ІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук і технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавра
(ступінь вищої освіти (освітній ступінь))

на тему «РОЗРОБКА НАВЧАЛЬНОЇ ВЕБПЛАТФОРМИ ДЛЯ ВИВЧЕННЯ КУРСУ»

Виконав: студент 4 курсу, групи КНТ-513сп
спеціальності 123 Комп'ютерна інженерія

Освітня програма (спеціалізація)
Комп'ютерна інженерія
(код і назва спеціальності)

ГУМЕНЮК Д.О.
(прізвище та ініціали)
Керівник ТІМЕНКО А.В..
(прізвище та ініціали)
Рецензент ТЯГУНОВ Д.В.
(прізвище та ініціали)

2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

(повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій

Кафедра «Комп'ютерні системи та мережі»

Ступінь вищої освіти (освітній ступінь) бакалаврський

Спеціальність 123 Комп'ютерна інженерія

(код і назва)

Освітня програма (спеціалізація) Комп'ютерна інженерія

(назва)

ЗАТВЕРДЖУЮ

Зав. кафедри

_____ Кудерметов Р.К.

"14" квітня 2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

ГУМЕНЮК Дмитро Олександрович

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) «Розробка навчальної вебплатформи для вивчення курсу»

керівник проекту (роботи) ТІМЕНКО А.В.., ст. викл.

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "14" квітня 2026 року № 160

2. Строк подання студентом проекту (роботи) 01.06. 2025 року

3. Вихідні дані до проекту (роботи) LMS , мова програмування Python, JavaScript, React, PostgreSQL, Docker

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Опис предметної області

2) Розробка вимог до програмного забезпечення

3) Реалізація комп'ютерної системи

4) Тестування та аналіз результатів роботи системи

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

Слайди презентації

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	ТІМЕНКО А.В., ст. викладач		
Нормоконтроль	ГОЛУБ Т.В., доцент		

7. Дата видачі завдання 05.05. 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Опис предметної області	07.04.2026	
2	Формування вимог до системи	10.04.2026	
3	Розробка вимог до програмного забезпечення	17.04.2026	
4	Обгрунтування вибору технологій розробки	23.04.2026	
5	Вибір системи керування базами даних	28.04.2026	
6	Проєктування архітектури програмного продукту	15.05.2026	
7	Проєктування бази даних	26.05.2026	
9	Тестування та аналіз результатів роботи	25.05.2026	

Студент _____ **Дмитро ГУМЕНЮК**
(підпис) (ініціали та прізвище)

Керівник проекту (роботи) _____ **Артур ТІМЕНКО**
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 103 с., 16 табл., 6 рис, 4 листингів, 2 додатки, 25 джерел.

DJANGO, FASTAPI, PYTHON, ІНТЕРФЕЙС КОРИСТУВАЧА, СИСТЕМИ ЕЛЕКТРОННОГО НАВЧАННЯ, СКБД

У сучасному світі інформаційні технології є однією з найбільш динамічних галузей, що стрімко розвиваються та інтегруються у всі сфери життя. Мова програмування Python стабільно посідає провідні позиції у світових рейтингах завдяки своїй простоті, універсальності та потужній екосистемі. Вона є індустріальним стандартом у таких напрямках, як веброзробка, аналіз даних, машинне навчання та штучний інтелект. Це створює стабільно високий попит на кваліфікованих фахівців, що робить вивчення Python критично важливим етапом підготовки студентів інженерних та ІТ-спеціальностей.

Попри існування великої кількості глобальних освітніх платформ, використання їх у локальному навчальному процесі часто супроводжується низкою проблем. Переважна більшість якісних інтерактивних курсів є англомовними, що може ускладнювати засвоєння базових концепцій програмування для початківців. Відсутність адаптації до навчальних планів. Обмежений контроль успішності.

Враховуючи сучасні тенденції до цифровізації освіти та стійкий запит на якісне дистанційне й змішане навчання, розробка власної навчальної вебплатформи є об'єктивною необхідністю. Створення спеціалізованої системи для курсу «Основи програмування на Python» дозволить вирішити такі ключові завдання:

– створення єдиного освітнього середовища - об'єднання теоретичного матеріалу, відеолекцій, інтерактивних тестів та середовища для написання і перевірки коду в межах одного вебдодатку;

– підвищення інтерактивності - забезпечення можливості для студентів виконувати практичні завдання безпосередньо в браузері (без необхідності локального налаштування середовища розробки), що суттєво знижує поріг входу;

– локалізація та доступність - надання якісного україномовного контенту, який відповідає державним стандартам освіти та специфіці підготовки вітчизняних фахівців;

– автоматизація рутини викладача - впровадження модулів автоматичної перевірки синтаксису та логіки коду студентів, а також генерації звітів про успішність групи.

Необхідність розробки навчальної вебплатформи зумовлена потребою у створенні сучасного, доступного та інтерактивного інструменту, який оптимізує процес вивчення програмування на Python, підвищить мотивацію студентів шляхом гейміфікації та практично-орієнтованого підходу, а також забезпечить викладачів зручним механізмом адміністрування навчального процесу. Реалізація даного проєкту має високу практичну цінність і може бути впроваджена безпосередньо в навчальний процес.

ABSTRACT

Explanatory note to the master's work: 103 p., 16 tables, 6 figures, 4 listing, 25 sources.

DJANGO, FASTAPI, PYTHON, USER INTERFACE OF THE E-LEARNING SYSTEM, SCBD

In today's world, information technology is one of the most dynamic sectors, developing rapidly and becoming integrated into all areas of life. The Python programming language consistently ranks among the top in global rankings thanks to its simplicity, versatility and robust ecosystem. It is the industry standard in fields such as web development, data analysis, machine learning and artificial intelligence. This creates a consistently high demand for qualified specialists, making the study of Python a critically important stage in the training of students in engineering and IT disciplines.

Despite the existence of a large number of global educational platforms, their use in the local teaching process is often accompanied by a number of problems. The vast majority of high-quality interactive courses are in English, which can make it difficult for beginners to grasp basic programming concepts. Lack of adaptation to curricula. Limited monitoring of progress.

Given current trends towards the digitalisation of education and the sustained demand for high-quality distance and blended learning, the development of a proprietary educational web platform is an objective necessity. The creation of a specialised system for the 'Fundamentals of Python Programming' course will enable the following key objectives to be met:

– creation of a unified learning environment – combining theoretical material, video lectures, interactive tests and an environment for writing and testing code within a single web application;

– enhancing interactivity – enabling students to complete practical tasks directly in the browser (without the need to set up a local development environment), which significantly lowers the barrier to entry;

– localisation and accessibility – providing high-quality Ukrainian-language content that meets national education standards and the specific requirements for training domestic specialists;

– automation of routine teaching tasks – implementing modules for automatic checking of students' code syntax and logic, as well as generating reports on the group's progress.

The need to develop an educational web platform stems from the requirement to create a modern, accessible and interactive tool that optimises the process of learning Python programming, increases student motivation through gamification and a practice-oriented approach, and provides teachers with a convenient mechanism for managing the learning process. The implementation of this project has high practical value and can be directly integrated into the teaching process.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

DBMS	Database Management Systems, Системи керування базами даних
DoS	Denial of Service, Відмова в обслуговуванні
LMS	Learning Management System, Системи електронного навчання
MVC	Model-View-Controller, Модель-Представлення-Контролер
MVCC	Multi-Version Concurrency Control, Багатовершинне керування конкурентністю
RCE	Remote Code Execution, Віддалене виконання коду
REST	Representational State Transfer, Передача стану представлення
SPA	Single Page Application.Односторінковий вебзастосунок
UA	Unauthorized Access, Несанкціонований доступ
UI	User Interface, Інтерфейс користувача

ВСТУП

Сучасний розвиток інформаційних технологій суттєво впливає на освітній процес, відкриваючи нові можливості для інтерактивного навчання та дистанційної взаємодії між студентами й викладачами. Одним із ключових інструментів у цьому напрямі є системи електронного навчання (LMS), які забезпечують організацію навчального процесу, управління курсами та контроль результатів.

Актуальність теми зумовлена потребою у створенні платформи, що поєднує зручний інтерфейс, адаптивність до різних пристроїв, інтеграцію редактора коду та можливість автоматизованої перевірки практичних завдань. Така система дозволяє студентам отримувати практичні навички програмування у реальному середовищі, а викладачам - ефективно організовувати навчальний процес і здійснювати контроль успішності.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Предметною областю даного дипломного проєкту є сфера електронного навчання (E-learning) та освітніх технологій (EdTech), зокрема, процес дистанційного та змішаного вивчення мов програмування. У вужчому розумінні досліджується специфіка викладання та засвоєння курсу «Основи програмування на Python» у закладах вищої освіти або в межах спеціалізованих ІТ-курсів [1].

На відміну від традиційних гуманітарних чи теоретичних дисциплін, вивчення програмування вимагає не лише засвоєння лекційного матеріалу, але й інтенсивної практичної роботи. Тому предметна область охоплює інтеграцію систем управління навчанням із середовищами розробки та автоматизованого тестування програмного коду.

У процесі функціонування майбутньої системи відбувається взаємодія з такими базовими інформаційними сутностями:

– навчальний контент - теоретичні матеріали, лекції, презентації, відеоролики, які структуровані за темами та модулями;

– практичні завдання - набори вправ, які включають умову задачі, вхідні дані, очікувані вихідні дані (еталонні результати) та обмеження по часу/пам'яті;

– тестові завдання - питання з варіантами відповідей (множинний вибір, відкрите питання) для швидкої перевірки теоретичних знань;

– програмний код - розв'язки завдань, написані студентами мовою Python;

– статистика та успішність - оцінки, історія спроб, витрачений час, рейтинг студентів.

Предметна область передбачає наявність кількох категорій користувачів із різним рівнем доступу та повноважень.

Студент (Здобувач освіти) - кінцевий споживач освітніх послуг. Його головні функції:

– реєстрація на курс;

– опрацювання теорії;

– проходження тестів;

– написання та відправка коду на перевірку;

– відстеження власного прогресу.

Викладач (Ментор/Тьютор) - організатор навчального процесу. Відповідає за наступне:

– створення, редагування та структурування курсу;

– додає нові завдання;

– налаштовує критерії оцінювання;

– аналізує загальну успішність групи;

– залишає коментарі до коду студентів (code review).

Адміністратор:

– забезпечує технічну підтримку платформи;

– керує правами доступу;

– реєструє викладачів;

– відповідає за резервне копіювання даних та стабільність роботи системи.

Ключовими процесами, що протікають у межах даної предметної області, є:

– управління контентом - процес створення навчальних модулів викладачем та їх публікація для доступу студентам;

– процес навчання - послідовний перехід студента від теорії до практики;

– процес практичної перевірки - найважливіший етап, що включає написання коду студентом у вебінтерфейсі платформи;

– відправка коду на сервер;

– безпечний запуск коду в ізольованому середовищі (sandbox), щоб уникнути пошкодження сервера шкідливим кодом;

– прогін написаного коду через серію автоматичних тестів (unit-тестів)

із різними наборами вхідних даних;

– повернення результату студенту (успішно / синтаксична помилка / логічна помилка / перевищення ліміту часу);

– оцінювання та аналітика - автоматичне формування оцінки на основі пройдених тестів або результатів ручної перевірки викладачем, формування загального рейтингу та звітів успішності.

Специфіка предметної області полягає в необхідності створення не просто інформаційного порталу, а складного апаратно-програмного комплексу. Цей комплекс повинен поєднувати функції традиційної LMS із надійними механізмами компіляції/інтерпретації коду користувачів у реальному часі. Автоматизація перевірки практичних завдань є критичною вимогою, оскільки вона дозволяє зняти рутинне навантаження з викладача та забезпечує миттєвий зворотний зв'язок для студента, що значно підвищує ефективність вивчення Python.

1.1 Види аналогів для створення вебплатформи курсу

Для формування чітких вимог до розроблюваної навчальної вебплатформи необхідно провести аналіз сучасного ринку систем LMS та спеціалізованих платформ для вивчення програмування. На сьогодні існує велика кількість рішень, проте кожне з них має свої специфічні особливості, які не завжди задовольняють потреби конкретного навчального закладу чи курсу.

Для порівняльного аналізу обрано три популярні системи, які представляють різні підходи до організації e-learning: Moodle, Coursera та Codecademy.

1.1.1 Система управління навчанням Moodle

Moodle - це найпопулярніша у світі безкоштовна LMS з відкритим вихідним кодом, яка широко використовується в українських закладах вищої освіти [2].

Переваги:

– безкоштовність та можливість локального розгортання на серверах університету;

– потужний інструментарій для створення тестів, журналів оцінок та управління контентом;

– велика спільнота та наявність локалізації (у тому числі української).

Недоліки (в контексті курсу Python):

– перевантажений та застарілий інтерфейс користувача, який складно кастомізувати.

– відсутність вбудованого «з коробки» середовища для виконання та перевірки програмного коду. Для реалізації такого функціоналу необхідно встановлювати та складно налаштовувати сторонні плагіни (наприклад, Virtual Programming Lab), що вимагає значних ресурсів адміністрування.

1.1.2 Масова відкрита онлайн-платформа Coursera

Coursera - один із флагманів серед платформ масових відкритих онлайн-курсів (MOOC), що співпрацює з провідними університетами світу [3].

Переваги:

– сучасний, інтуїтивно зрозумілий та стабільний користувацький інтерфейс;

– висока масштабованість та якісна інтеграція відеоконтенту;

– наявність системи грейдингу для перевірки складних завдань, зокрема через інтеграцію з Jupyter Notebooks для Python.

Недоліки:

– закрита пропрієтарна архітектура: заклад освіти не може безкоштовно розгорнути цю систему на своїх серверах або змінити її вихідний код;

– комерційна спрямованість платформи (доступ до практичних завдань та оцінювання здебільшого платний);

– орієнтація на глобальний ринок, а не на специфічні потреби локальних навчальних програм.

1.1.3 Інтерактивна платформа Codecademy

Codecademy - спеціалізована онлайн-платформа для інтерактивного вивчення мов програмування, де основний акцент робиться на практику написання коду безпосередньо в браузері [4].

Переваги:

– чудове вбудоване середовище розробки у браузері, що дозволяє миттєво запускати Python-код;

– покрокова система навчання з автоматичною перевіркою синтаксису та логіки коду в реальному часі;

– високий рівень гейміфікації та залученості студента.

Недоліки:

– відсутність можливості використовувати платформу як власну LMS для створення авторських курсів сторонніми викладачами (це закритий комерційний продукт);

– обмежена україномовна локалізація;

– відсутність інструментів для академічного адміністрування (журнали академічних груп, детальна статистика для локального викладача).

Для наочності результати аналізу зведено у таблицю 1.1 за ключовими критеріями, необхідними для вивчення курсу «Основи програмування на Python».

Проведений огляд показує, що існуючі рішення або є занадто універсальними та позбавленими специфічних інструментів для програмістів (Moodle), або є закритими комерційними продуктами без можливості локального розгортання та адаптації (Coursera, Codecademy).

Отже, розробка власної вебплатформи є обґрунтованою. Вона дозволить поєднати переваги академічної LMS (можливість створювати власні лекції, контролювати успішність студентів) із перевагами інтерактивних платформ (наявність вбудованого онлайн-редактора коду з безпечним середовищем для автоматичного виконання та тестування скриптів мовою Python), забезпечивши при цьому повну адаптацію під навчальний процес та україномовний інтерфейс.

Таблиця 1.1 - Порівняльна характеристика платформ

Критерій порівняння	Moodle	Coursera	Codecademy	Розроблювана система
Вбудоване середовище виконання коду (IDE)	Ні (тільки плагіни)	Частково (зовнішні інструменти)	Так	Так
Автоматична перевірка коду (Автогрейдер)	Ні	Так	Так	Так
Можливість локального розгортання (Self-hosted)	Так	Ні	Ні	Так
Налаштування під власні навчальні плани	Так	Ні	Ні	Так
Безкоштовність для закладу освіти	Так	Ні	Ні	Так

1.2 Постановка задачі та формування вимог

Метою даного дипломного проєкту є проєктування та розробка спеціалізованої навчальної вебплатформи для вивчення дисципліни «Основи програмування на Python», яка забезпечить інтерактивний процес навчання з можливістю написання та автоматизованої перевірки програмного коду в режимі реального часу.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– спроєктувати архітектуру клієнт-серверного вебдодатка;

– розробити логічну та фізичну структуру бази даних для зберігання навчального контенту, даних користувачів та статистики;

– реалізувати модуль онлайн-редактора коду з підсвічуванням синтаксису;

– створити ізольоване середовище (sandbox) для безпечного виконання

та автоматичного тестування користувацьких скриптів на стороні сервера;

– розробити інтуїтивно зрозумілий користувацький інтерфейс для різних ролей (студент, викладач, адміністратор).

1.2.1 Функціональні вимоги

Програмний продукт повинен забезпечувати виконання наступних функцій, які розділені відповідно до ролей користувачів у системі:

Вимоги до підсистеми «Студент»:

– реєстрація, авторизація та безпечне управління власним профілем;

– доступ до структурованих навчальних матеріалів (лекції, відеоматеріали);

– можливість написання, редагування та запуску коду мовою Python безпосередньо у вікні веббраузера без встановлення додаткового ПЗ;

– відправка розв'язків практичних завдань на автоматичну перевірку та отримання деталізованого зворотного зв'язку (результат тестів, інформація про синтаксичні або логічні помилки);

– відстеження власного прогресу та перегляд історії виконання завдань.

Вимоги до підсистеми «Викладач»:

– створення, редагування та структурування навчальних модулів і лекцій;

– формування банку практичних завдань із зазначенням умов, наборів вхідних і вихідних даних для автоматичних тестів (test cases);

– встановлення обмежень для завдань (ліміт часу виконання коду, ліміт використання оперативної пам'яті);

– перегляд аналітики та загальної статистики успішності академічної групи;

– доступ до вихідного коду, надісланого студентами, для проведення ручного аналізу (code review) та залишення коментарів.

Вимоги до підсистеми «Адміністратор»:

– керування обліковими записами користувачів (блокування, призначення ролей);

– моніторинг стану системи та навантаження на сервер обчислень під час масового запуску скриптів користувачами.

1.2.2 Нефункціональні вимоги

З огляду на специфіку архітектури платформи, яка передбачає виконання стороннього коду, до системи висуваються такі жорсткі нефункціональні вимоги.

Безпека (Security) - виконання коду користувачів повинно відбуватися у строго ізольованих контейнерах або віртуальних середовищах. Це необхідно для унеможливлення несанкціонованого доступу до файлової системи сервера, виконання шкідливих системних команд або нескінченного споживання ресурсів (захист від DoS-атак на рівні коду). Паролі користувачів мають зберігатися виключно у вигляді криптографічних хешів.

Продуктивність (Performance) - час очікування результатів автоматичної перевірки коду студента не повинен перевищувати 3–5 секунд за умови стандартного навантаження на сервер.

Кросплатформність та ергономічність - клієнтська частина додатку (Frontend) повинна мати адаптивний дизайн для коректного відображення на десктопних і мобільних пристроях, а також підтримуватися всіма сучасними веббраузерами.

Масштабованість (Scalability) - архітектура повинна дозволяти горизонтальне масштабування підсистеми перевірки коду (додавання нових worker-вузлів) у разі збільшення кількості одночасних запитів від користувачів.

Гнучкість інтеграції з зовнішніми сервісами та підтримка безперервного оновлення, що забезпечує довготривалу життєздатність платформи.

Врахування зазначених нефункціональних вимог є критично важливим для забезпечення стабільності та довготривалої ефективності платформи. Вони формують основу її надійності, дозволяють адаптуватися до змінних умов використання та гарантують відповідність сучасним стандартам розробки програмного забезпечення.

2 РОЗРОБКА ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Обґрунтування вибору технологій розробки

Для успішної реалізації поставлених завдань та забезпечення стабільної роботи платформи необхідно обрати оптимальний набір інструментів розробки (технологічний стек). Вибір базується на вимогах до продуктивності, безпеки, можливості масштабування та швидкості розробки програмного продукту.

2.1.1 Вибір мови програмування та серверного фреймворку

Оскільки цільовим призначенням платформи є вивчення мови програмування Python, логічним та архітектурно правильним рішенням є використання цієї ж мови для розробки серверної частини (Backend). Це спрощує інтеграцію модулів перевірки коду та дозволяє використовувати єдину екосистему [5].

Серед найпопулярніших вебфреймворків для Python виділяють Django та FastAPI.

FastAPI є сучасним, асинхронним та надзвичайно швидким інструментом, проте він вимагає самостійної реалізації багатьох базових компонентів (авторизації, роботи з базою даних) [6].

Django - це високорівневий фреймворк, що базується на принципі «все включено». Він містить потужну вбудовану систему об'єктно-реляційного відображення (ORM), готову панель адміністратора та надійні механізми захисту від типових вразливостей (SQL-ін'єкцій, CSRF-атак) [7].

Для розробки комплексної LMS-системи зі складною бізнес-логікою, ієрархією користувачів та необхідністю швидкого створення адмінпанелі обрано фреймворк Django (разом із Django REST Framework для побудови API).

2.2 Аналіз та вибір мови програмування серверної частини

Вибір базової мови програмування для розробки серверної частини є одним із найвідповідальніших етапів проєктування системи. Від цього рішення залежить швидкість розробки, продуктивність, масштабованість, а також безпека майбутньої навчальної платформи. Для обґрунтування оптимального технологічного рішення проведемо порівняльний аналіз чотирьох популярних мов програмування: C++, Java, JavaScript (Node.js) та Python.

2.2.1 C++

Мова програмування загального призначення, яка відзначається найвищою продуктивністю та надає розробнику прямий контроль над апаратними ресурсами комп'ютера та пам'яттю [8].

Переваги: незамінна при розробці систем реального часу, мікропроцесорної техніки (зокрема систем Інтернету речей - IoT), мережевої безпеки та високонавантажених обчислювальних ядер.

Недоліки (в контексті проєкту): C++ має дуже високий поріг входження та низьку швидкість розробки (Time-to-Market) для класичних вебдодатків. Відсутність сучасних високорівневих вебфреймворків робить реалізацію стандартного функціоналу (маршрутизація, ORM, авторизація) надзвичайно трудомісткою. Використання C++ для бекенду вебплатформи є технологічно та економічно недоцільним.

2.2.2 Java

Строго типізована, об'єктно-орієнтована мова програмування, яка є світовим стандартом у секторі корпоративної розробки [9].

Переваги: висока надійність, безпека, кросплатформність (завдяки віртуальній машині JVM) та наявність потужного вебфреймворку Spring. Чудово підходить для складних банківських чи державних систем.

Недоліки: Java характеризується високою «багатослівністю» коду, що збільшує обсяг кодової бази. Крім того, додатки на Java (особливо на базі Spring)

споживають значну кількість оперативної пам'яті, що може бути критичним недоліком при розгортанні локальної навчальної платформи на обмежених серверних ресурсах закладу освіти.

2.2.3 JavaScript

Одна з найпопулярніших мов для веб-розробки, яка завдяки середовищу Node.js дозволяє виконувати код на стороні сервера [10].

Переваги: можливість використовувати одну мову як для клієнтської (Frontend), так і для серверної (Backend) частини. Асинхронна неблокуюча модель вводу/виводу дозволяє ефективно обробляти велику кількість одночасних з'єднань користувачів.

Недоліки: Node.js є однопотоковим. Оскільки наша навчальна платформа передбачає запуск та перевірку студентського коду, використання однопотокової архітектури без складної системи мікросервісів може призвести до блокування головного потоку і зависання системи для всіх інших користувачів.

2.2.4 Python

Високорівнева мова програмування загального призначення, яка акцентує увагу на продуктивності розробника та читабельності коду [11].

Переваги: лаконічний синтаксис дозволяє реалізувати складну бізнес-логіку меншою кількістю рядків коду порівняно з Java чи C++. Наявність потужних екосистем та фреймворків (таких як Django та FastAPI) забезпечує швидке створення безпечних вебдодатків. Python має найкращу серед конкурентів підтримку інструментів для роботи з операційною системою, контейнеризацією та скриптингом.

Порівняльна характеристика обрання мови програмування для даної роботи представлена в таблиці 2.1.

Як видно з таблиці 2.1, хоча Python поступається мовам C++ та Java у базовій швидкості виконання скомпільованого коду, він має беззаперечну перевагу за критеріями швидкості розробки вебдодатків та зручності інтеграції нативних інструментів для тестування Python-скриптів. Оскільки «вузьким місцем» навчальної платформи є не швидкість обробки HTTP-запитів, а саме

процес ізольованого запуску користувацького коду, вибір Python є найбільш збалансованим та оптимальним інженерним рішенням.

