Міністерство освіти і науки України

Український державний університет науки і технологій

Факультет «Комп’ютерні технології і системи»

Кафедра «Комп’ютерні інформаційні технології»

Пояснювальна записка

до кваліфікаційної роботи

ОС Бакалавр

на тему «Розробка веб-додатку для профорієнтації абітурієнтів з урахуванням специфіки спеціальностей університету»

за освітньою програмою «12 Інженерія програмного забезпечення»

зі спеціальності «121 Інженерія програмного забезпечення»

Виконав: студент групи ПЗ22130: \_\_\_\_\_\_\_ /Артем БЄЛІКОВ/

Керівник: \_\_\_\_\_\_\_ /Олександр ЖЕВАГО/

Нормоконтролер: \_\_\_\_\_\_\_ /Світлана ВОЛКОВА /

Засвідчую, що у цій роботі немає

запозичень з праць інших авторів

без відповідних посилань.

Студент

Дніпро – 2025

Ministry of Education and Science of Ukraine

Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»

Department «Computer information technology»

Explanatory Note

to Bachelor's Thesis

on the topic: «Development of a web application for career guidance of applicants, taking into account the specifics of university specialties

according to educational curriculum «12 Software engineering»

in the Speciality: «121 Software engineering»

Done by the student of the group PZ22130: \_\_\_\_\_\_\_ /Artem BIELIKOV/

Scientific Supervisor: \_\_\_\_\_\_\_ /Oleksandr ZHEVAHO/

Normative controller: \_\_\_\_\_\_\_ /Svitlana VOLKOVA/

Dnipro – 2025

Міністерство освіти і науки України

Український державний університет науки і технологій

Факультет: «Комп'ютерні технології і системи»

Кафедра: «Комп'ютерні інформаційні технології»

Рівень вищої освіти: бакалавр

Освітня програма: «12 Інженерія програмного забезпечення»

Спеціальність: «121 Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри КІТ

/Вадим ГОРЯЧКІН/

Дата

**ЗАВДАННЯ**

на кваліфікаційну роботу Бакалавр

студента Бєлікова Артема Євгеновича

1. Тема роботи: «Розробка веб-додатку для профорієнтації абітурієнтів з урахуванням специфіки спеціальностей університету»

Керівник роботи: Жеваго Олександр Олександрович, доцент

затверджені наказом від "03" 03 2025р. №328ст

1. Строк подання студентом роботи: 17.06.2025р.
2. Вихідні дані до роботи: програмний продукт.
3. Зміст пояснювальної записки:
   1. Вступ;
   2. Збір та аналіз вимог;
   3. Проєктування;
   4. Розробка програми;
   5. Тестування та налагодження;
   6. Висновки;
   7. Бібліографічний список.
4. Перелік демонстраційного матеріалу: презентація, відео роботи програми.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
| 1 | Вступ | 03.03.2025 – 05.03.2025 |  |
| 2 | Збір та аналіз вимог | 06.03.2025 – 11.03.2025 |  |
| 3 | Проєктування (зовнішнє проєктування) | 12.03.2025 – 22.03.2025 |  |
| 4 | Проєктування (внутрішнє проєктування) | 23.03.2025 – 02.05.2025 | 30% |
| 5 | Розробка програми | 03.05.2025 – 23.05.2025 | 60% |
| 6 | Тестування та налагодження | 24.05.2025 – 31.05.2025 |  |
| 7 | Аналіз та висновки | 01.06.2025 – 13.06.2025 | 100% |
| 8 | Подання кваліфікаційної роботи до кафедри | 17.06.2025 |  |
| 9 | Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії | 27.06.2025 |  |

Студент Артем БЄЛІКОВ

Керівник роботи: Олександр ЖЕВАГО

**РЕФЕРАТ**

Пояснювальна записка до кваліфікаційної роботи бакалавра:

100 с., 36 рис., 12 табл., 4 додатки, 11 джерел

Об’єкт розробки — веб-додаток для проведення профорієнтаційного тестування абітурієнтів з урахуванням особливостей навчальних програм Українського державного університету науки і технологій.

Мета роботи — розробити веб-додаток, який дозволяє абітурієнтам пройти профорієнтаційне тестування, і на основі аналізу результатів отримати персоналізовані рекомендації щодо вибору спеціальності.

Результати роботи — створено веб-додаток, який реалізує механізм тестування користувачів з подальшим аналізом їх відповідей. Забезпечено автоматичну генерацію рекомендацій спеціальностей на основі еталонних даних студентів. Реалізовано функціонал керування користувачами з різними ролями (адміністратор, експерт, викладач), створення спеціальностей, питань, а також аналізу подібності відповідей для видачі найбільш релевантних результатів.

Ключові слова: ПРОФОРІЄНТАЦІЯ, СПЕЦІАЛЬНІСТЬ, АБІТУРІЄНТ, ТЕСТУВАННЯ, РЕКОМЕНДАЦІЇ, ВЕБ-ДОДАТОК.

**ЗМІСТ**

[**ВСТУП**](#_Toc168324606) 7

[**1 ЗБІР ТА АНАЛІЗ ВИМОГ**](#_Toc168324607) 8

[1.1 Опис аналогів](#_Toc168324608) 8

[1.2 Постановка задачі](#_Toc168324609) 11

[**2 ПРОЄКТУВАННЯ**](#_Toc168324610) 13

[2.1 Зовнішнє проєктування](#_Toc168324611) 13

[2.1.1 Функціональне та експлуатаційне призначення](#_Toc168324612) 13

[2.1.2 Функціональні вимоги](#_Toc168324613) 13

[2.1.3 Вхідні дані](#_Toc168324614) 15

[2.1.4 Вихідні дані](#_Toc168324615) 15

[2.2Внутрішнє проєктування](#_Toc168324616) 15

[2.2.1 Мова програмування](#_Toc168324617) 15

[2.2.2 Проєктування архітектури програми](#_Toc168324618) 16

[2.2.3 Проєктування інтерфейсу користувача](#_Toc168324619) 20

[2.2.4 Проєктування бази даних](#_Toc168324620) 24

[**3** **РОЗРОБКА ПРОГРАМИ**](#_Toc168324621) 28

[**4** **ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ**](#_Toc168324622) 34

[4.1 Специфікації функцій для тестування](#_Toc168324623) 34

[4.2 Тестування методами білої скриньки](#_Toc168324624) 34

[4.3 Тестування методами чорної скриньки](#_Toc168324625) 37

[4.4 Налагодження програми](#_Toc168324626) 39

[**5** **ВИСНОВКИ**](#_Toc168324627) 40

[**БІБЛІОГРАФІЧНИЙ СПИСОК**](#_Toc168324628) 41

[**ДОДАТОК А**](#_Toc168324629) 42

[**ДОДАТОК Б**](#_Toc168324630) 55

[**ДОДАТОК В**](#_Toc168324631) 58

[**ДОДАТОК Г**](#_Toc168324632) 86

# **ВСТУП**

**Актуальність роботи.** У сучасних умовах зростаючої конкуренції між навчальними закладами та широкого вибору освітніх напрямів абітурієнти потребують інструментів, що допоможуть зробити усвідомлений вибір спеціальності. Стандартні профорієнтаційні тести часто не враховують специфіку конкретного університету, особливості навчальних програм та вимог до навичок студентів. Тому актуальним є створення веб-додатку, який дозволяє проходити тестування з урахуванням внутрішньої структури освітніх програм та досвіду студентів, які вже навчаються. Такий підхід забезпечує більш персоналізовані й точні рекомендації для абітурієнтів.

**Мета роботи.** Розробити веб-додаток для профорієнтаційного тестування абітурієнтів, який дозволяє на основі відповідей на питання, пов’язані з інтересами та навичками, визначати найбільш релевантні спеціальності. Система здійснює порівняння відповідей з еталонними даними студентів, що вже навчаються за відповідними програмами, і надає персоналізовані рекомендації щодо вибору спеціальності відповідно до навчальних програм Українського державного університету науки і технологій.

**Експлуатаційне призначення.** Розроблений веб-додаток призначений для використання абітурієнтами, які планують вступ до Українського державного університету науки і технологій, а також адміністраторами, які формують і супроводжують тестування. Абітурієнти можуть пройти тест, отримати рекомендації та ознайомитися з освітніми напрямами. Студенти з роллю експерта сприяють вдосконаленню системи, поповнюючи базу еталонних даних. Адміністратори мають змогу керувати ролями користувачів, редагувати тест, спеціальності, та контролювати повноту і якість тестового контенту.

# **1 ЗБІР ТА АНАЛІЗ ВИМОГ**

## **Опис аналогів**

**Аналог 1. “Дія.Освіта - Профорієнтаційний тест”.**

Проєкт створений Мінцифрою. Містить тест, який включає 94 запитання. Тест вказує на вашу схильність до певного роду дільності [1].

**Переваги:**

Інтерфейс платформи виконаний у світлих нейтральних тонах, з використанням палітри чорного та білого кольорів. Така кольорова гама знижує навантаження на зір і створює відчуття спокійної взаємодії.

Кількість елементів інтерфейсу на сторінках мінімальна: є лише найнеобхідніші компоненти, такі як кнопки відповіді, індикатор прогресу та заголовок поточного запитання. Відсутність зайвих графічних блоків забезпечує низький рівень інформаційного шуму, що дозволяє користувачеві зосередитися безпосередньо на проходженні тесту.

Сайт не є комерційним, а є державною програмою, тому відсутні різного виду підписки або реклама тощо.

При проходженні тесту у результаті користувач отримує “Освітні серіали”, які містять декілька відео про рекомендовану професію. У цих серіалах розповідається про професію.

Простота користування. Користувачу інтуїтивно зрозуміло як користуватися сайтом.

**Недоліки:**

Тест повністю пов’язаний з питаннями про тип особистості людини та використовує мінімум запитань, які конкретно пов’язані з тою чи іншою професією (спеціальністю).

У рекомендованих вакансіях відображаються тільки ті вакансії про які відзняті “Освітні серіали” за рідким виключенням, що користувачу не дає обширний список. Так, наприклад, у результатах може відображатися “Менеджер з клінінгу”, “Майстер з поклейки шпалер”. Очевидно, що ці вакансії не є актуальними та жаданими для багатьох людей.

**Аналог 2. Всеукраїнський проєкт з профорієнтації та побудови кар’єри.**

Цифровий сервіс, який пропонує проходження двоетапного профорієнтаційного тесту. Перший етап спрямований на виявлення особистісного типу користувача, а другий — на визначення типу інтелекту. Питання подаються у формі простих тверджень, на які потрібно відповідати, вказуючи ступінь згоди [2].

На основі відповідей система генерує перелік рекомендованих професій, які найкраще відповідають виявленим характеристикам. Результати тестування автоматично зберігаються у профілі користувача на платформі, що дозволяє до них повернутись у будь-який час [2].

**Переваги:**

Після проходження тесту є обширний список рекомендованих професій.

Відсутність реклами та платних підписок.

Відображається тип інтелекту користувача.

**Недоліки:**

Незручно проходити тест, тому що сама концепція перетягування кнопок з однієї частини в іншу займає багато часу. Також їх потрібно перетягувати у порядку зростання пріоритетності, що уповільнює проходження тесту.

Дрібний шрифт, мало графічних об’єктів (графіків, картинок тощо).

**Аналог 3. Career Explorer.**

CareerExplorer від Університету Арізони – надає безкоштовний тест, який допомагає студентам знайти кар’єрні напрямки відповідно до їхніх навичок, інтересів та пріоритетів. Платформа також має інструменти для детального аналізу кар’єрних можливостей, що допомагає студентам обрати найбільш відповідні спеціальності та напрямки розвитку [3]. ​

**Переваги:**

Питання, які стосуються саме спеціальності та професії.

Зручне оформлення тесту: шкала від 1 до 5, зрозумілість питань.

Система не потребує якоїсь додаткової реєстрації тільки потрібно вказати email.

Достатня кількість рекомендованих професій та реалізовані окремі сторінки по професіям, де відображається інформація про них.

**Недоліки:**

Підтримка лише англійської мови.

Повторення подібних питань.

**Аналог 4. College Major and Career Quiz.**

Система університету Arizona State University (ASU) me3® пропонує інтерактивний тест, який допомагає студентам знайти найбільш відповідні спеціальності та кар'єрні шляхи [4].

**Переваги:**

Інтерактивний тест не є типовим, оскільки він базується на виборі картинок, які пов’язані з кар’єрою.

По завершенню тесту вказується ще коледжі та школи при університеті з рекомендованою для вас спеціальністю. Вказана адреса та сторінка навчального закладу.

Достатній список рекомендованих професій.

**Недоліки:**

Багато контактних даних потрібно вказати.

Навігація по сайту.

**Аналог 5. Careerizma.**

Careerizma – це онлайн-платформа, орієнтована на надання безкоштовного кар'єрного тестування для студентів і професіоналів, які шукають поради щодо вибору кар'єрного шляху [5].

**Переваги:**

Простота та зручність – тест легкий у проходженні та складається з простих питань.

Широкий спектр можливостей – тест охоплює багато різних професій і напрямків.

Наукова основа – тест спирається на методику Holland Code, яка є добре вивченою та широко застосовується в кар'єрному консультуванні, надаючи певну довіру до точності результатів.

**Недоліки:**

Залежність від самооцінки – результати тесту залежать від того, наскільки об'єктивно учасник оцінює свої інтереси та здібності, що може призвести до похибок у випадках, коли учасники мають обмежене уявлення про свої справжні інтереси​.

