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

Допускається до захисту
Завідувач кафедри

_____Олена ОЛЬХОВСЬКА

(підпис)

«_____»_____2026 р

КВАЛІФІКАЦІЙНА РОБОТА

на тему

«РОЗРОБКА ПЛАТФОРМИ ДЛЯ ПРОВЕДЕННЯ ОНЛАЙН ОЛІМПІАД І КОНКУРСІВ З КОМП’ЮТЕРНИХ НАУК»

зі спеціальності 122 «Комп’ютерні науки»

освітня програма «Комп’ютерні науки»

ступеня бакалавра

Виконавець роботи Горбань Владислав Віталійович

_____ «___»_____2026 р.

(підпис)

Науковий керівник к., ф.-м. н., доц., Черненко Оксана Олексіївна

_____ «___»_____2026 р.

(підпис)

Рецензент

ПОЛТАВА 2026

ЗАТВЕРДЖУЮ

Завідувач кафедри _____ Олена ОЛЬХОВСЬКА

«___» _____ 2026 р.

ЗАВДАННЯ ТА КАЛЕНДАРНИЙ ГРАФІК

ВИКОНАННЯ КВАЛІФАКАЦІЙНОЇ РОБОТИ

на тему «Розробка платформи для проведення онлайн олімпіад і конкурсів з комп'ютерних наук» зі спеціальності 122 «Комп'ютерні науки»
освітня програма «Комп'ютерні науки».
Прізвище, ім'я, по батькові здобувача: Горбань Владислав Віталійович, група КН б-41.
Затверджена наказом ректора № 181-Н від « »_____ 2026 р.
Термін подання студентом роботи «___» _____ 20__ р.
Вихідні дані до кваліфікаційного проєкту: публікації та навчальні матеріали з теми веб-розробки, організації онлайн-олімпіад і конкурсів з комп'ютерних наук, матеріали дистанційних курсів з програмування, приклади існуючих платформ для проведення змагань.
Зміст пояснювальної записки (перелік питань, які потрібно розробити)
ВСТУП
1. ПОСТАНОВКА ЗАДАЧІ
2. ІНФОРМАЦІЙНИЙ ОГЛЯД
2.1 Класифікація платформ для проведення програмних олімпіад
2.2 Переваги та недоліки онлайн-платформ для навчальних змагань
2.3 Огляд схожих цифрових платформ
2.4 Необхідність та актуальність теми роботи
3. ТЕОРЕТИЧНА ЧАСТИНА
3.1 Опис матеріалу з теми
3.2 Опис структури сайту
3.3 Алгоритм роботи платформи онлайн-олімпіад
3.4 UML-діаграма та архітектура системи
3.5 Обґрунтування вибору мов програмування і технологій
4. ПРАКТИЧНА ЧАСТИНА
4.1 Опис розробки програмного забезпечення платформи
4.2 Опис роботи сайту для проведення олімпіад і конкурсів
4.3 Аналіз роботи платформи на основі тестового використання
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
Перелік графічного матеріалу: схема архітектури веб-додатку, UML-діаграма варіантів використання, рисунки інтерфейсу, таблиці функціональних вимог,

маршрутів сервера й результатів тестування.
Консультанти розділів кваліфікаційної роботи

Розділ	ПІП, посада консультанта	Підпис, дата завдання видав	Підпис, дата завдання прийняв
Постановка задачі	Черненко О. О.		
Інформаційний огляд	Черненко О. О.		
Теоретична частина	Черненко О. О.		
Практична частина	Черненко О. О.		

Календарний графік виконання кваліфікаційного проекту

№	Зміст роботи	Термін виконання	Фактичне виконання
1	Вступ		
2	Постановка задачі та визначення вимог		
3	Інформаційний огляд платформ і технологій		
4	Проєктування структури веб-додатку		
5	Розробка клієнтської частини		

	сайту		
6	Розробка серверної частини та файлового зберігання даних		
7	Тестування функціоналу та оформлення кваліфікаційної записки		
8	Підготовка до захисту і перевірка оформлення		

Дата видачі завдання «____»_____ 202_ р.
Здобувач вищої освіти Владислав Горбань
Науковий керівник к. ф.-м. н., доц. Оксана ЧЕРНЕНКО

Результати захисту кваліфікаційної роботи
Кваліфікаційна робота оцінена на _____
(балів, оцінка за національною шкалою, оцінка за ECTS)

Протокол засідання ЕК № ____ від «____» _____ 202_ р.

Секретар ЕК_____ _____
(підпис) (ініціали та прізвище)

Затверджую	**Погоджено**
Зав.кафедрою _____	Науковий керівник _____
к.ф.-м.н. Олена ОЛЬХОВСЬКА	к.ф.-м. н., доц. Оксана ЧЕРНЕНКО
«____» _____ 202_ р.	«____» _____ 202_ р.

План

кваліфікаційної роботи ступеня бакалавра
на тему «Розробка платформи для проведення онлайн олімпіад і конкурсів з комп'ютерних наук»
зі спеціальності 122 Комп'ютерні науки
освітня програма 122 Комп'ютерні науки
ступеня бакалавр
Горбань Владислав Віталійович
Прізвище, ім'я, по батькові

Здобувач вищої освіти _____ Владислав Горбань

«____» _____ 202_ р.

РЕФЕРАТ

Записка: присвячена розробці веб-платформи для проведення онлайн олімпіад і конкурсів з комп'ютерних наук. Додано завдання, календарний графік, план, анотацію, перелік скорочень, розгорнуті теоретичні й практичні розділи, таблиці, рисунки та додатки з фрагментами програмного коду.

Мета роботи – проєктування та програмна реалізація навчальної веб-платформи, яка забезпечує реєстрацію користувачів, авторизацію, перегляд задач, проходження тестових і конкурсних завдань, автоматичне нарахування балів, формування рейтингу, роботу з курсами, онлайн-заняттями та повідомленнями.

Об'єкт розробки – процес організації та проведення онлайн-олімпіад з комп'ютерних наук у навчальному середовищі. Предмет розробки – веб-платформа, що поєднує клієнтську частину, серверну логіку та файлове зберігання даних у форматі JSON.

У процесі розробки використано - HTML, CSS, JavaScript, Node.js, Express, express-session, bcryptjs та JSON. Клієнтська частина реалізована у файлі index.html, серверна частина – у файлі server.js. Для зберігання інформації застосовано JSON-файли users.json, problems.json, contests.json, courses.json, online.json та system.json.

Результатом роботи - є навчальний прототип платформи CS Olympiad, який може запускатися локально на сервері Node.js і використовуватися для демонстрації основних етапів проведення онлайн-конкурсів: реєстрації, участі, перевірки відповідей, перегляду результатів і формування рейтингу.

Ключові1слова: ОНЛАЙН-ОЛІМПІАДА, ВЕБ-ПЛАТФОРМА, КОМП'ЮТЕРНІ НАУКИ, HTML, CSS, JAVASCRIPT, NODE.JS, EXPRESS, JSON, РЕЙТИНГ, ТЕСТУВАННЯ.

ЗМІСТ

ВСТУП

У сучасній освіті інформаційні технології стали важливою умовою якісної організації навчання, перевірки знань і проведення студентських змагань. Особливе значення мають онлайн-платформи для олімпіад з комп’ютерних наук, оскільки вони дають змогу поєднати навчальні матеріали, тестові завдання, рейтинги та автоматизовану перевірку результатів в одному середовищі.

Традиційне проведення олімпіад потребує аудиторій, друкованих матеріалів, ручної перевірки, присутності учасників і значних організаційних витрат. Веб-платформа може автоматизувати більшість цих процесів: реєстрацію, доступ до завдань, фіксацію відповідей, підрахунок балів, формування рейтингу й надсилання повідомлень учасникам.

Актуальність теми полягає в необхідності створення доступного й зрозумілого інструменту для організації онлайн-конкурсів у межах навчального закладу. Така система може бути корисною викладачам, які проводять навчальні змагання, і студентам, які готуються до олімпіад, тренують алгоритмічне мислення та бажають швидко бачити результат своєї роботи.

Метою кваліфікаційної робота проєкту є розробка веб-платформи для проведення онлайн олімпіад і конкурсів з комп’ютерних наук . Платформа повинна забезпечувати роботу з користувачами, тестами, конкурсами, навчальними курсами, онлайн-заняттями, задачами, повідомленнями, рейтингом і акаунтом користувача.

Об’єктом дослідження є процес проведення програмних змагань та навчальних конкурсів у дистанційному форматі. Предметом дослідження є веб-додаток для організації онлайн-олімпіад із використанням клієнт-серверної архітектури та файлового зберігання даних.

Для досягнення мети було використано практичний підхід до створення веб-додатку: розроблено клієнтську частину в index.html, серверну частину в server.js, механізм авторизації, систему сесій, маршрути API, JSON-файли для зберігання даних і логіку роботи рейтингу.

Практична значимість роботи полягає у створенні сайту, який можна запускати локально або розгортати на хостингу. Розроблена система може стати основою для подальшого вдосконалення: додавання повноцінної бази даних, автоматичної перевірки програмного коду, панелі адміністратора, експорту результатів та генерації сертифікатів.

Пояснювальна записка складається з чотирьох основних розділів. У першому розділі сформульовано постановку задачі та вимоги до системи. У другому розділі виконано інформаційний огляд платформ і технологій. У третьому розділі описано теоретичні засади, структуру сайту, алгоритми та архітектуру. У четвертому розділі розкрито практичну реалізацію, роботу модулів і результати тестування.

1. ПОСТАНОВКА ЗАДАЧІ

Задачею кваліфікаційної роботи є проектування та розробка веб-платформи з теми «Розробка платформи для проведення онлайн олімпіад і конкурсів з комп'ютерних наук». Система має бути зрозумілою для студентів, зручною для викладачів та достатньо гнучкою для подальшого розширення функціоналу.

Основна ідея платформи полягає в тому, щоб надати користувачеві єдиний веб-інтерфейс для підготовки до олімпіад, перегляду навчальних матеріалів, участі в конкурсах, проходження тестів і перегляду власного результату. На відміну від простого набору HTML-сторінок, розроблена система має серверну частину, яка обробляє авторизацію, маршрути API, зберігання користувачів і результати.

Під час постановки задачі важливо врахувати, що платформа створюється як навчальний прототип . Тому в ній застосовано прості та доступні технології: HTML, CSS і JavaScript для інтерфейсу; Node.js і Express для серверної логіки; JSON-файли для збереження даних. Такий підхід дозволяє легко пояснити структуру проекту та швидко запустити його у VS Code або на локальному сервері.

Для досягнення поставленої мети необхідно виконати такі завдання:

– розробити сучасний адаптивний веб-інтерфейс платформи;

– реалізувати розділи головної сторінки, задач, конкурсів, курсів, онлайн-занять, рейтингу, повідомлень, контактів і акаунта;

– створити систему реєстрації, входу та виходу користувачів;

– забезпечити базовий поділ ролей на учасника й організатора;

– розробити серверну частину на Node.js із використанням Express;

– організувати збереження даних у JSON-файлах;

– реалізувати проходження конкурсів, тестування та обчислення результатів;

– створити рейтинг користувачів із можливістю оновлення балів;

– описати архітектуру, структуру даних, алгоритми та модулі системи;

1 – провести тестування основних функцій платформи.

Таблиця 1.1 – Основні функціональні вимоги до платформи

№	Модуль	Функціональні вимоги
1	Акаунт	Реєстрація, вхід, вихід, відображення профілю, ролі користувача, останнього результату й повідомлень.
2	Задачі	Перегляд навчальних задач, пошук, відображення теми, складності, умови, формату введення та виведення.
3	Конкурси	Перегляд відкритих і майбутніх конкурсів, реєстрація, старт та завершення участі.
4	Тести	Відображення питань, таймера, прогресу, перевірка відповідей і підрахунок балів.
5	Рейтинг	Сортування користувачів за балами та відображення місця учасника.
6	Панель організатора	Створення задач, курсів, онлайн-занять і конкурсів.
7	Повідомлення	Відображення системних повідомлень про проходження тестів, результати та грамоти.

1 Нефункціональні вимоги також мають велике значення. Платформа повинна швидко завантажуватися, мати зрозумілу навігацію, підтримувати

роботу в сучасних браузерах, зберігати дані без втрати та бути придатною для запуску на локальному сервері.

Безпека в навчальному прототипі реалізована на базовому рівні: паролі користувачів не зберігаються у відкритому вигляді, а хешуються за допомогою бібліотеки bcryptjs. Для збереження стану входу використовується express-session, що дозволяє серверу пам’ятати авторизованого користувача протягом сесії.

Очікуваним результатом виконання роботи є працездатний веб-додаток, який демонструє основні можливості платформи онлайн-олімпіад та може бути використаний як основа для подальшого розвитку проєкту.

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1 Класифікація платформ для проведення програмних олімпіад

Платформи для проведення програмних олімпіад можна класифікувати за призначенням, рівнем складності, способом перевірки рішень, форматом доступу та роллю користувачів. Така класифікація допомагає зрозуміти, який тип системи доцільно реалізовувати для навчального закладу.

За призначенням платформи поділяються на навчальні, змагальні та комбіновані. Навчальні системи орієнтовані на поступове опанування тем, розв’язування простих задач і самоперевірку. Змагальні платформи забезпечують обмеження часу, рейтинг, таблицю результатів і контроль чесності. Комбіновані системи поєднують навчальні матеріали, курси, задачі та конкурси(Табл. 2.1).

За типом перевірки відповіді можуть бути тестовими, ручними або автоматичними. У тестовому форматі користувач обирає варіант відповіді, а система одразу підраховує результат. У ручному форматі організатор перевіряє роботу окремо. В автоматичному форматі код користувача запускається на наборах тестів, що є складнішим, але найбільш характерним для професійних олімпіадних систем.

Розроблена платформа належить до комбінованих навчально-змагальних систем. Вона містить розділи задач, конкурсів, тестів, курсів, онлайн-занять і рейтингу. Така структура робить її корисною як для підготовки, так і для організації невеликих конкурсів у межах університету.

Таблиця 2.1 – Класифікація платформ для проведення програмних олімпіад

Критерій	Типи платформ	Характеристика
Призначення	Навчальні, змагальні, комбіновані	Визначають, чи система призначена лише для навчання, лише для конкурсів або для обох задач.
Перевірка	Тестова, ручна, автоматична	Показує, як система обчислює результат і чи потрібна участь викладача.
Доступ	Локальні, веб-платформи, хмарні сервіси	Визначає спосіб запуску та доступу до системи.
Ролі	Учасник, організатор, адміністратор	Описує права різних типів користувачів.
Складність	Прості навчальні, університетські, професійні	Визначає кількість функцій, рівень безпеки та масштабованість.

2.2 Переваги та недоліки онлайн-платформ для навчальних змагань

Онлайн-платформи для навчальних змагань мають значні переваги перед традиційним форматом проведення олімпіад. Вони дозволяють автоматизувати реєстрацію, зменшити кількість паперових матеріалів, швидко обчислювати результати та зберігати історію участі користувачів.

Важливою перевагою є доступність. Студент може працювати з платформою з дому, аудиторії або іншого місця, де є комп’ютер і браузер.

Викладач може підготувати конкурс, оновити задачі та переглянути результати без ручного збору робіт.

Однак такі системи мають і недоліки. Вони залежать від стабільності сервера та інтернет-з'єднання. Якщо система не має достатнього захисту, можуть виникнути ризики втрати даних або несанкціонованого доступу. Крім того, автоматична перевірка програмного коду потребує складної інфраструктури, ізоляції виконання та захисту від шкідливих програм(Табл. 2.2).

У межах даного проєкту реалізовано навчальний прототип, тому акцент зроблено на зрозумілій структурі, роботі інтерфейсу, збереженні даних і базовій серверній логіці. Це дозволяє поступово розвивати платформу, додаючи складніші механізми в майбутньому.

Таблиця 2.2 – Переваги та недоліки онлайн-платформ для навчальних змагань

№	Перевага	Можливий недолік
1	Доступ до завдань із будь-якого комп'ютера через браузер.	Залежність від роботи сервера та мережі.
2	Швидкий підрахунок балів і формування рейтингу.	Потрібно правильно реалізувати логіку перевірки.
3	Єдине місце для задач, курсів, конкурсів і повідомлень.	Інтерфейс може стати перевантаженим без продуманого дизайну.
4	Можливість зберігати історію результатів користувачів.	Потрібно забезпечити захист персональних даних.
5	Зменшення навантаження на організаторів.	Потрібні навички адміністрування та підтримки

		системи.

2.3 Огляд схожих цифрових платформ

Для розробки власної платформи доцільно розглянути відомі сервіси, які використовуються для навчання програмуванню та проведення алгоритмічних змагань. Найчастіше згадуються Codeforces, HackerRank, LeetCode, AtCoder та e-olymp(Табл. 2.3).

Codeforces орієнтований на регулярні алгоритмічні контести та має потужну рейтингову систему. Платформа містить великий архів задач, обговорення й таблиці результатів. Водночас для новачків інтерфейс і рівень задач можуть бути складними.

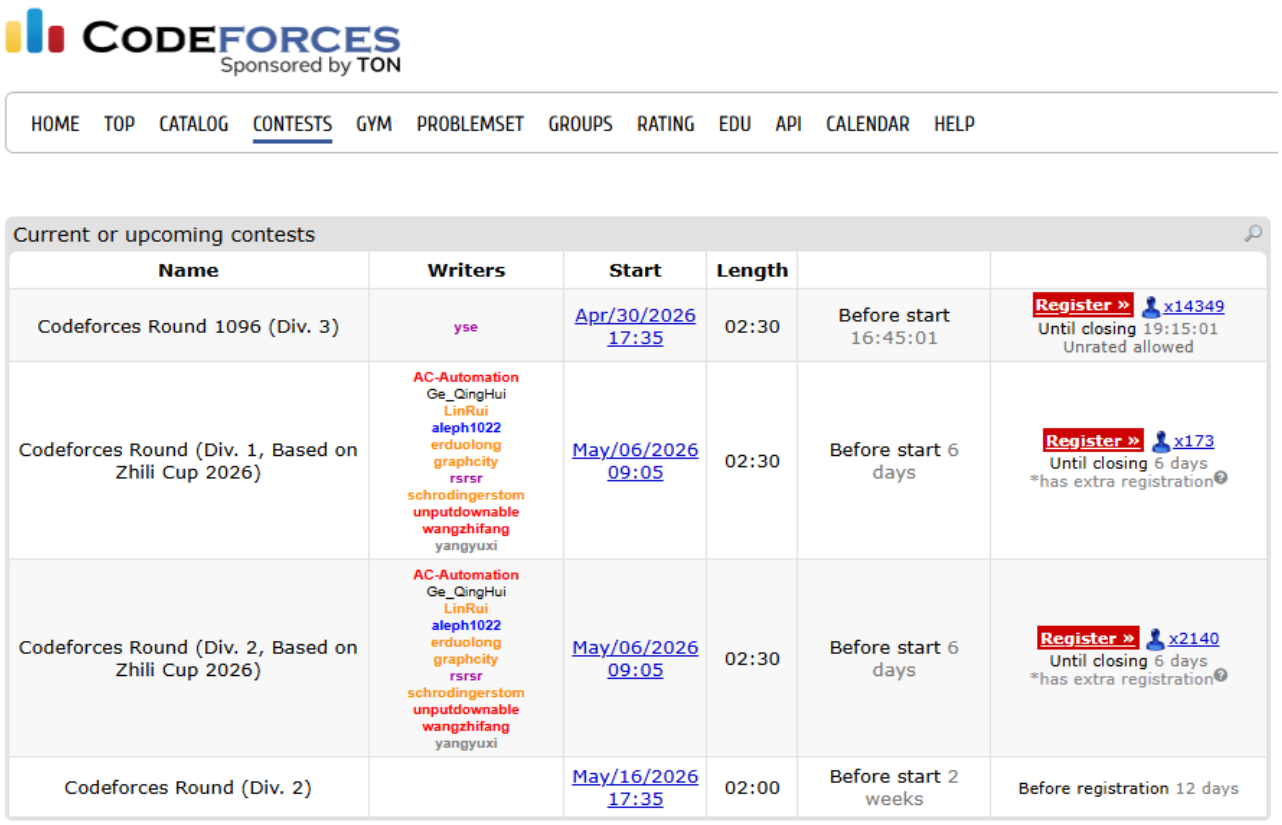


Рисунок 2.1 – Інтерфейс платформи Codeforces

HackerRank часто використовується для навчання, тренування різних тем програмування й оцінювання навичок користувачів. Платформа містить завдання з алгоритмів, структур даних, баз даних, штучного інтелекту та інших напрямів комп’ютерних наук. Вона також застосовується для перевірки рівня підготовки студентів і кандидатів під час технічного відбору. Завдяки автоматичній перевірці відповідей HackerRank дозволяє швидко отримати результат і зрозуміти, які теми потребують додаткового опрацювання.