Таблиця 2.1 - Порівняльна характеристика мов програмування для розробки серверної частини

Критерій порівняння	C++	Java	JavaScript (Node.js)	Python
Швидкість розробки (Time-to-Market)	Низька	Середня	Висока	Дуже висока
Швидкість виконання коду (Продуктивність)	Дуже висока	Висока	Середня	Середня
Споживання оперативної пам'яті (RAM)	Низьке	Високе	Середнє	Середнє
Екосистема для веброзробки (Фреймворки)	Обмежена	Розвинена (Spring)	Дуже розвинена (Express, Nest	Дуже розвинена (Django, FastAPI
Складність багатопотокової обробки (коду)для перевірки	Висока	Середня	Висока (через Event Loop)	Середня (через Celery/Asyncio)
Зручність інтеграції модуля перевірки Python-коду	Складна	Складна	Середня	Нативна (Максимальна

На основі проведеного аналізу в якості мови програмування для розробки серверної частини платформи обрано Python. Цей вибір є не лише технологічно обґрунтованим, але й концептуально необхідним з огляду на такі фактори:

– гомогенність екосистеми - оскільки предметною областю дипломного проєкту є розробка платформи для вивчення курсу «Основи програмування на Python», використання цієї ж мови для написання самої платформи є логічним кроком;

– спрощення розробки модуля тестування - використання Python на бекенді дозволить безшовно інтегрувати модуль автоматичної перевірки студентського коду (автогрейдер), можливість використовувати нативні бібліотеки (наприклад, `ast` для перевірки синтаксичних дерев студентських розв'язків або вбудовані модулі тестування `unittest` та `pytest`) без необхідності створювати складні «мости» між різними мовами програмування;

– швидкість розробки - наявність готових рішень для створення панелі адміністратора, системи авторизації та роботи з базою даних дозволить зосередити основні зусилля безпосередньо на розробці унікального освітнього функціоналу.

2.3 Вибір технологій клієнтської частини

Для забезпечення високої інтерактивності (зокрема, роботи онлайн-редактора коду без перезавантаження сторінки) клієнтську частину доцільно реалізувати у вигляді односторінкового додатку.

Основними конкурентами на ринку SPA-фреймворків є React, Vue.js та Angular:

– Angular має високий поріг входження і є надмірним для даного проєкту;

– React та Vue.js чудово підходять для створення динамічних інтерфейсів.

Вибір зроблено на користь бібліотеки React, оскільки вона є індустріальним стандартом, має величезну екосистему готових компонентів та високу продуктивність завдяки використанню віртуального DOM. Для безпосередньої реалізації редактора коду у браузері буде інтегровано бібліотеку Monaco Editor, ядро, на якому побудовано VS Code, що забезпечить звичний для розробників інтерфейс із підсвічуванням синтаксису.

Клієнтська частина навчальної платформи відіграє критично важливу роль, оскільки саме вона забезпечує взаємодію користувача із системою. Специфіка розроблюваного вебдодатка полягає в тому, що він не є звичайним інформаційним сайтом. Платформа повинна функціонувати як інтерактивне середовище навчання, де студент може переглядати лекції, писати код у вбудованому редакторі, запускати його та миттєво отримувати результати без перезавантаження сторінки.

Для реалізації такого складного користувацького інтерфейсу з динамічним управлінням станом (наприклад, збереження написаного коду під час навігації між вкладками), використання базового підходу (чистий HTML, CSS та Vanilla JavaScript) є неефективним. Тому доцільно проаналізувати сучасні компонентні JavaScript-фреймворки та бібліотеки: Angular, Vue.js та React.

2.3.1 Фреймворк Angular

Повноцінний, важкий фреймворк від компанії Google, який базується на архітектурі MVC та використовує мову TypeScript за замовчуванням [12].

Переваги: містить усі необхідні інструменти «з коробки» (маршрутизація, робота з HTTP, управління формами). Забезпечує сувору типізацію та чітку архітектуру, що ідеально підходить для великих корпоративних рішень.

Недоліки: має дуже високий поріг входження та складну криву навчання. Для відносно невеликого проєкту (яким є бакалаврська дипломна робота) Angular є занадто надлишковим, оскільки вимагає написання великої кількості шаблонного коду навіть для простих компонентів.

2.3.2 Фреймворк Vue.js

Прогресивний JavaScript-фреймворк, який здобув популярність завдяки своїй простоті та гнучкості [13].

Переваги: має найбільш плавно зростаючу криву навчання серед конкурентів, чудову документацію та легку вагу. Дозволяє поступово впроваджувати реактивність у вже існуючі проєкти. Забезпечує високу продуктивність завдяки оптимізованому Virtual DOM.

Недоліки: попри зростаючу популярність, екосистема Vue.js (кількість готових складних бібліотек та компонентів) все ще поступається лідеру ринку. Інтеграція специфічних інструментів, таких як повноцінний редактор коду (IDE у браузері), може потребувати більше ручного налаштування.

2.3.3 Фреймворк React

JavaScript-бібліотека для створення користувацьких інтерфейсів, розроблена та підтримувана компанією Meta (Facebook). React базується на компонентному підході та декларативному стилі програмування [14].

Переваги:

– гнучкість та екосистема - React не нав'язує жорсткої архітектури, дозволяючи розробнику самостійно обирати інструменти для маршрутизації (React Router) та управління станом (Redux, Zustand);

– Virtual DOM - забезпечує високу швидкість перемальовування інтерфейсу, що критично важливо при створенні онлайн-редактора коду, де зміни відбуваються з кожним натисканням клавіші;

– готові рішення - існує величезна кількість адаптованих бібліотек, зокрема існують потужні обгортки для інтеграції Monaco Editor - рушія, на якому побудований популярний редактор Visual Studio Code. Це дозволить без надмірних зусиль впровадити на платформу професійний редактор з підсвічуванням синтаксису Python, автодоповненням та перевіркою помилок на льоту.

Недоліки: вимагає додаткового часу на налаштування інфраструктури проєкту та самостійний вибір супутніх бібліотек (оскільки React є бібліотекою для відображення, а не повноцінним фреймворком).

Для візуалізації результатів аналізу, основні показники розглянутих технологій зведено у таблицю 2.2.

З огляду на необхідність створення високоінтерактивного Single Page Application з вбудованим середовищем для написання коду, базовий JavaScript є недостатнім, а Angular - надлишковим.

Таблиця 2.2 - Порівняльна характеристика технологій клієнтської частини

Критерій порівняння	Базовий стек (HTML/CSS/JS)	Angular	Vue.js	React
Архітектурний підхід	Імперативний	MVC / Компонентний	Компонентний	Компонентний
Підтримка концепції SPA	Складна (вручну)	Вбудована	Вбудована	Нативна (через роутер)
Поріг входження	Низький	Дуже високий	Низький	Середній
Розмір екосистеми (готові рішення)	Величезний, але розрізнений	Середній	Середній	Дуже великий
Зручність інтеграції IDE (Monaco Editor)	Середня	Середня	Добра	Висока (готові npm-пакети

Оптимальним вибором для реалізації клієнтської частини є бібліотека React у поєднанні з надмножиною мови TypeScript (для забезпечення статичної типізації та зменшення кількості помилок під час розробки). Використання React дозволить розбити складний інтерфейс (відеоплеєр, панель завдань, термінал виводу результатів, редактор коду) на незалежні компоненти, що значно спростить їх розробку, тестування та подальшу підтримку. Для реалізації самого редактора коду в браузері буде використано бібліотеку Monaco Editor, яка ідеально інтегрується в екосистему React.

2.4 Вибір системи керування базами даних

Навчальна платформа оперує структурованими даними зі складними зв'язками (користувачі, курси, модулі, завдання, спроби виконання, оцінки). У таких умовах реляційні бази даних є значно ефективнішими за NoSQL рішення.

Для проєкту обрано PostgreSQL - потужну об'єктно-реляційну СКБД з відкритим вихідним кодом. Її перевагами є [15]:

– висока надійність та дотримання стандартів ACID (атомарність, узгодженість, ізольованість, довговічність);

– ефективна робота зі складними запитами та великими обсягами даних;

– повна сумісність із Django ORM.

СКБД є фундаментальним компонентом будь-якої системи електронного навчання. Навчальна платформа оперує великою кількістю структурованої інформації, яка має складні ієрархічні зв'язки. Наприклад, ієрархія даних виглядає так: «Студент» пов'язаний із «Академічною групою», яка має доступ до «Курсу», що складається з «Модулів» та «Завдань», кожне з яких має безліч «Спроб виконання» та «Оцінок». Така структура вимагає забезпечення високого рівня цілісності даних.

Для вибору оптимального рішення проведемо аналіз чотирьох популярних СКБД, які представляють різні архітектурні підходи: MongoDB, SQLite, MySQL та PostgreSQL [16].

2.4.1 MongoDB

Одна з найпопулярніших документо-орієнтованих NoSQL баз даних. Вона зберігає дані у гнучкому форматі, подібному до JSON (BSON).

Переваги: висока швидкість запису та гнучка схема даних (можливість змінювати структуру документів «на льоту» без складних міграцій). Чудово підходить для аналітики великих неструктурованих даних Big Data або систем реального часу.

Недоліки: відсутність класичних зовнішніх ключів (Foreign Keys) та складність виконання багатотабличних запитів (JOIN). Враховуючи, що дані навчальної платформи (користувачі, завдання, оцінки) є строго реляційними, використання MongoDB призведе до дублювання даних та ускладнення логіки на рівні бекенду для підтримання їхньої узгодженості.

2.4.2 SQLite

Легка реляційна база даних, яка зберігає всю інформацію в одному локальному файлі і не вимагає розгортання окремого сервера СКБД.

Переваги: вбудована в стандартну бібліотеку Python, має нульовий поріг налаштування. Є ідеальним вибором для розробки, прототипування або локального тестування.

Недоліки: SQLite блокує весь файл бази даних під час операцій запису. Це означає, що при одночасному надсиланні розв'язків завдань навіть кількома десятками студентів, система почне генерувати помилки та зависати. Ця СКБД абсолютно непридатна для використання у виробничому (production) середовищі вебдодатка.

2.4.3 MySQL

Класична, надзвичайно популярна реляційна СКБД з відкритим вихідним кодом.

Переваги: висока швидкість виконання операцій читання, величезна спільнота, підтримка всіма існуючими вебхостингами.

Недоліки: історично MySQL має менш суворе дотримання стандартів SQL порівняно з конкурентами. Обробка складних типів даних (наприклад, багатовимірних масивів або JSON-об'єктів, які зручно використовувати для зберігання наборів тестових даних) реалізована менш ефективно, ніж у PostgreSQL.

2.4.4 PostgreSQL

Передова об'єктно-реляційна система керування базами даних з відкритим вихідним кодом, яка позиціонується як найдосконаліша у світі.

Переваги:

– повна відповідність стандартам ACID (атомарність, узгодженість, ізольованість, довговічність), що гарантує збереження фінансових та академічних даних (оцінок) навіть у разі апаратних збоїв;

– потужна система керування паралельним доступом за допомогою багатоверсійності MVCC, яка дозволяє ефективно обробляти тисячі одночасних підключень без блокування таблиць;

– відмінна підтримка нестандартних типів даних, зокрема JSONB, що дозволить оптимізувати зберігання конфігурацій завдань та історії їх перевірок (логів автогрейдера).

– максимальна сумісність із фреймворком Django. Багато розширених функцій Django ORM (наприклад, повнотекстовий пошук або специфічні індекси) працюють виключно з PostgreSQL.

Недоліки: вимагає більше оперативної пам'яті для роботи порівняно з MySQL та є дещо складнішою в початковому налаштуванні.

Для візуалізації результатів аналізу основні характеристики розглянутих СКБД наведено у таблиці 2.3.

Таблиця 2.3 - Порівняльна характеристика систем керування базами даних

Критерій порівняння	MongoDB	SQLite	MySQL	PostgreSQL
Тип	NoSQL	Реляційна	Реляційна	Об'єктно-реляційна
Підтримка транзакцій (ACID)	Часткова	Так	Так	Повна, строга
Обробка конкурентних запитів	Висока	Дуже низька	Висока	Дуже висока
Інтеграція з Django ORM	Складна	Базова	Добра	Максимальна
Робота зі специфічними даними (JSON)	Відмінна	Обмежена	Задовільна	Відмінна

Враховуючи вимоги до надійності зберігання академічної інформації, необхідність підтримання складної реляційної структури даних курсу та архітектурний вибір бекенд-фреймворку Django, найбільш доцільним вибором є СКБД PostgreSQL.

Використання NoSQL є невиправданим через реляційну природу даних, а SQLite може застосовуватися виключно на етапі локального написання коду платформою. PostgreSQL забезпечить необхідний рівень масштабованості, надійну обробку одночасних запитів від студентів під час складання тестів, а також надасть розширений інструментарій для роботи з JSON-структурами для модуля автоматичного оцінювання коду.

2.5 Технології безпечного виконання коду

Найбільш критичним архітектурним компонентом системи є модуль перевірки користувацького коду. Запуск неперевіреного коду безпосередньо на сервері є критичною загрозою безпеці.

Для вирішення цієї проблеми обрано платформу контейнеризації Docker [17]. Надісланий студентом Python-код буде виконуватися всередині ізольованого Docker-контейнера з налаштованими обмеженнями:

– обмеження часу виконання, щоб уникнути нескінченних циклів;

– обмеження оперативної пам'яті, щоб запобігти вичерпанню ресурсів сервера;

– відсутність доступу до мережі Інтернет та файлової системи хостової машини.

Найбільш критичним та потенційно вразливим компонентом будь-якої інтерактивної платформи для вивчення програмування є модуль автоматичної перевірки (автогрейдер). Запуск неперевіреного користувацького коду Python-скриптів безпосередньо на робочому сервері є неприпустимим, оскільки це створює низку критичних загроз безпеці:

– несанкціонований доступ - зловмисник може написати скрипт для читання системних файлів, доступу до бази даних або крадіжки вихідного коду платформи;

– відмова в обслуговуванні - навмисне або випадкове створення нескінченних циклів або виділення надмірної кількості пам'яті, що призведе до падіння сервера;

– мережеві атаки - використання сервера як проксі-вузла для здійснення DDoS-атак або розсилки спаму.

Для запобігання цим загрозам необхідно використовувати технологію «пісочниці» (Sandbox) - строго ізольованого середовища з жорстко обмеженими ресурсами. Розглянемо три основні підходи до вирішення цієї задачі.

2.5.1 Апаратна віртуалізація

Цей підхід передбачає запуск повноцінної гостьової операційної системи (наприклад, мінімального образу Linux) для кожного користувацького запиту.

Переваги: забезпечує найвищий рівень ізоляції на рівні гіпервізора та апаратного забезпечення. Навіть у разі злому гостьової ОС, хостовий сервер залишається у безпеці.

Недоліки: надзвичайно високе споживання ресурсів (кожна віртуальна машина вимагає виділення фіксованого обсягу RAM) та дуже довгий час запуску (від кількох секунд до хвилин). Це робить віртуальні машини абсолютно непридатними для синхронної перевірки коду, де студент очікує миттєвого результату (менше 3-5 секунд).

2.5.2 Клієнтське виконання

Сучасний підхід, який дозволяє компілювати інтерпретатор Python (наприклад, CPython) у формат WebAssembly і запускати код безпосередньо у веббраузері студента за допомогою бібліотеки Pyodide.

Переваги: повністю знімає обчислювальне навантаження з сервера. Код виконується миттєво в ізольованому середовищі браузера, що унеможливлює шкоду для серверної інфраструктури.

Недоліки: вразливість до академічної недоброчесності. Для перевірки коду на стороні клієнта необхідно передати в браузер студента всі приховані тести та еталонні відповіді. Кваліфікований студент може легко отримати до них доступ через панель розробника і підробити результати проходження тесту.

2.5.3 Контейнеризація на рівні ОС

Контейнери розділяють ядро однієї хостової операційної системи між багатьма ізольованими процесами.

Переваги: запуск нового контейнера займає лічені мілісекунди. Завдяки використанню механізмів Linux, контейнери дозволяють встановити жорсткі ліміти на використання процесорного часу та оперативної пам'яті. Вони забезпечують надійну ізоляцію файлової системи без необхідності емулювати апаратне забезпечення.

Недоліки: рівень ізоляції дещо нижчий порівняно з віртуальними машинами (оскільки ядро ОС спільне), проте при правильному налаштуванні прав доступу він є більш ніж достатнім для освітніх цілей.

Результати аналізу різних підходів до створення Sandbox-середовища зведено у таблиці 2.4.

Таблиця 2.4 - Порівняльна характеристика технологій ізоляції коду

Критерій порівняння	Віртуальні машини (VM)	WebAssembly (Pyodide)	Контейнери (Docker)
Місце виконання	Сервер	Клієнт (Браузер)	Сервер
Швидкість запуску	Повільна (секунди/хвилини)	Миттєва	Дуже швидка (мілісекунди)
Споживання ресурсів сервера	Дуже високе	Нульове	Низьке / Середнє
Складність контролю ресурсів (ліміти)	Легко	Складно	Легко
Захист від підробки результатів тестів	Високий	Низький (дані на клієнті)	Високий

Оскільки навчальна платформа вимагає компромісу між високою безпекою серверної інфраструктури, захистом тестів від списування та високою швидкістю надання зворотного зв'язку студенту, найкращим рішенням є використання технології контейнеризації Docker.

Для забезпечення безпечного середовища платформа динамічно створюватиме тимчасові Docker-контейнери для кожного надісланого розв'язку з наступними обмеженнями конфігурації.

Вимкнення доступу до мережі Інтернет (`--network none`):

– обмеження оперативної пам'яті (наприклад, `--memory 128m`), щоб уникнути вичерпання ресурсів;

– запуск коду від імені непривілейованого користувача (`non-root`) у контейнері зі статусом Read-Only, щоб унеможливити створення або зміну файлів;

– автоматичне знищення контейнера (`--rm`) відразу після завершення виконання скрипта або по закінченню відведеного ліміту часу (`timeout`).

В результаті проведеного аналізу сформовано наступний технологічний стек розробки:

– backend - мова Python;

– frontend - бібліотека React, мова JavaScript/TypeScript, редактор Monaco Editor;

– база даних - СКБД PostgreSQL;

– інфраструктура та безпека - Docker.

Обрані інструменти повністю відповідають сучасним інженерним стандартам веб-розробки та дозволяють у повній мірі реалізувати функціональні та нефункціональні вимоги, висунуті до навчальної платформи.

2.6 Аапаратна частина системи

Для забезпечення стабільної та безперебійної роботи розроблюваної навчальної вебплатформи з вивчення курсу «Основи програмування на Python» необхідно провести ретельний аналіз та вибір апаратного забезпечення.

Специфіка даного програмного продукту полягає в тому, що він поєднує в собі як класичну систему управління навчальним контентом, так і вимогливе до обчислювальних ресурсів середовище для безпечного виконання користувацького коду в режимі реального часу, що вимагає специфічних підходів до розподілу

процесорного часу та оперативної пам'яті. Оскільки архітектура системи побудована за клієнт-серверною моделлю, апаратне забезпечення логічно розділяється на серверну частину, де відбуваються основні обчислення, та клієнтську частину, яка відповідає за відображення інтерфейсу та взаємодію з користувачем. Загальна діаграма компонентів розгортання [18] апаратних та програмних вузлів платформи представлена нижче (рис. 2.1).

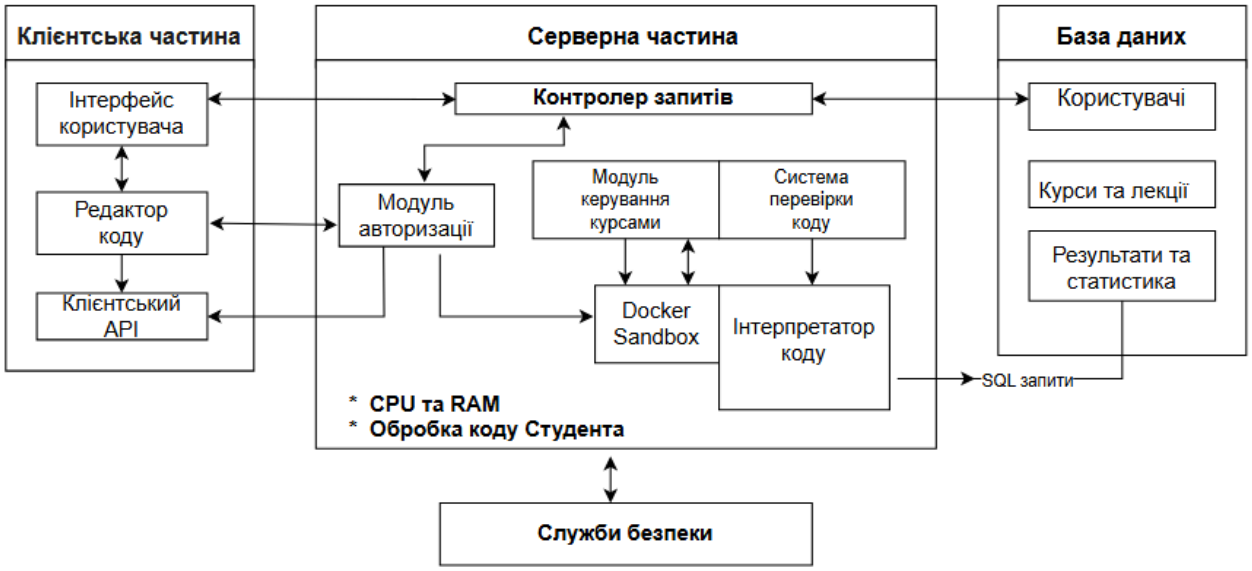

Рисунок 2.1 - Діаграма компонентів розгортання апаратних вузлів системи

Як видно з наведеної діаграми розгортання, основне навантаження припадає на сервер додатків та обчислень, оскільки саме там відбувається ініціалізація ізольованих контейнерів Docker для перевірки коду студентів. Кожен запуск коду вимагає виділення окремого обсягу оперативної пам'яті та процесорних квот, тому сервер повинен мати достатній запас ресурсів для одночасної обробки запитів від усієї академічної групи. Для забезпечення достатнього рівня продуктивності бази даних доцільно виділити її на окремий апаратний вузол або використовувати високошвидкісні твердотільні накопичувачі (NVMe SSD), які значно пришвидшують операції читання та запису при роботі з реляційними даними.

Зведені мінімальні та рекомендовані вимоги до апаратного забезпечення серверної частини наведені у таблиці (табл. 2.5).

Таблиця 2.5 - Вимоги до апаратного забезпечення сервера

Компонент	Мінімальні вимоги	Рекомендовані вимоги
Процесор (CPU)	4 ядра (наприклад, Intel Xeon або AMD EPYC)	8 ядер та більше для паралельних обчислень
Оперативна пам'ять (RAM)	8 ГБ	16 ГБ або більше для кешування та платформи Docker
Дисковий простір	50 ГБ SSD	100 ГБ NVMe SSD для швидкого доступу до БД
Мережевий інтерфейс	100 Мбіт/с	1 Гбіт/с для швидкої передачі статичних файлів

Обґрунтовуючи дані, наведені у вимогах таблиці, варто зазначити, що оперативна пам'ять є найбільш критичним ресурсом, оскільки кожен ізольований запуск Python-скрипта може споживати від 50 до 100 мегабайт пам'яті, а при одночасному тестуванні десятків рішень цей показник стрімко зростає. Щоб уникнути перевантаження апаратного забезпечення сервера та аварійного завершення процесів, програмно встановлюються жорсткі ліміти на використання ресурсів на рівні ядра операційної системи. Нижче наведено приклад конфігурації, яка обмежує споживання апаратних ресурсів для одного користувацького завдання (Лістинг 2.1).

Лістинг 2.1 - Конфігурація обмеження апаратних ресурсів для ізольованого контейнера

```
Python
container = client.containers.run(
    image=«python:3.10-slim»,
    command=f»python3 -c \»{user_code}\»«,
    mem_limit=«128m»,            # Обмеження оперативної пам'яті
    memswap_limit=«128m»,        # Відключення файлу підкачки для
контейнера
    cpu_period=100000,
    cpu_quota=50000,             # Використання максимум 50%
потужності одного ядра CPU
```

```
    network_disabled=True,      # Відключення мережевого адаптера
для безпеки
    detach=True
)
```

Розглядаючи клієнтську частину апаратного забезпечення, слід враховувати парадигму «тонкого клієнта», яка масово застосовується при розробці сучасних вебдодатків. Оскільки компіляція та виконання програмного коду, а також перевірка тестів відбуваються виключно на стороні сервера, вимоги до пристроїв кінцевих користувачів є мінімальними, що робить платформу інклюзивною та доступною для студентів з будь-яким технічним оснащенням. Основні процеси на стороні клієнта включають рендеринг інтерфейсу за допомогою браузерного рушія JavaScript та забезпечення роботи вбудованого редактора коду Monaco Editor [19]. Характеристики, необхідні для комфортної роботи користувача з вебплатформою, підсумовані у наступній таблиці (табл. 2.6.).