Обмеження для нетрадиційних професій – тест більше орієнтований на класичні кар'єрні напрями, тому може бути недостатньо адаптованим для нових спеціальностей, які стали популярними в останні роки, наприклад, у сфері IT чи технологій.

Розробка даного проєкту відрізняється від існуючих аналогів.

Більшість профорієнтаційних систем є загальними й не враховують специфіку навчальних програм окремих закладів. У нашому випадку тестування орієнтоване саме на спеціальності Українського державного університету науки і технологій, що забезпечує точніші та корисніші рекомендації.

Замість абстрактних оцінок або типових профілів, система аналізує реальні відповіді студентів, які вже навчаються за відповідними спеціальностями. Це дозволяє формувати персоналізовані рекомендації на основі фактичних даних, а не лише за шаблонною логікою.

Динамічне оновлення бази даних. Еталонні відповіді оновлюються зі зростанням кількості відповідів студентів.

Система має окремі інтерфейси та права доступу для адміністратора, викладача, студента та анонімного користувача.

## **Постановка задачі**

Розробити веб-додаток для профорієнтації абітурієнтів з урахуванням специфіки спеціальностей університету.

Ключові функціональні вимоги:

* забезпечити використання додатку для трьох типів користувачів (абітурієнт, експерт, адміністратор);
* надати користувачам веб-додатку можливість переглядати головну сторінку, перелік спеціальностей та інформацію про університет;
* реалізувати тестування абітурієнтів із обчисленням рекомендацій на основі косинусної подібності їхніх відповідей з еталонними профілями;
* дозволити експертам проходити аналогічний тест для формування «еталонних» векторів спеціальностей;
* через адмін-панель Strapi забезпечити керування контентом: створення, редагування та видалення питань, спеціальностей, та експертних відповідей;
* забезпечити розмежування прав доступу та захист даних відповідно до ролей.

# **2 ПРОЄКТУВАННЯ**

## **2.1 Зовнішнє проєктування**

### **2.1.1 Функціональне та експлуатаційне призначення**

Фунціональне призначення

Розроблений веб-застосунок призначений для проходження абітурієнтами профорієнтаційного тестування з метою визначення найбільш релевантних напрямів підготовки (спеціальностей) на основі їхніх відповідей. Основна функціональність полягає у наданні користувачеві списку рекомендованих спеціальностей, що базується на обчисленні косинусної подібності між відповідями користувача та еталонними профілями спеціальностей, сформованими експертами.

Функції системи включають:

* формування, редагування та впорядкування тестових питань адміністратором;
* створення та керування спеціальностями (категоріями);
* проходження тесту абітурієнтами та експертами;
* збереження та обробка відповідей;
* формування та оновлення еталонних векторів на основі експертних відповідей.

Експлуатаційне призначення

Система призначена для використання:

* у закладах вищої освіти під час профорієнтаційної роботи;
* кар'єрними консультантами та працівниками приймальних комісій;
* учнями та абітурієнтами у самостійному режимі онлайн через веб-браузер.

### **2.1.2 Функціональні вимоги**

Функціональні вимоги повинні забезпечити можливість:

* проходження тестування абітурієнтами без авторизації;
* авторизованим експертам проходити тест для формування еталонних відповідей;
* збереження відповідей користувачів та експертів;
* автоматичного обчислення результатів на основі косинусної подібності;
* виведення результатів у вигляді діаграми з рекомендаціями спеціальностей;
* створення, редагування та впорядкування питань адміністратором у CMS;
* створення, публікації та верифікації категорій спеціальностей;
* оновлення еталонних векторів категорій на основі експертних оцінок;
* автоматичної генерації зв’язків між питаннями та категоріями;
* забезпечення зручного керування вмістом через панель адміністратора.

Специфікація функціональних вимог представлена у вигляді діаграми прецедентів на рисунку 2.1.



Рисунок 2.1 – Діаграма прецедентів

### **2.1.3 Вхідні дані**

### Для системи профорієнтаційного тестування вхідні дані включають наступне:

### перелік запитань для тестування;

### інформація про спеціальності (категорії);

### відповіді абітурієнтів на тестові запитання;

### відповіді експертів, що формують еталонні вектори;

### параметри доступу користувачів (адміністратор, експерт);

### налаштування тесту, порядок запитань.

### **2.1.4 Вихідні дані**

## Вихідними даними є:

## рекомендації щодо вибору спеціальності для абітурієнта;

## графічне представлення результатів (діаграма подібності до спеціальностей);

## збережені результати тестування;

## оновлені еталонні вектори спеціальностей на основі експертних відповідей;

## інтерфейс керування контентом для адміністратора (питання, категорії, відповіді).

## **2.2 Внутрішнє проєктування**

### **2.2.1 Мова програмування**

Для розробки веб-додатку було обрано мову програмування JavaScript та TypeScript, а також використано сучасні технології React для клієнтської частини та Strapi (Node.js) для серверної частини (CMS).

JavaScript є однією з найпоширеніших мов програмування у світі, зокрема у сфері веб-розробки. Вона підтримується всіма сучасними браузерами та має широку екосистему бібліотек і фреймворків. Для забезпечення більшої надійності та типізації в коді застосовується TypeScript, що є надбудовою над JavaScript [6].

Фронтенд-застосунок створено за допомогою React — популярної бібліотеки для побудови користувацьких інтерфейсів. React дозволяє будувати компонентну структуру додатку, забезпечує високу продуктивність завдяки віртуальному DOM та має активну спільноту розробників [7].

У якості backend-рішення використано Strapi — headless CMS, яка побудована на базі Node.js і дозволяє швидко створювати REST/GraphQL API для взаємодії з клієнтською частиною. Strapi має інтуїтивно зрозумілий інтерфейс керування контентом, підтримує налаштування ролей і прав доступу, а також дозволяє кастомізувати структуру бази даних без прямого редагування SQL-коду [8].

У якості СКБД використано SQLite, що є простою у налаштуванні, легкою та зручною для невеликих та середніх проєктів. Її інтеграція з Strapi забезпечує швидке збереження та доступ до тестових даних [9].

Такий стек технологій забезпечує кросплатформенність, швидкість розробки, гнучкість та масштабованість рішення.

### **2.2.2 Проєктування архітектури програми**

Розроблений веб-додаток має багаторівневу архітектуру, яка умовно поділяється на чотири основні рівні:

Data Layer (Рівень даних) — реалізований за допомогою CMS Strapi та бази даних SQLite. Відповідає за зберігання, обробку та доступ до даних (питання, категорії, відповіді, користувачі).

Business Logic Layer (Рівень бізнес-логіки) — реалізується через кастомні сервіси Strapi та обробку на фронтенді. Сюди входить логіка обчислення косинусної подібності, оновлення векторів категорій, логіка побудови результатів.

Presentation Layer (Рівень представлення) — побудований на React з використанням бібліотеки Material UI. Забезпечує інтерфейс для користувачів (абітурієнтів, експертів, адміністраторів).

Authentication Layer (Ідентифікація/авторизація) — реалізований через Strapi Roles & Permissions plugin. Дає змогу керувати доступом для різних ролей: адміністратор, експерт, гостьовий користувач.

Загальна архітектура системи представлена на рисунку 2.2.

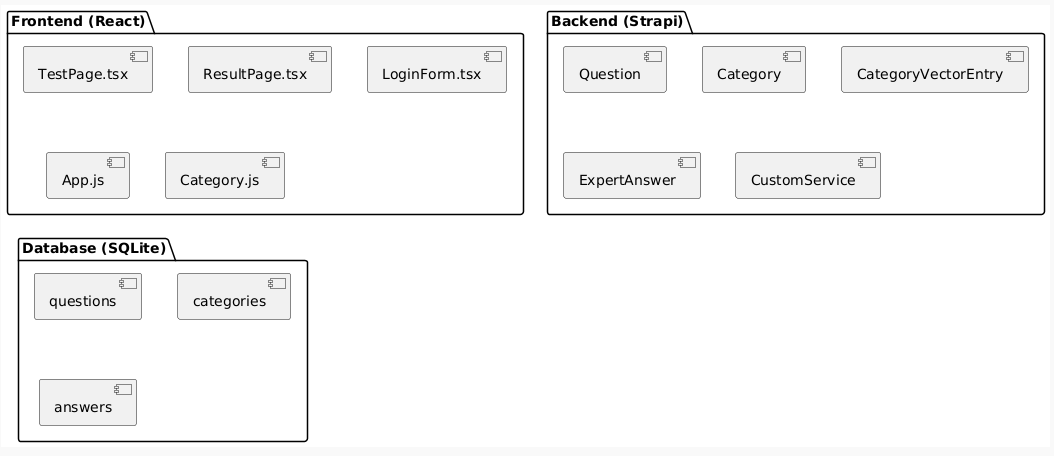


Рисунок 2.2 – Компоненти додатку

Програма використовує підхід модульності та розділення відповідальностей, що дозволяє легко масштабувати та супроводжувати код. Використання headless CMS дає змогу адміністратору змінювати структуру тестів без втручання у код, а фронтенд — швидко рендерити динамічний вміст.

Клієнтська частина реалізована за патерном MVC [10], де:

* модель — структура даних, яку отримано з API (питання, відповіді, категорії);
* представлення (View) — компоненти React, які відображають дані (інтерфейс);
* контролер — логіка компонентів, яка реагує на дії користувача та викликає API.

Реалізація патерну MVC показана на рисунку 2.3.

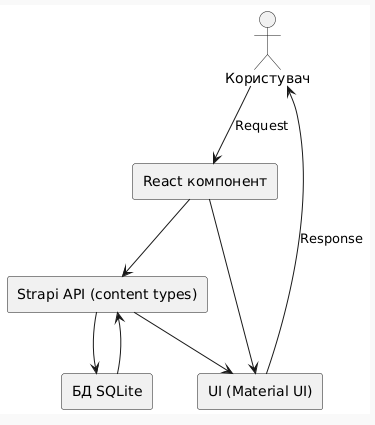


Рисунок 2.3 – Реалізація патерну MVC

На рисунку 2.4 представлено діаграму класів для рівня бізнес логіки.

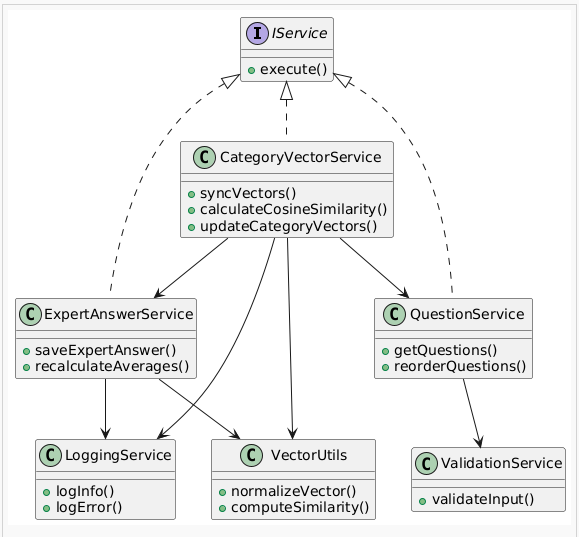


Рисунок 2.4 – Діаграма класів для BLL

На рисунку 2.5 представлено діаграму класів для рівня доступу до даних.

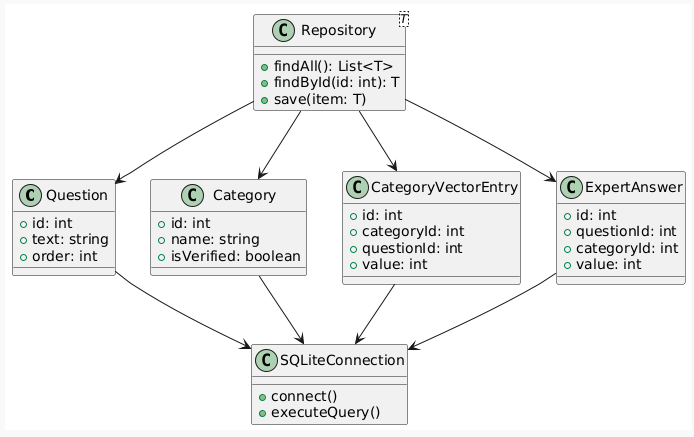


Рисунок 2.5 – Діаграма класів для DAL

На рисунках 2.6-2.7 представлено діаграми класів для рівня представлення даних.

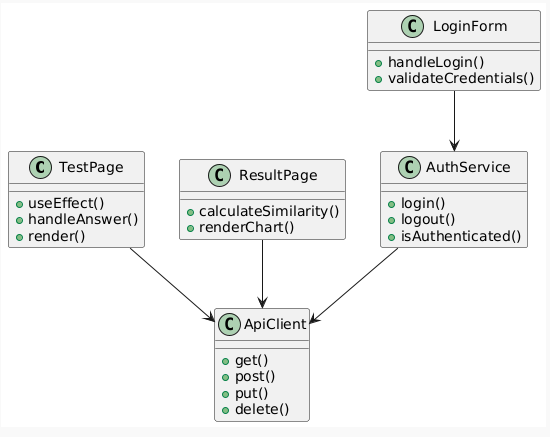


Рисунок 2.6 – Діаграма класів для PL, контролери

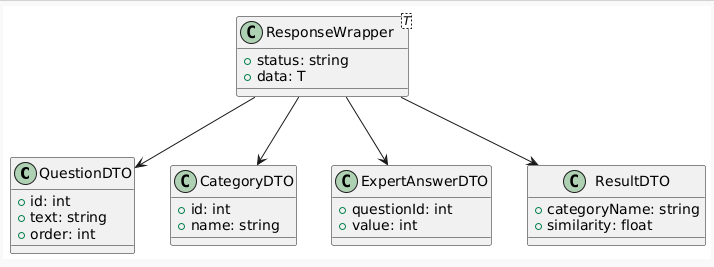


Рисунок 2.7 – Діаграма класів для PL, представлення

На рисунках 2.8 представлено діаграми класів для рівня Identity.

