Міністерство освіти і науки України
Черкаський національний університет імені Богдана Хмельницького
Факультет обчислювальної техніки, інтелектуальних та управляючих систем
Кафедра програмного забезпечення автоматизованих систем

КУРСОВА РОБОТА З БАЗ ДАНИХ

на тему «Облік результатів учнівської олімпіади з програмування»

Студента (ки) 3 курсу, групи КС-22
спеціальності 121 “Інженерія програмного забезпечення”
Мартиненка О.С
(прізвище та ініціали)
Керівник:
ст. викладач Порубльов І.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Оцінка за шкалою:
_____
(національною, кількість балів, ECTS)

Члени комісії:

_____ _____
(підпис) (прізвище та ініціали)

_____ _____
(підпис) (прізвище та ініціали)

_____ _____
(підпис) (прізвище та ініціали)

Черкаси – 2025 рік

ЗМІСТ

ВСТУП

У сучасному суспільстві інформаційні технології відіграють важливу роль у різних сферах життя, зокрема в освітньому процесі. У зв’язку з цим значного значення набуває якісна підготовка учнів до професійної діяльності в галузі ІТ. Одним із ефективних способів виявлення та підтримки обдарованої молоді є участь у шкільних олімпіадах з програмування. Такі змагання не лише сприяють глибшому засвоєнню інформатики, але й формують аналітичне мислення та навички побудови алгоритмів.

Зі збільшенням кількості учасників, розширенням масштабу змагань та ускладненням завдань виникає потреба у зручному механізмі збереження, обробки й аналізу результатів. Існуючі платформи, зокрема Codeforces, AtCoder та Eolymp, орієнтовані переважно на великі онлайн-турніри. Проте вони не завжди враховують специфіку локальних олімпіад, особливо в частині аналізу результатів за окремими навчальними закладами, класами, викладачами чи мовами програмування.

Це зумовлює необхідність створення спеціалізованої бази даних, яка дозволить централізовано зберігати та структурувати відомості про учасників і їхні досягнення, а також надасть можливість здійснювати різні типи аналітики. Бази даних є ключовим елементом сучасних інформаційних систем, адже забезпечують впорядковане збереження даних, їхню ефективну обробку і швидкий доступ до інформації. У випадку з олімпіадами це стосується зокрема персональних даних учасників, виконаних завдань, мов програмування, отриманих балів тощо.

Метою цієї курсової роботи є розробка структури бази даних для збереження результатів учнівських олімпіад з програмування, а також створення засобів для внесення, обробки та аналізу цієї інформації. У рамках реалізації передбачається підтримка імпорту результатів у форматі JSON, збереження

історії виконаних спроб, формування вибірок за різними критеріями, а також виконання аналітичних запитів.

Основою для зберігання даних обрано систему управління базами даних PostgreSQL. Це реляційна СКБД з відкритим кодом, яка підтримує складні запити, процедури та інструменти для аналітичної обробки даних.

Серверну частину програмного забезпечення реалізовано з використанням мови JavaScript у середовищі Node.js та фреймворку Express.js. Такий підхід забезпечує гнучкість, масштабованість і зручність підтримки. Крім того, буде створено веб-інтерфейс для введення й перегляду даних, що зробить систему доступною через браузер на будь-якому пристрої.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Огляд предметної області та порівняння з аналогічними рішеннями

Олімпіади з програмування вже давно стали важливою частиною підготовки учнів, які цікавляться інформаційними технологіями. У школах вони виконують не лише роль змагань, а й мотивують учнів до навчання, допомагають розвивати логічне та алгоритмічне мислення. Участь у таких заходах дає можливість спробувати себе в розв’язанні нестандартних задач, що часто мають прикладне значення і наближені до реальних проблем у сфері ІТ.

Щороку змагання охоплюють усе більше учасників, що, своєю чергою, ускладнює організацію процесу. Зростає обсяг інформації, яку потрібно зберігати та аналізувати: реєстрація учасників, їхні спроби, бали, рейтинг, статистика за роками. Часто ці процеси реалізуються за допомогою звичайних електронних таблиць — наприклад, у Microsoft Excel чи Google Sheets. Це зручно на перших етапах, але в міру зростання кількості даних виникає низка труднощів:

- таблиці не пов’язані між собою, що ускладнює перевірку даних
- дублювання записів і відсутність унікальних ідентифікаторів призводять до помилок
- ручне введення інформації часто спричиняє неточності
- для фільтрації або обчислень доводиться будувати складні формули
- система погано масштабується при збільшенні кількості учасників або нових змагань.

Ці недоліки стають особливо відчутними, коли йдеться про обробку даних з різних шкіл, класів, мов програмування або про порівняння результатів у динаміці — наприклад, за кілька років. Також ускладнюється обмін даними з іншими освітніми системами або платформами.

Поряд із локальними змаганнями активно використовуються великі онлайн-платформи, зокрема Codeforces, AtCoder, LeetCode, E-olymp. Вони справді частково вирішують подібні завдання — надають автоматичне оцінювання, рейтинги, базову аналітику. Але аналіз їхньої функціональності показує певні обмеження:

- вони орієнтовані передусім на індивідуальних учасників, без врахування навчального закладу чи класу
- не зберігають дані про вчителів або школи в структурованому вигляді
- не дозволяють легко імпортувати локальні дані, зокрема у форматі JSON
- аналітика зосереджена здебільшого на загальних рейтингах, і не дає змоги формувати, наприклад, рейтинг вчителів за середнім балом їхніх учнів.

З огляду на ці фактори, виникає потреба у створенні спеціалізованої інформаційної системи, яка б дозволяла зручно вести облік учасників, аналізувати результати, будувати звіти та підтримувати інтеграцію з іншими інструментами. Такий підхід допоможе організаторам шкільних і регіональних олімпіад працювати ефективніше та приймати обґрунтовані рішення на основі даних.

1.2. Постановка основних функціональних задач

1.2.1. Ключові задачі реалізації системи

Під час розробки інформаційної системи необхідно вирішити кілька важливих технічних і логічних задач. Основний акцент робиться на побудові архітектури, організації зберігання та обробки даних, а також на забезпеченні зручної взаємодії з користувачем. До ключових завдань входить:

- побудова структури бази даних, у якій чітко визначено взаємозв’язки між основними сутностями такими як учасниками, викладачами,

навчальними закладами, задачами, спробами, мовами програмування та вердиктами

- створення REST API з реалізацією повного набору CRUD-операцій для всіх основних об’єктів системи, що дозволяє централізовано керувати даними з боку клієнтського застосунку
- розробка інтерфейсу користувача, який забезпечуватиме зручний перегляд і редагування даних, включаючи пошук, фільтрацію, сортування та використання інтерактивних елементів у таблицях

1.2.2. Моделювання предметної області

Для реалізації системи необхідно спершу сформулювати перелік ключових сутностей предметної області та описати їхні властивості. Це дозволить забезпечити повноту й узгодженість даних. Після цього будується реляційна модель бази даних, що враховує принципи нормалізації та встановлює зв’язки між таблицями за допомогою зовнішніх ключів.

Далі визначаються основні сценарії взаємодії користувача із системою — введення нових даних, редагування записів, перегляд та видалення інформації. Окрім цього, має бути реалізована аналітична складова: можливість формувати звіти, будувати рейтинги, аналізувати динаміку результатів і виявляти певні закономірності у даних.

Інформація, яку необхідно зберігати у базі даних:

- учасники — ПІБ, клас, навчальний заклад, викладач
- викладачі — ПІБ, навчальний заклад
- навчальні заклади — назва школи або ліцею
- класи — позначення (наприклад, 10-А)
- мови програмування — назва мови (наприклад, Python, Java, C++)
- задачі — назва, короткий опис, мова програмування, для якої вона призначена
- вердикти — код (наприклад, AC, WA, TLE) та пояснення

– спроби — учасник, задача, дата і час подачі, мова, кількість балів, текст розв’язку, вердикт

Основні функції, які має підтримувати система:

– додавання, редагування та видалення записів для всіх сутностей
– пошук та фільтрація даних за різними критеріями: ім’я учасника чи викладача, назва закладу або клас, назва задачі, мова програмування, дата спроби, кількість балів
– сортування результатів за кількістю балів, датою подачі, прізвищем учасника або назвою задачі
– формування даних для побудови графіків та аналітики на клієнтській стороні

Запити, на які має відповідати система:

– Хто з учасників набрав найбільшу кількість балів сумарно?
– Який викладач демонструє найвищі середні результати серед своїх учнів?
– Які задачі мають найбільший відсоток успішних спроб?
– Яка мова програмування використовується найефективніше?
– Яку мову програмування найчастіше обирали для розв’язання певної задачі?

1.2.3. Розробка користувацького інтерфейсу системи

Наступним планується створення веб-інтерфейсу для користувачів з різними ролями (адміністратор, викладач, учасник), що дозволить взаємодіяти із системою без потреби у технічних знаннях.

– Додавання, редагування й видалення будь-яких записів (учасники, викладачі, задачі, спроби тощо).
– Авторизацію та контроль доступу з різними рівнями прав (адміністратор, гість).

- Зручний пошук і фільтрацію за ключовими параметрами (ПІБ, мова програмування, бали, дата).
- Сортування за різними критеріями: прізвище, кількість балів, дата.
- Генерацію статистики та аналітичних даних: рейтинги, динаміка результатів, загальна успішність.
- Візуалізацію даних за допомогою графіків, діаграм, таблиць.

1.2.4. Побудова API та логіки взаємодії між компонентами

Для забезпечення узгодженої роботи клієнтської та серверної частин буде розроблено REST API, що дозволить централізовано взаємодіяти з базою даних. Це включатиме:

- Розробку окремих ендпоінтів для взаємодії з кожним відношенням, функцією або тригером (учасники, задачі, викладачі, спроби тощо).
- Реалізацію стандартних CRUD-операцій такі як створення, перегляд, оновлення, видалення.
- Узгодження структури відповіді для зручності обробки даних у клієнтському інтерфейсі.

РОЗДІЛ 2
ПРОЕКТУВАННЯ БАЗИ ДАНИХ

2.1. Інфологічне та даталогічне проектування.

2.1.1. Інфологічне проетування.

У результаті аналізу предметної області, пов’язаної з проведенням олімпіад з програмування, було виокремлено основні сутності, які формують логіку роботи майбутньої системи. До них належать: навчальні заклади, класи, викладачі, учасники, задачі, мови програмування, вердикти автоматичної перевірки та спроби розв’язання задач.

На ER-діаграмі (рис. 2.1) відображено ці сутності, а також їхні атрибути та зв’язки між ними. Зокрема, кожен учасник має прізвище, ім’я, по батькові, належить до певного класу та навчається у конкретному навчальному закладі. За кожним учасником закріплений викладач, який також прив’язаний до певної школи або ліцею.

Задачі, що використовуються на олімпіадах, мають назву, опис і визначену мову програмування, на якій очікується їх реалізація. Спроба розв’язання задачі містить інформацію про дату та час подання, мову, яку було використано, кількість набраних балів, текст програми, а також вердикт, який система повернула після автоматичної перевірки.

Взаємозв’язки між сутностями побудовані таким чином: один викладач може працювати з кількома учнями; кожен учасник може подати кілька спроб на різні задачі; кожна спроба прив’язується до однієї задачі, реалізується лише однією мовою програмування й має один результат перевірки.

Загалом така структура дозволяє логічно і повно охопити основні процеси, які відбуваються в рамках олімпіадного програмування, та створити гнучку модель для подальшої роботи з базою даних.

