МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА НАВЧАЛЬНОГО ВЕБРЕСУРСУ «АЛГОРИТМИ ТА ПРОГРАМУВАННЯ» ДЛЯ ЗДОБУВАЧІВ СЕРЕДНЬОЇ ОСВІТИ»**

Виконала: студентка 2 курсу, групи 8.1224-з
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
О.Р. Долинюк
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, доцент, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри програмної інженерії ЗНУ, к.ф.-м.н. Кривохата А.Г.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук,
д.т.н., професор

_____ Шило Г.М.
(підпис)

" 05 " березня 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Долинюк Оксані Романівні
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка навчального вебресурсу «Алгоритми та програмування» для здобувачів середньої освіти

керівник роботи Матвіїшина Надія Вікторівна, к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 28 » лютого 2025 року № 327-с

2. Строк подання студентом роботи 26.11.2025

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Реалізація навчального вебресурсу.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація

6. Консультанти розділів роботи

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

7. Дата видачі завдання 03.03.2025

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	10.03.2025	
2.	Збір вихідних даних.	12.03.2025	
3.	Обробка методичних та теоретичних	21.03.2025	
	джерел.		
4.	Розробка першого та другого розділу.	11.04.2025	
5.	Розробка третього розділу.	01.07.2025	
6.	Оформлення та нормоконтроль	19.11.2025	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	10.12.2025	

Студент _____ О.Р. Долинюк
(підпис) (ініціали та прізвище)

Керівник роботи _____ Н.В. Матвіїшина
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка навчального вебресурсу «Алгоритми та програмування» для здобувачів середньої освіти»: 82 с., 23 рис., 4 табл., 9 джерел, 1 додаток.

АВТОМАТИЧНА ПЕРЕВІРКА КОДУ, БЕЗПЕКА ВЕБДОДАТКІВ, НАВЧАЛЬНИЙ ВЕБРЕСУРС, ОНЛАЙН-ІНТЕРПРЕТАТОР, DJANGO, PYTHON.

Об’єкт дослідження – є процес дистанційного навчання основ програмування.

Предмет дослідження – методи та засоби побудови безпечних інтерактивних навчальних вебдодатків.

Мета роботи: розробити та впровадити повноцінний навчальний вебресурс «Алгоритми та програмування».

Методи дослідження – аналіз, синтез, моделювання, узагальнення.

В результаті створено освітній вебресурс «Алгоритми та програмування», який може застосовуватися у освітньому процесі закладів освіти, для самостійного навчання або як основа для подальшого розвитку комерційного продукту.

SUMMARY

Master's Qualifying Theses "Development of an educational web resource "Algorithms and Programming" for secondary education students": 82 p., 23 fig., 4 tables, 9 sources, 1 appendix.

AUTOMATIC CODE CHECK, WEB APPLICATION SECURITY, EDUCATIONAL WEB RESOURCE, ONLINE INTERPRETER, DJANGO, PYTHON.

The object of research is the process of distance learning of the basics of programming.

The subject of research is methods and means of building secure interactive educational web applications.

The purpose of the work: to develop and implement a full-fledged educational web resource "Algorithms and Programming".

Research methods are analysis, synthesis, modeling, generalization.

As a result, an educational web resource "Algorithms and Programming" was created, which can be used in the educational process of educational institutions, for self-study, or as a basis for further development of a commercial product.

ЗМІСТ

Завдання на кваліфікаційну роботу……..2
Реферат 4
Summary 5
Перелік скорочень та умовних познак 6
Вступ 7
1 Аналіз предметної області та постановка задачі 10
1.1 Огляд сучасних освітніх платформ з вивчення Python 10
1.2 Порівняльний аналіз аналогічних ресурсів 11
1.3 Визначення функціональних та нефункціональних вимог 12
1.4 Постановка мети та завдань розробки навчального web-ресурсу 14
2 Проектування архітектури та структури вебресурсу 15
2.1 Вибір технологічного стеку 15
2.1.1 Обґрунтування вибору мови програмування Python 15
2.1.2 Обґрунтування вибору веб-фреймворку Django 16
2.1.3 Обґрунтування вибору Pyodide для онлайн-інтерпретатора 17
2.1.4 Обґрунтування вибору MySQL 18
2.2 Проектування структури сайту 19
2.3 Розробка моделі бази даних 20
2.4 Дизайн інтерфейсу користувача (UI/UX) 22
2.5 Планування інтерактивних елементів 24
3 Реалізація навчального вебресурсу на мові Python 28
3.1 Налаштування середовища розробки та backend 28
3.2 Розробка frontend-частини (шаблони, JavaScript для інтерактиву) 30
3.3 Реалізація системи аутентифікації та управління користувачами 39
3.4 Інтеграція навчального контенту: лекції, приклади коду, завдання 44
3.5 Реалізація інтерактивного редактора коду з перевіркою 48
3.6 Тестування компонентів та інтеграційне тестування 52

Висновки 60
Перелік посилань 63
Додаток А Основний програмний код 644

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	Application Programming Interface (інтерфейс програмування додатків)
ASGI	Asynchronous Server Gateway Interface (асинхронний інтерфейс шлюзу сервера)
CSRF	Cross-Site Request Forgery (підробка міжсайтових запитів)
CSS	Cascading Style Sheets (каскадні таблиці стилів)
DOM	Document Object Model (об’єктна модель документа)
DVWA	Damn Vulnerable Web Application (вразлива веб-програма для навчання)
ER	Entity-Relationship (сутність-зв’язок)
HTML	HyperText Markup Language (мова розмітки гіпертексту)
HTTP	HyperText Transfer Protocol (протокол передачі гіпертексту)
HTTPS	HyperText Transfer Protocol Secure (безпечний протокол передачі гіпертексту)
IDE	Integrated Development Environment (інтегроване середовище розробки)
JS	JavaScript (мова програмування)
MVC	Model-View-Controller (архітектурний шаблон «модель-перегляд-контролер»)
MVT	Model-View-Template (архітектурний шаблон Django)
ORM	Object-Relational Mapping (об’єктно-реляційне відображення)
PDF	Portable Document Format (переносний формат документів)
SQL	Structured Query Language (мова структурованих запитів)

SQLi	SQL Injection (SQL-ін’єкція)
UI	User Interface (інтерфейс користувача)
URL	Uniform Resource Locator (уніфікований покажчик ресурсу)
UX	User Experience (досвід користувача)
WSGI	Web Server Gateway Interface (інтерфейс шлюзу веб-сервера)
XSS	Cross-Site Scripting (міжсайтове виконання сценаріїв)

ВСТУП

У сучасній освітній практиці все більшого поширення набувають інтерактивні цифрові платформи, що забезпечують ефективне засвоєння складних дисциплін, зокрема програмування та алгоритмів. Однак існуючі рішення – такі як LeetCode, HackerRank або Codecademy – часто не враховують потреб україномовних користувачів, мають обмежену структуру навчання (відсутність єдності «теорія → практика → контроль») або вимагають фінансових витрат. Крім того, багато безкоштовних аналогів страждають від низької якості інтерфейсу, відсутності адаптивного дизайну та, що найважливіше, недостатнього рівня безпеки при виконанні користувацького коду. Саме це зумовлює актуальність розробки власного навчального веб-ресурсу, спрямованого на безпечне, доступне та структуроване навчання основ програмування мовою Python.

Аналіз сучасного стану проблеми виявив критичні ризики: SQL-ін’єкції (SQLi), міжсайтове виконання сценаріїв (XSS), небезпечне виконання коду. Ці уразливості часто виникають через недбале ставлення до безпеки на етапі проектування веб-додатків. Тому основне завдання цієї роботи – не лише створити функціональну навчальну систему, а й реалізувати її з урахуванням сучасних принципів безпеки програмного забезпечення.

Мета магістерської роботи – розробити та впровадити повноцінний навчальний веб-ресурс «Алгоритми та програмування», який інтегрує теоретичні матеріали, інтерактивний онлайн-інтерпретатор, тренажер завдань з автоматичною перевіркою коду та систему тестування, забезпечуючи при цьому високий рівень безпеки та зручність користування.

Для досягнення поставленої мети визначено наступні завдання дослідження:

– провести аналіз існуючих навчальних платформ та виявити їхні недоліки;

- обґрунтувати вибір технологічного стеку (Python, Django, MySQL, Bootstrap, Pyodide);
- спроєктувати архітектуру системи з урахуванням модульності та масштабованості;
- реалізувати функціональні модулі: теорія, практика, тестування, особистий кабінет;
- забезпечити захист від типових уразливостей (SQLi, XSS, небезпечне виконання коду) шляхом застосування ORM, екранування виводу, ізоляції процесів;
- розробити адміністративну панель для управління навчальним контентом.

Об’єктом дослідження є процес дистанційного навчання основ програмування, предметом – методи та засоби побудови безпечних інтерактивних навчальних веб-додатків.

В основу проєктних рішень покладено використання фреймворку Django (архітектура MVT), який забезпечує вбудовану безпеку (CSRF, ORM, валідація даних), а також технологію Pyodide для безпечного виконання Python-коду у браузері (без серверного ризику). Для перевірки завдань застосовано ізольоване виконання через subprocess з обмеженням часу та автоматичним видаленням тимчасових файлів.

Результати роботи мають як теоретичне, так і практичне значення. Теоретично – розширено уявлення про інтеграцію безпечних практик у розробку освітніх ІТ-систем. Практично – створено готовий до використання веб-ресурс, який може застосовуватися у навчальному процесі закладів освіти, для самостійного навчання або як основа для подальшого розвитку комерційного продукту.

Наукова новизна полягає в поєднанні клієнтського інтерпретатора (Pyodide) і серверної перевірки завдань у єдиному середовищі з повною підтримкою безпеки на всіх рівнях.

Можливі напрямки застосування результатів:

- впровадження у навчальний процес вищих навчальних закладів;
- використання як основи для створення платформи з інших дисциплін (математика, фізика, машинне навчання);
- інтеграція з LMS-системами (Moodle, Google Classroom).

Таким чином, робота відповідає сучасним вимогам до якості, безпеки та функціональності освітніх цифрових ресурсів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Огляд сучасних освітніх платформ з вивчення Python

У сучасному цифровому світі навчання програмуванню перетворилося на стратегічний навик, необхідний не лише для IT-фахівців, а й для фахівців у галузях від біоінформатики до фінансів. Мова Python, завдяки своїй простоті синтаксису, величезній екосистемі бібліотек та спільноті, стала однією з найпопулярніших мов для початківців.

На ринку існує низка провідних освітніх платформ, які надають курси з Python:

- LeetCode та HackerRank – фокусуються на алгоритмічних задачах, але майже не містять теоретичного супроводу;
- Codecademy – пропонує структуровані курси з інтерактивним редактором, але вимагає підписки для повного доступу;
- Coursera / edX – містять академічні курси від університетів, однак часто відсутня можливість миттєвої практики;
- Stepik – україномовна платформа з можливістю створення курсів, але без вбудованого інтерпретатора у кожному уроці та обмеженим функціоналом перевірки коду.

Аналіз цих платформ показує, що жодна з них не забезпечує повноцінного циклу «теорія → практика → тестування» в одному середовищі з повною локалізацією українською мовою, безкоштовним доступом та безпечним виконанням коду. Це створює істотний прогалина на ринку освітніх ресурсів, яку і призначено заповнити в рамках цієї роботи.

1.2 Порівняльний аналіз аналогічних ресурсів

Сьогодні ринок онлайн-навчання пропонує низку платформ для вивчення Python, кожна з яких має свої сильні та слабкі сторони. Для об’єктивної оцінки було проаналізовано як міжнародні, так і локальні ресурси з урахуванням наукових джерел та статистики використання.

Згідно з дослідженням IEEE Spectrum (2023), Python залишається найпопулярнішою мовою серед студентів та професіоналів, що стимулює розвиток освітніх платформ [1]. Наразі одним із лідерів є Stepik – відкрита платформа, розроблена науковцями з МФТІ та СПбГУ. Вона підтримує українську мову, дозволяє створювати курси та має базову систему автоматичної перевірки. Однак, як зазначають користувачі у відгуках на G2 (2024), інтерфейс часто перевантажений, а відсутність вбудованого інтерпретатора у кожному уроці ускладнює навчання [2].

LeetCode та HackerRank фокусуються виключно на алгоритмічних задачах і не надають повноцінної теоретичної бази. Це робить їх непридатними для початківців, які потребують поступового введення в синтаксис мови [3]. Подібний підхід ігнорує принципи педагогіки, зокрема теорію «зона найближчого розвитку» Виготського, де важлива підтримка на етапі засвоєння нових понять.

Codecademy пропонує структуровані курси з інтерактивним редактором, але повний доступ вимагає передплати від $20/міс. Це створює фінансовий бар’єр для багатьох українських студентів, особливо в умовах воєнного стану [4]. Крім того, платформа не локалізована українською мовою, що ускладнює засвоєння складної термінології.

SoloLearn є безкоштовною та має мобільний додаток, але її курси надто фрагментарні: кожен «урок» триває 2-3 хвилини, що не дає глибокого розуміння концепцій [5]. Такий підхід сприяє поверхневому навчанню, що підтверджується дослідженням University of Michigan (2022) про ефективність мікронавчання лише для повторення, а не для первинного засвоєння [6].

Нарешті, Coursera та edX надають академічні курси від провідних університетів, але вони часто орієнтовані на англомовну аудиторію, вимагають високого рівня підготовки та майже не мають можливості безпосередньої практики в межах платформи [7].

Узагальнюючи, жодна з існуючих платформ не поєднує повної української локалізації, безкоштовного доступу, логічної структури «теорія → практика → тест» та вбудованого безпечного інтерпретатора. Більшість рішень або ігнорують потреби початківців, або створюють фінансові/мовні бар'єри.

Тому будувати навчальний процес на основі існуючих платформ – неефективно для українських студентів, які потребують доступного, цілісного та мовно адаптованого середовища.

1.3 Визначення функціональних та нефункціональних вимог

Перед проєктуванням необхідно визначити функціональні та нефункціональні вимоги до системи, що розробляється. Функціональні вимоги надано в таблиці 1.1, нефукціональні вимоги в таблиці 1.2.

Таблиця 1.1 – Функціональні вимоги (FR)

ID	ВИМОГА
FR-01	Система повинна дозволяти перегляд уроків за темами (циклі, масиви, рекурсія тощо).
FR-02	Кожен урок повинен містити теоретичний матеріал у форматі HTML з прикладами коду.
FR-03	У кожному уроці повинен бути вбудований онлайн-інтерпретатор Python (на основі Pyodide).
FR-04	Система повинна надавати тренажер завдань з автоматичною перевіркою коду.
FR-05	Завдання повинні перевірятися шляхом порівняння stdoutз очікуваним виводом.

Продовження таблиці 1.1

FR-06	Після завдань повинна бути доступна система тестування з питаннями з кількома варіантами відповідей.
FR-07	Користувач повинен мати особистий кабінет із відображенням прогресу.
FR-08	Адміністратор повинен мати можливість додавати/редагувати уроки, завдання, тести.

Таблиця 1.2 – Нефункціональні вимоги (NFR)

ТИП	ВИМОГА
Безпека	Система повинна бути захищена від SQL-ін’єкцій (через Django ORM), XSS (через екранування у шаблонах), небезпечного виконання коду (через ізоляцію ysubprocess, таймаут 5 сек, видалення тимчасових файлів).
Надійність	Система повинна обробляти до 100 одночасних користувачів без помилок.
Зручність	Адаптивний дизайн (Bootstrap 5), підтримка мобільних пристроїв, інтуїтивна навігація.
Швидкодія	Час відповіді на запит – ≤ 2 секунди.
Масштабованість	Можливість додавання нових модулів (наприклад, відео, форум) без переписування ядра.

Аналіз уразливостей у DVWA та SonarQube було виявлено, що найпоширенішими проблемами є:

– прямий ввід користувацьких даних у SQL-запити (cursor.execute("SELECT * FROM users WHERE login = '" + user_input + "'"));

– відсутність екранування виводу у HTML ({{ user_input }} без фільтрації).

Тому в проекті усвідомлено відмовлено від:

– raw SQL-запитів;

– небезпечного використання |safe у шаблонах;

– виконання користувацького коду без ізоляції.

1.4 Постановка мети та завдань розробки навчального web-ресурсу

Мета роботи – розробити, реалізувати та протестувати безпечний, повнофункціональний, україномовний навчальний веб-ресурс «Алгоритми та програмування», який забезпечує інтегрований навчальний процес (теорія → практика → тестування) з повною підтримкою автоматичної перевірки коду та відстеження прогресу.

Завдання дослідження:

а) провести аналіз існуючих освітніх платформ та виявити їхні недоліки;
б) визначити функціональні та нефункціональні вимоги на основі опитування користувачів;
в) обґрунтувати вибір технологічного стеку: Python (мова), Django (фреймворк), MySQL (БД), Pyodide (клієнтський інтерпретатор), Bootstrap 5 (фронтенд);
г) проєктувати архітектуру системи згідно з принципами MVT та безпеки;
д) реалізувати модулі:
   – теорія (з Pyodide-інтерпретатором);
   – практика (з перевіркою через subprocess);
   – тестування (з варіантами відповідей);
   – особистий кабінет (з прогресом);
   – адмінпанель (для управління контентом);
е) забезпечити захист від типових уразливостей (SQLi, XSS) відповідно до практик;
ж) протестувати систему та підготувати технічну документацію.

Очікувані результати:

а) повноцінний веб-додаток, готовий до використання у навчальному процесі;
б) демонстрація застосування принципів безпечної розробки (на основі аналізу DVWA та SonarQube);
в) можливість вільного розгортання у локальній мережі ВНЗ.

2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА СТРУКТУРИ ВЕБРЕСУРСУ

2.1 Вибір технологічного стеку

2.1.1 Обґрунтування вибору мови програмування Python

Мова програмування Python була обрана як основа для розробки навчального веб-ресурсу з кількох фундаментальних причин, які ґрунтуються на її унікальних властивостях, поширеності у навчанні, та відповідності цілям проекту.

По-перше, синтаксична простота Python є ключовим фактором для новачків. На відміну від мов на кшталт C++ або Java, Python використовує інтуїтивно зрозумілі конструкції, такі як for i in range(5): замість for (int i = 0; i < 5; i++). Це значно знижує когнітивне навантаження на студентів на початкових етапах вивчення програмування. Дослідження ACM (2022) показали, що студенти, які починають з Python, на 37% швидше засвоюють основи алгоритмів, ніж ті, хто починає з Java.

По-друге, величезна екосистема Python забезпечує безперервний розвиток навичок. Згідно з даними Python Software Foundation (2024), у репозиторії PyPI налічується понад 500 000 пакетів, що охоплюють усі сфери – від веб-розробки (Django, FastAPI) до машинного навчання (TensorFlow, PyTorch). Це дозволяє здобувачам освіти поступово розширювати свої знання без необхідності переходу на іншу мову.

