-ecosystems_660f8da1.html).
10. Sciarretta E. Inclusive Education: Is Applying Web Content Accessibility Guidelines to Digital Learning Materials Enough?. URL: [https://www.researchgate.net/publication/391041663\\_Inclusive\\_Education\\_Is\\_Applying\\_Web\\_Content\\_Accessibility\\_Guidelines\\_to\\_Digital\\_Learning\\_Materials\\_Enough](https://www.researchgate.net/publication/391041663_Inclusive_Education_Is_Applying_Web_Content_Accessibility_Guidelines_to_Digital_Learning_Materials_Enough).
11. Palanci A. Learning analytics in distance education: A systematic review study. URL: [https://www.researchgate.net/publication/380573150\\_Learning\\_analytics\\_in\\_distance\\_education\\_A\\_systematic\\_review\\_study](https://www.researchgate.net/publication/380573150_Learning_analytics_in_distance_education_A_systematic_review_study).
12. 4. Christian Maphalala M., Nkosi N. Proctoring Online Assessments: Enhancing Security and Academic Integrity in Open Distance eLearning. URL: [https://www.researchgate.net/publication/397971765\\_Proctoring\\_Online\\_Assessments\\_Enhancing\\_Security\\_and\\_Academic\\_Integrity\\_in\\_Open\\_Distance\\_eLearning](https://www.researchgate.net/publication/397971765_Proctoring_Online_Assessments_Enhancing_Security_and_Academic_Integrity_in_Open_Distance_eLearning).
13. Dicheva D., Dichev C. Gamification in Education: A Systematic Mapping Study. URL: [https://www.researchgate.net/publication/270273830\\_Gamification\\_in\\_Education\\_A\\_Systematic\\_Mapping\\_Study](https://www.researchgate.net/publication/270273830_Gamification_in_Education_A_Systematic_Mapping_Study).
14. Babik D., Gehringer E., Kidd J. A systematic review of educational online peer-review and assessment systems: charting the landscape. URL: <https://link.springer.com/article/10.1007/s11423-024-10349-x>.
15. Turnbull D., Chugh R. Learning Management Systems, An Overview. URL: [https://www.researchgate.net/publication/335463920\\_Learning\\_Management\\_Systems\\_An\\_Overview](https://www.researchgate.net/publication/335463920_Learning_Management_Systems_An_Overview).
16. A. Al-Ajlan, Zedan H. Why Moodle. URL: [https://www.researchgate.net/publication/232615507\\_Why\\_Moodle](https://www.researchgate.net/publication/232615507_Why_Moodle).

17. Mathew Philipose C., Rajagopal S. *Google Classroom as a Learning Management System (LMS) for Teaching English*. URL: [https://www.fortell.org/wp-content/uploads/2020/12/issue-38-fortell\\_jan\\_2019\\_final\\_bw-81-90.pdf](https://www.fortell.org/wp-content/uploads/2020/12/issue-38-fortell_jan_2019_final_bw-81-90.pdf).
18. Instructure Releases National State of Learning and Readiness Report Revealing How Higher Education and Lifelong Learning Can Shape a Future-Proof Workforce. URL: <https://www.prnewswire.com/news-releases/instructure-releases-national-state-of-learning-and-readiness-report-revealing-how-higher-education-and-lifelong-learning-can-shape-a-future-proof-workforce-302596217.html>.
19. 11. Inge Wang A., Tahir R. The effect of using Kahoot! for learning – A literature review. URL: [https://www.researchgate.net/publication/338943272\\_The\\_effect\\_of\\_using\\_Kahoot\\_for\\_learning\\_-\\_A\\_literature\\_review](https://www.researchgate.net/publication/338943272_The_effect_of_using_Kahoot_for_learning_-_A_literature_review).
20. L. I. Zhyvtsova. Distance learning using modern information technologies. URL: [https://www.researchgate.net/publication/367603127\\_DISTANCE\\_LEARNING\\_USING\\_MODERN\\_INFORMATION\\_TECHNOLOGIES](https://www.researchgate.net/publication/367603127_DISTANCE_LEARNING_USING_MODERN_INFORMATION_TECHNOLOGIES).
21. Kaddoura S., Hemanth J. A systematic review on machine learning models for online learning and examination systems. URL: [https://www.researchgate.net/publication/360685889\\_A\\_systematic\\_review\\_on\\_machine\\_learning\\_models\\_for\\_online\\_learning\\_and\\_examination\\_systems](https://www.researchgate.net/publication/360685889_A_systematic_review_on_machine_learning_models_for_online_learning_and_examination_systems).
22. Liu X., Loken E. The Impact of Missing Data on Parameter Estimation: Three Examples in Computerized Adaptive Testing. URL: <https://journals.sagepub.com/doi/10.1177/00131644241306990>.
23. 1. Gamma E., Helm R. Design Patterns: Abstraction and Reuse of Object-Oriented Design. URL: [https://www.researchgate.net/publication/221496095\\_Design\\_Patterns\\_Abstraction\\_and\\_Reuse\\_of\\_Object-Oriented\\_Design](https://www.researchgate.net/publication/221496095_Design_Patterns_Abstraction_and_Reuse_of_Object-Oriented_Design).
24. Dongdong L., Shiliang X. Role - based Access Control in Educational Administration System. URL: [https://www.researchgate.net/publication/321583614\\_Role\\_-\\_based\\_Access\\_Control\\_in\\_Educational\\_Administration\\_System](https://www.researchgate.net/publication/321583614_Role_-_based_Access_Control_in_Educational_Administration_System).
25. Alseelawi N., Th. Salim Alrikabi H. Design and Implementation of an E-learning Platform Using N-Tier Architecture. URL:

- [https://www.researchgate.net/publication/340687289\\_Design\\_and\\_Implementation\\_of\\_an\\_E-learning\\_Platform\\_Using\\_N-Tier\\_Architecture](https://www.researchgate.net/publication/340687289_Design_and_Implementation_of_an_E-learning_Platform_Using_N-Tier_Architecture).
26. Python. URL: <https://uk.wikipedia.org/wiki/Python>.
  27. Що таке Django?. URL: <https://tutorial.djangogirls.org/uk/django/>.
  28. S. Vincent W. DJANGO for PROFESSIONALS. URL: [https://ia800604.us.archive.org/3/items/ebooks\\_202307/djangoforprofessionals.pdf](https://ia800604.us.archive.org/3/items/ebooks_202307/djangoforprofessionals.pdf).
  29. Korkh O. What Are the Main Security Features of Django?. URL: <https://www.planeks.net/django-security-challenges/>.
  30. Ksiazek K. An Overview of MySQL Database Indexing. URL: <https://severalnines.com/blog/overview-mysql-database-indexing/>.
  31. MySQL Workbench. URL: [https://uk.wikipedia.org/wiki/MySQL\\_Workbench](https://uk.wikipedia.org/wiki/MySQL_Workbench).
  32. Bootstrap 5. URL: <https://itvdn.com/ua/video/bootstrap5>.
  33. PyCharm як найкраща IDE для розробки ПЗ на Python. URL: <https://foxminded.ua/pycharm-tse/>.
  34. Інструкція Git для новачків: що це таке, як він працює та які є основні команди. URL: <https://dan-it.com.ua/uk/blog/instrukcija-git-dlja-novachkiv-shho-ce-take-jak-vin-pracjuje-ta-jaki-ie-osnovni-komandi/>.
  35. Що таке Docker і навіщо він?. URL: <https://qagroup.com.ua/publications/shcho-take-docker-i-navishcho-vin/>.
  36. Ткаченко К.О., Зінченко Д.В. Вебзастосунок перевірки знань студентів з підтримкою автоматичних тестів та інтерактивних вправ. *Інформаційні технології в культурі, мистецтві, освіті, науці, економіці та бізнесі*: матеріали Х міжнародної науково-практичної конференції (Київ, 24-25 квітня 2025 р.). Київ: КНУКіМ, 2025. С. 36-38.
  37. Ткаченко О.І., Зінченко Д.В. (2025). Онлайн-тестування – інструмент цифрової освіти: проблеми та перспективи. *IT Sinergy*, № (1;2) (8;9)