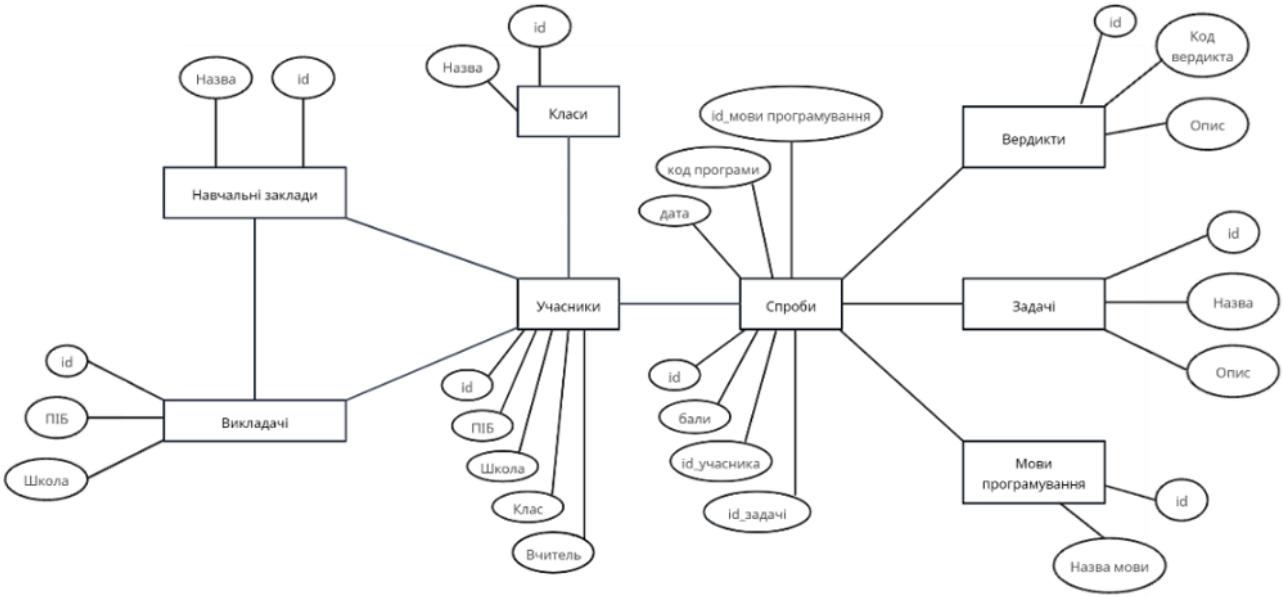

Рис. 2.1 – Інфологічна модель предметної області «Облік результатів учнівських олімпіад з програмування»

2.1.2. Даталогічне проектування.

На підставі виявлених сутностей, їхніх атрибутів та встановлених між ними взаємозв’язків було виконано даталогічне проєктування системи. Було розроблено логічну структуру бази даних, яка реалізована за допомогою реляційної СКБД PostgreSQL. Кожна сутність реалізована у вигляді окремої таблиці з чітко визначеними атрибутами, відповідними типами даних, ключами та обмеженнями цілісності (рис. 2.2).

До складу бази даних входять такі таблиці:

- Навчальні заклади, класи, мови програмування, вердикти — довідкові таблиці, які використовуються як зовнішні ключі в основних таблицях.
- Викладачі, учасники, задачі та спроби подання розв’язку, основні таблиці, що зберігають інформацію про учасників олімпіад, викладачів, завдання та результати розв’язків.

Таблиця SCHOOLS (Навчальні заклади):

- поле id є автоматично згенерованим первинним ключем. Оскільки загальна кількість навчальних закладів не перевищує 1000, довжина поля обмежена 3 символами
- поле name містить назву навчального закладу та обмежене довжиною в 100 символів. Для унеможливлення дублювання назв, поле є унікальним.

Таблиця CLASSES (Класи):

- поле id є автоматично згенерованим первинним ключем. Оскільки кількість класів обмежена, довжина поля встановлена до 3 символів
- поле grade зберігає назву класу (наприклад, «10-А») і обмежене довжиною до 10 символів, що повністю покриває всі можливі варіанти.

Таблиця LANGUAGES (Мови програмування):

- поле id є автоматично згенерованим первинним ключем
- поле name містить назву мови програмування (наприклад, Python, C++) та обмежене довжиною в 20 символів. Поле є унікальним.

Таблиця TESTS (Задачі):

- поле id є автоматично згенерованим первинним ключем
- поле name містить назву задачі і є унікальним, довжина обмежена 100 символами
- поле description містить текстовий опис задачі довільної довжини.

Таблиця VERDICTS (Вердикти):

- поле id є автоматично згенерованим первинним ключем
- поле code містить короткий код вердикту (наприклад, AC, WA, CE), обмежене 10 символами, унікальне
- поле description містить опис вердикту, довжина до 100 символів.

Таблиця TEACHERS (Викладачі):

- поле id є автоматично згенерованим первинним ключем

- поле full_name зберігає ПІБ викладача, довжина обмежена 100 символами
- поле school_id є зовнішнім ключем на поле id у таблиці schools і вказує, до якого закладу належить викладач. При видаленні школи значення очищується (ON DELETE SET NULL)
- для уникнення дублювань в одному закладі застосовується обмеження унікальності на комбінацію full_name + school_id.

Таблиця PARTICIPANTS (Учасники):

- поле id є автоматично згенерованим первинним ключем
- поле full_name зберігає ПІБ учасника, довжина обмежена 100 символами
- поле class_id, зовнішній ключ на таблицю classes, що позначає клас учасника
- поле school_id, зовнішній ключ на таблицю schools, що позначає заклад, у якому навчається учасник
- поле teacher_id, зовнішній ключ на таблицю teachers, що позначає наставника учасника
- унікальність запису забезпечується комбінацією full_name, school_id та class_id, що унеможливлює дублювання в межах одного класу і школи.

Таблиця SUBMISSIONS (Спроби подання рішень):

- поле id є автоматично згенерованим первинним ключем
- поле participant_id є зовнішнім ключем на таблицю participants і вказує, хто подав розв’язок
- поле test_id, зовнішній ключ на таблицю tests і вказує, яке завдання було виконане
- поле language_id, зовнішній ключ на таблицю languages і вказує, якою мовою написано розв’язок

- поле score містить оцінку за задачу, тип INTEGER, значення повинне бути невід'ємним (перевірка score >= 0)
- поле code_text містить текст програми, довільної довжини
- поле submitted_at, дата подання розв'язку, за замовчуванням встановлена на 2001-01-01
- поле verdict_id, зовнішній ключ на таблицю verdicts, що позначає результат перевірки розв'язку
- обмеження унікальності (UNIQUE) на комбінацію participant_id, test_id, language_id, submitted_at запобігає повторному поданню одного й того ж рішення.

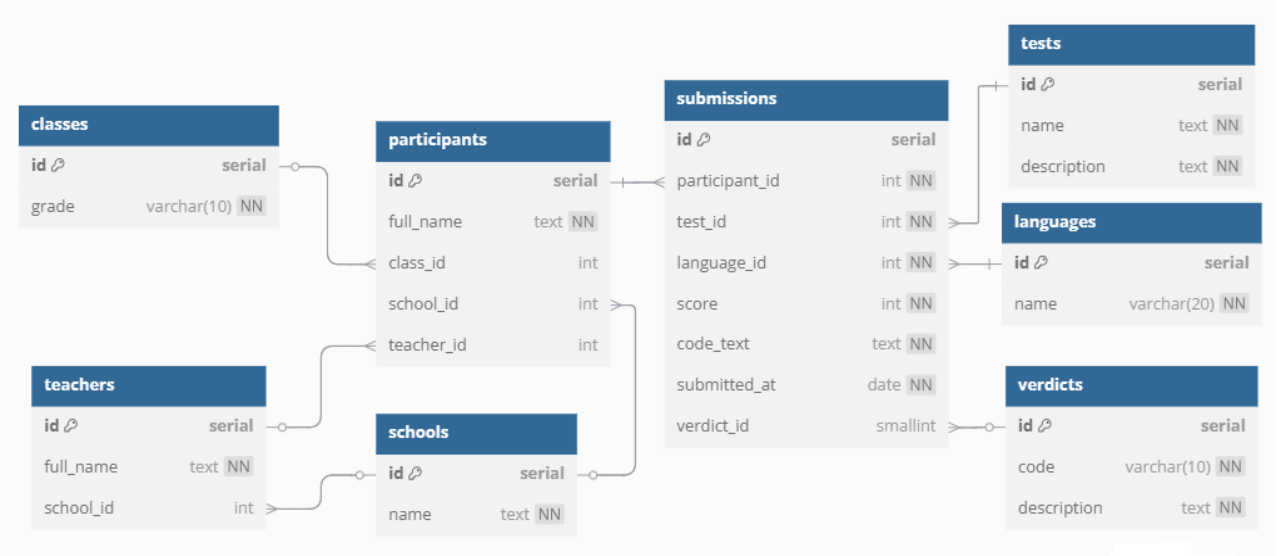

Рис. 2.2 - Деталогічна модель предметної області «Облік результатів учнівських олімпіад з програмування»

У ході проєктування логіки бази даних була перевірена відповідність її структури основним правилам нормалізації. Зокрема, було проаналізовано, чи перебуває вона у третій нормальній формі (3НФ), що є важливим для уникнення надлишковості та забезпечення узгодженості даних.

Перша нормальна форма (1НФ) передбачає, що всі поля таблиць повинні містити лише атомарні (тобто неподільні) значення. У нашій базі це дотримано: всі атрибути зберігають єдине значення, без масивів або списків. Наприклад,

поле full_name у таблицях teachers та participants містить лише одне повне ім’я, а не кілька варіантів.

Друга нормальна форма (2НФ) вимагає, щоб усі неключові атрибути залежали від усього первинного ключа, а не тільки від його частини. Це важливо, коли таблиця має складений ключ, однак у нашій базі скрізь використано прості ключі здебільшого поле id. Наприклад, у таблиці submissions усі поля, як-от score, verdict_id чи code_text, напряму пов’язані з єдиним первинним ключем — id.

Третя нормальна форма (3НФ) передбачає відсутність транзитивних залежностей тобто жоден неключовий атрибут не повинен залежати від іншого неключового атрибута. У структурі бази це реалізовано за допомогою зовнішніх ключів. Наприклад, у таблиці participants не зберігається напряму назва школи або класу тільки їхні school_id та class_id, які ведуть до відповідних записів у таблицях schools та classes. Це не лише зменшує дублювання, а й спрощує оновлення даних.

Загалом, після перевірки можна зробити висновок: структура бази відповідає всім трьом нормальним формам і є коректно нормалізованою, що дозволяє її ефективно використовувати в системі.

Крім цього, для контролю правильності та цілісності даних у базі були реалізовані такі обмеження:

- Унікальність значень. У тих полях, де не допускається повторень, встановлено обмеження UNIQUE. Наприклад, у таблиці teachers комбінація full_name + school_id повинна бути унікальною — це гарантує, що один і той самий викладач не буде зареєстрований кілька разів у межах одного закладу.
- Зовнішні ключі. Для зв’язку між таблицями використано зовнішні ключі. Наприклад, якщо видаляється запис з таблиці participants, система автоматично видаляє всі пов’язані спроби в submissions завдяки ON DELETE CASCADE. В інших випадках, як-от при

видаленні школи або викладача, застосовується ON DELETE SET NULL, що дозволяє зберегти історію, навіть якщо структура навчального закладу зміниться.

- Перевірка допустимих значень. У таблиці submissions встановлено обмеження CHECK, яке не дозволяє записувати від'ємні бали — значення в полі score має бути не менше нуля.
- Складені унікальні ключі. У деяких таблицях унікальність залежить не від одного, а від кількох полів. Наприклад, у таблиці participants комбінація full_name, school_id і class_id має бути унікальною — це виключає ситуації, коли той самий учасник випадково додається кілька разів у межах однієї школи й класу.