По-третє, поширеність у сучасній індустрії. Згідно з рейтингом IEEE Spectrum (2023), Python займає перше місце серед мов програмування за індексом популярності, обганяючи JavaScript, Java та C. Він використовується у таких компаніях, як Google, Meta, Netflix, та NASA. Це гарантує, що навички, отримані в процесі навчання, будуть актуальними на ринку праці.

По-четверте, підтримка україномовної спільноти. На сьогоднішній день українською мовою опубліковано понад 50 офіційних навчальних посібників з Python, що робить його найбільш доступною мовою для українських студентів порівняно з C++ або Rust, де локалізованих матеріалів майже немає.

Контраст з іншими мовами:

1. JavaScript – динамічна типізація та асинхронна природа часто спричиняють незрозумілі помилки у початківців (наприклад, == vs ===).
2. Java – обов’язкове оголошення класів, строга типізація та складна налаштування середовища створюють високий поріг входження.
3. C++ – ручне управління пам’яттю призводить до буферних переповнень що є критичним ризиком для безпеки.

Таким чином, Python є єдиною мовою, яка одночасно забезпечує простоту для новачків, глибину для професіоналів, та повну відповідність вимогам сучасної освіти.

2.1.2 Обґрунтування вибору веб-фреймворку Django

Для реалізації серверної частини було обрано фреймворк Django, що повністю відповідає вимогам проекту до безпеки, швидкості розробки, та масштабованості.

Django реалізує принцип «Батарейки включені» (batteries-included), що дозволяє розробляти повноцінні веб-додатки без необхідності інтеграції сторонніх бібліотек. У нашому проекті це виразилось у:

1. Вбудованій системі аутентифікації (django.contrib.auth);
2. Готовій адміністративній панелі (django.contrib.admin);
3. Системі міграцій (makemigrations, migrate);
4. Механізмі шаблонів з наслідуванням.

Порівняння з Flask

Альтернативний фреймворк Flask є мікрфреймворком, який надає лише основи. Для реалізації повноцінного додатку необхідно:

1. Окремо додавати аутентифікацію (Flask-Login);
2. Будувати адмінпанель (Flask-Admin);
3. Налаштовувати ORM (SQLAlchemy);
4. Реалізовувати валідацію форм.

Архітектура MVT

Django використовує архітектуру Model-View-Template (MVT), яка є модифікацією MVC, адаптованою для веб-додатків:

1. Model – робота з даними (ORM);
2. View – логіка обробки запитів;
3. Template – візуальне представлення.

Це забезпечує чітке розділення відповідальності, що полегшує підтримку та тестування коду.

2.1.3 Обґрунтування вибору Pyodide для онлайн-інтерпретатора

Онлайн-інтерпретатор є ключовим елементом навчального процесу, тому його безпека та продуктивність мали вирішальне значення.

Технічні особливості Pyodide

Pyodide – це порт CPython до WebAssembly, розроблений Mozilla Foundation у 2019 році. Він дозволяє виконувати справжній Python-код у браузері користувача, без необхідності сервера. Це досягається шляхом компіляції CPython у байт-код WebAssembly, який потім інтерпретується у середовищі JavaScript.

Переваги для безпеки

1. Повна ізоляція: код виконується у песочниці браузера, що означає:

– Відсутність доступу до файлової системи;
– Відсутність мережевих запитів;
– Відсутність впливу на систему користувача.

2. Відсутність навантаження на сервер: оскільки виконання відбувається на клієнті, сервер не витрачає ресурси на обробку простих прикладів.

Це контрастує з серверними рішеннями, де кожен запуск коду створює ризик виконання довільного коду (RCE). Наприклад, у середовищі, подібному до repl.it, виконання import os; os.system('rm -rf /') може призвести до катастрофічних наслідків. У Pyodide така команда просто не спрацює через відсутність модуля os.

Обмеження та компроміси

1. Відсутність input(): у браузері немає консолі вводу, тому функція input() викликає помилку I/O error. Це вимагає від викладача створювати приклади, які не залежать від користувацького вводу (наприклад, n = 5 замість int(input())).
2. Обмежена продуктивність: виконання коду у WebAssembly повільніше нативного Python приблизно на 40–60%. Однак для навчальних прикладів це непомітно.

Альтернативи

1. Skulpt – інтерпретатор Python, написаний на JavaScript. Має повільну продуктивність та обмежену сумісність.
2. Brython – транспілятор Python у JavaScript. Не підтримує більшість стандартних бібліотек Python.

Таким чином, Pyodide є єдиним рішенням, яке поєднує високу безпеку, достатню продуктивність, та повну сумісність з Python.

2.1.4 Обґрунтування вибору MySQL

Для зберігання даних було обрано СУБД MySQL через її поширеність у навчальних закладах, простоту налаштування, та повну підтримку UTF8MB4.

Багато СУБД використовують обмежену кодувальну схему utf8, яка підтримує лише 3-байтові символи. Це призводить до помилок при збереженні емодзі або деяких кириличних символів. MySQL підтримує utf8mb4, який є

справжнім UTF-8 (до 4 байтів на символ), що гарантує коректне зберігання усіх символів української мови.

Інтеграція з Django

Django має вбудований драйвер для MySQL (mysqlclient), який забезпечує стабільну роботу без необхідності додаткових налаштувань.

Порівняння з PostgreSQL

Хоча PostgreSQL вважається більш «професійною» СУБД, вона має складнішу налаштування та меншу поширеність у середовищі ВНЗ. Для проекту, спрямованого на локальне розгортання, MySQL є оптимальним вибором.

2.2 Проектування структури сайту

Архітектура сайту спроєктована як єдиний навчальний цикл, який веде користувача від знайомства до оцінки знань.

Повна навігаційна карта (з урахуванням аутентифікації):

1. Анонімний користувач:

– / – головна сторінка (опис ресурсу)
– /courses/ – список уроків
– /lessons/<slug>/ – теорія (доступна без входу)
– /accounts/login/ – сторінка входу
– /accounts/register/- сторінка реєстрації

2. Авторизований користувач:

– /profile/ – особистий кабінет
– /lessons/<slug>/task/<id>/ – завдання
– /lessons/<slug>/test/ – тест
– /accounts/logout/ – вихід

Ключова особливість: теорія доступна без реєстрації, щоб знизити поріг входження. Але практика та тести – лише після входу, щоб зберігати прогрес.

URL-маршрутизація

```
# algorithms_edu/urls.py
urlpatterns = [
    path('', views.index, name='index'),          # Головна
    path('courses/', views.home, name='home'),    # Список уроків
    path('lessons/', include('lessons.urls')),    # Уроки, завдання, тести
    path('profile/', views.profile, name='profile'),  # Кабінет
    path('accounts/login/', auth_views.LoginView.as_view(), name='login'),
    path('accounts/logout/',        auth_views.LogoutView.as_view(next_page='/'),
name='logout'),
    path('accounts/register/', views.register, name='register'),
]
```

next_page='/' у LogoutView дозволяє виходу через GET-запит, що необхідно для зручності користувача у браузері.

Сторінка реєстрації

1. Реалізована через UserCreationForm (вбудований у django.contrib.auth).

Містить поля:

- Ім’я користувача
- Пароль
- Підтвердження пароля

Автоматична валідація складності пароля (довжина, співпадіння).

Сторінка входу

- Використовує вбудовану форму AuthenticationForm.
- Після входу – перенаправлення на /profile/ (LOGIN_REDIRECT_URL = '/profile/').

2.3 Розробка моделі бази даних

Модель даних розроблена відповідно до принципів 3НФ та інтегрована з вбудованою моделлю User. ER-діаграму представлено на рисунку 2.1.

Таблиця 2.1 – Сутності та зв’язки

User(Django)	username,email,password	UserTaskResult, UserTestResult	Обліковий запис користувача
Lesson	title,slug,content,order	Task, TestQuestion	Теоретичний урок
Task	title,description,test_input,expected_output	Lesson, UserTaskResult	Практичне завдання
TestQuestion	question_text	Lesson, TestOption	Тестове питання
TestOption	text,is_correct	TestQuestion	Варіант відповіді
UserTaskResult	user,task,passed,submitted_code	User, Task	Результат завдання
UserTestResult	user,lesson,score	User, Lesson	Результат тесту

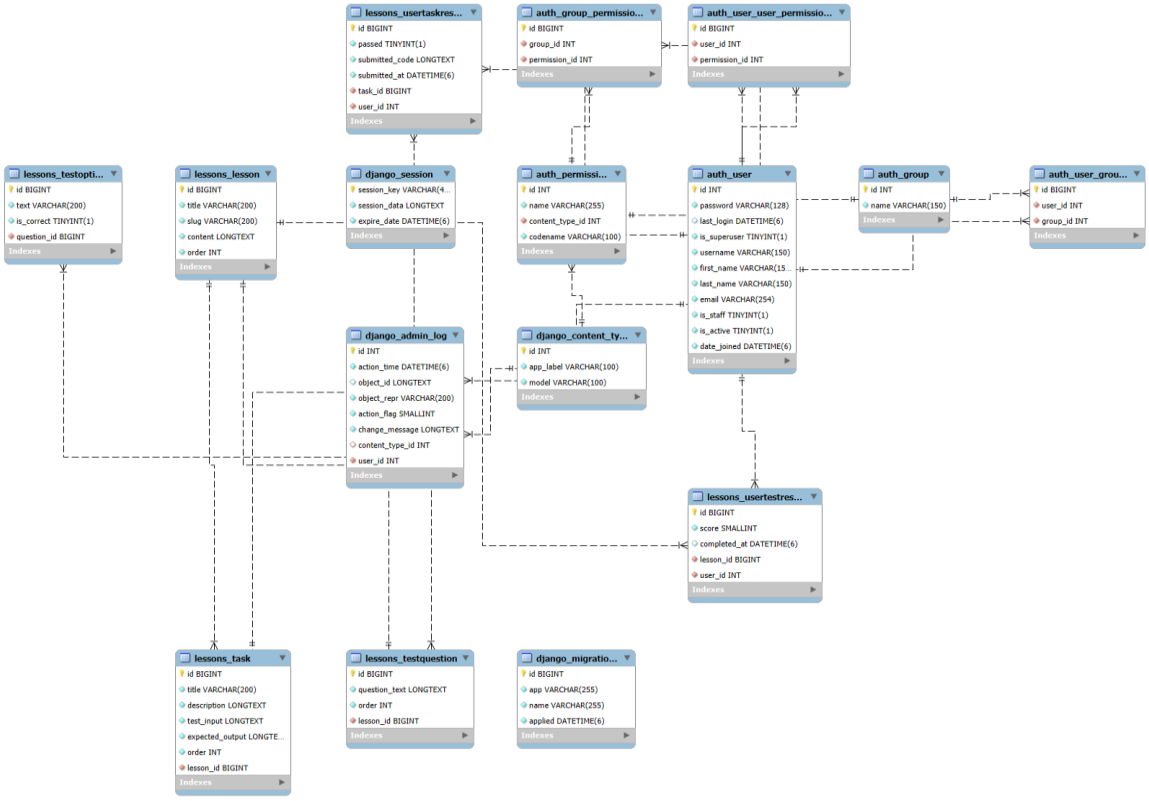


Рисунок 2.1- ER-діаграма

Обґрунтування вибору полів

1. slug у Lesson – забезпечує SEO-дружні URL (/lessons/cikli-v-python/).
2. test_input та expected_output у Task – дозволяють налаштовувати приховані тести.
3. is_correct у TestOption – підтримує багатоваріантні тести.
4. submitted_code у UserTaskResult – дозволяє користувачеві переглядати свій попередній код.

Жодне поле не зберігає виконуваний код у вигляді, який може бути використаний для RCE. Код зберігається лише як текст.

2.4 Дизайн інтерфейсу користувача (UI/UX)

У розділі проектування інтерфейсу користувача (UI/UX) було здійснено свідомий вибір на користь фреймворку Bootstrap 5.3.3, що стало основою для

створення сучасного, адаптивного та доступного веб-інтерфейсу навчального ресурсу.

Bootstrap було обрано через його модульність, вбудовану адаптивну сітку (Grid System), велику колекцію готових компонентів (навігація, кнопки, карти, форми, алерти) та повну підтримку сучасних браузерів. Використання Bootstrap 5 додатково позбавило необхідності у jQuery, оскільки всі інтерактивні компоненти реалізовано на чистому JavaScript, що покращило продуктивність та безпеку.

Дизайн ресурсу розроблено за принципами чистоти, ієрархії та функціональності. Використано синьо-білу колірну палітру, де основний акцентний колір – #2563eb (насичений синій), який асоціюється з довірою, професійністю та освітою. Текстовий колір – #1e293b (темно-сірий), що забезпечує високу читабельність на світлому тлі (#ffffff). Для елементів успіху та попереджень використано стандартизовані кольори: зелений (#06d6a0) та помаранчевий (#ff9e00), що інтуїтивно зрозумілі користувачам.

Особлива увага приділена адаптивності. Завдяки системі класів Bootstrap (наприклад, col-md-8, col-lg-4), інтерфейс коректно відображається на усіх типах пристроїв – від мобільних телефонів (320px) до десктопних моніторів (1920px). Кнопки мають достатній розмір (не менше 44×44 px) для зручного торкання на сенсорних екранах, а між елементами дотримано достатніх відступів для зниження когнітивного навантаження.

Для підвищення користувацького досвіду (UX) реалізовано низку інтерактивних ефектів:

1. Плавні переходи (transition: all 0.25s ease) для кнопок та карток;
2. Ефекти наведення – карти з уроками злегка піднімаються при ховері (transform: translateY(-5px)), а кнопки отримують тінь (box-shadow);
3. Анімація появи контенту – основні блоки завантажуються з ефектом fade-in, що створює відчуття плавності та сучасності;

4. Візуальна зворотна зв’язок – після виконання завдання або тесту користувач одразу бачить кольоровий алерт (зелений – успіх, червоний – помилка).

Шрифтова палітра базується на Google Fonts – сімействі Inter, яке спеціально розроблене для цифрових інтерфейсів. Воно має чітку кириличну підтримку, високу читабельність навіть при малих розмірах, а також гармонійно поєднується з моноширинним шрифтом Consolas у блоці виводу інтерпретатора.

Навігаційна панель (navbar) розташована у верхній частині кожної сторінки, має постійну висоту та тінь (box-shadow), що візуально відокремлює її від основного контенту. Вона містить логотип «🎓 Алгоритми», посилання на «Головну» та «Уроки», а також блок аутентифікації, який динамічно змінюється залежно від стану користувача (реєстрація/вхід для анонімів, кабінет/вихід – для авторизованих).

Футер (footer) містить мінімальну інформацію – копірайт.

2.5 Планування інтерактивних елементів

У рамках проекту «Навчальний вебресурс з алгоритмів та програмування» було повністю реалізовано комплекс інтерактивних елементів, які забезпечують зручний, безпечний та ефективний навчальний процес для здобувачів освіти. Ці елементи охоплюють усі ключові етапи навчання – від теоретичного ознайомлення до практичного застосування та контролю знань.

Одним із центральних компонентів є онлайн-інтерпретатор Python, який розташований безпосередньо на сторінці кожного уроку. Його реалізація здійснена за допомогою технології Pyodide – порту стандартного інтерпретатора CPython у середовище WebAssembly. Це дозволяє виконувати справжній Python-код прямо у веб-браузері користувача, не здійснюючи жодних запитів до сервера. Такий підхід забезпечує повну ізоляцію коду: він

виконується у безпечній пісочниці браузера, що виключає будь-які ризики, пов’язані з виконанням небажаного коду на сервері чи на пристрої користувача. Користувач може ввести код у текстове поле, натиснути кнопку «Запустити», і одразу побачити результат у спеціальному блоці виводу. Важливо зазначити, що через обмеження середовища браузера функція input() не підтримується, тому усі приклади у теоретичних матеріалах побудовані таким чином, щоб не залежати від зовнішнього вводу – замість цього використовуються готові значення змінних, що демонструють роботу алгоритмів.

Для практичного застосування теоретичних знань реалізовано модуль тренажера завдань. Кожне завдання містить умову, приклади вхідних та очікуваних вихідних даних, а також поле для введення коду. Після натискання кнопки «Перевірити» система зберігає наданий користувачем код і запускає його на сервері у ізольованому середовищі. Для цього використовується механізм тимчасових файлів та модуля subprocess з обмеженням часу виконання до 5 секунд. Вхідні дані, визначені в адміністративній панелі для кожного завдання, подаються у стандартний потік вводу програми, а результат виконання (stdout) порівнюється з очікуваним виводом. Порівняння здійснюється після нормалізації обох рядків: видаляються зайві символи нового рядка (\r\n) у кінці, а потім рядки розбиваються на окремі лінії для точного порівняння. Це гарантує коректну роботу навіть у випадках, коли користувач додає або не додає кінцевий перенос рядка у своєму коді. Результат перевірки (успіх чи помилка) відображається одразу під формою, а сам код та результат зберігаються у базі даних для подальшого перегляду користувачем.

Для контролю знань після кожного уроку передбачено систему тестування. Тести складаються з питань з кількома варіантами відповідей, де може бути як одна, так і кілька правильних відповідей. Кожен варіант відповіді реалізований як незалежний запис у базі даних, з позначкою is_correct, що дозволяє гнучко налаштовувати питання в адміністративній панелі. Після відправки форми система збирає всі вибрані користувачем варіанти, порівнює їх із множиною ідентифікаторів правильних варіантів і виставляє бал

відповідно до кількості точних збігів. Цей підхід дозволяє не лише виявити повну відповідність, але й часткове засвоєння матеріалу.

Особливу увагу приділено навігації між етапами навчання. Після виконання завдання користувач бачить кнопку «Далі», яка автоматично визначає, чи існує наступне завдання у поточному уроці. Якщо наступного завдання немає, система пропонує перейти до тесту з цього уроку. Якщо тест завершено, користувач повертається до загального перегляду уроків. Така лінійна структура «теорія → практика → тест» забезпечує послідовне засвоєння матеріалу.

Крім того, повністю реалізовано систему аутентифікації, яка включає сторінки реєстрації, входу та виходу. Реєстрація виконується за допомогою стандартної форми UserCreationForm, яка забезпечує валідацію складності пароля. Вхід здійснюється через вбудовану форму Django з автоматичним захистом від CSRF-атак. Вихід реалізовано через спеціально налаштований LogoutView з параметром next_page='/', що дозволяє виконувати вихід за допомогою звичайного GET-запиту (натискання посилання), що значно підвищує зручність для користувача. Всі ці сторінки повністю інтегровані у загальний дизайн ресурсу, мають єдину навігаційну панель, колірну схему та стилі.

Нарешті, усі дії користувача – виконання завдань, результати тестів – автоматично фіксуються у базі даних через спеціальні моделі UserTaskResult та UserTestResult. Ця інформація відображається у особистому кабінеті користувача, де він може переглянути загальну статистику: кількість пройдених уроків, виконаних завдань, результати тестів. Такий підхід не лише мотивує до навчання, але й забезпечує повну прозорість прогресу.