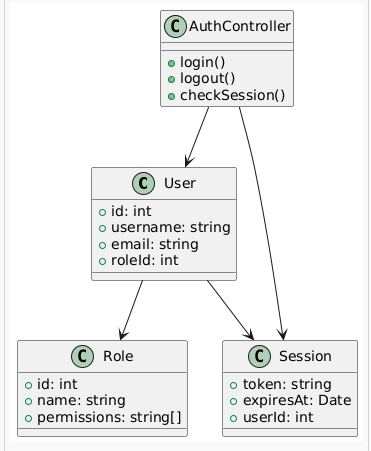


Рисунок 2.8 – Діаграма класів для Identity

### **2.2.3 Проєктування інтерфейсу користувача**

Прецедент: проходження тесту абітурієнтом.

Ініціатор – користувач (абітурієнт).

Користувач відкриває інтерфейс тестування, послідовно відповідає на запитання та надсилає відповіді. Форма відображає поточне запитання, збирає вибрані відповіді й передає їх до контролера. Контролер здійснює перевірку, формує масив відповідей і надсилає на обробку до бекенду. На стороні сервера дані зберігаються, а система виконує обчислення подібності до кожної категорії. Результат обчислень повертається до форми, яка виводить діаграму з рекомендаціями.

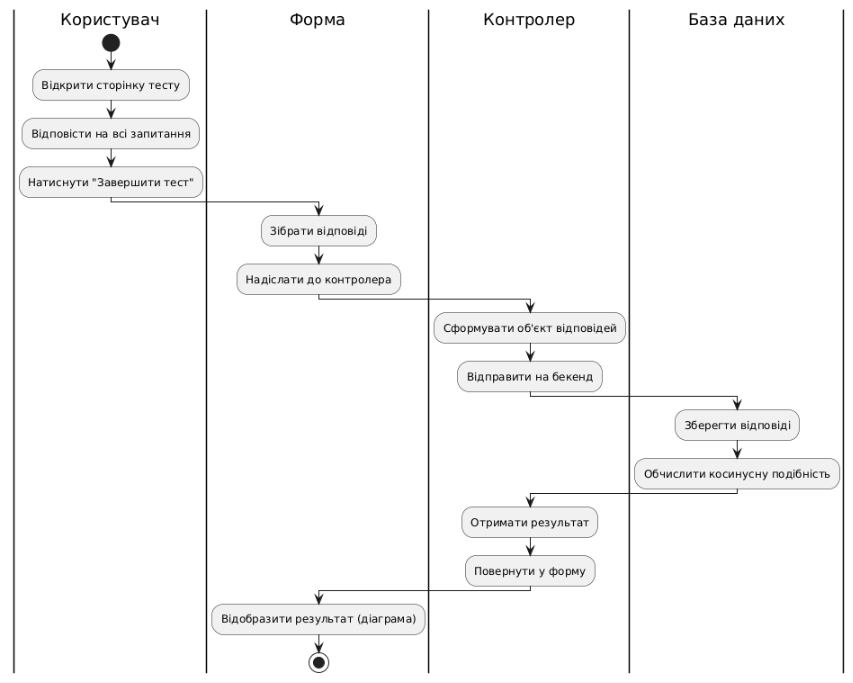


Рисунок 2.8 – Діаграма послідовності для прецеденту «Проходження тесту абітурієнтом»

Прецедент: Проходження тесту експертом (еталонні відповіді).

Ініціатор – експерт (студент відмінник).

Користувач авторизується у системі та переходить до розділу тестування. Після проходження тесту відповіді передаються до контролера, який перевіряє роль користувача та передає дані на збереження. У базі даних нові відповіді зберігаються із зазначенням категорії. Система оновлює вектор категорії через обчислення нового середнього значення. Після завершення експерт отримує підтвердження про оновлення.

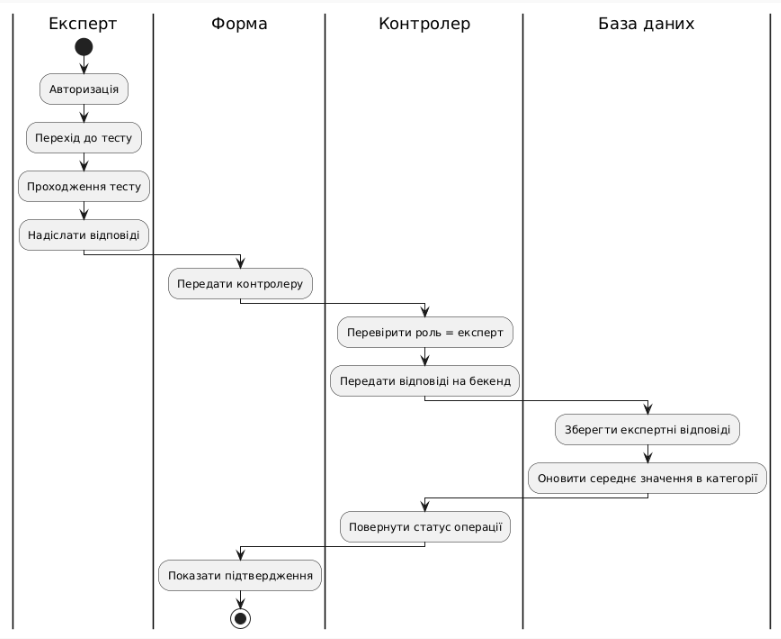


Рисунок 2.9 – Діаграма послідовності для прецеденту «Проходження тесту експертром»

Прецедент: додавання нового питання адміністратором.

Ініціатор – користувач (адміністратор).

Адміністратор відкриває інтерфейс керування контентом у CMS Strapi, переходить до розділу «Questions» та натискає кнопку «Create new entry». У форму вводиться текст питання, після чого запис зберігається. Strapi автоматично присвоює порядковий номер (order) залежно від кількості наявних питань, а також створює зв’язки з усіма наявними категоріями через модель CategoryVectorEntry. Після успішного збереження адміністратор отримує підтвердження.

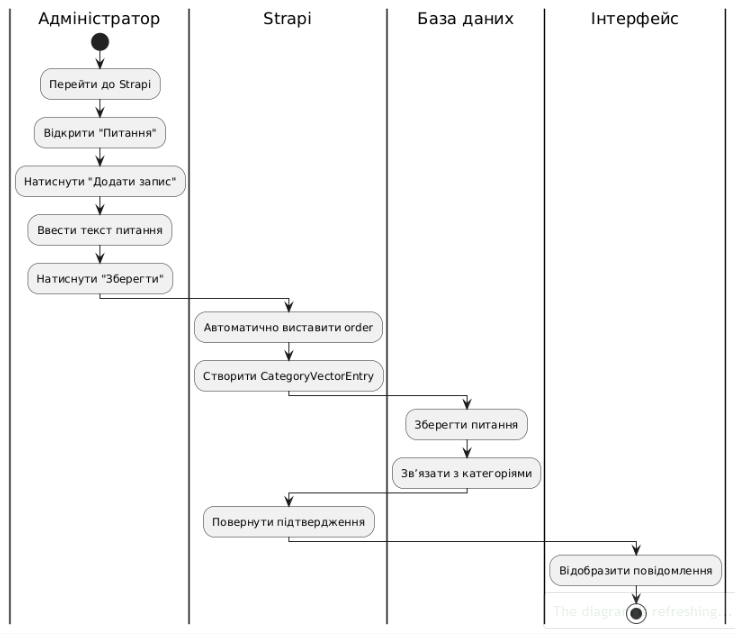


Рисунок 2.10 – Діаграма послідовності для прецеденту «Додавання нового питання адміністратором»

Для побудови інтерфейсу користувача застосовано React — сучасну JavaScript-бібліотеку для розробки односторінкових застосунків. React дозволяє динамічно рендерити компоненти на основі стану і підтримує декларативний підхід до побудови інтерфейсів.

Основна ідея полягає в тому, що клієнтська частина (frontend) отримує дані з серверної частини (Strapi CMS) через REST API або GraphQL, після чого ці дані відображаються у відповідних компонентах інтерфейсу.

Приклад React JSX-розмітки сторінки наведено на рисунку 2.11.

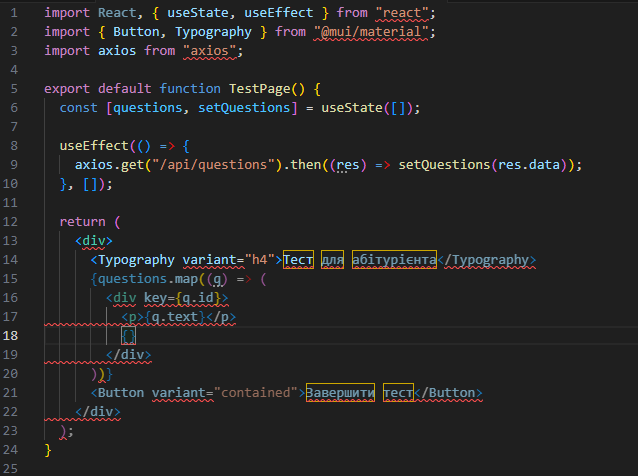


Рисунок 2.11­ – Розмітка сторінки

### **2.2.4 Проєктування бази даних**

У веб-додатку використовується система керування базами даних SQLite [9]. Це легка, вбудована СКБД, що не вимагає встановлення окремого серверу, підтримується Strapi CMS і є оптимальним варіантом для середніх за обсягом застосунків, які не потребують високого рівня навантаження.

Взаємодія з базою даних здійснюється автоматично через механізми Strapi ORM, які дозволяють:

* створювати структуру таблиць через JSON-схеми моделей (Content Type Builder);
* будувати зв’язки між сутностями без написання SQL-запитів;
* автоматично виконувати CRUD-операції;
* підтримувати зв’язки типу one-to-many, many-to-one та many-to-many;
* зберігати дані у зручному вигляді у файлі data.db.

Для реалізації системи було розроблено 4 основні таблиці, опис яких наведено у таблицях 2.1–2.4.

Таблиця 2.1 – Question

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Опис поля | Тип даних | Ключ |
| id | Унікальний ідентифікатор питання | integer | PK, AUTOINCR |
| text | Текст питання | varchar | - |
| order | Порядковий номер | integer | - |

Таблиця 2.2 – Categories

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Опис поля | Тип даних | Ключ |
| id | Унікальний ідентифікатор категорії | integer | PK, AUTOINCR |
| name | Назва спеціальності | varchar | - |
| isVerified | Ознака, чи категорія верифікована | boolean | - |
| vector | Вектор еталонних відповідей | text |  |

Таблиця 2.3 – Category\_vector\_entries

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Опис поля | Тип даних | Ключ |
| id | Унікальний ідентифікатор запису | integer | PK, AUTOINCR |
| category | Зв'язок із таблицею категорій | integer | FK |
| question | Зв'язок із таблицею питань | integer | FK |
| value | Еталонне значення (0–5) | Decimal (1,2) | - |

Таблиця 2.4 – Expert\_answers

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Опис поля | Тип даних | Ключ |
| id | Унікальний ідентифікатор запису | integer | PK, AUTOINCR |
| user\_id | Зв’язок з таблицею юзерів | integer | FK |
| category\_id | Зв’язок з таблицею категорій | integer | FK |
| answer | Масив відповідей користувача | varchar | - |
| is\_confirmed | Чи підтвердив адміністратор цю відповідь | boolean | - |

Таблиця 2.5 – User

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Опис поля | Тип даних | Ключ |
| id | Унікальний ідентифікатор запису | integer | PK, AUTOINCR |
| username | Унікальне ім’я | varchar | - |
| email | Електронна пошта | varchar | - |
| password\_hash | Хеш пароля | varchar | - |
| blocked | Заблоковано | boolean | - |

Схема бази даних наведено на рисунку 2.12.

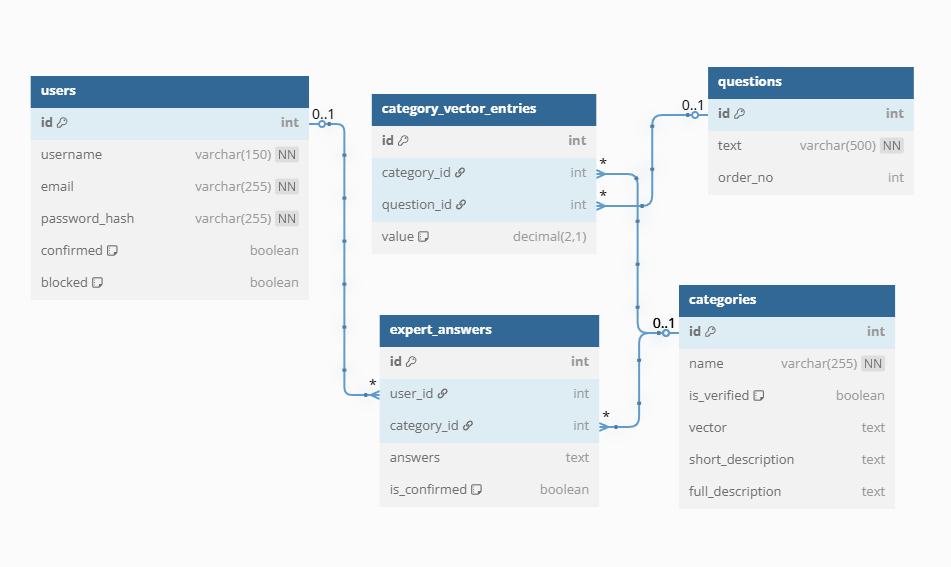


Рисунок 2.12 – Схема бази даних

# **РОЗРОБКА ПРОГРАМИ**

У цьому розділі детально розглянуто логіку функціонування веб-додатку для профорієнтації абітурієнтів. Представлено діаграми, які відображають поведінку основних типів користувачів (абітурієнтів та адміністратора), а також архітектуру системи та зміни станів у процесі проходження тестування.