## ДОДАТКИ

## ДОДАТОК 1: Вихідний код Forms

```

from django import forms
from .models import Classroom, Assignment, AssignmentSubmission
class ClassroomCreationForm(forms.ModelForm):
    class Meta:
        model = Classroom
        fields = ['name', 'description']
        widgets = {
            'name': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Наприклад: Математика 9 "А"'}) ,
            'description': forms.Textarea(
                attrs={'class': 'form-control', 'rows': 4, 'placeholder': 'Короткий опис класу'}),
        }
        labels = {
            'name': 'Назва класу',
            'description': 'Опис класу (необов\язково)',
        }
class JoinClassroomForm(forms.Form):
    join_code = forms.CharField(
        max_length=8,
        label="Код класу",
        widget=forms.TextInput(attrs={'placeholder': 'Введіть код класу (8 символів)'})
    )
class AssignmentCreationForm(forms.ModelForm):
    class Meta:
        model = Assignment
        fields = ['title', 'description', 'file', 'max_points', 'due_date']
        widgets = {
            'title': forms.TextInput(attrs={'class': 'form-control'}),
            'description': forms.Textarea(attrs={'class': 'form-control', 'rows': 3}),
            'file': forms.FileInput(attrs={'class': 'form-control'}),
            'max_points': forms.NumberInput(attrs={'class': 'form-control'}),
            'due_date': forms.DateTimeInput(attrs={'class': 'form-control', 'type': 'datetime-local'}),
        }
        labels = {
            'title': 'Назва',
            'description': 'Опис',
            'file': 'Прикріпити файл (необов\язково)',
            'max_points': 'Бали',
            'due_date': 'Дедлайн',
        }
# Форма для студента
class AssignmentSubmissionForm(forms.ModelForm):

```

```

class Meta:
    model = AssignmentSubmission
    fields = ['file']
    widgets = {
        'file': forms.FileInput(attrs={'class': 'form-
control'})
    }
# Форма для вчителя
class GradingForm(forms.ModelForm):
    class Meta:
        model = AssignmentSubmission
        fields = ['grade', 'teacher_feedback']
        widgets = {
            'grade': forms.NumberInput(attrs={'class': 'form-
control'}),
            'teacher_feedback': forms.Textarea(attrs={'class':
'form-control', 'rows': 3}),
        }

```

## ДОДАТОК 2: Вихідний код Models

```

import os
import uuid
from django.db import models
from django.conf import settings # Використовуємо стандартний
спосіб отримання моделі користувача
def generate_unique_code():
    # Генеруємо унікальний 6-символьний код
    code = str(uuid.uuid4()).replace('-', '')[:6].upper()
    # Перевіряємо, чи є такий код вже в базі (для гарантії
унікальності)
    # ПРИМІТКА: Для коректного імпорту необхідно перенести цю
функцію в окремий файл utils.py
    # або імпортувати Classroom всередині функції
    from .models import Classroom # Тимчасовий імпорт для
прикладу
    if Classroom.objects.filter(join_code=code).exists():
        return generate_unique_code() # Якщо є, генеруємо знову
    return code
class Classroom(models.Model):
    """Модель для представлення класу (навчальної кімнати)."""
    # Посилання на користувача, який є вчителем і створив цей
клас
    teacher = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
        related_name='created_classrooms',
        verbose_name='Вчитель'
    )
    name = models.CharField(
        max_length=100,
        verbose_name='Назва класу'
    )
    description = models.TextField(
        blank=True,
        verbose_name='Опис'
    )

```

```

# Унікальний код для приєднання до класу (генерується при
збереженні)
join_code = models.CharField(
    max_length=8,
    unique=True,
    default=generate_unique_code, # Використовуємо функцію
для заповнення за замовчуванням
    editable=False, # Не дозволяємо вчителю змінювати його
напрямую
    verbose_name="Код для приєднання" )
students = models.ManyToManyField(
    settings.AUTH_USER_MODEL,
    related_name='joined_classrooms',
    blank=True,
    verbose_name="Учні" )
created_at = models.DateTimeField(
    auto_now_add=True,
    verbose_name='Дата створення' )
def save(self, *args, **kwargs):
    """Перевизначаємо save() для генерації join_code, якщо
він порожній."""
    if not self.join_code:
        # Імпортуємо функцію для генерації коду тут, щоб
унікнути циклічних імпортів
        # Якщо generate_unique_code знаходиться в цьому ж
файлі, цей рядок можна видалити:
        # from .utils import generate_unique_join_code
        self.join_code = generate_unique_code() # Викликаємо
функцію, якщо join_code не заданий
    super().save(*args, **kwargs)
def __str__(self):
    return f"{self.name} ({self.teacher.username})"
class Meta:
    verbose_name = 'Клас (кімната)'
    verbose_name_plural = 'Класи (кімнати)'
class Assignment(models.Model):
    """Модель завдання всередині класу."""
    classroom = models.ForeignKey(
        Classroom,
        on_delete=models.CASCADE,
        related_name='assignments',
        verbose_name="Клас" )
    title = models.CharField(max_length=200, verbose_name="Назва
завдання")
    description = models.TextField(verbose_name="Опис завдання")
    # Поле для файлу. upload_to вказує папку всередині
MEDIA_ROOT
    file = models.FileField(
        upload_to='assignments/',
        blank=True,
        null=True,
        verbose_name="Файл завдання" )

```

```

    max_points =
models.PositiveIntegerField(verbose_name="Максимальна кількість
балів")
    due_date = models.DateTimeField(verbose_name="Дата здачі")
    created_at = models.DateTimeField(auto_now_add=True)
    def __str__(self):
        return self.title
    def filename(self):
        return os.path.basename(self.file.name)
    class Meta:
        verbose_name = 'Завдання'
        verbose_name_plural = 'Завдання'
class AssignmentSubmission(models.Model):
    """Модель виконаного завдання учнем."""
    assignment = models.ForeignKey(
        Assignment,
        on_delete=models.CASCADE,
        related_name='submissions',
        verbose_name='Завдання' )
    student = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
        related_name='submissions',
        verbose_name='Учень' )
    file = models.FileField(
        upload_to='submissions/',
        verbose_name="Файл відповіді" )
    submitted_at = models.DateTimeField(
        auto_now_add=True,
        verbose_name='Дата здачі' )
    # Оцінювання
    grade = models.PositiveIntegerField(null=True, blank=True,
verbose_name="Оцінка")
    teacher_feedback = models.TextField(blank=True,
verbose_name="Коментар вчителя")
    class Meta:
        # Один учень може здати одне завдання тільки один раз
        (або перезаписати)
        unique_together = ('assignment', 'student')
        verbose_name = 'Надіслане завдання'
        verbose_name_plural = 'Надіслані завдання'
    def __str__(self):
        return f"{self.student} - {self.assignment.title}"
    def filename(self):
        return os.path.basename(self.file.name)

```

### ДОДАТОК 3: Вихідний код Views

```

from django.shortcuts import render, redirect, get_object_or_404
from django.views.generic import CreateView, ListView,
DetailView
from django.contrib import messages # Для виведення повідомлень
користувачеві