Таблиця 2.6 - Вимоги до апаратного забезпечення клієнтського пристрою

Компонент	Вимоги для стабільної роботи системи
Процесор (CPU)	Двоядерний процесор з тактовою частотою від 1.6 ГГц
Оперативна пам'ять (RAM)	4 ГБ (достатньо для стабільної роботи сучасного веббраузера)
Дисплей	Роздільна здатність від 1280x720 (рекомендовано 1920x1080 для роботи з кодом
Пристрої введення	Клавіатура, комп'ютерна миша або вбудований тачпад
Мережа	Стабільне підключення до Інтернету зі швидкістю від 2 Мбіт/с

Для кращого розуміння взаємодії апаратних компонентів під час виконання типового сценарію перевірки коду, можна розглянути схему розподілу обчислювальних ресурсів. Коли користувач надсилає запит, відбувається наступний ланцюжок виділення апаратних потужностей:

– мережевий адаптер клієнта передає пакет даних через маршрутизатори до мережевої карти сервера;

– центральний процесор сервера обробляє отриманий запит та виділяє оперативну пам'ять для фонового процесу-обробника;

– дискова підсистема виконує читання еталонних тестів з бази даних;

– підсистема віртуалізації резервує ізольований сегмент оперативної пам'яті та процесорні такти для виконання алгоритму.

Такий чіткий розподіл навантаження між апаратними складовими гарантує швидкий відгук інтерфейсу та унеможливлює зависання клієнтського пристрою незалежно від складності написаних студентом програм, що повністю задовольняє сучасні інженерні вимоги до створення електронних освітніх платформ.

3 РЕАЛІЗАЦІЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

3.1 Проєктування архітектури програмного продукту

Для реалізації навчальної вебплатформи обрано класичну клієнт-серверну архітектуру з використанням архітектурного стилю REST. Система будується за принципом розділення відповідальності, де клієнтська частина та серверна частина є незалежними додатками, які обмінюються даними виключно через мережеві HTTP-запити у форматі JSON [20].

Такий підхід, на відміну від монолітних систем, забезпечує високу гнучкість, спрощує масштабування та дозволяє в майбутньому легко підключити мобільний додаток, використовуючи існуючий серверний API.

Логічно архітектуру платформи можна розділити на три основні рівні:

– рівень представлення (клієнтський рівень) - односторінковий вебдодаток на базі React;

– рівень бізнес-логіки (серверний рівень) - API-сервер на базі Django REST Framework та ізольоване середовище виконання коду Docker Sandbox;

– рівень даних - об'єктно-реляційна база даних PostgreSQL.

3.1.1 Архітектура клієнтської частини

Клієнтський додаток завантажується в браузер користувача один раз під час першого відвідування сайту. Усі подальші переходи між сторінками (наприклад, від списку лекцій до вікна редактора коду) відбуваються миттєво без перезавантаження сторінки завдяки клієнтській маршрутизації React Router.

Основні компоненти клієнтської архітектури:

– модуль авторизації - відповідає за зберігання сесії користувача, взаємодія із сервером здійснюється за допомогою JWT-токенів (JSON Web Token), які додаються до заголовків кожного запиту для ідентифікації студента або викладача;

– інтерфейс користувача - набір незалежних React-компонентів (навігаційна панель, список курсів, дашборд статистики);

– модуль редактора коду - інтегрована бібліотека Monaco Editor, яка обробляє введення тексту, забезпечує підсвічування синтаксису Python та відстежує стан написаного студентом коду перед його відправкою на сервер.

3.1.2 Архітектура серверної частини та бази даних

Серверна частина, розроблена на базі Django, виконує роль центрального диспетчера. Вона обробляє вхідні HTTP-запити від клієнта, виконує перевірку прав доступу, взаємодіє з базою даних та формує відповіді.

Взаємодія з рівнем даних PostgreSQL відбувається за допомогою вбудованої системи об'єктно-реляційного відображення Django ORM. Це дозволяє абстрагуватися від написання «сирих» SQL-запитів і працювати з таблицями бази даних як з об'єктами класів Python (наприклад, User, Course, Task, Attempt).

3.1.3 Архітектура підсистеми автоматичної перевірки коду

Найскладнішим архітектурним вузлом платформи є процес обробки та виконання студентського коду. Оскільки запуск стороннього коду є ресурсомістким і потенційно небезпечним процесом, він повинен виконуватися асинхронно та в ізольованому середовищі.

Життєвий цикл перевірки практичного завдання виглядає наступним чином:

а) ініціалізація:

1) студент натискає кнопку «Перевірити код» у браузері:

- React-додаток формує POST-запит, який містить ідентифікатор завдання та написаний Python-код у вигляді рядка тексту;

б) прийом запиту:

1) сервер Django приймає запит, перевіряє JWT-токен (чи має право цей студент виконувати це завдання) та створює в базі даних запис про нову спробу (Attempt) зі статусом «В процесі» (Pending);

в) передача в чергу (Асинхронність):

1) задача передається у фоновий брокер повідомлень (наприклад, Celery + Redis):

- сервер негайно повертає клієнту відповідь, що код прийнято в обробку.

г) виконання в Sandbox:

1) фоновий процес (Worker) бере задачу з черги:

- зчитує з бази даних приховані тести (вхідні та очікувані вихідні дані) для цього завдання;

- ініціює запуск тимчасового Docker-контейнера з образом Python (наприклад, python:3.11-alpine);

- контейнер запускається з жорсткими обмеженнями (відсутність мережі, ліміт RAM у 128 МБ, ліміт часу у 5 секунд);

- код студента та тести передаються в контейнер через стандартний потік вводу (stdin) або монтування тимчасового файлу;

д) збір результатів:

1) після завершення роботи скрипта (або після його примусового завершення через перевищення ліміту часу - Timeout), фоновий процес збирає дані зі стандартного потоку виводу (stdout) та потоку помилок (stderr);

е) знищення середовища:

1) Docker-контейнер негайно видаляється з пам'яті сервера;

ж) оцінювання:

1) фоновий процес порівнює фактичний вивід програми з еталонним:

- на основі результату в базі даних статус спроби оновлюється на «Успішно» (Passed) або «Помилка» (Failed/Syntax Error);

к) відображення клієнту:

1) Frontend-додаток, який періодично опитує сервер або використовує технологію WebSockets, отримує оновлений статус і відображає результати студенту.

Спроєктована мікросервісно-орієнтована архітектура гарантує високий рівень безпеки сервера. Використання асинхронних черг для запуску коду забезпечує стабільну роботу платформи навіть під час пікових навантажень (наприклад, під час складання підсумкового модульного контролю всією академічною групою одночасно).

3.2 Проєктування бази даних

Для забезпечення надійного зберігання даних, підтримки цілісності інформації та високої швидкості обробки запитів необхідно розробити раціональну структуру бази даних. Оскільки для реалізації обрано реляційну СКБД PostgreSQL, проєктування бази даних передбачає визначення основних сутностей, їх атрибутів та встановлення логічних зв’язків між ними.

3.2.1 Інфологічна модель бази даних

Логічна структура бази даних навчальної платформи складається з кількох взаємопов'язаних модулів (рис.3.1) [21]:

- управління користувачами;
- структура навчального контенту;
- система обліку практичних завдань.

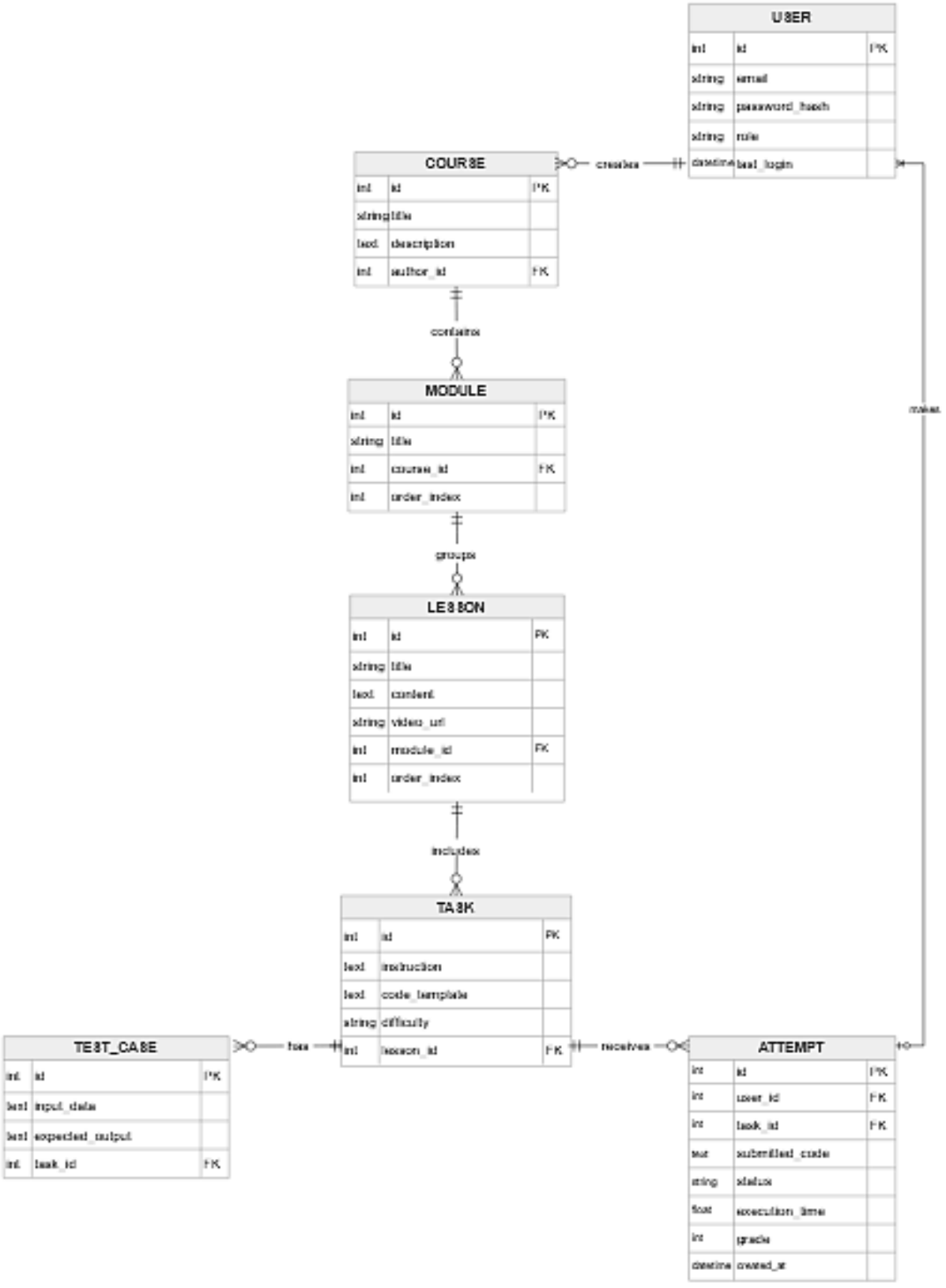

Рисунок 3.1 – ER модель бази даних

Основними сутностями системи є:

– «Користувач» (User) - зберігає облікові дані, інформацію про роль (студент, викладач, адміністратор) та дату останнього входу;

– «Курс» (Course) -базова одиниця контенту, що містить назву, опис та автора;

– «Модуль» (Module) - логічний розділ курсу, що групує уроки за темами;

– «Урок» (Lesson) - містить теоретичний матеріал у форматі HTML або Markdown та посилання на відео лекції;

– «Завдання» (Task) - практична вправа з програмування, що містить умову, початковий шаблон коду та рівень складності;

– «Тестовий випадок» (TestCase) - набір прихованих вхідних і очікуваних вихідних даних для автоматичної перевірки завдання;

– «Спроба» (Attempt) - результат відправки коду студентом, що зберігає сам код, статус перевірки, час виконання та отриману оцінку.

3.2.2 Специфікація таблиць бази даних

Нижче наведено опис структури основних таблиць, які будуть реалізовані за допомогою Django ORM (табл..3.1 – 3.3).

Таблиця 3.1 - Структура таблиці users_user (Користувачі)

Назва поля	Тип даних	Опис
id	BigInt (PK)	Унікальний ідентифікатор користувача
username	Varchar(150)	Логін для входу
email	Varchar(254)	Електронна пошта
password	Varchar(128)	Хеш-сума пароля
role	Enum	Роль: student, teacher, admin
date_joined	DateTime	Дата реєстрації

Таблиця 3.2- Структура таблиці tasks_task (Практичні завдання)

Назва поля	Тип даних	Опис
id	BigInt (PK)	Ідентифікатор завдання
lesson_id	ForeignKey	Зв'язок з таблицею уроків (1:M)
title	Varchar(200)	Назва завдання
description	Text	Детальна умова завдання
boilerplate	Text	Початковий код (шаблон) для студента
time_limit	Float	Максимальний час виконання (сек)
memory_limit	Integer	Максимальна пам'ять (МБ)

Таблиця 3.3 - Структура таблиці tasks_attempt (Спроби виконання)

Назва поля	Тип даних	Опис
id	BigInt (PK)	Ідентифікатор спроби
task_id	ForeignKey	Зв'язок із завданням (1:M)
student_id	ForeignKey	Зв'язок із користувачем (1:M)
code	Text	Вихідний код, надісланий студентом
status	Varchar(20)	Статус: Pending, Passed, Failed, Error
execution_time	Float	Фактичний час виконання
created_at	DateTime	Дата та час відправки

3.2.3 Забезпечення цілісності та зв'язків

Зв’язки між таблицями реалізовані за допомогою зовнішніх ключів з наступними правилами каскадності:

– CASCADE - при видаленні курсу автоматично видаляються всі пов’язані з ним модулі та уроки;

– PROTECT - неможливо видалити завдання, якщо в базі існують спроби його виконання студентами (для збереження історії успішності);

– індексація - для полів username, email та зовнішніх ключів будуть створені індекси (B-Tree), що дозволить пришвидшити пошук та фільтрацію даних при великій кількості користувачів.

Спроєктована реляційна модель бази даних повністю охоплює всі бізнес-процеси навчальної платформи. Використання PostgreSQL дозволяє не лише надійно зберігати текстові дані, а й ефективно оперувати великими обсягами коду та результатами тестування, забезпечуючи високу швидкість доступу до інформації та масштабованість системи в майбутньому.

3.3 Об’єктно-орієнтоване проєктування системи

Об’єктно-орієнтований аналіз та проєктування дозволяють візуалізувати функціональні вимоги до платформи та визначити послідовність взаємодії між користувачами та програмними модулями. Для цього використано два основні типи діаграм: діаграму прецедентів та діаграму послідовностей [22].

3.3.1 Розробка діаграми прецедентів

Діаграма прецедентів описує функціональність системи з точки зору зовнішніх користувачів. Вона дозволяє чітко розмежувати права доступу та визначити основні сценарії використання вебплатформи.

Для розроблюваної системи визначено три ролі користувачів (рис.3.2-3.4):

а) роль «Студент»:

1) авторизація та керування профілем;

2) перегляд списку доступних курсів та лекцій;

3) виконання практичних завдань у вбудованому редакторі коду;

4) запуск коду на виконання та отримання результатів тестування;

5) перегляд особистого рейтингу та статистики успішності;

б) роль «Викладач»:

1) наповнення курсів теоретичним матеріалом;

2) створення та редагування практичних завдань (включно з написанням тестів);

3) моніторинг прогресу академічних груп;

4) перевірка коду студентів;

в) роль «Адміністратор»:

1) керування обліковими записами користувачів;

2) модерація контенту та технічне обслуговування системи.

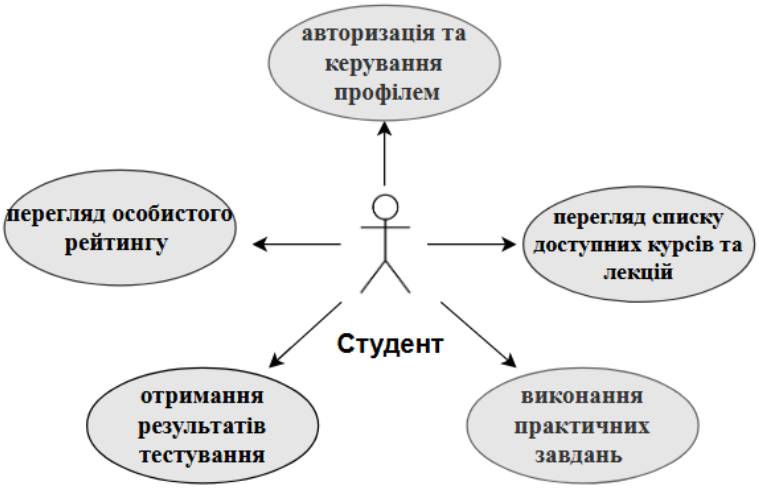

Рисунок 3.2 - Діаграми прецедентів ролі «Студент»

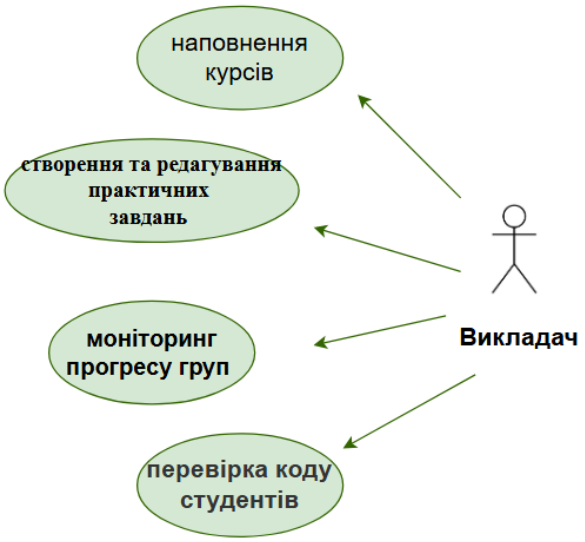

Рисунок 3.3 - Діаграми прецедентів ролі «Викладач»

Рисунок 3.4 - Діаграми прецедентів ролі «Адміністратор»

3.3.2 Розробка діаграми послідовностей

Діаграма послідовностей візуалізує динаміку взаємодії об’єктів у часі. Найбільш критичним процесом для даної платформи є сценарій відправки студентського коду на автоматичну перевірку.

Опис процесу взаємодії:

– «Користувач» (Студент) вводить код у веб-редакторі React та натискає кнопку «Перевірити»;

– Frontend-додаток відправляє HTTP-запит POST з ідентифікатором завдання та вихідним кодом до Backend-сервера Django;

– Backend перевіряє авторизацію користувача, зберігає спробу в БД та ініціює асинхронну задачу в Celery;

– Worker отримує задачу, звертається до Docker API та створює ізольований контейнер;

– код виконується всередині Docker-контейнера на наборі прихованих тестів. Результати повертаються до Worker-а;

– Backend оновлює статус спроби в базі даних PostgreSQL;

– Frontend отримує оновлені дані через Polling або WebSockets та відображає вердикт студенту.

Для наочного представлення логіки роботи системи на ристунку 3.5 наведена інтерактивна модель діаграми послідовностей. Вона демонструє шлях проходження коду від моменту написання до отримання оцінки.

Використання UML-діаграм дозволило формалізувати процеси взаємодії користувачів із платформою та детально спроектувати механізм асинхронної перевірки коду. Це забезпечує цілісне бачення архітектури системи перед початком етапу безпосередньої програмної реалізації та допомагає уникнути логічних помилок при побудові складних клієнт-серверних зв’язків.

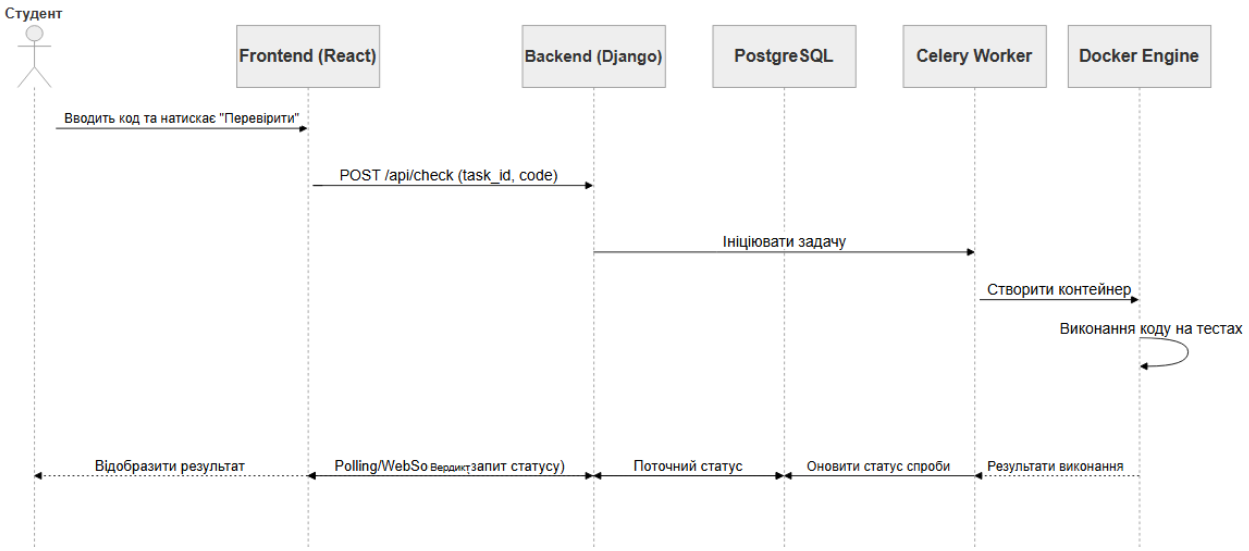

Рисунок 3.5 - Інтерактивна модель діаграми послідовностей

Отже, побудова діаграми послідовностей дала змогу не лише відобразити ключовий сценарій роботи платформи, а й формалізувати логіку взаємодії між її компонентами. Такий підхід забезпечує прозорість процесів, підвищує надійність архітектури та створює основу для подальшої реалізації й тестування системи.

3.4 Реалізація клієнтської частини

Розробка клієнтської частини навчальної вебплатформи здійснювалася на базі компонентної бібліотеки React [23]. Для забезпечення статичної типізації, зменшення кількості помилок під час виконання та покращення якості коду було використано мову TypeScript. Збірка проєкту налаштована за допомогою сучасного інструменту Vite, що гарантує високу швидкість гарячого перезавантаження модулів у середовищі розробки та оптимізований бандл для кінцевого використання.

3.4.1 Структура компонентів та маршрутизація

Клієнтський додаток побудований за принципом SPA. Маршрутизація між сторінками реалізована за допомогою бібліотеки react-router-dom. Інтерфейс

розділено на незалежні перевикористовувані компоненти.

Ключовим екраном платформи є сторінка виконання практичного завдання «TaskExecutionPage», яка функціонально поділена на три ізольовані зони:

– панель з умовою задачі (ліва частина);

– область редагування коду (центральна частина);

– панель терміналу для відображення результатів тестування (нижня частина).

3.4.2 Інтеграція та налаштування Monaco Editor

Для забезпечення повноцінного досвіду програмування безпосередньо у веббраузері, у платформу було інтегровано Monaco Editor - потужний рушій, на якому базується популярний редактор Visual Studio Code. Для зручної роботи з життєвим циклом компонентів React використано спеціалізований пакет @monaco-editor/react [24].

Під час налаштування компонента редактора було визначено наступні ключові параметри:

Увімкнено підсвічування синтаксису виключно для мови Python:

– застосовано темну тему оформлення (vs-dark), яка знижує навантаження на очі студентів;

– вимкнено відображення мінікарти коду (minimap: false) для економії робочого простору екрана;

– налаштовано автоматичне масштабування (automaticLayout: true) для коректного відображення під час зміни розміру вікна браузера.

Лістинг 3.1 демонструє програмну реалізацію React-компонента обгортки для редактора коду.

Лістинг 3.1 - Реалізацію React-компонента обгортки для редактора коду

```
TypeScript
// Лістинг 3.1. Компонент CodeEditor.tsx
import React, { useState } from 'react';
import Editor from '@monaco-editor/react';

interface CodeEditorProps {
```

```
    initialCode: string;
    onChange: (code: string) => void;
}

const CodeEditor: React.FC<CodeEditorProps> = ({ initialCode,
onChange }) => {
    const [code, setCode] = useState<string>(initialCode);

    const handleEditorChange = (value: string | undefined) => {
        const newValue = value || ««;
        setCode(newValue);
        onChange(newValue); // Передача оновленого коду до
батьківського компонента
    };

    return (
        <div className=«editor-container» style={{ height: '100%',
width: '100%' }}>
            <Editor
                height=«100%»
                defaultLanguage=«python»
                theme=«vs-dark»
                value={code}
                onChange={handleEditorChange}
                options={{
                    minimap: { enabled: false },
                    fontSize: 14,
                    wordWrap: «on»,
                    scrollBeyondLastLine: false,
                    automaticLayout: true,
                }}
            />
        </div>
    );
};

export default CodeEditor;
```

3.4.3 Взаємодія з Backend API та обробка асинхронних результатів

Оскільки виконання коду на стороні сервера у Docker відбувається асинхронно, фронтенд повинен вміти обробляти затримку між відправкою розв'язку та отриманням фінальної оцінки.