На рисунку 3.1 наведено діаграму діяльності, що відображає основні дії користувача (абітурієнта) в системі профорієнтації: перегляд інформації, проходження тесту, отримання рекомендацій.



Рисунок 3.1 – Діаграма діяльності абітурієнта

Пояснення етапів:

Початок: Користувач відкриває сайт.

Перегляд категорій, спеціальностей: Може вільно знайомитися з інформацією.

Чи хоче пройти тест?: Визначається намір користувача.

Відповіді → Надсилання: Користувач відповідає на запитання й надсилає їх.

Рекомендації: Система обробляє результати й надає рекомендації щодо спеціальностей.

На рисунку 3.2. наведено діаграму діяльності, що демонструє основні дії адміністратора в CMS Strapi. Адміністратор має можливість створювати, редагувати питання, категорії, перевіряти відповіді експертів та оновлювати вектори.

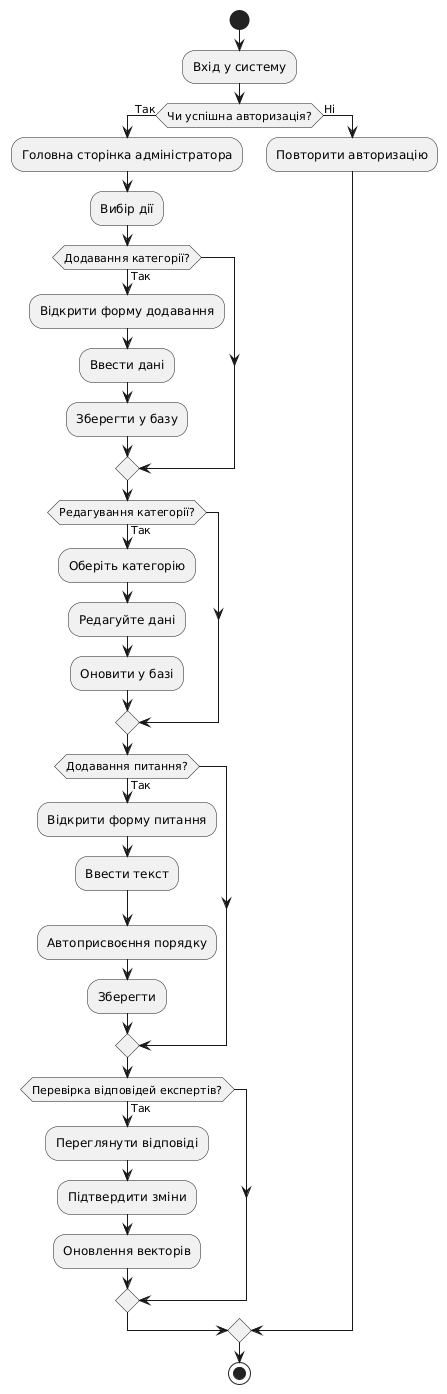


Рисунок 3.2 – Діаграма діяльності адміністратора

На рисунку 3.3 наведено діаграму розгортання, що показує архітектуру розгортання веб-додатку.

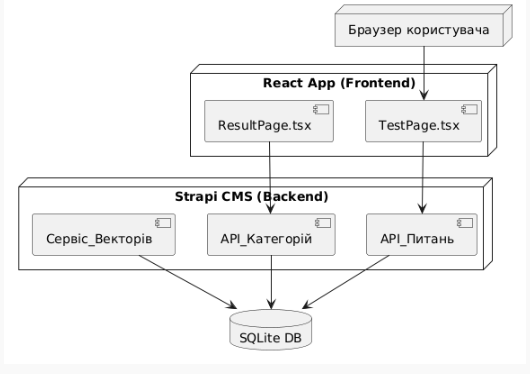
**Система складається з:**

* браузера користувача;
* клієнтської частини на React;
* серверної частини на Strapi CMS (Node.js);
* бази даних SQLite.

**Зв’язки:**

* користувач взаємодіє з клієнтською частиною;
* React надсилає запити до API Strapi;
* Strapi взаємодіє з SQLite.

Ця архітектура забезпечує масштабованість і можливість централізованого керування через CMS.



**Рисунок 3.3 – Діаграма розгортання системи**

На рисунку 3.4. наведено діаграму, що описує зміну станів користувача в процесі взаємодії з системою:

Становий перехід:

Viewing → Testing → ShowingResults → Viewing

Дана діаграма ілюструє цикл: Перегляд → Тестування → Результати → (Перегляд або Повтор тесту).

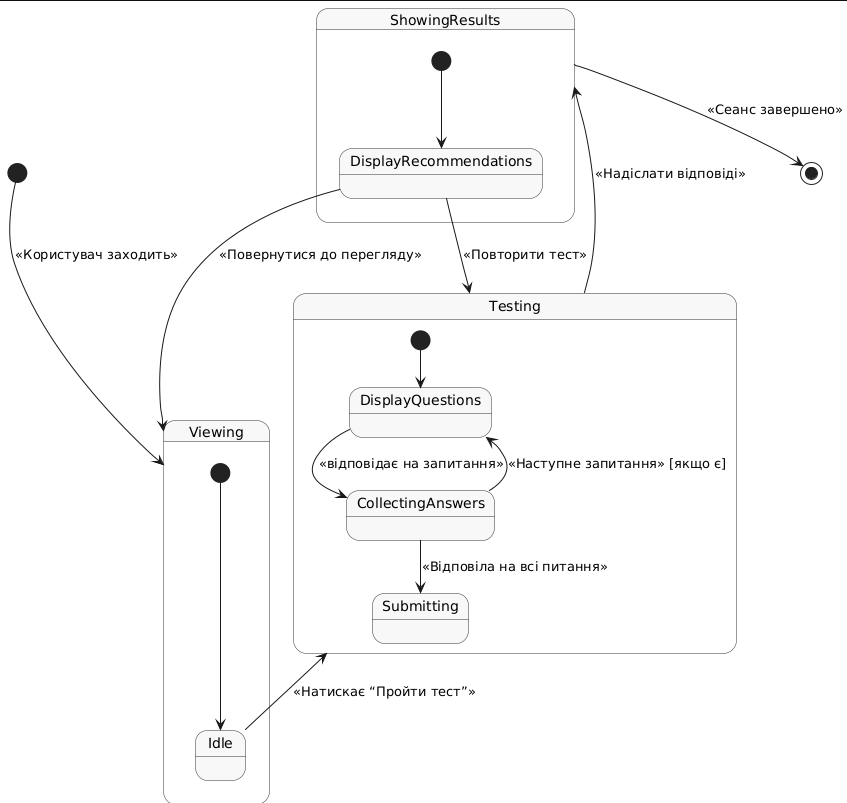


Рисунок 3.4 – Діаграма станів користувача в системі

На рисунку 3.5 наведено блок-схему, що описує алгоритм роботи функції handleSubmit() у компоненті TestPage веб-додатку.

Спочатку перевіряється наявність JWT-токена у localStorage. Якщо токен знайдено (гілка “так”), здійснюється послідовність дій: зчитується об’єкт користувача, формується тіло POST-запиту з його відповідями, відправляється запит на /api/expert-answers, після чого у разі успіху виводиться повідомлення про успішне збереження результату, інакше — логування помилки. Якщо ж токена немає (гілка “ні”), виконується GET-запит до /api/categories для отримання векторів категорій, перевіряється статус відповіді та парситься JSON. Далі ініціалізується порожній список scored, після чого у блоці “Compute CosineSimilarity” для кожної категорії поелементно рахується скалярний добуток і норми векторів, обчислюється коефіцієнт косинусної подібності (або нуль, якщо норми рівні нулю) і результат додається до масиву scored. По завершенні циклу масив сортується за спаданням значення подібності, з нього вибираються три найкращі позиції (top3), і викликається setResults(top3) для відображення рекомендацій користувачу. Таким чином, блок-схема чітко демонструє розгалужену логіку обробки — від збереження експертних відповідей до побудови персоніфікованих рекомендацій для анонімних відвідувачів.

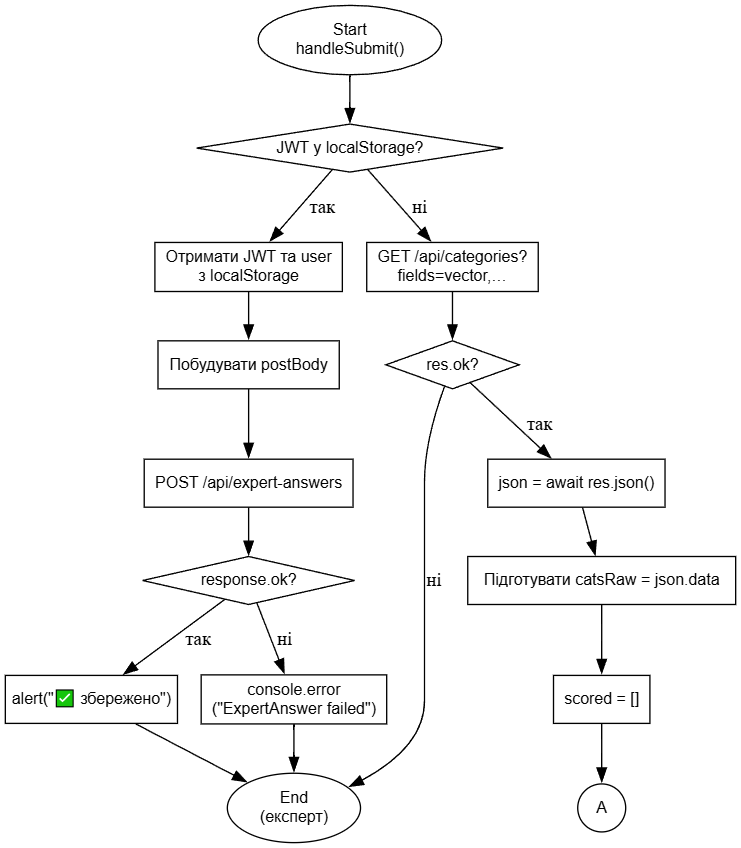


Рисунок 3.5 – Блок-схема алгоритму handleSubmit

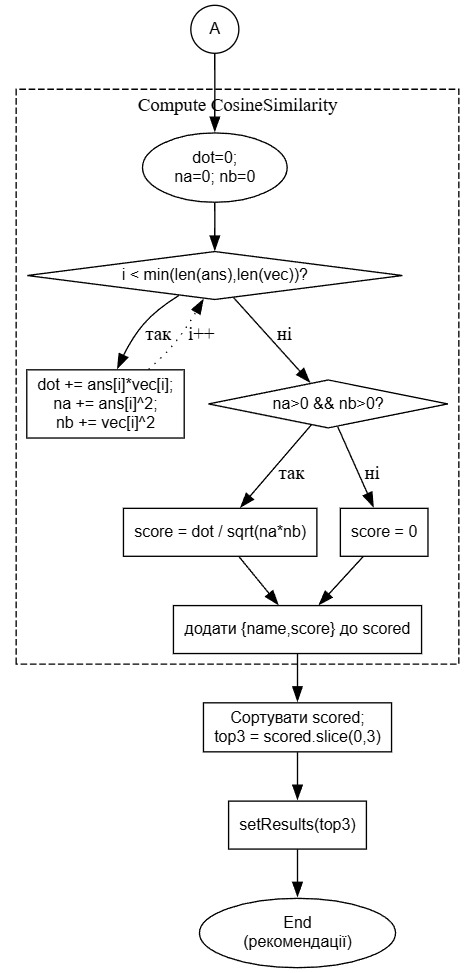


Рисунок 3.6 – Блок-схема алгоритму handleSubmit (доповнення)

# **ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ**

Тестування програмного забезпечення є вирішальним етапом для гарантування його стабільності та відповідності вимогам. Для перевірки застосовують два підходи: «чорна скринька» та «білий ящик». У першому випадку оцінка проводиться без ознайомлення з вихідним кодом — використовуються лише функції, доступні кінцевому користувачеві. У підході «білого ящика» навпаки аналізують внутрішню структуру програми й логіку її виконання для досягнення необхідного рівня тестового покриття [11].

У даній роботі для тестування «білого ящика» обрано метод покриття умов, що забезпечує активацію всіх гілок коду, а для «чорної скриньки» — метод еквівалентного розбиття, який дозволяє мінімізувати кількість необхідних тестових випадків.

## **Специфікації функцій для тестування**

Випадок 1. Тестування правильної відправки експертних відповідей. Функція handleSubmit.

Випадок 2. Тестування відображення рекомендацій. Функція handleSubmit.