```

```

from django.views import View # Імпортуємо базовий клас View
from django.urls import reverse
from django.contrib.auth.mixins import LoginRequiredMixin
from django.db.models import Q
from django.contrib.auth.decorators import login_required
from django.http import HttpResponseRedirect # Для обмеження
доступу
from .models import Classroom, Assignment, AssignmentSubmission
from .forms import ClassroomCreationForm, JoinClassroomForm,
AssignmentCreationForm, AssignmentSubmissionForm, GradingForm
# Припустимо, що у вас є функція чи метод для перевірки ролі
"Teacher"
# class TeacherRequiredMixin(LoginRequiredMixin):
#     def dispatch(self, request, *args, **kwargs):
#         if not request.user.is_teacher: # Замініть на вашу
логіку перевірки ролі
#             return redirect('some_unauthorized_url')
#         return super().dispatch(request, *args, **kwargs)
class ClassroomCreateView(LoginRequiredMixin, CreateView):
    """Представлення для створення нового класу."""
    model = Classroom
    form_class = ClassroomCreationForm
    template_name = 'classroom/classroom_create.html'
    def form_valid(self, form):
        """Автоматично встановлюємо поточного користувача як
вчителя."""
        form.instance.teacher = self.request.user
        return super().form_valid(form)
    def get_success_url(self):
        """
        Перенаправляємо на сторінку детального перегляду
створеного класу.
        self.object доступний лише після успішного виконання
form_valid.
        """
        return reverse('classroom_detail', kwargs={'pk':
self.object.pk})
class ClassroomDetailView(LoginRequiredMixin, DetailView):
    """Представлення для відображення детальної інформації про
один клас."""
    model = Classroom
    template_name = 'classroom/classroom_detail.html'
    context_object_name = 'classroom'
    def get_queryset(self):
        user = self.request.user
        return Classroom.objects.filter(
            Q(teacher=user) | Q(students=user)
        ).distinct()
class ClassroomJoinView(LoginRequiredMixin, View):
    """
    Дозволяє авторизованому учневі приєднатися до класу за
кодом.
    """

```

```

def get(self, request):
    form = JoinClassroomForm()
    return render(request, 'classroom/join.html', {'form':
form})
def post(self, request):
    form = JoinClassroomForm(request.POST)
    if form.is_valid():
        code = form.cleaned_data['join_code'].upper() #
Коди зберігаються у верхньому регістрі
        user = request.user
        try:
            classroom =
Classroom.objects.get(join_code=code)
        except Classroom.DoesNotExist:
            messages.error(request, 'Клас із таким кодом не
знайдено. Перевірте код і спробуйте знову.')
            return render(request, 'classroom/join.html',
{'form': form})
            # Перевірка: чи є користувач вчителем цього класу?
            if classroom.teacher == user:
                messages.warning(request, 'Ви є вчителем цього
класу і не можете приєднатися як учень.')
                return redirect('classroom_detail',
pk=classroom.pk)
            # Перевірка: чи вже учень у класі?
            if classroom.students.filter(pk=user.pk).exists():
                messages.info(request, f'Ви вже є учасником
класу "{classroom.name}"')
                return redirect('classroom_detail',
pk=classroom.pk)
            # Додавання учня до класу
            classroom.students.add(user)
            messages.success(request, f'Ви успішно приєдналися
до класу "{classroom.name}"!')
            return redirect('classroom_detail', pk=classroom.pk)
            # Якщо форма невалідна (наприклад, код занадто
довгий/короткий)
            return render(request, 'classroom/join.html', {'form':
form})
@login_required
def create_assignment(request, classroom_id):
    classroom = get_object_or_404(Classroom, pk=classroom_id)
    # Перевірка: тільки вчитель цього класу може створювати
завдання
    if request.user != classroom.teacher:
        return HttpResponseForbidden("Тільки вчитель може
створювати завдання.")
    if request.method == 'POST':
        # ВАЖЛИВО: request.FILES потрібен для обробки
завантажених файлів
        form = AssignmentCreationForm(request.POST,
request.FILES)
        if form.is_valid():

```

```

        assignment = form.save(commit=False)
        assignment.classroom = classroom
        assignment.save()
        # Після створення повертаємо вчителя назад до класу
        return redirect('classroom_detail', pk=classroom.id)
    else:
        form = AssignmentCreationForm()
        return render(request, 'classroom/create_assignment.html', {
            'form': form,
            'classroom': classroom
        })
@login_required
def assignment_detail(request, pk):
    assignment = get_object_or_404(Assignment, pk=pk)
    classroom = assignment.classroom
    user = request.user
    # 1. Визначаємо полі
    is_teacher = (user == classroom.teacher)
    is_student = classroom.students.filter(pk=user.pk).exists()
    # 2. Перевірка доступу
    if not (is_teacher or is_student):
        return HttpResponseForbidden("Ви не учасник цього
класу.")
    # Змінні для контексту
    submission = None
    submission_form = None
    all_submissions = None
    grading_form = None # Додаємо для потенційного використання
у шаблоні
    # --- ЛОГІКА ДЛЯ СТУДЕНТА ---
    if is_student:
        submission =
AssignmentSubmission.objects.filter(assignment=assignment,
student=user).first()
        if request.method == 'POST':
            # Якщо робота вже є - запобігаємо повторній задачі,
якщо це не передбачено (у цьому коді забороняємо оновлення)
            if submission:
                messages.warning(request, "Ви вже здали цю
роботу.")
                return redirect('assignment_detail', pk=pk)
            submission_form =
AssignmentSubmissionForm(request.POST, request.FILES)
            if submission_form.is_valid():
                sub = submission_form.save(commit=False)
                sub.assignment = assignment
                sub.student = user
                sub.save()
                messages.success(request, "Роботу успішно
здано!")
            return redirect('assignment_detail', pk=pk)
        else:
            submission_form = AssignmentSubmissionForm()
    # --- ЛОГІКА ДЛЯ ВЧИТЕЛЯ ---

```

```

    if is_teacher:
        # Отримуємо список усіх робіт для таблиці
        all_submissions =
assignment.submissions.all().select_related('student')
        # Для вчителя форма для здачі роботи не потрібна, але
        потрібна форма для оцінювання
        grading_form = GradingForm() # Можна використовувати для
        inline-оцінювання, якщо потрібно
        return render(request, 'classroom/assignment_detail.html', {
            'assignment': assignment,
            'is_teacher': is_teacher,
            'is_student': is_student,
            'submission': submission,
            'submission_form': submission_form,
            'all_submissions': all_submissions,
            'grading_form': grading_form,    })
@login_required
def grade_submission(request, submission_id):
    """Сторінка для вчителя: оцінювання конкретної роботи."""
    submission = get_object_or_404(AssignmentSubmission,
pk=submission_id)
    # Перевірка прав
    if request.user != submission.assignment.classroom.teacher:
        return HttpResponseForbidden("Тільки вчитель може
оцінювати.")
    if request.method == 'POST':
        form = GradingForm(request.POST, instance=submission)
        if form.is_valid():
            form.save()
            messages.success(request, f"Оцінку для
{submission.student} збережено!")
            # Перенаправляємо назад на деталі завдання, оскільки
            таблиця робіт знаходиться там.
            return redirect('assignment_detail',
pk=submission.assignment.id)
        else:
            form = GradingForm(instance=submission)
            return render(request, 'classroom/grade_submission.html', {
                'form': form,
                'submission': submission    })

```

Додаток 4: Вихідний код assignment\_detail

```

{% extends 'base.html' %}
{% block title %}{{ assignment.title }}{% endblock %}
{% block content %}
<style>
    /* Сучасна стилізація для інпуту файлу */
    .file-input-wrapper input[type="file"] {
        width: 100%;
        padding: 10px;
        background: #f9fafb;
        border: 1px solid #e5e7eb;
        border-radius: 8px;
        cursor: pointer;
    }