Для цього реалізовано механізм «опитування» (Long Polling):

– після натискання студентом кнопки «Запустити код», клієнт відправляє POST-запит на API та миттєво отримує ідентифікатор спроби (`attempt_id`);

– інтерфейс переходить у стан завантаження (відображається спінер-індикатор);

– клієнтський додаток використовує функцію setInterval, яка кожні 1.5 секунди надсилає GET-запит на сервер для перевірки статусу за отриманим `attempt_id`;

– як тільки статус змінюється з `pending` на `passed`, `failed` або `error`, таймер опитування зупиняється, а результат тестування (вивід програми та наявність помилок) відображається у стилізованому вікні вебтерміналу.

Такий підхід забезпечує плавність роботи інтерфейсу та інформує користувача про те, що система обробляє його запит, не викликаючи зависання браузера.

3.5 Програмна реалізація ізольованого середовища виконання коду

Найважливішим етапом розробки платформи є реалізація модуля автоматичної перевірки (автогрейдера), який повинен безпечно виконувати довільний Python-код, надісланий студентом. Для вирішення цієї задачі було розроблено систему, що поєднує фонову чергу завдань Celery та технологію контейнеризації Docker.

Процес перевірки реалізовано у вигляді асинхронної задачі. Вибір асинхронної моделі зумовлений тим, що запуск Docker-контейнера та виконання тестів може тривати кілька секунд. Використання фонового процесу дозволяє основному вебсерверу залишатися вільним для обробки запитів інших користувачів.

Для взаємодії з демоном Docker із коду на Python використано бібліотеку `docker-py` (Docker SDK for Python). Це дозволяє програмно керувати життєвим циклом контейнерів. Код студента виконується всередині легкового образу python:3.10-slim.

Для забезпечення безпеки під час запуску контейнера застосовуються наступні жорсткі обмеження:

– `network_disabled=True` - повне відключення доступу до мережі Інтернет для запобігання витоку даних або мережевих атак;

– `mem_limit=«128m»` - обмеження оперативної пам'яті для захисту від атак типу «витоку пам'яті»;

– `cpu_quota=50000` (50% одного ядра) - обмеження використання процесора;

– `read_only=True` - монтування файлової системи контейнера в режимі «тільки для читання».

Лістинг 3.2 демонструє спрощену логіку роботи фонового воркера, який виконує код студента.

Лістинг 3.2 - Логіка роботи фонового воркера

```
Python
# Лістинг 3.2. Програмна реалізація Sandbox-воркера (tasks.py)
import docker
import time
from celery import shared_task
from .models import Attempt

client = docker.from_env()

@shared_task
def run_code_in_sandbox(attempt_id):
    attempt = Attempt.objects.get(id=attempt_id)
    task = attempt.task

    # Підготовка команди для запуску коду та тестів
    # У реальній системі сюди додається механізм перевірки через
unit-тести
    user_code = attempt.code

    try:
        # Запуск ізольованого контейнера
        container = client.containers.run(
            image=«python:3.10-slim»,
            command=f»python3 -c \»{user_code}\»«,
            detach=True,
            network_disabled=True,
            mem_limit=«128m»,
            stderr=True,
```

```
            stdout=True
        )

        # Очікування завершення або примусова зупинка за таймаутом
        try:
            result = container.wait(timeout=task.time_limit)
            output = container.logs(stdout=True,
stderr=False).decode('utf-8')
            errors = container.logs(stdout=False,
stderr=True).decode('utf-8')

            if result['StatusCode'] == 0:
                attempt.status = 'passed'
            else:
                attempt.status = 'failed'

            attempt.execution_output = output or errors
        except Exception:
            container.kill()
            attempt.status = 'failed'
            attempt.execution_output = «Перевищено ліміт часу
виконання (Timeout)»

        container.remove()

    except Exception as e:
        attempt.status = 'error'
        attempt.execution_output = str(e)

    attempt.save()
```

Механізм перевірки не обмежується лише запуском коду. Система підтримує порівняння результатів роботи програми студента з набором прихованих тестів.

Кожен тест складається з двох частин:

`input_data` - подається на стандартний вхід stdin;

`expected_output` - система очікує побачити у stdout.

Спроба вважається успішною лише у випадку, якщо програма студента видала ідентичний еталону текст для всіх тестових випадків, передбачених викладачем для даного завдання.

Програмна реалізація платформи базується на поєднанні сучасних вебтехнологій Django, React та засобів системної ізоляції Docker. Створена архітектура дозволяє безпечно вивчати основи програмування на Python у

браузері, забезпечуючи миттєвий зворотний зв’язок та автоматизацію процесу оцінювання успішності студентів.

4 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ СИСТЕМИ

4.1 Функціональне тестування системи

Функціональне тестування проводилося методом «чорного ящика» з метою перевірки правильності реалізації бізнес-логіки для кожної ролі користувача (Студент, Викладач, Адміністратор). Результати наведено у таблиці 4.1.

Таблиця 4.1 - Результати функціонального тестування основних модулів

Назва тест-кейсу	Опис дії	Очікуваний результат	Фактичний результат	Статус
Реєстрація та вхід	Користувач вводить email, логін та пароль	Створення запису в БД, генерація JWT-токена, перехід на головну сторінку	Токен успішно згенеровано та збережено в LocalStorage	Успішно
Створення завдання (Викладач)	Викладач створює нове завдання, додає умову та приховані тести (Input/Output)	Завдання з'являється у відповідному модулі курсу	Завдання коректно відображається у базі даних та доступне студентам	Успішно
Виконання завдання (Студент)	Студент вводить синтаксично правильний код, що вирішує задачу, та натискає «Перевірити»	Статус спроби змінюється на Pending, потім на Passed. Нарахування балів	Сервер повертає статус Passed, вивід програми збігається з еталоном	Успішно

Продовження таблиці 4.1

Назва тест-кейсу	Опис дії	Очікуваний результат	Фактичний результат	Статус
Синтаксич-на помилка	Студент надсилає код із пропущеною двокрапкою (SyntaxError	Статус змінюється на Error. Студент бачить текст помилки з логів Python.	Термінал браузера коректно відображає рядок, де виникла помилка (Traceback).	Успішно

Усі базові сценарії використання платформи відпрацювали коректно, що підтверджує правильність інтеграції клієнтської та серверної частин.

4.2 Тестування безпеки та ізоляції модуля Sandbox

Оскільки платформа дозволяє виконання довільного коду користувачів, найважливішим етапом є тестування системи безпеки на стійкість до зловмисних дій (вразливостей). Тестування проводилося шляхом надсилання шкідливих Python-скриптів [25].

Сценарій 1: спроба несанкціонованого доступу до файлової системи сервера (RCE). Студент намагається прочитати конфігураційні файли хостової машини (наприклад, файл з паролями) або видалити файли проєкту:

– надісланий код:

```
Python
import os
print(os.popen('cat /etc/passwd').read())
```

– результат роботи - код виконується всередині ізольованого Docker-контейнера, скрипт виводить вміст файлу `/etc/passwd` самого контейнера (який містить лише стандартного root користувача легковагового образу Linux), а не хостового сервера;

– доступу до вихідного коду платформи Django немає - загрозу успішно нейтралізовано.

Сценарій 2: атака типу «Відмова в обслуговуванні» - нескінченний цикл. Студент навмисно (чи випадково) створює цикл, який ніколи не завершується, щоб перевантажити процесор сервера:

– надісланий код:

```Python
while True:
    pass
```

– результат роботи - контейнер починає споживати виділені йому 50% процесорного часу. Через 5 секунд (встановлений time_limit для завдання) фоновий процес Celery примусово надсилає сигнал SIGKILL контейнеру. Статус спроби в базі даних оновлюється на «Failed (Timeout)». Робота платформи для інших користувачів не переривається.

Сценарій 3: вичерпання оперативної пам'яті (Memory Bomb). Студент намагається виділити гігабайти пам'яті, генеруючи величезні масиви даних, щоб викликати падіння сервера (Out of Memory):

– надісланий код:

```Python
memory_killer = []
while True:
    memory_killer.append(' ' * 10**6) # Додаємо по 1 МБ у масив
```

– результат роботи - як тільки скрипт досягає встановленого ліміту пам'яті у 128 МБ (`mem_limit=«128m»`), підсистема ядра Linux примусово знищує процес. Воркер фіксує помилку виконання і повертає студенту статус «Error: Memory Limit Exceeded».

Контейнерна ізоляція успішно витримала всі симульовані атаки. Жоден шкідливий скрипт не зміг порушити працездатність головного сервера або отримати доступ до конфіденційних даних інших користувачів.

4.3 Оцінка продуктивності та навантажувальне тестування

Для оцінки того, як платформа впорається з реальними умовами навчального процесу (наприклад, під час проведення модульного контролю, коли

вся академічна група одночасно надсилає код), було використано інструмент Apache JMeter.

В ході тестування було згенеровано 50 одночасних запитів на компіляцію та перевірку коду:

– обробка черги - завдяки використанню брокера повідомлень Redis та черги Celery, жоден запит не було втрачено, усі вони були поставлені в чергу;

– час відгуку - середній час очікування студентом результату збільшився з 1.5 секунди (при одиночному запиті) до 4.8 секунди (при піковому навантаженні);

– масштабованість - тестування підтвердило, що архітектура дозволяє легко масштабувати продуктивність шляхом збільшення кількості Worker-вузлів (додавання нових серверів для обробки черги Celery).

4.4 Аналіз користувацького інтерфейсу

Проведено візуальне та ергономічне тестування клієнтської частини додатка.

Інтерфейс системи має адаптивний дизайн, що забезпечує коректне масштабування на будь-яких пристроях, включно з мобільними. Навігаційне меню автоматично перетворюється на «гамбургер», а шрифти залишаються чіткими та зручними для читання.

Вбудований редактор Monaco Editor працює стабільно й без затримок введення, підтримує автодоповнення коду та підсвічування синтаксису Python, що створює досвід, близький до настільних IDE.

Зворотний зв’язок реалізовано через анімації завантаження та кольорову індикацію результатів тестування - зелений сигналізує про успішне виконання, червоний про помилку.

Така візуальна логіка робить взаємодію інтуїтивною й сприяє позитивному користувацькому досвіду (рис.4.2 – 4.6).

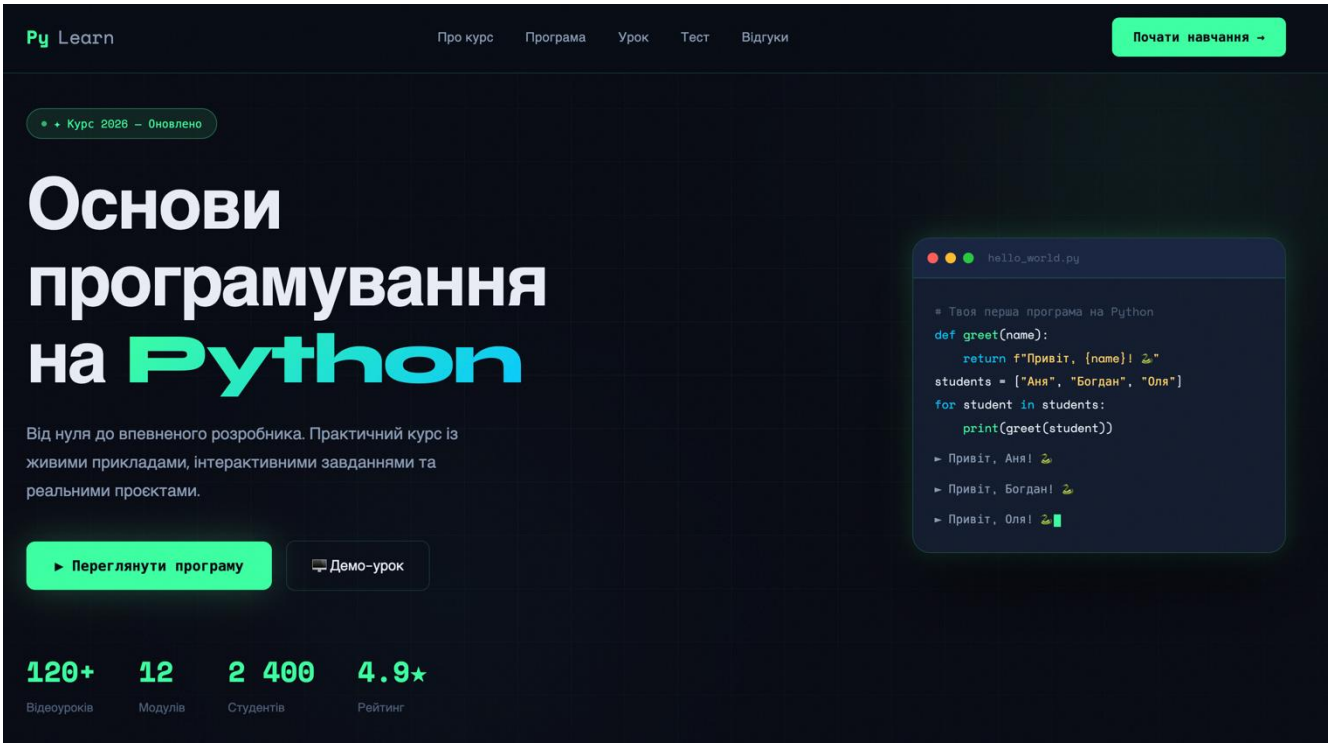

Рисунок 4.2 – Головна сторінка

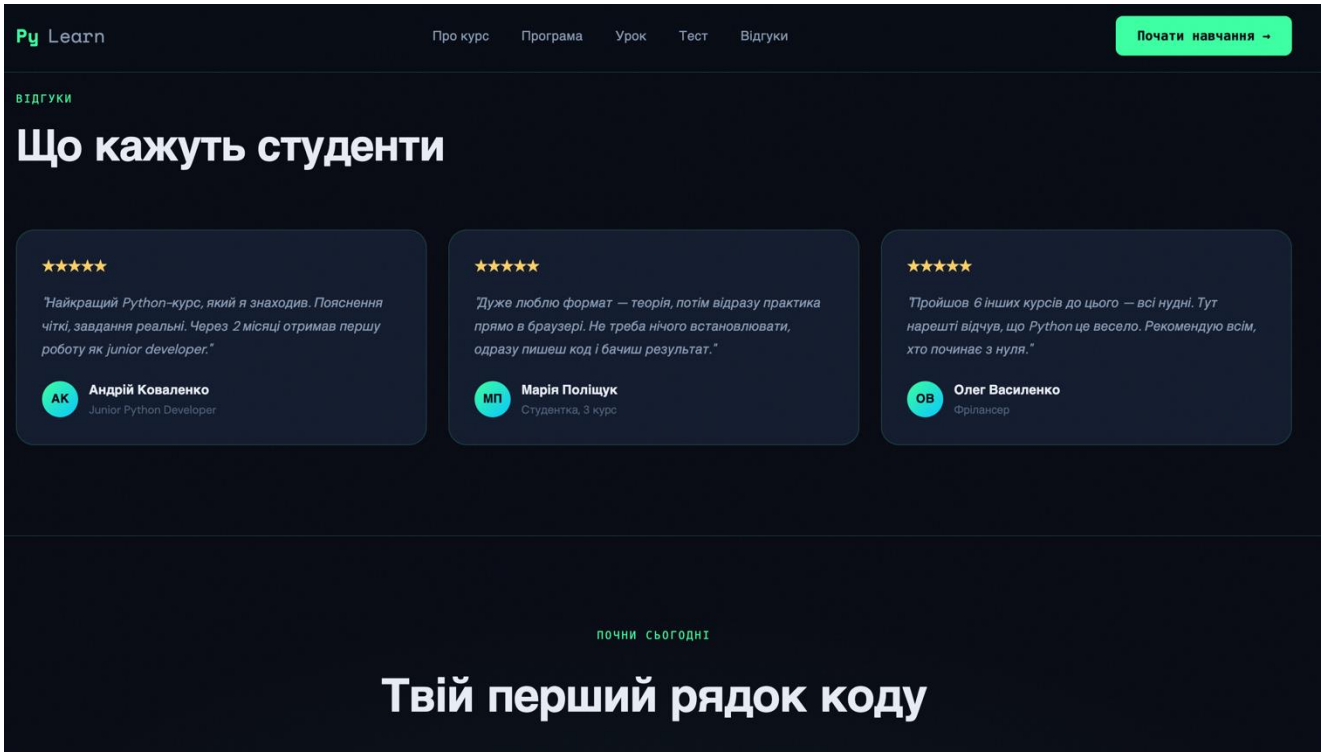

Рисунок 4.3 – Сторінка «Відгуки»

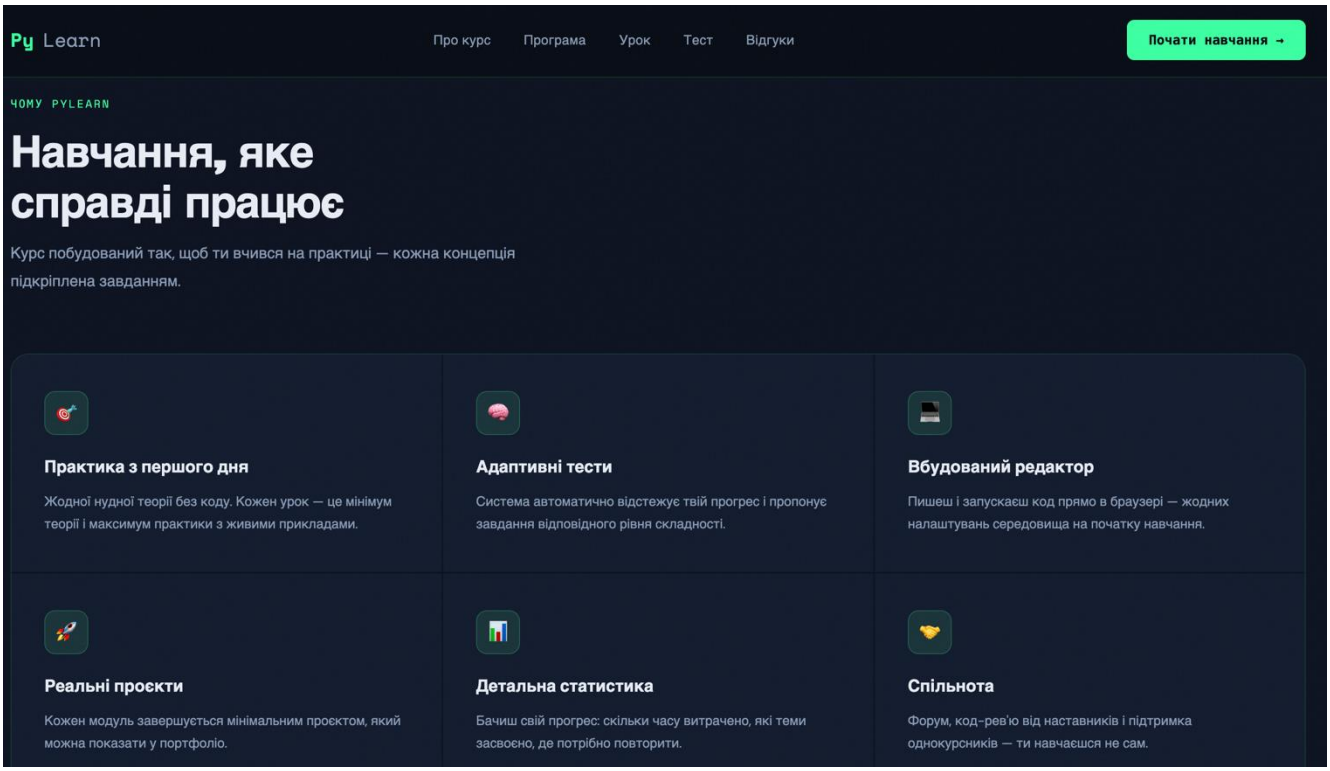

Рисунок 4.4 – Сторінка «Основні дані»

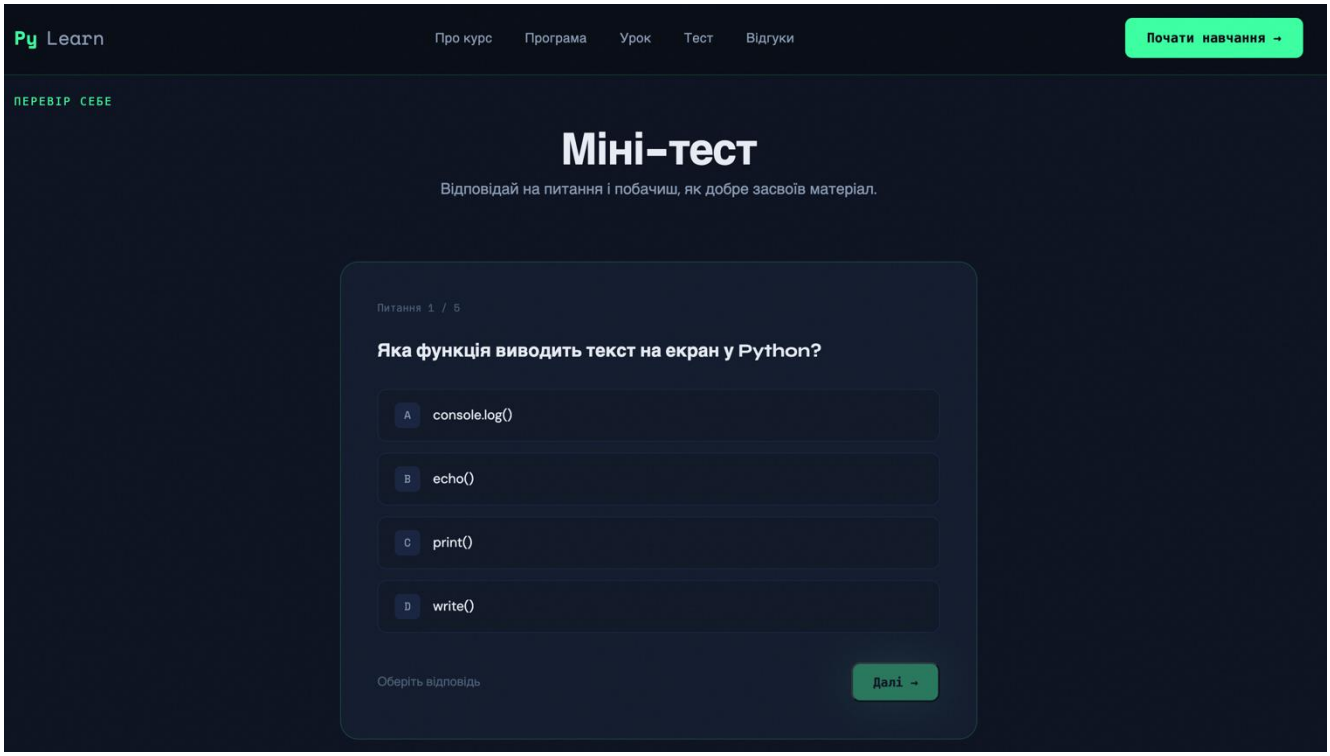

Рисунок 4.5– Сторінка «Тест на перевірку знань»

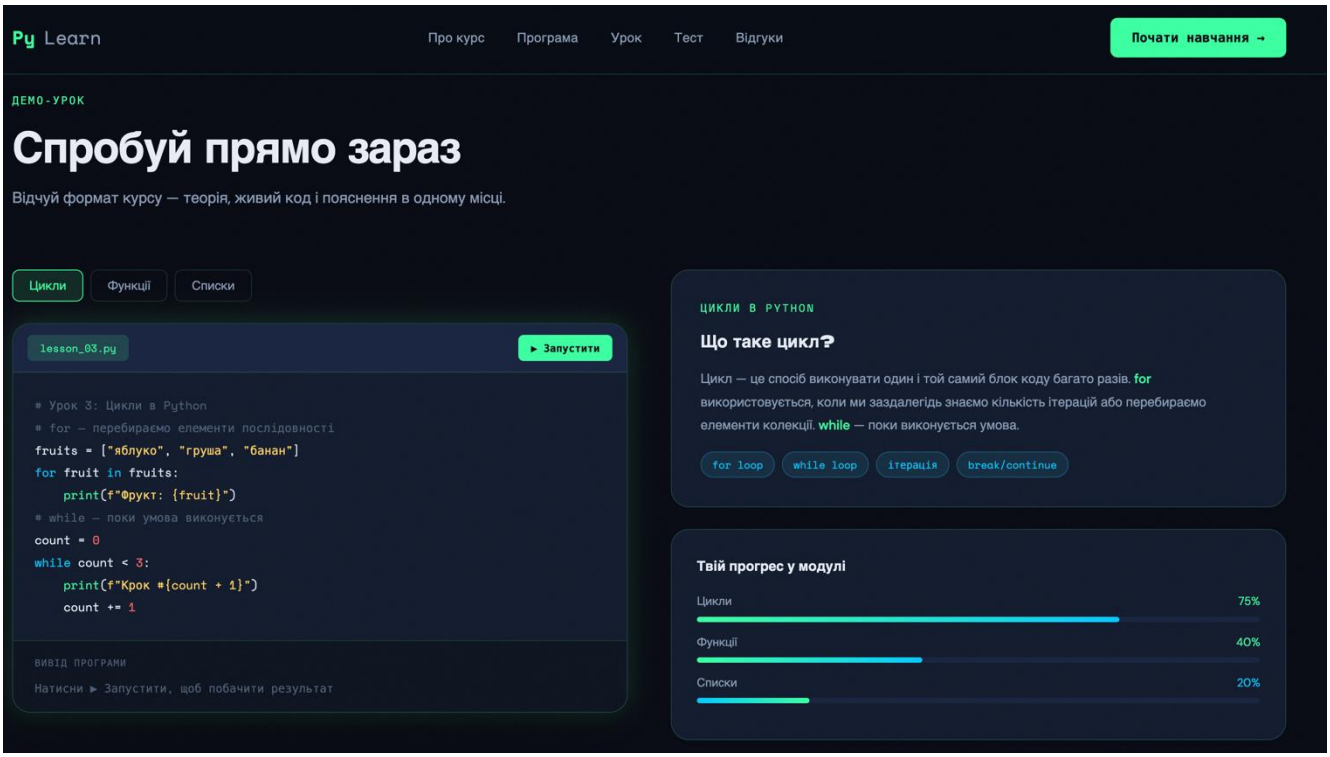

Рисунок 4.6 – Сторінка «Робота у компіляторі онлайн»

Проведене комплексне тестування підтвердило, що розроблена навчальна вебплатформа відповідає всім заявленим вимогам та демонструє високий рівень надійності. Система є функціонально повною: реалізовано адаптивний інтерфейс, інтегрований редактор коду з підтримкою автодоповнення та підсвічування синтаксису, а також механізми перевірки завдань у захищеному середовищі. Особливу увагу приділено безпеці - авторизація користувачів та ізоляція виконання коду в контейнерах гарантують захист даних і стабільність роботи.