Призначення: функція handleSubmit збирає масив відповідей, для авторизованих користувачів надсилає його на сервер через POST, а для неавторизованих — виконує GET запит до категорій, обчислює косинусну схожість і відображає топ-3 спеціальності.

Інтерфейс: async function handleSubmit(): Promise<void>.

Повертає:

Якщо є JWT → запит на POST /api/expert-answers

Інакше → запит на GET /api/categories та обчислення косинусної схожості → оновлення стану results/.

## **Тестування методами білої скриньки**

Для тестування випадку 1 методом покриття умов на рисунку 4.1 наведено текст функції.

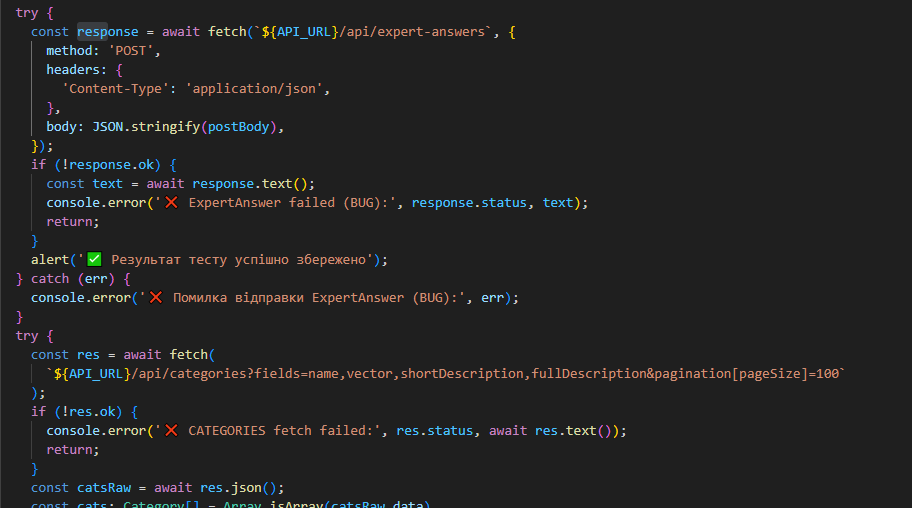


Рисунок 4.1 – Текст функції handleSubmit

Для позначення умов використано такі додаткові умовні позначення:

* U1 = виклик POST /api/expert-answers;
* U2 = виклик GET /api/categories (res.ok === true).

Тести для покриття умов:

Тест 1.

Вхід:

jws = null

GET /api/categories → HTTP 500

Тест 2.

Вхід:

jws = null

GET /api/categories → HTTP 200.

Таблиця 4.1 – результати тестування методом покриття умов для випадку 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | U1 | | U2 | |
| + | - | + | - |
| Тест 1 | \* |  |  | \* |
| Тест 2 | \* |  | \* |  |

Для тестування випадку 2 методом покриття умов на рисунку 4.2 наведено текст функції.

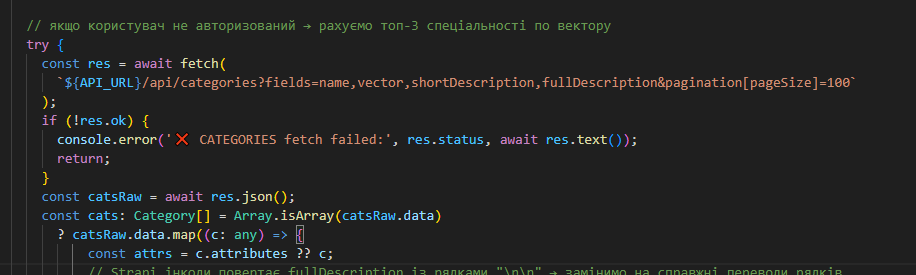


Рисунок 4.2 – Текст функції handleSubmit (відображення рекомендцій)

Для позначення умов використано такі додаткові умовні позначення:

* U1 = виклик fetch('/api/categories');
* U2 = res.ok === true.

Тести для покриття умов:

Тест 1.

Вхід:

jws = eyJhbGciOiJIU…

Очікуваний результат: не виконується GET /api/categories, відправляються відповівді експерта на сервер.

Тест 2.

Вхід:

jws = null

Очікуваний результат: GET /api/categories → HTTP 200.

Таблиця 4.2 – результати тестування методом покриття умов для випадку 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | U1 | | U2 | |
| + | - | + | - |
| Тест 1 |  | \* |  | \* |
| Тест 2 | \* |  | \* |  |

## **Тестування методами чорної скриньки**

Тестування методом еквівалентного розбиття для випадку 1 наведено в таблицях 4.3 – 4.4.

Таблиця 4.3 – класи еквівалентності для випадку 1

|  |  |  |
| --- | --- | --- |
| Вхідні умови | Правильні класи еквівалентності | Неправильні класи еквівалентності |
| Авторизація (JWT) | A1: JWT присутній | A2: JWT відсутній |
| CMS доступність | C1: GET /api/categories → HTTP 200 | C2: GET /api/categories → HTTP = 500 |

Таблиця 4.4 – тести за методом еквівалентних розбиттів для випадку 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Клас | Тест | Авторизація | CMS | Очікувана поведінка | Фактична поведінка (BUG) |
| A2, C2 | 1 | JWT = відсутній | HTTP 500 | – НЕ надсилати POST /expert-answers;– показати помилку завантаження. | Виконується POST (з user.connect = undefined), потім GET 500 → console.error. |
| A2, C1 | 2 | JWT = відсутній | HTTP 200 | – Лише GET + відобразити results;– жодного POST. | Спочатку POST (з user.connect = undefined), потім GET і відображення results. |
| A1, C1 | 3 | JWT = присутній | HTTP 200 | – POST + alert;– НЕ виконувати GET і НЕ відображати results. | POST + alert, але далі виконується GET і рендериться results. |
| A1, C2 | 4 | JWT = присутній | HTTP 500 | – POST + alert;– НЕ виконувати GET. | POST + alert, потім GET 500 → console.error; results не відображаються (програма падає). |

Тестування методом еквівалентного розбиття для випадку 2 наведено в таблицях 4.5 – 4.6.

Таблиця 4.5 – класи еквівалентності для випадку 2

|  |  |  |
| --- | --- | --- |
| Вхідні умови | Правильні класи еквівалентності | Неправильні класи еквівалентності |
| Авторизація (JWT) | A1: токен присутній | A2: токен відсутній |
| Доступ до категорій | C1: GET /api/categories → 200 | C2: GET /api/categories → ≠200 |

Таблиця 4.6 – класи еквівалентності для випадку 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Клас | Тест | Вхід | Очікувана поведінка | | Фактична поведінка | | --- |  |  | | --- | |  | |
| A1, C1 | 1 | jws = 'valid-token'GET /categories → 200 | Виконується POST → alert('✅…'); GET не виконується → results не рендериться | Виконується POST → alert('✅…'); GET не виконується → results не рендериться |
| A1, C2 | 2 | jws = 'valid-token'GET /categories → 403 або 500 | Виконується POST → alert('✅…'); GET не виконується → results не рендериться | Виконується POST → alert('✅…'); GET не виконується → results не рендериться |
| A2, C1 | 3 | jws = nullGET /categories → 200 | POST не виконується; Виконується GET → обчислення косинусної схожості → рендер топ-3 | Фактично помилка не відображються рекомендації щодо спеціальностей (категорій). |
| A2, C2 | 4 | jws = nullGET /categories → 500 | POST не виконується; Виконується GET → console.error('❌ CATEGORIES fetch failed: 500…'); results не рендериться | POST не виконується; Виконується GET → console.error('❌ CATEGORIES fetch failed: 500…'); results не рендериться |

## **. Налагодження програми**

В таблиці 4.7 наведено дії по налагодженню програми, де описано помилку на етапі тестування.

Таблиця 4.7 – Налагодження програми

|  |  |  |
| --- | --- | --- |
| Опис помилки | Опис ситуації | Дії для усунення |
| Відправка вектора відповідей неавторизованим користувачем | Незалогінений користувач натискає «Завершити тест», у handleSubmit() відсутня перевірка JWT → завжди виконується POST /api/expert-answers | Додати перевірку на наявність валідного JWT перед POST-запитом. |
| Рекомендації не відображаються незалогованому користувачу | Незалогінений користувач проходить тест, результат (top-3) обчислюється, але на UI нічого не показує, бо в Strapi для ролі Public не включені права доступу (find/findOne) до колекції category | У панелі Strapi: Settings → Roles → Public → Permissions → знайти Category → поставити галочки find та findOne → Save |

На рисунку 4.3 показано одне з виправлень.

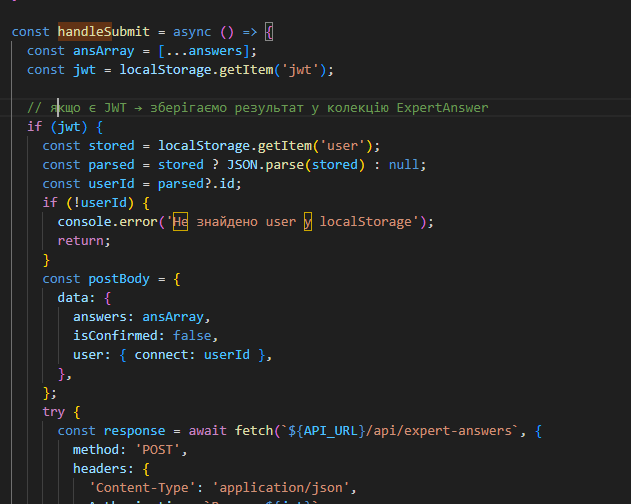


Рисунок 4.3 – ­ Виправлення помилки

# **ВИСНОВКИ**

У результаті розробки дипломної роботи було створено веб-додаток для профорієнтації абітурієнтів, що включає модулі тестування, перегляду спеціальностей та інформаційні сторінки про університет.

Даний веб-додаток полегшує процес вибору майбутньої спеціальності: студенти можуть відповісти на серію запитань, отримати персоналізовані рекомендації у вигляді топ-3 спеціальностей, а також ознайомитися з детальними описами та перевагами кожної.

Адміністратори отримали розширені інструменти для ведення обліку та модерації: можуть оперативно змінювати, додавати або видаляти інформацію про спеціальності й результати тестування, а також налаштовувати параметри самої системи через інтуїтивно зрозумілий інтерфейс Strapi. Це сприяє швидкому реагуванню на потреби закладу та збереженню актуальності даних.

В основу реалізації покладено сучасні веб-технології: фронтенд виконано на React із використанням Material-UI для швидкого створення адаптивного інтерфейсу, бекенд організовано на Strapi з базою даних SQLite. Така комбінація забезпечує високий рівень продуктивності, безперервну інтеграцію нових модулів і гнучке масштабування рішення відповідно до зростання навантаження.

# **БІБЛІОГРАФІЧНИЙ СПИСОК**

* 1. Дія.Освіта – Профорієнтаційний тест [Електронний ресурс]. – Режим доступу: https://osvita.diia.gov.ua/prof-orientation-quiz – Назва з екрана.
  2. Hryoutest – Профорієнтаційне тестування для молоді [Електронний ресурс]. – Режим доступу: https://hryoutest.in.ua/ – Назва з екрана.
  3. CareerExplorer for Organizations [Електронний ресурс]. – Режим доступу: https://www.careerexplorer.com/for-organizations/ – Назва з екрана.
  4. Arizona State University. ME3 Career Matching Tool [Електронний ресурс]. – Режим доступу: https://yourfuture.asu.edu/me3 – Назва з екрана.
  5. Careerizma – Free Career Test [Електронний ресурс]. – Режим доступу: https://www.careerizma.com/app/ct/freecareertest.php – Назва з екрана.
  6. TypeScript – Documentation [Електронний ресурс]. – Режим доступу: https://www.typescriptlang.org/docs/ – Назва з екрана.
  7. React – API Reference [Електронний ресурс]. – Режим доступу: https://react.dev/reference/react – Назва з екрана.
  8. Strapi – Headless CMS Documentation [Електронний ресурс]. – Режим доступу: https://docs.strapi.io/cms/intro – Назва з екрана.
  9. SQLite – Documentation [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/docs.html – Назва з екрана.
  10. Microsoft Learn – ASP.NET MVC Overview [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/aspnet/mvc/overview/ – Назва з екрана.
  11. Якість програмного забезпечення та тестування [Текст]: методичні вказівки до лабораторних робіт / уклад.: В. І. Шинкаренко, О. С. Куроп’ятник, Г. В. Забула, Д. О. Пєтін, Є. В. Лукін, Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во ПФ «Стандарт-Сервіс», 2018. – 50 с.

# **ДОДАТОК А**

**Технічне завдання**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ

Перший проректор Українського

державного університету

науки і технологій

Анатолій РАДКЕВИЧ

ВЕБ-ДОДАТОК ДЛЯ ПРОФОРІЄНТАЦІЇ АБІТУРІЄНТІВ

Технічне завдання

ЛИСТ ЗАТВЕРДЖЕННЯ

44165850.1507-01-ЛЗ

Завідувач кафедри КІТ

Вадим ГОРЯЧКІН

Керівник розробки

Олександр ЖЕВАГО

Виконавець

Артем БЄЛІКОВ

Нормоконтролер

Світлана ВОЛКОВА

2025

ЗАТВЕРДЖЕНО

44165850.1507-01-ЛЗ



Технічне завдання

44165850.1507-01

Листів 12

202544165850.1507-01

2

ЗМІСТ

1. Введення3
2. Підстава для розробки4
3. Призначення розробки5
4. Вимоги до програмного продукту6
   1. Вимоги до функціональних характеристик6
   2. Вимоги до надійності6
   3. Умови експлуатації6
   4. Вимоги до складу і параметрів технічних засобів7
   5. Вимоги до інформаційної і програмної сумісності7
   6. Вимоги до маркування і упаковки8
   7. Вимоги до транспортування і зберігання8
5. Вимоги до програмної документації9
6. Стадії та етапи розробки10
7. Порядок контролю і приймання11
8. Бібліографічнийсписок12

44165850.1507-01

3

1. ВВЕДЕННЯ

Назва програмного продукту: Веб-додаток для профорієнтації абітурієнтів.