```

```

        color: #4b5563;    }
    .file-input-wrapper input[type="file"]::file-selector-button
    {
        margin-right: 15px;
        background-color: #4f46e5;
        color: white;
        border: none;
        padding: 8px 16px;
        border-radius: 6px;
        cursor: pointer;
        transition: background .2s;    }
    .file-input-wrapper input[type="file"]::file-selector-
button:hover {
        background-color: #4338ca;    }
    /* Ефект наведення для карток */
    .glass-card {
        transition: transform 0.2s ease, box-shadow 0.2s ease; }
    .glass-card:hover {
        box-shadow: 0 12px 40px rgba(0,0,0,0.08) !important; }
    /* Адаптивність для відступів */
    @media (max-width: 768px) {
        .dashboard-container {
            padding-left: 15px !important;
            padding-right: 15px !important;
        }
        .glass-card {
            padding: 25px !important; /* Менші відступи
всередині картки на мобільному */        }    }
</style>
<div class="container dashboard-container" style="max-width:
900px; margin: 0 auto; padding: 0 20px;">
    <div class="glass-card" style="
        background: rgba(255,255,255,0.8);
        backdrop-filter: blur(24px);
        border: 1px solid rgba(255,255,255,0.6);
        padding: 40px;
        border-radius: 24px;
        box-shadow: 0 4px 24px rgba(0,0,0,0.04);
        margin-bottom: 30px;
        box-sizing: border-box; /* Гарантує, що padding не
розширює блок */
    ">
        <div style="display: flex; justify-content: space-
between; align-items: flex-start; margin-bottom: 25px; flex-
wrap: wrap; gap: 10px;">
            <div style="display: flex; gap: 10px; align-items:
center;">
                <span style="
                    background: linear-gradient(135deg, #6366f1
0%, #4f46e5 100%);
                    color: white; padding: 6px 14px; border-
radius: 30px;

```

```

        font-size: 0.85em; font-weight: 600; letter-
spacing: 0.5px;
        box-shadow: 0 2px 8px rgba(99, 102, 241,
0.3);">
        ЗАВДАННЯ
    </span>
    <span style="
        background: #fff; border: 1px solid #e5e7eb;
color: #374151;
        padding: 5px 12px; border-radius: 30px;
font-size: 0.85em; font-weight: 600;">
        🏆 {{ assignment.max_points }} балів
    </span>
</div>
<div style="text-align: right;">
    <small style="color: #6b7280; display: block;
margin-bottom: 2px;">Дедлайн</small>
    <div style="color: #dc2626; font-weight: 700;
background: #fee2e2; padding: 4px 10px; border-radius: 8px;
display: inline-block;">
        🕒 {{ assignment.due_date|date:"d.m.Y H:i"
}}
    </div>
</div>
</div>
<h1 style="margin: 0 0 20px 0; font-size: 2.2rem; color:
#111827; font-weight: 800; letter-spacing: -0.5px; word-wrap:
break-word;">
    {{ assignment.title }}
</h1>
<div style="
    font-size: 1.05rem; line-height: 1.7; color:
#4b5563;
    background: rgba(255,255,255,0.5); padding: 25px;
border-radius: 16px;
    border-left: 4px solid #4f46e5; margin-bottom: 30px;
    white-space: pre-wrap; word-wrap: break-word;
">{{ assignment.description }}</div>
{% if assignment.file %}
    <div style="margin-top: 20px;">
        <a href="{{ assignment.file.url }}"
target="_blank" style="
            display: inline-flex; align-items: center;
gap: 10px;
            text-decoration: none; background: #fff;
border: 1px solid #e5e7eb;
            padding: 12px 20px; border-radius: 12px;
color: #1f2937; font-weight: 500;
            transition: all 0.2s;
            " onmouseover="this.style.borderColor='#4f46e5';
this.style.color='#4f46e5';"
            onmouseout="this.style.borderColor='#e5e7eb';
this.style.color='#1f2937';">

```

```

        <span style="font-size: 1.2em;"><img alt="document icon" data-bbox="771 68 788 85"/></span>
Завантажити матеріали до завдання
    </a>
</div>
{% endif %}
</div>
{% if not is_teacher %}
<div class="glass-card" style="
    background: #ffffff;
    border: 1px solid #e5e7eb;
    border-radius: 24px;
    padding: 40px;
    text-align: center;
    position: relative;
    overflow: hidden;
    box-shadow: 0 10px 30px rgba(0,0,0,0.03);
    box-sizing: border-box;
">
    <div style="position: absolute; top: 0; left: 0; width:
100%; height: 6px; background: linear-gradient(90deg, #10b981,
#3b82f6);"></div>
    <h3 style="color: #1f2937; font-weight: 700; margin-
bottom: 25px;"><img alt="lock icon" data-bbox="325 431 342 448"/> Ваша відповідь</h3>
    {% if submission %}
        <div style="
            background: #ecfdf5; border: 1px solid #10b981;
            padding: 25px; border-radius: 16px; display:
inline-block;
            margin-bottom: 10px; text-align: left; width:
100%; box-sizing: border-box;">
            <div style="display: flex; align-items: center;
gap: 15px; margin-bottom: 15px;">
                <div style="width: 40px; height: 40px;
background: #10b981; border-radius: 50%; display: flex; align-
items: center; justify-content: center; color: white; font-size:
1.2em; flex-shrink: 0;"><img alt="checkmark icon" data-bbox="438 641 455 658"/></div>
                <div>
                    <strong style="color: #064e3b; display:
block; font-size: 1.1em;">Роботу здано!</strong>
                    <small style="color: #059669;">{{
submission.submitted_at|date:"d M Y, H:i" }}</small>
                </div>
            </div>
            <div style="background: white; padding: 15px;
border-radius: 12px; border: 1px solid #a7f3d0; display: flex;
align-items: center; justify-content: space-between; gap:
10px;">
                <div style="display: flex; align-items:
center; gap: 10px; overflow: hidden;">
                    <span style="font-size:
1.5em;"><img alt="document icon" data-bbox="241 888 258 905"/></span>

```

```

        <a href="{{ submission.file.url }}"
target="_blank" style="color: #059669; font-weight: 600; text-
decoration: none; text-overflow: ellipsis; white-space: nowrap;
overflow: hidden; display: block;">
            {{ submission.filename }}
        </a>
    </div>
    <a href="{{ submission.file.url }}" download
class="btn btn-sm btn-outline-success" style="border-radius:
50%; width: 32px; height: 32px; padding: 0; display: flex;
align-items: center; justify-content: center; flex-shrink:
0;">↓</a>
</div>
    {% if submission.grade is not None %}
        <hr style="margin: 20px 0; border-color:
rgba(0,0,0,0.1);">
        <div style="display: flex; justify-content:
space-between; align-items: center;">
            <span style="font-size: 1.2em; font-
weight: bold; color: #1f2937;">Оцінка:</span>
            <span style="font-size: 1.4em; font-
weight: bold; color: #059669;">{{ submission.grade }} / {{
assignment.max_points }}</span>
        </div>
        {% if submission.teacher_feedback %}
            <div style="margin-top: 15px;
background: #f3f4f6; padding: 15px; border-radius: 12px;">
                <small style="color: #6b7280; text-
transform: uppercase; font-size: 0.75em; font-weight:
bold;">Коментар вчителя:</small>
                <p style="margin: 5px 0; font-style:
italic; color: #374151;">"{{ submission.teacher_feedback }}"</p>
            </div>
            {% endif %}
        {% else %}
            <div style="margin-top: 15px; padding-top:
15px; border-top: 1px dashed #a7f3d0; color: #6b7280; font-size:
0.9em; text-align: center;">
                ⌚ Очікує перевірки вчителем
            </div>
            {% endif %}
        </div>
        <p style="color: #ef4444; font-size: 0.9em; margin-
top: 15px;">
            🔒 Редагування відповіді недоступне після здачі.
        </p>
    {% else %}
        <div style="margin-bottom: 25px; color: #6b7280;">
            <p>Завантажте файл з виконаним завданням.
<br><strong>Увага:</strong> після відправки змінити файл буде
неможливо.</p>
        </div>