2.2. Проектування серверної логіки бази даних

2.2.1. Схема і об'єкти бази даних

Опис структури наведено нижче:

- schools (див. Додаток А.1):
  - id SERIAL PRIMARY KEY
  - name TEXT NOT NULL UNIQUE
- classes (див. Додаток А.2):
  - id SERIAL PRIMARY KEY
  - grade VARCHAR(10) NOT NULL UNIQUE
- languages (див. Додаток А.3):
  - id SERIAL PRIMARY KEY
  - name VARCHAR(20) NOT NULL UNIQUE
- tests (див. Додаток А.4):
  - id SERIAL PRIMARY KEY
  - name TEXT NOT NULL UNIQUE
  - description TEXT NOT NULL
- verdicts (див. Додаток А.5):

- id SERIAL PRIMARY KEY
- code VARCHAR(10) NOT NULL UNIQUE
- description TEXT NOT NULL

– teachers (див. Додаток А.6):
  - id SERIAL PRIMARY KEY
  - full_name TEXT NOT NULL
  - school_id INTEGER REFERENCES schools(id) ON DELETE SET NULL
  - Унікальність: full_name + school_id
– participants (див. Додаток А.7):
  - id SERIAL PRIMARY KEY
  - full_name TEXT NOT NULL
  - class_id INTEGER REFERENCES classes(id) ON DELETE SET NULL
  - school_id INTEGER REFERENCES schools(id) ON DELETE SET NULL
  - teacher_id INTEGER REFERENCES teachers(id) ON DELETE SET NULL
  - Унікальність: full_name + school_id + class_id
– submissions (див. Додаток А.8):
  - id SERIAL PRIMARY KEY
  - participant_id INTEGER NOT NULL REFERENCES participants(id) ON DELETE CASCADE
  - test_id INTEGER NOT NULL REFERENCES tests(id) ON DELETE CASCADE
  - language_id INTEGER NOT NULL REFERENCES languages(id) ON DELETE SET NULL
  - score INTEGER CHECK (score >= 0)
  - code_text TEXT NOT NULL

- submitted_at DATE NOT NULL DEFAULT '2001-01-01'
- verdict_id SMALLINT REFERENCES verdicts(id) ON DELETE SET NULL
- Унікальність: participant_id + test_id + language_id + submitted_at

2.2.2. Виконання запитів

У цій підсистемі ключову роль відіграють SQL-запити, які забезпечують доступ до даних, їх фільтрацію, агрегацію, об'єднання та аналіз. Загалом реалізовано 27 запитів, які охоплюють як прості вибірки, так і складніші запити з підзапитами, агрегатними функціями та з’єднаннями. Результати виконання кожного запиту наведено у додатках Б.1–Б.27.

На початковому етапі використовуються базові запити вибірки, що дозволяють отримати повну інформацію з окремих таблиць. Наприклад, для перегляду всіх навчальних закладів застосовується запит до таблиці schools (додаток Б.1). Існує також запит, який відбирає результати в межах заданого діапазону балів, зокрема від 70 до 90 (додаток Б.2).

У деяких випадках необхідно відібрати записи, що відповідають певному переліку значень. Наприклад, перелік класів з назвами, які входять до заданого списку (додаток Б.3). Реалізовано пошук за шаблоном, що дозволяє знаходити, наприклад, викладачів, чиї імена починаються на «Олена» (додаток Б.4).

Розроблено запити з логічними умовами, як-от вибірка учасників, які одночасно належать до визначеної школи та класу (додаток Б.5), або пошук записів, які задовольняють хоча б одну з кількох умов (додаток Б.6).

Щоб уникнути повторення, застосовуються запити на вибірку унікальних значень, наприклад, унікальні значення балів (додаток Б.7). Інші приклади включають пошук мінімального балу (додаток Б.8) або обчислення середнього значення (додаток Б.9).

Важливою є можливість підрахунку кількості записів. Наприклад, один із запитів визначає кількість зареєстрованих викладачів у системі (додаток Б.10).

Застосування агрегатних функцій у поєднанні з групуванням забезпечує більш глибокий аналіз. Наприклад, обчислення середнього балу для кожного викладача (додаток Б.11) або визначення кількості високих результатів, де бал перевищує 90, для кожного тесту (додаток Б.12). Якщо потрібно відібрати лише ті групи, де середній бал перевищує 80, використовується фільтрація за агрегованим значенням (додаток Б.13).

Складніший запит дозволяє обчислити середній бал по кожній мові програмування, враховуючи лише результати, що перевищують 50 балів, і впорядкувати їх за спаданням (додаток Б.14).

Важливою частиною є запити зі з’єднанням таблиць. Внутрішнє з’єднання між таблицями submissions та participants дає змогу отримати інформацію про кожну спробу разом з даними учасника (додаток Б.15). Ліве з’єднання застосовується для відображення всіх викладачів, включно з тими, у кого ще немає учасників або спроб (додаток Б.16). Праве з’єднання дозволяє показати всіх учасників, навіть тих, хто ще не має жодної спроби (додаток Б.17).

Запити на об'єднання часто містять додаткові умови. Наприклад, виведення всіх тестів, де учасники набрали понад 90 балів (додаток Б.18), або учасників, чиї імена починаються на «Іван», разом з назвами тестів, які вони виконували (додаток Б.19). Інші запити визначають середній бал по кожному тесту (додаток Б.20) або підраховують кількість учасників у кожній школі, якщо їх більше п’яти (додаток Б.21).

Підзапити забезпечують більш гнучку логіку. Наприклад, можна знайти учасників, які мають хоча б одну спробу з результатом, вищим за середній бал усіх спроб (додаток Б.22), або визначити спробу з найвищим результатом у системі (додаток Б.23).

Для перевірки наявності пов’язаних записів у підлеглих таблицях застосовується конструкція EXISTS. Наприклад, пошук учасників, які мають хоча б одну спробу

(додаток Б.24). Запити з ANY або SOME дозволяють відібрати спроби з балами, що перевищують хоча б один результат за певною мовою програмування (додаток Б.25).

Також можна визначити всіх викладачів, у яких є учні, використовуючи підзапит із перевіркою входження (додаток Б.26). Один із найскладніших запитів поєднує підзапит і з’єднання для того, щоб вивести учасників із хоча б однією незадовільною спробою (менше 60 балів) разом з назвами тестів, які вони виконували (додаток Б.27).

РОЗДІЛ 3
ОПИС КЛІЄНТСЬКО-СЕРВЕРНОЇ АРХІТЕКТУРИ ДОДАТКУ

3.1. Архітектура клієнтської частини

Клієнтська частина додатку розроблена на основі фреймворка Laravel версії 10.x із шаблонізатором Blade та інструментом збірки фронтенду Vite. Для стилізації інтерфейсу застосовано Bootstrap 5, динамічні елементи керуються за допомогою jQuery та чистого JavaScript.

Клієнтська частина взаємодіє із сервером через REST-подібні HTTP-запити та отримує дані у форматі JSON.

Мінімальні технічні вимоги:

- Операційна система: Windows 10 (64-bit) або новіша, Linux (наприклад, Ubuntu 20.04+)
- Процесор: Intel Core i3 (2-го покоління або новіший) чи еквівалентний AMD
- Оперативна пам’ять: щонайменше 4 ГБ
- Браузер: будь-який сучасний веббраузер з підтримкою HTML5, CSS3 та ECMAScript 6 (наприклад, Google Chrome, Mozilla Firefox, Microsoft Edge, Safari)
- Інтернет-з’єднання: стабільне підключення зі швидкістю від 1 Мбіт/с

У системі реалізовано розмежування прав доступу користувачів за допомогою механізму RBAC (Role-Based Access Control). Це дозволяє чітко визначити доступний функціонал для кожної категорії користувачів. Всього в системі реалізовано дві основні ролі: адміністратор і гість.

Для користувачів, які мають роль «гість» було розроблено окремий інтерфейс (див. Додаток В.1), що складається з чотирьох основних функціональних розділів.

Перший розділ це перегляд інформації про учасників (див. Додаток В.2). У ньому відображаються основні відомості про школярів, зокрема прізвище, ім’я

та по батькові, навчальний клас, навчальний заклад, який вони представляють, а також ім’я вчителя, який їх підготував. Цей розділ дає змогу швидко орієнтуватися в складі учасників і шукати необхідні дані за конкретною особою чи закладом освіти. Крім того, реалізована функція перегляду детальної інформації про кожного учня. При натисканні на конкретного учасника відкривається окрема сторінка (див. Додаток В.3), де відображається як базова інформація про учня, так і історія його спроб у тестах. Кожна спроба включає назву завдання, мову програмування, кількість набраних балів, дату подання розв’язку та вердикт системи перевірки.

Другий розділ стосується перегляду всіх спроб учасників (див. Додаток В.4). Кожна спроба містить дані про набрані бали, програмний код, який було подано на перевірку, а також мову програмування, якою користувався учасник під час виконання завдання. Для перегляду детальнішої інформації про спробу конкретного учасника у системі реалізовано окрему сторінку (див. Додаток В.5), яка відкривається після вибору відповідного запису зі списку спроб. На цій сторінці, наведено повну інформацію про вибрану спробу: ім’я та прізвище учня, мову програмування, кількість набраних балів, а також дату подання рішення. У центральній частині інтерфейсу розміщено інформаційний блок із назвою завдання та його коротким описом. Нижче розташовано текст вихідного коду, який був поданий учасником на перевірку. Такий формат представлення дозволяє легко переглядати зміст програмного рішення, що є зручним для аналізу якості розв’язку, стилю кодування або виявлення типових помилок.

Третій доступний розділ, вчителі (див. Додаток В.6). Тут надається можливість ознайомитися з переліком педагогів, які готували учнів до олімпіади. Для кожного вчителя відображається його ПІБ, заклад освіти та список учнів, пов’язаних з ним. Окрім загального списку викладачів, система дозволяє переглянути історію спроб учнів конкретного педагога (див. Додаток В.7). Ця функціональність відкривається після вибору певного вчителя зі списку. На сторінці відображаються загальні відомості про викладача, його ПІБ та

навчальний заклад, а також таблиця з інформацією про всі спроби, які здійснили його учні. Кожен запис у таблиці містить ПІБ учасника, клас, мову програмування, кількість балів та дату подання рішення.

Четвертий функціональний розділ аналітика (див. Додаток В.8). У цьому розділі користувач може переглянути звіти за різними напрямками.

На вкладці «Учасники», відображається загальна кількість школярів, які взяли участь в олімпіаді, а також визначається найпопулярніша мова програмування (у прикладі це Ruby). Нижче наведено два рейтинги: ТОП-5 учасників за середнім балом та ТОП-5 за максимальним балом, що дає змогу швидко оцінити результати найуспішніших школярів. Окрім цього, у розділі також доступні вкладки «Вчителі», «Спроби» та «Тести», які надають аналітичну інформацію відповідно про наставників, подані розв'язки та окремі завдання олімпіади.

На всіх сторінках інтерфейсу доступні функції пошуку, сортування та фільтрації, що значно полегшує роботу з великими обсягами даних. Наприклад, можна швидко знайти всі спроби учня з певної школи або відсортувати список учасників за алфавітом чи рівнем набраних балів. Проте варто зазначити, що користувачі з цією роллю не мають можливості створювати нові записи, редагувати наявні або видаляти дані. Така обмеженість у правах доступу є навмисною і забезпечує надійний захист бази даних від несанкціонованого втручання.

Для користувачів, які мають роль адміністратора, у системі реалізовано розширений інтерфейс з повним доступом до всіх функцій платформи. Адміністратор має можливість не лише переглядати інформацію, як гість, але й здійснювати управління даними: додавати, редагувати та видаляти записи, а також завантажувати нові дані до бази.

На головній сторінці інтерфейсу адміністратора (див. Додаток В.9) відображені шість основних функціональних розділів, до двох з який мають доступ тільки адміністратори, а саме:

– Імпорт даних (див. Додаток В.10): окремий розділ, який дозволяє завантажувати інформацію у систему з дані з файлів у форматі JSON. Це значно полегшує початкове заповнення бази даних або її оновлення великими обсягами інформації.
– Управління даними (див. Додаток В.11): надає доступ до редагування таблиць бази даних безпосередньо з інтерфейсу. Через цей розділ адміністратор може швидко вносити зміни у будь-яку категорію даних, що присутні в системі.