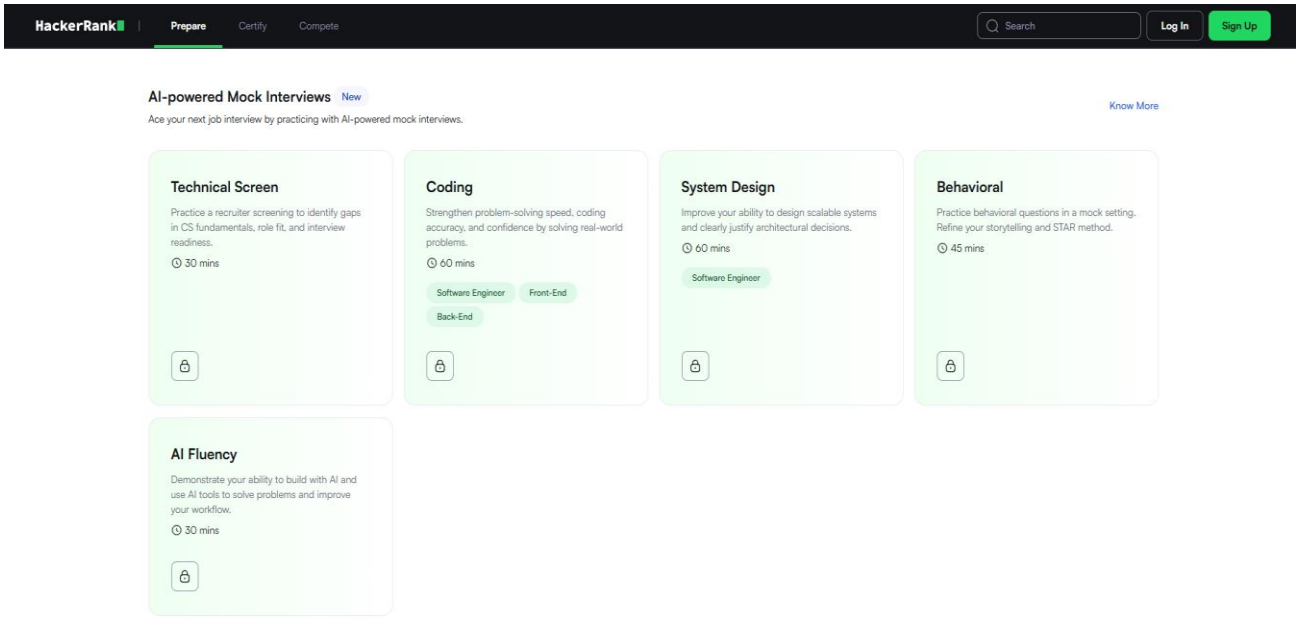


Рисунок 2.2 – Інтерфейс платформи HackerRank

LeetCode популярний серед студентів і фахівців, які готуються до технічних співбесід. Платформа містить велику кількість задач з алгоритмів, структур даних, баз даних та інших тем програмування. Користувачі можуть розв’язувати завдання різного рівня складності, перевіряти правильність своїх рішень і поступово покращувати навички написання коду. LeetCode також надає статистику виконання задач, що допомагає відстежувати прогрес підготовки. Основною перевагою платформи є практична спрямованість завдань, однак для проведення університетських олімпіад вона підходить не повністю, оскільки

більше орієнтована саме на підготовку до співбесід.

Водночас LeetCode є корисним прикладом для аналізу, оскільки демонструє зручну систему добору задач за рівнем складності та темами. Такі можливості можуть бути враховані під час розробки власної платформи, щоб користувачі могли поступово переходити від простіших завдань до складніших і краще готуватися до олімпіад з комп'ютерних наук.

Рисунок 2.3 – Інтерфейс платформи LeetCode

AtCoder має багато якісних контестів, які регулярно проводяться для учасників різного рівня підготовки. Платформа особливо популярна серед тих, хто цікавиться алгоритмічним програмуванням і хоче покращити навички розв'язування складних задач. Завдання на AtCoder зазвичай мають чіткі умови, різні рівні складності та автоматичну систему перевірки рішень. Це дозволяє користувачам швидко отримувати результат і аналізувати свої помилки. AtCoder також має рейтингову систему, завдяки якій учасники можуть порівнювати свої результати з іншими програмістами. Основним недоліком платформи для українських студентів може бути англомовний інтерфейс та

досить високий рівень складності деяких змагань.

Водночас AtCoder є корисним прикладом для розробки власної навчальної платформи, оскільки демонструє важливість чіткої структури задач, автоматичної перевірки відповідей і прозорої системи оцінювання. Саме ці принципи можна використати під час створення платформи для проведення онлайн-олімпіад у навчальному середовищі

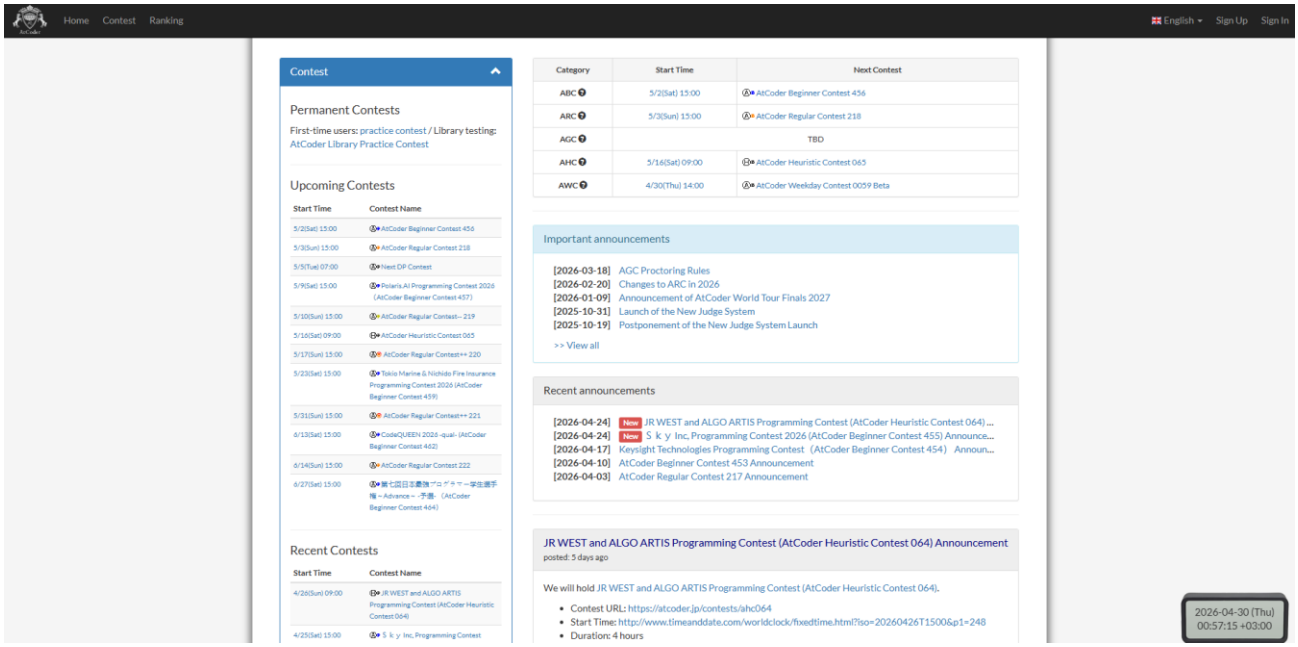


Рисунок 2.4 – Інтерфейс платформи AtCoder

e-olymp широко застосовується в українському освітньому середовищі для підготовки учнів і студентів до олімпіад з програмування. Платформа містить велику базу алгоритмічних задач різного рівня складності, що дозволяє користувачам поступово розвивати навички розв’язування задач. e-olymp підтримує автоматичну перевірку програмного коду, тому учасник одразу отримує результат після відправлення розв’язку. Це зручно як для самостійного навчання, так і для використання викладачами під час занять, тренувань або конкурсів. Перевагою платформи є її орієнтація на освітні потреби та доступність для українських користувачів. Водночас інтерфейс і

функціональність e-olymp можуть бути складними для новачків, тому власна навчальна платформа з простішим дизайном може бути корисною для студентів університету.

Також e-olymp є корисним прикладом для аналізу, оскільки показує важливість великої бази завдань, автоматичної перевірки та швидкого отримання результатів. Ці можливості можна врахувати під час розробки власної платформи, але зробити її більш простою, зрозумілою та адаптованою до потреб конкретного навчального закладу.

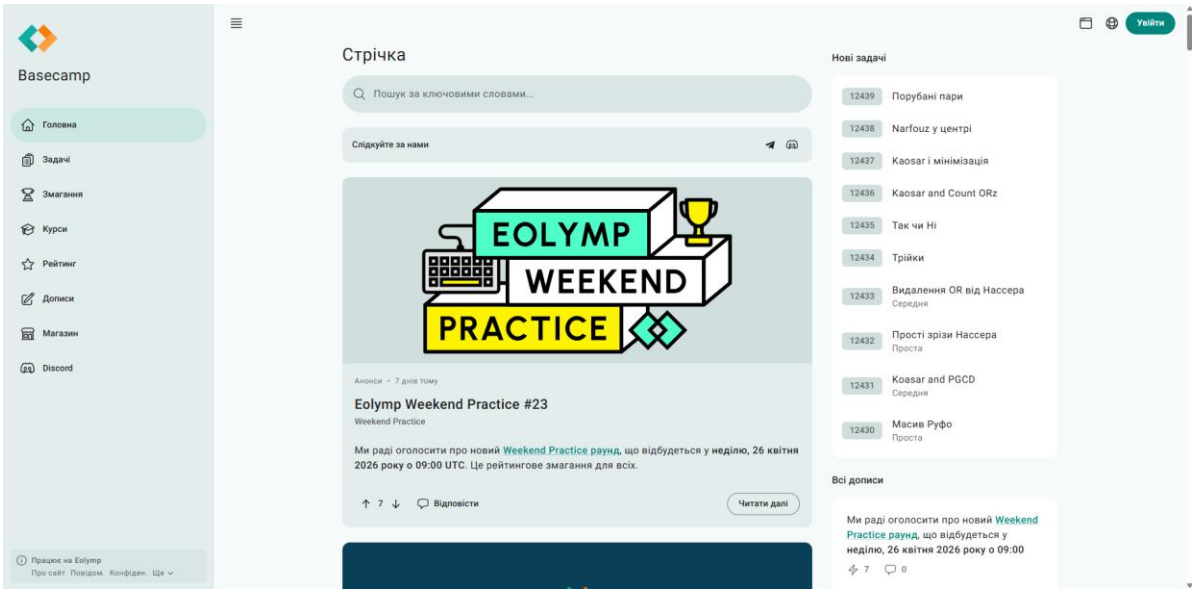


Рисунок 2.5 – Інтерфейс платформи e-olymp

Незважаючи на наявність потужних готових платформ, навчальному закладу може бути потрібна власна система. Вона дозволяє адаптувати набір задач під програму дисципліни, використовувати український інтерфейс, створювати локальні конкурси та не залежати від зовнішніх правил сторонніх сервісів.

Таблиця 2.3 – Порівняльна характеристика схожих цифрових платформ

Платформа	**Призначення**	**Переваги**	**Недоліки**

Codeforces	Алгоритмічні контести	Рейтинг, архів задач, активна спільнота	Складна для новачків
HackerRank	Навчання і відбір	Багато тем, різні формати завдань	Частина функцій орієнтована на компанії
LeetCode	Підготовка до співбесід	Зручні задачі, статистика, обговорення	Менше підходить для університетських олімпіад
AtCoder	Регулярні контести	Якісні задачі й чіткі правила	Англомовний інтерфейс
e-olymp	Освітні олімпіади	Поширення в навчальному середовищі	Не завжди підходить для власної структури курсів

2.4 Необхідність та актуальність теми роботи

Створення веб-платформи для проведення онлайн-олімпіад і конкурсів з комп’ютерних наук є актуальним завданням, оскільки сучасний освітній процес усе активніше використовує дистанційні технології, електронні ресурси та автоматизовані системи перевірки знань. Для студентів спеціальності «Комп’ютерні науки» важливо не лише вивчати теоретичний матеріал, а й регулярно виконувати практичні завдання, проходити тестування, брати участь у конкурсах та аналізувати власні результати [1],[3].

Традиційне проведення олімпіад потребує значної організаційної підготовки: формування завдань, реєстрації учасників, контролю часу, перевірки відповідей і підрахунку результатів. Якщо кількість учасників є великою, ручна перевірка

та обробка результатів займає багато часу. Використання онлайн-платформи дозволяє автоматизувати ці етапи, зробити процес проведення конкурсів швидшим, зручнішим і доступнішим для студентів та викладачів.

Необхідність розробки власної платформи пояснюється тим, що готові сервіси для програмних змагань не завжди повністю відповідають потребам конкретного навчального закладу. Наприклад, Codeforces, HackerRank, LeetCode та AtCoder мають широкий функціонал і використовуються для тренування навичок програмування, але вони переважно орієнтовані на загальну аудиторію, міжнародні змагання або підготовку до технічних співбесід [7], [8], [9],[10]. Через це такі платформи не завжди зручно адаптувати до конкретної навчальної дисципліни, групи студентів або внутрішнього університетського конкурсу.

Власна веб-платформа дає можливість створювати завдання, тести, конкурси й навчальні матеріали відповідно до рівня підготовки студентів та вимог викладача. Крім того, така система може мати український інтерфейс, просту навігацію, зрозумілу структуру та набір функцій, потрібних саме для навчального процесу. Це робить її зручнішою для використання в межах університету.1

Розроблена платформа CS Olympiad поєднує кілька основних модулів: реєстрацію та авторизацію користувачів, перегляд задач, участь у конкурсах, проходження тестових завдань, формування рейтингу, роботу з курсами, онлайн-заняттями, повідомленнями та акаунтом користувача. Така структура дозволяє використовувати платформу не тільки для проведення олімпіад, а й для підготовки студентів до конкурсів з комп’ютерних наук.

Для студентів така система є корисною, оскільки вона дозволяє тренувати алгоритмічне мислення, перевіряти знання, бачити власний прогрес і порівнювати результати з іншими учасниками. Наявність рейтингу підвищує мотивацію до навчання, оскільки студент бачить своє місце серед інших користувачів і може прагнути покращити результат під час наступних тестів або

конкурсів.

Для викладачів та організаторів платформа спрощує створення завдань, проведення конкурсів, перегляд результатів і контроль активності учасників. Це зменшує кількість ручної роботи та дозволяє швидше оцінювати рівень підготовки студентів. Крім того, збереження результатів у цифровому вигляді дає можливість аналізувати успішність учасників і використовувати ці дані для подальшого вдосконалення навчального процесу.

З технічної точки зору розробка такої платформи є доцільною, оскільки для її реалізації можна використати доступні та поширені веб-технології: HTML, CSS, JavaScript, Node.js, Express і JSON. Використання Node.js та Express дозволяє створити серверну частину для обробки запитів користувачів, а формат JSON є простим способом зберігання даних у навчальному прототипі [4], [5],[6].

Отже, актуальність теми полягає в потребі створення зручного навчального веб-середовища для організації онлайн-олімпіад і конкурсів з комп’ютерних наук. Розроблена платформа є доцільною для використання в освітньому процесі, оскільки поєднує навчання, практичну підготовку, контроль знань і змагальний елемент. У подальшому систему можна розширити, додавши повноцінну базу даних, автоматичну перевірку програмного коду, кабінет викладача, експорт результатів і систему сертифікатів.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1 Опис матеріалу з теми

1 Теоретичною основою розробки веб-платформи для проведення онлайн-олімпіад і конкурсів з комп’ютерних наук є поєднання принципів веброзробки, клієнт-серверної архітектури, організації навчального тестування та збереження результатів користувачів. Така система повинна не лише відображати інформацію, а й забезпечувати взаємодію користувача із сервером, обробку запитів, перевірку відповідей, збереження даних і формування рейтингу.

З погляду програмної інженерії, веб-платформа є програмною системою, яка складається з окремих функціональних модулів. До них належать модуль реєстрації та авторизації, модуль задач, модуль конкурсів, модуль тестування, модуль рейтингу, модуль повідомлень і модуль акаунта користувача. Поділ системи на модулі спрощує розробку, тестування та подальше вдосконалення програмного продукту [1], [2].

Основним призначенням платформи є створення навчального середовища, у якому користувач може готуватися до олімпіад, переглядати задачі, проходити тести, брати участь у конкурсах та аналізувати власні результати. Для студентів така система є зручною, оскільки дозволяє працювати з навчальними матеріалами в одному місці, а для викладачів і організаторів — спрощує процес підготовки та проведення конкурсів.

Важливим елементом теми є автоматизація перевірки результатів. У традиційному форматі відповіді учасників часто перевіряються вручну, що потребує часу та може призводити до помилок. У веб-платформі частина цього процесу автоматизується: користувач надсилає відповідь, система порівнює її з

правильним варіантом, обчислює результат і оновлює інформацію про користувача.

З технічної точки зору, розробка такої платформи потребує використання клієнтської та серверної частини. Клієнтська частина відповідає за інтерфейс користувача, відображення сторінок, кнопок, форм, таблиць, карток задач і конкурсів. Серверна частина відповідає за обробку даних, маршрути API, авторизацію, роботу з файлами та збереження результатів. Такий поділ відповідає загальним принципам побудови сучасних веб-додатків [4], [5].

У даній роботі платформа реалізується як навчальний прототип, тому для збереження даних використовується формат JSON. Він є простим для розуміння, легко обробляється засобами JavaScript і підходить для демонстрації основної логіки роботи системи [6]. У подальшому такий підхід можна замінити повноцінною базою даних, якщо платформа буде розширюватися.

Таким чином, матеріал теми охоплює теоретичні основи організації онлайн-олімпіад, принципи побудови веб-додатків, модульну структуру програмного забезпечення, клієнт-серверну взаємодію та використання сучасних вебтехнологій. Це дозволяє створити зрозумілу й функціональну платформу для проведення конкурсів з комп’ютерних наук.

3.2 Опис структури сайту

Сайт реалізовано як односторінковий веб-додаток, у якому розділи перемикаються без переходу на окремі HTML-файли. У файлі index.html кожна логічна сторінка має окремий блок із класом page. За допомогою JavaScript-функції openPage активний блок змінюється, а навігаційна кнопка отримує відповідний стан(Табл. 3.1).