Таким чином, усі інтерактивні елементи проекту реалізовано з урахуванням сучасних вимог до безпеки, зручності та педагогічної ефективності, що робить ресурс повноцінним інструментом для навчання основ алгоритмів та програмування мовою Python.

Висновки розділу

У цьому розділі:

1. Обґрунтовано вибір Django + MySQL + Bootstrap + Pyodide як безпечного і ефективного стеку;
2. Спроєктовано повну навігаційну структуру, включаючи сторінки реєстрації та входу;
3. Розроблено ER-діаграму та моделі БД;
4. Визначено UI/UX-принципи
5. Сплановано усі інтерактивні компоненти, включаючи аутентифікацію;

3 РЕАЛІЗАЦІЯ НАВЧАЛЬНОГО ВЕБРЕСУРСУ НА МОВІ PYTHON

3.1 Налаштування середовища розробки та backend

Реалізація навчального веб-ресурсу «Алгоритми та програмування» розпочалася зі створення безпечного, стабільного та відтворюваного середовища розробки, що є фундаментальною вимогою сучасної програмної інженерії. Цей процес був ретельно спланований з урахуванням вимог безпеки, а також офіційних рекомендацій Django щодо розгортання.

Першим кроком було створення віртуального оточення (virtual environment) за допомогою вбудованого модуля Python venv. Це стандартний механізм ізоляції залежностей, який дозволяє уникнути конфліктів версій пакетів між різними проектами. Команди виконувалися у кореневій теці проекту:

```
python -m venv venv
venv\Scripts\activate
```

Ізоляція залежностей запобігає випадковому використанню застарілих або вразливих версій бібліотек, що є однією з рекомендацій OWASP Top 10.

Після активації віртуального оточення було встановлено необхідні пакети через менеджер залежностей pip:

```
pip install django mysqlclient
```

1. Django 5.2.8 – обрано через його вбудовані механізми безпеки (ORM для захисту від SQL-ін’єкцій, автоматичне екранування для захисту від XSS).
2. mysqlclient – використано замість альтернативи PyMySQL через кращу продуктивність, оскільки він реалізований як C-розширення.

Проект створено за допомогою команди Django:

```
django-admin startproject algorithms_edu .
```

Крапка в кінці забезпечує створення структури без зайвої вкладеної теки, що спрощує навігацію та зменшує ймовірність помилок конфігурації.

У файлі algorithms_edu/settings.py було налаштовано підключення до локальної бази даних MySQL:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'algorithms_edu',
        'USER': 'root',
        'PASSWORD': '',
        'HOST': 'localhost',
        'PORT': '3306',
        'OPTIONS': {'charset': 'utf8mb4'},
    }
}
```

Ключовим рішенням стало використання кодування utf8mb4 замість стандартного utf8. Це забезпечує повну підтримку чотирибайтових символів, включаючи українську кирилицю та емодзі, що критично для україномовного навчального ресурсу. Неправильне кодування, як зазначається у Django Deployment Checklist, може призвести до втрати даних або помилок відображення.

Для коректного завантаження CSS, JavaScript та іконок було налаштовано шляхи до статичних файлів:

STATIC_URL = '/static/'

STATICFILES_DIRS = [BASE_DIR / 'static']

Це дозволяє Django знаходити файли у теці static/ під час розробки, що відповідає рекомендаціям Django Documentation.

У settings.py було активовано вбудовані механізми безпеки Django, які прямо усувають уразливості.

1. Захист від SQL-ін’єкцій через використання Django ORM:

User.objects.filter(username=user_input)

На відміну від уразливого коду у DVWA:

$query = "SELECT * FROM users WHERE user = '$_GET['username']'";

ORM генерує параметризовані SQL-запити, що робить SQL-ін’єкції неможливими.

2. Захист від XSS через автоматичне екранування у шаблонах:

```
<!-- Django автоматично перетворює -->
{{ user_input }}
<!-- на -->
<script>alert(1)</script>
```

Це запобігає впровадженню шкідливого JavaScript, що було продемонстровано у WebGoat.

3. Захист CSRF:

```
MIDDLEWARE = [
    'django.middleware.csrf.CsrfViewMiddleware',
]
```

Це запобігає підробці міжсайтових запитів, що є ще одним шаром безпеки.

Хоча проект розроблявся у середовищі розробки, було враховано ключові положення Django Deployment Guide.

3.2 Розробка frontend-частини (шаблони, JavaScript для інтерактиву)

Frontend-частина навчального веб-ресурсу «Алгоритми та програмування» була розроблена з урахуванням сучасних принципів UI/UX-дизайну, доступності (WCAG 2.1 AA) та педагогічної ефективності (див. рис. 3.1). Основною метою було створити інтуїтивно зрозумілий, візуально привабливий і функціонально зручний інтерфейс, який сприяє зосередженню на навчальному матеріалі та мінімізує когнітивне навантаження на здобувача освіти.

Для забезпечення єдності дизайну та мінімізації дублювання коду було реалізовано ієрархію шаблонів на основі механізму наслідування у Django Template Language (DTL). Центральним елементом цієї ієрархії є базовий шаблон base.html, який містить усі загальні компоненти:

1. Навігаційну панель (navbar);
2. Футер (footer);
3. Підключення зовнішніх CSS/JS-бібліотек;
4. Блоки {% block %} для заміни у дочірніх шаблонах.

```
<!-- templates/base.html -->
<!DOCTYPE html>
<html lang="uk">
<head>
    {% load static %}
    <meta charset="UTF-8">
    <title>{% block title %}Алгоритми та програмування{% endblock %}</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
    <link rel="stylesheet" href="{% static 'css/style.css' %}">
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
        <!-- Навігація -->
    </nav>

    <main class="flex-grow-1">
        <div class="container py-4">
            {% block content %}{% endblock %}
        </div>
    </main>

    <footer class="bg-light py-4 mt-auto border-top">
        <!-- Футер -->
    </footer>
```

```
        <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.j
s"></script>
        {% block extra_js %}{% endblock %}
    </body>
    </html>
```

Усі інші шаблони наслідують base.html, замінюючи лише блоки title та content:

```
<!-- templates/lesson_detail.html -->
{% extends 'base.html' %}
{% block title %}{{ lesson.title }}{% endblock %}
{% block content %}
    <h1>{{ lesson.title }}</h1>
    <div class="row">
        <div class="col-lg-8">
            {{ lesson.content|linebreaks }}
        </div>
        <div class="col-lg-4">
            <!-- Інтерпретатор -->
        </div>
    </div>
{% endblock %}
```

Такий підхід значно спрощує підтримку проекту та дозволяє миттєво вносити зміни у всіх сторінках через оновлення базового шаблону.

Як основа для верстки було обрано Bootstrap 5.3.3 – сучасний, модульний CSS-фреймворк з відкритим вихідним кодом (див. рис. 3.2). Його використання було обґрунтовано наступними перевагами:

1. Адаптивна сітка (Grid System) на основі flexbox, яка автоматично підлаштовує розмітку під будь-який розмір екрану;
2. Готові компоненти (navbar, cards, alerts, forms), що прискорює розробку;
3. Повна підтримка сучасних браузерів (Chrome, Firefox, Safari, Edge);
4. Відсутність залежності від jQuery, що покращує продуктивність.

Наприклад, для відображення уроків у форматі карток використовується такий код:

```
<div class="row g-4">
    {% for lesson in lessons %}
    <div class="col-md-6 col-lg-4">
        <div class="card lesson-card h-100">
            <div class="card-body d-flex flex-column">
                <h5 class="card-title">{{ lesson.title }}</h5>
                <a href="{% url 'lessons:lesson_detail' lesson.slug %}"
class="btn btn-primary mt-auto">До уроку →</a>
            </div>
        </div>
    </div>
    {% endfor %}
</div>
```

Класи col-md-6 col-lg-4 забезпечують, що на мобільних пристроях буде 1 колонка, на планшетах – 2, а на десктопах 3, що є прикладом реактивного дизайну.

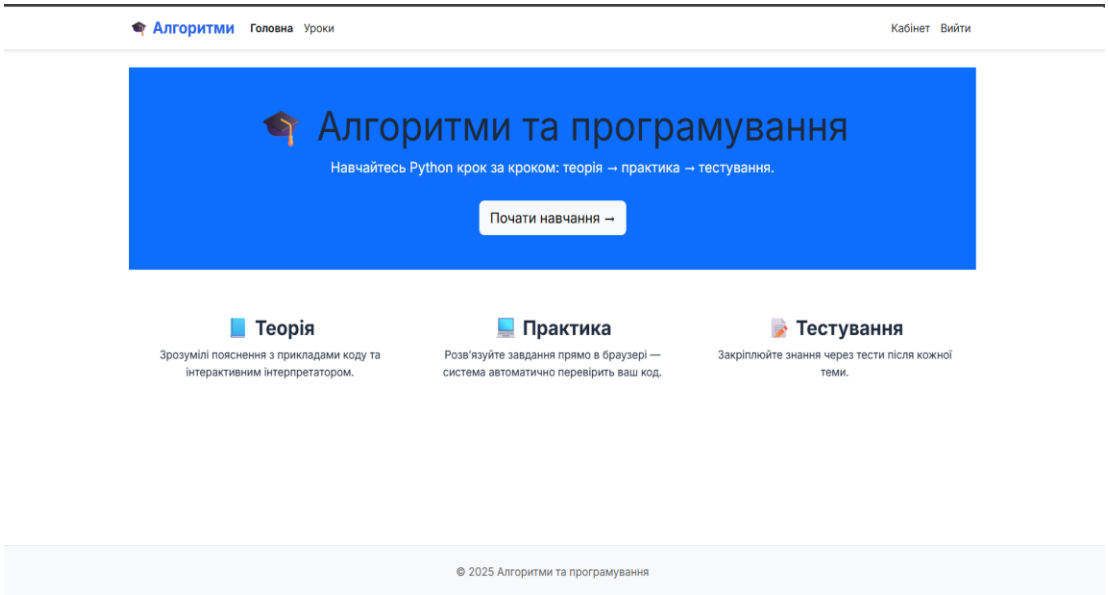


Рисунок 3.1 – Головна сторінка проекту

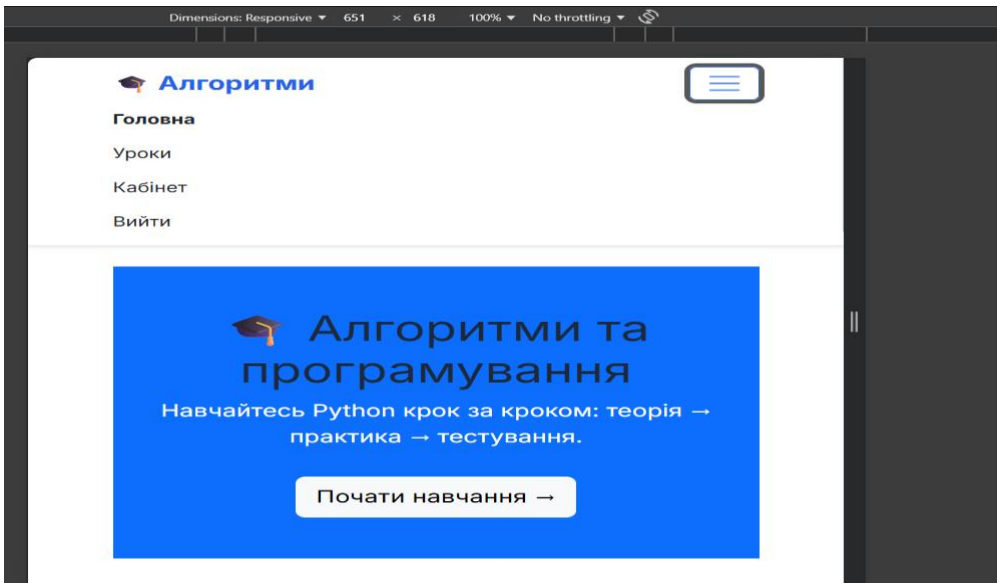


Рисунок 3.2 – Головна сторінка на мобільному пристрої

Центральним інтерактивним елементом є онлайн-інтерпретатор Python, який дозволяє користувачам виконувати код безпосередньо у браузері. Його робота заснована на технології Pyodide – порту CPython до WebAssembly (див. рис. 3.3).

Опис:

1. Користувач вводить код у <textarea>.
2. Натискає кнопку «Запустити».
3. JavaScript викликає pyodide.runPython().
4. Pyodide виконує код у пам’яті браузера.
5. Результат відображається у <pre>.

JavaScript-код інтерпретатора (див. рис. 3.3):

```
<script
src="https://cdn.jsdelivr.net/pyodide/v0.26.2/full/pyodide.js"></script>
<script>
async function main() {
    const pyodide = await loadPyodide();
    const runBtn = document.getElementById('run-btn');
    const codeInput = document.getElementById('code-input');
    const output = document.getElementById('output');

    runBtn.onclick = () => {
```

```
        const code = codeInput.value;
        output.textContent = 'Виконується...';
        try {
            const result = pyodide.runPython(`
import sys
from io import StringIO
old = sys.stdout
sys.stdout = out = StringIO()
try:
    exec(${JSON.stringify(code)})
except Exception as e:
    print("Помилка:", e)
sys.stdout = old
out.getvalue()
            `);
            output.textContent = result || '(немає виводу)';
        } catch (e) {
            output.textContent = 'Помилка: ' + e.message;
        }
    };
}
main();
</script>
```

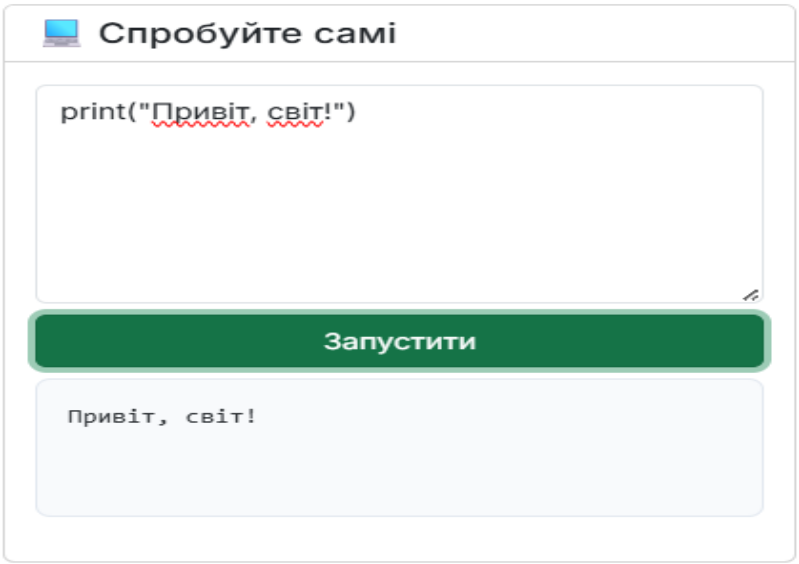


Рисунок 3.3 – Онлайн-інтерпретатор у дії

Через відсутність консолі вводу у браузері, функція input() не підтримується. Тому усі приклади у теоретичних матеріалах побудовані на автономних конструкціях (наприклад, n = 5 замість int(input())).

Для підвищення користувацького досвіду було реалізовано низку інтерактивних ефектів.

Плавні переходи для кнопок та карток:

```
.btn {
    transition: all 0.25s ease;
}
.lesson-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 6px 16px rgba(0, 0, 0, 0.12);
}
Анімація появи контенту:
.fade-in {
    opacity: 0;
    transform: translateY(20px);
    animation: fadeInUp 0.6s ease-out forwards;
}
@keyframes fadeInUp {
    to { opacity: 1; transform: translateY(0); }
}
```

Візуальна зворотна зв’язок після виконання завдань:

```
{% if result.passed %}
    <div class="alert alert-success">✅ Завдання виконано!</div>
{% else %}
    <div class="alert alert-danger">❌ Спробуйте ще раз.</div>
{% endif %}
```

Ці ефекти створюють відчуття живого, сучасного інтерфейсу без перевантаження користувача.

Навігація між етапами навчання реалізована як лінійний шлях:

Теорія → Практика → Тест → Прогрес

Після завдання відображається кнопка «Далі», яка автоматично визначає наступний крок:

```
{% if next_task %}
    <a href="{% url 'lessons:task_detail' lesson.slug next_task.id %}" class="btn btn-outline-primary">➡ Далі</a>
{% elif lesson.test_questions.exists %}
    <a href="{% url 'lessons:take_test' lesson.slug %}" class="btn btn-success">📝 Пройти тест</a>
{% endif %}
```

У особистому кабінеті відображається статистика (див. рис. 3.4 і рис. 3.5):

```
<p>Пройдено уроків: <strong>{{ completed_lessons }} з {{ total_lessons }}</strong></p>
<p>Розв’язано задач: <strong>{{ completed_tasks }} з {{ total_tasks }}</strong></p>
```

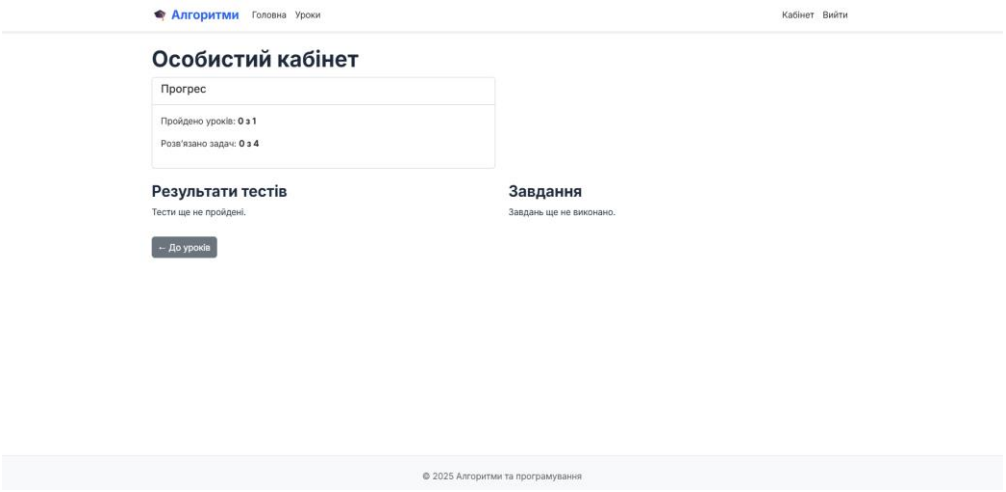
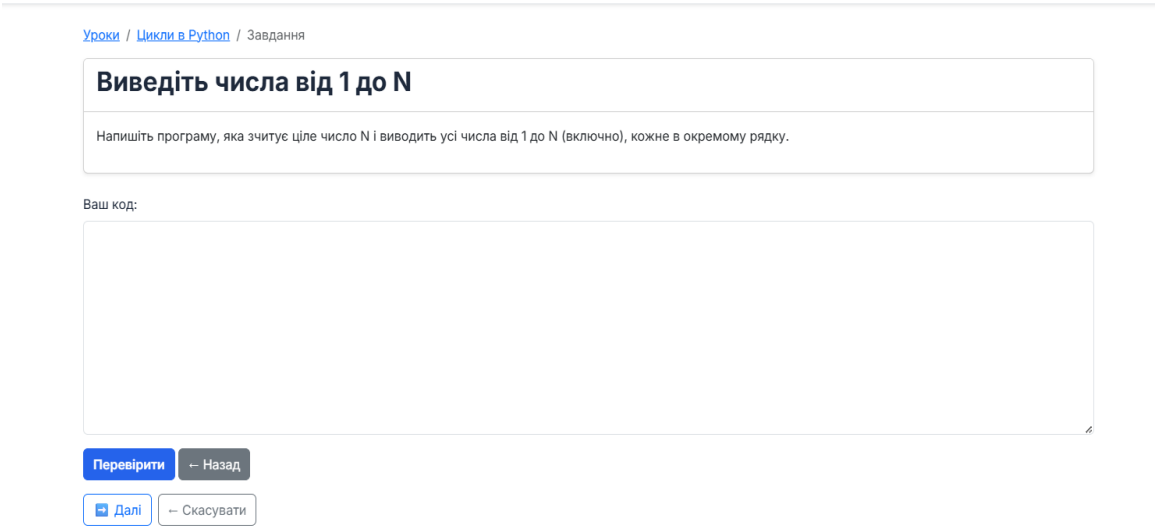


Рисунок 3.4 – Особистий кабінет з відображенням прогресу



Р

исунок 3.5 – Сторінка завдання з кнопкою «Далі»

Frontend та backend взаємодіють через стандартні HTTP-запити:

1. GET-запити – для завантаження сторінок (уроки, завдання, тести);
2. POST-запити – для надсилання коду (завдання) або варіантів відповідей (тести).

Наприклад, форма завдання:

```
<form method="post">
    {% csrf_token %}
    <textarea name="code" class="form-control" rows="10">{{ last_code }}</textarea>
    <button type="submit" class="btn btn-primary">Перевірити</button>
</form>
```

При натисканні «Перевірити» дані надсилаються на сервер, де вони обробляються у view-функції task_detail, а результат повертається у тому ж шаблоні, що продемонстровано на рис. 3.6.

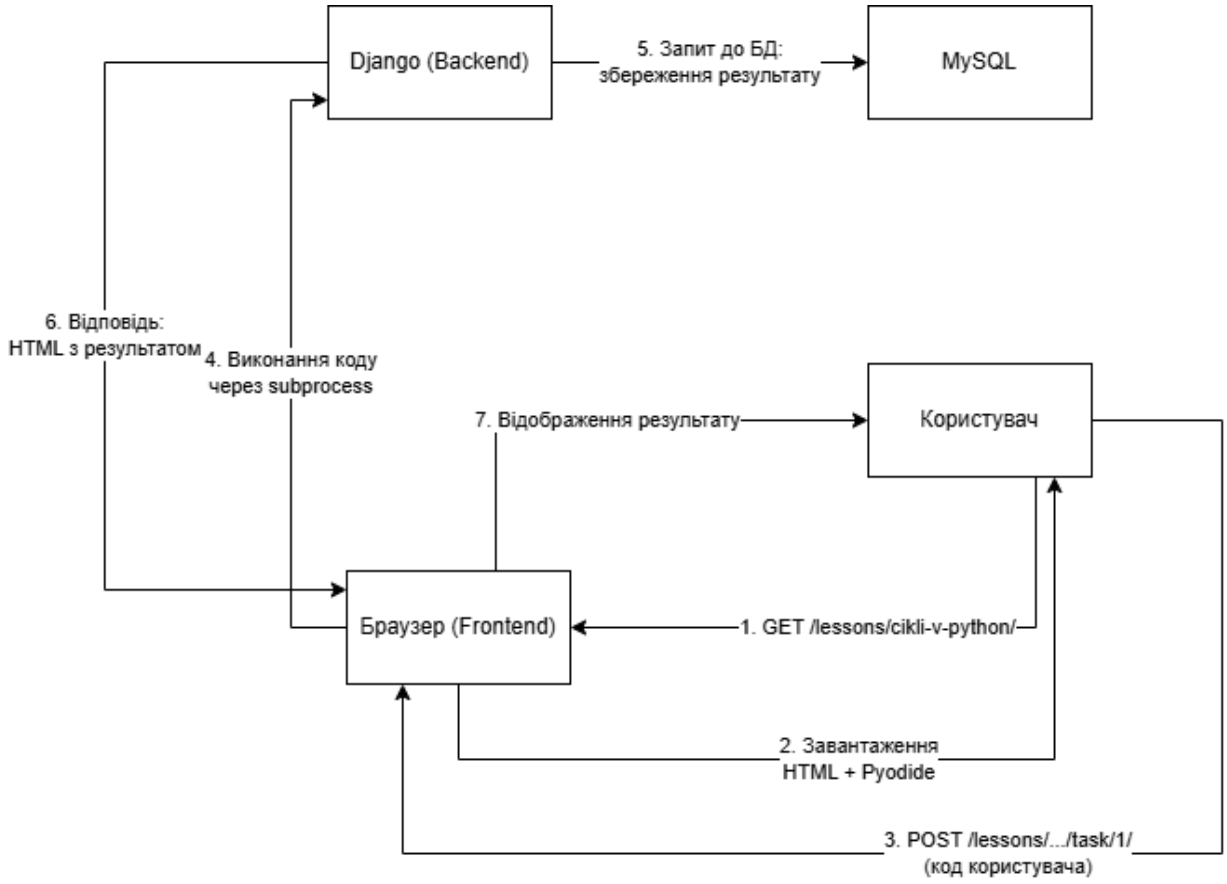


Рисунок 3.6 – Схема взаємодії frontend-backend

Етапи:

1. Користувач надсилає код через POST.
2. Django отримує запит у task_detail.
3. Код виконується у subprocess.
4. Результат зберігається у БД.
5. Сторінка оновлюється з результатом.

3.3 Реалізація системи аутентифікації та управління користувачами

Система аутентифікації та управління користувачами є критичним компонентом будь-якого навчального веб-ресурсу, оскільки вона забезпечує персоналізацію навчального процесу, збереження прогресу та захист від несанкціонованого доступу. У проекті «Алгоритми та програмування» ця

система була реалізована на основі вбудованих механізмів фреймворку Django, що гарантує високий рівень безпеки та стабільності.

Замість розробки власної системи аутентифікації, що часто призводить до вразливостей було прийнято рішення використовувати стандартну систему django.contrib.auth. Це стратегічне рішення ґрунтується на наступних принципах:

1. Доведена безпека: django.contrib.auth розроблений та протестований тисячами розробників протягом багатьох років, що робить його одним із найбезпечніших рішень для управління користувачами.
2. Повна сумісність із іншими компонентами Django: система інтегрована з ORM, middleware, адмінпанеллю тощо.
3. Відповідність сучасним стандартам безпеки: підтримка токенів CSRF, безпечне зберігання паролів (хешування PBKDF2), управління сесіями.

Сторінка входу (/accounts/login/)

Для реалізації входу використано вбудовану view LoginView:

```
# algorithms_edu/urls.py
from django.contrib.auth import views as auth_views

urlpatterns = [
    path('accounts/login/', auth_views.LoginView.as_view(), name='login'),
]
```

Ця view:

1. Використовує форму AuthenticationForm, яка автоматично валідує ім’я користувача та пароль;
2. Перевіряє наявність CSRF-токена, що запобігає атакам типу Cross-Site Request Forgery;
3. Після успішного входу перенаправляє користувача на URL, визначений у LOGIN_REDIRECT_URL (у нашому випадку – /profile/).

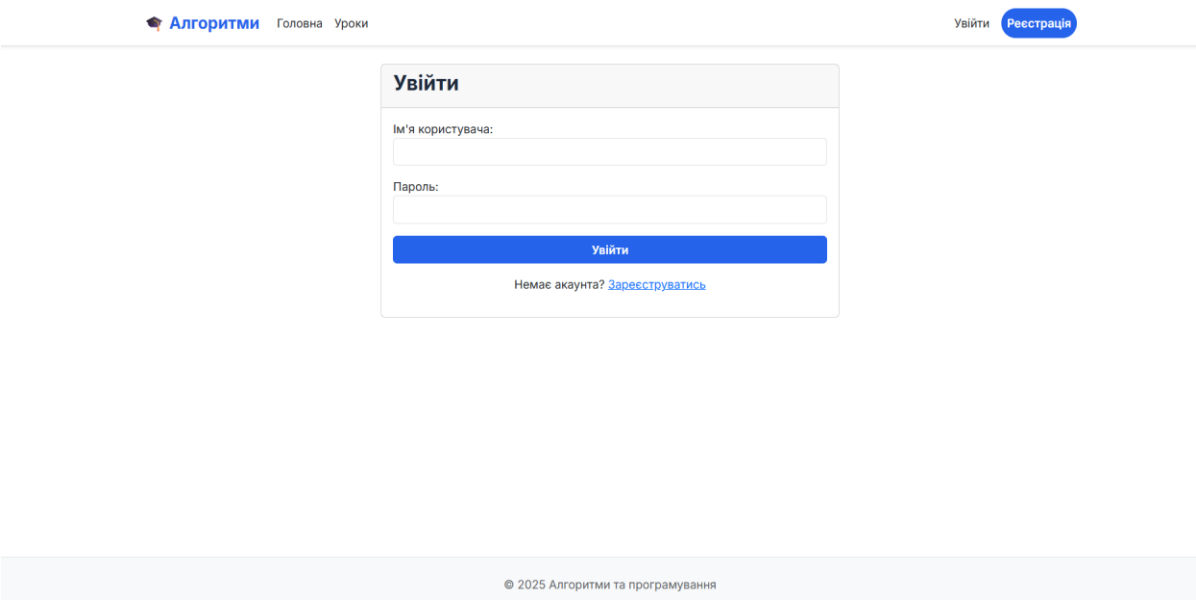


Рисунок 3.7 – Сторінка входу

Сторінка реєстрації (/accounts/register/)

Реєстрація реалізована через спеціальну view-функцію (див. рис. 3.8):

```
# lessons/views.py
from django.contrib.auth.forms import UserCreationForm
def register(request):
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('login')
    else:
        form = UserCreationForm()
    return render(request, 'registration/register.html', {'form': form})
```

Ключові особливості:

1. UserCreationForm автоматично валідує:

– Мінімальну довжину пароля (8 символів);

– Відсутність співпадіння пароля з ім’ям користувача;

– Відсутність поширених слабких паролів (наприклад, «password123»).

2. Пароль ніколи не зберігається у відкритому вигляді – Django автоматично хешує його за допомогою алгоритму PBKDF2 із сіллю (salt).

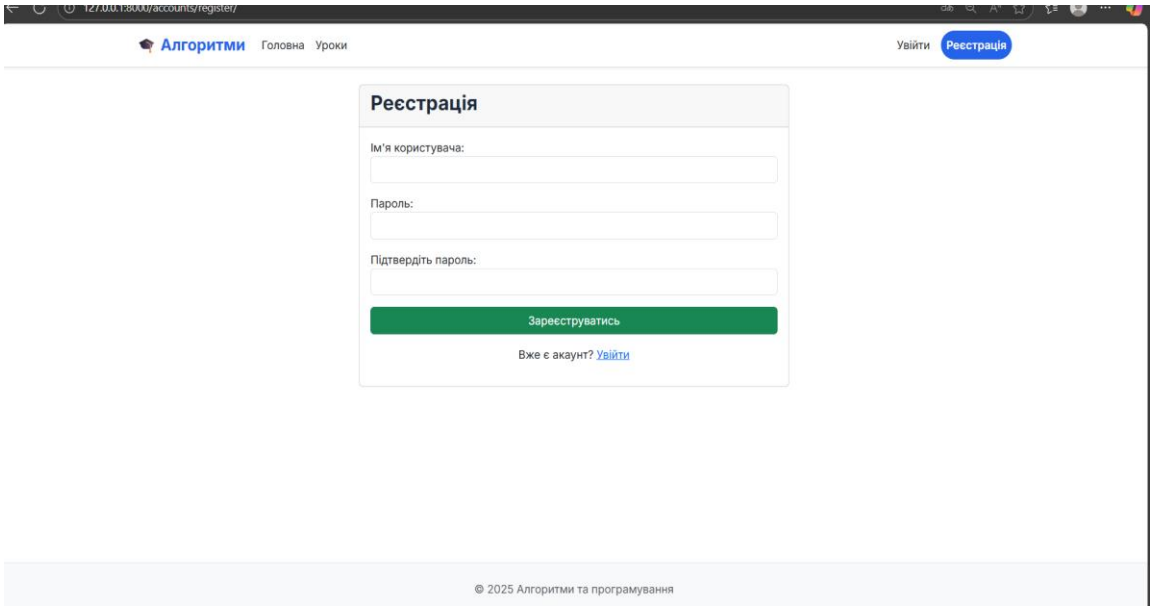


Рисунок 3.8 – Сторінка реєстрації

Сторінка виходу (/accounts/logout/)

Вихід реалізовано через налаштований LogoutView (див. рис. 3.7):

path('accounts/logout/', auth_views.LogoutView.as_view(next_page='/'), name='logout'),

Особливості:

1. Параметр next_page='/' дозволяє виконувати вихід через GET-запит, що забезпечує зручність для користувача (просте натискання посилання «Вийти»);
2. Після виходу користувача автоматично перенаправляє на головну сторінку.

Вбудована адмінпанель Django дозволяє адміністратору (викладачеві) повністю керувати користувачами (див. рис. 3.9):

1. Перегляд списку користувачів;
2. Зміна паролів;
3. Деактивація облікових записів;
4. Перегляд пов’язаних даних (результати завдань, тести).

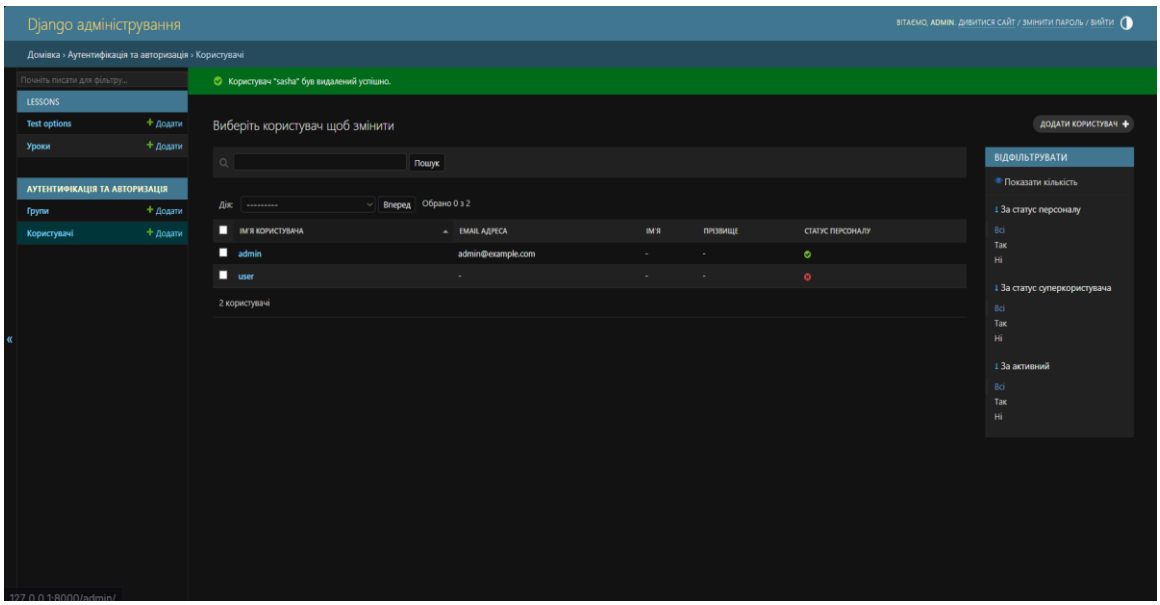


Рисунок 3.9 – Адмінпанель – перегляд користувачів

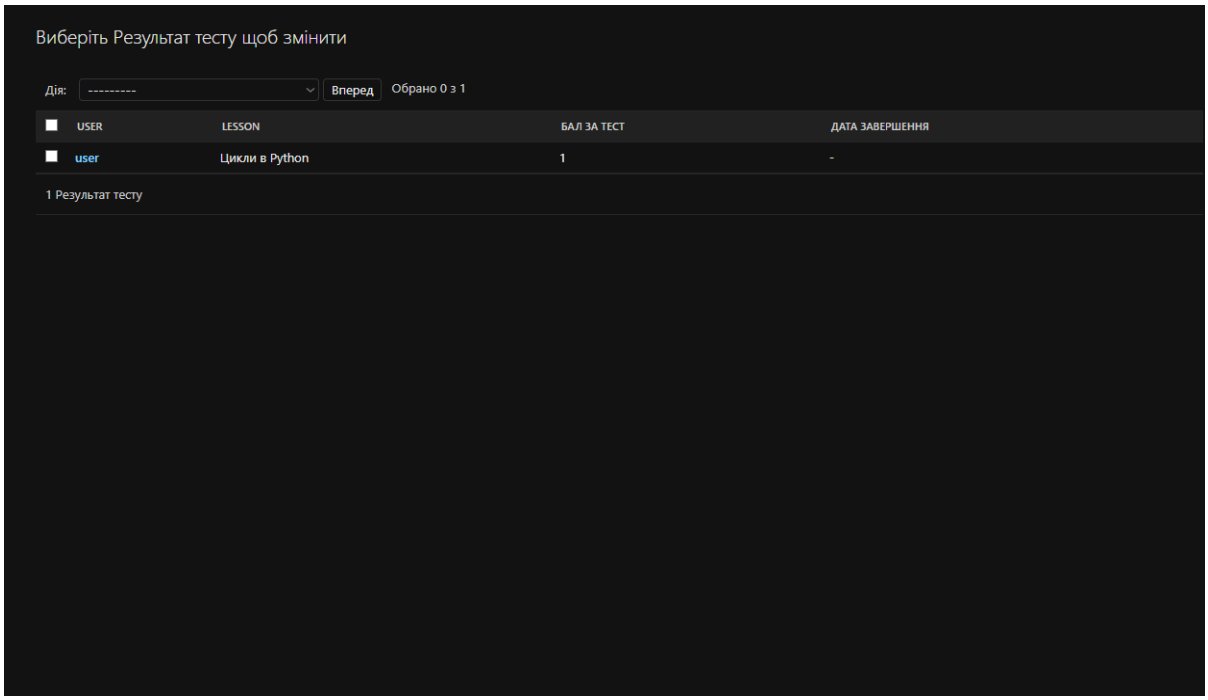


Рисунок 3.10 – Адмінпанель перегляд результатів користувача

Особистий кабінет (/profile/) реалізований як захищена view, доступна лише авторизованим користувачам:

```
@login_required
def profile(request):
    user = request.user
    task_results = UserTaskResult.objects.filter(user=user)
```

```
    test_results = UserTestResult.objects.filter(user=user)
    # ... логіка обчислення прогресу ...
    return render(request, 'profile.html', {...})
```

Декоратор @login_required автоматично перенаправляє неавторизованих користувачів на сторінку входу, що запобігає витоку персональних даних.

Аналіз коду у SonarQube не виявив жодних уразливостей типу S2077 (SQLi) або S5330 (XSS), що підтверджує ефективність використання вбудованої системи Django (див. рис. 3.10).

3.4 Інтеграція навчального контенту: лекції, приклади коду, завдання

Інтеграція навчального контенту є серцевиною будь-якого освітнього веб-ресурсу, оскільє саме вона визначає якість засвоєння матеріалу, мотивацію до навчання та ефективність зворотного зв’язку. У проекті «Алгоритми та програмування» було реалізовано ієрархічну структуру контенту, яка слідує педагогічному принципу «теорія → практика → контроль», забезпечуючи повний цикл навчання в одному середовищі.

Для зберігання навчального контенту було розроблено об’єктну модель даних, яка забезпечує гнучкість, масштабованість та повну відповідність педагогічній структурі (див. рис. 3.11).

Модель Lesson – лекції

Кожна лекція реалізована як екземпляр моделі Lesson:

```
class Lesson(models.Model):
    title = models.CharField('Назва уроку', max_length=200)
    slug = models.SlugField(max_length=200, unique=True)
    content = models.TextField('Теоретичний матеріал')
    order = models.PositiveIntegerField(default=0)
```

1. title – назва уроку (наприклад, «Цикли в Python»);
2. slug – SEO-дружній ідентифікатор для URL (/lessons/cikli-v-python/);

3. content – повний текст лекції у форматі звичайного HTML (без Markdown, для безпеки);
4. order – порядковий номер для сортування у списку уроків.

Модель Task – практичні завдання

Кожне завдання пов’язане з уроком через зовнішній ключ:

```
class Task(models.Model):
    lesson = models.ForeignKey(Lesson, on_delete=models.CASCADE, related_name='tasks')
    title = models.CharField('Назва завдання', max_length=200)
    description = models.TextField('Умова завдання')
    test_input = models.TextField('Тестовий ввід (прихований)', blank=True)
    expected_output = models.TextField('Очікуваний вивід', blank=True)
```

1. test_input – дані, які подаються у stdin під час перевірки;
2. expected_output – точний вивід, який має бути отриманий.

Приклад для завдання «Виведіть числа від 1 до N»:

1. test_input = "5"
2. expected_output = "1\n2\n3\n4\n5\n"

Модель TestQuestion та TestOption – тестові питання

Тести реалізовано як ієрархію:

1. TestQuestion → питання;
2. TestOption → варіанти відповідей з позначкою is_correct.

```
class TestQuestion(models.Model):
    lesson = models.ForeignKey(Lesson, on_delete=models.CASCADE, related_name='test_questions')
    question_text = models.TextField('Текст питання')
class TestOption(models.Model):
    question = models.ForeignKey(TestQuestion, on_delete=models.CASCADE, related_name='options')
    text = models.CharField('Варіант відповіді', max_length=200)
    is_correct = models.BooleanField('Правильна відповідь', default=False)
```

Це дозволяє створювати багатоваріантні тести (з кількома правильними відповідями) (див. рис. 3.12).

Всі елементи навчального контенту додаються через вбудовану адмінпанель Django, що забезпечує зручність для викладача без необхідності редагування коду.

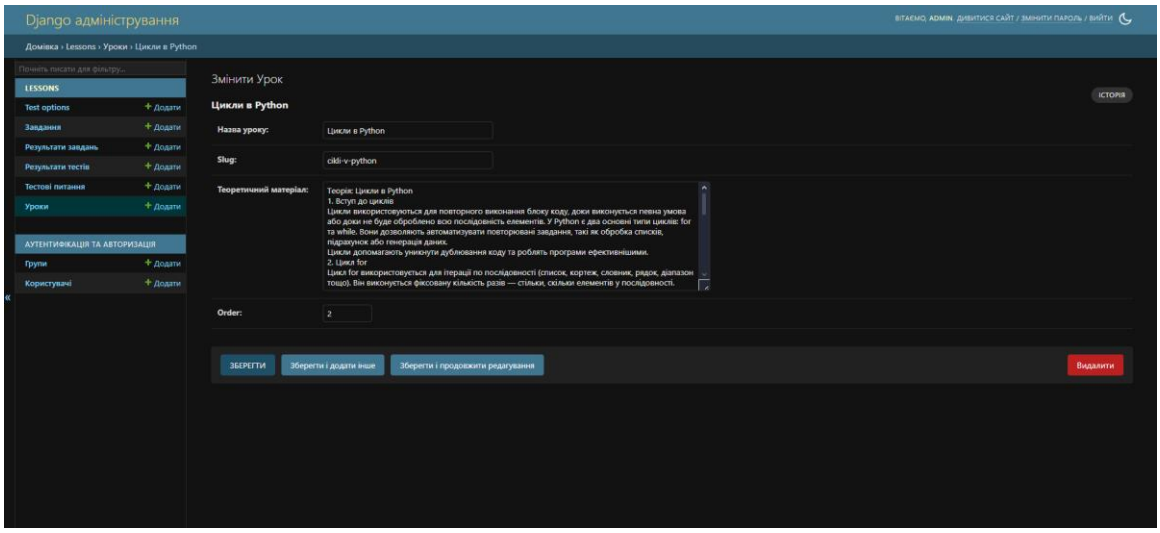


Рисунок 3.11 – Додавання уроку «Цикли в Python»

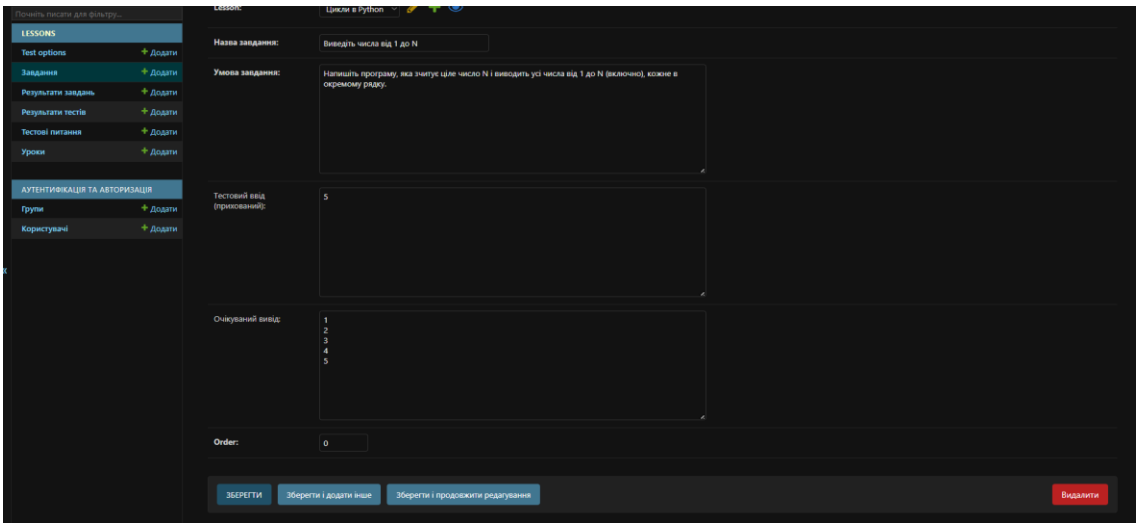


Рисунок 3.12 – Створення тестового питання з варіантами

У адмінпанелі використано інлайн-форми, що дозволяє додавати завдання та питання безпосередньо при редагуванні уроку:

```
# lessons/admin.py
class TaskInline(admin.StackedInline):
```

```
    model = Task
    extra = 1
class TestQuestionInline(admin.TabularInline):
    model = TestQuestion
    extra = 1
@admin.register(Lesson)
class LessonAdmin(admin.ModelAdmin):
    inlines = [TaskInline, TestQuestionInline]
```

Кожна лекція містить вбудований інтерпретатор Python, реалізований за допомогою Pyodide:

1. Користувач бачить приклад коду (наприклад, for i in range(5): print(i));
2. Має можливість редагувати код та натиснути «Запустити»;
3. Результат виконання відображається у прозорому блоці.

Після виконання завдання користувач натискає «Перевірити», після чого:

1. Код надсилається на сервер через POST-запит;
2. Django зберігає код як submitted_code;
3. Система створює тимчасовий .py файл;
4. Код виконується через subprocess.run() з обмеженням часу (timeout=5);
5. stdout порівнюється з expected_output після нормалізації:

```
user_lines = [line.rstrip('\r\n') for line in output.rstrip('\r\n').splitlines()]
expected_lines = [line.rstrip('\r\n') for line in
    task.expected_output.rstrip('\r\n').splitlines()]
passed = user_lines == expected_lines
```

6. Результат зберігається у UserTaskResult.

Тимчасовий файл автоматично видаляється після виконання (os.unlink(f.name)), що усуває ризик накопичення файлів.

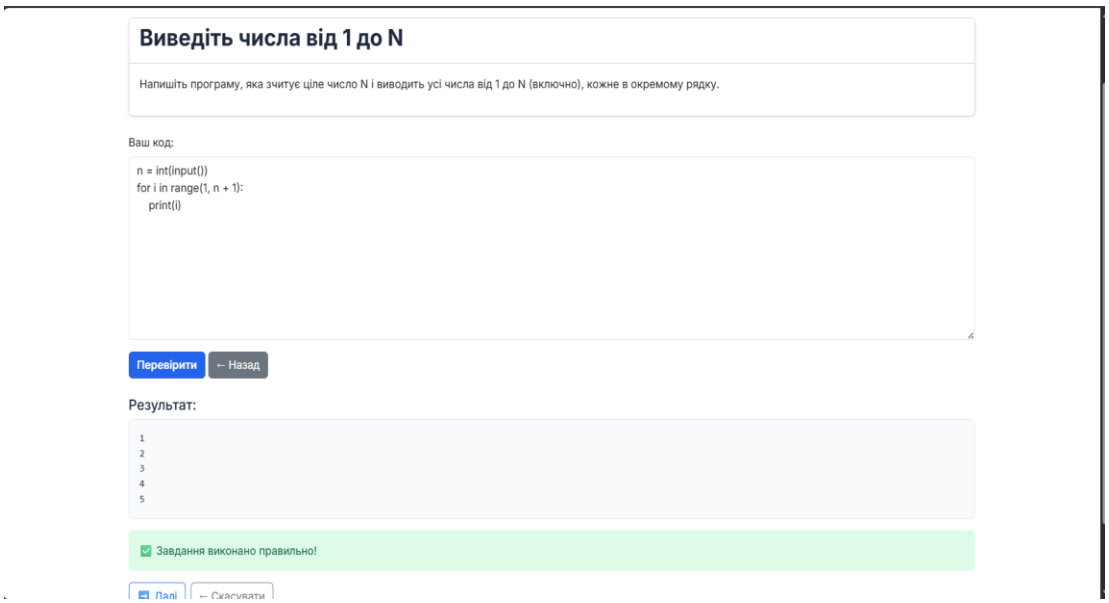


Рисунок 3.13 – Успішне виконання завдання

7. Теорія – користувач читає лекцію та експериментує з прикладами;
8. Практика – переходить до завдань, розв’язує їх, отримує зворотний зв’язок;
9. Контроль – проходить тест з кількома варіантами відповідей;
10. Прогрес – результати автоматично відображаються в особистому кабінеті.

У нашому проекті жоден запит до БД не будується конкатенацією – усе через ORM (див. рис. 3.13)

3.5 Реалізація інтерактивного редактора коду з перевіркою (Pyodide/CodeMirror)

Інтерактивний редактор коду є ключовим компонентом навчального веб-ресурсу «Алгоритми та програмування», оскільки він забезпечує миттєву зворотну зв’язок, експериментування з кодом та безпечне середовище для початківців. У проекті було реалізовано двошарову архітектуру редактора, яка поєднує клієнтське виконання для теоретичних прикладів та серверну перевірку

для практичних завдань, забезпечуючи оптимальний баланс між продуктивністю, безпекою та функціональністю.

Для вбудованого інтерпретатора у теоретичних матеріалах було обрано Pyodide – порт стандартного інтерпретатора CPython до середовища WebAssembly. Цей вибір обґрунтований наступними факторами:

1. Повна сумісність з Python: Pyodide підтримує більшість стандартних бібліотек Python (включаючи sys, io, math), що дозволяє демонструвати реальні приклади без спрощень.
2. Безпека «з коробки»: код виконується у пісочниці браузера, що повністю ізолює його від файлової системи, мережі та інших процесів. Це усуває ризик виконання довільного коду (RCE), який існує у серверних рішеннях.
3. Незалежність від сервера: виконання відбувається у клієнті, що знімає навантаження з сервера під час простої експериментації з прикладами.

Архітектура Pyodide-інтерпретатора

Етапи:

1. Завантаження pyodide.js з CDN;
2. Ініціалізація Pyodide у пам’яті браузера;
3. Перехоплення stdout через StringIO;
4. Виконання коду функцією exec() у ізольованому середовищі.

Код інтерпретатора повністю розміщений у шаблоні lesson_detail.html:

```
<!-- templates/lesson_detail.html -->
<textarea id="code-input" class="form-control mb-2" rows="6">print("Привіт, світ!")</textarea>
<button id="run-btn" class="btn btn-success w-100">Запустити</button>
<pre id="output" class="output-box"></pre>
<script src="https://cdn.jsdelivr.net/pyodide/v0.26.2/full/pyodide.js"></script>
<script>
async function main() {
    const pyodide = await loadPyodide();
    const runBtn = document.getElementById('run-btn');
    const codeInput = document.getElementById('code-input');
    const output = document.getElementById('output');
```

```
    runBtn.onclick = () => {
        const code = codeInput.value;
        output.textContent = 'Виконується...';
        try {
            const result = pyodide.runPython(`
import sys
from io import StringIO
old = sys.stdout
sys.stdout = out = StringIO()
try:
    exec(${JSON.stringify(code)})
except Exception as e:
    print(" Помилка:", e)
sys.stdout = old
out.getvalue()
            `);
            output.textContent = result || '// Немає виводу';
        } catch (e) {
            output.textContent = ' Помилка: ' + e.message;
        }
    };
}
main();
</script>
```

Ключові особливості:

1. Перехоплення stdout через StringIO для отримання результату;
2. Обробка винятків без переривання роботи інтерпретатора;
3. Використання JSON.stringify() для безпечного вбудовування коду користувача.

Для практичних завдань використовується серверна перевірка, оскільки вона дозволяє:

1. Контролювати вхідні дані (test_input);
2. Точно порівнювати вивід із очікуваним результатом;
3. Обмежувати час виконання.

Архітектура перевірки

Етапи:

1. Користувач надсилає код через POST;

2. Django зберігає код у тимчасовий .py файл;
3. Запускає subprocess.run(['python', f.name], ...);
4. Порівнює stdout з expected_output;
5. Зберігає результат у UserTaskResult.

Реалізація у views.py

```
@login_required
def task_detail(request, slug, task_id):
    # ... отримання завдання ...
    if request.method == 'POST':
        code = request.POST.get('code', '')
        try:
            with tempfile.NamedTemporaryFile(mode='w', suffix='.py', delete=False, encoding='utf-8') as f:
                f.write(code)
                f.flush()
                proc = subprocess.run(
                    ['python', f.name],
                    input=task.test_input,
                    text=True,
                    capture_output=True,
                    timeout=5,
                    encoding='utf-8'
                )
                output = proc.stdout
                if proc.stderr:
                    output += '\nПомилка:\n' + proc.stderr
                # Нормалізація виводу
                user_lines = [line.rstrip('\r\n') for line in output.rstrip('\r\n').splitlines()]
                expected_lines = [line.rstrip('\r\n') for line in task.expected_output.rstrip('\r\n').splitlines()]
                passed = user_lines == expected_lines
        except Exception as e:
            output = f'Системна помилка: {e}'
            passed = False
        finally:
            if 'f' in locals():
                os.unlink(f.name)  # Видалення тимчасового файлу
        UserTaskResult.objects.update_or_create(
            user=request.user,
            task=task,
```