3.2. Серверна взаємодія клієнтської частини з базою даних через API

У розробленій системі обліку результатів учнівських олімпіад з програмування клієнтська частина взаємодіє з сервером через REST API. Це дозволяє забезпечити ефективну та уніфіковану комунікацію між інтерфейсом користувача та серверною частиною, яка, у свою чергу, звертається до бази даних.

Серверна логіка реалізована за допомогою середовища виконання Node.js, із застосуванням фреймворку Express, що спрощує створення маршрутизаторів та обробку HTTP-запитів.

REST API побудований відповідно до принципів CRUD (Create, Read, Update, Delete), які відповідають базовим операціям над даними. Кожен API-запит повертає результат у форматі JSON, що спрощує обробку інформації на клієнтській стороні, написаній із використанням JavaScript та фреймворку Bootstrap.

Приклад API-запитів які використовуються в предметній області:

- GET /api/participants

  Повертає список усіх учасників олімпіади з основною інформацією: прізвище, ім’я та по батькові, школа, клас, кількість спроб.
- POST /api/submissions

Додає нову спробу учасника. На вхід приймає JSON-об'єкт, що містить такі параметри: мова програмування, кількість набраних балів, дата подання, ідентифікатор учасника та ідентифікатор завдання.

- GET /api/teachers/:id

  Повертає повну інформацію про викладача за його унікальним ідентифікатором. Результат містить також пов’язану школу та список учнів, яких він підготував.

- PUT /api/tests/:id

  Оновлює дані тесту (завдання) за його ідентифікатором. JSON-об’єкт, який передається у запиті, може містити оновлені поля: назву завдання, опис, максимальну кількість балів тощо.

- GET /api/analytics/tests

  Повертає зведену статистику щодо проходження тестів усіма учасниками. Містить дані про середній відсоток успішності, кількість спроб, найчастіше використовувані мови програмування.

ВИСНОВКИ

У результаті виконання курсової роботи було повністю реалізовано інформаційну систему для обліку та аналізу результатів учнівських олімпіад з програмування. На початковому етапі детально проаналізовано предметну область: вивчено процедури реєстрації та оцінювання учасників, а також розглянуто можливості й обмеження популярних онлайн-платформ, таких як Codeforces, AtCoder, LeetCode та E-olymp. Цей аналіз дозволив обґрунтувати доцільність створення власної системи, яка б забезпечувала цілісність даних, уникнення дублювання записів і можливість масштабування при збільшенні обсягу інформації.

Далі було спроєктовано реляційну базу даних у середовищі PostgreSQL з дотриманням принципів нормалізації до третьої нормальної форми. Структура БД включає ключові сутності: учасники, викладачі, навчальні заклади, класи, тести, спроби, мови програмування та вердикти. Зв’язки між ними реалізовано через первинні та зовнішні ключі, що гарантує логічну цілісність даних і високу продуктивність запитів.

Серверна частина системи реалізована на основі Node.js з використанням фреймворку Express. Побудований REST API підтримує повний набір CRUD-операцій. API дозволяє отримувати списки учасників і викладачів, переглядати детальну інформацію про тестові спроби, редагувати параметри завдань та формувати аналітичні звіти — зокрема щодо середньої успішності проходження тестів та популярності мов програмування.

Клієнтську частину розроблено у середовищі Laravel із застосуванням шаблонізатора Blade для генерації HTML-сторінок. Для стилізації інтерфейсу використано Bootstrap, що забезпечує адаптивність і сучасний вигляд елементів. У розділі «Аналітика» інтегровано бібліотеку Chart.js для побудови графіків різного типу — кругових, лінійних та радарних. Інтерфейс містить основні функціональні розділи: «Учасники», «Спроби», «Вчителі», «Аналітика»,

«Імпорт даних» та «Управління даними», кожен із яких представлений у вигляді зручних табличних форм.

Усі модулі підтримують ключові можливості взаємодії з даними: пошук, фільтрацію, сортування та пагінацію. Це суттєво покращує зручність користування, особливо при роботі з великими обсягами інформації. Інтерфейс системи відповідає сучасним вимогам до вебзастосунків — він зрозумілий, функціональний і придатний до подальшого розширення.

Розроблена система має вагоме практичне значення. Вона дозволяє автоматизувати обробку результатів олімпіад, спрощує підготовку звітів і сприяє формуванню аналітичних та методичних матеріалів із мінімальними витратами часу.

Отже, в межах курсової роботи було здійснено повний цикл розробки: від аналізу предметної області до реалізації бази даних, створення серверної та клієнтської частин, тестування і початкової оптимізації. Усі поставлені завдання виконано в повному обсязі. Система готова до впровадження в реальних умовах і має потенціал для подальшого розвитку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розломій І.О. Методичні вказівки до виконання та оформлення курсової роботи з дисципліни «Організація баз даних і знань», 2022. – 30 с.
2. Гладкий Ю. М. Проєктування інформаційних систем: навч. посібник. – Київ: Ліра-К, 2020. – 320 с.
3. Date C. J. Основи систем баз даних. – 8-е вид. – К.: Діалектика, 2019. – 880 с.
4. Сидоренко В. О. Бази даних. Проектування та реалізація: навч. посібник. – Харків: ХНУРЕ, 2021. – 278 с.
5. Соловей В. І. Інформаційні технології управління навчальними процесами. – Львів: ЛНУ ім. Івана Франка, 2020. – 234 с.
6. Крутько С. І., Романенко Т. В. Web-програмування: HTML, CSS, JavaScript, PHP, MySQL. – Київ: Університет «Україна», 2022. – 256 с.
7. Laravel. Офіційна документація [Електронний ресурс]. – URL: https://laravel.com/docs – Перевірено: 06.04.2025.
8. Bootstrap. Офіційна документація [Електронний ресурс]. – URL: https://getbootstrap.com/ – Перевірено: 24.04.2025.
9. PostgreSQL. Official Documentation [Електронний ресурс]. – URL: https://www.postgresql.org/docs/ – Перевірено: 15.04.2025.
10. Chart.js. Documentation [Електронний ресурс]. – URL: https://www.chartjs.org/docs/latest/ – Перевірено: 09.05.2025.

ДОДАТОК А. ЕКРАННІ ФОРМИ ТАБЛИЦЬ

	id [PK] integer	name text
1	1	Ліцей №1 імені Тараса Шевченка
2	2	Гімназія "Інтелект"
3	3	Школа №23
4	4	Спеціалізована школа №5
5	5	Ліцей "Науковий"
6	6	Загальноосвітня школа №14
7	7	Гімназія "Гармонія"
8	8	Школа №9 імені Лесі Українки
9	9	Ліцей "Платон"
10	10	Школа мистецтв №3
11	11	Гімназія №12
12	12	ЗОШ №17
13	13	Навчально-виховний комплекс "Паросток"
14	14	Школа №4
15	15	Ліцей "Прем'єр"

Рис. А.1 – Таблиця schools

	id [PK] integer	grade character varying (10)
1	1	5-А
2	2	5-Б
3	3	6-А
4	4	6-Б
5	5	7-А
6	6	7-Б
7	7	8-А
8	8	8-Б
9	9	9-А
10	10	9-Б
11	11	10-А
12	12	10-Б
13	13	11-А
14	14	11-Б
15	15	12-А

Рис. А.2 – Таблиця **classes**

	id [PK] integer	name character varying (20)
1	1	Python
2	2	Java
3	3	C++
4	4	JavaScript
5	5	Ruby
6	6	Go
7	7	Swift
8	8	Kotlin
9	9	PHP
10	10	TypeScript
11	11	Scala
12	12	Perl
13	13	Rust
14	14	Pascal
15	15	Haskell

Рис. А.3 – Таблиця languages

	id [PK] integer	name text	description text
1	1	Тест 1	Основи програмування на Python
2	2	Тест 2	Масиви та цикли
3	3	Тест 3	Алгоритми сортування
4	4	Тест 4	Рекурсія
5	5	Тест 5	Об'єктно-орієнтоване програмування
6	6	Тест 6	Робота з файлами
7	7	Тест 7	Основи баз даних
8	8	Тест 8	Алгоритми на графах
9	9	Тест 9	Робота з API
10	10	Тест 10	Тестування коду
11	11	Тест 11	Мережеве програмування
12	12	Тест 12	Завдання на логіку
13	13	Тест 13	Програмування на C++
14	14	Тест 14	Frontend-розробка
15	15	Тест 15	Оптимізація алгоритмів

Рис. А.4 – Таблиця tests

	id [PK] integer	code character varying (10)	description text
1	1	AC	Прийнято
2	2	WA	Неправильна відповідь
3	3	TLE	Перевищено ліміт часу
4	4	MLE	Перевищено ліміт пам'яті
5	5	RE	Помилка виконання
6	6	CE	Помилка компіляції
7	7	PE	Помилка формату виводу
8	8	TL	Час очікування перевищено
9	9	SK	Пропущено
10	10	OK	Виконано успішно
11	11	IE	Внутрішня помилка
12	12	NO	Немає відповіді
13	13	DE	Заборонена дія
14	14	SE	Системна помилка
15	15	BR	Помилкове рішення

Рис. А.5 – Таблиця verdicts

	id [PK] integer	full_name text	school_id integer
1	1	Олена Іваненко	1
2	2	Микола Ковальчук	2
3	3	Ірина Павленко	3
4	4	Сергій Петренко	4
5	5	Ганна Ткаченко	5
6	6	Оксана Лисенко	6
7	7	Віталій Мельник	7
8	8	Людмила Шевченко	8
9	9	Роман Дяченко	9
10	10	Катерина Козак	10
11	11	Олег Сидоренко	11
12	12	Наталя Жукова	12
13	13	Марина Савченко	13
14	14	Ігор Степаненко	14
15	15	Тетяна Хоменко	15

Рис. А.6 – Таблиця teachers

	id [PK] integer	full_name text	class_id integer	school_id integer	teacher_id integer
1	1	Андрій Шевчук	1	1	1
2	2	Ірина Бондаренко	2	2	2
3	3	Максим Ткачук	3	3	3
4	4	Юлія Кравченко	4	4	4
5	5	Олександр Левченко	5	5	5
6	6	Софія Коваль	6	6	6
7	7	Богдан Петренко	7	7	7
8	8	Наталія Савчук	8	8	8
9	9	Владислав Зінченко	9	9	9
10	10	Катерина Омельченко	10	10	10
11	11	Олексій Дяченко	11	11	11
12	12	Анна Литвиненко	12	12	12
13	13	Євгенія Харченко	13	13	13
14	14	Марія Поліщук	14	14	14
15	15	Денис Романенко	15	15	15

Рис. А.7 – Таблиця participants

	id [PK] integer	participant_id integer	test_id integer	language_id integer	score integer	code_text text	submitted_at date	verdict_id smallint
1	1	1	1	1	95	print("Привіт, світ!")	2025-01-01	1
2	2	2	2	2	88	for i in range(5): print(i)	2025-01-02	1
3	3	3	3	3	70	int main() { return 0; }	2025-01-03	2
4	4	4	4	4	100	function test() { return true; }	2025-01-04	1
5	5	5	5	5	60	def main(): pass	2025-01-05	3
6	6	6	6	6	90	fmt.Println("Hi")	2025-01-06	1
7	7	7	7	7	77	print("File opened")	2025-01-07	1
8	8	8	8	8	85	fun main() {}	2025-01-08	1
9	9	9	9	9	92	<?php echo "Привіт"; ?>	2025-01-09	1
10	10	10	10	10	75	console.log("Test")	2025-01-10	2
11	11	11	11	11	65	val x = 1	2025-01-11	4
12	12	12	12	12	93	print "Success"	2025-01-12	1
13	13	13	13	13	99	fn main() {}	2025-01-13	1
14	14	14	14	14	84	begin writeln("Hi"); end.	2025-01-14	1
15	15	15	15	15	58	main = putStrLn "Hi"	2025-01-15	3