Тестові навантаження показали, що платформа стійка до одночасної роботи великої кількості користувачів, а продуктивність залишається стабільною навіть за інтенсивних сценаріїв використання.

Програмний продукт повністю відповідає технічному завданню, що було визначене на етапі проєктування, і готовий до впровадження у навчальний процес закладів вищої освіти. Його використання у курсі дозволить студентам отримати практичні навички роботи з мовою, а викладачам - ефективно організувати контроль та оцінювання результатів.

ВИСНОВКИ

У результаті виконання дипломної роботи було проведено аналіз та реалізація проєкту, сформовано сучасний технологічний стек, що забезпечує повну відповідність функціональним та нефункціональним вимогам навчальної вебплатформи.

На рівні Backend використано мову програмування Python, яка гарантує гнучкість та надійність серверної частини. Frontend реалізовано за допомогою бібліотеки React (JavaScript/TypeScript) із інтегрованим редактором Monaco Editor, що забезпечує комфортну роботу з кодом та підтримку сучасних інструментів розробки.

Для зберігання даних застосовано PostgreSQL як потужну систему керування базами даних, а інфраструктура побудована на основі Docker, що гарантує ізоляцію середовищ, безпеку та простоту розгортання.

Обрані інструменти відповідають сучасним інженерним стандартам веброзробки та дозволяють у повній мірі реалізувати поставлені завдання. Проведене тестування підтвердило, що програмний продукт є функціонально повним, безпечним та стійким до навантажень. Він повністю відповідає технічному завданню, визначеному на етапі проєктування, і готовий до впровадження у навчальний процес закладів вищої освіти.

Використання платформи у курсі «Основи програмування на Python» сприятиме формуванню практичних навичок програмування у студентів, а також забезпечить викладачам ефективні інструменти для організації контролю та оцінювання результатів навчання. Таким чином, розроблений програмний продукт має всі передумови для успішної інтеграції у навчальний процес та подальшого розвитку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кухаренко В. М. Електронне навчання у вищій освіті: тенденції та перспективи розвитку. – Харків: ХНУ ім. В. Н. Каразіна, 2023. – 212 с.

2. Moodle Docs. Moodle 4.3 User Guide. URL: https://docs.moodle.org/43/ (дата звернення: 28.03.2026).

3. Коваленко О. С. Цифрові освітні платформи: Coursera як інструмент самоосвіти. – Київ: КНЕУ, 2023. – С. 57–64.

4. Codecademy. Learning with Interactive Coding Environments. – New York: Codecademy Press, 2023. – 72 p.

5. Sommerville I. Software Engineering, 11th Edition. – Pearson, 2023. – С. 145–162.

6. Brown T. Building Modern APIs with FastAPI. – O'Reilly Media, 2023. – С. 33–78.

7. Django Project. Official Documentation 2024. URL: https://docs.djangoproject.com/en/4.2 (дата звернення: 08.04.2026).

8. Stroustrup B. Programming: Principles and Practice Using C++ (3rd ed.). – Addison-Wesley, 2023. – С. 25–90.

9. Bloch J. Effective Java, 4th Edition. – Addison-Wesley, 2024. – С. 10–65.

10. Tilkov S. Node.js Design Patterns, 4th Edition. – Packt Publishing, 2023. – С. 40–110.

11. Van Rossum G., Warsaw B. Python 3.12 Developer's Guide. – Python Software Foundation. URL: https://docs.python.org/3.12 (дата звернення: 25.03.2026).

12. Google Developers. Angular 17 Documentation. URL: https://angular.dev (дата звернення: 17.04.2026).

13. You E. Vue.js 3 Guide. URL: https://vuejs.org/guide (дата звернення: 08.05.2026).

14. Facebook Open Source. React 18 Documentation. URL: https://react.dev (дата звернення: 05.04.2026).

15. PostgreSQL Global Development Group. PostgreSQL 16 Documentation. URL : https://www.postgresql.org/docs/16 (дата звернення: 19.04.2026).

16. Database Systems: Concepts, Design and Applications. – Springer, 2023. – С. 210–245.

17. Docker Inc. Docker Engine Documentation 2024. URL: https://docs.docker.com/engine (дата звернення: 07.04.2026).

18. Fowler M. UML Distilled, 5th Edition. – Addison-Wesley, 2023. – С. 88–97.

19. Intel Corp. Hardware Requirements for Cloud Systems 2024. URL: https://intel.com/cloud-hardware (дата звернення: 28.03.2026).

20. Fielding R. Architectural Styles and the Design of Network-based Software Architectures. – University of California, 2023. – С. 45–60.

21. Coronel C., Morris S. Database Systems: Design, Implementation, and Management, 14th ed. – Cengage, 2024. – С. 120–155.

22. Booch G. Object-Oriented Analysis and Design with Applications, 4th ed. – Addison-Wesley, 2023. – С. 75–130.

23. Microsoft Docs. Monaco Editor API Reference. URL: https://microsoft.github.io/monaco-editor (дата звернення: 02.05.2026).

24. Google Cloud. Secure Sandbox Environments Overview. URL: https://cloud.google.com/security/sandbox (дата звернення: 27.04.2026).

25. Creatio Academy. E-learning courses/ Sandbox. URL: https://academy.terrasoft.ua/docs/7-16/developer/front_end_razrabotka/sandbox/ (дата звернення: 28.04.2026).