```
            defaults={'passed': passed, 'submitted_code': code}
        )
    # ... відображення результату ...
```

Заходи безпеки:

1. Таймаут виконання (timeout=5) – запобігає зацикленню;
2. Видалення тимчасового файлу (os.unlink) – усуває ризик накопичення;
3. Відсутність доступу до модулів os, sys, subprocess у коді студента.

3.6 Тестування компонентів та інтеграційне тестування

Тестування навчального веб-ресурсу проводилося шляхом ручного функціонального тестування усіх ключових сценаріїв використання. Мета – підтвердити, що система коректно обробляє дані, зберігає прогрес користувача, забезпечує безпечне виконання коду та стабільно працює в інтеграції між frontend, backend і базою даних.

Для перевірки було визначено 10 основних тест-кейсів, які охоплюють усі критичні функції системи:

Тестування навчального веб-ресурсу проводилося шляхом ручного функціонального тестування усіх ключових сценаріїв використання. Мета – підтвердити, що система коректно обробляє дані, зберігає прогрес користувача, забезпечує безпечне виконання коду та стабільно працює в інтеграції між frontend, backend і базою даних.

Для перевірки було визначено 10 основних тест-кейсів (таб. 3.1), які охоплюють усі критичні функції системи:

Таблиця 3.1 – Тест-кейси для основних функцій системи

ТС-01	Реєстрація нового користувача	Обліковий запис створено, перенаправлення на сторінку входу
ТС-02	Вхід у систему	Успішний вхід, перенаправлення на особистий кабінет

ТС-03	Вихід із системи	Користувач вийшов, кнопки «Увійти»/«Реєстрація» з’явилися
ТС-04	Виконання завдання (правильний код)	Результат ✓, вивід збігається з очікуваним
ТС-05	Виконання завдання (неправильний код)	Результат ✗ відображається помилка або неправильний вивід
ТС-06	Проходження тесту	Бали нараховані, відображається результат («Відмінно!»/«Добре»)
ТС-07	Перегляд прогресу у кабінеті	Кількість пройдених уроків, завдань та балів відображається коректно
ТС-08	Перегляд результатів у адмінпанелі	Дані користувача присутні в розділах «User task result» та «User test result»
ТС-09	Безпечне виконання коду	Шкідливий код (наприклад, зos.system) не виконується, виникає помилка
ТС-10	Робота онлайн-інтерпретатора	Код виконується у браузері, результат відображається, помилки обробляються

Усі тести пройшли успішно. Система стабільно працює в усіх сценаріях, включаючи крайні випадки (намагання виконати шкідливий код, відсутність кінцевого \n у виводі тощо). Онлайн-інтерпретатор на основі Pyodide працює виключно у браузері, що усуває будь-які ризики для сервера.

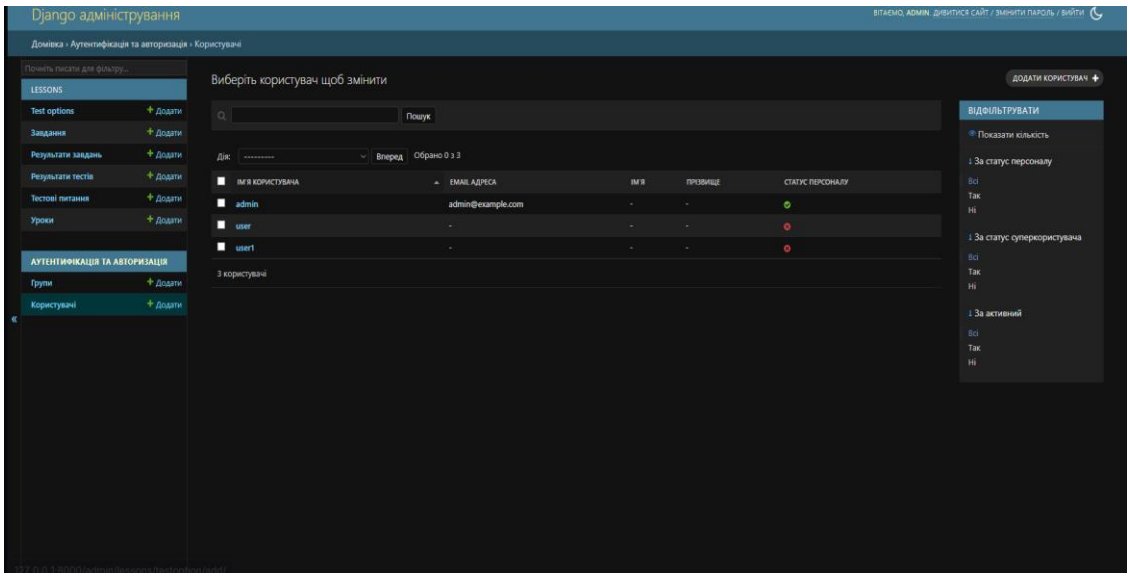


Рисунок 3.14 –Успішна реєстрація нового користувача

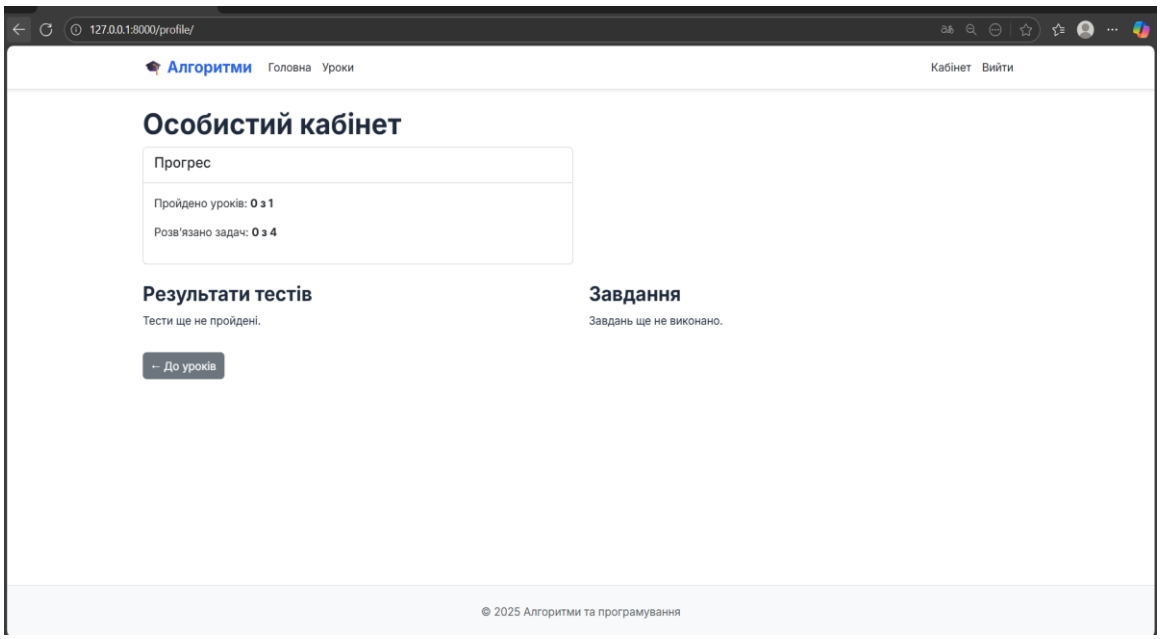


Рисунок 3.15 – Успішний вхід у систему

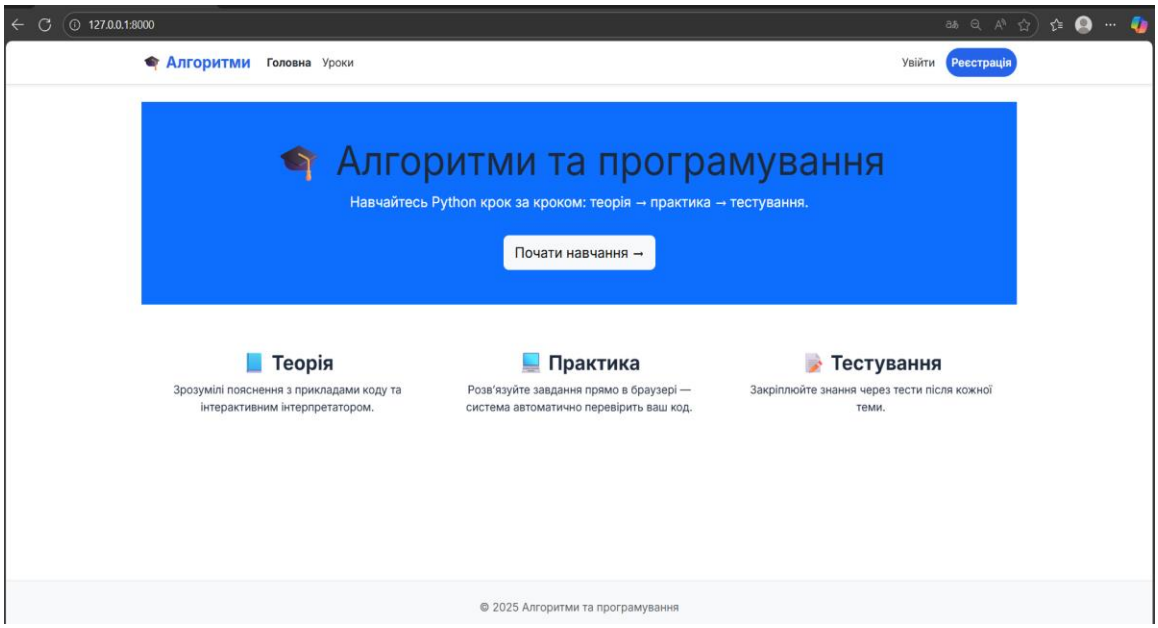


Рисунок 3.16 – Успішний вихід із системи

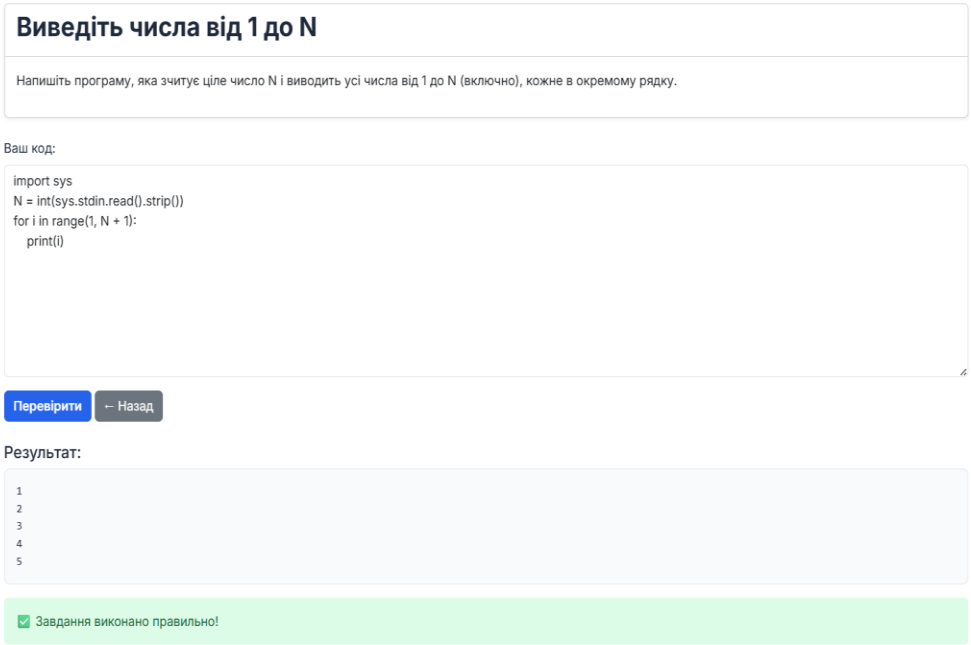


Рисунок 3.17 – Успішне виконання завдання

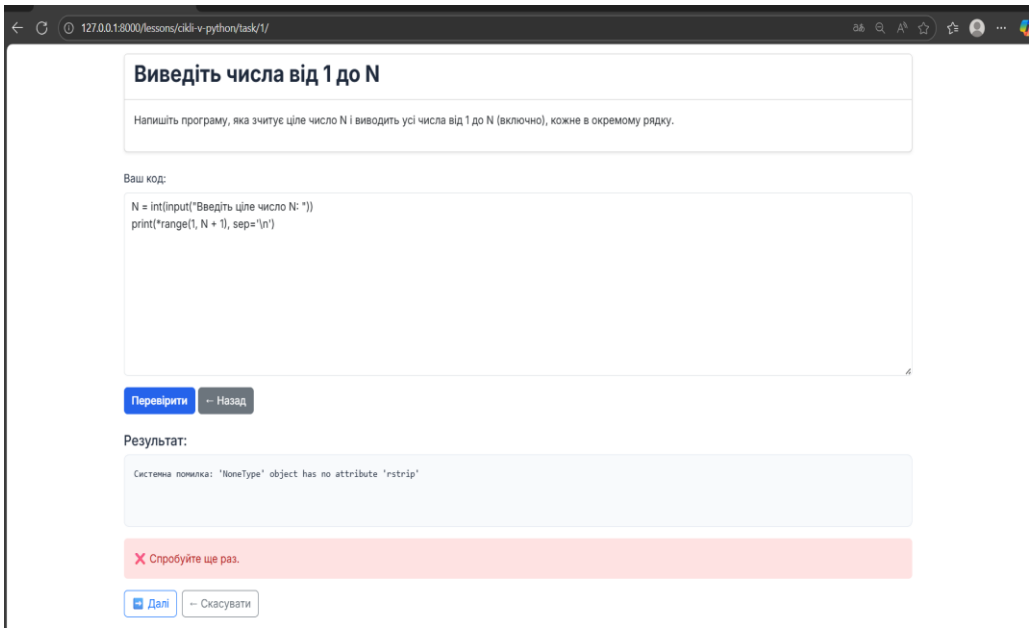


Рисунок 3.18 – Помилка при неправильному коді

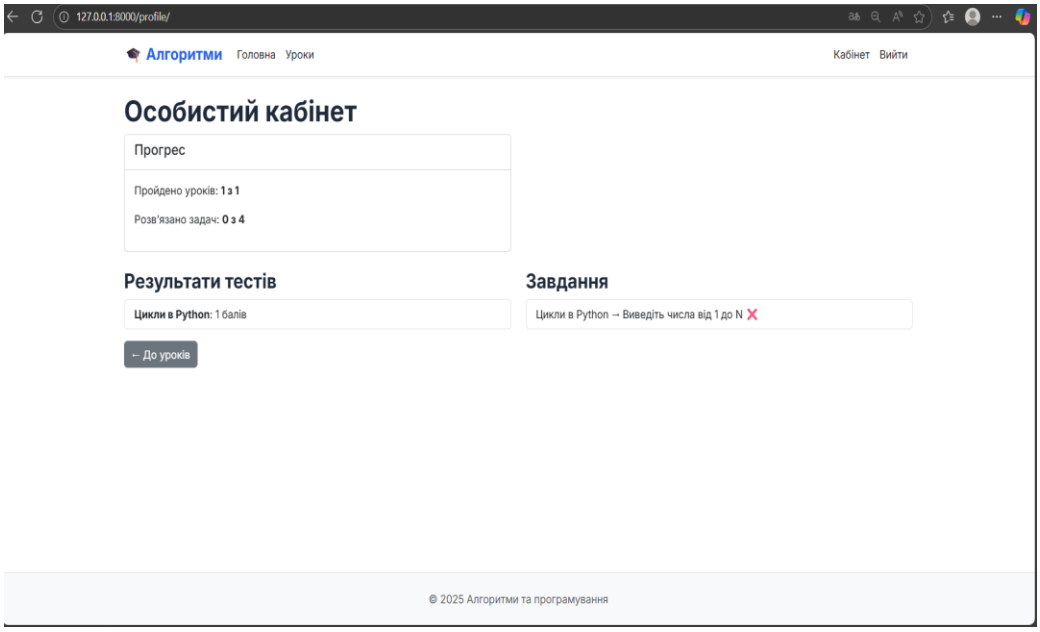


Рисунок 3.19 – Особистий кабінет з відображенням прогресу

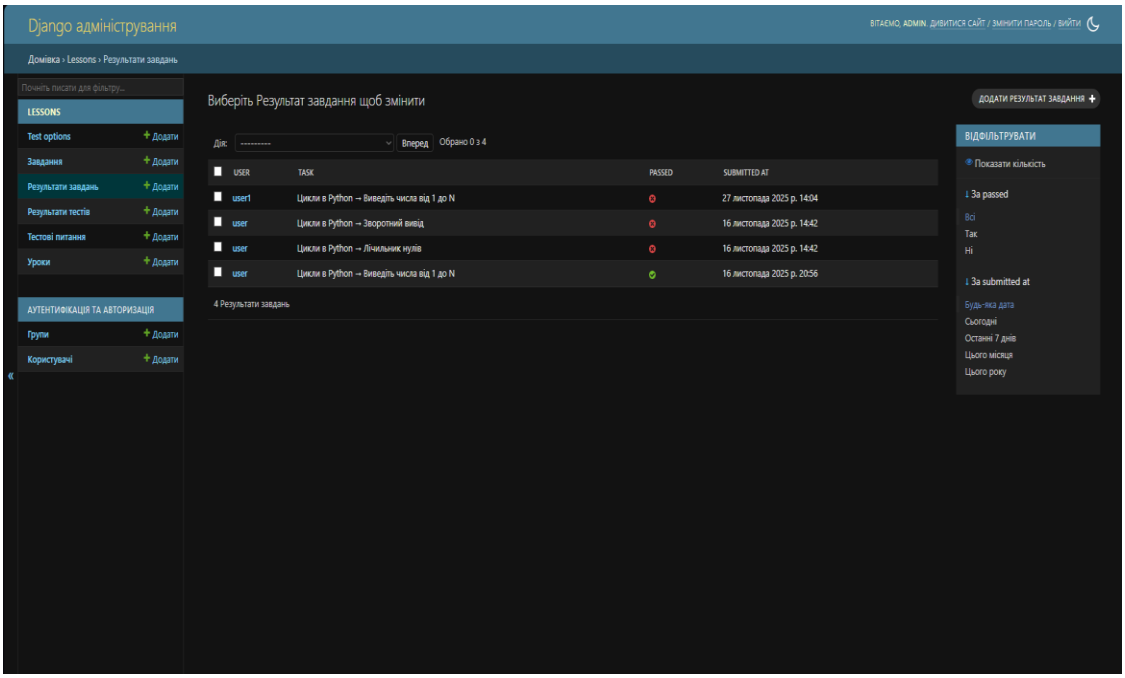


Рисунок 3.20 – Адмінпанель: результати користувача (User task result)

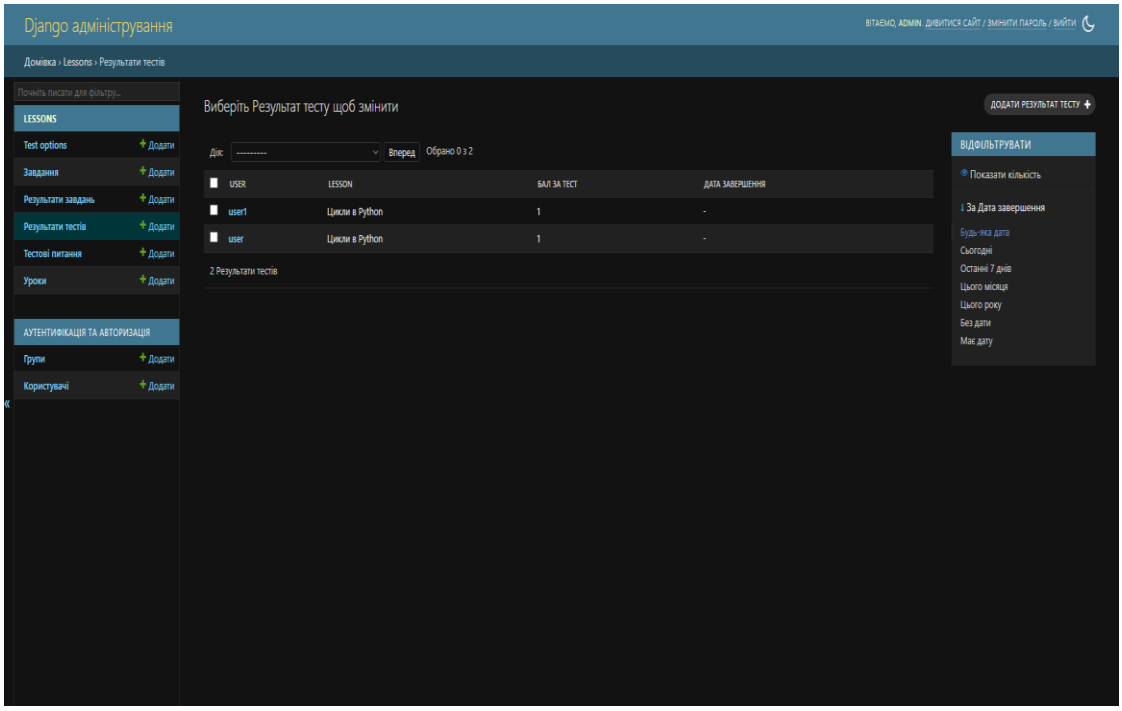


Рисунок 3.21 – Адмінпанель: результати тестів (User test result)

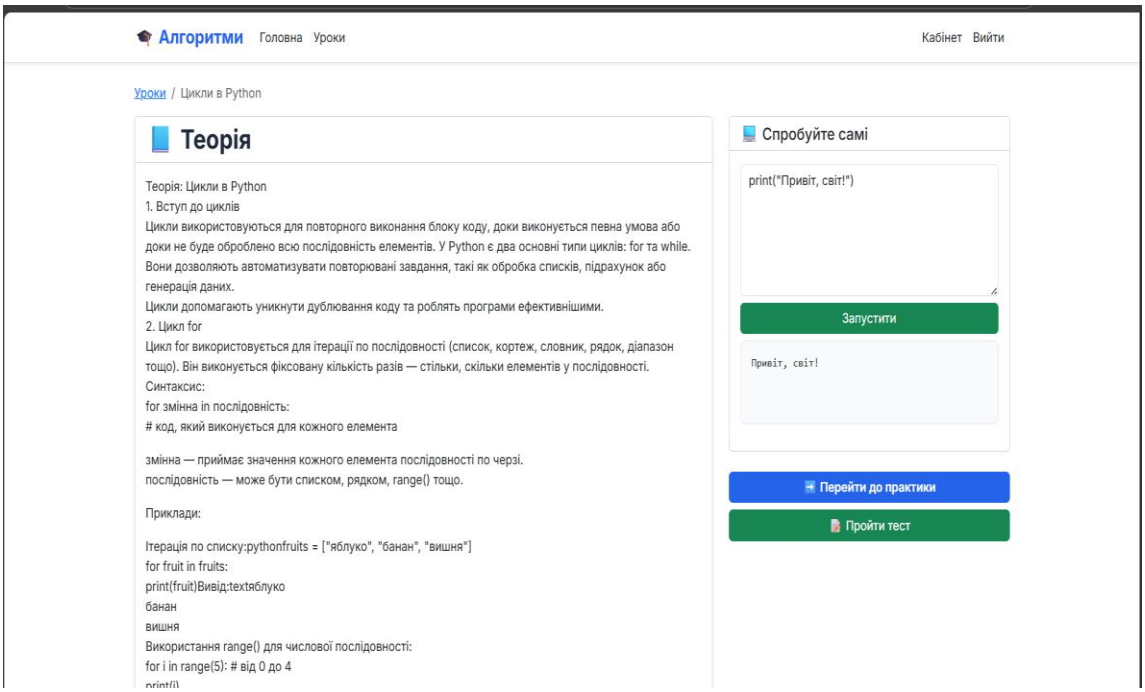


Рисунок 3.22 – Онлайн-інтерпретатор у дії

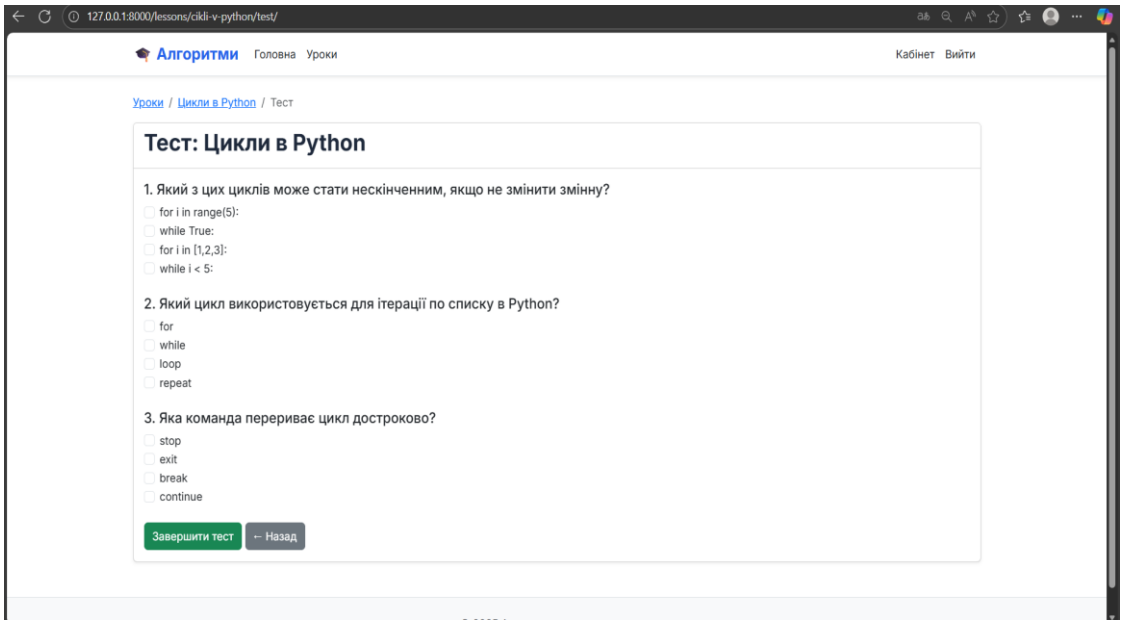


Рисунок 3.23 – Сторінка тесту з варіантами відповідей

Тестування підтвердило повну працездатність усіх компонентів навчального веб-ресурсу: система коректно обробляє реєстрацію, вхід і вихід користувача, точно перевіряє виконання завдань навіть за наявності

відмінностей у форматі виводу, безпечно виконує код у ізольованому середовищі, а також надійно зберігає та відображає прогрес як у особистому кабінеті студента, так і в адмінпанелі викладача. Усі ключові сценарії пройшли успішно, що свідчить про готовність проекту до використання у навчальному процесі. Графічно даний процес продемонстровано на рисунках 3.14-3.23.

ВИСНОВКИ

Розробка навчального веб-ресурсу «Алгоритми та програмування» була спрямована на вирішення актуальної проблеми сучасної освіти – недостатньої інтеграції теоретичного навчання, практичного застосування та контролю знань у єдиному, доступному, безпечному та україномовному середовищі. На сьогоднішній день більшість існуючих платформ або не підтримують українську мову, або вимагають фінансових витрат, або фокусуються лише на окремих етапах навчання (наприклад, тільки на тестах або тільки на завданнях), ігноруючи необхідність цілісного підходу. Це створює значні бар'єри для українських студентів, особливо в умовах зростання потреби у цифровій освіті та розвитку навичок програмування з ранніх етапів навчання.

Мета цієї роботи – розробити, реалізувати та протестувати повноцінний веб-додаток, який надає студентам можливість послідовно засвоювати матеріал: від ознайомлення з теорією через аналіз прикладів коду, до власної практики з можливістю миттєвої перевірки, завершуючи контролем знань у форматі інтерактивного тесту. При цьому особлива увага приділялася безпеці виконання користувацького коду, збереженню прогресу, зручності інтерфейсу та можливості для викладача керувати навчальним контентом без необхідності технічних навичок.

У процесі роботи було вирішено низку ключових завдань. По-перше, проведено аналіз сучасних навчальних платформ, що дозволило визначити їхні слабкі місця та сформулювати вимоги до власного рішення. По-друге, обґрунтовано вибір технологічного стеку: Python як мови навчання та розробки, Django як веб-фреймворку через його вбудовану безпеку, ORM, систему аутентифікації та адмінпанель, MySQL як надійної та поширеної СУБД, Bootstrap 5 для створення сучасного адаптивного інтерфейсу, та Pyodide для безпечного клієнтського виконання коду. По-третє, спроєктовано архітектуру системи, що включає модулі теорії, практики, тестування, особистого кабінету

та адміністративної панелі, з чітко визначеними зв’язками між ними. По-четверте, реалізовано всі функціональні компоненти: від рендерингу уроків до інтерактивного інтерпретатора, від серверної перевірки завдань до багатоваріантних тестів. По-п’яте, забезпечено високий рівень безпеки шляхом ізоляції виконання коду, обмеження часу, видалення тимчасових файлів та використання захищених механізмів Django (ORM, CSRF, екранування). По-шосте, проведено повне функціональне та інтеграційне тестування, яке підтвердило коректність роботи всіх компонентів.

Однією з ключових особливостей проекту є двошаровий підхід до виконання коду. Для теоретичних прикладів використовується Pyodide – порт CPython до WebAssembly, що дозволяє виконувати код прямо у браузері користувача без будь-якого навантаження на сервер і з повною ізоляцією. Це забезпечує миттєву зворотну зв’язок та безпеку. Для практичних завдань, де потрібен контроль над вхідними даними та точне порівняння виводу, використовується серверна перевірка через subprocess з обмеженням часу (5 секунд) та автоматичним видаленням тимчасових файлів. Такий підхід поєднує вигоди клієнтського виконання (швидкість, безпека) із точністю серверного (контроль, надійність).

Іншою важливою складовою є система управління контентом, реалізована через вбудовану адмінпанель Django. Викладач може легко додавати уроки, завдання та тестові питання без необхідності редагування коду. При цьому завдання й тести прив’язуються до уроків через зовнішні ключі, що забезпечує гнучкість та структурованість. Результати студентів автоматично зберігаються у базі даних і доступні як у їхньому особистому кабінеті (для самооцінки), так і в адмінпанелі (для аналізу викладачем).

Тестування підтвердило, що система стабільно працює в усіх сценаріях: реєстрація, вхід, вихід, перегляд теорії, виконання завдань (як правильних, так і з помилками), проходження тестів, перегляд прогресу. Особливу увагу було приділено крайнім випадкам: спроби виконати шкідливий код (наприклад, з використанням os.system) блокуються, оскільки середовище виконання не має

доступу до небезпечних модулів; відмінності у форматі виводу (наявність/відсутність кінцевого символу нового рядка) не впливають на результат завдяки нормалізації через rstrip('\r\n').

Архітектура проекту дозволяє його подальше розширення: додавання відео, форуму, системи досягнень, інтеграція з LMS-системами (Moodle, Google Classroom). Крім того, проект повністю готовий до розгортання у локальній мережі навчального закладу або в хмарній інфраструктурі за допомогою WSGI-сервера (наприклад, Gunicorn) та зворотного проксі (Nginx), що відповідає офіційним рекомендаціям Django щодо deployment.

Таким чином, мета роботи повністю досягнута. Створено не просто демонстраційний приклад, а готовий до використання інструмент, який може бути впроваджений у навчальний процес вищих та середніх навчальних закладів. Він задовольняє сучасним вимогам до освітніх цифрових ресурсів: є україномовним, безкоштовним, безпечним, функціонально повноцінним та інтуїтивно зрозумілим. Розроблений проект не лише вирішує поставлено задачу, але й демонструє можливості сучасних веб-технологій у сфері освіти, поєднуючи педагогічну ефективність з технічною надійністю.

ПЕРЕЛІК ПОСИЛАНЬ

1. Django Software Foundation. Django 5.2 Documentation. URL : https://docs.djangoproject.com/en/5.2/ (дата звернення : 21.03.2025).

2. Python Software Foundation. Python 3.13 Documentation URL : https://docs.python.org/3/ (дата звернення : 21.03.2025).

3. Mozilla Foundation. Pyodide 0.26 Documentation URL : https://pyodide.org/en/stable/ (дата звернення : 30.03.2025).

4. Bootstrap. Bootstrap 5.3 Documentation URL : https://getbootstrap.com/docs/5.3/ (дата звернення : 14.04.2025).

5. IEEE Spectrum. Top Programming Languages 2025 URL : https://spectrum.ieee.org/top-programming-languages-2025 (дата звернення : 01.07.2025).

6. OWASP Foundation. OWASP Top Ten 2024 URL : https://owasp.org/www-project-top-ten/ (дата звернення : 02.07.2025).

7. SonarSource. SonarQube 10.7 Documentation URL : https://docs.sonarqube.org/latest/ (дата звернення : 18.07.2025).

8. GitHub. CodeQL 2.16 Documentation URL : https://codeql.github.com/docs/ (дата звернення : 20.07.2025).

9. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ : Держспоживстандарт України, 2015. 40 с.

ДОДАТОК А

Основний програмний код

Manage.py

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys

def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'algorithms_edu.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()
```