Основними розділами сайту є головна сторінка, задачі, конкурси, курси, онлайн-заняття, рейтинг, повідомлення, контакти та акаунт. Така структура

відповідає темі роботи й робить платформу схожою на навчальний портал для підготовки та участі в олімпіадах.

На головній сторінці користувач бачить загальну інформацію про платформу, швидку статистику, новини та переходи до важливих розділів. Розділ задач призначений для ознайомлення з умовами завдань. Розділ конкурсів містить перелік змагань, статуси та кнопки для участі. Розділ курсів і онлайн-занять допомагає організувати навчальні матеріали.

Розділ акаунта містить форми реєстрації та входу, а також інформацію про користувача після авторизації. Рейтинг відображає користувачів, відсортованих за результатами. Повідомлення використовуються для інформування про завершення тестів, отримання балів або грамоти.

Таблиця 3.1 – Структура розділів веб-платформи

Розділ	**Призначення**	**Елемент у коді**
Головна	Опис платформи, статистика, швидкі переходи	id="home"
Задачі	Перегляд задач для підготовки	id="problems"
Конкурси	Участь у конкурсах і тестових змаганнях	id="contest"
Курси	Навчальні матеріали та підготовка	id="courses"
Онлайн	Онлайн-заняття і зустрічі	id="online"
Рейтинг	Список користувачів за результатами	id="rating"
Повідомлення	Системні повідомлення користувача	id="messages"
Контакти	Інформація для зв’язку	id="contacts"

Акаунт	Реєстрація, вхід і профіль	id="account"

3.3 Алгоритм роботи платформи онлайн-олімпіад

Алгоритм роботи платформи онлайн-олімпіад ґрунтується на послідовній взаємодії користувача з інтерфейсом, клієнтської частини із сервером та сервера з файлами даних. Такий підхід дозволяє забезпечити реєстрацію користувачів, проходження тестів, участь у конкурсах і збереження результатів.

Першим етапом роботи є відкриття сайту користувачем у браузері. Після завантаження сторінки користувач бачить головну сторінку платформи та навігаційне меню. За допомогою кнопок меню він може перейти до потрібного розділу: задач, конкурсів, курсів, рейтингу або акаунта.

Другим етапом є реєстрація або авторизація користувача. Під час реєстрації користувач вводить ім’я, електронну пошту, пароль і роль. Ці дані надсилаються на сервер за допомогою HTTP-запиту. Сервер перевіряє коректність введених даних, хешує пароль і записує нового користувача у файл з даними. Під час входу сервер перевіряє електронну пошту та пароль, після чого створює сесію користувача. Такий принцип роботи відповідає загальній логіці клієнт-серверної взаємодії у веб-додатках [4], [5].

Третім етапом є перегляд навчальних матеріалів і задач. Користувач може відкривати розділ задач, переглядати умови, складність, тему та приклади. Дані можуть завантажуватися з JSON-файлів, які зберігають інформацію про задачі, конкурси, курси та користувачів [6].

Четвертим етапом є участь у конкурсі або проходження тестування. Користувач обирає конкурс, запускає його та відповідає на запитання. Після завершення тесту відповіді передаються на сервер, де виконується їх перевірка. Система підраховує кількість правильних відповідей, визначає результат і оновлює дані користувача.

П’ятим етапом є формування рейтингу. Після оновлення результатів сервер

зберігає бали користувача та повертає оновлену інформацію клієнтській частині. Рейтинг дозволяє порівнювати результати учасників і визначати найкращих користувачів платформи.

Алгоритм роботи платформи можна подати так:

1. Користувач відкриває сайт у браузері.
2. Система завантажує головну сторінку та навігаційне меню.
3. Користувач проходить реєстрацію або авторизацію.
4. Сервер перевіряє дані та створює сесію.
5. Користувач переглядає задачі, курси або конкурси.
6. Користувач проходить тестування або бере участь у конкурсі.
7. Сервер перевіряє відповіді та підраховує результат.
8. Дані записуються у JSON-файли.
9. Користувач переглядає результат і місце у рейтингу.

Загалом алгоритм роботи платформи демонструє послідовний процес взаємодії користувача із системою. Кожна дія користувача викликає відповідну реакцію клієнтської або серверної частини, що забезпечує цілісну роботу веб-додатку.

3.4 UML-діаграма та архітектура системи

Архітектура платформи побудована за принципом поділу на клієнтську частину, серверну частину та файлове сховище. Клієнтська частина відповідає за інтерфейс і взаємодію з користувачем. Серверна частина обробляє запити, авторизацію, маршрути, оновлення балів і даних. Файлове сховище представлено JSON-файлами.

На рисунку 3.1 показано спрощену архітектуру системи. Браузер надсилає HTTP-запити до сервера Express, сервер читає або оновлює JSON-файли та

повертає відповідь у форматі JSON. Такий принцип дозволяє реалізувати навчальний прототип без використання складної бази даних.

Рисунок 3.1 – Архітектурна схема веб-платформи

На рисунку 3.2 наведено UML-діаграму варіантів використання. Вона демонструє два основні типи користувачів: учасника та організатора. Учасник реєструється, переглядає задачі, проходить тести, бере участь у конкурсах і переглядає рейтинг. Організатор має додаткові можливості створення задач, курсів і конкурсів.

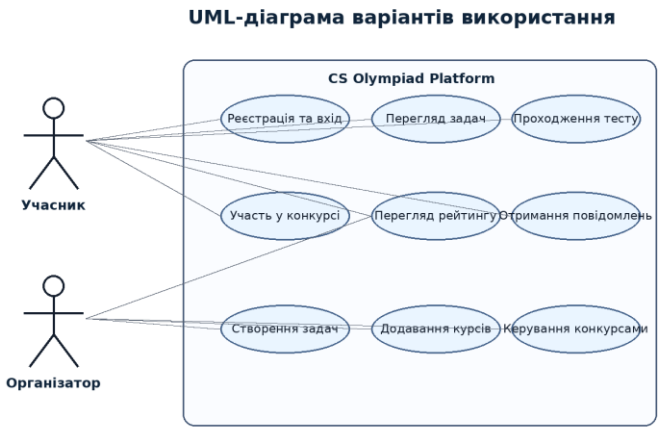


Рисунок 3.2 – UML-діаграма варіантів використання системи

3.5 Обґрунтування вибору мов програмування і технологій

Для реалізації проєкту обрано набір технологій, який є доступним, поширеним і зручним для навчальної веб-розробки. Основними технологіями стали HTML, CSS, JavaScript, Node.js, Express, express-session, bcryptjs та JSON. Такий вибір пояснюється тим, що ці інструменти дозволяють створити повноцінний веб-додаток із клієнтською частиною, серверною логікою, авторизацією користувачів і збереженням даних.

HTML використано для побудови структури сторінок веб-платформи. За допомогою HTML створюються основні блоки сайту: головна сторінка, розділи конкурсів, тестів, задач, курсів, рейтингу, акаунта, повідомлень і контактів. HTML є базовою мовою розмітки веб-сторінок, тому її використання є обов’язковим під час створення будь-якого сайту[12].

CSS застосовано для оформлення зовнішнього вигляду платформи. За допомогою CSS налаштовано кольори, шрифти, відступи, картки, кнопки, таблиці, адаптивність і загальний стиль інтерфейсу. Завдяки стилям сайт виглядає сучасно та зручно для користувача. Також CSS дає змогу реалізувати світлу й темну тему, ефекти наведення, красиві блоки статистики та зручне розміщення елементів на сторінці[13].

JavaScript використано для реалізації динамічної поведінки сайту. Саме JavaScript відповідає за перемикання між сторінками без повного перезавантаження, роботу тестів, підрахунок результатів, оновлення прогресу, взаємодію з кнопками, відображення повідомлень і передачу даних на сервер. Завдяки JavaScript платформа стає інтерактивною, а користувач може виконувати дії без зайвих переходів між сторінками[14] .

Важливою перевагою JavaScript є те, що ця мова використовується як на клієнтській, так і на серверній частині проєкту. Це робить розробку більш зрозумілою та зручною, оскільки не потрібно використовувати різні мови програмування для різних частин системи.

Node.js дозволяє виконувати JavaScript на сервері. У даному проєкті Node.js використовується для запуску серверної частини платформи. Сервер приймає запити від браузера, перевіряє дані користувачів, працює з файлами, оновлює результати та повертає відповіді клієнтській частині. Використання Node.js є доцільним, оскільки він добре підходить для створення швидких навчальних веб-додатків і не потребує складного налаштування[4].

Express застосовано для створення серверних маршрутів. За допомогою цього фреймворку реалізовано обробку запитів реєстрації, входу, виходу, отримання користувачів, оновлення рейтингу, роботи з конкурсами, задачами, курсами, онлайн-заняттями та повідомленнями. Express спрощує створення серверної логіки, оскільки дозволяє зручно описувати маршрути типу GET, POST і DELETE.

Наприклад, під час реєстрації користувача клієнтська частина надсилає запит на сервер, а Express приймає цей запит і передає його відповідному обробнику. Після цього сервер перевіряє введені дані, додає нового користувача до файлу та повертає відповідь про успішну реєстрацію або помилку.

Для авторизації користувачів у проєкті використано express-session. Цей модуль дозволяє зберігати стан входу користувача на сервері. Після успішного входу система запам’ятовує користувача, і він може працювати з платформою без повторного введення логіна та пароля під час кожного запиту. Це робить роботу сайту зручнішою та ближчою до реальних веб-сервісів.

Для захисту паролів застосовано bcryptjs. Зберігати паролі у відкритому вигляді небезпечно, тому перед записом у файл users.json пароль перетворюється на хеш. Хешування дозволяє підвищити рівень безпеки,

оскільки навіть у разі відкриття файлу з користувачами реальні паролі не будуть видимі. Під час входу система порівнює введений пароль із хешем і визначає, чи правильні дані ввів користувач.

JSON-файли обрано як простий спосіб зберігання даних у навчальному прототипі. У проєкті використовуються файли users.json, problems.json, contests.json, courses.json, online.json, messages.json та system.json. У них зберігаються користувачі, задачі, конкурси, курси, онлайн-заняття, повідомлення та службова інформація для роботи системи[6].

Такий підхід не потребує налаштування повноцінної бази даних, але демонструє основний принцип постійного збереження інформації. Дані не зникають після перезавантаження сторінки, а залишаються у відповідних файлах. Це зручно для навчального проєкту, оскільки структуру файлів легко переглядати, редагувати та аналізувати.

Окремою перевагою використання JSON є зрозумілість формату. Дані зберігаються у вигляді об’єктів і масивів, що добре поєднується з JavaScript. Наприклад, список користувачів або конкурсів можна легко зчитати, обробити та вивести на сторінку. Це спрощує розробку та тестування платформи.

Обраний набір технологій також добре підходить для запуску проєкту на локальному комп’ютері. Для роботи достатньо встановити Node.js, відкрити папку з проєктом у середовищі Visual Studio Code, встановити залежності через npm install і запустити сервер командою node server.js або npm start. Після цього сайт можна відкрити в браузері через локальну адресу.

Використання саме цих технологій є доцільним ще й тому, що вони мають велику кількість документації, прикладів і навчальних матеріалів. Це важливо для студентського проєкту, оскільки дозволяє швидше знаходити рішення помилок, розуміти принципи роботи коду та поступово вдосконалювати систему.

У майбутньому JSON-файли можна замінити на повноцінну базу даних, наприклад MongoDB або PostgreSQL. MongoDB добре підходить для роботи з документами, схожими на JSON, а PostgreSQL є надійною реляційною базою даних для складніших систем. Також у подальшому можна додати React для побудови інтерфейсу, JWT для авторизації, панель адміністратора та автоматичну перевірку програмного коду.

Отже, вибір HTML, CSS, JavaScript, Node.js, Express, express-session, bcryptjs і JSON є обґрунтованим для даного курсового проєкту(Табл. 3.2). Ці технології дозволяють створити зрозумілу, функціональну та розширювану веб-платформу для проведення онлайн-олімпіад і конкурсів з комп'ютерних наук. Вони забезпечують роботу інтерфейсу, серверної логіки, авторизації, рейтингу, конкурсів, тестів і збереження даних, що повністю відповідає меті розробки.

Таблиця 3.2 – Обґрунтування вибору технологій розробки

Технологія	**Роль у проєкті**	**Причина вибору**
HTML	Структура сторінок	Підтримується всіма браузерами та є основою веб-інтерфейсу.
CSS	Оформлення сайту	Дає змогу створити адаптивний дизайн, теми, картки, кнопки та візуальні ефекти.
JavaScript	Клієнтська логіка	Забезпечує перемикання сторінок, тестування, прогрес, взаємодію з API.
Node.js	Серверна частина	Дозволяє запускати JavaScript на сервері та створювати локальний веб-сервер.
Express	Маршрути API	Спрощує створення HTTP-маршрутів і обробку запитів.

express-session	Сесії користувачів	Зберігає стан авторизації між запитами.
bcryptjs	Захист паролів	Хешує паролі перед записом у файл користувачів.
JSON	Зберігання даних	Простий формат для навчального прототипу.

4. ПРАКТИЧНА ЧАСТИНА

4.1 Опис розробки програмного забезпечення платформи

1 Практична реалізація платформи виконана у вигляді веб-додатку CS Olympiad Platform. Проєкт розміщений у каталозі cs-olympiad-platform і містить клієнтський файл index.html, серверний файл server.js, файл залежностей package.json та набір JSON-файлів для збереження даних(Табл. 4.1).

Файл index.html є найбільшим у проєкті, оскільки містить розмітку сторінок, стилі та клієнтські JavaScript-функції. У ньому описано навігацію, головний екран, сторінку задач, конкурси, курси, онлайн-заняття, рейтинг, повідомлення, контакти, акаунт, а також форми організатора. Фактично цей файл виконує роль основної клієнтської частини сайту, з якою безпосередньо взаємодіє користувач.

У index.html реалізовано перемикання між розділами платформи без переходу на окремі HTML-сторінки. Це робить сайт схожим на односторінковий веб-додаток. Користувач може натискати на пункти меню, відкривати потрібні блоки, проходити тести, переглядати конкурси, дивитися рейтинг і працювати з акаунтом. Такий підхід спрощує структуру проєкту та робить роботу сайту більш зручною.

Окрему роль у клієнтській частині відіграють CSS-стилі. Вони відповідають за зовнішній вигляд платформи: кольори, відступи, кнопки, картки, таблиці, адаптивність, світлу й темну тему. Завдяки стилям інтерфейс виглядає сучасно та зрозуміло для користувача. Для навчальної платформи це важливо, оскільки студент повинен швидко знаходити потрібні розділи й не витрачати час на складну навігацію.

JavaScript-функції у файлі index.html забезпечують інтерактивність сайту. Вони відповідають за відкриття сторінок, обробку кнопок, відправлення даних на сервер, проходження тестів, оновлення прогресу, виведення результатів,

роботу з конкурсами та рейтингом. Саме завдяки JavaScript платформа не є статичною сторінкою, а реагує на дії користувача.

1 Файл server.js відповідає за роботу локального сервера. Він підключає Express, fs, bcryptjs, express-session і path. Сервер обслуговує статичні файли, приймає JSON-запити, працює з сесіями, читає та записує JSON-файли. Основний порт запуску – 3000.

Серверна частина обробляє основні запити користувача. Наприклад, під час реєстрації дані надсилаються з браузера на сервер, після чого сервер перевіряє їх, хешує пароль і записує нового користувача у файл users.json. Під час входу система перевіряє правильність електронної пошти та пароля, а після успішної авторизації створює сесію користувача.

Для збереження даних у проєкті використовуються окремі JSON-файли. У users.json зберігаються користувачі та їхні результати, у problems.json — задачі, у contests.json — конкурси, у courses.json — навчальні курси, в online.json — онлайн-заняття, а в system.json — службова інформація для роботи системи. Така структура є простою та зручною для навчального проєкту, оскільки дозволяє легко переглядати й редагувати дані.

У package.json вказані залежності: express, express-session і bcryptjs. Цей файл потрібен для встановлення необхідних модулів і запуску проєкту. Запуск платформи виконується командою npm start або node server.js. Після запуску сайт доступний за адресою http://localhost:3000.

Таким чином, практична реалізація платформи поєднує клієнтську та серверну частини. Клієнтська частина відповідає за інтерфейс і взаємодію з користувачем, а серверна — за обробку запитів, авторизацію, збереження даних і роботу основних модулів. Така структура дозволяє продемонструвати повний цикл роботи веб-додатку для проведення онлайн-олімпіад і конкурсів з комп’ютерних наук.

Таблиця 4.1 – Файли проєкту та їх призначення

Файл	**Призначення**
index.html	Клієнтська частина: HTML-структура, CSS-оформлення, JavaScript-логіка сторінок.
server.js	Серверна частина: маршрути API, авторизація, сесії, робота з JSON-файлами.
package.json	Опис проєкту, команда запуску та список залежностей.
users.json	Збереження користувачів, хешів паролів, балів, конкурсів і повідомлень.
system.json	Дата останнього щоденного скидання рейтингу.
problems.json	Список задач для підготовки.
contests.json	Список конкурсів і тестових змагань.
courses.json	Список навчальних курсів.
online.json	Список онлайн-занять.

Таблиця 4.2 – Основні маршрути серверної частини

Метод	**Маршрут**	**Призначення**
POST	/register	Реєстрація нового користувача
POST	/login	Авторизація користувача
POST	/logout	Вихід із системи
GET	/users	Отримання рейтингу користувачів
GET	/messages	Отримання повідомлень користувача
GET	/me	Отримання поточного користувача
GET	/problems	Робота із задачами

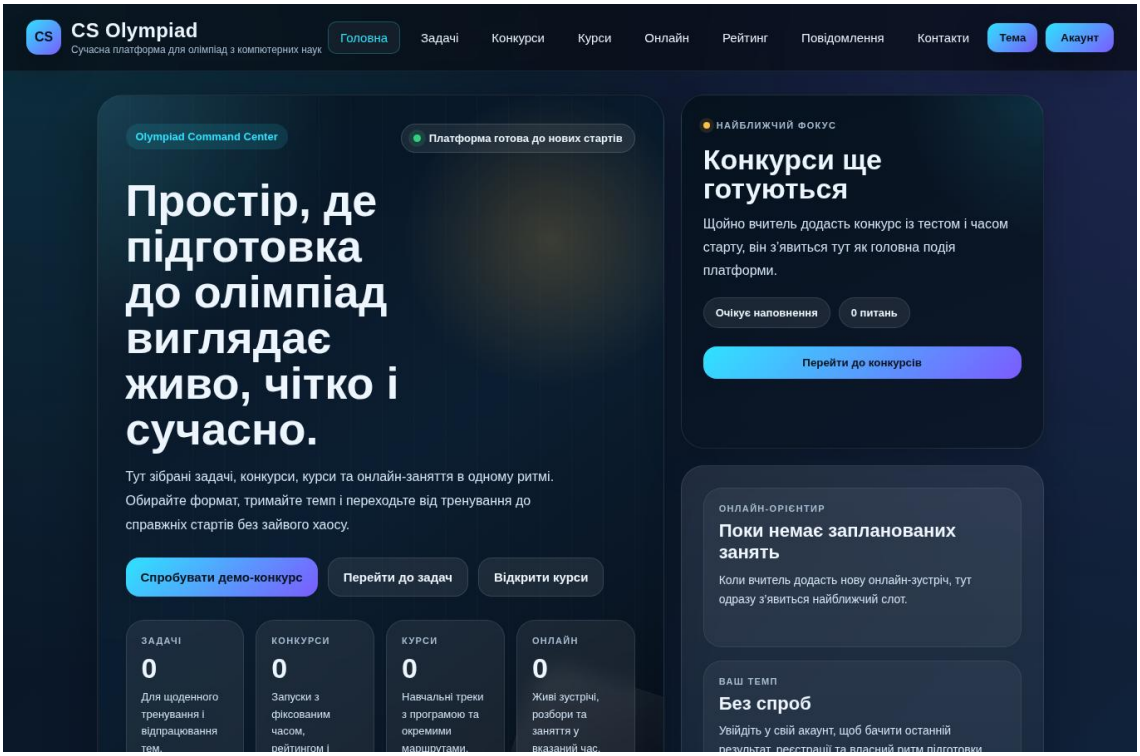
POST	/problems	Робота із задачами
GET	/online-lessons	Робота з онлайн-заняттями
POST	/online-lessons	Робота з онлайн-заняттями
GET	/contests	Робота з конкурсами
POST	/contests	Робота з конкурсами
DELETE	/contests/:id	Видалення конкурсу
GET	/courses	Робота з курсами
POST	/courses	Робота з курсами
DELETE	/courses/:id	Видалення курсу
DELETE	/problems/:id	Видалення задачі
DELETE	/online-lessons/:id	Видалення онлайн-заняття
POST	/update-score	Оновлення балів користувача
POST	/contest/register	Реєстрація на конкурс
POST	/contest/start	Початок участі в конкурсі
POST	/contest/finish	Завершення конкурсу й підрахунок результату

4.2 Опис роботи сайту для проведення олімпіад і конкурсів

1 Розроблений сайт для проведення онлайн-олімпіад і конкурсів з комп’ютерних наук є навчальною веб-платформою, яка об’єднує основні функції для підготовки студентів, організації конкурсів, проходження тестових завдань, перегляду результатів і формування рейтингу учасників. Сайт має зручну структуру, зрозумілу навігацію та поділ на окремі функціональні розділи.