Рис. А.8 – Таблиця submissions

ДОДАТОК Б. ВИКОНАННЯ ЗАПИТІВ

```
1  SELECT * FROM schools;
```

Data Output | Messages | Notifications

	id [PK] integer	name text
1	1	Ліцей №1 імені Тараса Шевченка
2	2	Гімназія "Інтелект"
3	3	Школа №23
4	4	Спеціалізована школа №5
5	5	Ліцей "Науковий"
6	6	Загальноосвітня школа №14
7	7	Гімназія "Гармонія"
8	8	Школа №9 імені Лесі Українки
9	9	Ліцей "Платон"
10	10	Школа мистецтв №3
11	11	Гімназія №12
12	12	ЗОШ №17
13	13	Навчально-виховний комплекс "Паросток"
14	14	Школа №4
15	15	Ліцей "Прем'єр"

Рис. Б.1 – Результат роботи запиту schools_list_all

```
1  SELECT * FROM submissions WHERE score BETWEEN 70 AND 90;
```

Data Output | Messages | Notifications

Showing rows: 1 to 7

	id [PK] integer	participant_id integer	test_id integer	language_id integer	score integer	code_text text	submitted_at date	verdict_id smallint
1	2	2	2	2	88	for i in range(5): print(i)	2025-01-02	1
2	3	3	3	3	70	int main() { return 0; }	2025-01-03	2
3	6	6	6	6	90	fmt.Println("Hi")	2025-01-06	1
4	7	7	7	7	77	print("File opened")	2025-01-07	1
5	8	8	8	8	85	fun main() {}	2025-01-08	1
6	10	10	10	10	75	console.log("Test")	2025-01-10	2
7	14	14	14	14	84	begin writeln('Hi'); en...	2025-01-14	1

Рис. Б.2 – Результат роботи запиту submissions_score_between_70_90

```
1  SELECT * FROM classes WHERE grade IN ('5-А', '6-Б', '7-В');
```

Data Output | Messages | Notifications

	id [PK] integer	grade character varying (10)
1	1	5-А
2	4	6-Б

Рис. Б.3 – Результат роботи запиту classes_selected_by_grades

```
1  SELECT * FROM teachers WHERE full_name LIKE 'Олена%';
```

Data Output | Messages | Notifications

	id [PK] integer	full_name text	school_id integer
1	1	Олена Іваненко	1

Рис. Б.4 – Результат роботи запиту teachers_name_starts_with_olena

```
1  SELECT * FROM participants WHERE school_id = 1 AND class_id = 1;
```

Data Output | Messages | Notifications

	id [PK] integer	full_name text	class_id integer	school_id integer	teacher_id integer
1	1	Андрій Шевчук	1	1	1

Рис. Б.5 – Результат роботи запиту participants_by_school_and_class

```
1  SELECT * FROM tests WHERE name = 'Тест 1' OR name = 'Контрольна робота';
```

Data Output | Messages | Notifications

	id [PK] integer	name text	description text
1	1	Тест 1	Основи програмування на Python

Рис. Б.6 – Результат роботи запиту tests_named_olympiad_or_control

```
1  SELECT DISTINCT score FROM submissions;
```

Data Output | Messages | Notifications

	score integer
1	70
2	84
3	92
4	60
5	65
6	88
7	85

Рис. Б.7 – Результат роботи запиту distinct_scores_in_submissions

```
1  SELECT MIN(score) AS мін_бал FROM submissions;
```

Data Output | Messages | Notifications

	мін_бал integer
1	58

Рис. Б.8 – Результат роботи запиту min_score_overall

```
1  SELECT AVG(score) AS середній_бал FROM submissions;
```

Data Output | Messages | Notifications

	середній_бал numeric
1	82.0666666666666667

Рис. Б.9 – Результат роботи запиту average_score_all

```
1  SELECT COUNT(*) AS кількість_вчителів FROM teachers;
```

Data Output | Messages | Notifications

	кількість_вчителів bigint
1	15

Рис. Б.10 – Результат роботи запиту total_teachers_count

```
1  SELECT participant_id, AVG(score) FROM submissions GROUP BY participant_id;
```

Data Output | Messages | Notifications

	participant_id integer	avg numeric
1	11	65.0000000000000000
2	9	92.0000000000000000
3	15	58.0000000000000000
4	3	70.0000000000000000

Рис. Б.11 – Результат роботи запиту avg_score_by_participant

```
1  SELECT test_id, COUNT(*) FROM submissions WHERE score >= 90 GROUP BY test_id;
```

Data Output | Messages | Notifications

	test_id integer	count bigint
1	9	1
2	4	1
3	6	1
4	13	1
5	12	1
6	1	1

Рис. Б.12 – Результат роботи запиту high_scores_count_by_test

```
1  SELECT test_id, AVG(score) FROM submissions GROUP BY test_id HAVING AVG(score) > 80;
```

Data Output | Messages | Notifications

	test_id integer	avg numeric
1	9	92.0000000000000000
2	4	100.0000000000000000
3	6	90.0000000000000000
4	14	84.0000000000000000
5	13	99.0000000000000000
6	2	88.0000000000000000
7	12	93.0000000000000000
8	1	95.0000000000000000
9	8	85.0000000000000000

Рис. Б.13 – Результат роботи запиту tests_with_avg_score_above_80

```sql
SELECT language_id, AVG(score) AS avg_score
FROM submissions
WHERE score >= 50
GROUP BY language_id
HAVING AVG(score) > 90
ORDER BY avg_score DESC;
```

Data Output | Messages | Notifications

	language_id integer	avg_score numeric
1	4	100.0000000000000000
2	13	99.0000000000000000
3	1	95.0000000000000000
4	12	93.0000000000000000
5	9	92.0000000000000000

Рис. Б.14 – Результат роботи запиту languages_avg_score_gt_60

```sql
SELECT s.id, s.score, p.full_name
FROM submissions s
INNER JOIN participants p ON s.participant_id = p.id;
```

Data Output | Messages | Notifications

	id integer	score integer	full_name text
1	1	95	Андрій Шевчук
2	2	88	Ірина Бондаренко
3	3	70	Максим Ткачук
4	4	100	Юлія Кравченко
5	5	60	Олександр Левченко
6	6	90	Софія Коваль
7	7	77	Богдан Петренко
8	8	85	Наталія Савчук

Рис. Б.15 – Результат роботи запиту join_submissions_with_participants

```
1 SELECT t.full_name, s.score
2 FROM teachers t
3 LEFT JOIN participants p ON t.id = p.teacher_id
4 LEFT JOIN submissions s ON p.id = s.participant_id;
```

Data Output | Messages | Notifications

	full_name text	score integer
1	Олена Іваненко	95
2	Микола Ковальчук	88
3	Ірина Павленко	70
4	Сергій Петренко	100
5	Ганна Ткаченко	60
6	Оксана Лисенко	90
7	Віталій Мельник	77
8	Людмила Шевченко	85

Рис. Б.16 – Результат роботи запиту teachers_with_or_without_results

```
1 SELECT p.full_name, s.score
2 FROM submissions s
3 RIGHT JOIN participants p ON s.participant_id = p.id;
```

Data Output | Messages | Notifications

	full_name text	score integer
1	Андрій Шевчук	95
2	Ірина Бондаренко	88
3	Максим Ткачук	70
4	Юлія Кравченко	100
5	Олександр Левченко	60
6	Софія Коваль	90
7	Богдан Петренко	77

Рис. Б.17 – Результат роботи запиту all_participants_even_without_scores

```sql
SELECT t.name, s.score
FROM tests t
INNER JOIN submissions s ON t.id = s.test_id
WHERE s.score >= 90;
```

Data Output | Messages | Notifications

	name text	score integer
1	Тест 1	95
2	Тест 4	100
3	Тест 6	90
4	Тест 9	92
5	Тест 12	93
6	Тест 13	99

Рис. Б.18 – Результат роботи запиту tests_with_scores_90_or_more

```sql
SELECT p.full_name, t.name
FROM participants p
INNER JOIN submissions s ON p.id = s.participant_id
INNER JOIN tests t ON s.test_id = t.id
WHERE p.full_name LIKE 'Богдан%';
```

Data Output | Messages | Notifications

	full_name text	name text
1	Богдан Петренко	Тест 7

Рис. Б.19 – Результат роботи запиту participants_named_bohdan_with_tests

```sql
SELECT t.name, AVG(s.score)
FROM tests t
INNER JOIN submissions s ON t.id = s.test_id
GROUP BY t.name;
```

Data Output | Messages | Notifications

	name text	avg numeric
1	Тест 14	84.0000000000000000
2	Тест 11	65.0000000000000000
3	Тест 12	93.0000000000000000
4	Тест 15	58.0000000000000000
5	Тест 10	75.0000000000000000

Рис. Б.20 – Результат роботи запиту avg_score_per_test

```sql
SELECT sch.name, COUNT(p.id)
FROM schools sch
INNER JOIN participants p ON sch.id = p.school_id
GROUP BY sch.name
HAVING COUNT(p.id) >= 1;
```

Data Output | Messages | Notifications

	name text	count bigint
1	Ліцей №1 імені Тараса Шевченка	1
2	Школа мистецтв №3	1
3	Школа №4	1
4	Гімназія "Гармонія"	1
5	Ліцей "Платон"	1

Рис. Б.21 – Результат роботи запиту schools_with_1_plus_participants

```sql
SELECT full_name FROM participants WHERE id IN (
  SELECT participant_id FROM submissions
  WHERE score > (SELECT AVG(score) FROM submissions)
);
```

Data Output | Messages | Notifications

	full_name text
1	Андрій Шевчук
2	Ірина Бондаренко
3	Юлія Кравченко
4	Софія Коваль

Рис. Б.22 – Результат роботи запиту participants_above_avg_score

```sql
SELECT * FROM submissions
WHERE score = (SELECT MAX(score) FROM submissions);
```

Data Output | Messages | Notifications

Showing rows: 1 to 1

	id [PK] integer	participant_id integer	test_id integer	language_id integer	score integer	code_text text	submitted_at date	verdict_id smallint
1	4	4	4	4	100	function test() { return true; }	2025-01-04	1

Рис. Б.23 – Результат роботи запиту submissions_with_max_score

```
SELECT * FROM participants p
WHERE EXISTS (SELECT 1 FROM submissions s WHERE s.participant_id = p.id);
```

Data Output | Messages | Notifications

	id [PK] integer	full_name text	class_id integer	school_id integer	teacher_id integer
1	1	Андрій Шевчук	1	1	1
2	2	Ірина Бондаренко	2	2	2
3	3	Максим Ткачук	3	3	3
4	4	Юлія Кравченко	4	4	4
5	5	Олександр Левченко	5	5	5

Рис. Б.24 – Результат роботи запиту participants_with_submissions

```
SELECT * FROM submissions
WHERE score > ANY (SELECT score FROM submissions WHERE language_id = 1);
```

Data Output | Messages | Notifications

Showing rows: 1 to 2

	id [PK] integer	participant_id integer	test_id integer	language_id integer	score integer	code_text text	submitted_at date	verdict_id smallint
1	4	4	4	4	100	function test() { return true; }	2025-01-04	1
2	13	13	13	13	99	fn main() {}	2025-01-13	1

Рис. Б.25 – Результат роботи запиту better_than_some_language_1

```
SELECT * FROM teachers
WHERE id IN (SELECT teacher_id FROM participants);
```

Data Output | Messages | Notifications

	id [PK] integer	full_name text	school_id integer
1	1	Олена Іваненко	1
2	2	Микола Ковальчук	2
3	3	Ірина Павленко	3
4	4	Сергій Петренко	4
5	5	Ганна Ткаченко	5

Рис. Б.26 – Результат роботи запиту teachers_having_students

```sql
SELECT p.full_name, t.name FROM participants p
JOIN submissions s ON p.id = s.participant_id
JOIN tests t ON s.test_id = t.id
WHERE p.id IN (
  SELECT participant_id FROM submissions WHERE score < 60
);
```