Веб-додаток створений для допомоги абітурієнтам у виборі спеціальності через проходження профорієнтаційного тесту. Вона враховує специфіку освітніх програм університету, інтереси, навички та досвід студентів.

Основна термінологія:

Абітурієнт – особа, яка планує вступ до вищих навчальних закладів.

Профорієнтаційне тестування – визначення схильностей до професій.

Веб-додаток – програмне забезпечення, що працює у веб-браузері та забезпечує користувачам доступ до певного функціоналу через інтернет.

Фронтенд – інтерфейс для взаємодії з користувачем.

Бекенд – серверна частина, що обробляє дані.

База даних (БД) – сховище інформації.

CMS – система керування контентом.

React – бібліотека для створення інтерфейсу.

Node.js – серверне середовище для JavaScript.

API – спосіб взаємодії між програмами.

Причина виникнення розробки веб-додатку є потреба у створенні більш гнучкої та адаптованої до потреб конкретного закладу освіти системи профорієнтації.

Галузь застосування: система призначена для використання в межах Українського державного університету науки і технологій. Вона орієнтована на абітурієнтів, які планують вступ до цього закладу, та допомагає їм обрати спеціальність, що найбільше відповідає їхнім інтересам і навичкам. Додаток може бути використаний як автоматизований інструмент підтримки вступної кампанії, що дозволяє абітурієнтам самостійно пройти профорієнтаційне тестування.

44165850.1507-01

4

1. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ від 03.03.2025 року №328ст виконуючого обов’язки ректора Українського державного університету науки і технологій «Про призначення керівників та затвердження тем бакалаврських робіт» за спеціальністю 121 Інженерія програмного забезпечення факультету «Комп’ютерних технологій і систем» по кафедрі «Комп’ютерні інформаційні технології».

Тема дипломної роботи — «Розробка веб-додатку для профорієнтації абітурієнтів з урахуванням специфіки спеціальностей університету». Керівник — Жеваго О.О.

44165850.1507-01

5

1. ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення програми: програма забезпечує проходження профорієнтаційного тестування, аналіз результатів та надання рекомендацій щодо вибору спеціальності. Система підтримує створення та редагування спеціальностей, тестових питань, формувати тести. Також реалізовано механізм формування та оновлення еталонних даних, що використовуються для подальшого порівняння відповідей анонімних користувачів і видачі персоналізованих рекомендацій.

Експлуатаційне призначення програми: додаток автоматизує процес профорієнтації, скорочує потребу в особистих консультаціях та спрощує вибір напряму навчання для абітурієнтів.

44165850.1507-01

6

4 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

4.1 Вимоги до функціональних характеристик

Програма повинна надавати користувачам можливість проходження профорієнтаційного тестування, отримання персоналізованих рекомендацій, перегляду інформації про освітні напрями та спеціальності.

Основні функції веб-додатку:

– реєстрація та автентифікація користувачів з розподілом за ролями (адміністратор, викладач, експерт);

– створення, редагування та перегляд переліку питань, спеціальностей, користувачів, відповідей експертів;

– автоматичне формування синтетичних еталонних даних та оновлення їх на основі відповідей експертів;

– аналіз відповідей анонімних користувачів з порівнянням до еталонних даних і генерація рекомендацій;

– адміністративний доступ для керування користувачами, контентом, тестами та еталонними даними.

4.2 Вимоги до надійності

Вимоги до надійності наступні:

– збереження результатів тестування: усі відповіді, надані користувачами, мають зберігатися в базі даних;

– контроль валідності даних: система перевіряє відповідність довжини вектора відповідей до кількості актуальних питань перед обробкою результату; необхідні поля (наприклад, answers, user) не можуть бути порожніми при створенні запису експертної відповіді;

– безпечна обробка змін адміністратора: зміни, внесені адміністратором автоматично синхронізуються без втрати даних.

4.3 Умови експлуатації

Вимоги до експлуатації наступні:

44165850.1507-01

7

– рекомендований мікроклімат приміщення: температура 21–25 °C, вологість 40–60%, швидкість повітря ≤ 0,1 м/с;

– рівень шуму: ≤ 50 дБА;

– освітлення: освітленість на робочих поверхнях столів повинна бути не нижче 400 люкс, а на поверхні екрана — не більше 200 люкс;

– ергономіка: відстань до екрану 50–70 см, верхній край на рівні очей, регульовані стіл і стілець;

– режим роботи: рекомендується робити перерви тривалістю 5–10 хвилин після кожної години роботи за комп'ютером для зниження втоми та напруження очей.

Дані вимоги ґрунтуються на ДСанПіН 3.3.2.007-98 та ДСН 3.3.6.042-99.

4.4 Вимоги до складу і параметрів технічних засобів

Для експлуатації програмного продукту необхідні такі системні вимоги:

— процесор: Intel Core i3 / AMD Ryzen 3 або вище;

— оперативна пам’ять: 4 ГБ або більше;

— графічний адаптер: вбудоване відеоядро з підтримкою OpenGL 3.0 або вище;

— дисплей: мінімальна роздільна здатність – 1920×1080;

— підключення до Інтернету: мінімальна швидкість – 15 Мбіт/с;

— операційна система: Windows 10/11, macOS, Linux, Android, iOS;

— браузери: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari (останні версії);

— мова інтерфейсу: українська / англійська;

— вхідні пристрої: клавіатура, миша або сенсорний екран;

— доступ до USB-порту (за необхідності завантаження файлів).

4.5 Вимоги до інформаційної і програмної сумісності

Вимоги до інформаційної і програмної сумісності наступні:

– підтримка форматів обміну даними JSON і CSV;

44165850.1507-01

8

– програмна сумісність із сучасними браузерами та мобільними пристроями.

4.6 Вимоги до маркування і упаковки

Вимоги до маркування і упаковки наступні:

Маркування повинно містити назву Веб-додатка для профорієнтації абітурієнтів з урахуванням специфіки спеціальностей університету, версію програми, дату випуску та контактні дані технічної підтримки.

На рис. 1 приведений приклад маркування.

|  |
| --- |
| Веб-додаток для профорієнтації абітурієнтів з урахуванням специфіки спеціальностей університету Розробник: Бєліков А. Є.  УДУНТ, кафедра КІТ  м. Дніпро, вул. Лазаряна 2  2025 |

Рис. 1. Приклад маркування

4.7. Вимоги до транспортування і зберігання

Вимоги до транспортування і зберігання наступні:

Транспортування повинне забезпечувати збереження програмного продукту, його цілісність і запобігання несанкціонованому доступу до нього. Програмний виріб міститься на USB флеш-носії і повинен мати відповідну упаковку для захисту від механічних ушкоджень та атмосферного впливу (пластиковий футляр).

44165850.1507-01

9

1. ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Програмна документація повинна складатися з наступних документів:

* специфікація;
* текст програми;
* керівництво користувача.

Вся документація до програмного продукту повинна задовольняти вимоги до програмної документації [1].

44165850.1507-01

10

1. СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

|  |  |  |
| --- | --- | --- |
| Стадія | Зміст робіт | Терміни виконання |
| Технічне завдання | Постановка завдання, збір початкових матеріалів, вибір і обґрунтування критеріїв ефективності і якості. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання. | 03.03.25 – 31.03.25 |
| Робочий проект | Програмування та налагодження програми | 01.04.25 ­– 23.05.25 |
| Тестування програми | 24.05.25 – 31.05.25 |
| Розробка, узгодження і затвердження програмної документації | 01.06.25 – 13.06.25 |

44165850.1507-01

11

1. ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ

Контроль за виконанням роботи здійснює керівник розробки Жеваго О.О.

Прийом програмного продукту здійснюється уповноваженою комісією.

44165850.1507-01

12

1. БІБЛІОГРАФІЧНИЙ СПИСОК

1. Івченко, Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетровський національний університет залізничного транспорту ім. акад. В. Лазаряна. – Д.: Видавництво Дніпропетровський національний університет залізничного транспорту ім. акад. В. Лазаряна, 2009. - 38 с.

# **ДОДАТОК Б**

**Специфікація**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ

Перший проректор Українського

державного університету

науки і технологій

Анатолій РАДКЕВИЧ

ВЕБ-ДОДАТОК ДЛЯ ПРОФОРІЄНТАЦІЇ АБІТУРІЄНТІВ

Специфікація

ЛИСТ ЗАТВЕРДЖЕННЯ

44165850.1507-01-ЛЗ

Завідувач кафедри КІТ

Вадим ГОРЯЧКІН

Керівник розробки

Олександр ЖЕВАГО

Виконавець

Артем БЄЛІКОВ

Нормоконтролер

Світлана ВОЛКОВА

2025

ЗАТВЕРДЖЕНО

44165850.1507-01-ЛЗ

ВЕБ-ДОДАТОК ДЛЯ ПРОФОРІЄНТАЦІЇ АБІТУРІЄНТІВ

Специфікація

44165850.1507-01

Листів 2

2025

44165850.1507-01

3

|  |  |  |
| --- | --- | --- |
| Позначення | Найменування | Примітка |
|  | Документація |  |
| 44165850.1507-01-ЛЗ | Лист затвердження |  |
| 44165850.1507-01 12 01-ЛЗ | Лист затвердження |  |
| 44165850.1507-01 12 01 | Текст програми |  |
| 44165850.1507-01 І3 01-ЛЗ | Лист затвердження |  |
| 44165850.1507-01 І3 01 | Керівництво користувача |  |

# **ДОДАТОК В**

**Текст програми**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ

Перший проректор Українського

державного університету

науки і технологій

Анатолій РАДКЕВИЧ

ВЕБ-ДОДАТОК ДЛЯ ПРОФОРІЄНТАЦІЇ АБІТУРІЄНТІВ

Текст програми

ЛИСТ ЗАТВЕРДЖЕННЯ

44165850.1507-01 12 01-ЛЗ

Завідувач кафедри КІТ

Вадим ГОРЯЧКІН

Керівник розробки

Олександр ЖЕВАГО

Виконавець

Артем БЄЛІКОВ

Нормоконтролер

Світлана ВОЛКОВА

2025

ЗАТВЕРДЖЕНО

44165850.1507-01 12 01-ЛЗ

ВЕБ-ДОДАТОК ДЛЯ ПРОФОРІЄНТАЦІЇ АБІТУРІЄНТІВ

Текст програми

44165850.1507-01 12 01

Листів 24

2025

44165850.1507-01 12 01

2

АНОТАЦІЯ

Документ 44165850.1507-01 12 01 «Веб-додаток для профорієнтації абітурієнтів. Текст програми» входить до складу програмної документації на програму, що реалізує систему профорієнтації для абітурієнтів, яка дозволяє проводити тестування, аналіз результатів та надання рекомендацій щодо вибору професії та освітнього напряму.

У даному документі представлений текст програми. Програма написана на мовах JavaScript, TypeScript, CSS, SQL. Об’єм пам’яті, що займає програма комплексу складає 1,34 ГБ. Конфігурація комп’ютера стандартна. Комплекс функціонує в середовищі MS WINDOWS 11.