Робота користувача із сайтом починається з головної сторінки. На ній розміщено назву платформи, короткий опис її призначення, основні переваги та кнопки для переходу до ключових розділів. Головна сторінка виконує інформаційну функцію та дає користувачеві загальне уявлення про можливості

системи. Також на ній можуть відображатися короткі статистичні блоки, наприклад кількість конкурсів, задач, користувачів або доступних навчальних матеріалів.

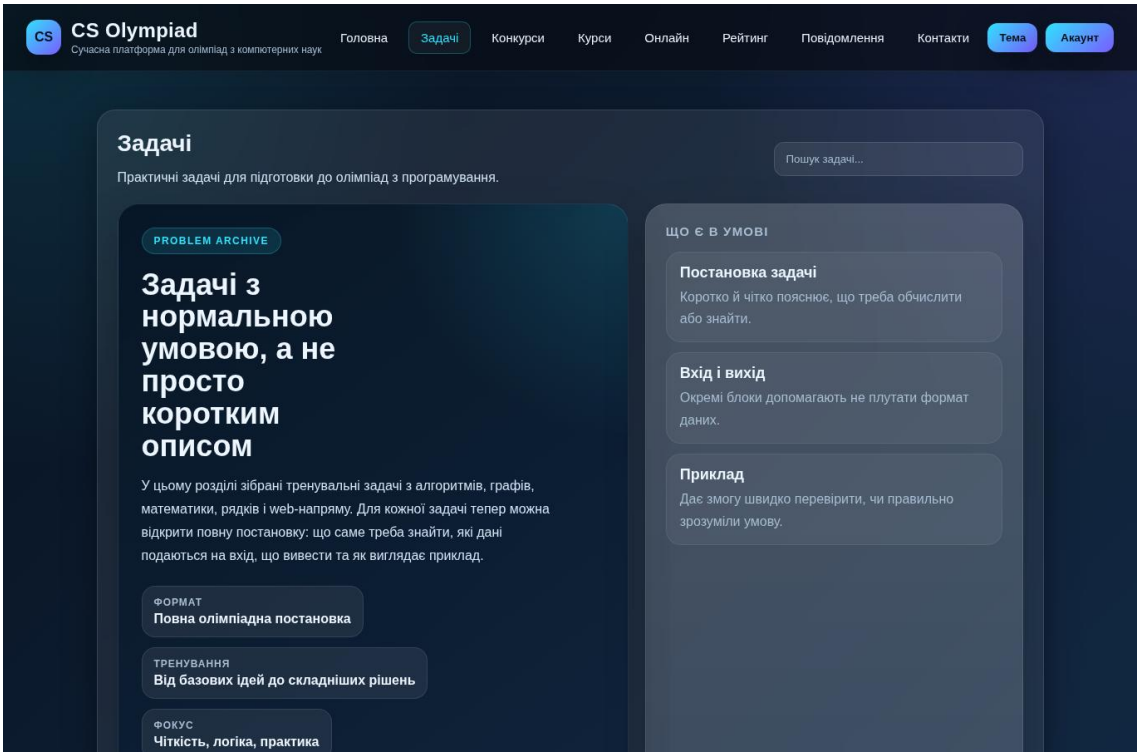


1 Рисунок 4.1 – Головна сторінка платформи CS Olympiad

Розділ «Задачі» призначений для підготовки учасників до олімпіад і конкурсів. На цій сторінці користувач може переглядати навчальні задачі з програмування. Кожна задача містить назву, тему, рівень складності, умову, формат вхідних і вихідних даних, а також приклад. Такий формат подання матеріалу наближений до реальних олімпіадних платформ, де користувач повинен уважно прочитати умову задачі та підготувати правильне рішення.

Для зручності користувача задачі можуть бути представлені у вигляді карток або списку. Це дозволяє швидко переглядати доступні завдання та обирати потрібне. Якщо задач багато, доцільним є використання пошуку або фільтрації

за темою чи складністю. Такий підхід робить підготовку до олімпіад більш структурованою.

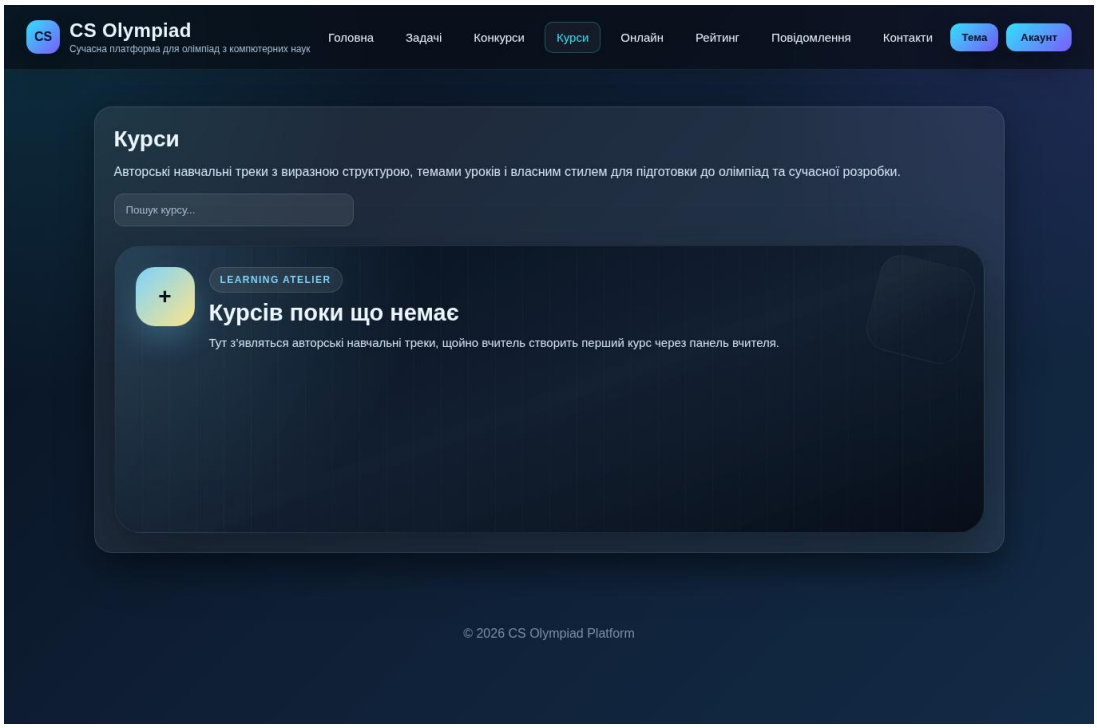


1 Рисунок 4.2 – Сторінка задач для підготовки до олімпіад

На сайті реалізовано розділ «Курси». Він призначений для розміщення навчальних матеріалів з комп’ютерних наук, програмування, алгоритмів і підготовки до олімпіад. Користувач може переглядати доступні курси, ознайомлюватися з їх описом і використовувати матеріали для самостійного навчання. Наявність такого розділу робить платформу корисною не лише для проведення конкурсів, а й для системної підготовки студентів.

Курси можуть містити теоретичні пояснення, приклади розв’язування задач, практичні завдання та рекомендації для самостійної роботи. Завдяки цьому студент має можливість поступово опановувати потрібні теми, повторювати складний матеріал і готуватися до участі в олімпіадах у зручному темпі. Для

викладача цей розділ є корисним інструментом, оскільки він дозволяє структурувати навчальні матеріали та доповнювати їх новими темами відповідно до потреб студентів. Таким чином, розділ «Курси» розширює функціональність платформи й перетворює її на повноцінне навчальне середовище.



1 Рисунок 4.3 – Розділ навчальних курсів

Розділ «Онлайн» може використовуватися для організації дистанційних занять, консультацій або зустрічей. У цьому розділі можуть відображатися назва заняття, дата, час, опис і посилання для підключення. Така функція є корисною для викладачів, які проводять підготовку до олімпіад у дистанційному або змішаному форматі.

За допомогою цього розділу студенти можуть заздалегідь дізнаватися про заплановані заняття, не пропускати важливі консультації та швидко переходити до онлайн-зустрічі. Викладач, у свою чергу, може повідомляти учасників про тему заняття, час проведення та додаткові матеріали для підготовки. Це робить

навчальний процес більш організованим і зручним, особливо коли учасники перебувають у різних місцях. Таким чином, розділ «Онлайн» доповнює платформу засобами дистанційної взаємодії та підтримує постійний зв’язок між викладачем і студентами.

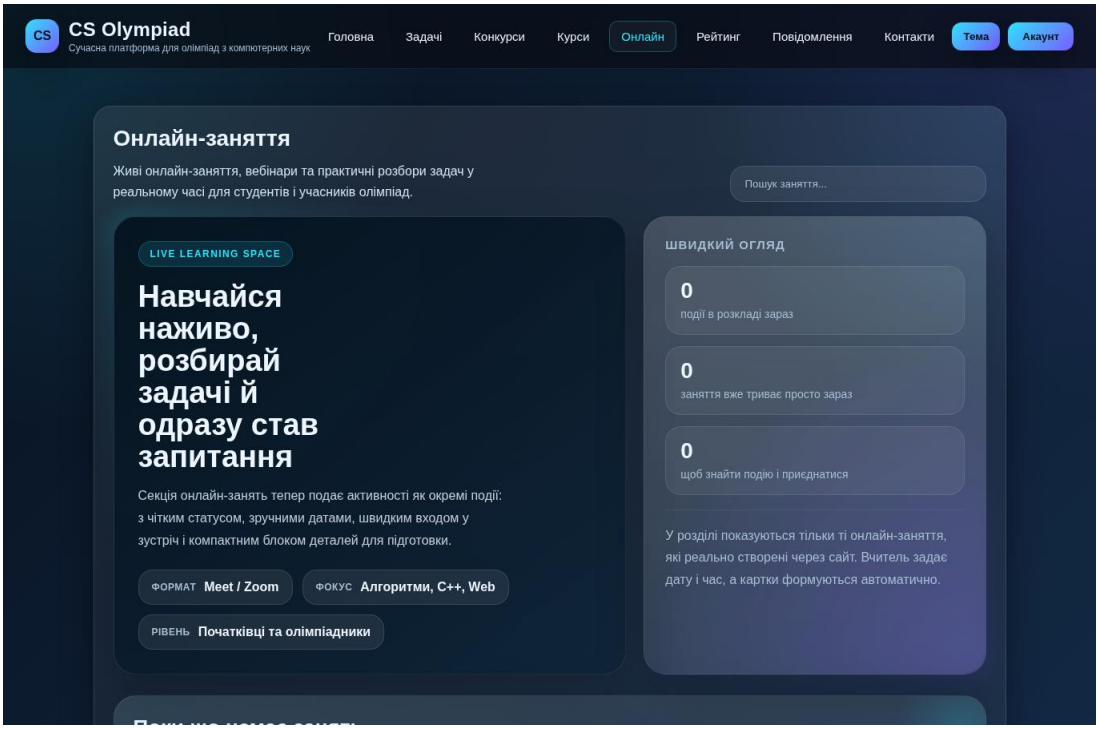


Рисунок 4.4 – Розділ онлайн-занять

У верхній частині сайту розміщено навігаційне меню. Воно дозволяє швидко переходити між основними сторінками платформи: «Головна», «Задачі», «Конкурси», «Курси», «Онлайн», «Рейтинг», «Повідомлення», «Контакти» та «Акаунт». Така структура робить сайт зручним для користувача, оскільки всі основні можливості розміщені в одному місці. Активний розділ візуально виділяється, що допомагає користувачеві краще орієнтуватися в системі.

Однією з важливих частин сайту є сторінка акаунта. У цьому розділі користувач може зареєструватися або увійти в систему. Під час реєстрації користувач вводить ім’я, електронну пошту, пароль і обирає роль. Після натискання кнопки реєстрації дані передаються на сервер, де перевіряються та

зберігаються у файлі користувачів. Пароль не зберігається у відкритому вигляді, а хешується за допомогою бібліотеки bcryptjs, що підвищує безпеку системи.

Після входу користувач отримує доступ до особистого профілю. У профілі може відображатися ім’я користувача, електронна пошта, роль, останній результат, повідомлення та інформація про участь у конкурсах. Якщо користувач має роль організатора або викладача, йому можуть бути доступні додаткові можливості для створення задач, конкурсів, курсів та онлайн-занять.

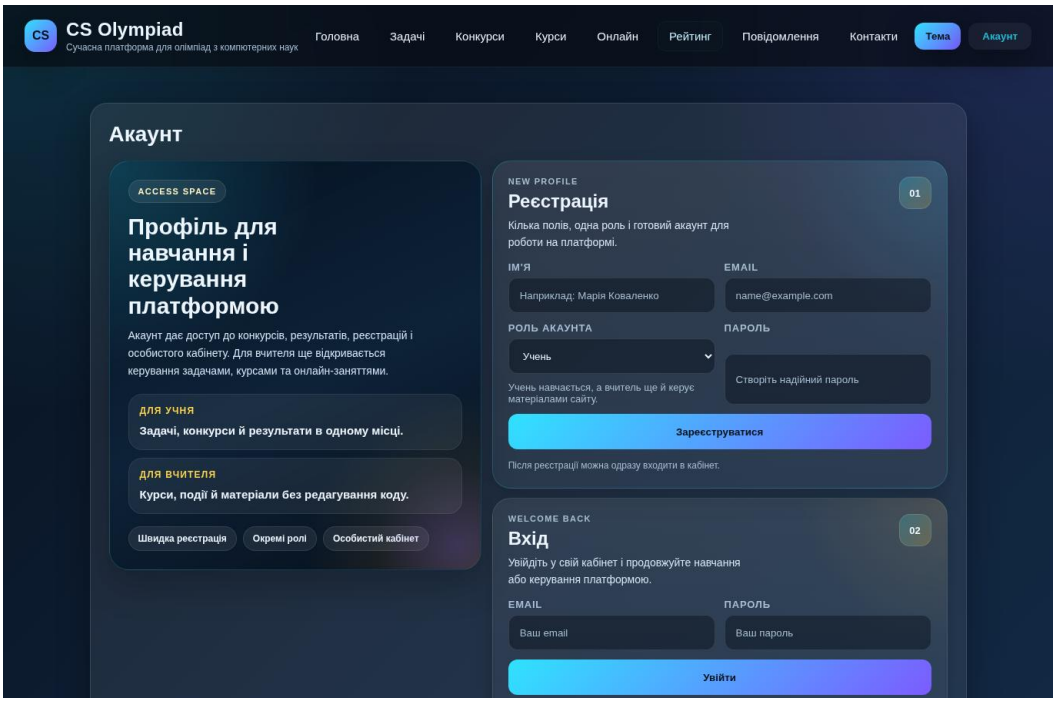


Рисунок 4.5 – Сторінка акаунта та форми авторизації

4.3 Аналіз роботи платформи на основі тестового використання

Після розробки основних модулів було проведено тестове використання платформи. Метою тестування була перевірка запуску сервера, відкриття сторінок, роботи навігації, реєстрації, входу, перегляду задач, проходження тестів і оновлення рейтингу.

Запуск сервера виконується командою node server.js або npm start. Якщо сервер працює правильно, у консолі з’являється повідомлення про запуск на

http://localhost:3000. Після цього користувач відкриває сайт у браузері та може переходити між сторінками.

Під час перевірки реєстрації користувач вводить ім’я, електронну пошту, пароль і роль. Сервер перевіряє, чи не існує користувач із такою поштою, хешує пароль і записує новий об’єкт у users.json. Під час входу сервер порівнює введений пароль із хешем і створює сесію.

Під час проходження тесту перевіряється робота таймера, прогресу, вибору відповідей і підрахунку результату. Після завершення правильні відповіді підсвічуються зеленим кольором, а неправильні – червоним. Результат зберігається в користувача й може відображатися у рейтингу.

Окремо перевірено роботу щоденного скидання рейтингу. У server.js реалізована функція resetScoresIfNeeded, яка порівнює дату в system.json із поточною датою. Якщо дата змінилася, бали користувачів обнуляються, а system.json оновлюється.

Таблиця 4.3 – Результати тестування функцій платформи

№	**Перевірена функція**	**Очікуваний результат**	**Фактичний результат**
1	Запуск сервера	Сервер відкриває сайт на localhost:3000	Виконано
2	Перемикання сторінок	Розділи відкриваються без перезавантаження	Виконано
3	Реєстрація користувача	Користувач додається до users.json	Виконано
4	Вхід користувача	Після успішної перевірки створюється сесія	Виконано
5	Перегляд задач	Задачі відображаються у відповідному розділі	Виконано

6	Проходження тесту	Система підраховує бали та показує правильні відповіді	Виконано
7	Оновлення рейтингу	Результат користувача оновлюється на сервері	Виконано
8	Повідомлення	Після тестування користувач отримує повідомлення	Виконано

Загалом тестування підтвердило працездатність основних модулів. Платформа демонструє повний цикл роботи навчального онлайн-конкурсу: від реєстрації користувача до проходження завдань, підрахунку балів і перегляду рейтингу.

Разом із тим у майбутньому доцільно вдосконалити систему зберігання даних, замінивши JSON-файли на базу даних. Також варто додати захищену панель адміністратора, автоматичну перевірку програмного коду, журнал дій користувачів і можливість експорту результатів у таблицю.

ВИСНОВКИ

У кваліфікаційному проєкті було розроблено веб-платформу для проведення онлайн олімпіад і конкурсів з комп'ютерних наук. Створена система поєднує навчальні матеріали, задачі, конкурси, тести, рейтинг, повідомлення та акаунт користувача.

У процесі виконання роботи проаналізовано актуальність онлайн-платформ для навчальних змагань, розглянуто існуючі сервіси, визначено вимоги до користувачів і сформовано структуру майбутнього веб-додатку.

Клієнтська частина реалізована у файлі index.html за допомогою HTML, CSS і JavaScript. Інтерфейс містить сучасне оформлення, картки, адаптивні блоки, навігаційне меню, перемикання теми, форми реєстрації й входу та сторінки основних модулів.