```

```

        <form method="post" enctype="multipart/form-data"
style="max-width: 450px; margin: 0 auto; width: 100%;">
            {% csrf_token %}
            <div class="file-input-wrapper" style="margin-
bottom: 20px;">
                {{ submission_form.file }}
            </div>
            <button type="submit" style="
                background: linear-gradient(135deg, #10b981
0%, #059669 100%);
                color: white; border: none; padding: 12px
40px; border-radius: 12px;
                font-weight: 600; font-size: 1rem; cursor:
pointer;
                box-shadow: 0 4px 15px rgba(16, 185, 129,
0.4);
                transition: transform 0.2s; width: 100%;
            ">
onmouseover="this.style.transform='translateY(-2px)'"
onmouseout="this.style.transform='translateY(0)'">
                 Здати роботу
            </button>
        </form>
    {% endif %}
</div>
{% endif %}
{% if is_teacher %}
    <div style="margin-top: 40px;">
        <h2 style="text-align: center; margin-bottom: 25px;
color: #1f2937; font-weight: 800;">
             Роботи учнів <span style="font-weight: 400;
color: #6b7280; font-size: 0.8em;">({{ all_submissions|length
}})</span>
        </h2>
        <div class="glass-card" style="
            padding: 0;
            overflow: hidden;
            border-radius: 24px;
            background: white;
            border: 1px solid #e5e7eb;
            box-shadow: 0 10px 30px rgba(0,0,0,0.05);
        ">
            <div style="overflow-x: auto;">
                <table style="width: 100%; border-collapse:
collapse; min-width: 600px;">
                    <thead>
                        <tr style="background: #f3f4f6;
border-bottom: 1px solid #e5e7eb;">
                            <th style="padding: 18px 25px;
color: #4b5563; font-weight: 700; font-size: 0.85em; text-
transform: uppercase; letter-spacing: 0.5px; text-align:
left;">Учень</th>

```

```

<th style="padding: 18px 25px;
color: #4b5563; font-weight: 700; font-size: 0.85em; text-
transform: uppercase; letter-spacing: 0.5px; text-align:
center;">Дата здачі</th>

<th style="padding: 18px 25px;
color: #4b5563; font-weight: 700; font-size: 0.85em; text-
transform: uppercase; letter-spacing: 0.5px; text-align:
center;">Файл</th>

<th style="padding: 18px 25px;
color: #4b5563; font-weight: 700; font-size: 0.85em; text-
transform: uppercase; letter-spacing: 0.5px; text-align:
center;">Оцінка</th>

<th style="padding: 18px 25px;
color: #4b5563; font-weight: 700; font-size: 0.85em; text-
transform: uppercase; letter-spacing: 0.5px; text-align:
center;">Дія</th>

</tr>
</thead>
<tbody>
{% for sub in all_submissions %}
    <tr style="border-bottom: 1px
solid #f3f4f6; transition: background 0.2s;"
onmouseover="this.style.background='#f9fafb'"
onmouseout="this.style.background='white'">
        <td style="padding: 16px
25px; color: #111827; font-weight: 600; text-align: left;">
            <div style="display:
flex; align-items: center; gap: 12px;">
                <div style="
                    width: 36px;
height: 36px;
                    background:
linear-gradient(135deg, #e0e7ff 0%, #c7d2fe 100%);
                    color: #4338ca;
border-radius: 50%;
                    display: flex;
align-items: center; justify-content: center;
                    font-size:
0.9em; font-weight: bold;
                    border: 2px
solid white; box-shadow: 0 2px 5px rgba(0,0,0,0.05);
                ">
                    {{
sub.student.full_name|first|default:sub.student.email|first|uppe
r }}
                </div>
                {{
sub.student.full_name|default:sub.student.email }}
                </div>
            </td>
            <td style="padding: 16px
25px; color: #6b7280; font-size: 0.95em; text-align: center;">

```

```

sub.submitted_at|date:"d.m.Y H:i" }}
    </td>
    <td style="padding: 16px
25px; text-align: center;">
        <a href="{{ sub.file.url
}} " target="_blank" style="
            color: #4f46e5;
            display: inline-
            flex; align-items: center; gap: 6px;
            padding: 6px 12px;
            background: #eef2ff; border-radius: 8px;
            transition:
            background 0.2s;
            "
onmouseover="this.style.background='#e0e7ff'"
onmouseout="this.style.background='#eef2ff'">
             <span
style="">Файл</span>
        </a>
    </td>
    <td style="padding: 16px
25px; text-align: center;">
        {% if sub.grade is not
None %}
            <span style="
                background:
                #ecfdf5; color: #065f46;
                padding: 6px
                12px; border-radius: 20px;
                font-weight:
                700; font-size: 0.9em;
                border: 1px
                solid #a7f3d0;
                ">
                {{ sub.grade }}
            </span>
        {% else %}
            <span style="color:
            #9ca3af; font-size: 2em; line-height: 0.5;">.</span>
        {% endif %}
    </td>
    <td style="padding: 16px
25px; text-align: center;">
        <a href="{% url
'grade_submission' sub.id %}" class="btn btn-sm" style="
            background: #4f46e5;
            color: white;
            padding: 8px 18px;
            border-radius: 10px;

```

```

text-decoration:
none; font-size: 0.85em; font-weight: 600;
6px rgba(79, 70, 229, 0.2);
transition:
display: inline-
block;
"
onmouseover="this.style.transform='translateY(-2px)'"
onmouseout="this.style.transform='translateY(0)'">
Оцінити
</a>
</td>
</tr>
{% empty %}
<tr>
<td colspan="5"
style="padding: 60px; text-align: center; color: #9ca3af;">
<div style="font-size:
3em; margin-bottom: 10px;"></div>
<p>Ще ніхто не здав
роботи.</p>
</td>
</tr>
{% endfor %}
</tbody>
</table>
</div>
</div>
{% endif %}
<div style="margin-top: 40px; text-align: center;">
<a href="{% url 'classroom_detail'
assignment.classroom.id %}" style="
color: #6b7280; text-decoration: none; font-weight:
500;
padding: 12px 24px; background: white; border-
radius: 50px;
box-shadow: 0 2px 10px rgba(0,0,0,0.05); border: 1px
solid #f3f4f6;
transition: all 0.2s;
display: inline-block;
" onmouseover="this.style.boxShadow='0 4px 15px
rgba(0,0,0,0.1)'"
onmouseout="this.style.boxShadow='0 2px 10px
rgba(0,0,0,0.05)'">
← Повернутися до класу
</a>
</div>
</div>
{% endblock %}

```

## ДОДАТОК 5: Вихідний код classroom\_create

```

{% extends "base.html" %}
{% block title %}Створити новий клас{% endblock %}
{% block content %}
<style>
  /* Центрування */
  .dashboard-container {
    max-width: 600px;
    margin: 0 auto;
    padding: 0 20px;    }
  /* Скляна картка */
  .glass-card {
    background: rgba(255, 255, 255, 0.85);
    backdrop-filter: blur(24px);
    border: 1px solid rgba(255, 255, 255, 0.6);
    padding: 40px;
    border-radius: 24px;
    box-shadow: 0 10px 40px rgba(0,0,0,0.05);
    position: relative;
    overflow: hidden;    }
  /* Заголовки */
  h2 {
    text-align: center;
    margin-top: 0;
    margin-bottom: 10px;
    color: #1f2937;
    font-weight: 800;
    font-size: 1.8rem;
    position: relative;
    z-index: 2;    }
  p.subtitle {
    text-align: center;
    color: #6b7280;
    margin-bottom: 30px;
    position: relative;
    z-index: 2;    }
  /* Стилiзацiя полiв форми */
  form p {
    margin-bottom: 20px;    }
  form label {
    display: block;
    margin-bottom: 8px;
    font-weight: 600;
    color: #374151;
    font-size: 0.95rem;    }
  form input[type="text"],
  form textarea {
    width: 100%;
    padding: 12px 16px;
    background: #f9fafb;
    border: 1px solid #e5e7eb;
    border-radius: 12px;