Data Output | Messages | Notifications

	full_name text	name text
1	Денис Романенко	Тест 15

Рис. Б.27 – Результат роботи запиту participants_below_60_with_tests

ДОДАТОК В. ЕКРАННИЙ ВИГЛЯД КЛІЄНТСЬКОГО ДОДАТКУ

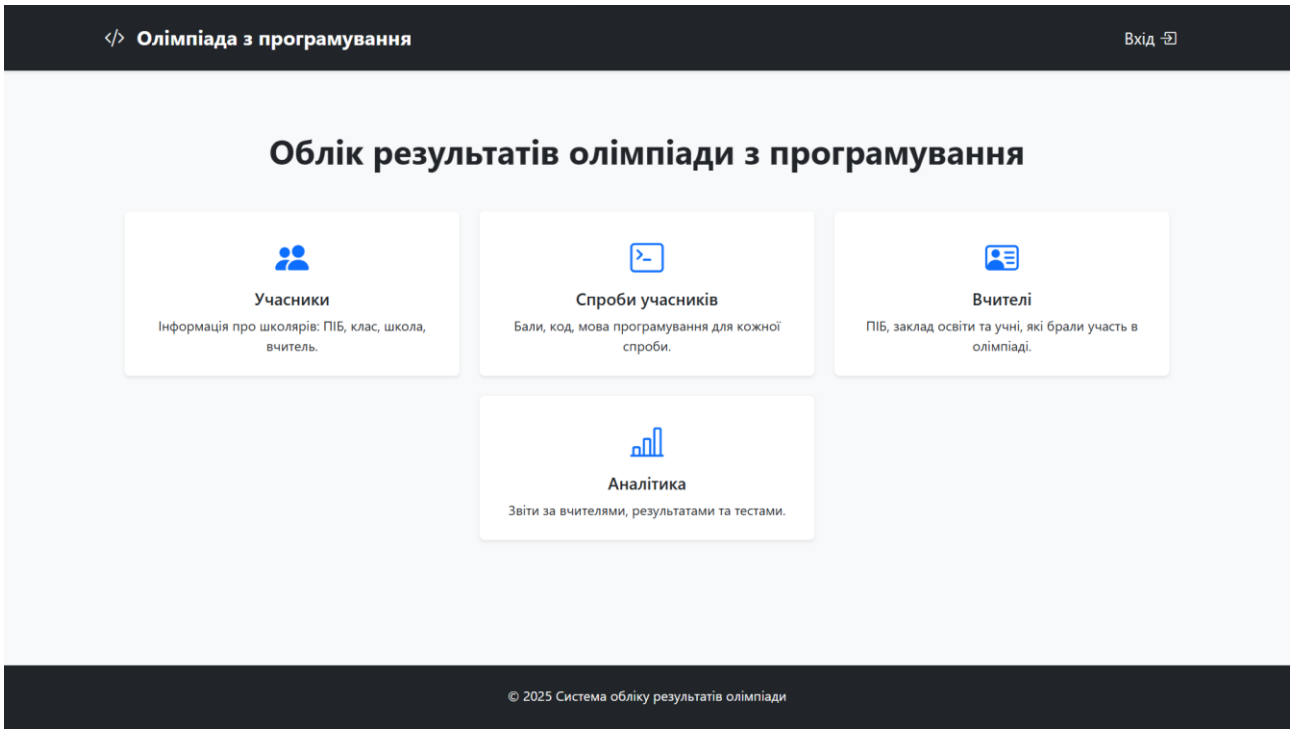

Рис. В.1 – Інтерфейс головної сторінки для користувача з роллю «Гість»

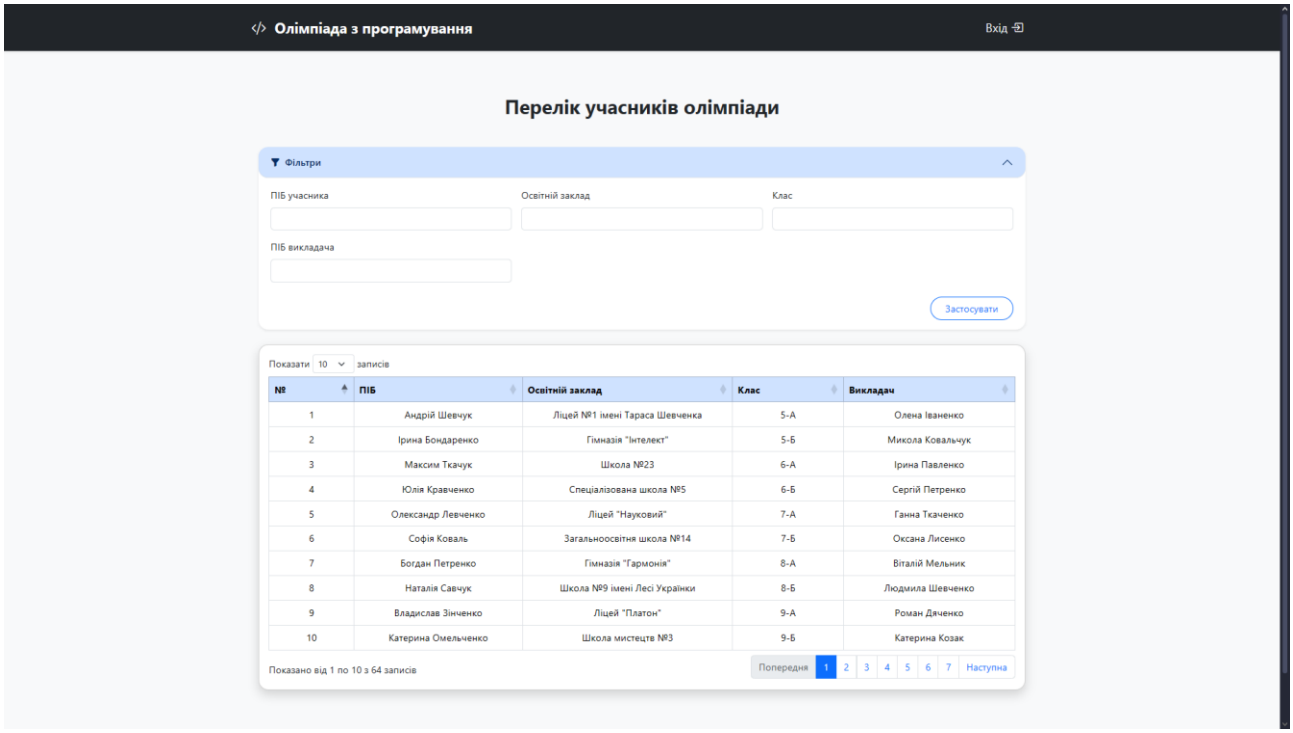

Рис. В.2 – Інтерфейс розділу «Учасники»

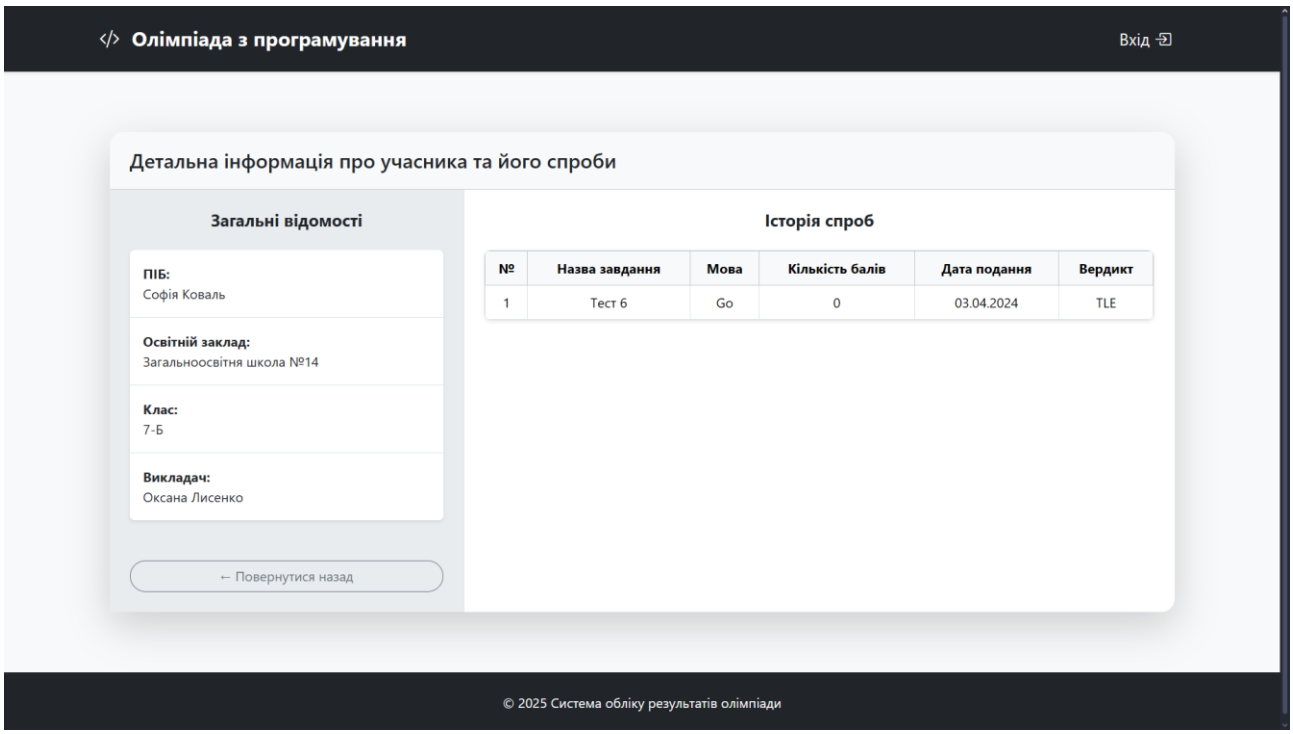

Рис. В.3 – Детальна інформація про учасника та історія його спроб

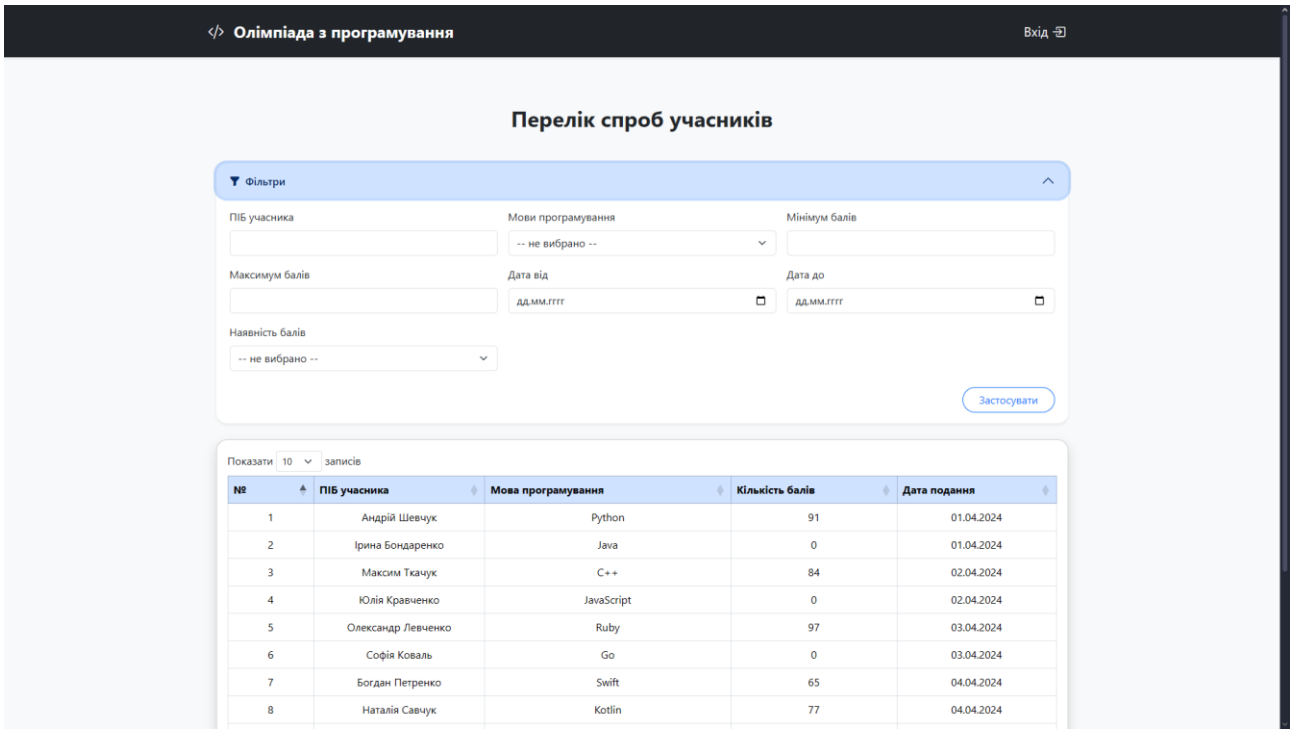

Рис. В.4 – Інтерфейс розділу «Перелік спроб учасників»