Серверна частина реалізована у файлі server.js на Node.js із використанням Express. Сервер обробляє маршрути реєстрації, авторизації, виходу, отримання користувачів, повідомлень, задач, конкурсів, курсів, онлайн-занять і оновлення результатів.

Для збереження даних у навчальному прототипі використано JSON-файли. Такий підхід є простим і зручним для демонстрації, хоча в майбутньому його доцільно замінити на повноцінну базу даних.

Результати тестового використання підтвердили, що платформа може виконувати основні функції: запускатися на локальному сервері, реєструвати користувачів, виконувати вхід, перемикати розділи, проводити тестування, оновлювати бали й формувати рейтинг.

Практична цінність роботи полягає в тому, що створений проєкт може використовуватися як основа для подальшого розвитку університетської платформи онлайн-олімпіад. До перспектив удосконалення належать

автоматична перевірка програмного коду, база даних, кабінет викладача, експорт результатів, сертифікати та розгортання на хостингу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Биков В. Ю. (2010). Інформаційні технології в освіті. Київ: Педагогічна думка. 230 с.
2. Морзе Н. В. (2012). Основи інформаційних технологій. Київ: Наукова думка. 180 с.
3. Полтавський університет економіки і торгівлі. Освітня програма «Комп'ютерні науки». ПУЕТ, 2025.
4. Node.js Documentation. Офіційна документація середовища виконання JavaScript на сервері. URL: https://nodejs.org/en/docs .
5. Express.js Documentation. Офіційна документація фреймворку Express для Node.js. URL: https://expressjs.com .
6. JSON Documentation. Опис формату зберігання та обміну даними JSON. URL: https://www.json.org/json-en.html.
7. Codeforces. Платформа для проведення змагань з програмування. URL: https://codeforces.com .
8. HackerRank. Платформа для навчання програмуванню та розв'язування задач. URL: https://www.hackerrank.com .
9. LeetCode. Платформа для практики задач з програмування. URL: https://leetcode.com .
10. AtCoder. Платформа для проведення програмних конкурсів. URL: https://atcoder.jp .
11. Eolymp. Освітня платформа для вивчення програмування та проведення олімпіад. URL: https://www.eolymp.com
12. MDN Web Docs. HTML: HyperText Markup Language. URL: https://developer.mozilla.org/en-US/docs/Web/HTML.
13. MDN Web Docs. CSS: Cascading Style Sheets. URL: https://developer.mozilla.org/en-US/docs/Web/CSS.

14. MDN Web Docs. JavaScript. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

ДОДАТОК А

ФРАГМЕНТИ КОДУ ПРОГРАМИ

У додатку наведено фрагменти програмного коду, взяті з проєкту cs-olympiad-platform. Для зручності великі файли подано не повністю, а у вигляді ключових фрагментів, які демонструють структуру, залежності, маршрути, авторизацію та логіку клієнтської частини.

А.1 Файл package.json

1 Файл package.json визначає назву проєкту, залежності та команду запуску сервера.

```
0001  {
0002    "name": "cs-olympiad-platform",
0003    "version": "1.0.0",
0004    "description": "",
0005    "main": "server.js",
0006    "scripts": {
0007      "test": "echo \"Error: no test specified\" && exit 1",
0008      "start": "node server.js"
0009    },
0010    "keywords": [],
0011    "author": "",
0012    "license": "ISC",
0013    "type": "commonjs",
0014    "dependencies": {
0015      "bcryptjs": "^3.0.3",
0016      "express": "^5.2.1",
0017      "express-session": "^1.19.0"
0018    }
0019  }
```

А.2 – Підключення модулів, дата рейтингу та стартові налаштування

1 **сервера**

На початку server.js підключаються модулі, задаються службові функції дати та налаштовується Express.

```
0001  const express = require("express");
0002  const fs = require("fs");
```

```
const bcrypt = require("bcryptjs");
const session = require("express-session");
const APP_TIME_ZONE = "Europe/Kiev";

function getDateKey(date = new Date()) {
  const safeDate = date instanceof Date ? date : new Date(date);

  if (Number.isNaN(safeDate.getTime())) {
    return "";
  }

  const parts = new Intl.DateTimeFormat("en-CA", {
    timeZone: APP_TIME_ZONE,
    year: "numeric",
    month: "2-digit",
    day: "2-digit"
  }).formatToParts(safeDate);

  const year = parts.find((part) => part.type === "year")?.value || "";
  const month = parts.find((part) => part.type === "month")?.value || "";
  const day = parts.find((part) => part.type === "day")?.value || "";

  return year && month && day ? `${year}-${month}-${day}` : "";
}

function getToday(){
  return getDateKey(new Date());
}

function getTodayLabel(now = new Date()) {
  return new Intl.DateTimeFormat("uk-UA", {
    timeZone: APP_TIME_ZONE,
    day: "numeric",
    month: "long"
  }).format(now);
}

function getVisitDateKey(value) {
  const parsedDate = new Date(String(value || "").trim());
  return Number.isNaN(parsedDate.getTime()) ? "" : getDateKey(parsedDate);
}

function hasUserVisitedToday(user, todayKey = getToday()) {
  return Boolean(todayKey) && getVisitDateKey(user?.lastVisitAt) ===
todayKey;
}
```

```
0048
0049  function touchUserVisit(user, now = new Date()) {
0050    const nextVisitAt = now.toISOString();
0051
0052    if (String(user?.lastVisitAt || "") === nextVisitAt) {
0053      return false;
0054    }
0055
0056    user.lastVisitAt = nextVisitAt;
0057    return true;
0058  }
0059
0060  function resetScoresIfNeeded(){
0061    if(!fs.existsSync("system.json")){
0062      fs.writeFileSync("system.json", JSON.stringify({ lastReset: getToday() }));
0063    }
0064
0065    const system = JSON.parse(fs.readFileSync("system.json"));
0066    const today = getToday();
0067
0068    if(system.lastReset !== today){
0069
0070      let users = JSON.parse(fs.readFileSync("users.json"));
0071
0072      users = users.map(user => ({
0073        ...user,
0074        score: 0
0075      }));
0076
0077      fs.writeFileSync("users.json", JSON.stringify(users, null, 2));
0078      fs.writeFileSync("system.json", JSON.stringify({ lastReset: today }));
0079
0080      console.log("Щоденний рейтинг скинуто");
0081    }
0082  }
0083  const path = require("path");
0084
0085  const app = express();
0086  const PORT = 3000;
0087  const USERS_FILE = path.join(__dirname, "users.json");
0088  const PROBLEMS_FILE = path.join(__dirname, "problems.json");
0089  const ONLINE_FILE = path.join(__dirname, "online.json");
0090  const CONTESTS_FILE = path.join(__dirname, "contests.json");
0091  const COURSES_FILE = path.join(__dirname, "courses.json");
0092
0093  app.use(express.json());
```

```
0094  app.use(express.static(__dirname));
0095  app.use(session({
0096    secret: "cs-olympiad-secret-key",
0097    resave: false,
0098    saveUninitialized: false,
0099    cookie: {
0100      maxAge: 1000 * 60 * 60 * 24
0101    }
0102  }));
0103
0104  function readUsers() {
```

А.3 – Робота з користувачами та очищення рейтингу

Фрагмент містить функції читання/запису users.json і службове очищення рейтингових даних користувача.

```
0104  function readUsers() {
0105    if (!fs.existsSync(USERS_FILE)) {
0106      fs.writeFileSync(USERS_FILE, "[]", "utf8");
0107    }
0108    return JSON.parse(fs.readFileSync(USERS_FILE, "utf8"));
0109  }
0110
0111  function writeUsers(users) {
0112    fs.writeFileSync(USERS_FILE, JSON.stringify(users, null, 2), "utf8");
0113  }
0114
0115  function clearUserRating(user) {
0116    let changed = false;
0117
0118    if (Number(user?.score) !== 0) {
0119      user.score = 0;
0120      changed = true;
0121    }
0122
0123    if (String(user?.lastResult || "").trim()) {
0124      user.lastResult = "";
0125      changed = true;
0126    }
0127
0128    if (String(user?.lastScoreContest || "").trim()) {
0129      user.lastScoreContest = "";
0130      changed = true;
0131    }
0132
```

```
  return changed;
}

function decodeBasicHtmlEntities(value) {
  return String(value || "")
    .replace(/ /gi, " ")
    .replace(/&lt;/gi, "<")
    .replace(/&gt;/gi, ">")
    .replace(/&quot;/gi, "\"")
    .replace(/&#39;/gi, "'")
    .replace(/&amp;/gi, "&");
}

function stripProblemMarkup(value) {
  return decodeBasicHtmlEntities(String(value || ""))
    .replace(/<\/p>\s*<p>/gi, "\n\n")
    .replace(/<br\s*\/?>/gi, "\n")
    .replace(/<\/li>\s*<li>/gi, "\n")
    .replace(/<li[^>]*>/gi, "- ")
    .replace(/<\/(p|ul|ol|li)>/gi, "\n")
    .replace(/<[^>]+>/g, "")
    .replace(/\r/g, "")
    .replace(/\n{3,}/g, "\n\n")
    .trim();
}

function normalizeProblemRecord(problem) {
  return {
    id: Number(problem?.id) || 0,
    title: String(problem?.title || "").trim(),
    topic: String(problem?.topic || "").trim(),
    difficulty: String(problem?.difficulty || "").trim(),
    text: stripProblemMarkup(problem?.text),
    input: stripProblemMarkup(problem?.input),
    output: stripProblemMarkup(problem?.output),
    constraints: stripProblemMarkup(problem?.constraints),
    example: stripProblemMarkup(problem?.example),
    authorName: String(problem?.authorName || "CS Olympiad
Platform").trim(),
    createdAt: String(problem?.createdAt || new Date().toISOString())
  };
}

function defaultProblems() {
```

А.4 – Підготовка питань конкурсу та підрахунок результату

Код нормалізує питання конкурсу, перевіряє варіанти відповідей і формує результат учасника.

```
0451  function normalizeContestQuestions(questions) {
0452    if (!Array.isArray(questions)) {
0453      return [];
0454    }
0455
0456    return questions
0457      .map((question) => {
0458        const text = String(question?.text || "").trim();
0459        const options = Array.isArray(question?.options)
0460          ? question.options.map((option) => String(option ||
"").trim()).filter(Boolean)
0461          : [];
0462        const correctIndex = Number(question?.correctIndex);
0463
0464        if (!text || options.length < 2) {
0465          return null;
0466        }
0467
0468        if (!Number.isInteger(correctIndex) || correctIndex < 0 || correctIndex >=
options.length) {
0469          return null;
0470        }
0471
0472        return {
0473          text,
0474          options,
0475          correctIndex
0476        };
0477      })
0478      .filter(Boolean);
0479  }
0480
0481  function calculateContestResult(questions, answers) {
0482    const normalizedAnswers = Array.isArray(answers) ? answers : [];
0483
0484    let score = 0;
0485    const review = questions.map((question, index) => {
0486      const selectedIndex = Number(normalizedAnswers[index]);
0487      const safeSelectedIndex =
0488        Number.isInteger(selectedIndex) &&
0489        selectedIndex >= 0 &&
```

```
      selectedIndex < question.options.length
        ? selectedIndex
        : null;
    const isCorrect = safeSelectedIndex === question.correctIndex;

    if (isCorrect) {
      score++;
    }

    return {
      selectedIndex: safeSelectedIndex,
      correctIndex: question.correctIndex,
      isCorrect
    };
  });

  return { score, review };
}

function isContestFinishedStatus(value) {
  const normalized = String(value || "").toLowerCase();
  return normalized.includes("успішно завершено") || normalized === "завершено";
}

function stripRecordOwnerMeta(record, ownerEmailFieldName) {
  if (!record || typeof record !== "object") {
    return record;
  }

  const clone = { ...record };
  delete clone[ownerEmailFieldName];
  return clone;
}

function canManageOwnedRecord(record, user, ownerFieldName, ownerEmailFieldName) {
  const recordOwnerEmail = String(record?.[ownerEmailFieldName] || "").trim().toLowerCase();
  const userEmail = String(user?.email || "").trim().toLowerCase();

  if (recordOwnerEmail && userEmail) {
    return recordOwnerEmail === userEmail;
  }

```

```
0532    return String(record?.[ownerFieldName] || "").trim() === String(user?.name || "").trim();
0533  }
0534
      0535          function          syncUserContestStatuses(user,          contests          =
readJsonCollection(CONTESTS_FILE)) {
```

А.5 – Реєстрація, авторизація та вихід користувача

Маршрути /register, /login і /logout забезпечують роботу акаунта користувача.

```
0583  function getSessionUser(req) {
0584    const users = readUsers();
0585    return users.find((u) => u.email === req.session.userEmail) || null;
0586  }
0587
0588  app.post("/register", async (req, res) => {
0589    try {
0590      const { name, email, password, role } = req.body;
0591
0592      if (!name || !email || !password) {
0593        return res.json({ success: false, message: "Заповніть всі поля" });
0594      }
0595
0596      const cleanName = String(name).trim();
0597      const cleanEmail = String(email).trim().toLowerCase();
0598      const cleanPassword = String(password).trim();
0599      const cleanRole = role === "teacher" ? "teacher" : "student";
0600
0601      if (cleanPassword.length < 6) {
0602        return res.json({
0603          success: false,
0604          message: "Пароль має містити мінімум 6 символів"
0605        });
0606      }
0607
0608      const users = readUsers();
0609      const exists = users.find((u) => u.email.toLowerCase() === cleanEmail);
0610
0611      if (exists) {
0612        return res.json({ success: false, message: "Користувач уже існує" });
0613      }
0614
```

```
0615     const hashedPassword = await bcrypt.hash(cleanPassword, 10);
0616
0617     users.push({
0618       name: cleanName,
0619       email: cleanEmail,
0620       password: hashedPassword,
0621       role: cleanRole,
0622       score: 0,
0623       lastResult: "",
0624       lastScoreContest: "",
0625       activeContest: "",
0626       lastVisitAt: "",
0627       messages: [],
0628       myContests: []
0629     });
0630     writeUsers(users);
0631
0632     res.json({ success: true, message: "Реєстрація успішна" });
0633   } catch (error) {
0634     console.error("Помилка реєстрації:", error);
0635     res.status(500).json({ success: false, message: "Помилка сервера" });
0636   }
0637 });
0638
0639 app.post("/login", async (req, res) => {
0640   try {
0641     resetScoresIfNeeded();
0642     const { email, password } = req.body;
0643
0644     const cleanEmail = String(email || "").trim().toLowerCase();
0645     const cleanPassword = String(password || "").trim();
0646
0647     const users = readUsers();
0648     const user = users.find((u) => u.email.toLowerCase() === cleanEmail);
0649
0650     if (!user) {
0651       return res.json({
0652         success: false,
0653         message: "Неправильний email або пароль"
0654       });
0655     }
0656
0657     const passwordMatch = await bcrypt.compare(cleanPassword,
user.password);
0658
0659     if (!passwordMatch) {
```

```
0660       return res.json({
0661         success: false,
0662         message: "Неправильний email або пароль"
0663       });
0664     }
0665
0666     touchUserVisit(user);
0667     writeUsers(users);
0668     req.session.userEmail = user.email;
0669
0670     return res.json({
0671       success: true,
0672       name: user.name,
0673       email: user.email,
0674       role: user.role || "student"
0675     });
0676   } catch (error) {
0677     console.error("Помилка входу:", error);
0678     return res.status(500).json({
0679       success: false,
0680       message: "Помилка сервера"
0681     });
0682   }
0683 });
0684
       0685 app.post("/logout", (req, res) => {
```

A.6 – Серверні маршрути для задач та онлайн-занять

Маршрути використовуються для отримання й додавання задач та онлайн-занять.

```
0787 app.get("/problems", (req, res) => {
0788   const problems = readProblems().map((problem) =>
stripRecordOwnerMeta(problem, "authorEmail"));
0789   res.json({
0790     success: true,
0791     problems
0792   });
0793 });
0794
0795 app.post("/problems", (req, res) => {
0796   const user = getSessionUser(req);
```

```
0797
0798    if (!user) {
0799      return res.status(401).json({ success: false, message: "Увійдіть в акаунт"
});
0800    }
0801
0802    if ((user.role || "student") !== "teacher") {
0803      return res.status(403).json({ success: false, message: "Лише вчитель може
додавати задачі" });
0804    }
0805
0806    const {
0807      title,
0808      topic,
0809      difficulty,
0810      text,
0811      input,
0812      output,
0813      constraints,
0814      example
0815    } = req.body;
0816
0817    const requiredFields = [title, topic, difficulty, text, input, output, constraints,
example];
0818    if (requiredFields.some((item) => !String(item || "").trim())) {
0819      return res.json({ success: false, message: "Заповніть усі поля задачі" });
0820    }
0821
0822    const problems = readProblems();
0823    const nextId = problems.length ? Math.max(...problems.map((item) =>
Number(item.id) || 0)) + 1 : 1;
0824
0825    const newProblem = {
0826      id: nextId,
0827      title: String(title).trim(),
0828      topic: String(topic).trim(),
0829      difficulty: String(difficulty).trim(),
0830      text: String(text).trim(),
0831      input: String(input).trim(),
0832      output: String(output).trim(),
0833      constraints: String(constraints).trim(),
0834      example: String(example).trim(),
0835      authorName: user.name,
0836      authorEmail: user.email,
0837      createdAt: new Date().toISOString()
0838    };
```

```
0839
0840    problems.push(newProblem);
0841    writeProblems(problems);
0842
0843    return res.json({
0844      success: true,
0845      message: "Задачу додано",
0846      problem: stripRecordOwnerMeta(newProblem, "authorEmail")
0847    });
0848  });
0849
0850  app.get("/online-lessons", (req, res) => {
0851    const lessons = readOnlineLessons().map((lesson) =>
stripRecordOwnerMeta(lesson, "teacherEmail"));
0852    res.json({
0853      success: true,
0854      lessons
0855    });
0856  });
0857
0858  app.post("/online-lessons", (req, res) => {
0859    const user = getSessionUser(req);
0860
0861    if (!user) {
0862      return res.status(401).json({ success: false, message: "Увійдіть в акаунт"
});
0863    }
0864
0865    if ((user.role || "student") !== "teacher") {
0866      return res.status(403).json({ success: false, message: "Лише вчитель може
створювати онлайн-заняття" });
0867    }
0868
0869    const {
0870      title,
0871      description,
0872      date,
0873      time,
0874      platform,
0875      link,
0876      details
0877    } = req.body;
0878
0879    const requiredFields = [title, description, date, time, platform, link, details];
0880    if (requiredFields.some((item) => !String(item || "").trim())) {
```

```
0881      return res.json({ success: false, message: "Заповніть усі поля онлайн-заняття" });
0882    }
0883
0884    const lessons = readOnlineLessons();
0885    const nextId = lessons.length ? Math.max(...lessons.map((item) => Number(item.id) || 0)) + 1 : 1;
0886
0887    const lesson = {
0888      id: nextId,
0889      title: String(title).trim(),
0890      description: String(description).trim(),
0891      date: String(date).trim(),
0892      time: String(time).trim(),
0893      platform: String(platform).trim(),
0894      link: String(link).trim(),
0895      details: String(details).trim(),
0896      teacherName: user.name,
0897      teacherEmail: user.email,
0898      createdAt: new Date().toISOString()
0899    };
0900
0901    lessons.push(lesson);
0902    lessons.sort((a, b) => {
0903      const aDate = new Date(`${a.date}T${a.time}`);
0904      const bDate = new Date(`${b.date}T${b.time}`);
0905      return aDate - bDate;
0906    });
0907    writeOnlineLessons(lessons);
0908
0909    return res.json({
0910      success: true,
0911      message: "Онлайн-заняття створено",
0912      lesson: stripRecordOwnerMeta(lesson, "teacherEmail")
0913    });
0914  });
0915
      0916  app.get("/contests", (req, res) => {
```