44165850.1507-01 12 01

3

ЗМІСТ

1. Схема взаємодії компонентів4
2. Текст програми5
   1. Strapi5
   2. Frontend9

44165850.1507-01 12 01

4

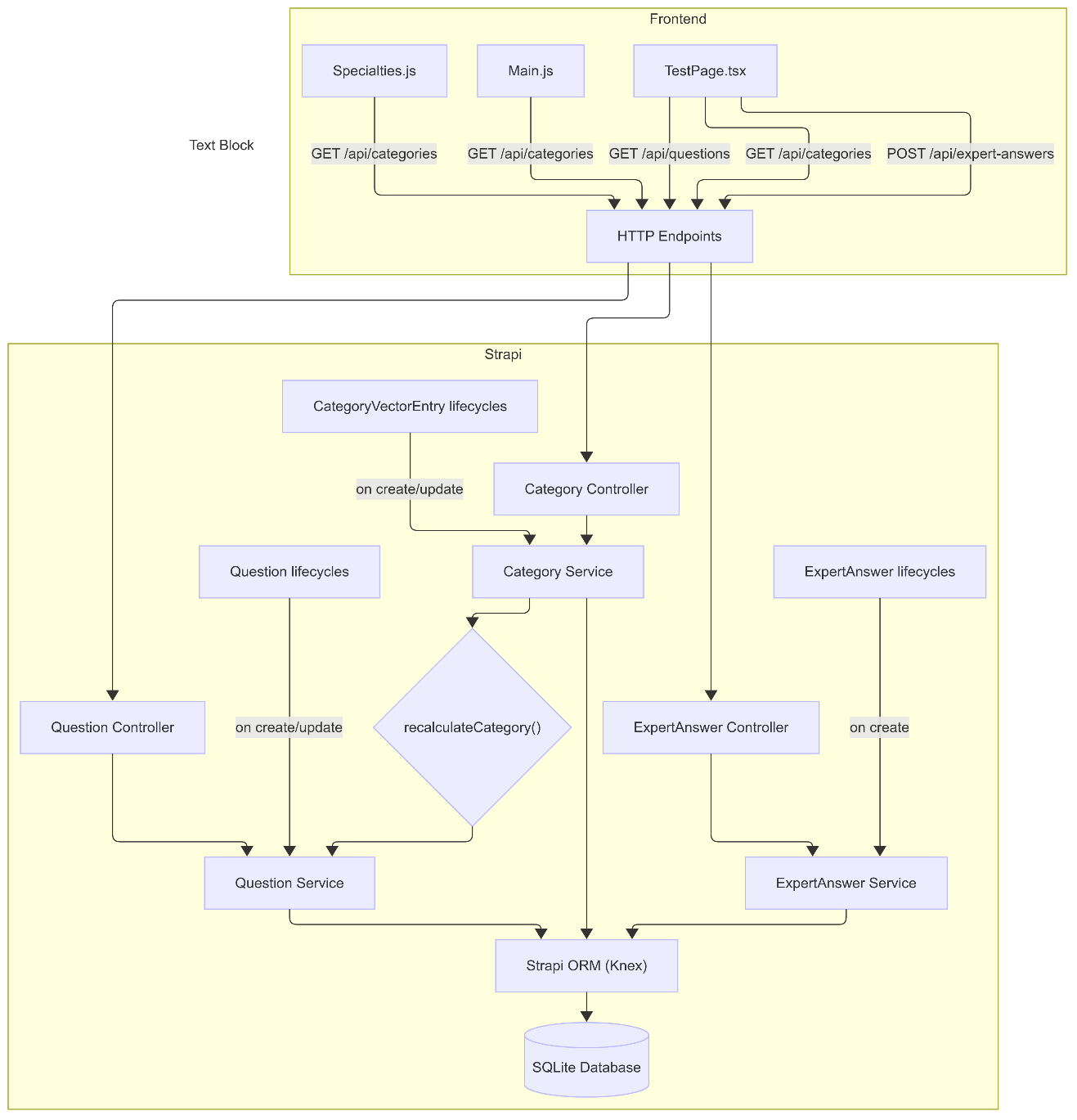
1 CХЕМА ВЗАЄМОДІЇ КОМПОНЕНТІВ

Рисунок В.1. – Схема взаємодії компонентів системи

44165850.1507-01 12 01

5

2 ТЕКСТ ПРОГРАМИ

* 1. Strapi

Файли lifecycles.ts і відповідні сервіси реалізують бізнес-логіку Strapi для обробки подій моделей (створення, оновлення, видалення), забезпечення цілісності записів CategoryVectorEntry та автоматичного перерахунку поля vector у категоріях.

Файл content-types\category\lifecycles.ts

declare const strapi: any;

interface Category { id: number; isVerified: boolean; }

async function populateCategoryVectorEntries(categoryId: number): Promise<void> {

strapi.log.info(`Start populate for categoryId=${categoryId}`);

const questions = await strapi.entityService.findMany('api::question.question', {

fields: ['id'],

publicationState: 'live',

sort: { order: 'asc' },

});

const qids = questions.map(q => q.id).join(', ');

strapi.log.info(`Question IDs: [${qids}]`);

let createdCount = 0;

for (const { id: questionId } of questions) {

const exists = await strapi.db.query('api::category-vector-entry.category-vector-entry')

.findOne({ where: { category: categoryId, question: questionId } });

if (exists) continue;

strapi.log.info(`Creating CVE for category=${categoryId}, question=${questionId}`);

try {

const created = await strapi.entityService.create('api::category-vector-entry.category-vector-entry', {

data: {

category: { connect: { id: categoryId } },

question: { connect: { id: questionId } },

value: 3,

},

populate: ['category', 'question'],

});

createdCount++;

strapi.log.info(`Created entryId=${created.id}`);

} catch (err) {

strapi.log.error(`Error creating CVE for questionId=${questionId}`, err);

}

}

strapi.log.info(`Finished populate for categoryId=${categoryId}, created=${createdCount}`);

}

export default {

async afterCreate(event: any) {

if (event.model?.uid !== 'api::category.category') return;

44165850.1507-01 12 01

6

const { id, isVerified }: Category = event.result;

if (isVerified) {

await populateCategoryVectorEntries(id);

}

},

async afterUpdate(event: any) {

if (event.model?.uid !== 'api::category.category') return;

const { id, isVerified }: Category = event.result;

const was = event.params.data.isVerified;

if (was === true && isVerified) {

await populateCategoryVectorEntries(id);

}

},

};

Файл services\category\recalculateCategory.ts.ts

declare const strapi: any;

export async function recalculateCategory(strapi: any, categoryId: number): Promise<void> {

strapi.log.info(`[recalculateCategory] START categoryId=${categoryId}`);

const questions = await strapi.entityService.findMany('api::question.question', {

fields: ['id'],

publicationState: 'live',

sort: { order: 'asc' },

}) as Array<{ id: number }>;

for (const { id: questionId } of questions) {

const exists = await strapi.db

.query('api::category-vector-entry.category-vector-entry')

.findOne({ where: { category: categoryId, question: questionId } });

if (!exists) {

await strapi.entityService.create('api::category-vector-entry.category-vector-entry', {

data: {

category: { connect: { id: categoryId } },

question: { connect: { id: questionId } },

value: 0,

},

});

strapi.log.debug(`CVE created for category=${categoryId}, question=${questionId}`);

}

}

const rawEntries = await strapi.entityService.findMany(

'api::category-vector-entry.category-vector-entry',

{

filters: { category: { id: categoryId } },

populate: { question: { fields: ['order'] } },

}

) as Array<{ value: number; question: { order: number } }>;

const vector = rawEntries

.sort((a, b) => a.question.order - b.question.order)

.map(e => e.value);

await strapi.entityService.update(

'api::category.category',

categoryId,

44165850.1507-01 12 01

7

{ data: { vector } }

);

strapi.log.info(

`[recalculateCategory] DONE categoryId=${categoryId}, vector=[${vector.join(', ')}]`

);

}

Файл content-types\category-vector-entry\lifecycles.ts

declare const strapi: any;

interface CategoryVectorEntry { value: number; question: { order: number }; }

async function updateCategoryVectorAt(categoryId: number, questionOrder: number, newValue: number) {

const cat = await strapi.entityService.findOne('api::category.category', categoryId, { fields: ['vector'] });

const vector: number[] = Array.isArray(cat.vector) ? cat.vector : [];

const raws = await strapi.entityService.findMany('api::category-vector-entry.category-vector-entry', {

filters: { category: { id: categoryId } },

populate: { question: { fields: ['order'] } },

});

const entries = raws as CategoryVectorEntry[];

const sortedOrders = entries.map(e => e.question.order).sort((a, b) => a - b);

const idx = sortedOrders.indexOf(questionOrder);

if (idx === -1) return;

vector[idx] = newValue;

await strapi.entityService.update('api::category.category', categoryId, { data: { vector } });

}

async function updateCategoryVector(categoryId: number) {

const raws = await strapi.entityService.findMany('api::category-vector-entry.category-vector-entry', {

filters: { category: { id: categoryId } },

populate: { question: { fields: ['order'] } },

});

const entries = (raws as CategoryVectorEntry[]).sort((a, b) => a.question.order - b.question.order);

const vector = entries.map(e => e.value);

await strapi.entityService.update('api::category.category', categoryId, { data: { vector } });

}

export default {

async afterCreate(event: { result: any }) {

const { id } = event.result as { id: number };

const fresh = await strapi.entityService.findOne('api::category-vector-entry.category-vector-entry', id, { populate: ['question', 'category'] });

await updateCategoryVectorAt(fresh.category.id, fresh.question.order, fresh.value);

},

async afterUpdate(event: { result: any }) {

const { id } = event.result as { id: number };

const fresh = await strapi.entityService.findOne('api::category-vector-entry.category-vector-entry', id, { populate: ['question', 'category'] });

await updateCategoryVectorAt(fresh.category.id, fresh.question.order, fresh.value);

},

44165850.1507-01 12 01

8

async beforeDelete(event: { params: { where: { id: number }; categoryId?: number } }) {

const id = event.params.where.id;

const entry = await strapi.entityService.findOne('api::category-vector-entry.category-vector-entry', id, { populate: ['category'] });

event.params.categoryId = (entry as any).category.id;

},

async afterDelete(event: { params: { categoryId?: number } }) {

const categoryId = event.params.categoryId;

if (categoryId) await updateCategoryVector(categoryId);

},

};

Файл content-types\expert-answer\lifecycles.ts

declare const strapi: any;

export default {

async afterCreate(event: { result: any }) {

const created = event.result as any;

if (!created.isConfirmed) return;

const full = await strapi.entityService.findOne(

'api::expert-answer.expert-answer',

created.id,

{ populate: ['category'] }

);

const categoryId = (full as any).category?.id;

if (!categoryId) return;

strapi.log.info(`confirmed answer id=${created.id}, categoryId=${categoryId}`);

await recalcCVE(categoryId);

},

async afterUpdate(event: { result: any; params: { data: any } }) {

if (event.params.data.isConfirmed !== true) return;

const full = await strapi.entityService.findOne(

'api::expert-answer.expert-answer',

event.result.id,

{ populate: ['category'] }

);

const categoryId = (full as any).category?.id;

if (!categoryId) return;

strapi.log.info(`confirmed answer id=${full.id}, categoryId=${categoryId}`);

await recalcCVE(categoryId);

},

};

async function recalcCVE(categoryId: number) {

strapi.log.info(`recalcCVE START categoryId=${categoryId}`);

const answers = await strapi.entityService.findMany(

'api::expert-answer.expert-answer',

{

filters: { category: { id: categoryId }, isConfirmed: true },

fields: ['answers'],

}

) as Array<{ answers: number[] }>;

if (answers.length === 0) return;

const arrays = answers.map(a => a.answers);

const maxLen = Math.max(...arrays.map(a => a.length));

const sums = Array(maxLen).fill(0);

const counts = Array(maxLen).fill(0);

44165850.1507-01 12 01

9

arrays.forEach(arr => {

arr.forEach((v, i) => {

sums[i] += v;

counts[i] += 1;

});

});

const means = sums.map((s, i) => (counts[i] > 0 ? s / counts[i] : 0));

const rawCves = await strapi.entityService.findMany(

'api::category-vector-entry.category-vector-entry',

{

filters: { category: { id: categoryId } },

populate: { question: { fields: ['order'] } },

}

) as Array<{ id: number; question: { order: number } }>;

for (const { id, question } of rawCves) {

const ord = question.order;

const newVal = means[ord - 1] ?? 0;

await strapi.entityService.update(

'api::category-vector-entry.category-vector-entry',

id,

{ data: { value: newVal } }

);

strapi.log.info(`CVE id=${id}, order=${ord}, value=${newVal}`);

}

strapi.log.info(`recalcCVE END categoryId=${categoryId}`);

}

Файл content-types\question\lifecycles.ts

declare const strapi: any;

import { recalculateCategory } from '../../../category/services/recalculateCategory';

export default {

async beforeCreate(event: { params: { data: any } }) {

const { data } = event.params;

if (data.order == null) {

const existing = await strapi.entityService.findMany('api::question.question', {

sort: { order: 'desc' },

limit: 1,

fields: ['order'],

});

const max = existing[0]?.order ?? 0;

data.order = max + 1;

}

},

async afterCreate() { await recalcAllCats(); },

async afterUpdate() { await recalcAllCats(); },

async afterDelete() { await recalcAllCats(); },

};

async function recalcAllCats() {

const cats = await strapi.entityService.findMany('api::category.category', { fields: ['id'] });

for (const { id } of cats) {

await recalculateCategory(strapi, id);

}

}

44165850.1507-01 12 01

9

* 1. Frontend

Файли фронтенду включають Header.tsx із адаптивною навігацією та модалями для входу й тесту, Footer.jsx із логотипом, контактами та соцмережами, Main.tsx і University.tsx зі статичним контентом, Specialties.jsx із акордеоном описів спеціальностей та TestPage.tsx із логікою завантаження питань, введення відповідей та їх відправлення на сервер, і всі вони взаємодіють із Strapi API, обробляючи стани завантаження, помилок і успіху.