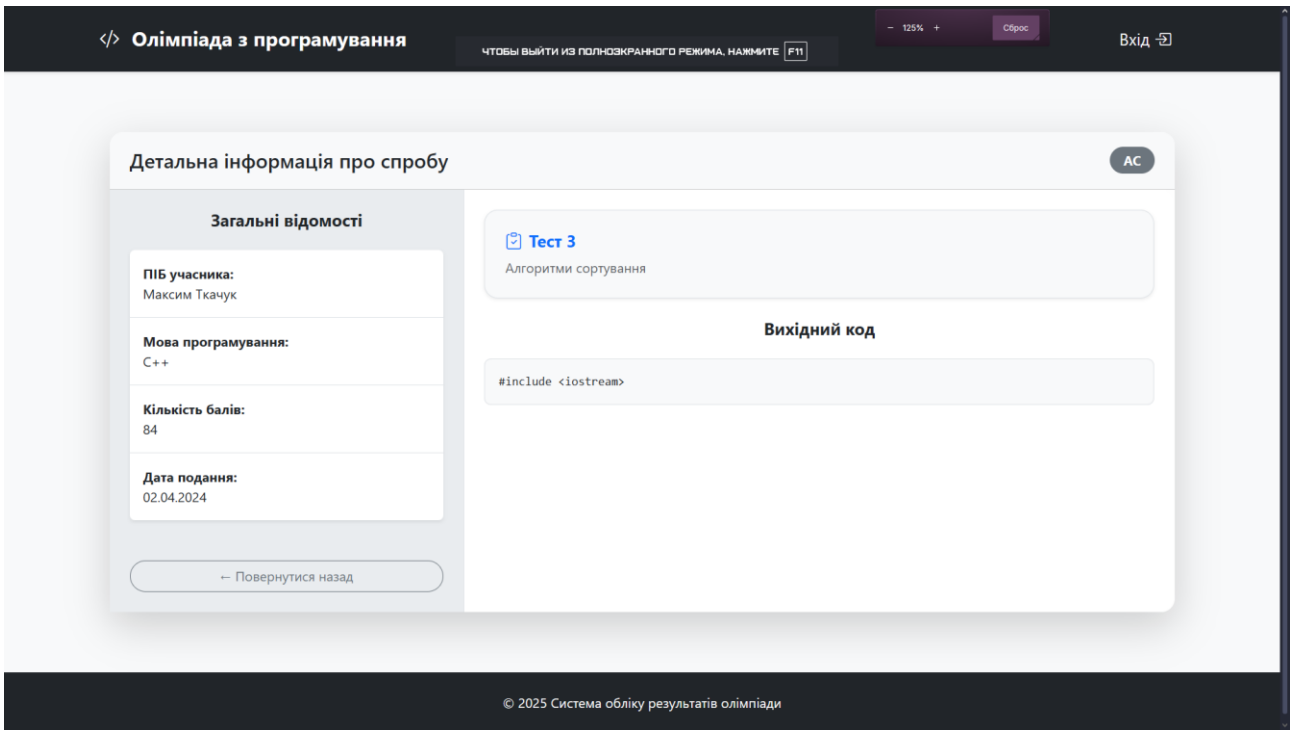

Рис. В.5 – Детальна інформація про спробу учасника

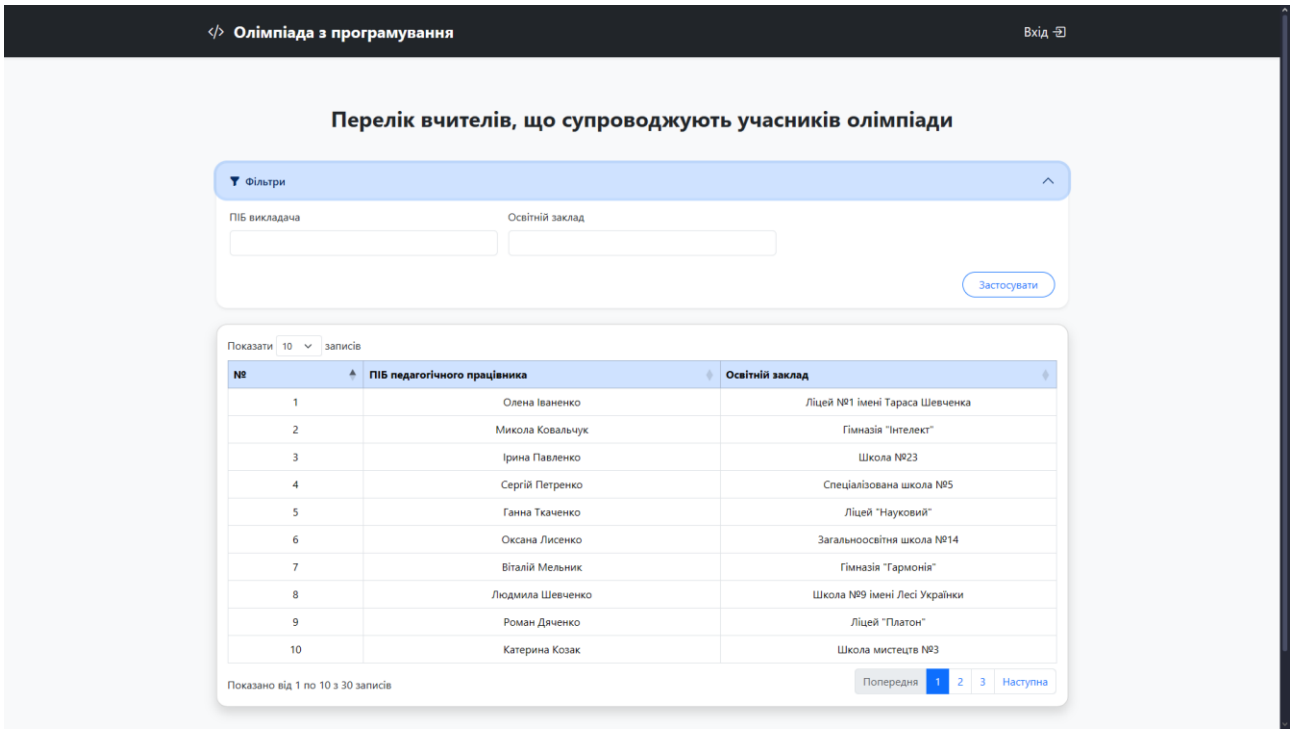

Рис. В.6 – Інтерфейс розділу «Вчителі»