А.7 – Серверні маршрути для конкурсів

Фрагмент відповідає за отримання, створення та видалення конкурсів.

```
0916  app.get("/contests", (req, res) => {
```

```
0917    const contests =
sortContestsBySchedule(readJsonCollection(CONTESTS_FILE)).map((contest) =>
0918      decorateContest(contest)
0919    );
0920    res.json({ success: true, contests });
0921  });
0922
0923  app.post("/contests", (req, res) => {
0924    const user = getSessionUser(req);
0925    if (!user) {
0926      return res.status(401).json({ success: false, message:
"\u0423\u0432\u0456\u0439\u0434\u0456\u0442\u044c \u0432
      \u0430\u043a\u0430\u0443\u043d\u0442" });
0927    }
0928    if ((user.role || "student") !== "teacher") {
0929      return res.status(403).json({ success: false, message:
"\u041b\u0438\u0448\u0435 \u0432\u0447\u0438\u0442\u0435\u043b\u044c
      \u043c\u043e\u0436\u0435
\u0434\u043e\u0434\u0430\u0432\u0430\u0442\u0438
\u043a\u043e\u043d\u043a\u0443\u0440\u0441\u0438" });
0930    }
0931
0932    const { title, tag, description, date, time, format, details, questions } =
req.body;
0933
0934    const detailList = Array.isArray(details)
0935      ? details.map((item) => String(item || "").trim()).filter(Boolean)
0936      : [];
0937    const questionList = normalizeContestQuestions(questions);
0938
0939    const requiredFields = [title, tag, description, date, time, format];
0940    if (requiredFields.some((item) => !String(item || "").trim()) ||
!detailList.length || !questionList.length) {
0941      return res.json({ success: false, message:
"\u0417\u0430\u043f\u043e\u0432\u043d\u0456\u0442\u044c \u0443\u0441\u0456
      \u043f\u043e\u043b\u044f
\u043a\u043e\u043d\u043a\u0443\u0440\u0441\u0443" });
0942    }
0943
0944    const parsedStart = parseContestDateTime(date, time);
0945    if (!parsedStart) {
0946      return res.json({ success: false, message: "Вкажіть коректні дату і час
конкурсу" });
0947    }
0948
0949    const contests = readJsonCollection(CONTESTS_FILE);
```

```
0950    const nextId = contests.length ? Math.max(...contests.map((item) =>
Number(item.id) || 0)) + 1 : 1;
0951
0952    const contest = {
0953      id: nextId,
0954      title: String(title).trim(),
0955      tag: String(tag).trim(),
0956      description: String(description).trim(),
0957      date: String(date).trim(),
0958      time: String(time).trim(),
0959      format: String(format).trim(),
0960      details: detailList,
0961      questions: questionList,
0962      authorName: user.name,
0963      authorEmail: user.email,
0964      createdAt: new Date().toISOString()
0965    };
0966
0967    const contestWithState = decorateContest(contest);
0968    contests.push(contest);
0969    writeJsonCollection(CONTESTS_FILE, sortContestsBySchedule(contests));
0970
0971    res.json({ success: true, message:
"\u041a\u043e\u043d\u043a\u0443\u0440\u0441
\u0434\u043e\u0434\u0430\u043d\u043e", contest:
      contestWithState });
0972  });
0973
0974  app.delete("/contests/:id", (req, res) => {
0975    const user = getSessionUser(req);
0976    if (!user) {
0977      return res.status(401).json({ success: false, message: "Увійдіть в акаунт"
});
0978    }
0979    if ((user.role || "student") !== "teacher") {
0980      return res.status(403).json({ success: false, message: "Лише вчитель може
видаляти конкурси" });
0981    }
0982
0983    const contests = readJsonCollection(CONTESTS_FILE);
0984    const targetContest = contests.find((item) => Number(item.id) ===
Number(req.params.id));
0985
0986    if (!targetContest) {
0987      return res.status(404).json({ success: false, message: "Конкурс не
знайдено" });
```

```
0988    }
0989
0990    if (!canManageOwnedRecord(targetContest, user, "authorName",
"authorEmail")) {
0991      return res.status(403).json({ success: false, message: "Можна видаляти
лише власні конкурси" });
0992    }
0993
0994    const filtered = contests.filter((item) => Number(item.id) !==
Number(req.params.id));
0995    writeJsonCollection(CONTESTS_FILE, filtered);
0996
0997    const validContestTitles = getContestTitleSet(filtered);
0998    const users = readUsers();
0999    let usersChanged = false;
1000
1001    users.forEach((item) => {
1002      if (syncUserContestsWithCatalog(item, validContestTitles)) {
1003        usersChanged = true;
1004      }
1005    });
1006
1007    if (usersChanged) {
1008      writeUsers(users);
1009    }
1010
1011    res.json({ success: true, message: "Конкурс видалено" });
1012  });
1013
        1014  app.get("/courses", (req, res) => {
```

А.8 – Оновлення рейтингу, старт і завершення конкурсу

Код зберігає результат користувача, реєструє його на конкурс, фіксує старт і завершення участі.

```
1149  app.post("/update-score", (req, res) => {
1150    if (!req.session.userEmail) {
1151      return res.status(401).json({
1152        success: false,
1153        message: "Потрібно увійти в акаунт"
1154      });
1155    }
```

```
1156
1157    return res.status(403).json({
1158      success: false,
1159      message: "Пряме оновлення балів вимкнено. Результат зберігається
лише після серверної перевірки конкурсу."
1160    });
1161  });
1162  app.post("/contest/register", (req, res) => {
1163    if (!req.session.userEmail) {
1164      return res.status(401).json({ success: false, message: "Увійдіть в акаунт"
});
1165    }
1166
1167    const cleanContestName = String(req.body?.contestName || "").trim();
1168    const contests = readJsonCollection(CONTESTS_FILE);
1169    const validContestTitles = getContestTitleSet(contests);
1170    const contest = contests.find((item) => String(item?.title || "").trim() ===
cleanContestName);
1171
1172    if (!contest || !validContestTitles.has(cleanContestName)) {
1173      return res.json({ success: false, message: "Цей конкурс уже недоступний"
});
1174    }
1175
1176    const questionList = normalizeContestQuestions(contest.questions);
1177    if (!questionList.length) {
1178      return res.json({ success: false, message: "Для цього конкурсу ще не
додано тест" });
1179    }
1180
1181    const lifecycle = getContestLifecycle(contest);
1182    if (lifecycle.isClosed) {
1183      return res.json({ success: false, message: "Цей конкурс уже завершено"
});
1184    }
1185
1186    let users = readUsers();
1187    const user = users.find(u => u.email === req.session.userEmail);
1188
1189    if (!user) {
1190      return res.status(404).json({ success: false, message: "Користувача не
знайдено" });
1191    }
1192
1193    syncUserContestsWithCatalog(user, validContestTitles);
1194    syncUserContestStatuses(user, contests);
```

```
1195
1196    const registrationStatus = lifecycle.isOpen
1197      ? "Можна починати"
1198      : "Зареєстровано. Очікує старту";
1199
1200    const exists = user.myContests.find(item => item.contest ===
cleanContestName);
1201
1202    if (exists && (exists.certificate || isContestFinishedStatus(exists.status))) {
1203      return res.json({ success: false, message: "Ви вже завершили цей конкурс"
});
1204    }
1205
1206    if (!exists) {
1207      user.myContests.push({
1208        contest: cleanContestName,
1209        status: registrationStatus,
1210        certificate: false
1211      });
1212    } else if (!exists.certificate && !String(exists.status ||
"").toLowerCase().includes("заверш")) {
1213      exists.status = registrationStatus;
1214    }
1215
1216    writeUsers(users);
1217
1218    res.json({
1219      success: true,
1220      message: lifecycle.isOpen
1221        ? "Конкурс уже активний. Його можна одразу відкрити."
1222        : `Ви зареєструвалися. Старт конкурсу: ${contest.date} о
${contest.time}.`
1223    });
1224  });
1225
1226  app.post("/contest/start", (req, res) => {
1227    if (!req.session.userEmail) {
1228      return res.status(401).json({ success: false, message: "Увійдіть в акаунт"
});
1229    }
1230
1231    const cleanContestName = String(req.body?.contestName || "").trim();
1232    const contests = readJsonCollection(CONTESTS_FILE);
1233    const validContestTitles = getContestTitleSet(contests);
1234    const contest = contests.find((item) => String(item?.title || "").trim() ===
cleanContestName);
```

```
1235
1236    if (!contest || !validContestTitles.has(cleanContestName)) {
1237      return res.json({ success: false, message: "Цей конкурс уже недоступний"
});
1238    }
1239
1240    const questionList = normalizeContestQuestions(contest.questions);
1241    if (!questionList.length) {
1242      return res.json({ success: false, message: "Для цього конкурсу ще не
додано тест" });
1243    }
1244
1245    const lifecycle = getContestLifecycle(contest);
1246
1247    if (lifecycle.isUpcoming) {
1248      return res.json({
1249        success: false,
1250        message: `Конкурс відкриється ${contest.date} о ${contest.time}.`
1251      });
1252    }
1253
1254    if (lifecycle.isClosed) {
1255      return res.json({ success: false, message: "Час цього конкурсу вже минув"
});
1256    }
1257
1258    let users = readUsers();
1259    const user = users.find((item) => item.email === req.session.userEmail);
1260
1261    if (!user) {
1262      return res.status(404).json({ success: false, message: "Користувача не
знайдено" });
1263    }
1264
1265    syncUserContestsWithCatalog(user, validContestTitles);
1266    syncUserContestStatuses(user, contests);
1267
1268    if (!Array.isArray(user.myContests)) {
1269      user.myContests = [];
1270    }
1271
1272    const existingContest = user.myContests.find((item) => item.contest ===
cleanContestName);
1273
1274    if (existingContest && (existingContest.certificate ||
isContestFinishedStatus(existingContest.status))) {
```

```
1275      return res.json({ success: false, message: "Цей конкурс уже завершено для вашого акаунта" });
1276    }
1277
1278    if (!existingContest) {
1279      user.myContests.push({
1280        contest: cleanContestName,
1281        status: "Йде зараз",
1282        certificate: false
1283      });
1284    } else if (!existingContest.certificate) {
1285      existingContest.status = "Йде зараз";
1286    }
1287
1288    user.activeContest = cleanContestName;
1289    writeUsers(users);
1290
1291    res.json({
1292      success: true,
1293      message: `Конкурс "${cleanContestName}" уже відкритий.`
1294    });
1295  });
1296  app.post("/contest/finish", (req, res) => {
1297    if (!req.session.userEmail) {
1298      return res.status(401).json({ success: false, message: "Увійдіть в акаунт" });
1299    }
1300
1301    const { contestName, answers } = req.body;
1302
1303    let users = readUsers();
1304    const user = users.find(u => u.email === req.session.userEmail);
1305
1306    if (!user) {
1307      return res.status(404).json({ success: false, message: "Користувача не знайдено" });
1308    }
1309
1310    if (!user.myContests) user.myContests = [];
1311    if (!user.messages) user.messages = [];
1312
1313    const activeContest = user.activeContest;
1314    const contests = readJsonCollection(CONTESTS_FILE);
1315    const activeContestName = String(activeContest || "").trim();
1316    const requestedContestName = String(contestName || "").trim();
1317
```

```
1318    if (!activeContestName || requestedContestName !== activeContestName) {
1319      return res.json({ success: false, message: "Активний конкурс не знайдено"
});
1320    }
1321
1322    const activeContestData = contests.find((item) => String(item?.title ||
"").trim() === activeContestName);
1323    const questionList =
normalizeContestQuestions(activeContestData?.questions);
1324    const totalQuestions = questionList.length;
1325
1326    if (!totalQuestions) {
1327      return res.json({ success: false, message: "Активний конкурс не знайдено"
});
1328    }
1329
1330    const result = calculateContestResult(questionList, answers);
1331    const score = result.score;
1332    const certificateThreshold = Math.max(1, Math.ceil(totalQuestions * 0.8));
1333    const hasCertificate = score >= certificateThreshold;
1334
1335    if (activeContest) {
1336      user.myContests = user.myContests.map(item => {
1337        if (item.contest === activeContest) {
1338          return {
1339            ...item,
1340            status: hasCertificate ? "Успішно завершено" : "Завершено",
1341            certificate: hasCertificate
1342          };
1343        }
1344        return item;
1345      });
1346    }
1347
1348    user.activeContest = "";
1349    user.score = score;
1350    user.lastResult = `Ваш результат: ${score} / ${totalQuestions}`;
1351    user.lastScoreContest = activeContestName;
1352    user.messages.unshift({
1353      text: `Ви завершили тест. Результат: ${score} з ${totalQuestions}.`,
1354      date: new Date().toLocaleString()
1355    });
1356
1357    if (hasCertificate) {
1358      user.messages.unshift({
1359        text: "Вітаємо! Ви отримали право на грамоту.",
```

```
1360       date: new Date().toLocaleString()
1361     });
1362   }
1363
1364   writeUsers(users);
1365
1366   res.json({
1367     success: true,
1368     score,
1369     totalQuestions,
1370     hasCertificate,
1371     review: result.review
1372   });
1373 });
1374
1375 app.listen(PORT, () => {
1376   console.log(`Server running: http://localhost:${PORT}`);
1377 });
```

А.9 – CSS-змінні та оформлення теми сайту

Фрагмент визначає кольори, змінні темної та світлої теми, базовий вигляд сторінки.

```
0039 :root{
0040   --bg-1:#07111f;
0041   --bg-2:#0f1f35;
0042   --bg-3:#122b46;
0043   --card:rgba(255,255,255,0.08);
0044   --card-border:rgba(255,255,255,0.12);
0045   --text:#eef6ff;
0046   --text-soft:#d6e7f8;
0047   --muted:#a9bfd4;
0048   --accent:#30e3ff;
0049   --accent-2:#7c5cff;
0050   --success:#35d07f;
0051   --danger:#ff6b6b;
0052   --warning:#ffbf47;
0053   --shadow:0 18px 40px rgba(0,0,0,0.35);
0054 }
0055
0056 body.light{
0057   --bg-1:#eef6ff;
0058   --bg-2:#dcecff;
```

```
  --bg-3:#c8e0ff;
  --card:rgba(255,255,255,0.7);
  --card-border:rgba(0,0,0,0.08);
  --text:#0d1b2a;
  --text-soft:#274158;
  --muted:#4e647a;
  --accent:#0ea5e9;
  --accent-2:#6d28d9;
  --success:#16a34a;
  --danger:#dc2626;
  --warning:#d97706;
  --shadow:0 14px 30px rgba(0,0,0,0.12);
}

*{
  margin:0;
  padding:0;
  box-sizing:border-box;
  font-family:Segoe UI, sans-serif;
}

html{
  scroll-behavior:smooth;
  min-height:100%;
  background:
    radial-gradient(circle at top left, rgba(48,227,255,0.14), transparent 30%),
    radial-gradient(circle at top right, rgba(124,92,255,0.15), transparent 28%),
    linear-gradient(135deg,var(--bg-1),var(--bg-2),var(--bg-3));
}

html.light-root{
  background:
    radial-gradient(circle at top left, rgba(14,165,233,0.12), transparent 30%),
    radial-gradient(circle at top right, rgba(109,40,217,0.12), transparent 28%),
    linear-gradient(135deg,#eef6ff,#dcecff,#c8e0ff);
}

body{
  background:
    radial-gradient(circle at top left, rgba(48,227,255,0.14), transparent 30%),
    radial-gradient(circle at top right, rgba(124,92,255,0.15), transparent 28%),
    linear-gradient(135deg,var(--bg-1),var(--bg-2),var(--bg-3));
  color:var(--text);
  min-height:100vh;
  transition:background .3s ease,color .3s ease;
}
```

```
0105
0106  header{
0107    position:sticky;
0108    top:0;
0109    z-index:50;
0110    display:flex;
0111    justify-content:space-between;
0112    align-items:center;
0113    padding:18px 28px;
0114    background:rgba(8,12,20,0.65);
0115    backdrop-filter:blur(10px);
0116    border-bottom:1px solid rgba(255,255,255,0.08);
0117  }
0118
0119  body.light header{
        0120    background:rgba(255,255,255,0.7);
```

А.10 – HTML-структура шапки сайту та головної сторінки

У коді описано верхню навігацію, кнопку акаунта, перемикач теми та головний екран.

```
4021  <body>
4022
4023  <header>
4024    <div class="logo-wrap">
4025      <div class="logo-badge">CS</div>
4026      <div>
4027        <div class="logo">CS Olympiad</div>
4028        <div class="logo-sub">Сучасна платформа для олімпіад з компютерних
наук</div>
4029      </div>
4030    </div>
4031
4032    <nav>
4033      <button class="active-nav" data-page="home" onclick="openPage('home',
this)">Головна</button>
4034      <button data-page="problems" onclick="openPage('problems',
this)">Задачі</button>
4035      <button data-page="contest" onclick="openPage('contest',
this)">Конкурси</button>
4036      <button data-page="courses" onclick="openPage('courses',
this)">Курси</button>
```

```
4037      <button data-page="online" onclick="openPage('online',
this)">Онлайн</button>
4038      <button data-page="rating" onclick="openPage('rating',
this)">Рейтинг</button>
4039      <button data-page="messages" onclick="openPage('messages',
this)">Повідомлення</button>
4040      <button data-page="contacts" onclick="openPage('contacts',
this)">Контакти</button>
4041    </nav>
4042
4043    <div class="top-actions">
4044      <button class="icon-btn" onclick="toggleTheme()">Тема</button>
4045      <button class="account-btn" data-page="account"
onclick="openPage('account', this)">Акаунт</button>
4046    </div>
4047  </header>
4048
4049  <div class="wrapper">
4050
4051    <div id="home" class="page active">
4052      <div class="home-shell">
4053        <div class="home-hero-board">
4054          <div class="home-hero-main">
4055            <div>
4056              <div class="home-kicker-row">
4057                <div class="badge">Olympiad Command Center</div>
4058                <div class="home-live-pill">
4059                  <span class="home-live-dot"></span>
4060                  <span id="homeLivePill">Платформа готова до нових
стартів</span>
4061                </div>
4062              </div>
4063
4064              <h1>Простір, де підготовка до олімпіад виглядає живо, чітко і
сучасно.</h1>
4065              <p class="home-hero-copy" id="homeWelcomeLine">
4066                Тут зібрані задачі, конкурси, курси та онлайн-заняття в одному
ритмі. Обирайте формат, тримайте темп і переходьте від
      тренування до справжніх стартів без зайвого хаосу.
4067              </p>
4068
4069              <div class="home-action-row">
4070                <button class="main-btn"
onclick="startDemoContest()">Спробувати демо-конкурс</button>
4071                <button class="secondary-btn" onclick="openPage('problems',
null)">Перейти до задач</button>
```

```
4072              <button class="secondary-btn" onclick="openPage('courses',
null)">Відкрити курси</button>
4073            </div>
4074          </div>
4075
4076          <div class="home-signal-row">
4077            <div class="home-signal">
4078              <span>Задачі</span>
4079              <strong id="homeProblemsCount">0</strong>
4080              <p>Для щоденного тренування і відпрацювання тем.</p>
4081            </div>
4082            <div class="home-signal">
4083              <span>Конкурси</span>
4084              <strong id="homeContestsCount">0</strong>
4085              <p>Запуски з фіксованим часом, рейтингом і результатами.</p>
4086            </div>
4087            <div class="home-signal">
4088              <span>Курси</span>
4089              <strong id="homeCoursesCount">0</strong>
4090              <p>Навчальні треки з програмою та окремими маршрутами.</p>
4091            </div>
4092            <div class="home-signal">
4093              <span>Онлайн</span>
4094              <strong id="homeOnlineCount">0</strong>
4095              <p>Живі зустрічі, розбори та заняття у вказаний час.</p>
4096            </div>
4097          </div>
4098        </div>
4099
4100        <div class="home-side-rail">
4101          <div class="home-spotlight-card">
4102            <div class="home-panel-label">Найближчий фокус</div>
4103            <h3 id="homeNextContestTitle">Конкурси ще готуються</h3>
4104            <p id="homeNextContestMeta">Щойно вчитель додасть конкурс із
тестом і часом старту, він з’явиться тут як головна подія
    платформи.</p>
4105
4106            <div class="home-mini-pills">
4107              <div class="home-mini-pill" id="homeNextContestState">Очікує
наповнення</div>
4108              <div class="home-mini-pill" id="homeQuestionCount">0
питань</div>
```