```

```

        font-size: 1rem;
        color: #1f2937;
        transition: all 0.2s;
        box-sizing: border-box;
        font-family: inherit;    }
form input:focus,
form textarea:focus {
    outline: none;
    border-color: #4f46e5;
    box-shadow: 0 0 0 4px rgba(79, 70, 229, 0.1);
    background: white;    }
/* Головна кнопка */
.btn-create {
    background: linear-gradient(135deg, #4f46e5 0%, #4338ca
100%);
    color: white;
    border: none;
    padding: 14px 30px;
    border-radius: 12px;
    font-weight: 600;
    font-size: 1rem;
    cursor: pointer;
    width: 100%;
    margin-top: 20px;
    box-shadow: 0 4px 15px rgba(79, 70, 229, 0.3);
    transition: transform 0.2s;
    display: block;    }
.btn-create:hover {
    transform: translateY(-2px);    }
/* Кнопка скасування */
.btn-cancel {
    display: block;
    text-align: center;
    margin-top: 20px;
    color: #6b7280;
    text-decoration: none;
    font-weight: 500;
    transition: color 0.2s;
    position: relative;
    z-index: 2;    }
.btn-cancel:hover {
    color: #111827;    }
</style>
<div class="dashboard-container">
    <div class="glass-card">
        <div style="position: absolute; top: -80px; right: -
80px; width: 200px; height: 200px; background: radial-
gradient(circle, rgba(99,102,241,0.15) 0%, rgba(255,255,255,0)
70%); border-radius: 50%;"></div>
        <div style="position: absolute; bottom: -60px; left: -
60px; width: 150px; height: 150px; background: radial-
gradient(circle, rgba(168,85,247,0.1) 0%, rgba(255,255,255,0)
70%); border-radius: 50%;"></div>

```

```

    <h2>Створення класу</h2>
    <p class="subtitle">Заповніть форму, щоб створити новий
простір для навчання.</p>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit" class="btn-create">Створити
клас</button>
        <a href="{% url 'dashboard' %}" class="btn-
cancel">Скасувати</a>
    </form>
</div>
</div>
{% endblock %}

```

## ДОДАТОК 6: Вихідний код classroom\_detail

```

{% extends "base.html" %}
{% block title %}Клас: {{ classroom.name }}{% endblock %}
{% block content %}
<style>
    /* Глобальні стилі для цієї сторінки */
    .dashboard-container {
        max-width: 1000px;
        margin: 0 auto;
        padding: 0 20px;
        box-sizing: border-box; /* Додано для безпеки */    }
    /* Картка елемента (тест або завдання) */
    .item-card {
        background: rgba(255, 255, 255, 0.8);
        backdrop-filter: blur(12px);
        border: 1px solid #e5e7eb;
        border-radius: 16px;
        padding: 24px;
        transition: all 0.3s cubic-bezier(0.4, 0, 0.2, 1);
        display: flex;
        justify-content: space-between;
        align-items: center;
        position: relative;
        overflow: hidden;
        box-sizing: border-box; /* Гарантує, що padding не
розширює блок */    }
    .item-card:hover {
        transform: translateY(-4px);
        box-shadow: 0 12px 24px rgba(0, 0, 0, 0.06);
        border-color: #6366f1;    }
    .item-card::before {
        content: '';
        position: absolute;
        left: 0;
        top: 0;
        bottom: 0;
        width: 4px;

```

```

        background: #6366f1;
        opacity: 0;
        transition: opacity 0.3s;    }
.item-card:hover::before {
    opacity: 1;    }
/* Стили для бейджів */
.badge {
    padding: 6px 12px;
    border-radius: 8px;
    font-size: 0.85em;
    font-weight: 600;
    display: inline-block;    }
.badge-gray { background: #f3f4f6; color: #4b5563; }
.badge-red { background: #fee2e2; color: #991b1b; }
.badge-blue { background: #eff6ff; color: #1e40af; }
.badge-green { background: #d1fae5; color: #065f46; }
/* Заголовки секцій */
.section-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 20px;
    margin-top: 40px;    }
.section-title {
    font-size: 1.5rem;
    font-weight: 700;
    color: #1f2937;
    display: flex;
    align-items: center;
    gap: 10px;    }
/* Кнопка дії */
.btn-action {
    text-decoration: none;
    padding: 8px 16px;
    border-radius: 10px;
    font-weight: 500;
    transition: all 0.2s;
    border: 1px solid transparent;
    white-space: nowrap;    }
.btn-action-primary {
    background: white;
    color: #4f46e5;
    border-color: #e5e7eb;    }
.btn-action-primary:hover {
    background: #4f46e5;
    color: white;
    border-color: #4f46e5;    }
/* Пустий стан (ВИПРАВЛЕНО) */
.empty-state {
    text-align: center;
    padding: 40px;
    background: rgba(255,255,255,0.4);
    border: 2px dashed #e5e7eb;

```

```

border-radius: 16px;
color: #9ca3af;
width: 100%;
box-sizing: border-box; /* !!! Це виправляє вихід за
межі !!! */
</style>
<div class="dashboard-container">
  <div class="welcome-card" style="
background: rgba(255,255,255,0.85);
backdrop-filter: blur(24px);
border: 1px solid rgba(255,255,255,0.5);
padding: 40px;
border-radius: 24px;
box-shadow: 0 4px 20px rgba(0,0,0,0.05);
position: relative;
overflow: hidden;
margin-bottom: 20px;
box-sizing: border-box; /* Додано для безпеки */
">
  <div style="position: absolute; top: -60px; right: -
60px; width: 200px; height: 200px; background: radial-
gradient(circle, rgba(99,102,241,0.15) 0%, rgba(255,255,255,0)
70%); border-radius: 50%;"></div>
  <div style="position: relative; z-index: 1; display:
flex; justify-content: space-between; gap: 30px; flex-wrap:
wrap;">
    <div>
      <h1 style="margin: 0 0 10px 0; font-size:
2.5rem; font-weight: 800; color: #111827; letter-spacing: -
1px;">
        {{ classroom.name }}
      </h1>
      <p style="color: #4b5563; font-size: 1.1rem;
max-width: 600px; line-height: 1.6; margin-bottom: 15px;">
        {{ classroom.description|default:"Опис
відсутній" }}
      </p>
      <div style="display: flex; gap: 10px; align-
items: center;">
        <span style="width: 30px; height: 30px;
background: #e0e7ff; color: #4338ca; border-radius: 50%;
display: flex; align-items: center; justify-content: center;
font-size: 0.9em;"><img alt="Teacher icon" data-bbox="375 748 398 765"/></span>
        <span style="color: #6b7280; font-weight:
500;">{{
classroom.teacher.full_name|default:classroom.teacher.email
}}</span>
      </div>
    </div>
    <div style="text-align: right;">
      <div style="margin-bottom: 8px; color: #6b7280;
font-size: 0.9em; font-weight: 600; text-transform: uppercase;
letter-spacing: 1px;">Код доступу</div>