ДОДАТОК А
КОД ПРОГРАМНОГО ПРОДУКТУ

Лістинг А.1 – Код програмного продукту

```
<!DOCTYPE html>
<html lang=«uk»>
<head>
<meta charset=«UTF-8»>
<meta name=«viewport» content=«width=device-width, initial-
scale=1.0»>
<title>PyLearn - Основи програмування на Python</title>
<link
href=«https://fonts.googleapis.com/css2?family=Space+Mono:ital,wght@
0,400;0,700;1,400&family=Syne:wght@400;600;700;800&family=DM+Sans:wg
ht@300;400;500&display=swap» rel=«stylesheet»>
<style>
  :root {
    --bg: #080c14;
    --bg2: #0d1321;
    --bg3: #111827;
    --surface: #141c2e;
    --surface2: #1a2540;
    --accent: #3dffa0;
    --accent2: #00c8ff;
    --accent3: #ff6b6b;
    --accent4: #ffd166;
    --text: #e8edf5;
    --text2: #8a9bb5;
    --text3: #4d5f78;
    --border: rgba(61,255,160,0.15);
    --glow: 0 0 30px rgba(61,255,160,0.2);
    --glow2: 0 0 30px rgba(0,200,255,0.2);
    --radius: 12px;
    --radius2: 20px;
  }

  *, *::before, *::after { box-sizing: border-box; margin: 0;
padding: 0; }

  html { scroll-behavior: smooth; }

  body {
    background: var(--bg);
    color: var(--text);
    font-family: 'DM Sans', sans-serif;
    font-size: 16px;
    line-height: 1.65;
    overflow-x: hidden;
```

```
  }

  /* ── NOISE OVERLAY ── */
  body::before {
    content: '';
    position: fixed; inset: 0;
    background-image: url(«data:image/svg+xml,%3Csvg viewBox='0 0
256 256' xmlns='http://www.w3.org/2000/svg'%3E%3Cfilter
id='noise'%3E%3CfeTurbulence type='fractalNoise' baseFrequency='0.9'
numOctaves='4' stitchTiles='stitch'/%3E%3C/filter%3E%3Crect
width='100%25' height='100%25' filter='url(%23noise)'
opacity='0.04'/%3E%3C/svg%3E»);
    pointer-events: none; z-index: 9999; opacity: .4;
  }

  /* ── NAV ── */
  nav {
    position: fixed; top: 0; left: 0; right: 0; z-index: 1000;
    display: flex; align-items: center; justify-content: space-
between;
    padding: 18px 48px;
    background: rgba(8,12,20,0.85);
    backdrop-filter: blur(20px);
    border-bottom: 1px solid var(--border);
  }

  .nav-logo {
    font-family: 'Space Mono', monospace;
    font-size: 1.3rem; font-weight: 700;
    color: var(--accent);
    text-decoration: none;
    display: flex; align-items: center; gap: 10px;
  }

  .nav-logo span { color: var(--text2); font-weight: 400; }

  .nav-links {
    display: flex; gap: 36px; list-style: none;
  }

  .nav-links a {
    color: var(--text2); text-decoration: none;
    font-size: .9rem; font-weight: 500;
    transition: color .2s;
    letter-spacing: .02em;
  }

  .nav-links a:hover { color: var(--accent); }

  .nav-cta {
    background: var(--accent); color: #080c14;
    padding: 10px 24px; border-radius: 8px;
    font-weight: 700; font-size: .9rem;
```

```
    text-decoration: none; transition: opacity .2s, transform .15s;
    font-family: 'Space Mono', monospace;
  }

  .nav-cta:hover { opacity: .9; transform: translateY(-1px); }

  /* ── HERO ── */
  .hero {
    min-height: 100vh;
    display: flex; flex-direction: column;
    justify-content: center; align-items: flex-start;
    padding: 120px 48px 80px;
    position: relative; overflow: hidden;
  }

  .hero-grid {
    position: absolute; inset: 0;
    background-image:
      linear-gradient(rgba(61,255,160,.04) 1px, transparent 1px),
      linear-gradient(90deg, rgba(61,255,160,.04) 1px, transparent
1px);
    background-size: 60px 60px;
    mask-image: radial-gradient(ellipse 80% 80% at 50% 50%, black,
transparent);
  }

  .hero-orb {
    position: absolute; border-radius: 50%;
    filter: blur(100px); pointer-events: none;
  }

  .orb1 {
    width: 600px; height: 600px;
    background: radial-gradient(circle, rgba(61,255,160,.12) 0%,
transparent 70%);
    top: -100px; right: -100px;
    animation: orb-pulse 8s ease-in-out infinite;
  }

  .orb2 {
    width: 400px; height: 400px;
    background: radial-gradient(circle, rgba(0,200,255,.08) 0%,
transparent 70%);
    bottom: 0; left: 20%;
    animation: orb-pulse 12s ease-in-out infinite reverse;
  }

  @keyframes orb-pulse {
    0%, 100% { transform: scale(1) translate(0,0); opacity: 1; }
    50% { transform: scale(1.1) translate(20px,-20px); opacity: .7;
}
  }
```

```
  .hero-badge {
    display: inline-flex; align-items: center; gap: 8px;
    background: rgba(61,255,160,.1);
    border: 1px solid rgba(61,255,160,.3);
    padding: 6px 16px; border-radius: 100px;
    font-family: 'Space Mono', monospace;
    font-size: .78rem; color: var(--accent);
    margin-bottom: 32px;
    animation: fade-up .6s ease both;
  }

  .hero-badge::before {
    content: '';
    width: 6px; height: 6px; border-radius: 50%;
    background: var(--accent);
    animation: blink 1.5s ease-in-out infinite;
  }

  @keyframes blink { 0%,100%{opacity:1} 50%{opacity:.2} }

  .hero h1 {
    font-family: 'Syne', sans-serif;
    font-size: clamp(2.8rem, 6vw, 5rem);
    font-weight: 800; line-height: 1.08;
    letter-spacing: -.03em;
    max-width: 760px;
    animation: fade-up .6s .1s ease both;
  }

  .hero h1 em {
    font-style: normal;
    background: linear-gradient(135deg, var(--accent), var(--
accent2));
    -webkit-background-clip: text; -webkit-text-fill-color:
transparent;
    background-clip: text;
  }

  .hero-sub {
    margin-top: 24px;
    font-size: 1.15rem; color: var(--text2);
    max-width: 540px; line-height: 1.75;
    animation: fade-up .6s .2s ease both;
  }

  .hero-actions {
    display: flex; gap: 16px; align-items: center;
    margin-top: 40px;
    animation: fade-up .6s .3s ease both;
  }

  .btn-primary {
    background: var(--accent); color: #080c14;
```

```
    padding: 14px 32px; border-radius: 10px;
    font-weight: 700; font-size: 1rem;
    text-decoration: none; transition: all .2s;
    font-family: 'Space Mono', monospace;
    display: inline-flex; align-items: center; gap: 10px;
    box-shadow: 0 0 40px rgba(61,255,160,.25);
  }

  .btn-primary:hover { opacity: .9; transform: translateY(-2px);
box-shadow: 0 0 60px rgba(61,255,160,.35); }

  .btn-secondary {
    border: 1px solid var(--border);
    color: var(--text); padding: 14px 28px;
    border-radius: 10px; text-decoration: none;
    font-size: 1rem; transition: all .2s;
    display: inline-flex; align-items: center; gap: 10px;
  }

  .btn-secondary:hover { border-color: var(--accent); color: var(--
accent); }

  .hero-stats {
    display: flex; gap: 48px; margin-top: 64px;
    animation: fade-up .6s .4s ease both;
  }

  .stat-item { display: flex; flex-direction: column; gap: 4px; }

  .stat-num {
    font-family: 'Space Mono', monospace;
    font-size: 2rem; font-weight: 700; color: var(--accent);
  }

  .stat-label { font-size: .85rem; color: var(--text3); }

  /* ── TERMINAL WINDOW ── */
  .hero-terminal {
    position: absolute; right: 48px; top: 50%;
    transform: translateY(-50%);
    width: 420px;
    background: var(--surface);
    border: 1px solid var(--border);
    border-radius: var(--radius2);
    overflow: hidden;
    box-shadow: 0 40px 80px rgba(0,0,0,.5), var(--glow);
    animation: fade-left .8s .2s ease both;
  }

  @keyframes fade-left {
    from { opacity: 0; transform: translateY(-50%) translateX(40px);
}
    to { opacity: 1; transform: translateY(-50%) translateX(0); }
```

```
  }

  .terminal-header {
    background: var(--surface2);
    padding: 12px 16px;
    display: flex; align-items: center; gap: 8px;
    border-bottom: 1px solid var(--border);
  }

  .terminal-dot {
    width: 12px; height: 12px; border-radius: 50%;
  }

  .dot-red { background: #ff5f57; }
  .dot-yellow { background: #ffbd2e; }
  .dot-green { background: #28ca41; }

  .terminal-title {
    font-family: 'Space Mono', monospace;
    font-size: .75rem; color: var(--text3);
    margin-left: 8px;
  }

  .terminal-body {
    padding: 24px;
    font-family: 'Space Mono', monospace;
    font-size: .82rem; line-height: 2;
  }

  .t-comment { color: var(--text3); }
  .t-keyword { color: var(--accent2); }
  .t-string { color: var(--accent4); }
  .t-func { color: var(--accent); }
  .t-num { color: var(--accent3); }
  .t-output { color: var(--text2); margin-top: 8px; }
  .t-cursor {
    display: inline-block; width: 8px; height: 14px;
    background: var(--accent); margin-left: 2px; vertical-align:
middle;
    animation: blink 1s step-end infinite;
  }

  /* ── SECTIONS ── */
  section { padding: 100px 48px; }

  .section-label {
    font-family: 'Space Mono', monospace;
    font-size: .75rem; letter-spacing: .15em;
    color: var(--accent); text-transform: uppercase;
    margin-bottom: 16px;
  }

  .section-title {
```

```
    font-family: 'Syne', sans-serif;
    font-size: clamp(2rem, 4vw, 3rem);
    font-weight: 800; line-height: 1.15;
    letter-spacing: -.025em;
  }

  .section-sub {
    color: var(--text2); font-size: 1.05rem;
    max-width: 560px; margin-top: 12px; line-height: 1.8;
  }

  /* ── FEATURES ── */
  .features-grid {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 2px;
    margin-top: 64px;
    border: 1px solid var(--border);
    border-radius: var(--radius2);
    overflow: hidden;
  }

  .feature-card {
    background: var(--surface);
    padding: 40px 36px;
    transition: background .2s;
    position: relative;
  }

  .feature-card::after {
    content: '';
    position: absolute; inset: 0;
    background: linear-gradient(135deg, rgba(61,255,160,.05) 0%,
transparent 60%);
    opacity: 0; transition: opacity .3s;
  }

  .feature-card:hover { background: var(--surface2); }
  .feature-card:hover::after { opacity: 1; }

  .feature-icon {
    width: 48px; height: 48px;
    background: rgba(61,255,160,.1);
    border: 1px solid rgba(61,255,160,.2);
    border-radius: 12px;
    display: flex; align-items: center; justify-content: center;
    font-size: 1.4rem;
    margin-bottom: 20px;
  }

  .feature-card h3 {
    font-family: 'Syne', sans-serif;
    font-size: 1.15rem; font-weight: 700;
```

```
    margin-bottom: 10px;
  }

  .feature-card p { color: var(--text2); font-size: .9rem; line-
height: 1.7; }

  /* ── CURRICULUM ── */
  .curriculum { background: var(--bg2); }

  .curriculum-layout {
    display: grid; grid-template-columns: 1fr 1fr;
    gap: 48px; margin-top: 64px; align-items: start;
  }

  .module-list { display: flex; flex-direction: column; gap: 12px; }

  .module-item {
    background: var(--surface);
    border: 1px solid transparent;
    border-radius: var(--radius);
    padding: 20px 24px;
    cursor: pointer; transition: all .25s;
    display: flex; align-items: center; gap: 16px;
  }

  .module-item:hover, .module-item.active {
    border-color: var(--accent);
    background: var(--surface2);
    box-shadow: var(--glow);
  }

  .module-num {
    font-family: 'Space Mono', monospace;
    font-size: .75rem; color: var(--accent);
    min-width: 28px;
  }

  .module-info { flex: 1; }

  .module-title {
    font-weight: 600; font-size: .95rem;
    margin-bottom: 2px;
  }

  .module-meta { font-size: .8rem; color: var(--text3); }

  .module-arrow {
    color: var(--text3); font-size: .9rem;
    transition: transform .2s, color .2s;
  }

  .module-item:hover .module-arrow,
  .module-item.active .module-arrow {
```

```
    transform: translateX(4px); color: var(--accent);
  }

  .module-detail {
    background: var(--surface);
    border: 1px solid var(--border);
    border-radius: var(--radius2);
    padding: 36px;
    position: sticky; top: 100px;
  }

  .module-detail-badge {
    font-family: 'Space Mono', monospace;
    font-size: .72rem; color: var(--accent);
    background: rgba(61,255,160,.1);
    border: 1px solid rgba(61,255,160,.2);
    padding: 4px 12px; border-radius: 100px;
    display: inline-block; margin-bottom: 16px;
  }

  .module-detail h3 {
    font-family: 'Syne', sans-serif;
    font-size: 1.4rem; font-weight: 800;
    margin-bottom: 12px;
  }

  .module-detail p { color: var(--text2); font-size: .92rem; line-
height: 1.75; }

  .lesson-list {
    margin-top: 24px;
    list-style: none;
    display: flex; flex-direction: column; gap: 8px;
  }

  .lesson-list li {
    display: flex; align-items: center; gap: 12px;
    font-size: .88rem; color: var(--text2);
    padding: 10px 0;
    border-bottom: 1px solid rgba(255,255,255,.04);
  }

  .lesson-check {
    width: 20px; height: 20px; border-radius: 50%;
    background: rgba(61,255,160,.1);
    border: 1px solid rgba(61,255,160,.3);
    display: flex; align-items: center; justify-content: center;
    font-size: .7rem; color: var(--accent); flex-shrink: 0;
  }

  .lesson-duration {
    margin-left: auto;
    font-family: 'Space Mono', monospace;
```

```
    font-size: .72rem; color: var(--text3);
  }

  /* ── LESSON PREVIEW ── */
  .lesson-section { background: var(--bg); }

  .lesson-preview-wrap {
    display: grid; grid-template-columns: 1fr 1fr;
    gap: 48px; margin-top: 64px;
  }

  .code-window {
    background: var(--surface);
    border: 1px solid var(--border);
    border-radius: var(--radius2);
    overflow: hidden;
    box-shadow: 0 20px 60px rgba(0,0,0,.4), var(--glow);
  }

  .code-header {
    background: var(--surface2);
    padding: 12px 16px;
    display: flex; align-items: center; justify-content: space-
between;
    border-bottom: 1px solid var(--border);
  }

  .code-tabs {
    display: flex; gap: 2px;
  }

  .code-tab {
    padding: 5px 14px; border-radius: 6px;
    font-family: 'Space Mono', monospace; font-size: .72rem;
    cursor: pointer; transition: background .2s;
    color: var(--text3);
  }

  .code-tab.active {
    background: rgba(61,255,160,.15);
    color: var(--accent);
  }

  .code-run {
    background: var(--accent); color: #080c14;
    border: none; padding: 6px 14px;
    border-radius: 6px; cursor: pointer;
    font-family: 'Space Mono', monospace; font-size: .72rem;
    font-weight: 700; display: flex; align-items: center; gap: 6px;
    transition: all .2s;
  }

  .code-run:hover { opacity: .85; }
```

```
.code-body {
  padding: 24px;
  font-family: 'Space Mono', monospace;
  font-size: .82rem; line-height: 1.9;
  min-height: 260px;
  overflow-x: auto;
}

.code-output {
  background: rgba(0,0,0,.3);
  border-top: 1px solid var(--border);
  padding: 16px 24px;
  font-family: 'Space Mono', monospace;
  font-size: .8rem; color: var(--text2);
  min-height: 60px;
}

.output-label {
  font-size: .65rem; color: var(--text3);
  letter-spacing: .1em; text-transform: uppercase;
  margin-bottom: 8px;
}

.output-line { color: var(--accent); margin-bottom: 2px; }

.lesson-info { display: flex; flex-direction: column; gap: 24px; }

.lesson-theory-card {
  background: var(--surface);
  border: 1px solid var(--border);
  border-radius: var(--radius2);
  padding: 32px;
}

.lesson-theory-card h3 {
  font-family: 'Syne', sans-serif;
  font-size: 1.2rem; font-weight: 800;
  margin-bottom: 12px;
}

.lesson-theory-card p {
  color: var(--text2); font-size: .9rem; line-height: 1.8;
}

.concept-tags {
  display: flex; flex-wrap: wrap; gap: 8px; margin-top: 16px;
}

.concept-tag {
  font-family: 'Space Mono', monospace; font-size: .72rem;
  padding: 4px 12px; border-radius: 100px;
  background: rgba(0,200,255,.1);
```

```
    border: 1px solid rgba(0,200,255,.25);
    color: var(--accent2);
  }

  .progress-card {
    background: var(--surface);
    border: 1px solid var(--border);
    border-radius: var(--radius2);
    padding: 28px;
  }

  .progress-card h4 {
    font-weight: 600; margin-bottom: 16px; font-size: .95rem;
  }

  .progress-bar-wrap { margin-bottom: 12px; }
  .progress-bar-label {
    display: flex; justify-content: space-between;
    font-size: .8rem; color: var(--text2); margin-bottom: 6px;
  }

  .progress-bar {
    height: 6px; background: var(--surface2);
    border-radius: 100px; overflow: hidden;
  }

  .progress-fill {
    height: 100%; border-radius: 100px;
    background: linear-gradient(90deg, var(--accent), var(--
accent2));
    transition: width 1s ease;
  }

  /* ── QUIZ ── */
  .quiz-section { background: var(--bg2); }

  .quiz-card {
    background: var(--surface);
    border: 1px solid var(--border);
    border-radius: var(--radius2);
    padding: 40px;
    max-width: 700px; margin: 64px auto 0;
  }

  .quiz-step {
    font-family: 'Space Mono', monospace;
    font-size: .72rem; color: var(--text3);
    margin-bottom: 24px;
  }

  .quiz-question {
    font-family: 'Syne', sans-serif;
    font-size: 1.25rem; font-weight: 700;
```

```
    margin-bottom: 28px; line-height: 1.4;
  }

  .quiz-options {
    display: flex; flex-direction: column; gap: 12px;
  }

  .quiz-option {
    display: flex; align-items: center; gap: 14px;
    background: var(--bg3); border: 1px solid var(--surface2);
    border-radius: 10px; padding: 14px 18px;
    cursor: pointer; transition: all .2s;
    font-size: .92rem;
  }

  .quiz-option:hover { border-color: var(--accent); background:
var(--surface2); }

  .quiz-option.correct { border-color: var(--accent); background:
rgba(61,255,160,.08); }
  .quiz-option.incorrect { border-color: var(--accent3); background:
rgba(255,107,107,.08); }

  .option-key {
    font-family: 'Space Mono', monospace; font-size: .75rem;
    width: 28px; height: 28px; border-radius: 6px;
    background: var(--surface2); display: flex;
    align-items: center; justify-content: center;
    color: var(--text2); flex-shrink: 0;
    transition: all .2s;
  }

  .quiz-option:hover .option-key { background: var(--accent); color:
#080c14; }
  .quiz-option.correct .option-key { background: var(--accent);
color: #080c14; }
  .quiz-option.incorrect .option-key { background: var(--accent3);
color: #080c14; }

  .quiz-nav {
    display: flex; justify-content: space-between;
    align-items: center; margin-top: 32px;
  }

  .quiz-result {
    display: none; text-align: center; padding: 20px 0;
  }

  .quiz-result.show { display: block; }

  .quiz-result h3 {
    font-family: 'Syne', sans-serif;
    font-size: 1.5rem; font-weight: 800;
```

```
    color: var(--accent); margin-bottom: 8px;
  }

  /* ── TESTIMONIALS ── */
  .testimonials-grid {
    display: grid; grid-template-columns: repeat(3, 1fr);
    gap: 24px; margin-top: 64px;
  }

  .testimonial-card {
    background: var(--surface);
    border: 1px solid var(--border);
    border-radius: var(--radius2);
    padding: 28px;
    transition: transform .2s, border-color .2s;
  }

  .testimonial-card:hover { transform: translateY(-4px); border-
color: rgba(61,255,160,.3); }

  .stars { color: var(--accent4); font-size: .9rem; margin-bottom:
16px; }

  .testimonial-text {
    color: var(--text2); font-size: .92rem; line-height: 1.75;
    margin-bottom: 20px; font-style: italic;
  }

  .testimonial-author {
    display: flex; align-items: center; gap: 12px;
  }

  .author-avatar {
    width: 40px; height: 40px; border-radius: 50%;
    background: linear-gradient(135deg, var(--accent), var(--
accent2));
    display: flex; align-items: center; justify-content: center;
    font-weight: 700; font-size: .85rem; color: #080c14;
    flex-shrink: 0;
  }

  .author-name { font-weight: 600; font-size: .9rem; }
  .author-role { font-size: .78rem; color: var(--text3); }

  /* ── CTA ── */
  .cta-section {
    text-align: center;
    background: radial-gradient(ellipse 80% 60% at 50% 50%,
rgba(61,255,160,.06) 0%, transparent 70%),
                var(--bg);
    border-top: 1px solid var(--border);
    border-bottom: 1px solid var(--border);
  }
```

```
  .cta-section h2 {
    font-family: 'Syne', sans-serif;
    font-size: clamp(2rem, 4vw, 3.2rem);
    font-weight: 800; letter-spacing: -.025em;
    max-width: 640px; margin: 0 auto 24px;
  }

  .cta-section p { color: var(--text2); font-size: 1.05rem; margin-
bottom: 40px; }

  .pricing-card {
    background: var(--surface);
    border: 1px solid var(--border);
    border-radius: var(--radius2);
    padding: 40px; max-width: 380px;
    margin: 0 auto;
    box-shadow: var(--glow);
  }

  .price-label { font-family: 'Space Mono', monospace; font-size:
.75rem; color: var(--text3); }

  .price-amount {
    font-family: 'Syne', sans-serif;
    font-size: 3rem; font-weight: 800;
    color: var(--accent); margin: 8px 0;
  }

  .price-period { font-size: .85rem; color: var(--text2); }

  .price-features {
    list-style: none; margin: 24px 0;
    display: flex; flex-direction: column; gap: 10px;
  }

  .price-features li {
    display: flex; align-items: center; gap: 10px;
    font-size: .88rem; color: var(--text2);
  }

  .price-features li::before {
    content: '✓'; color: var(--accent);
    font-family: 'Space Mono', monospace; font-size: .85rem;
  }

  /* ── FOOTER ── */
  footer {
    padding: 48px;
    border-top: 1px solid var(--border);
    display: flex; justify-content: space-between; align-items:
center;
  }
```

```
  .footer-brand {
    font-family: 'Space Mono', monospace;
    color: var(--accent); font-size: 1.1rem;
  }

  .footer-copy { color: var(--text3); font-size: .82rem; }

  .footer-links { display: flex; gap: 24px; }
  .footer-links a { color: var(--text2); text-decoration: none;
font-size: .85rem; transition: color .2s; }
  .footer-links a:hover { color: var(--accent); }

  /* ── ANIMATIONS ── */
  @keyframes fade-up {
    from { opacity: 0; transform: translateY(20px); }
    to { opacity: 1; transform: translateY(0); }
  }

  .reveal {
    opacity: 0; transform: translateY(28px);
    transition: opacity .7s ease, transform .7s ease;
  }

  .reveal.visible { opacity: 1; transform: translateY(0); }

  /* ── MOBILE ── */
  @media (max-width: 900px) {
    nav { padding: 16px 24px; }
    .nav-links { display: none; }
    .hero { padding: 100px 24px 60px; }
    .hero-terminal { display: none; }
    .hero h1 { font-size: 2.4rem; }
    .hero-stats { gap: 28px; }
    section { padding: 72px 24px; }
    .features-grid { grid-template-columns: 1fr; gap: 1px; }
    .curriculum-layout { grid-template-columns: 1fr; }
    .lesson-preview-wrap { grid-template-columns: 1fr; }
    .testimonials-grid { grid-template-columns: 1fr; }
    footer { flex-direction: column; gap: 20px; text-align: center;
}
  }

  /* ── TABS for LESSON ── */
  .lesson-tabs { display: flex; gap: 8px; margin-bottom: 24px; }

  .lesson-tab-btn {
    padding: 8px 18px; border-radius: 8px; border: 1px solid var(--
border);
    background: transparent; color: var(--text2);
    cursor: pointer; font-size: .85rem; font-family: 'DM Sans',
sans-serif;
    transition: all .2s;
```

```
  }

  .lesson-tab-btn.active { background: rgba(61,255,160,.12); border-
color: var(--accent); color: var(--accent); }
</style>
</head>
<body>

<!-- ── NAV ── -->
<nav>
  <a href=«#» class=«nav-logo»>Py<span>Learn</span></a>
  <ul class=«nav-links»>
    <li><a href=«#about»>Про курс</a></li>
    <li><a href=«#curriculum»>Програма</a></li>
    <li><a href=«#lesson»>Урок</a></li>
    <li><a href=«#quiz»>Тест</a></li>
    <li><a href=«#reviews»>Відгуки</a></li>
  </ul>
  <a href=«#cta» class=«nav-cta»>Почати навчання →</a>
</nav>

<!-- ── HERO ── -->
<section class=«hero» id=«home»>
  <div class=«hero-grid»></div>
  <div class=«hero-orb orb1»></div>
  <div class=«hero-orb orb2»></div>

  <div class=«hero-badge»>✦ Курс 2026 - Оновлено</div>
  <h1>Основи<br>програмування<br>на <em>Python</em></h1>
  <p class=«hero-sub»>
    Від нуля до впевненого розробника. Практичний курс із живими
прикладами,
    інтерактивними завданнями та реальними проєктами.
  </p>
  <div class=«hero-actions»>
    <a href=«#curriculum» class=«btn-primary»>▶ Переглянути
програму</a>
    <a href=«#lesson» class=«btn-secondary»>□ Демо-урок</a>
  </div>
  <div class=«hero-stats»>
    <div class=«stat-item»>
      <span class=«stat-num»>120+</span>
      <span class=«stat-label»>Відеоуроків</span>
    </div>
    <div class=«stat-item»>
      <span class=«stat-num»>12</span>
      <span class=«stat-label»>Модулів</span>
    </div>
    <div class=«stat-item»>
      <span class=«stat-num»>2 400</span>
      <span class=«stat-label»>Студентів</span>
    </div>
    <div class=«stat-item»>
```

```
      <span class=«stat-num»>4.9★</span>
      <span class=«stat-label»>Рейтинг</span>
    </div>
  </div>

  <!-- Terminal Window -->
  <div class=«hero-terminal»>
    <div class=«terminal-header»>
      <div class=«terminal-dot dot-red»></div>
      <div class=«terminal-dot dot-yellow»></div>
      <div class=«terminal-dot dot-green»></div>
      <span class=«terminal-title»>hello_world.py</span>
    </div>
    <div class=«terminal-body»>
      <div class=«t-comment»># Твоя перша програма на Python</div>
      <div><span class=«t-keyword»>def</span> <span class=«t-
func»>greet</span>(name):</div>
      <div>    <span class=«t-
keyword»>return</span> <span class=«t-string»>f»Привіт, {name}!
»</span></div>
      <div></div>
      <div>students = [<span class=«t-string»>«Аня»</span>, <span
class=«t-string»>«Богдан»</span>, <span class=«t-
string»>«Оля»</span>]</div>
      <div></div>
      <div><span class=«t-keyword»>for</span> student <span
class=«t-keyword»>in</span> students:</div>
      <div>    <span class=«t-
func»>print</span>(greet(student))</div>
      <div class=«t-output»>▶ Привіт, Аня! </div>
      <div class=«t-output»>▶ Привіт, Богдан! </div>
      <div class=«t-output»>▶ Привіт, Оля! <span class=«t-
cursor»></span></div>