А.11 – Розділи конкурсів, тестування та рейтингу

1 Фрагмент створює основні блоки для конкурсів, тестів і відображення рейтингу.

```
4261    <div id="contest" class="page">
4262    <div class="card">
4263      <div class="section-head">
4264        <div>
4265          <h2>Олімпіадні конкурси</h2>
4266          <p>Оберіть конкурс, зареєструйтесь та візьміть участь у визначений
час.</p>
4267        </div>
4268        <input class="search" id="contestSearch" placeholder="Пошук
конкурсу..." oninput="filterContests()">
4269      </div>
4270
4271      <div class="contest-grid" id="contestGrid">
4272  </div>
4273    </div>
4274
4275    <div class="card">
4276      <h2>Мої реєстрації</h2>
4277      <p>Тут відображаються конкурси, на які зареєструвався
користувач.</p>
4278      <table>
4279        <thead>
4280          <tr>
4281            <th>Конкурс</th>
4282            <th>Статус</th>
4283            <th>Дія / грамота</th>
4284          </tr>
4285        </thead>
4286        <tbody id="myContestTable">
4287          <tr>
4288            <td colspan="3">Ще немає реєстрацій.</td>
4289          </tr>
4290        </tbody>
4291      </table>
4292    </div>
4293  </div>
4294
4295    <div id="tests" class="page">
4296      <div class="card">
4297        <div class="section-head">
4298          <div>
4299            <div class="problems-kicker">Contest mode</div>
4300            <h2 id="testPageTitle">Тест конкурсу</h2>
```

```
4301          <p id="testPageDescription">Оберіть активний конкурс, щоб пройти
його тест із питаннями, які додав учитель.</p>
4302        </div>
4303        <div class="timer" id="timer">Час: 10:00</div>
4304      </div>
4305
4306      <div id="testContestMeta" class="contest-test-meta"></div>
4307
4308      <div class="progress-wrap">
4309        <div class="progress-row">
4310          <span id="progressText">Прогрес: 0 / 0</span>
4311          <span id="progressPercent">0%</span>
4312        </div>
4313        <div class="progress">
4314          <div class="progress-bar" id="progressBar"></div>
4315        </div>
4316      </div>
4317
4318      <form id="quiz"></form>
4319
4320      <div id="result" class="result"></div>
4321      <div id="resultSub" class="result-sub"></div>
4322    </div>
4323  </div>
4324
4325  <div id="rating" class="page">
4326    <div class="card rating-card">
4327      <div class="section-head rating-head">
4328        <div>
4329          <h2>Рейтинг</h2>
4330          <p>Показуємо лише тих учасників, які заходили на сайт
сьогодні.</p>
4331        </div>
4332        <div class="rating-summary">
4333          <div class="rating-pill" id="ratingTodayLabel">Сьогодні</div>
4334          <div class="rating-pill" id="ratingCountLabel">0 учасників</div>
4335        </div>
4336      </div>
4337
4338      <div class="rating-table-wrap">
4339        <table class="rating-table">
4340 <thead>
4341   <tr>
4342     <th>Місце</th>
4343     <th>Ім'я</th>
4344     <th>Бали</th>
```

```
4345    </tr>
4346  </thead>
4347  <tbody id="ratingTable"></tbody>
4348  </table>
4349          <div class="rating-empty" id="ratingEmpty"></div>
4350        </div>
4351      </div>
4352    </div>
4353
```

А.12 – Форми реєстрації та входу користувача

Код містить поля реєстрації й авторизації, які використовуються на сторінці акаунта.

```
4354    <div id="account" class="page">
4355      <div class="card">
4356        <h2>Акаунт</h2>
4357
4358        <div id="authArea">
4359          <div class="auth-shell">
4360            <div class="auth-showcase">
4361              <div>
4362                <div class="auth-kicker">Access Space</div>
4363                <h3>Профіль для навчання і керування платформою</h3>
4364                <p class="auth-copy">Акаунт дає доступ до конкурсів,
 результатів, реєстрацій і особистого кабінету. Для вчителя ще
      відкривається керування задачами, курсами та онлайн-заняттями.</p>
4365
4366              <div class="auth-feature-grid">
4367                <div class="auth-feature">
4368                  <span>Для учня</span>
4369                  <strong>Задачі, конкурси й результати в одному місці.</strong>
4370                </div>
4371                <div class="auth-feature">
4372                  <span>Для вчителя</span>
4373                  <strong>Курси, події й матеріали без редагування
 коду.</strong>
4374                </div>
4375              </div>
4376            </div>
4377
4378            <div class="auth-flags">
4379              <div class="auth-flag">Швидка реєстрація</div>
```

```
4380              <div class="auth-flag">Окремі ролі</div>
4381              <div class="auth-flag">Особистий кабінет</div>
4382            </div>
4383          </div>
4384
4385          <div class="auth-stack">
4386            <div class="auth-card auth-register-card">
4387              <div class="auth-card-head">
4388                <div>
4389                  <div class="auth-eyebrow">New profile</div>
4390                  <h3>Реєстрація</h3>
4391                  <p>Кілька полів, одна роль і готовий акаунт для роботи на
платформі.</p>
4392                </div>
4393                <div class="auth-card-mark">01</div>
4394              </div>
4395
4396              <form class="auth-form" onsubmit="event.preventDefault();
register()">
4397                <div class="auth-field-grid">
4398                  <div class="auth-field-stack">
4399                    <label class="field-label" for="regName">Ім'я</label>
4400                    <input id="regName" autocomplete="name"
placeholder="Наприклад: Марія Коваленко">
4401                  </div>
4402                  <div class="auth-field-stack">
4403                    <label class="field-label" for="regEmail">Email</label>
4404                    <input id="regEmail" type="email" autocomplete="email"
placeholder="name@example.com">
4405                  </div>
4406                </div>
4407
4408                <div class="auth-field-grid">
4409                  <div class="auth-field-stack">
4410                    <label class="field-label" for="regRole">Роль акаунта</label>
4411                    <select id="regRole">
4412                      <option value="student">Учень</option>
4413                      <option value="teacher">Вчитель</option>
4414                    </select>
4415                    <div class="field-note">Учень навчається, а вчитель ще й керує
матеріалами сайту.</div>
4416                  </div>
4417                  <div class="auth-field-stack">
4418                    <label class="field-label" for="regPass">Пароль</label>
4419                    <input id="regPass" type="password" autocomplete="new-
password" placeholder="Створіть надійний пароль">
```

```
4420                 </div>
4421               </div>
4422
4423               <div class="auth-actions">
4424                 <button type="submit" class="main-
btn">Зареєструватися</button>
        4425               </div>
```

А.13 – Розділи повідомлень, курсів та онлайн-занять

Фрагмент демонструє навчальні й інформаційні сторінки платформи.

```
4797  <div id="messages" class="page">
4798
4799  <div class="card">
4800
4801  <h2>Повідомлення</h2>
4802  <p>Останні новини та результати.</p>
4803
4804  <div id="messagesList">
4805
4806  <div class="stat">
4807  Поки що немає повідомлень.
4808  </div>
4809
4810  </div>
4811
4812  </div>
4813
4814  </div>
4815  <div id="courses" class="page">
4816    <div class="card">
4817      <div class="section-head">
4818     <div>
4819    <h2>Курси</h2>
4820    <p>Авторські навчальні треки з виразною структурою, темами уроків і
власним стилем для підготовки до олімпіад та сучасної
      розробки.</p>
4821  </div>
4822  <input class="search" id="courseSearch" placeholder="Пошук курсу..."
oninput="filterCourses()">
4823      </div>
4824
4825     <div class="contest-grid" id="courseGrid">
```

```
4826  </div>
4827  </div>
4828  </div>
4829  <div id="online" class="page">
4830    <div class="card">
4831      <div class="section-head">
4832        <div>
4833          <h2>Онлайн-заняття</h2>
4834          <p>Живі онлайн-заняття, вебінари та практичні розбори задач у
реальному часі для студентів і учасників олімпіад.</p>
4835        </div>
4836        <input class="search" id="onlineSearch" placeholder="Пошук заняття..."
oninput="filterOnlineLessons()">
4837      </div>
4838
4839      <div class="online-hero">
4840        <div class="online-showcase">
4841          <div class="online-kicker">Live learning space</div>
4842          <h3>Навчайся наживо, розбирай задачі й одразу став запитання</h3>
4843          <p>Секція онлайн-занять тепер подає активності як окремі події: з
чітким статусом, зручними датами, швидким входом у зустріч
      і компактним блоком деталей для підготовки.</p>
4844          <div class="online-badges">
4845            <div class="online-badge"><span>Формат</span> Meet / Zoom</div>
4846            <div class="online-badge"><span>Фокус</span> Алгоритми, C++,
Web</div>
4847            <div class="online-badge"><span>Рівень</span> Початківці та
олімпіадники</div>
4848          </div>
4849        </div>
4850
4851        <div class="online-summary">
4852          <h4>Швидкий огляд</h4>
4853          <div class="online-stats">
4854            <div class="online-stat">
4855              <strong>4</strong>
4856              <span>події в розкладі зараз</span>
4857            </div>
4858            <div class="online-stat">
4859              <strong>1 live</strong>
4860              <span>заняття вже триває просто зараз</span>
4861            </div>
4862            <div class="online-stat">
4863              <strong>2 кліки</strong>
4864              <span>щоб знайти подію і приєднатися</span>
4865            </div>
```

```
4866          </div>
4867          <p class="online-summary-note">Кожна картка тепер має візуальний
акцент, окремий блок дати й структуровані метадані, тому
      секція читається швидше і виглядає значно сучасніше.</p>
4868        </div>
4869      </div>
4870
4871      <div class="online-grid" id="onlineGrid">
4872        <div class="online-card" data-title="Розбір задач з алгоритмів">
4873          <div class="online-card-head">
4874            <div class="online-title-wrap">
4875              <div class="online-status live">● Онлайн зараз</div>
4876              <h3>Розбір задач з алгоритмів</h3>
4877            </div>
4878            <div class="online-day">
4879              <strong>28</strong>
4880              <span>бер</span>
4881            </div>
4882          </div>
4883          <p>Практичний онлайн-розбір типових задач на масиви, сортування
та бінарний пошук.</p>
4884
4885          <div class="online-meta">
4886            <div><b>Дата</b>28.03.2026</div>
4887            <div><b>Час</b>18:00</div>
4888            <div><b>Викладач</b>Іваненко О.О.</div>
4889            <div><b>Платформа</b>Google Meet</div>
        4890          </div>
```

А.14 – Розділ задач і контактна сторінка

Код містить структуру сторінки задач, таблицю задач і контактну інформацію.

```
4998  <div id="problems" class="page">
4999    <div class="card">
5000      <div class="section-head">
5001        <div>
5002          <h2>Задачі</h2>
5003          <p>Практичні задачі для підготовки до олімпіад з
програмування.</p>
5004        </div>
```

```
5005        <input class="search" id="problemSearch" placeholder="Пошук задачі..."
oninput="filterProblems()">
5006      </div>
5007
5008      <div class="problems-hero">
5009        <div class="problems-showcase">
5010          <div class="problems-kicker">Problem archive</div>
5011          <h3>Задачі з нормальною умовою, а не просто коротким
описом</h3>
5012          <p>У цьому розділі зібрані тренувальні задачі з алгоритмів, графів,
математики, рядків і web-напряму. Для кожної задачі
      тепер можна відкрити повну постановку: що саме треба знайти, які дані
подаються на вхід, що вивести та як виглядає приклад.</p>
5013          <div class="problem-points">
5014            <div class="problem-point"><span>Формат</span>Повна олімпіадна
постановка</div>
5015            <div class="problem-point"><span>Тренування</span>Від базових
ідей до складніших рішень</div>
5016            <div class="problem-point"><span>Фокус</span>Чіткість, логіка,
практика</div>
5017          </div>
5018        </div>
5019
5020        <div class="problems-side">
5021          <h4>Що є в умові</h4>
5022          <div class="problem-checklist">
5023            <div class="problem-check">
5024              <strong>Постановка задачі</strong>
5025              <span>Коротко й чітко пояснює, що треба обчислити або
знайти.</span>
5026            </div>
5027            <div class="problem-check">
5028              <strong>Вхід і вихід</strong>
5029              <span>Окремі блоки допомагають не плутати формат
даних.</span>
5030            </div>
5031            <div class="problem-check">
5032              <strong>Приклад</strong>
5033              <span>Дає змогу швидко перевірити, чи правильно зрозуміли
умову.</span>
5034            </div>
5035          </div>
5036        </div>
5037      </div>
5038
5039      <div class="toolbar">
```

```
5040      <select id="topicFilter" onchange="filterProblems()">
5041        <option value="">Усі теми</option>
5042        <option value="Алгоритми">Алгоритми</option>
5043        <option value="Графи">Графи</option>
5044        <option value="Математика">Математика</option>
5045        <option value="Рядки">Рядки</option>
5046        <option value="Web">Web</option>
5047      </select>
5048
5049      <select id="difficultyFilter" onchange="filterProblems()">
5050        <option value="">Уся складність</option>
5051        <option value="Легка">Легка</option>
5052        <option value="Середня">Середня</option>
5053        <option value="Складна">Складна</option>
5054      </select>
5055
5056      <select id="sortProblems" onchange="sortProblemsList()">
5057        <option value="id">Сортувати: за номером</option>
5058        <option value="name">Сортувати: за назвою</option>
5059        <option value="difficulty">Сортувати: за складністю</option>
5060      </select>
5061    </div>
5062
5063    <table>
5064      <thead>
5065        <tr>
5066          <th>ID</th>
5067          <th>Назва</th>
5068          <th>Тема</th>
5069          <th>Складність</th>
5070          <th>Автор</th>
5071          <th>Дія</th>
5072        </tr>
5073      </thead>
5074      <tbody id="problemsTable"></tbody>
5075    </table>
5076  </div>
5077
5078  </div>
5079 <!-- CONTACTS PAGE -->
5080
```

А.15 – Відображення конкурсного тесту

1 Функції створюють тестовий блок активного конкурсу і формують картки конкурсів.

```
6167  function renderContestQuizEmpty(message = "Оберіть активний конкурс,
щоб пройти тест.", description = "Після старту конкурс
      відкриється тут із питаннями, які додав учитель."){
6168    const quiz = document.getElementById("quiz");
6169    if(!quiz) return;
6170
6171    currentQuizQuestions = [];
6172    currentQuizContestName = "";
6173    testFinished = false;
6174    stopTimer();
6175    timeLeft = 600;
6176
6177    document.getElementById("testPageTitle").textContent = "Тест конкурсу";
6178    document.getElementById("testPageDescription").textContent = description;
6179    document.getElementById("testContestMeta").innerHTML = "";
6180    document.getElementById("result").textContent = "";
6181    document.getElementById("resultSub").textContent = "";
6182    quiz.innerHTML = `
6183      <div class="quiz-empty">
6184        <h3>${escapeHtml(message)}</h3>
6185        <p>${escapeHtml(description)}</p>
6186        <div class="toolbar">
6187          <button class="secondary-btn" type="button"
onclick="openPage('contest', null)">Перейти до конкурсів</button>
6188        </div>
6189      </div>
6190    `;
6191
6192    updateProgress();
6193    renderTimer();
6194  }
6195
6196  function renderContestQuiz(contestName, resetState = true){
6197    const contest = getContestByTitle(contestName);
6198
6199    if(!contest){
6200      activeContestSessionName = "";
6201      renderContestQuizEmpty("Цей конкурс більше недоступний.", "Оновіть
сторінку або оберіть інший конкурс у списку.");
6202      return false;
6203    }
```

```
6204
6205    const questions = normalizeContestQuestionsClient(contest.questions);
6206
6207    if(!questions.length){
6208      activeContestSessionName = "";
6209      renderContestQuizEmpty("Для цього конкурсу ще не додано тест.",
"Поверніться до конкурсів або попросіть вчителя додати
      питання.");
6210      return false;
6211    }
6212
6213    currentQuizContestName = contest.title || "";
6214    currentQuizQuestions = questions;
6215
6216    if(resetState){
6217      testFinished = false;
6218      stopTimer();
6219      timeLeft = 600;
6220    }
6221
6222    document.getElementById("testPageTitle").textContent = contest.title ||
"Тест конкурсу";
6223    document.getElementById("testPageDescription").textContent =
contest.description || "Пройдіть тест конкурсу та одразу дізнайтеся
      свій результат.";
6224    document.getElementById("testContestMeta").innerHTML = `
6225      <div class="contest-test-chip"><b>Питань</b> ${questions.length}</div>
6226      <div class="contest-test-chip"><b>Старт</b>
${escapeHtml(formatContestScheduleLabel(contest))}</div>
6227      <div class="contest-test-chip"><b>Формат</b>
${escapeHtml(contest.format || "Онлайн")}</div>
6228    `;
6229    document.getElementById("result").textContent = "";
6230    document.getElementById("resultSub").textContent = "";
6231
6232    document.getElementById("quiz").innerHTML = `
6233      ${questions.map((question, index) => `
6234        <div class="question" data-question-index="${index}">
6235          <div class="question-head">
6236            <div class="question-index">Питання ${index + 1}</div>
6237            <div class="question-note">${question.options.length} варіанти
відповіді</div>
6238          </div>
6239          <p>${escapeHtml(question.text)}</p>
6240          <div class="question-options">
6241            ${question.options.map((option, optionIndex) => `
```

```
            <label class="option">
              <input type="radio" name="contestQuestion_${index}" value="${optionIndex}" onchange="updateProgress()">
              <span class="option-copy">${escapeHtml(option)}</span>
            </label>
          `).join("")}
        </div>
      </div>
    `).join("")}
    <div class="quiz-toolbar">
      <button id="quizFinishButton" class="main-btn" type="button" onclick="checkTest()">Завершити</button>
      <button id="quizResetButton" class="secondary-btn" type="button" onclick="resetTest()">Скинути тест</button>
    </div>
  `;

  updateProgress();
  renderTimer();
  syncContestTimer();
  return true;
}

async function handleContestPrimaryAction(contestId){
  const contest = contestsData.find(item => Number(item.id) === Number(contestId));

  if(!contest){
    showToast("Конкурс не знайдено");
    return;
  }

  if(!getContestQuestionCount(contest)){
    showToast("Для цього конкурсу вчитель ще не додав тестові питання");
    return;
  }

  const availability = getContestAvailability(contest);

  if(availability.action === "start"){
    await startContest(contest.title);
    return;
  }

  if(availability.action === "register"){
    await registerContest(contest.title);
```

```
6284      return;
6285    }
6286
6287    showToast("Цей конкурс уже завершено");
6288  }
6289
6290  function renderContestGrid(rows = contestsData){
6291    const grid = document.getElementById("contestGrid");
6292    if(!grid) return;
6293
6294    if(!rows.length){
6295      grid.innerHTML = `<div class="contest-card"><div class="contest-
tag">Новий старт</div><h3>Конкурсів поки що немає</h3><p>У цьому
      розділі з’являться конкурси, щойно вчитель додасть їх через панель
вчителя.</p></div>`;
6296      return;
6297    }
6298
6299    grid.innerHTML = rows.map(contest => {
6300      const questionCount = getContestQuestionCount(contest);
6301      const hasQuestions = questionCount > 0;
6302      const availability = hasQuestions
6303        ? getContestAvailability(contest)
6304        : {
6305            state: "closed",
6306            statusText: "Тест ще не додано",
6307            buttonText: "Тест готується",
6308            action: "closed"
6309          };
6310      const contestId = Number(contest.id) || 0;
6311      const primaryButton =
6312        availability.action === "closed"
6313          ? `<button class="secondary-btn"
disabled>${availability.buttonText}</button>`
6314          : `<button class="main-btn"
onclick="handleContestPrimaryAction(${contestId})">${availability.buttonText}</b
utton>`;
6315
6316      return `
6317        <div class="contest-card" data-id="${contestId}" data-
title="${escapeHtml(contest.title)}">
6318          <div class="contest-tag">${escapeHtml(contest.tag)}</div>
6319          <h3>${escapeHtml(contest.title)}</h3>
6320          <p>${escapeHtml(contest.description)}</p>
6321          <div class="contest-meta">Дата:
${formatContestDate(contest.date)}</div>
```

```
6322          <div class="contest-meta">Час: ${escapeHtml(contest.time)}</div>
6323          <div class="contest-meta">Формат:
${escapeHtml(contest.format)}</div>
6324          <div class="contest-meta">Тест: ${questionCount} питань</div>
6325          <div class="contest-meta status-text">Статус: <span class="status-
${availability.state === "soon" ? "soon" :
      availability.state === "closed" ? "closed" :
"open"}">${escapeHtml(availability.statusText)}</span></div>
6326          <div class="toolbar">
6327            ${primaryButton}
6328            <button class="secondary-btn"
onclick="toggleContest(this)">Деталі</button>
6329          </div>
6330          <div class="contest-content hidden" style="margin-top:14px">
6331            <p><b>Старт:</b>
${escapeHtml(formatContestScheduleLabel(contest))}</p>
6332            ${(contest.details || []).map(item =>
`<p><b>${escapeHtml(item.split(":")[0])}:</b>
      ${escapeHtml(item.split(":").slice(1).join(":").trim())}</p>`).join("")}
6333          </div>
6334        </div>
6335      `;
6336    }).join("");
6337  }
6338
        6339  async function loadContests(){
```