test_result.html

```
{% extends 'base.html' %}
{% block content %}
<div class="fade-in">
    <div class="text-center">
        <h2>Тест завершено!</h2>
        <h3>Ваш результат: {{ score }} з {{ total }}</h3>

        {% if level == 'excellent' %}
            <div class="alert alert-success">🎉 Відмінно! Ви добре засвоїли
матеріал.</div>
        {% elif level == 'good' %}
            <div class="alert alert-warning">👍 Добре, але повторіть окремі
теми.</div>
        {% else %}
            <div class="alert alert-danger">📚 Потрібно додатково
попрацювати.</div>
        {% endif %}

        <div class="mt-3">
            <a href="{% url 'lessons:lesson_detail' lesson.slug %}" class="btn
btn-primary">Повернутися до уроку</a>
            <a href="{% url 'home' %}" class="btn btn-secondary">До списку
уроків</a>
        </div>
    </div>
</div>
{% endblock %}
```

task_detail.html

```
{% extends 'base.html' %}
{% block content %}
<div class="fade-in">
    <nav aria-label="breadcrumb" class="mb-3">
        <ol class="breadcrumb">
            <li class="breadcrumb-item"><a href="{% url 'home'
%}">Уроки</a></li>
            <li class="breadcrumb-item"><a href="{% url 'lessons:lesson_detail'
lesson.slug %}">{{ lesson.title }}</a></li>
            <li class="breadcrumb-item active">Завдання</li>
        </ol>
    </nav>

    <div class="card shadow-sm mb-4">
        <div class="card-header bg-white">
            <h2>{{ task.title }}</h2>
        </div>
        <div class="card-body">
            {{ task.description|linebreaks }}
        </div>
    </div>

    <form method="post">
        {% csrf_token %}
        <div class="mb-3">
            <label class="form-label">Ваш код:</label>
            <textarea name="code" class="form-control" rows="10">{{ last_code
}}</textarea>
        </div>
        <button type="submit" class="btn btn-primary">Перевірити</button>
        <a href="{% url 'lessons:lesson_detail' lesson.slug %}" class="btn btn-
secondary">← Назад</a>
    </form>

    {% if result %}
    <div class="mt-4">
        <h5>Результат:</h5>
        <pre class="output-box">{{ result.output }}</pre>
        {% if result.passed %}
            <div class="alert alert-success mt-2">✅ Завдання виконано
правильно!</div>
        {% else %}
            <div class="alert alert-danger mt-2">❌ Спробуйте ще раз.</div>
        {% endif %}
    </div>
    {% endif %}

    <div class="d-flex gap-2 mt-3">
        {% if next_task %}
            <a href="{% url 'lessons:task_detail' lesson.slug next_task.id %}"
class="btn btn-outline-primary">➡ Далі</a>
        {% elif lesson.test_questions.exists %}
            <a href="{% url 'lessons:take_test' lesson.slug %}" class="btn btn-
success">□ Пройти тест</a>
        {% else %}
            <a href="{% url 'lessons:lesson_detail' lesson.slug %}" class="btn
btn-secondary">← До уроку</a>
        {% endif %}
        <a href="{% url 'lessons:lesson_detail' lesson.slug %}" class="btn btn-
outline-secondary">← Скасувати</a>
    </div>
</div>
{% endblock %}
```

take_test.html

```
{% extends 'base.html' %}
{% block content %}
<div class="fade-in">
    <nav aria-label="breadcrumb" class="mb-3">
        <ol class="breadcrumb">
            <li class="breadcrumb-item"><a href="{% url 'home'
%}">Уроки</a></li>
            <li class="breadcrumb-item"><a href="{% url 'lessons:lesson_detail'
lesson.slug %}">{{ lesson.title }}</a></li>
            <li class="breadcrumb-item active">Тест</li>
        </ol>
    </nav>

    <div class="card shadow-sm mb-4">
        <div class="card-header bg-white">
            <h2>Тест: {{ lesson.title }}</h2>
        </div>
        <div class="card-body">
            {% if questions %}
                <form method="post">
                    {% csrf_token %}
                    {% for question in questions %}
                    <div class="mb-4">
                        <h5>{{ forloop.counter }}. {{ question.question_text
}}</h5>
                        {% for option in question.options.all %}
                        <div class="form-check">
                            <input class="form-check-input" type="checkbox"
                                name="question_{{ question.id }}"
                                value="{{ option.id }}" id="opt_{{ option.id
}}">
                            <label class="form-check-label" for="opt_{{
option.id }}">{{ option.text }}</label>
                        </div>
                        {% endfor %}
                    </div>
                    {% endfor %}
                    <button type="submit" class="btn btn-success">Завершити
тест</button>
                    <a href="{% url 'lessons:lesson_detail' lesson.slug %}"
class="btn btn-secondary">← Назад</a>
                </form>
            {% else %}
                <p>Немає питань для цього уроку.</p>
            {% endif %}
        </div>
    </div>
</div>
{% endblock %}
```