    </div>
  </div>
</section>

<!-- ── FEATURES ── -->
<section id=«about» style=«background: var(--bg2);»>
  <div class=«section-label»>Чому PyLearn</div>
  <h2 class=«section-title reveal»>Навчання, яке<br>справді
працює</h2>
  <p class=«section-sub reveal»>Курс побудований так, щоб ти вчився
на практиці - кожна концепція підкріплена завданням.</p>

  <div class=«features-grid reveal»>
    <div class=«feature-card»>
      <div class=«feature-icon»></div>
      <h3>Практика з першого дня</h3>
      <p>Жодної нудної теорії без коду. Кожен урок - це мінімум
теорії і максимум практики з живими прикладами.</p>
    </div>
```

```
    <div class=«feature-card»>
      <div class=«feature-icon»>□</div>
      <h3>Адаптивні тести</h3>
      <p>Система автоматично відстежує твій прогрес і пропонує
завдання відповідного рівня складності.</p>
    </div>
    <div class=«feature-card»>
      <div class=«feature-icon»>□</div>
      <h3>Вбудований редактор</h3>
      <p>Пишеш і запускаєш код прямо в браузері - жодних налаштувань
середовища на початку навчання.</p>
    </div>
    <div class=«feature-card»>
      <div class=«feature-icon»>□</div>
      <h3>Реальні проєкти</h3>
      <p>Кожен модуль завершується мінімальним проєктом, який можна
показати у портфоліо.</p>
    </div>
    <div class=«feature-card»>
      <div class=«feature-icon»>□</div>
      <h3>Детальна статистика</h3>
      <p>Бачиш свій прогрес: скільки часу витрачено, які теми
засвоєно, де потрібно повторити.</p>
    </div>
    <div class=«feature-card»>
      <div class=«feature-icon»>□</div>
      <h3>Спільнота</h3>
      <p>Форум, код-рев'ю від наставників і підтримка однокурсників
- ти навчаєшся не сам.</p>
    </div>
  </div>
</section>

<!-- ── CURRICULUM ── -->
<section class=«curriculum» id=«curriculum»>
  <div class=«section-label»>Програма курсу</div>
  <h2 class=«section-title reveal»>12 модулів -<br>від основ до
проєктів</h2>
  <p class=«section-sub reveal»>Структурована програма, яка проведе
тебе від першого рядка коду до написання повноцінних програм.</p>

  <div class=«curriculum-layout reveal»>
    <div class=«module-list» id=«moduleList»>
      <!-- filled by JS -->
    </div>
    <div class=«module-detail» id=«moduleDetail»>
      <!-- filled by JS -->
    </div>
  </div>
</section>

<!-- ── LESSON PREVIEW ── -->
<section class=«lesson-section» id=«lesson»>
```

```
  <div class=«section-label»>Демо-урок</div>
  <h2 class=«section-title reveal»>Спробуй прямо зараз</h2>
  <p class=«section-sub reveal»>Відчуй формат курсу - теорія, живий
код і пояснення в одному місці.</p>

  <div class=«lesson-preview-wrap reveal»>
    <div>
      <div class=«lesson-tabs»>
        <button class=«lesson-tab-btn active»
onclick=«switchTab('loops')»>Цикли</button>
        <button class=«lesson-tab-btn»
onclick=«switchTab('funcs')»>Функції</button>
        <button class=«lesson-tab-btn»
onclick=«switchTab('lists')»>Списки</button>
      </div>
      <div class=«code-window»>
        <div class=«code-header»>
          <div class=«code-tabs»>
            <span class=«code-tab active»
id=«tabName»>lesson_03.py</span>
          </div>
          <button class=«code-run» onclick=«runCode()»>▶
Запустити</button>
        </div>
        <div class=«code-body» id=«codeBody»>
          <div><span class=«t-comment»># Урок 3: Цикли в
Python</span></div>
          <div><span class=«t-comment»># for - перебираємо елементи
послідовності</span></div>
          <div></div>
          <div>fruits = [<span class=«t-string»>«яблуко»</span>,
<span class=«t-string»>«груша»</span>, <span class=«t-
string»>«банан»</span>]</div>
          <div></div>
          <div><span class=«t-keyword»>for</span> fruit <span
class=«t-keyword»>in</span> fruits:</div>
          <div>    <span class=«t-
func»>print</span>(<span class=«t-string»>f»Фрукт:
{fruit}»</span>)</div>
          <div></div>
          <div><span class=«t-comment»># while - поки умова
виконується</span></div>
          <div>count = <span class=«t-num»>0</span></div>
          <div><span class=«t-keyword»>while</span> count &lt; <span
class=«t-num»>3</span>:</div>
          <div>    <span class=«t-
func»>print</span>(<span class=«t-string»>f»Крок #{count +
1}»</span>)</div>
          <div>    count += <span class=«t-
num»>1</span></div>
        </div>
        <div class=«code-output»>
          <div class=«output-label»>Вивід програми</div>
```

```
          <div id=«codeOutput» style=«color: var(--text3); font-
size: .8rem;»>Натисни ▶ Запустити, щоб побачити результат</div>
        </div>
      </div>
    </div>

    <div class=«lesson-info»>
      <div class=«lesson-theory-card»>
        <div class=«section-label» style=«margin-bottom:12px»
id=«theoryTitle»>Цикли в Python</div>
        <h3 id=«theoryH3»>Що таке цикл?</h3>
        <p id=«theoryText»>
          Цикл - це спосіб виконувати один і той самий блок коду
багато разів.
          <strong style=«color:var(--accent)»>for</strong>
використовується, коли ми заздалегідь знаємо кількість ітерацій або
перебираємо елементи колекції.
          <strong style=«color:var(--accent)»>while</strong> - поки
виконується умова.
        </p>
        <div class=«concept-tags» id=«conceptTags»>
          <span class=«concept-tag»>for loop</span>
          <span class=«concept-tag»>while loop</span>
          <span class=«concept-tag»>ітерація</span>
          <span class=«concept-tag»>break/continue</span>
        </div>
      </div>

      <div class=«progress-card»>
        <h4>Твій прогрес у модулі</h4>
        <div class=«progress-bar-wrap»>
          <div class=«progress-bar-label»><span>Цикли</span><span
style=«color:var(--accent)»>75%</span></div>
          <div class=«progress-bar»><div class=«progress-fill»
style=«width:75%»></div></div>
        </div>
        <div class=«progress-bar-wrap»>
          <div class=«progress-bar-label»><span>Функції</span><span
style=«color:var(--accent)»>40%</span></div>
          <div class=«progress-bar»><div class=«progress-fill»
style=«width:40%»></div></div>
        </div>
        <div class=«progress-bar-wrap»>
          <div class=«progress-bar-label»><span>Списки</span><span
style=«color:var(--accent2)»>20%</span></div>
          <div class=«progress-bar»><div class=«progress-fill»
style=«width:20%; background: linear-gradient(90deg, var(--accent2),
var(--accent))»></div></div>
        </div>
      </div>
    </div>
  </div>
</section>
```

```
<!-- ── QUIZ ── -->
<section class=«quiz-section» id=«quiz»>
  <div class=«section-label»>Перевір себе</div>
  <h2 class=«section-title reveal» style=«text-align:center»>Міні-
тест</h2>
  <p class=«section-sub reveal» style=«text-align:center; margin: 0
auto;»>Відповідай на питання і побачиш, як добре засвоїв
матеріал.</p>

  <div class=«quiz-card reveal» id=«quizCard»>
    <div class=«quiz-step» id=«quizStep»>Питання 1 / 5</div>
    <div class=«quiz-question» id=«quizQ»>...</div>
    <div class=«quiz-options» id=«quizOptions»></div>
    <div class=«quiz-result» id=«quizResult»>
      <h3 id=«resultScore»></h3>
      <p id=«resultMsg» style=«color:var(--text2); margin-
bottom:20px»></p>
      <button class=«btn-primary» onclick=«resetQuiz()»
style=«margin: 0 auto; font-size:.9rem; padding: 12px 24px;»>↺ Пройти
знову</button>
    </div>
    <div class=«quiz-nav» id=«quizNav»>
      <span style=«color:var(--text3); font-size:.85rem»
id=«quizHint»>Оберіть відповідь</span>
      <button class=«btn-primary» id=«nextBtn»
onclick=«nextQuestion()» style=«opacity:.4; pointer-events:none;
font-size:.85rem; padding:10px 22px;»>Далі →</button>
    </div>
  </div>
</section>

<!-- ── TESTIMONIALS ── -->
<section id=«reviews» style=«background:var(--bg)»>
  <div class=«section-label»>Відгуки</div>
  <h2 class=«section-title reveal»>Що кажуть студенти</h2>

  <div class=«testimonials-grid reveal»>
    <div class=«testimonial-card»>
      <div class=«stars»>★★★★★</div>
      <p class=«testimonial-text»>«Найкращий Python-курс, який я
знаходив. Пояснення чіткі, завдання реальні. Через 2 місяці отримав
першу роботу як junior developer.»</p>
      <div class=«testimonial-author»>
        <div class=«author-avatar»>АК</div>
        <div>
          <div class=«author-name»>Андрій Коваленко</div>
          <div class=«author-role»>Junior Python Developer</div>
        </div>
      </div>
    </div>
    <div class=«testimonial-card»>
      <div class=«stars»>★★★★★</div>
```

```
      <p class=«testimonial-text»>«Дуже люблю формат - теорія, потім відразу практика прямо в браузері. Не треба нічого встановлювати, одразу пишеш код і бачиш результат.»</p>
      <div class=«testimonial-author»>
        <div class=«author-avatar»>МП</div>
        <div>
          <div class=«author-name»>Марія Поліщук</div>
          <div class=«author-role»>Студентка, 3 курс</div>
        </div>
      </div>
    </div>
    <div class=«testimonial-card»>
      <div class=«stars»>★★★★★</div>
      <p class=«testimonial-text»>«Пройшов 6 інших курсів до цього - всі нудні. Тут нарешті відчув, що Python це весело. Рекомендую всім, хто починає з нуля.»</p>
      <div class=«testimonial-author»>
        <div class=«author-avatar»>ОВ</div>
        <div>
          <div class=«author-name»>Олег Василенко</div>
          <div class=«author-role»>Фрілансер</div>
        </div>
      </div>
    </div>
  </div>
</section>

<!-- ── CTA ── -->
<section class=«cta-section» id=«cta»>
  <div class=«section-label» style=«text-align:center»>Почни сьогодні</div>
  <h2>Твій перший рядок коду - за 5 хвилин</h2>
  <p>Повний доступ до всіх матеріалів, проєктів і підтримки спільноти.</p>

  <div class=«pricing-card reveal»>
    <div class=«price-label»>Повний доступ</div>
    <div class=«price-amount»>Безкоштовно</div>
    <div class=«price-period»>Відкритий доступ на весь курс</div>
    <ul class=«price-features»>
      <li>120+ відеоуроків і конспектів</li>
      <li>Вбудований Python-редактор</li>
      <li>50+ практичних завдань</li>
      <li>5 реальних міні-проєктів</li>
      <li>Сертифікат після завершення</li>
      <li>Доступ до закритого форуму</li>
    </ul>
    <a href=«#lesson» class=«btn-primary» style=«width:100%; justify-content:center; font-size:.95rem;»>
      Розпочати навчання →
    </a>
  </div>
</section>
```

```
<!-- ── FOOTER ── -->
<footer>
  <div class=«footer-brand»>PyLearn </div>
  <div class=«footer-copy»>© 2026 PyLearn - Навчальна платформа
Python</div>
  <div class=«footer-links»>
    <a href=«#»>Умови</a>
    <a href=«#»>Конфіденційність</a>
    <a href=«#»>Контакти</a>
  </div>
</footer>

<script>
/* ── MODULE DATA ── */
const modules = [
  { num: '01', title: 'Вступ до Python', lessons: 8, duration: '3г
20хв', badge: 'Старт',
    desc: 'Знайомство з мовою програмування Python, встановлення
середовища і перша програма. Розуміємо, що таке інтерпретатор,
змінні і типи даних.',
    topics: ['Що таке Python і де використовується', 'Встановлення
Python і VS Code', 'Перша програма: print() і input()', 'Типи даних:
int, float, str, bool', 'Арифметичні оператори', 'Перетворення
типів', 'Коментарі в коді', 'Практика: Калькулятор'] },
  { num: '02', title: 'Умови та логіка', lessons: 6, duration: '2г
40хв', badge: 'Основи',
    desc: 'Вчимося приймати рішення в програмі: оператори
порівняння, логічні оператори та розгалуження if-elif-else.',
    topics: ['Оператори порівняння', 'Логічні оператори: and, or,
not', 'if / elif / else', 'Вкладені умови', 'Тернарний оператор',
'Практика: вгадай число'] },
  { num: '03', title: 'Цикли', lessons: 7, duration: '3г 00хв',
badge: 'Основи',
    desc: 'Автоматизація повторюваних дій: цикли for і while,
керування потоком break/continue/pass.',
    topics: ['Цикл for і range()', 'Цикл while', 'break і continue',
'Вкладені цикли', 'else у циклах', 'Практика: таблиця множення',
'Практика: пошук простих чисел'] },
  { num: '04', title: 'Функції', lessons: 9, duration: '4г 10хв',
badge: 'Середній',
    desc: 'Організація коду в функції: параметри, аргументи,
значення за замовчуванням, лямбда-функції та рекурсія.',
    topics: ['Оголошення функцій def', 'Параметри і аргументи',
'return і None', 'Значення за замовчуванням', 'args і kwargs',
'Лямбда-функції', 'Рекурсія', 'Область видимості', 'Практика:
обробка тексту'] },
  { num: '05', title: 'Списки і кортежі', lessons: 8, duration: '3г
30хв', badge: 'Середній',
    desc: 'Колекції даних: список (list) і незмінний кортеж (tuple),
зрізи, методи і list comprehension.',
```

```
    topics: ['Списки: створення і доступ', 'Методи списків',
'Зрізи', 'Сортування', 'Кортежі', 'List comprehension', 'Вкладені
списки', 'Практика: студентський журнал'] },
  { num: '06', title: 'Словники та множини', lessons: 6, duration:
'2г 50хв', badge: 'Середній',
    desc: 'Зберігання пар ключ-значення у словниках і унікальних
елементів у множинах.',
    topics: ['Словники: CRUD операції', 'Методи словників', 'Перебір
словника', 'Множини і операції над ними', 'Dict comprehension',
'Практика: телефонна книга'] },
  { num: '07', title: 'Рядки та текст', lessons: 7, duration: '3г
00хв', badge: 'Середній',
    desc: 'Робота з текстом: рядкові методи, форматування, регулярні
вирази (вступ).',
    topics: ['Рядкові методи', 'Форматування: f-strings', 'split() і
join()', 'Зрізи рядків', 'Пошук і заміна', 'Вступ до regex',
'Практика: обробник тексту'] },
  { num: '08', title: 'Файли та винятки', lessons: 7, duration: '3г
20хв', badge: 'Просунутий',
    desc: 'Читання і запис файлів, обробка помилок через try-except,
контекстні менеджери.',
    topics: ['Відкриття і закриття файлів', 'Читання: read,
readline, readlines', 'Запис у файл', 'Режими відкриття', 'try /
except / finally', 'raise і власні винятки', 'Практика: нотатник']
},
  { num: '09', title: 'ООП: Основи', lessons: 10, duration: '4г
30хв', badge: 'Просунутий',
    desc: 'Об\'єктно-орієнтоване програмування: класи, об\'єкти,
інкапсуляція, наслідування.',
    topics: ['Що таке клас і об\'єкт', '__init__ і self', 'Атрибути
та методи', 'Геттери і сеттери', 'Наслідування', 'super()',
'Поліморфізм', 'Магічні методи', 'Практика: бібліотека книг',
'Практика: банківський рахунок'] },
  { num: '10', title: 'Модулі та бібліотеки', lessons: 8, duration:
'3г 40хв', badge: 'Просунутий',
    desc: 'Структуризація великих проєктів, стандартна бібліотека
Python, встановлення зовнішніх пакетів через pip.',
    topics: ['import і from...import', 'Власні модулі', '__name__ ==
«__main__»', 'os і sys', 'datetime і time', 'random і math', 'pip і
virtual env', 'Практика: утиліти командного рядка'] },
  { num: '11', title: 'Робота з даними', lessons: 9, duration: '4г
00хв', badge: 'Проєкт',
    desc: 'Практична робота з JSON, CSV та основи роботи з pandas
для аналізу даних.',
    topics: ['JSON: читання і запис', 'CSV: обробка таблиць', 'Вступ
до pandas', 'DataFrame: базові операції', 'Фільтрація і сортування',
'Групування', 'Matplotlib: перші графіки', 'Практика: аналіз
продажів', 'Проєкт: звіт з даних'] },
  { num: '12', title: 'Фінальний проєкт', lessons: 5, duration: '5г
00хв', badge: 'Фінал',
    desc: 'Застосовуєш все вивчене: розробляєш повноцінний
консольний додаток з файловою базою даних, обробкою помилок та
ООП.',
```

```
    topics: ['Планування архітектури', 'Розробка моделей', 'Бізнес-
логіка', 'Інтерфейс командного рядка', 'Тестування та дебаг'] },
];

/* ── RENDER MODULES ── */
let activeModule = 0;

function renderModules() {
  const list = document.getElementById('moduleList');
  list.innerHTML = modules.map((m, i) => `
    <div class=«module-item ${i===activeModule?'active':''}»
onclick=«selectModule(${i})»>
      <span class=«module-num»>${m.num}</span>
      <div class=«module-info»>
        <div class=«module-title»>${m.title}</div>
        <div class=«module-meta»>${m.lessons} уроків ·
${m.duration}</div>
      </div>
      <span class=«module-arrow»>→</span>
    </div>
  `).join('');
  renderDetail();
}

function selectModule(i) {
  activeModule = i;
  renderModules();
}

function renderDetail() {
  const m = modules[activeModule];
  document.getElementById('moduleDetail').innerHTML = `
    <div class=«module-detail-badge»>Модуль ${m.num} ·
${m.badge}</div>
    <h3>${m.title}</h3>
    <p>${m.desc}</p>
    <ul class=«lesson-list»>
      ${m.topics.map((t,i) => `
        <li>
          <div class=«lesson-check»>${i<3?'✓':''}</div>
          ${t}
          <span class=«lesson-
duration»>${Math.floor(Math.random()*15+8)}хв</span>
        </li>
      `).join('')}
    </ul>
  `;
}

renderModules();

/* ── LESSON TABS ── */
const tabData = {
```

```
  loops: {
    file: 'lesson_03.py',
    code: `<div><span class=«t-comment»># Урок 3: Цикли в
Python</span></div>
<div><span class=«t-comment»># for - перебираємо елементи
послідовності</span></div>
<div></div>
<div>fruits = [<span class=«t-string»>«яблуко»</span>, <span
class=«t-string»>«груша»</span>, <span class=«t-
string»>«банан»</span>]</div>
<div></div>
<div><span class=«t-keyword»>for</span> fruit <span class=«t-
keyword»>in</span> fruits:</div>
<div>    <span class=«t-func»>print</span>(<span
class=«t-string»>f»Фрукт: {fruit}»</span>)</div>
<div></div>
<div><span class=«t-comment»># while - поки умова
виконується</span></div>
<div>count = <span class=«t-num»>0</span></div>
<div><span class=«t-keyword»>while</span> count &lt; <span class=«t-
num»>3</span>:</div>
<div>    <span class=«t-func»>print</span>(<span
class=«t-string»>f»Крок #{count + 1}»</span>)</div>
<div>    count += <span class=«t-
num»>1</span></div>`,
    output: ['Фрукт: яблуко','Фрукт: груша','Фрукт: банан','Крок
#1','Крок #2','Крок #3'],
    title: 'Цикли в Python', h3: 'Що таке цикл?',
    text: 'Цикл - спосіб виконувати один і той самий блок коду
багато разів. <strong style=«color:var(--accent)»>for</strong>
перебирає елементи колекції. <strong style=«color:var(--
accent)»>while</strong> виконується поки умова true.',
    tags: ['for loop', 'while loop', 'ітерація', 'break/continue'],
  },
  funcs: {
    file: 'lesson_04.py',
    code: `<div><span class=«t-comment»># Урок 4:
Функції</span></div>
<div></div>
<div><span class=«t-keyword»>def</span> <span class=«t-
func»>add</span>(a, b):</div>
<div>    <span class=«t-keyword»>return</span> a
+ b</div>
<div></div>
<div><span class=«t-keyword»>def</span> <span class=«t-
func»>greet</span>(name=<span class=«t-
string»>«Друже»</span>):</div>
<div>    <span class=«t-keyword»>return</span>
<span class=«t-string»>f»Привіт, {name}!»</span></div>
<div></div>
<div><span class=«t-func»>print</span>(add(<span class=«t-
num»>10</span>, <span class=«t-num»>25</span>))</div>
<div><span class=«t-func»>print</span>(greet())</div>
```

```
<div><span class=«t-func»>print</span>(greet(<span class=«t-
string»>«Соня»</span>))</div>`,
    output: ['35', 'Привіт, Друже!', 'Привіт, Соня!'],
    title: 'Функції', h3: 'Навіщо потрібні функції?',
    text: 'Функції дозволяють уникнути повторення коду та розбити
програму на логічні блоки. Функція приймає <strong
style=«color:var(--accent)»>параметри</strong> і повертає результат
через <strong style=«color:var(--accent)»>return</strong>.',
    tags: ['def', 'return', 'параметри', 'default args'],
  },
  lists: {
    file: 'lesson_05.py',
    code: `<div><span class=«t-comment»># Урок 5:
Списки</span></div>
<div></div>
<div>nums = [<span class=«t-num»>3</span>, <span class=«t-
num»>1</span>, <span class=«t-num»>4</span>, <span class=«t-
num»>1</span>, <span class=«t-num»>5</span>, <span class=«t-
num»>9</span>]</div>
<div></div>
<div><span class=«t-comment»># Сортування</span></div>
<div><span class=«t-func»>print</span>(sorted(nums))</div>
<div></div>
<div><span class=«t-comment»># List comprehension</span></div>
<div>squares = [x**<span class=«t-num»>2</span> <span class=«t-
keyword»>for</span> x <span class=«t-keyword»>in</span> nums]</div>
<div><span class=«t-func»>print</span>(squares)</div>
<div></div>
<div><span class=«t-comment»># Фільтрація</span></div>
<div>evens = [x <span class=«t-keyword»>for</span> x <span class=«t-
keyword»>in</span> nums <span class=«t-keyword»>if</span> x % <span
class=«t-num»>2</span> == <span class=«t-num»>0</span>]</div>
<div><span class=«t-func»>print</span>(evens)</div>`,
    output: ['[1, 1, 3, 4, 5, 9]', '[9, 1, 16, 1, 25, 81]', '[4]'],
    title: 'Списки', h3: 'Що таке список?',
    text: 'Список (list) - впорядкована змінна колекція елементів.
Підтримує <strong style=«color:var(--accent)»>зрізи</strong>,
методи, <strong style=«color:var(--accent)»>list
comprehension</strong> для компактного синтаксису.',
    tags: ['list', 'append', 'comprehension', 'slicing'],
  }
};

let currentTab = 'loops';

function switchTab(tab) {
  currentTab = tab;
  const data = tabData[tab];
  document.querySelectorAll('.lesson-tab-btn').forEach((b,i) =>
b.classList.toggle('active', ['loops','funcs','lists'][i] === tab));
  document.getElementById('tabName').textContent = data.file;
  document.getElementById('codeBody').innerHTML = data.code;
```

```
  document.getElementById('codeOutput').innerHTML = `<span
style=«color:var(--text3); font-size:.8rem;»>Натисни ▶ Запустити, щоб
побачити результат</span>`;
  document.getElementById('theoryTitle').textContent = data.title;
  document.getElementById('theoryH3').textContent = data.h3;
  document.getElementById('theoryText').innerHTML = data.text;
  document.getElementById('conceptTags').innerHTML = data.tags.map(t
=> `<span class=«concept-tag»>${t}</span>`).join('');
}

function runCode() {
  const data = tabData[currentTab];
  const btn = document.querySelector('.code-run');
  btn.textContent = '↻ Виконання...';
  setTimeout(() => {
    btn.textContent = '▶ Запустити';
    let html = '';
    data.output.forEach((line, i) => {
      setTimeout(() => {
        html += `<div class=«output-line»>&gt;&gt; ${line}</div>`;
        document.getElementById('codeOutput').innerHTML = html;
      }, i * 120);
    });
  }, 500);
}

/* ── QUIZ ── */
const questions = [
  { q: 'Яка функція виводить текст на екран у Python?', opts:
['console.log()', 'echo()', 'print()', 'write()'], ans: 2 },
  { q: 'Як оголосити змінну з цілим числом 42 у Python?', opts:
['var x = 42', 'int x = 42', 'x = 42', 'x := 42'], ans: 2 },
  { q: 'Що виведе цей код: print(type(3.14))?', opts: [«<class
'int'>«, «<class 'float'>«, «<class 'str'>«, «<class 'double'>«],
ans: 1 },
  { q: 'Який оператор використовується для перевірки рівності в
Python?', opts: ['=', ':=', '==', '==='], ans: 2 },
  { q: 'Як правильно написати коментар у Python?', opts: ['//
коментар', '/* коментар */', '# коментар', '-- коментар'], ans: 3 },
];

let current = 0, score = 0, answered = false;

function renderQuestion() {
  const q = questions[current];
  document.getElementById('quizStep').textContent = `Питання
${current+1} / ${questions.length}`;
  document.getElementById('quizQ').textContent = q.q;
  document.getElementById('quizOptions').innerHTML =
q.opts.map((o,i) => `
    <div class=«quiz-option» onclick=«answer(${i})»>
      <span class=«option-key»>${['A','B','C','D'][i]}</span>
      ${o}
```

```
    </div>
  `).join('');
  document.getElementById('quizHint').textContent = 'Оберіть
відповідь';
  const btn = document.getElementById('nextBtn');
  btn.style.opacity = '.4'; btn.style.pointerEvents = 'none';
  answered = false;
}

function answer(i) {
  if (answered) return;
  answered = true;
  const q = questions[current];
  const opts = document.querySelectorAll('.quiz-option');
  opts[i].classList.add(i === q.ans ? 'correct' : 'incorrect');
  opts[q.ans].classList.add('correct');
  if (i === q.ans) score++;
  const hint = document.getElementById('quizHint');
  hint.textContent = i === q.ans ? '✓ Правильно!' : '✗ Неправильно';
  hint.style.color = i === q.ans ? 'var(--accent)' : 'var(--
accent3)';
  const btn = document.getElementById('nextBtn');
  btn.style.opacity = '1'; btn.style.pointerEvents = 'auto';
}

function nextQuestion() {
  current++;
  if (current >= questions.length) {
    document.getElementById('quizOptions').innerHTML = '';
    document.getElementById('quizStep').textContent = '';
    document.getElementById('quizQ').textContent = '';
    document.getElementById('quizNav').style.display = 'none';
    const result = document.getElementById('quizResult');
    result.classList.add('show');
    document.getElementById('resultScore').textContent = `${score} /
${questions.length} правильно`;
    const msgs = ['Чудовий результат! 🎉','Непогано! Є над чим
попрацювати 💪','Продовжуй практикуватись 📚'];
    document.getElementById('resultMsg').textContent = score >= 4 ?
msgs[0] : score >= 2 ? msgs[1] : msgs[2];
  } else {
    renderQuestion();
  }
}

function resetQuiz() {
  current = 0; score = 0; answered = false;
  document.getElementById('quizNav').style.display = 'flex';
  document.getElementById('quizResult').classList.remove('show');
  document.getElementById('quizHint').style.color = '';
  renderQuestion();
}
```

```
renderQuestion();

/* ── REVEAL ON SCROLL ── */
const observer = new IntersectionObserver((entries) => {
  entries.forEach(e => { if (e.isIntersecting)
e.target.classList.add('visible'); });
}, { threshold: 0.1 });

document.querySelectorAll('.reveal').forEach(el =>
observer.observe(el));
</script>
</body>
</html>
```