А.16 – Навігація, прогрес тесту та перевірка відповідей

Фрагмент відповідає за перемикання сторінок, прогрес виконання тесту та збір відповідей.

```
7044  function openPage(page, btn){
7045    document.querySelectorAll(".page").forEach(p => {
7046      p.classList.remove("active");
7047    });
7048
7049    const targetPage = document.getElementById(page);
7050    if(targetPage){
7051      targetPage.classList.add("active");
7052    }
7053
7054    const activePage = page === "tests" ? "contest" : page;
```

```
    const mobileActivePage = isMobileMorePage(activePage) ? "more" : activePage;

    closeMobileMoreMenu();

    document.querySelectorAll("nav button, .account-btn, .mobile-nav-btn").forEach(button => {
      button.classList.remove("active-nav");
    });

    document.querySelectorAll(`[data-page="${activePage}"]`).forEach(button => {
      button.classList.add("active-nav");
    });

    document.querySelectorAll(`.mobile-nav-btn[data-page="${mobileActivePage}"]`).forEach(button => {
      button.classList.add("active-nav");
    });

    if(page === "home"){
      renderHomeOverview();
      renderHomeNews();
    }

    syncContestTimer();
  }


  function showToast(text){
    const toast = document.getElementById("toast");
    toast.textContent = text;
    toast.classList.add("show");
    setTimeout(() => toast.classList.remove("show"), 2400);
  }

  function getAnsweredContestQuestionsCount(){
    return currentQuizQuestions.reduce((count, _, index) => {
      return document.querySelector(`input[name="contestQuestion_${index}"]:checked`) ? count + 1 : count;
    }, 0);
  }

  function updateProgress(){
    const total = currentQuizQuestions.length;
```

```
  const answered = getAnsweredContestQuestionsCount();
  const percent = total ? Math.round((answered / total) * 100) : 0;
  document.getElementById("progressBar").style.width = percent + "%";
  document.getElementById("progressText").textContent = `Прогрес: ${answered} / ${total}`;
  document.getElementById("progressPercent").textContent = percent + "%";
}

function clearHighlights(){
  document.querySelectorAll("#quiz .option").forEach(opt => {
    opt.classList.remove("correct", "wrong");
  });
}

function setQuizInputsDisabled(disabled){
  document.querySelectorAll('#quiz input[type="radio"]').forEach(input => {
    input.disabled = disabled;
  });
}

function collectQuizAnswers(){
  return currentQuizQuestions.map((_, index) => {
    const checked = document.querySelector(`input[name="contestQuestion_${index}"]:checked`);
    return checked ? Number(checked.value) : null;
  });
}

function applyContestReview(review = []){
  review.forEach((item, index) => {
    const options = document.querySelectorAll(`input[name="contestQuestion_${index}"]`);

    options.forEach((input, optionIndex) => {
      const label = input.closest(".option");
      if(!label) return;

      if(optionIndex === Number(item?.correctIndex)){
        label.classList.add("correct");
      }

      if(
        item?.selectedIndex !== null &&
        item?.selectedIndex !== undefined &&
        optionIndex === Number(item.selectedIndex) &&
        Number(item.selectedIndex) !== Number(item.correctIndex)
```

```
      ){
        label.classList.add("wrong");
      }
    });
  });
}

async function checkTest(){
  if(!currentQuizQuestions.length){
    renderContestQuizEmpty();
    openPage("contest", null);
    return;
  }
```

А.17 – Реєстрація, вхід і профіль користувача на клієнті

Код надсилає запити до сервера для реєстрації, входу, виходу та відображення профілю.

```
async function register(){
  const name = document.getElementById("regName").value.trim();
  const email = document.getElementById("regEmail").value.trim();
  const role = document.getElementById("regRole").value;
  const pass = document.getElementById("regPass").value.trim();

  if(!name || !email || !pass){
    showToast("Заповніть усі поля");
    return;
  }

  try{
    const res = await fetch("/register", {
      method: "POST",
      headers: {
        "Content-Type": "application/json"
      },
      body: JSON.stringify({
        name: name,
        email: email,
        role: role,
        password: pass
      })
```

```
7320      });
7321
7322      const data = await res.json();
7323
7324  if(data.success){
7325    document.getElementById("regName").value = "";
7326    document.getElementById("regEmail").value = "";
7327    document.getElementById("regRole").value = "student";
7328    document.getElementById("regPass").value = "";
7329    showToast("Реєстрація успішна");
7330  } else {
7331    showToast(data.message || "Помилка реєстрації");
7332  }
7333    } catch(e){
7334      showToast("Сервер недоступний");
7335    }
7336  }
7337
7338  async function registerContest(contestName){
7339    try{
7340      const res = await fetch("/contest/register", {
7341        method: "POST",
7342        headers: { "Content-Type": "application/json" },
7343        body: JSON.stringify({ contestName })
7344      });
7345
7346      const data = await res.json();
7347
7348      if(!data.success){
7349        showToast(data.message || "Помилка реєстрації на конкурс");
7350        if(res.status === 401){
7351          openPage("account", null);
7352        }
7353        return;
7354      }
7355
7356      await renderMyContests();
7357      showToast(data.message || `Ви зареєструвалися на конкурс
"${contestName}"`);
7358    } catch(e){
7359      showToast("Сервер недоступний");
7360    }
7361  }
7362
7363  async function startContest(contestName){
7364    try{
```

```
7365      const res = await fetch("/contest/start", {
7366        method: "POST",
7367        headers: { "Content-Type": "application/json" },
7368        body: JSON.stringify({ contestName })
7369      });
7370
7371      const data = await res.json();
7372
7373      if(!data.success){
7374        showToast(data.message || "Поки що цей конкурс недоступний");
7375        if(res.status === 401){
7376          openPage("account", null);
7377        }
7378        return;
7379      }
7380
7381      activeContestSessionName = contestName;
7382      await renderMyContests();
7383      renderContestQuiz(contestName, true);
7384      openPage("tests", null);
7385      showToast(data.message || `Конкурс "${contestName}" відкрито`);
7386    } catch(e){
7387      showToast("Сервер недоступний");
7388    }
7389  }
7390
7391  async function renderMyContests(){
7392    const table = document.getElementById("myContestTable");
7393    if(!table) return;
7394
7395    try{
7396      const res = await fetch("/me");
7397      const data = await res.json();
7398
7399      if(!data.loggedIn){
7400        table.innerHTML = `<tr><td colspan="3">Увійдіть в акаунт, щоб бачити
свої конкурси.</td></tr>`;
7401        return;
7402      }
7403
7404      const userContests = data.user.myContests || [];
7405
7406      if(userContests.length === 0){
7407        table.innerHTML = `<tr><td colspan="3">Ще немає
реєстрацій.</td></tr>`;
7408        return;
```

```
7409      }
7410
7411      table.innerHTML = "";
7412
7413      userContests.forEach(item => {
7414    const row = document.createElement("tr");
7415    const contestMeta = contestsData.find(contest => (contest.title || "").trim()
=== String(item.contest || "").trim());
7416    const questionCount = contestMeta ? getContestQuestionCount(contestMeta)
: 0;
7417    const availability = contestMeta && questionCount ?
getContestAvailability(contestMeta) : null;
7418    const rawStatus = String(item.status || "");
7419    const normalizedStatus = rawStatus.toLowerCase();
7420
7421    const contestCell = document.createElement("td");
7422    contestCell.textContent = item.contest;
7423
7424    const statusCell = document.createElement("td");
7425    if(item.certificate || normalizedStatus.includes("успішно завершено") ||
normalizedStatus === "завершено"){
7426      statusCell.textContent = rawStatus;
7427    } else if(contestMeta && !questionCount){
7428      statusCell.textContent = "Тест ще не додано";
7429    } else if(normalizedStatus === "йде зараз"){
7430      statusCell.textContent = "Йде зараз";
7431    } else if(availability?.state === "open"){
7432      statusCell.textContent = "Можна починати";
7433    } else if(availability?.state === "closed"){
7434      statusCell.textContent = "Час конкурсу минув";
7435    } else if(availability){
7436      statusCell.textContent = "Зареєстровано. Очікує старту";
7437    } else {
7438      statusCell.textContent = rawStatus || "Зареєстровано";
7439    }
7440
7441    const certCell = document.createElement("td");
7442
7443    if(item.certificate){
7444      const btn = document.createElement("button");
7445      btn.className = "certificate-btn";
7446      btn.textContent = "Завантажити PDF";
7447      btn.onclick = function(){
7448        downloadCertificate(item.contest);
7449      };
7450      certCell.appendChild(btn);
```

```
7451    } else if(contestMeta && !questionCount){
7452      certCell.textContent = "Очікує наповнення тестом";
7453    } else if(availability?.state === "open"){
7454      const btn = document.createElement("button");
7455      btn.className = "main-btn";
7456      btn.style.marginTop = "0";
7457      btn.textContent = "Почати";
7458      btn.onclick = function(){
7459        startContest(item.contest);
7460      };
7461      certCell.appendChild(btn);
7462    } else if(availability?.state === "soon"){
7463      certCell.textContent = `Старт
${formatContestScheduleLabel(contestMeta)}`;
7464    } else if(availability?.state === "closed"){
7465      certCell.textContent = "Конкурс завершено";
7466    } else {
7467      certCell.textContent = "Ще недоступна";
7468    }
7469
7470    row.appendChild(contestCell);
7471    row.appendChild(statusCell);
7472    row.appendChild(certCell);
7473
7474    table.appendChild(row);
7475  });
7476    } catch(e){
7477      table.innerHTML = `<tr><td colspan="3">Не вдалося завантажити
конкурси.</td></tr>`;
7478    }
7479  }
7480
7481  async function login(){
7482    const email = document.getElementById("loginEmail").value.trim();
7483    const pass = document.getElementById("loginPass").value.trim();
7484
7485    if(!email || !pass){
7486      showToast("Введіть email і пароль");
7487      return;
7488    }
7489
7490    try{
7491      const res = await fetch("/login", {
7492        method: "POST",
7493        headers: { "Content-Type": "application/json" },
7494        body: JSON.stringify({ email, password: pass })
```

```
    });

    const data = await res.json();

    if(data.success){
      showProfile(data.name, data.email, data.role);
      await loadStudentCabinet();
      await renderMyContests();
      await renderMessages();
      showToast("Вхід успішний");
    } else {
      showToast(data.message || "Помилка входу");
    }
  } catch(e){
    showToast("Сервер недоступний");
  }
}

async function logout(){
  try{
    await fetch("/logout", { method: "POST" });
  } catch(e){}

  document.getElementById("authArea").style.display = "block";
  document.getElementById("profile").style.display = "none";

  document.getElementById("loginEmail").value = "";
  document.getElementById("loginPass").value = "";

  document.getElementById("studentScore").textContent = "0";
  document.getElementById("studentLastResult").textContent = "—";
  document.getElementById("studentRank").textContent = "—";
  document.getElementById("lastResultText").textContent = "Ще немає
спроб.";
  currentUserName = "";
  activeContestSessionName = "";
  setTeacherPanelVisibility("student");
  clearTeacherProblemForm();
  clearTeacherLessonForm();
  clearTeacherContestForm();
  clearTeacherCourseForm();
  renderContestQuizEmpty();

  showToast("Ви вийшли з акаунта");
  renderMyContests();
  renderMessages();
```

```
7540    renderHomeOverview();
7541    openPage("home", null);
7542  }
7543
7544  function showProfile(name, email, role = "student"){
7545    document.getElementById("authArea").style.display = "none";
7546    document.getElementById("profile").style.display = "block";
7547    document.getElementById("userName").innerText = name;
7548    document.getElementById("userEmail").innerText = email || "";
7549    document.getElementById("avatarLetter").innerText = name ?
name[0].toUpperCase() : "U";
7550    currentUserName = name || "";
7551    setTeacherPanelVisibility(role);
7552
7553    loadStudentCabinet();
7554  }
7555
        7556  function toggleTheme(){
```

А.18 – Завантаження рейтингу та сесії користувача

Фрагмент отримує список користувачів, сортує рейтинг і перевіряє активну сесію.

```
7877  function markContestStatuses(){
7878    if(!document.getElementById("contestGrid")) return;
7879
7880    const query =
document.getElementById("contestSearch")?.value.toLowerCase().trim() || "";
7881    const filtered = contestsData.filter(contest => (contest.title ||
"").toLowerCase().includes(query));
7882    renderContestGrid(query ? filtered : contestsData);
7883  }
7884
7885  function formatRatingVisitorCount(count){
7886    const safeCount = Math.max(0, Number(count) || 0);
7887    const lastTwoDigits = safeCount % 100;
7888    const lastDigit = safeCount % 10;
7889
7890    let noun = "учасників";
7891    if (lastTwoDigits < 11 || lastTwoDigits > 14) {
7892      if (lastDigit === 1) {
7893        noun = "учасник";
```

```
    } else if (lastDigit >= 2 && lastDigit <= 4) {
      noun = "учасники";
    }
  }

  return `${safeCount} ${noun} сьогодні`;
}

async function loadRating(){
  try{
    const res = await fetch("/users");
    const ratingPayload = await res.json();
    const users = Array.isArray(ratingPayload?.users)
      ? ratingPayload.users.slice()
      : Array.isArray(ratingPayload)
        ? ratingPayload.slice()
        : [];

    const table = document.getElementById("ratingTable");
    const empty = document.getElementById("ratingEmpty");
    const todayLabel = document.getElementById("ratingTodayLabel");
    const countLabel = document.getElementById("ratingCountLabel");

    table.innerHTML = "";
    users.sort((a, b) => (b.score || 0) - (a.score || 0));

    if(todayLabel){
      todayLabel.textContent = ratingPayload?.todayLabel
        ? `Сьогодні: ${ratingPayload.todayLabel}`
        : "Сьогодні";
    }

    if(countLabel){
      countLabel.textContent = formatRatingVisitorCount(users.length);
    }

    if(!users.length){
      if(empty){
        empty.style.display = "block";
        empty.textContent = "Сьогодні в рейтингу ще немає учасників. Він з'являється автоматично, щойно хтось заходить на сайт.";
      }
      return;
    }

    if(empty){
```

```
7939        empty.style.display = "none";
7940        empty.textContent = "";
7941      }
7942
7943      users.forEach((user, index) => {
7944        const row = document.createElement("tr");
7945
7946        const placeCell = document.createElement("td");
7947        placeCell.textContent = index + 1;
7948        placeCell.dataset.label = "Місце";
7949        placeCell.className = `rating-place ${index === 0 ? "rank-1" : index ===
1 ? "rank-2" : index === 2 ? "rank-3" : ""}`.trim();
7950
7951        const nameCell = document.createElement("td");
7952        nameCell.textContent = user.name;
7953        nameCell.dataset.label = "Ім'я";
7954        nameCell.className = "rating-name";
7955
7956        const scoreCell = document.createElement("td");
7957        scoreCell.textContent = user.score || 0;
7958        scoreCell.dataset.label = "Бали";
7959        scoreCell.className = "rating-score";
7960
7961        row.appendChild(placeCell);
7962        row.appendChild(nameCell);
7963        row.appendChild(scoreCell);
7964
7965        table.appendChild(row);
7966      });
7967    } catch(e){
7968      console.log("Не вдалося завантажити рейтинг", e);
7969    }
7970  }
7971
7972  async function checkSession(){
```

А.19 – Приклад структури users.json і system.json

JSON-файли використовуються для зберігання користувачів і службової інформації про дату оновлення рейтингу. У прикладі персональні дані замінено умовними значеннями.

```
0001  [
```

```
  {
    "name": "Student Demo",
    "email": "student@example.com",
    "password": "$2b$10$hashedPasswordExample",
    "role": "student",
    "score": 20,
    "myContests": [
      {
        "name": "Демо-конкурс з програмування",
        "status": "completed",
        "score": 20,
        "total": 20,
        "startedAt": "2026-04-30T10:00:00.000Z",
        "finishedAt": "2026-04-30T10:15:00.000Z"
      }
    ],
    "activeContest": "",
    "lastResult": "20",
    "messages": [
      {
        "text": "Вітаємо! Ви завершили тест. Результат: 20 балів.",
        "date": "30.04.2026, 10:15:00"
      }
    ]
  },
  {
    "name": "Teacher Demo",
    "email": "teacher@example.com",
    "password": "$2b$10$hashedPasswordExample",
    "role": "teacher",
    "score": 0,
    "myContests": [],
    "activeContest": "",
    "lastResult": "",
    "messages": []
  }
]

{
  "lastReset": "2026-04-30"
}
```

ДОДАТОК Б

СТРУКТУРА ФАЙЛІВ ПРОЄКТУ

Таблиця Б.1 – Структура файлів проєкту

Файл	Розмір	Призначення
contests.json	2 байт	зберігання даних про конкурси
courses.json	2 байт	зберігання даних про курси
index.html	231662 байт	клієнтська частина платформи
online.json	2 байт	зберігання даних про онлайн-заняття
package.json	397 байт	залежності проєкту та команда запуску
problems.json	2 байт	зберігання даних про задачі
server.js	46386 байт	серверна частина та API
system.json	26 байт	службова інформація для скидання рейтингу
users.json	5117 байт	користувачі, бали та повідомлення