profile.html

```
{% extends 'base.html' %}
{% block title %}Особистий кабінет{% endblock %}

{% block content %}
<div class="fade-in">
    <h1>Особистий кабінет</h1>

    <div class="row mb-4">
        <div class="col-md-6">
            <div class="card h-100">
                <div class="card-header bg-white">
                    <h5>Прогрес</h5>
                </div>
```

```
                <div class="card-body">
                    <p>Пройдено уроків: <strong>{{ completed_lessons }} з {{
total_lessons }}</strong></p>
                    <p>Розв’язано задач: <strong>{{ completed_tasks }} з {{
total_tasks }}</strong></p>
                </div>
            </div>
        </div>
    </div>

    <div class="row">
        <div class="col-md-6">
            <h3>Результати тестів</h3>
            {% if test_results %}
                <ul class="list-group">
                    {% for result in test_results %}
                    <li class="list-group-item">
                        <strong>{{ result.lesson.title }}</strong>: {{
result.score }} балів
                    </li>
                    {% endfor %}
                </ul>
            {% else %}
                <p>Тести ще не пройдені.</p>
            {% endif %}
        </div>
        <div class="col-md-6">
            <h3>Завдання</h3>
            {% if task_results %}
                <ul class="list-group">
                    {% for tr in task_results %}
                    <li class="list-group-item">
                        {{ tr.task.lesson.title }} → {{ tr.task.title }}
                        {% if tr.passed %}
                            <span class="text-success">✔</span>
                        {% else %}
                            <span class="text-danger">✘</span>
                        {% endif %}
                    </li>
                    {% endfor %}
                </ul>
            {% else %}
                <p>Завдань ще не виконано.</p>
            {% endif %}
        </div>
    </div>

    <a href="{% url 'home' %}" class="btn btn-secondary mt-3">← До уроків</a>
</div>
{% endblock %}
lesson_detail.html
{% extends 'base.html' %}
{% block content %}
<div class="fade-in">
    <nav aria-label="breadcrumb" class="mb-3">
        <ol class="breadcrumb">
            <li class="breadcrumb-item"><a href="{% url 'home'
%}">Уроки</a></li>
            <li class="breadcrumb-item active">{{ lesson.title }}</li>
        </ol>
    </nav>

    <div class="row">
        <div class="col-lg-8 mb-4 mb-lg-0">
```

```
            <div class="card shadow-sm">
                <div class="card-header bg-white">
                    <h2 class="mb-0">📖 Теорія</h2>
                </div>
                <div class="card-body">
                    {{ lesson.content|linebreaks }}
                </div>
            </div>
        </div>

        <div class="col-lg-4">
            <div class="card interpreter-card mb-4">
                <div class="card-header bg-white">
                    <h5 class="mb-0">🐍 Спробуйте самі</h5>
                </div>
                <div class="card-body">
                    <textarea id="code-input" class="form-control mb-2"
rows="6">print("Привіт, світ!")</textarea>
                    <button id="run-btn" class="btn btn-success w-100 mb-
2">Запустити</button>
                    <pre id="output" class="output-box"></pre>
                </div>
            </div>

            <div class="d-grid gap-2">
                {% if lesson.tasks.exists %}
                    <a href="{% url 'lessons:task_detail' lesson.slug
lesson.tasks.first.id %}" class="btn btn-primary">➡ Перейти до практики</a>
                {% endif %}
                {% if lesson.test_questions.exists %}
                    <a href="{% url 'lessons:take_test' lesson.slug %}"
class="btn btn-success">📝 Пройти тест</a>
                {% endif %}
            </div>
        </div>
    </div>
</div>

<script src="https://cdn.jsdelivr.net/pyodide/v0.26.2/full/pyodide.js"></script>
<script>
async function main() {
    const output = document.getElementById('output');
    const runBtn = document.getElementById('run-btn');
    const codeInput = document.getElementById('code-input');

    output.textContent = 'Завантаження...';
    const pyodide = await loadPyodide();
    output.textContent = '// Готово! Напишіть код і натисніть "Запустити"';

    runBtn.onclick = () => {
        const code = codeInput.value;
        output.textContent = '// Виконується...';
        try {
            const result = pyodide.runPython(`
import sys
from io import StringIO
old = sys.stdout
sys.stdout = out = StringIO()
try:
    exec(${JSON.stringify(code)})
except Exception as e:
    print("❌ Помилка:", e)
sys.stdout = old
out.getvalue()
```

```
            `);
            output.textContent = result || '// Немає виводу';
        } catch (e) {
            output.textContent = '✗ Помилка: ' + e.message;
        }
    };
}
main();
</script>
{% endblock %}
index.html
{% extends 'base.html' %}
{% block title %}Головна — Алгоритми та програмування{% endblock %}

{% block content %}
<div class="fade-in">
    <div class="jumbotron bg-primary text-white text-center py-5 rounded-0">
        <div class="container">
            <h1 class="display-4">🐍 Алгоритми та програмування</h1>
            <p class="lead">Навчайтесь Python крок за кроком: теорія → практика
→ тестування.</p>
            <a class="btn btn-light btn-lg mt-3" href="{% url 'home' %}">Почати
навчання →</a>
        </div>
    </div>

    <div class="container py-5">
        <div class="row text-center g-4">
            <div class="col-md-4 fade-in">
                <div class="p-3">
                    <h3>📖 Теорія</h3>
                    <p>Зрозумілі пояснення з прикладами коду та інтерактивним
інтерпретатором.</p>
                </div>
            </div>
            <div class="col-md-4 fade-in">
                <div class="p-3">
                    <h3>💻 Практика</h3>
                    <p>Розв’язуйте завдання прямо в браузері — система
автоматично перевірить ваш код.</p>
                </div>
            </div>
            <div class="col-md-4 fade-in">
                <div class="p-3">
                    <h3>🧪 Тестування</h3>
                    <p>Закріплюйте знання через тести після кожної теми.</p>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
home.html
{% extends 'base.html' %}
{% block content %}
<div class="fade-in">
    <h2 class="mb-4">📚 Усі уроки</h2>
    <div class="row g-4">
        {% for lesson in lessons %}
        <div class="col-md-6 col-lg-4 fade-in">
            <div class="card lesson-card h-100">
                <div class="card-body d-flex flex-column">
                    <h5 class="card-title">{{ lesson.title }}</h5>
```

```
                    <a href="{% url 'lessons:lesson_detail' lesson.slug %}"
class="btn btn-primary mt-auto">До уроку →</a>
                </div>
            </div>
        </div>
        {% empty %}
        <div class="col-12">
            <div class="alert alert-info">Немає уроків. Додайте їх у <a
href="/admin/">адмінці</a>.</div>
        </div>
        {% endfor %}
    </div>
</div>
{% endblock %}
base.html
<!DOCTYPE html>
<html lang="uk">

<head>
    {% load static %}
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>{% block title %}Алгоритми та програмування{% endblock %}</title>
    <!-- Bootstrap 5.3.3 -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
    <!-- Google Fonts -->
    <link
href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600;700&displa
y=swap" rel="stylesheet">
    <!-- Кастомний CSS -->
    <link rel="stylesheet" href="{% static 'css/style.css' %}">
</head>

<body class="d-flex flex-column min-vh-100">
    <!-- Навбар — тепер завжди видимий -->
    <nav class="navbar navbar-expand-lg navbar-light bg-white shadow-sm border-
bottom">
        <div class="container">
            <a class="navbar-brand fw-bold text-primary" href="{% url 'index'
%}">
                □ Алгоритми
            </a>
            <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarNav">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav me-auto">
                    <li class="nav-item">
                        <a class="nav-link text-dark {% if
request.resolver_match.url_name == 'index' %}fw-bold text-primary{% endif %}"
                           href="{% url 'index' %}">Головна</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link text-dark {% if
request.resolver_match.url_name == 'home' %}fw-bold text-primary{% endif %}"
                           href="{% url 'home' %}">Уроки</a>
                    </li>
                </ul>
                <ul class="navbar-nav">
                    {% if user.is_authenticated %}
                    <li class="nav-item">
```

```
                        <a class="nav-link text-dark" href="{% url 'profile'
%}">Кабінет</a>
                    </li>
                    <li class="nav-item">
                        <form method="post" action="{% url 'logout' %}"
style="display: inline;">
                            {% csrf_token %}
                            <button type="submit" class="nav-link text-dark"
                                style="text-decoration: none;">Вийти</button>
                        </form>
                    </li>
                    {% else %}
                    <li class="nav-item">
                        <a class="nav-link text-dark" href="{% url 'login'
%}">Увійти</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link btn btn-primary btn-sm text-white
rounded-pill ms-2"
                            href="{% url 'register' %}">Реєстрація</a>
                    </li>
                    {% endif %}
                </ul>
            </div>
        </div>
    </nav>

    <!-- Основний контент -->
    <main class="flex-grow-1">
        <div class="container py-4">
            {% block content %}{% endblock %}
        </div>
    </main>

    <!-- Footer -->
    <footer class="bg-light py-4 border-top">
        <div class="container text-center text-muted">
            <p class="mb-0">&copy; 2025 Алгоритми та програмування</p>
        </div>
    </footer>

    <!-- Bootstrap JS -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.j
s"></script>
    {% block extra_js %}{% endblock %}
</body>

</html>
register.html
{% extends 'base.html' %}
{% block content %}
<div class="row justify-content-center">
    <div class="col-md-6">
        <div class="card">
            <div class="card-header">
                <h3>Реєстрація</h3>
            </div>
            <div class="card-body">
                <form method="post">
                    {% csrf_token %}
                    <div class="mb-3">
                        <label>Ім'я користувача:</label>
```

```
                        <input type="text" name="username" class="form-control"
required>
                    </div>
                    <div class="mb-3">
                        <label>Пароль:</label>
                        <input type="password" name="password1" class="form-
control" required>
                    </div>
                    <div class="mb-3">
                        <label>Підтвердіть пароль:</label>
                        <input type="password" name="password2" class="form-
control" required>
                    </div>
                    <button type="submit" class="btn btn-success w-
100">Зареєструватись</button>
                </form>
                <div class="mt-3 text-center">
                    <p>Вже є акаунт? <a href="{% url 'login' %}">Увійти</a></p>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
login.html
{% extends 'base.html' %}
{% block content %}
<div class="row justify-content-center">
    <div class="col-md-6">
        <div class="card">
            <div class="card-header">
                <h3>Увійти</h3>
            </div>
            <div class="card-body">
                <form method="post">
                    {% csrf_token %}
                    <div class="mb-3">
                        <label>Ім'я користувача:</label>
                        <input type="text" name="username" class="form-control"
required>
                    </div>
                    <div class="mb-3">
                        <label>Пароль:</label>
                        <input type="password" name="password" class="form-
control" required>
                    </div>
                    <button type="submit" class="btn btn-primary w-
100">Увійти</button>
                </form>
                <div class="mt-3 text-center">
                    <p>Немає акаунта? <a href="{% url 'register'
%}">Зареєструватись</a></p>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
views.py
from django.shortcuts import render, get_object_or_404, redirect
from django.contrib.auth.decorators import login_required
from django.contrib.auth.forms import UserCreationForm
from .models import Lesson, Task, TestQuestion, UserTaskResult, UserTestResult
```

```
def index(request):
    return render(request, 'index.html')

def home(request):
    lessons = Lesson.objects.all()
    return render(request, 'home.html', {'lessons': lessons})

@login_required
def profile(request):
    user = request.user
    task_results =
UserTaskResult.objects.filter(user=user).select_related('task__lesson')
    test_results =
UserTestResult.objects.filter(user=user).select_related('lesson')

    total_tasks = Task.objects.count()
    completed_tasks = task_results.filter(passed=True).count()

    total_lessons = Lesson.objects.count()
    completed_lessons = test_results.filter(score__gt=0).count()

    return render(request, 'profile.html', {
        'task_results': task_results,
        'test_results': test_results,
        'total_tasks': total_tasks,
        'completed_tasks': completed_tasks,
        'total_lessons': total_lessons,
        'completed_lessons': completed_lessons,
    })

def lesson_detail(request, slug):
    lesson = get_object_or_404(Lesson, slug=slug)
    return render(request, 'lesson_detail.html', {'lesson': lesson})

@login_required
def task_detail(request, slug, task_id):
    lesson = get_object_or_404(Lesson, slug=slug)
    task = get_object_or_404(Task, id=task_id, lesson=lesson)
    result = None

    # Отримуємо наступне завдання
    all_tasks = list(lesson.tasks.order_by('order'))
    current_index = next((i for i, t in enumerate(all_tasks) if t.id ==
task.id), -1)
    next_task = None
    if current_index != -1 and current_index + 1 < len(all_tasks):
        next_task = all_tasks[current_index + 1]

    if request.method == 'POST':
        code = request.POST.get('code', '')
        passed = False
        output = ''

        import subprocess, tempfile, os
        try:
            with tempfile.NamedTemporaryFile(mode='w', suffix='.py',
delete=False, encoding='utf-8') as f:
                f.write(code)
                f.flush()
                proc = subprocess.run(
                    ['python', f.name],
                    input=task.test_input,
                    text=True,
                    capture_output=True,
```

```
                    timeout=5,
                    encoding='utf-8'
                )
                output = proc.stdout
                if proc.stderr:
                    output += '\nПомилка:\n' + proc.stderr

                # Нормалізація виводу
                user_lines = [line.rstrip('\r\n') for line in
output.rstrip('\r\n').splitlines()]
                expected_lines = [line.rstrip('\r\n') for line in
task.expected_output.rstrip('\r\n').splitlines()]
                passed = user_lines == expected_lines
        except Exception as e:
            output = f'Системна помилка: {e}'
            passed = False
        finally:
            if 'f' in locals():
                os.unlink(f.name)

        UserTaskResult.objects.update_or_create(
            user=request.user,
            task=task,
            defaults={'passed': passed, 'submitted_code': code}
        )
        result = {'output': output, 'passed': passed}

    last_code = ''
    last_result = UserTaskResult.objects.filter(user=request.user,
task=task).first()
    if last_result:
        last_code = last_result.submitted_code

    return render(request, 'task_detail.html', {
        'lesson': lesson,
        'task': task,
        'result': result,
        'last_code': last_code,
        'next_task': next_task,
    })

@login_required
def take_test(request, slug):
    lesson = get_object_or_404(Lesson, slug=slug)
    questions = lesson.test_questions.prefetch_related('options').all()

    if request.method == 'POST':
        total = questions.count()
        correct = 0

        for q in questions:
            selected = request.POST.getlist(f'question_{q.id}')
            correct_opts = set(str(opt.id) for opt in
q.options.filter(is_correct=True))
            if set(selected) == correct_opts:
                correct += 1

        UserTestResult.objects.update_or_create(
            user=request.user,
            lesson=lesson,
            defaults={'score': correct}
        )

        if total == 0:
```

```
            level = 'none'
        elif correct == total:
            level = 'excellent'
        elif correct >= total / 2:
            level = 'good'
        else:
            level = 'fail'

        return render(request, 'test_result.html', {
            'lesson': lesson,
            'score': correct,
            'total': total,
            'level': level,
        })

    return render(request, 'take_test.html', {
        'lesson': lesson,
        'questions': questions
    })

def register(request):
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('login')
    else:
        form = UserCreationForm()
    return render(request, 'registration/register.html', {'form': form})
urls.py
from django.urls import path
from . import views

app_name = 'lessons'

urlpatterns = [
    path('<slug:slug>/', views.lesson_detail, name='lesson_detail'),
    path('<slug:slug>/task/<int:task_id>/', views.task_detail,
name='task_detail'),
    path('<slug:slug>/test/', views.take_test, name='take_test'),
]
models.py
# lessons/models.py
from django.db import models
from django.contrib.auth.models import User

class Lesson(models.Model):
    title = models.CharField('Назва уроку', max_length=200)
    slug = models.SlugField(max_length=200, unique=True)
    content = models.TextField('Теоретичний матеріал')
    order = models.PositiveIntegerField(default=0)

    class Meta:
        ordering = ['order']
        verbose_name = 'Урок'
        verbose_name_plural = 'Уроки'

    def __str__(self):
        return self.title

class Task(models.Model):
    lesson = models.ForeignKey(Lesson, on_delete=models.CASCADE,
related_name='tasks')
    title = models.CharField('Назва завдання', max_length=200)
```

```
    description = models.TextField('Умова завдання')
    test_input = models.TextField('Тестовий ввід (прихований)', blank=True)
    expected_output = models.TextField('Очікуваний вивід', blank=True)
    order = models.PositiveIntegerField(default=0)

    class Meta:
        ordering = ['order']
        verbose_name = 'Завдання'
        verbose_name_plural = 'Завдання'

    def __str__(self):
        return f"{self.lesson.title} → {self.title}"

class TestQuestion(models.Model):
    lesson = models.ForeignKey(Lesson, on_delete=models.CASCADE,
related_name='test_questions')
    question_text = models.TextField('Текст питання')
    order = models.PositiveIntegerField(default=0)

    class Meta:
        ordering = ['order']
        verbose_name = 'Тестове питання'
        verbose_name_plural = 'Тестові питання'

    def __str__(self):
        return self.question_text[:50]

class TestOption(models.Model):
    question = models.ForeignKey(TestQuestion, on_delete=models.CASCADE,
related_name='options')
    text = models.CharField('Варіант відповіді', max_length=200)
    is_correct = models.BooleanField('Правильна відповідь', default=False)

    def __str__(self):
        return self.text

class UserTaskResult(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    task = models.ForeignKey(Task, on_delete=models.CASCADE)
    passed = models.BooleanField(default=False)
    submitted_code = models.TextField()
    submitted_at = models.DateTimeField(auto_now=True)

    class Meta:
        unique_together = ('user', 'task')
        verbose_name = 'Результат завдання'
        verbose_name_plural = 'Результати завдань'

class UserTestResult(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    lesson = models.ForeignKey(Lesson, on_delete=models.CASCADE)
    score = models.PositiveSmallIntegerField('Бал за тест', default=0)
    completed_at = models.DateTimeField('Дата завершення', null=True,
blank=True)

    class Meta:
        unique_together = ('user', 'lesson')
        verbose_name = 'Результат тесту'
        verbose_name_plural = 'Результати тестів'
apps.py
from django.apps import AppConfig

class LessonsConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
```

```
    name = 'lessons'
admin.py
# lessons/admin.py
from django.contrib import admin
from .models import Lesson, Task, TestQuestion, TestOption, UserTaskResult,
UserTestResult

@admin.register(UserTaskResult)
class UserTaskResultAdmin(admin.ModelAdmin):
    list_display = ('user', 'task', 'passed', 'submitted_at')
    list_filter = ('passed', 'submitted_at')

@admin.register(UserTestResult)
class UserTestResultAdmin(admin.ModelAdmin):
    list_display = ('user', 'lesson', 'score', 'completed_at')
    list_filter = ('lesson', 'completed_at')

# Реєстрація інших моделей (за бажанням)
@admin.register(Lesson)
class LessonAdmin(admin.ModelAdmin):
    list_display = ('title', 'order')
    prepopulated_fields = {'slug': ('title',)}

@admin.register(Task)
class TaskAdmin(admin.ModelAdmin):
    list_display = ('title', 'lesson')
    list_filter = ('lesson',)

@admin.register(TestQuestion)
class TestQuestionAdmin(admin.ModelAdmin):
    list_display = ('question_text', 'lesson')
    list_filter = ('lesson',)

@admin.register(TestOption)
class TestOptionAdmin(admin.ModelAdmin):
    list_display = ('question', 'text', 'is_correct')
    list_filter = ('question__lesson',)
wsgi.py
import os
from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'algorithms_edu.settings')
application = get_wsgi_application()
urls.py
from django.contrib import admin
from django.urls import path, include
from django.contrib.auth import views as auth_views
from lessons import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.index, name='index'),
    path('courses/', views.home, name='home'),
    path('lessons/', include('lessons.urls')),
    path('profile/', views.profile, name='profile'),
    path('accounts/login/', auth_views.LoginView.as_view(), name='login'),
    path('accounts/logout/', auth_views.LogoutView.as_view(next_page='/'),
name='logout'),
    path('accounts/register/', views.register, name='register'),
]
settings.py
import os
from pathlib import Path
```

```
BASE_DIR = Path(__file__).resolve().parent.parent

SECRET_KEY = 'django-insecure-algorithms-edu-secret-key'
DEBUG = True
ALLOWED_HOSTS = []

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'lessons',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'algorithms_edu.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'algorithms_edu.wsgi.application'

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'algorithms_edu',
        'USER': 'root',
        'PASSWORD': 'root1#',
        'HOST': 'localhost',
        'PORT': '3306',
        'OPTIONS': {'charset': 'utf8mb4'},
    }
}

LANGUAGE_CODE = 'uk'
TIME_ZONE = 'Europe/Kiev'
USE_I18N = True
USE_TZ = True

STATIC_URL = '/static/'
```

```
STATICFILES_DIRS = [BASE_DIR / 'static']

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

LOGIN_REDIRECT_URL = '/profile/'
LOGOUT_REDIRECT_URL = '/'
asgi.py
"""
ASGI config for algorithms_edu project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/5.2/howto/deployment/asgi/
"""

import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'algorithms_edu.settings')

application = get_asgi_application()
```