ДОДАТОК Б
СЛАЙДИ ПРЕЗЕНТАЦІЇ

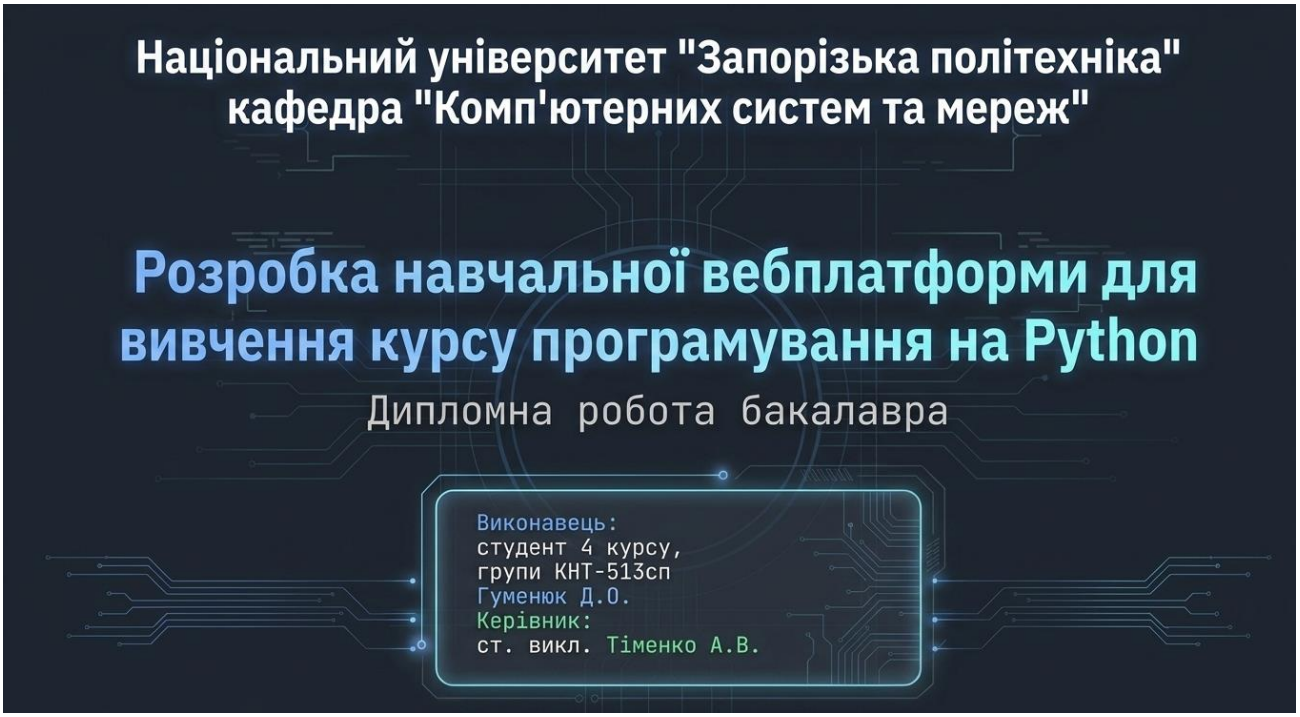

Рисунок Б.1 – Слайд 1

Рисунок Б.2 – Слайд 2

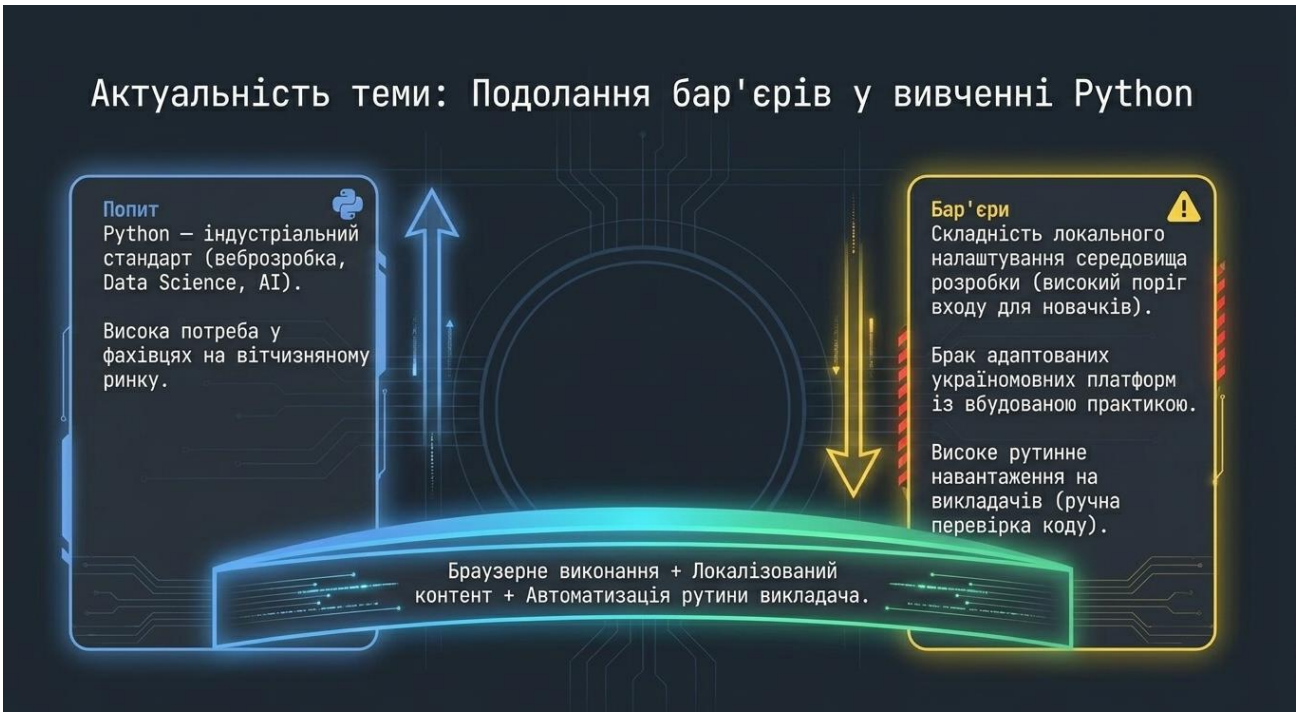

Рисунок Б.3 – Слайд 3

Рисунок Б.4 – Слайд 4

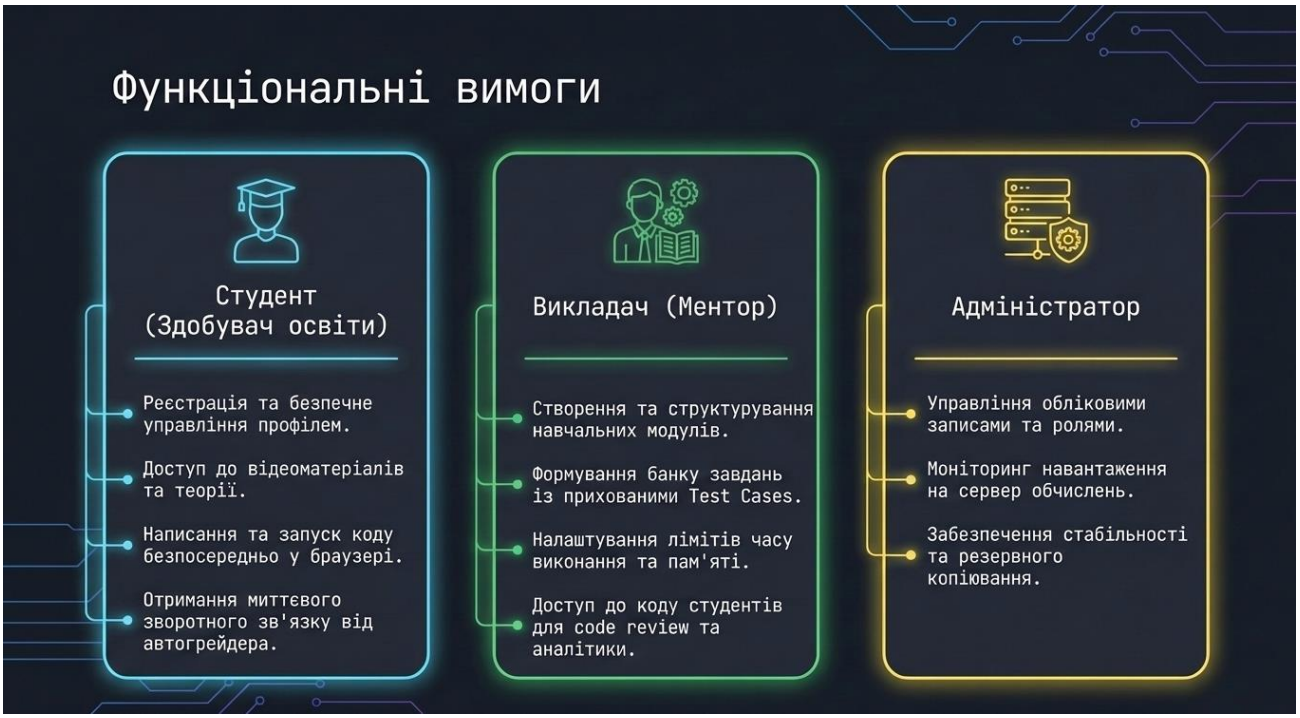

Рисунок Б.5 – Слайд 5

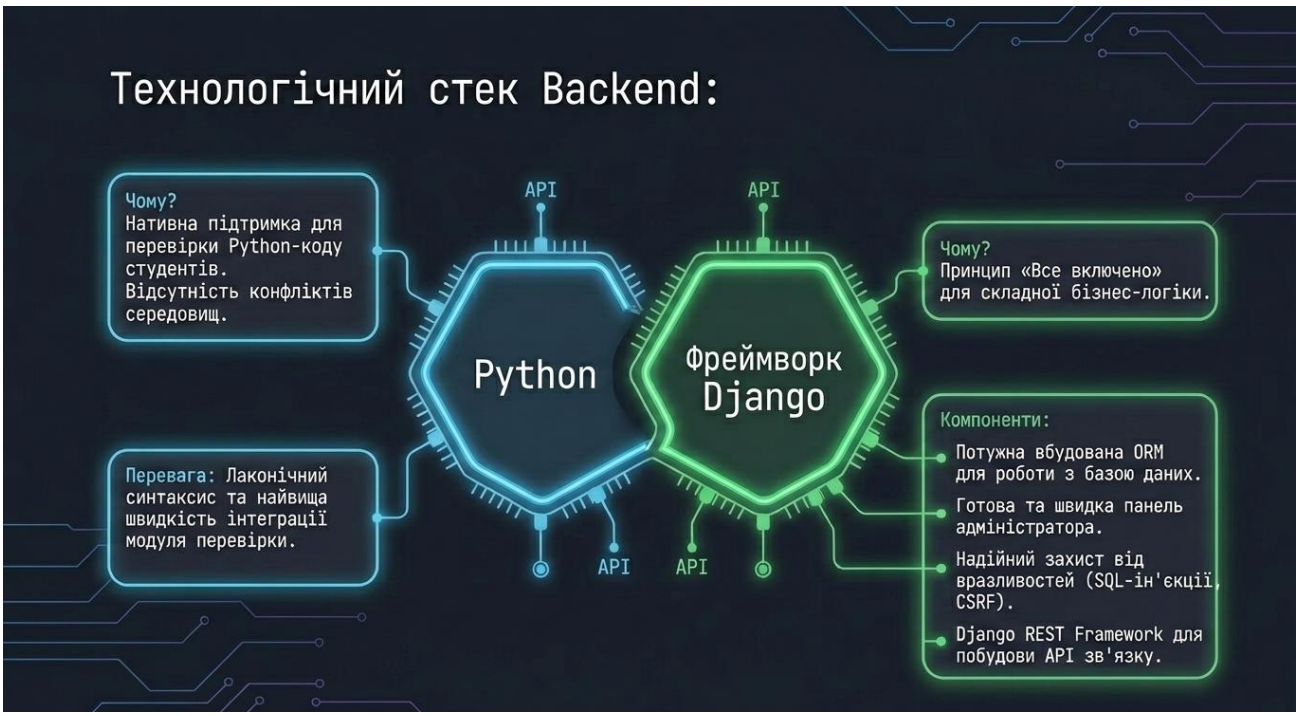

Рисунок Б.6 – Слайд 6

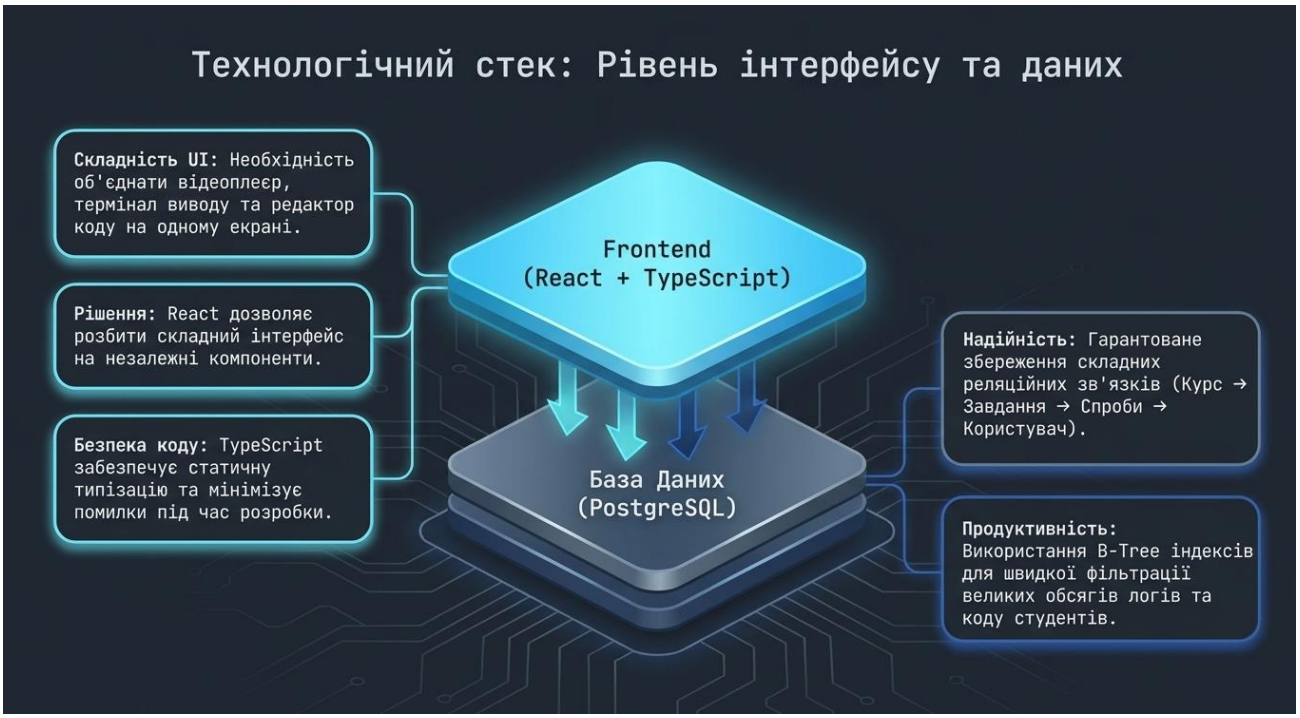

Рисунок Б.7 – Слайд 7

Рисунок Б.8 – Слайд 8

Рисунок Б.9 – Слайд 9

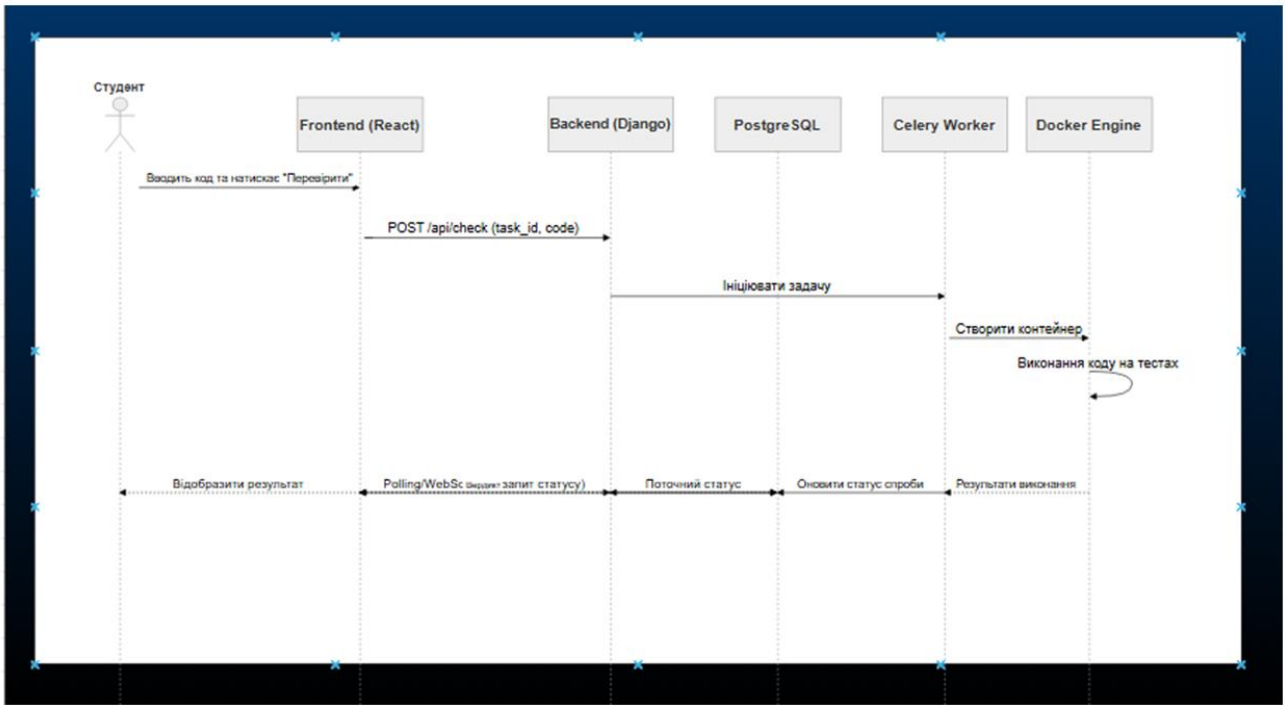

Рисунок Б.10 – Слайд 10

Рисунок Б.11 – Слайд 11

Рисунок Б.12 – Слайд 12

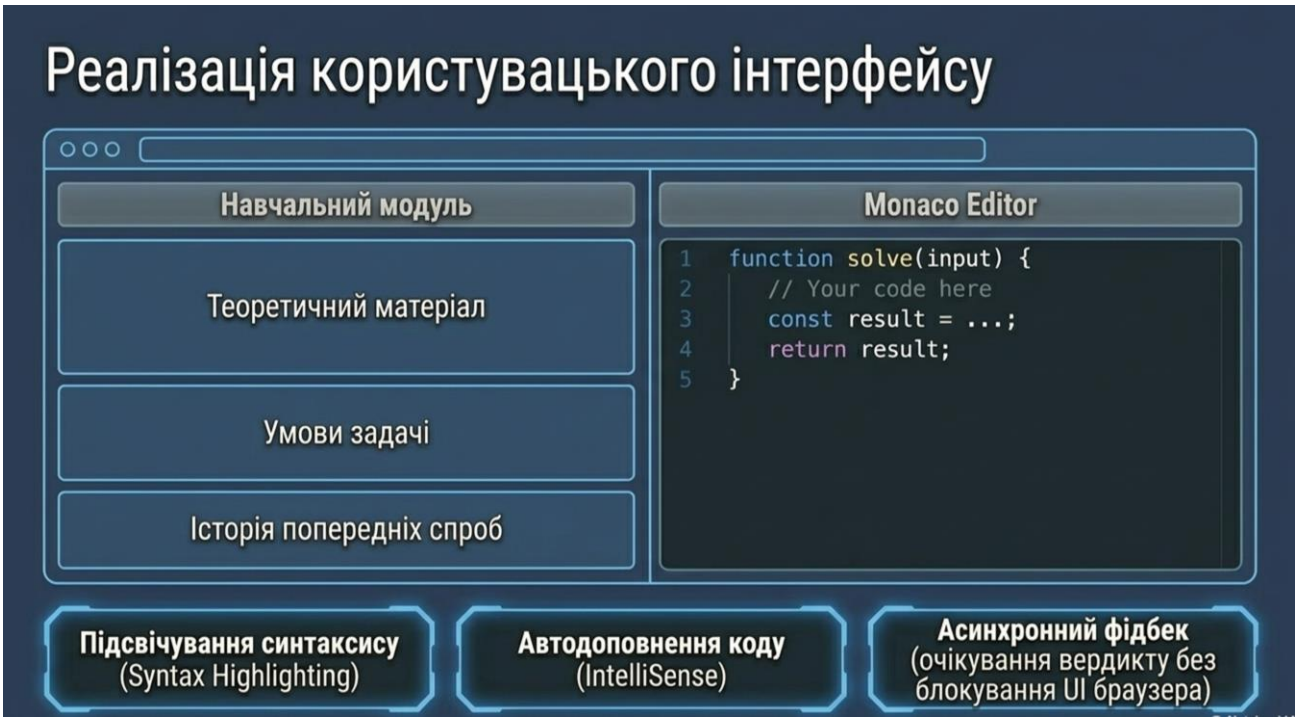

Рисунок Б.13 – Слайд 13

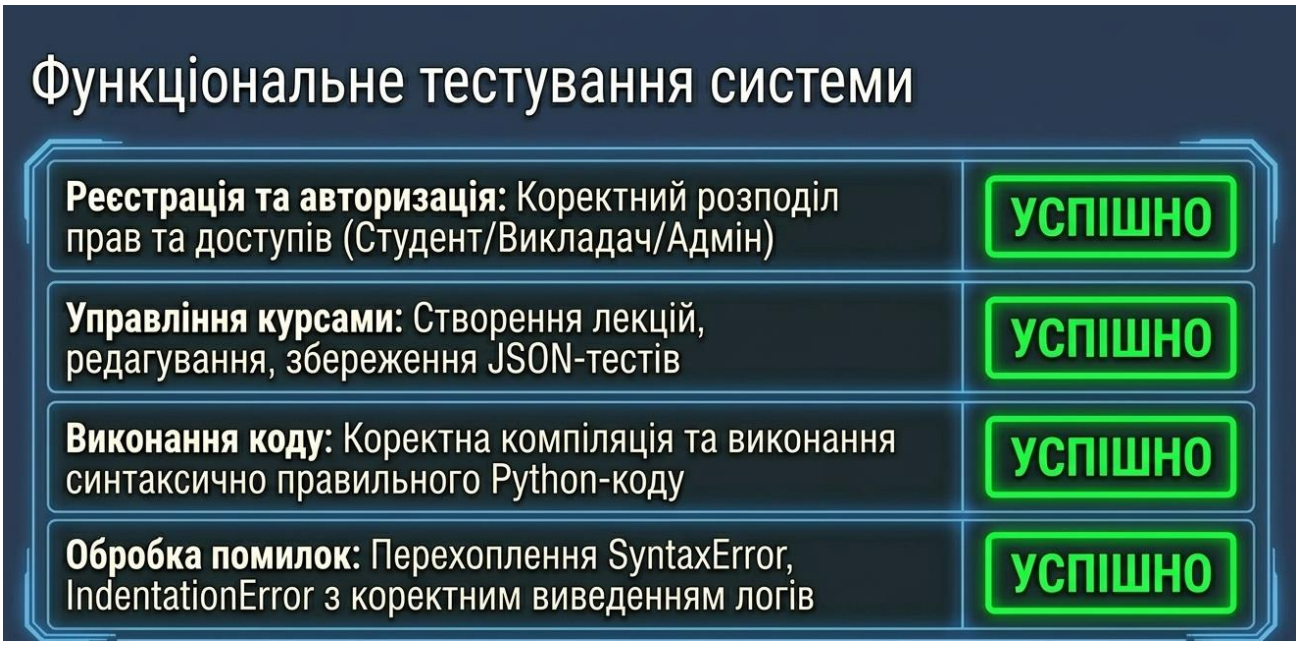

Рисунок Б.14 – Слайд 14

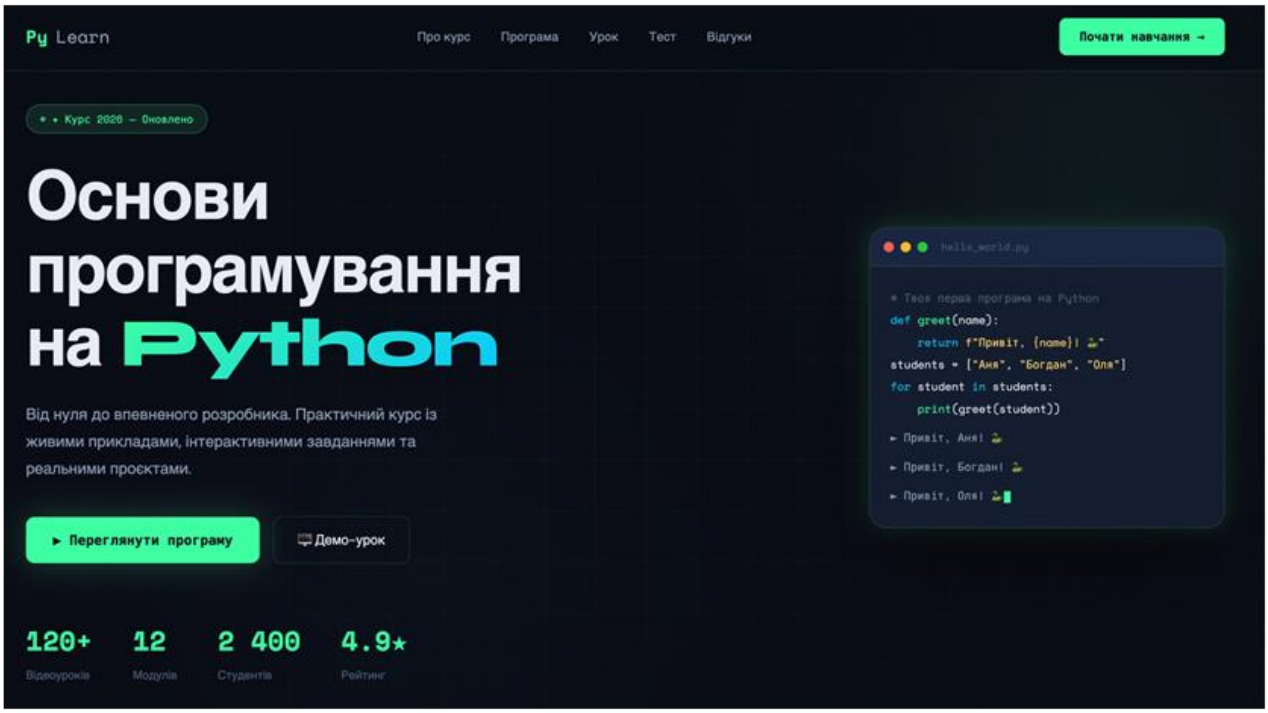

Рисунок Б.15 – Слайд 15

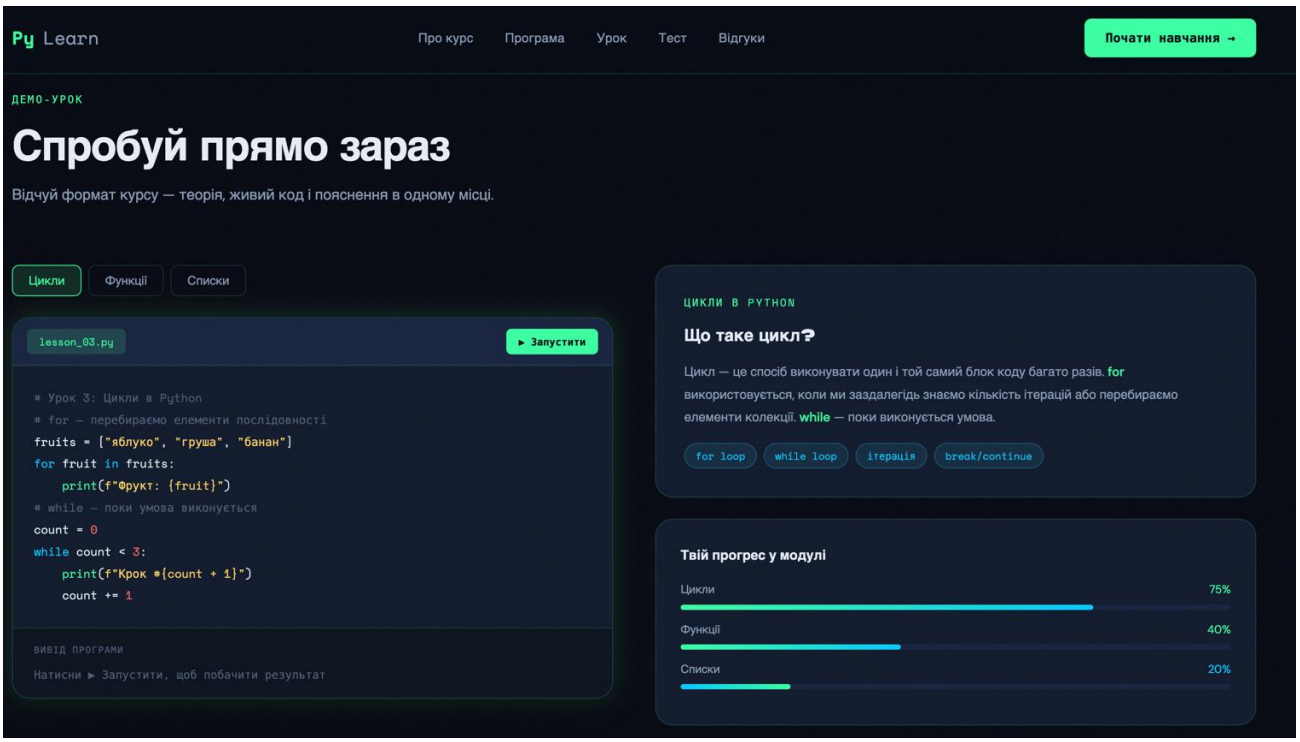

Рисунок Б.16 – Слайд 16

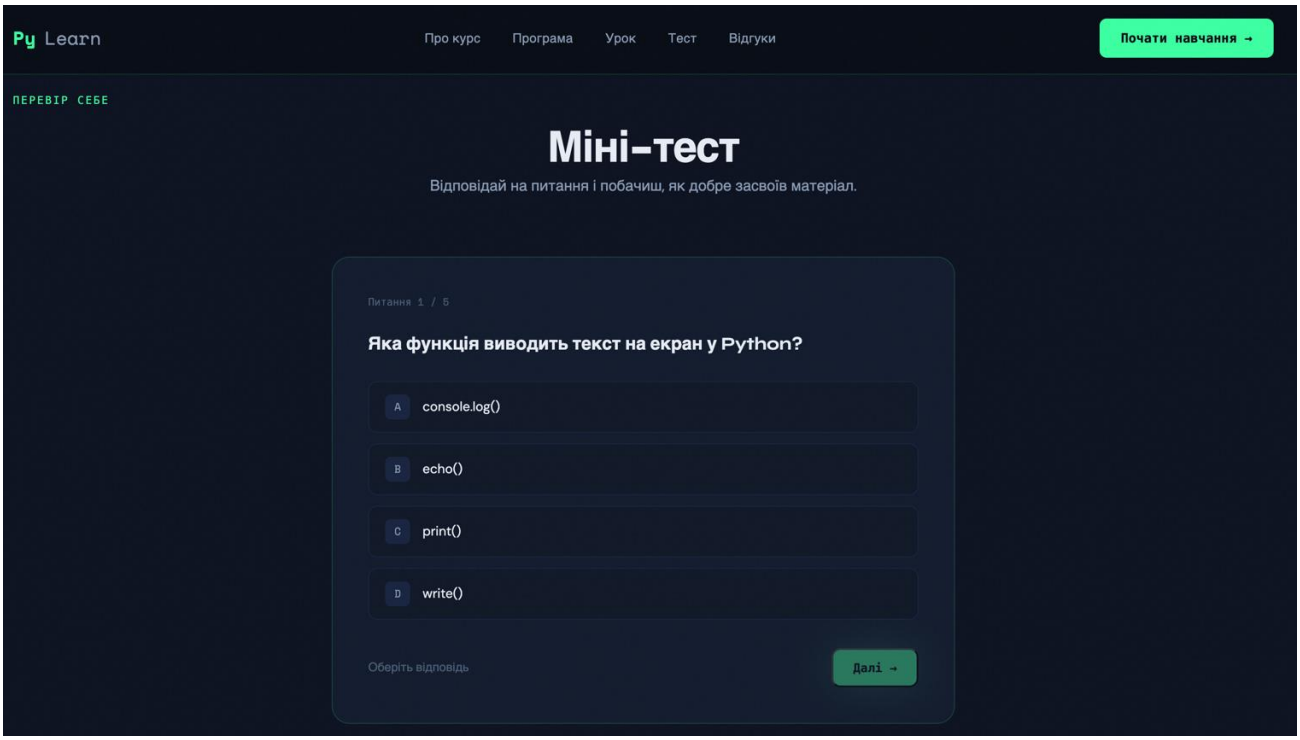

Рисунок Б.17 – Слайд 17

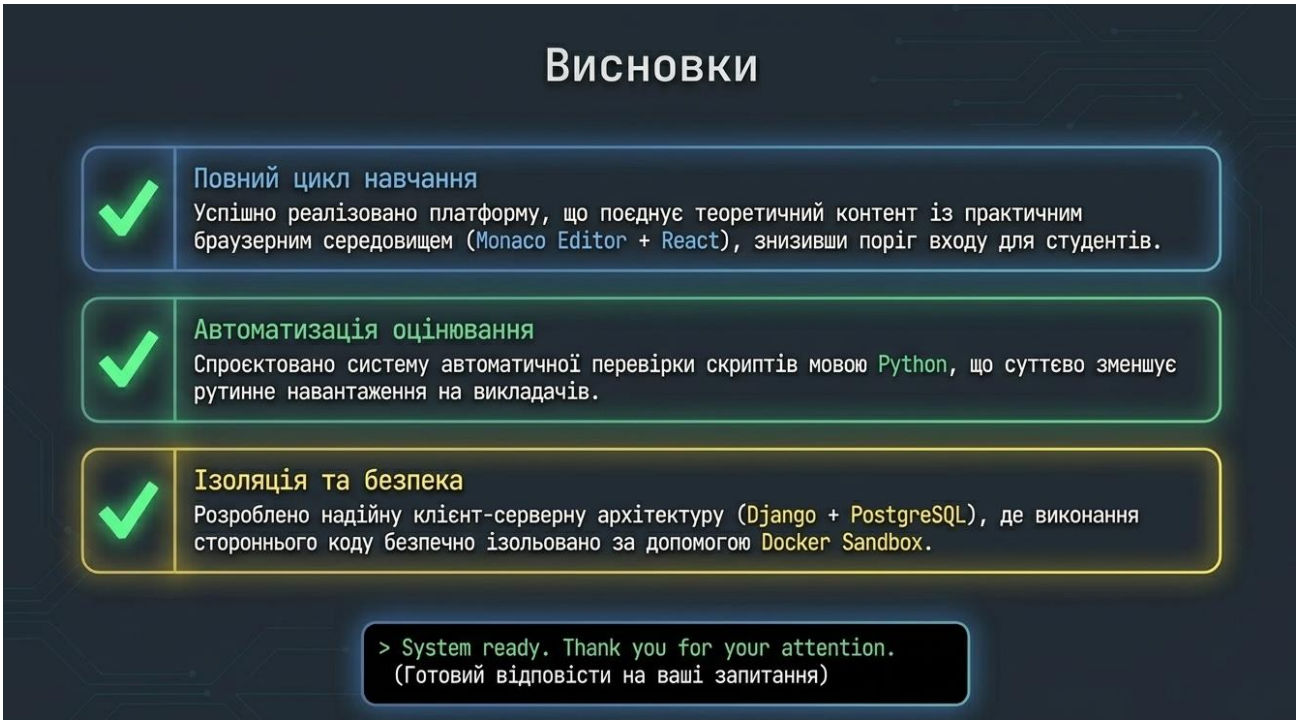

Рисунок Б.18 – Слайд 18