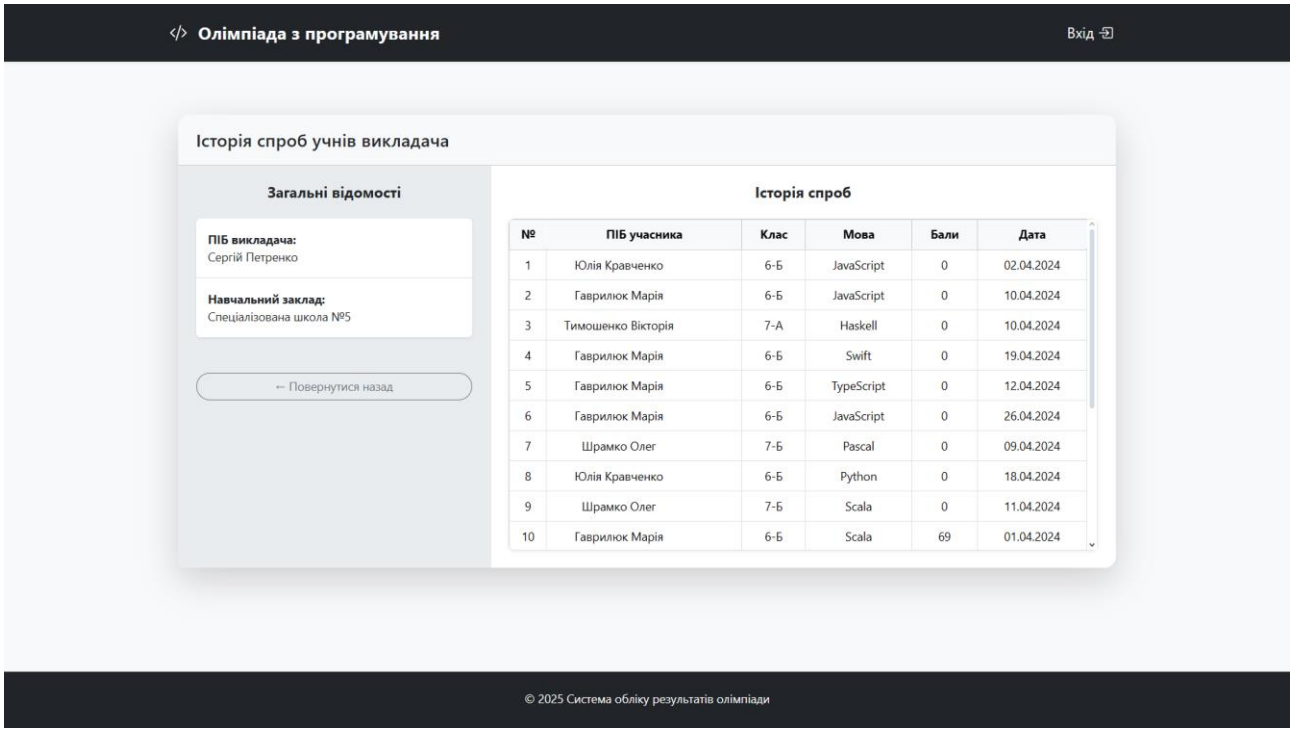

Рис. В.7 – Перегляд історії спроб учнів, підготованих конкретним викладачем

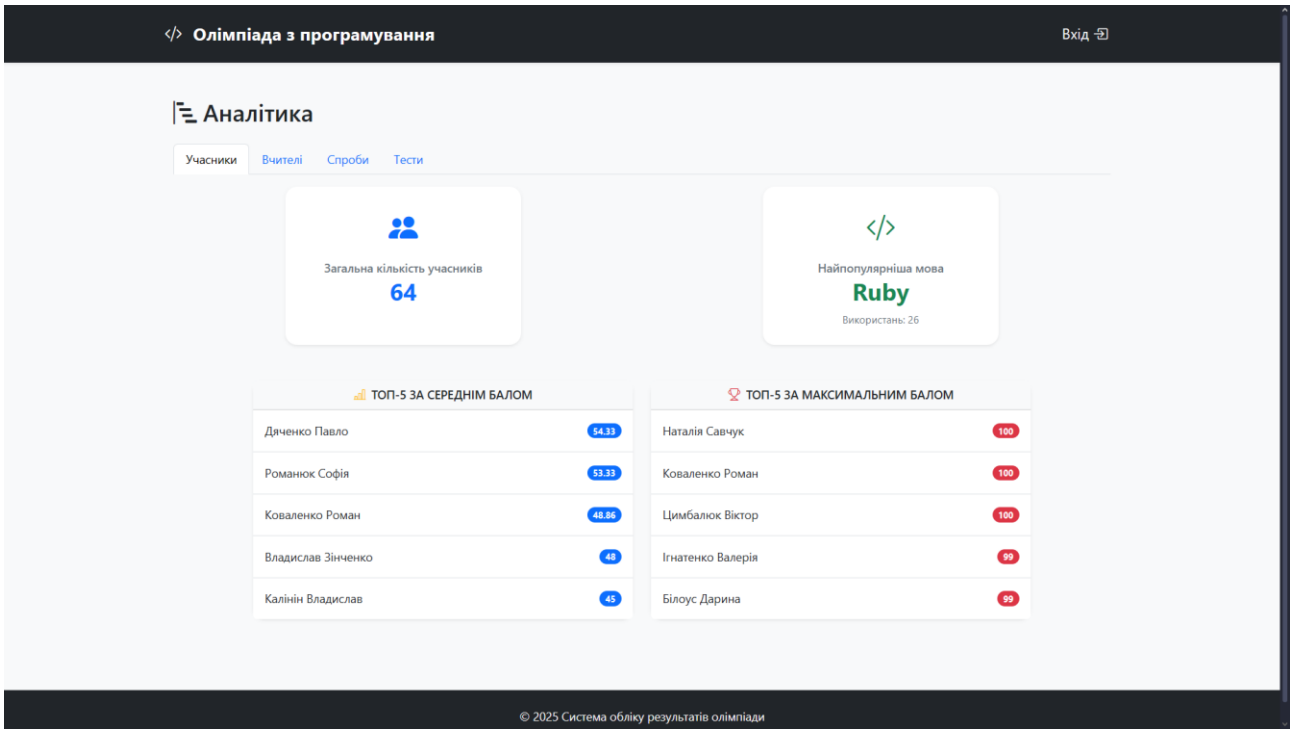

Рис. В.8 – Інтерфейс розділу «Аналітика» на вкладці «Учасники»

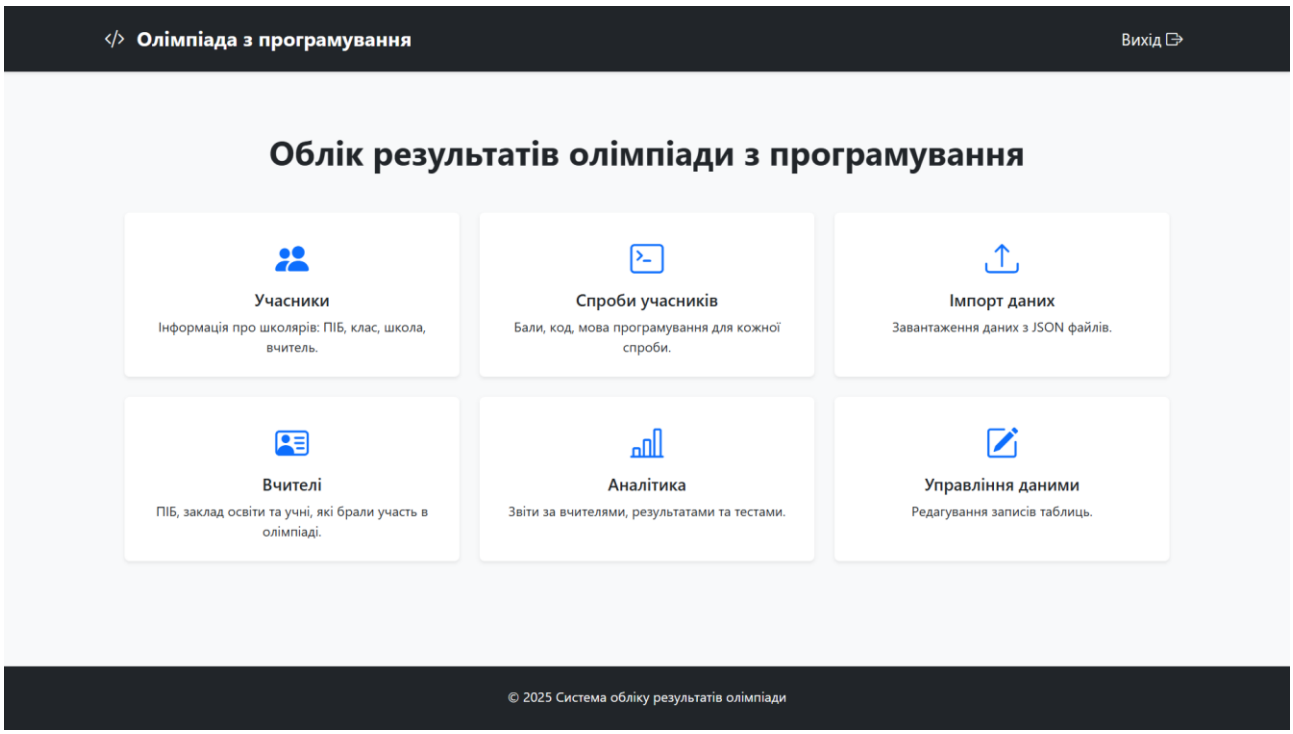

Рис. В.9 – Головна сторінка адміністратора з переліком доступних розділів системи

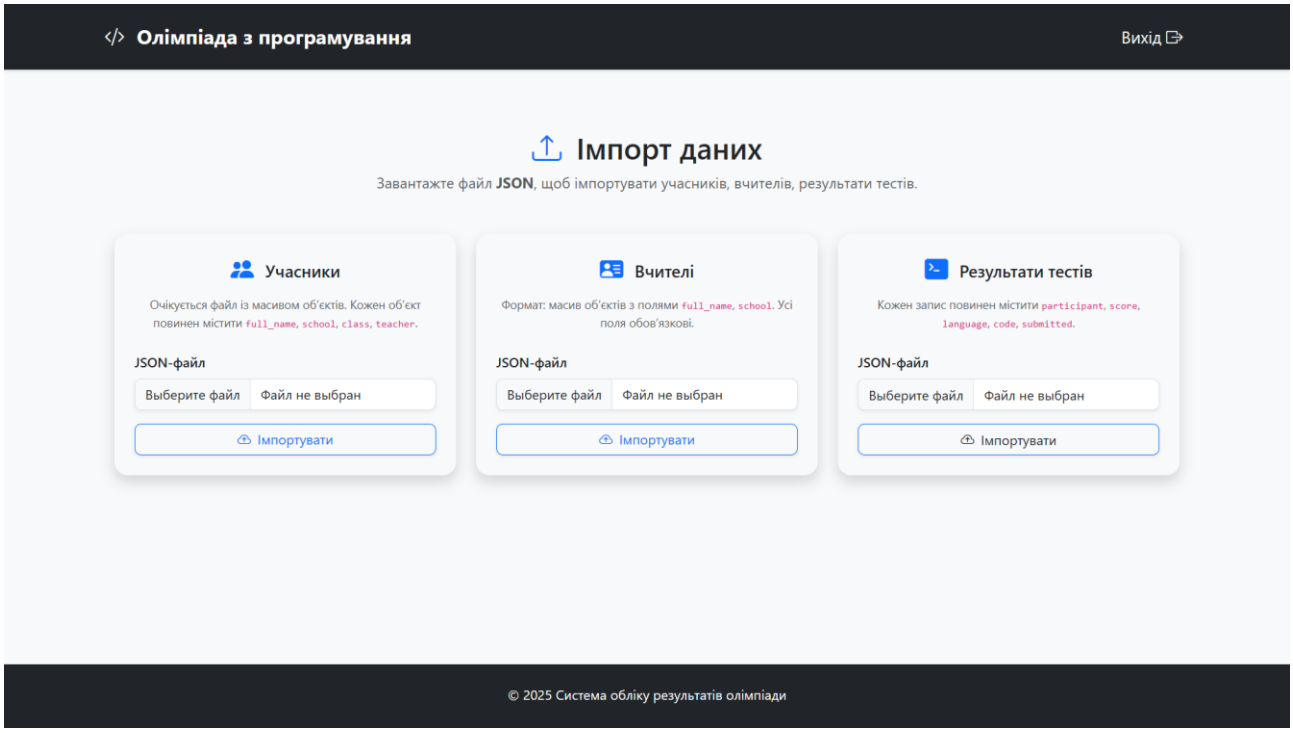

Рис. В.10 – Розділ «Імпорт даних» для завантаження JSON-файлів до бази даних

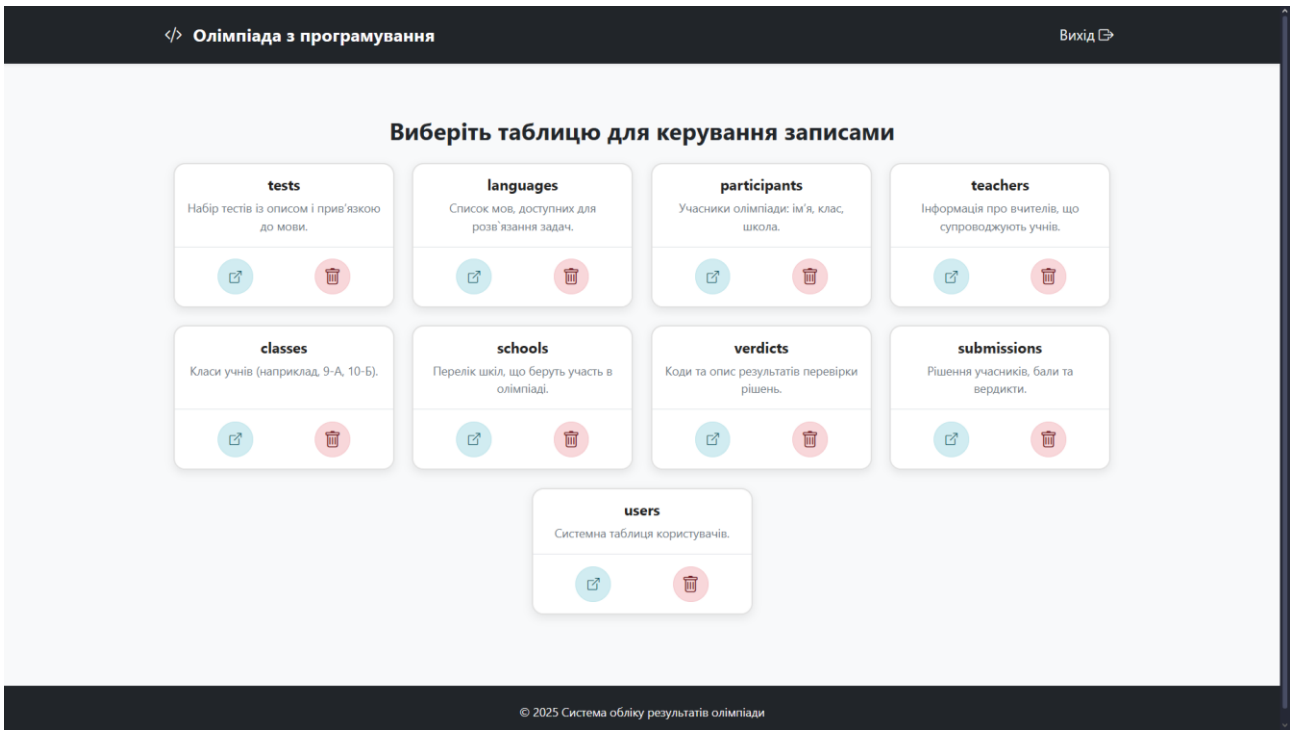

Рис. В.11 – Розділ «Управління даними» з доступом до редагування таблиць бази даних