```

```

        <div style="
            background: #f3f4f6; color: #1f2937;
padding: 10px 20px;
            border-radius: 12px; font-family: monospace;
font-size: 1.5em;
            font-weight: 700; letter-spacing: 2px;
border: 1px solid #e5e7eb;
            display: inline-block;
">
            {{ classroom.join_code }}
        </div>
    </div>
</div>
<div class="section-header">
    <div class="section-title">
        <span style="background: #e0e7ff; color: #4338ca;
width: 40px; height: 40px; border-radius: 12px; display: flex;
align-items: center; justify-content: center; font-size:
1.2rem;">✍️</span>
        Тести
    </div>
    {% if user == classroom.teacher %}
        <a href="{% url 'create_test' classroom.pk %}"
class="btn btn-success" style="
            background: linear-gradient(135deg, #10b981 0%,
#059669 100%);
            border: none; padding: 10px 20px; border-radius:
12px; font-weight: 600;
            box-shadow: 0 4px 10px rgba(16, 185, 129, 0.3);
color: white; text-decoration: none;">
            + Створити тест
        </a>
    {% endif %}
</div>
<div style="display: flex; flex-direction: column; gap:
16px;">
    {% for test in classroom.tests.all %}
        <div class="item-card">
            <div style="flex-grow: 1; padding-right: 20px;">
                <h3 style="margin: 0 0 8px 0; font-size:
1.2rem; color: #1f2937;">{{ test.name }}</h3>
                <div style="display: flex; gap: 8px;">
                    <span class="badge badge-gray">{{
test.subject }}</span>
                    <span class="badge badge-blue">{{
test.questions.count }} питань</span>
                </div>
            </div>
            <a href="{% url 'test_detail' test.id %}"
class="btn-action btn-action-primary">
                Відкрити
            </a>
        </div>
    {% endfor %}
</div>

```

```

        </div>
        {% empty %}
        <div class="empty-state">
            <p>У цьому класі поки що немає тестів.</p>
        </div>
        {% endfor %}
    </div>
    <div class="section-header">
        <div class="section-title">
            <span style="background: #fce7f3; color: #db2777;
width: 40px; height: 40px; border-radius: 12px; display: flex;
align-items: center; justify-content: center; font-size:
1.2rem;"><img alt="assignment icon" data-bbox="255 265 275 285"/></span>
            Завдання
        </div>
        {% if user == classroom.teacher %}
            <a href="{% url 'create_assignment' classroom.pk %}"
class="btn btn-primary" style="
                background: linear-gradient(135deg, #4f46e5 0%,
#4338ca 100%);
                border: none; padding: 10px 20px; border-radius:
12px; font-weight: 600;
                box-shadow: 0 4px 10px rgba(79, 70, 229, 0.3);
color: white; text-decoration: none;">
                + Додати завдання
            </a>
        {% endif %}
    </div>
    <div style="display: flex; flex-direction: column; gap:
16px;">
        {% for assignment in classroom.assignments.all %}
            <div class="item-card">
                <div style="flex-grow: 1; padding-right: 20px;">
                    <div style="display: flex; align-items:
center; gap: 10px; margin-bottom: 5px;">
                        <h4 style="margin: 0; font-size: 1.1rem;
color: #111827;">{{ assignment.title }}</h4>
                        {% if assignment.file %}
                            <span title="Є вкладені файли"
style="font-size: 0.9em; cursor: help;"><img alt="file icon" data-bbox="625 705 645 725"/></span>
                        {% endif %}
                    </div>
                    <div style="display: flex; gap: 15px; flex-
wrap: wrap; margin-top: 8px;">
                        <div style="display: flex; align-items:
center; gap: 5px; font-size: 0.9em; color: #6b7280;">
                            <span><img alt="clock icon" data-bbox="555 820 575 840"/></span>
                            <span>Дедлайн:</span>
                            <span class="badge badge-red"
style="padding: 2px 8px; font-size: 0.9em;">
                                {{
assignment.due_date|date:"d.m.Y H:i" }}

```

```

        </span>
      </div>
      <div style="display: flex; align-items:
center; gap: 5px; font-size: 0.9em; color: #6b7280;">
        <span><img alt="clock icon" data-bbox="555 135 575 155"/></span>
        <span>Бали:</span>
        <strong style="color: #1f2937;">{{
assignment.max_points }}</strong>
      </div>
    </div>
  </div>
  <a href="{% url 'assignment_detail'
assignment.pk %}" class="btn-action btn-action-primary">
    Переглянути
  </a>
</div>
{% empty %}
  <div class="empty-state">
    <p>Завдань поки немає.</p>
  </div>
{% endfor %}
</div>
<div style="margin-top: 50px; text-align: center;">
  <a href="{% url 'dashboard' %}" style="
display: inline-flex; align-items: center; gap: 8px;
color: #6b7280; text-decoration: none; font-weight:
600;
padding: 12px 30px; border-radius: 50px;
background: white; border: 1px solid #f3f4f6;
transition: all 0.2s; box-shadow: 0 2px 10px
rgba(0,0,0,0.05);
" onmouseover="this.style.transform='translateY(-2px)';
this.style.boxShadow='0 5px 15px rgba(0,0,0,0.1)';"
onmouseout="this.style.transform='translateY(0)';
this.style.boxShadow='0 2px 10px rgba(0,0,0,0.05)';">
    ← Повернутися на дашборд
  </a>
</div>
</div>
{% endblock %}

```

## ДОДАТОК 7: Вихідний код create\_assignment

```

{% extends 'base.html' %}
{% block title %}Нове завдання{% endblock %}
{% block content %}
<style>
  /* Центрування контейнера */
  .dashboard-container {
    max-width: 700px;
    margin: 0 auto;
    padding: 0 20px;
  }
  /* Скліяна картка */

```

```

.glass-card {
    background: rgba(255, 255, 255, 0.85);
    backdrop-filter: blur(24px);
    border: 1px solid rgba(255, 255, 255, 0.6);
    padding: 40px;
    border-radius: 24px;
    box-shadow: 0 10px 40px rgba(0,0,0,0.05);
    position: relative;
    overflow: hidden;    }
/* Заголовки */
h2 {
    text-align: center;
    margin-top: 0;
    margin-bottom: 10px;
    color: #1f2937;
    font-weight: 800;
    font-size: 1.8rem;
    position: relative;
    z-index: 2;    }
.subtitle {
    text-align: center;
    color: #6b7280;
    margin-bottom: 30px;
    font-size: 1.1em;
    position: relative;
    z-index: 2;    }
/* Стилізація стандартних полів Django */
form p {
    margin-bottom: 25px;    }
form label {
    display: block;
    margin-bottom: 8px;
    font-weight: 600;
    color: #374151;
    font-size: 0.95rem;    }
/* Інпути (Текст, Числа, Дата, Текстова область) */
form input[type="text"],
form input[type="number"],
form input[type="datetime-local"],
form textarea,
form select {
    width: 100%;
    padding: 12px 16px;
    background: #f9fafb;
    border: 1px solid #e5e7eb;
    border-radius: 12px;
    font-size: 1rem;
    color: #1f2937;
    transition: all 0.2s;
    box-sizing: border-box;
    font-family: inherit;    }
form input:focus,
form textarea:focus,

```

```

form select:focus {
    outline: none;
    border-color: #4f46e5;
    box-shadow: 0 0 0 4px rgba(79, 70, 229, 0.1);
    background: white;    }
/* Спеціальний стиль для завантаження файлу */
form input[type="file"] {
    width: 100%;
    padding: 10px;
    background: #f9fafb;
    border: 1px solid #e5e7eb;
    border-radius: 12px;
    cursor: pointer;
    color: #4b5563;    }
form input[type="file"]::file-selector-button {
    margin-right: 15px;
    background-color: #e0e7ff;
    color: #4338ca;
    border: none;
    padding: 8px 16px;
    border-radius: 8px;
    cursor: pointer;
    transition: background .2s;
    font-weight: 600;    }
form input[type="file"]::file-selector-button:hover {
    background-color: #c7d2fe;    }
/* Підказки (help_text) */
.helpertext {
    font-size: 0.85em;
    color: #6b7280;
    margin-top: 5px;
    display: block;    }
/* Головна кнопка */
.btn-create {
    background: linear-gradient(135deg, #4f46e5 0%, #4338ca
100%);
    color: white;
    border: none;
    padding: 14px 30px;
    border-radius: 12px;
    font-weight: 600;
    font-size: 1rem;
    cursor: pointer;
    width: 100%;
    margin-top: 20px;
    box-shadow: 0 4px 15px rgba(79, 70, 229, 0.3);
    transition: transform 0.2s;
    display: block;    }
.btn-create:hover {
    transform: translateY(-2px);    }
/* Кнопка скасування */
.btn-cancel {
    display: block;

```

```

        text-align: center;
        margin-top: 20px;
        color: #6b7280;
        text-decoration: none;
        font-weight: 500;
        transition: color 0.2s;
        position: relative;
        z-index: 2;    }
    .btn-cancel:hover {
        color: #111827;    }
</style>
<div class="dashboard-container">
    <div class="glass-card">
        <div style="position: absolute; top: -80px; right: -
80px; width: 250px; height: 250px; background: radial-
gradient(circle, rgba(99, 102, 241, 0.15) 0%,
rgba(255,255,255,0) 70%); border-radius: 50%;"></div>
        <div style="position: absolute; bottom: -60px; left: -
60px; width: 200px; height: 200px; background: radial-
gradient(circle, rgba(168, 85, 247, 0.1) 0%, rgba(255,255,255,0)
70%); border-radius: 50%;"></div>
        <h2>Створити завдання</h2>
        <p class="subtitle">Для класу: <strong style="color:
#4f46e5;">{{ classroom.name }}</strong></p>
        <form method="post" enctype="multipart/form-data">
            {% csrf_token %}
            {{ form.as_p }}
            <button type="submit" class="btn-
create">Опублікувати завдання</button>
            <a href="{% url 'classroom_detail' classroom.id %}"
class="btn-cancel">Скасувати</a>
        </form>
    </div>
</div>
{% endblock %}

```

## ДОДАТОК 8: Вихідний код grade\_submission

```

{% extends 'base.html' %}
{% block title %}Оцінювання: {{ submission.student.full_name }}{% endblock %}
{% block content %}
<style>
    /* Центрування */
    .dashboard-container {
        max-width: 600px;
        margin: 0 auto;
        padding: 0 20px;    }
    /* Склояна картка */
    .glass-card {
        background: rgba(255, 255, 255, 0.85);
        backdrop-filter: blur(24px);
        border: 1px solid rgba(255, 255, 255, 0.6);
        padding: 40px;
    }

```

```

        border-radius: 24px;
        box-shadow: 0 10px 40px rgba(0,0,0,0.05);
        position: relative;
        overflow: hidden;    }
/* Блок інформації про студента */
.student-header {
    display: flex;
    align-items: center;
    gap: 15px;
    margin-bottom: 30px;
    padding-bottom: 20px;
    border-bottom: 1px solid #e5e7eb;    }
.avatar-circle {
    width: 60px;
    height: 60px;
    background: linear-gradient(135deg, #e0e7ff 0%, #c7d2fe
100%);
    color: #4338ca;
    border-radius: 50%;
    display: flex;
    align-items: center;
    justify-content: center;
    font-size: 1.5rem;
    font-weight: 700;
    border: 3px solid white;
    box-shadow: 0 4px 10px rgba(0,0,0,0.05);    }
/* Інформація про завдання */
.meta-row {
    margin-bottom: 25px;    }
.meta-label {
    font-size: 0.85em;
    color: #6b7280;
    text-transform: uppercase;
    letter-spacing: 0.5px;
    font-weight: 700;
    margin-bottom: 5px;
    display: block;    }
/* Файл */
.file-box {
    display: flex;
    align-items: center;
    gap: 10px;
    background: #f9fafb;
    border: 1px solid #e5e7eb;
    padding: 12px 16px;
    border-radius: 12px;
    text-decoration: none;
    color: #1f2937;
    font-weight: 500;
    transition: all 0.2s;    }
.file-box:hover {
    border-color: #4f46e5;
    background: #eef2ff;

```

```

        color: #4f46e5;    }
/* Стилiзацiя форми */
label {
    display: block;
    font-weight: 600;
    color: #374151;
    margin-bottom: 8px;
    margin-top: 20px;    }
/* Стилiзуємо inputs через селектори, щоб перекрити дефолтнi
стилі */
input[type="number"],
textarea {
    width: 100%;
    padding: 12px 16px;
    background: #fff;
    border: 1px solid #d1d5db;
    border-radius: 12px;
    font-size: 1rem;
    color: #111827;
    transition: all 0.2s;
    box-sizing: border-box;
    font-family: inherit;    }
input[type="number"]:focus,
textarea:focus {
    outline: none;
    border-color: #4f46e5;
    box-shadow: 0 0 0 4px rgba(79, 70, 229, 0.1);    }
/* Кнопка збереження */
.btn-save {
    background: linear-gradient(135deg, #4f46e5 0%, #4338ca
100%);
    color: white;
    border: none;
    padding: 14px 30px;
    border-radius: 12px;
    font-weight: 600;
    font-size: 1rem;
    cursor: pointer;
    width: 100%;
    margin-top: 30px;
    box-shadow: 0 4px 15px rgba(79, 70, 229, 0.3);
    transition: transform 0.2s;    }
.btn-save:hover {
    transform: translateY(-2px);    }
/* Кнопка скасування */
.btn-cancel {
    display: block;
    text-align: center;
    margin-top: 20px;
    color: #6b7280;
    text-decoration: none;
    font-weight: 500;
    transition: color 0.2s;    }

```

```

        .btn-cancel:hover {
            color: #111827;
        }
    </style>
    <div class="dashboard-container">
        <div class="glass-card">
            <h2 style="text-align: center; margin-top: 0; margin-bottom: 20px; color: #1f2937;">Оцінювання роботи</h2>
            <div class="student-header">
                <div class="avatar-circle">
                    {{
                    submission.student.full_name|first|default:submission.student.email|first|upper }}
                </div>
                <div>
                    <h3 style="margin: 0; color: #111827; font-size: 1.2rem;">{{
                    submission.student.full_name|default:submission.student.email
                    }}</h3>
                    <span style="color: #6b7280; font-size: 0.9em;">Здано: {{ submission.submitted_at|date:"d.m.Y H:i"
                    }}</span>
                </div>
            </div>
            <div class="meta-row">
                <span class="meta-label">Завдання</span>
                <div style="font-weight: 500; color: #111827; margin-bottom: 15px;">{{ submission.assignment.title }}</div>
                <span class="meta-label">Файл студента</span>
                <a href="{{ submission.file.url }}" target="_blank" class="file-box">
                    <span style="font-size: 1.2em;">📄</span>
                    Переглянути прикріплений файл
                </a>
            </div>
            <hr style="border: 0; border-top: 1px solid #e5e7eb; margin: 25px 0;">
            <form method="post">
                {% csrf_token %}
                <div>
                    <label for="{{ form.grade.id_for_label }}" style="display: flex; justify-content: space-between;">
                        Оцінка
                        <span style="background: #e0e7ff; color: #4338ca; padding: 2px 8px; border-radius: 6px; font-size: 0.8em;">
                            Максимум: {{
                            submission.assignment.max_points }}
                        </span>
                    </label>
                    {{ form.grade }}
                </div>
            </div>
        </div>
    </div>

```

```

        <label for="{{
form.teacher_feedback.id_for_label }}">Коментар вчителя
(необов'язково)</label>
        {{ form.teacher_feedback }}
    </div>
    <button type="submit" class="btn-save">Зберегти
оцінку</button>
</form>
    <a href="{% url 'assignment_detail'
submission.assignment.id %}" class="btn-cancel">Скасувати</a>
</div>
{% endblock %}

```