Файл components\Footer.js

import React from 'react';

import './Footer.css';

const Footer = () => (

<footer className="footer-container">

<div className="footer-content">

<div className="footer-logo-container">

<a href="https://ust.edu.ua/" target="\_blank" rel="noopener noreferrer">

<img src="/images/logo2.png" alt="University Logo" className="footer-logo" />

</a>

</div>

<div className="footer-info-container">

<div className="footer-university-name">

<a

href="https://ust.edu.ua/"

target="\_blank"

rel="noopener noreferrer"

className="footer-university-link"

>

Український державний університет науки і технологій

</a>

</div>

<div className="footer-info-text">

49010, Україна, м. Дніпро, вул. Лазаряна, 2

</div>

<div className="footer-info-text">

Email: <a href="mailto:office@ust.edu.ua" className="footer-info-link">office@ust.edu.ua</a>

</div>

<div className="footer-info-text">

Приймальня ректора:

<a href="tel:+380567931900" className="footer-info-link">+38 (056) 793-19-00</a>;

<a href="tel:+380563731544" className="footer-info-link">+38 (056) 373-15-44</a>

</div>

<div className="footer-info-text">

Загальний відділ: <a href="tel:+380563731505" className="footer-info-link">+38 (056) 373-15-05</a>

</div>

</div>

<div className="footer-social-container">

44165850.1507-01 12 01

10

<div className="footer-social-links">

<a

href="https://www.facebook.com/university.ust"

target="\_blank"

rel="noopener noreferrer"

className="footer-social-icon"

>

<img src="/images/facebook-icon.png" alt="Facebook" />

</a>

<a

href="https://t.me/official\_ust"

target="\_blank"

rel="noopener noreferrer"

className="footer-social-icon"

>

<img src="/images/telegram-icon.png" alt="Telegram" />

</a>

<a

href="https://www.youtube.com/@ust.official"

target="\_blank"

rel="noopener noreferrer"

className="footer-social-icon"

>

<img src="/images/youtube-icon.png" alt="YouTube" />

</a>

<a

href="https://www.instagram.com/ust.university/"

target="\_blank"

rel="noopener noreferrer"

className="footer-social-icon"

>

<img src="/images/instagram-icon.png" alt="Instagram" />

</a>

</div>

</div>

</div>

</footer>

);

export default Footer;

Файл components\Header.tsx

import React, { useState, useEffect } from 'react';

import {

AppBar,

Toolbar,

Typography,

Button,

IconButton,

Box,

useMediaQuery,

useTheme,

Dialog,

Menu,

MenuItem,

} from '@mui/material';

import MenuIcon from '@mui/icons-material/Menu';

import { Link as RouterLink } from 'react-router-dom';

import './Header.css';

import LoginForm from './LoginForm';

import TestPage from './TestPage';

44165850.1507-01 12 01

11

type NavItem = { label: string; href: string } | { label: string; action: () => void };

export default function Header() {

const theme = useTheme();

const isMobile = useMediaQuery(theme.breakpoints.down('md'));

const [user, setUser] = useState<{ username: string } | null>(null);

const [loginOpen, setLoginOpen] = useState(false);

const [testOpen, setTestOpen] = useState(false);

const [mobileAnchor, setMobileAnchor] = useState<null | HTMLElement>(null);

useEffect(() => {

const stored = localStorage.getItem('user');

if (stored) {

try {

const u = JSON.parse(stored);

setUser({ username: u.username || u.email || 'Експерт' });

} catch {}

}

}, []);

const baseNav: NavItem[] = [

{ label: 'Головна', href: '/' },

{ label: 'Університет', href: '/university' },

{ label: 'Тест', action: () => setTestOpen(true) },

{ label: 'Спеціальності', href: '/specialties' },

];

if (user) {

baseNav.push(

{ label: `Привіт, ${user.username}`, action: () => {} },

{

label: 'Вийти',

action: () => {

localStorage.removeItem('jwt');

localStorage.removeItem('user');

setUser(null);

},

}

);

} else {

baseNav.push({ label: 'Увійти', action: () => setLoginOpen(true) });

}

const openMenu = (e: React.MouseEvent<HTMLElement>) => setMobileAnchor(e.currentTarget);

const closeMenu = () => setMobileAnchor(null);

return (

<>

<AppBar position="static" color="primary" elevation={0}>

<Toolbar className="header-toolbar">

<RouterLink to="/" className="header-brand">

<img src="/images/logo.png" alt="Logo" className="header-logo\_\_img" />

<Typography variant="h6" className="header-logo\_\_text">

Профорієнтація абітурієнтів

</Typography>

</RouterLink>

{isMobile ? (

<>

44165850.1507-01 12 01

12

<IconButton edge="end" color="inherit" onClick={openMenu}>

<MenuIcon />

</IconButton>

<Menu anchorEl={mobileAnchor} open={!!mobileAnchor} onClose={closeMenu}>

{baseNav.map((item) => (

<MenuItem

key={item.label}

onClick={() => {

closeMenu();

if ('href' in item) window.location.href = item.href;

else item.action();

}}

>

{item.label}

</MenuItem>

))}

</Menu>

</>

) : (

<Box className="header-nav">

{baseNav.map((item) =>

'href' in item ? (

<Button

key={item.label}

className="header-nav\_\_button"

component={RouterLink}

to={item.href}

color="inherit"

disableRipple

>

{item.label}

</Button>

) : (

<Button

key={item.label}

className="header-nav\_\_button"

onClick={item.action}

color="inherit"

disableRipple

>

{item.label}

</Button>

)

)}

</Box>

)}

</Toolbar>

</AppBar>

<Dialog open={loginOpen} onClose={() => setLoginOpen(false)} maxWidth="xs" fullWidth>

<LoginForm

onLoginSuccess={(uname) => {

setUser({ username: uname });

setLoginOpen(false);

}}

/>

</Dialog>

<Dialog open={testOpen} onClose={() => setTestOpen(false)} maxWidth="md" fullWidth>

<TestPage />

44165850.1507-01 12 01

13

</Dialog>

</>

);

}

Файл components\LoginForm.tsx

import React, { useState } from 'react';

import './LoginForm.css';

interface LoginFormProps {

onLoginSuccess: (username: string) => void;

}

const LoginForm: React.FC<LoginFormProps> = ({ onLoginSuccess }) => {

const [identifier, setIdentifier] = useState('');

const [password, setPassword] = useState('');

const [showPwd, setShowPwd] = useState(false);

const [error, setError] = useState('');

const [success, setSuccess] = useState(false);

const handleLogin = async (e: React.FormEvent) => {

e.preventDefault();

setError('');

try {

const res = await fetch('http://localhost:1337/api/auth/local', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ identifier, password }),

});

const data = await res.json();

if (res.ok) {

localStorage.setItem('jwt', data.jwt);

localStorage.setItem('user', JSON.stringify(data.user));

setSuccess(true);

onLoginSuccess(data.user.username || data.user.email);

} else {

setError(data.error?.message || 'Login failed');

}

} catch {

setError('Network error');

}

};

return (

<div className="login-form">

<h2 className="login-form\_\_title">Увійти як експерт</h2>

<form onSubmit={handleLogin}>

<div className="login-form\_\_field">

<input

type="text"

placeholder="Email"

value={identifier}

onChange={(e) => setIdentifier(e.target.value)}

required

/>

</div>

<div className="login-form\_\_field login-form\_\_field--password">

<input

type={showPwd ? 'text' : 'password'}

placeholder="Пароль"

44165850.1507-01 12 01

14

value={password}

onChange={(e) => setPassword(e.target.value)}

required

/>

<span

className="login-form\_\_toggle"

onClick={() => setShowPwd(!showPwd)}

aria-label={showPwd ? 'Сховати пароль' : 'Показати пароль'}

>

{showPwd ? '🙈' : '👁️'}

</span>

</div>

<button type="submit" className="login-form\_\_submit">

Вхід

</button>

</form>

{success && <div className="login-form\_\_success">Вхід успішний!</div>}

{error && <div className="login-form\_\_error">{error}</div>}

</div>

);

};

export default LoginForm;

Файл components\Main.js

// src/components/Main.jsx

import React from 'react';

import Slider from 'react-slick';

import { Grid, Box, Typography, useTheme, useMediaQuery } from '@mui/material';

import { useNavigate } from 'react-router-dom';

import 'slick-carousel/slick/slick.css';

import 'slick-carousel/slick/slick-theme.css';

import './Main.css';

const slides = [

{ src: '/images/slide1.jpg', alt: 'Slide 1', action: 'test' },

{ src: '/images/slide2.jpg', alt: 'Slide 2', action: '/university' },

{ src: '/images/slide3.jpg', alt: 'Slide 3', action: '/specialties' },

];

const NextArrow = ({ onClick }) => (

<Box className="main-slider\_\_arrow main-slider\_\_arrow--next" onClick={onClick}>

›

</Box>

);

const PrevArrow = ({ onClick }) => (

<Box className="main-slider\_\_arrow main-slider\_\_arrow--prev" onClick={onClick}>

‹

</Box>

);

export default function Main({ openTestDialog }) {

const theme = useTheme();

const isMobile = useMediaQuery(theme.breakpoints.down('sm'));

const isTablet = useMediaQuery(theme.breakpoints.between('sm', 'md'));

const navigate = useNavigate();

const sliderSettings = {

44165850.1507-01 12 01

15

dots: true,

infinite: true,

speed: 600,

slidesToShow: 1,

slidesToScroll: 1,

arrows: true,

autoplay: !isMobile,

autoplaySpeed: 3000,

nextArrow: <NextArrow />,

prevArrow: <PrevArrow />,

};

const handleSlideClick = (action) => {

if (action === 'test') {

openTestDialog?.();

} else {

navigate(action);

}

};

return (

<Box component="main" className="main-root">

<Grid

container

className="main-grid"

direction={isMobile || isTablet ? 'column' : 'row'}

>

{/\* Слайдер \*/}

<Grid item xs={12} md={6} className="main-slider-container">

<Slider {...sliderSettings} className="main-slider">

{slides.map(({ src, alt, action }, idx) => (

<Box

key={idx}

className="main-slide"

onClick={() => handleSlideClick(action)}

sx={{ cursor: 'pointer' }}

>

<img src={src} alt={alt} className="main-slide\_\_img" />

</Box>

))}

</Slider>

</Grid>

{/\* Текстовый блок \*/}

<Grid item xs={12} md={6} className="main-text-container">

<Typography

variant="h4"

className="main-text-container\_\_title"

gutterBottom

>

Твоя кар'єра починається прямо зараз!

</Typography>

<Typography

className="main-text-container\_\_subtitle"

paragraph

>

Український державний університет науки і технологій (УДУНТ) — це не просто

місце для навчання. Тут формується майбутнє кожного студента, відкриваються

44165850.1507-01 12 01

16

можливості для успішної кар’єри, наукових досліджень та міжнародних обмінів.

</Typography>

<Box className="main-bullet-list">

{[

{

bold: 'Понад 15 000 студентів',

text: ' у 6 інститутах та 24 факультетах, що гарантує мультидисциплінарне середовище та активний студентський рух.',

},

{

bold: '950 викладачів',

text: ' (139 професорів, понад 500 доцентів) — провідні експерти у своїх галузях, які поєднують наукові дослідження та практику.',

},

{

bold: 'Сучасна інфраструктура',

text: ': 21 гуртожиток із 100 % забезпеченням, 2 спорткомплекси (4 зали), 3 басейни, Wi-Fi на всій території, ботанічний сад та геологічний полігон.',

},

{

bold: 'Міжнародні програми',

text: ': можливість отримати подвійний диплом у партнерських університетах Франції, Норвегії, Польщі.',

},

{

bold: 'Працевлаштування та стажування',

text: ': стабільний попит на наших випускників, партнерські компанії, внутрішні стажування та гранти.',

},

].map((item, i) => (

<Typography

key={i}

component="div"

className="main-text-container\_\_body"

>

<Box component="span" className="main-bullet-marker" />

<Box component="span">

<strong>{item.bold}</strong>

{item.text}

</Box>

</Typography>

))}

</Box>

<Typography

className="main-text-container\_\_quote"

paragraph

>

“УДУНТ — це не просто освіта. Це успішний старт твоєї майбутньої кар’єри,

дружня команда однодумців і справжньо європейські можливості.”

</Typography>

<Box className="main-cta-container">

<a

href="https://pk.ust.edu.ua/"

target="\_blank"

rel="noopener noreferrer"

className="main-cta-link"

>

44165850.1507-01 12 01

17

Перейти до Приймальної комісії

</a>

</Box>

</Grid>

</Grid>

</Box>

);

}

Файл components\Specialties.js

import React, { useState, useEffect } from 'react';

import {

Box,

Typography,

Accordion,

AccordionSummary,

AccordionDetails,

CircularProgress

} from '@mui/material';

import ExpandMoreIcon from '@mui/icons-material/ExpandMore';

import ReactMarkdown from 'react-markdown';

import remarkGfm from 'remark-gfm';

import '../components/Specialties.css';

const API\_URL = process.env.REACT\_APP\_API\_URL || 'http://localhost:1337';

export default function Specialties() {

const [categories, setCategories] = useState([]);

const [loading, setLoading] = useState(true);

const [error, setError] = useState(null);

useEffect(() => {

fetch(`${API\_URL}/api/categories?pagination[pageSize]=100`)

.then(res => {

if (!res.ok) throw new Error(`Помилка завантаження: ${res.status}`);

return res.json();

})

.then(json => {

const items = Array.isArray(json.data) ? json.data : [];

const loaded = items.map(item => ({

id: item.id,

name: item.name || '',

shortDescription: item.shortDescription || '',

fullDescription: (item.fullDescription || '').replace(/\\n/g, '\n')

}));

setCategories(loaded);

})

.catch(e => {

setError(e.message || 'Невідома помилка');

})

.finally(() => {

setLoading(false);

});

}, []);

if (loading) {

return (

<Box className="specialties-loader-container">

<CircularProgress />

</Box>

44165850.1507-01 12 01

18

);

}

if (error) {

return (

<Typography color="error" align="center" className="specialties-error">

{`Помилка при завантаженні спеціальностей: ${error}`}

</Typography>

);

}

if (categories.length === 0) {

return (

<Typography align="center" className="specialties-no-data">

Спеціальностей не знайдено.

</Typography>

);

}

return (

<Box component="section" className="specialties-root">

<Typography

variant="h4"

gutterBottom

align="center"

className="specialties-title"

>

Спеціальності університету

</Typography>

<Typography

variant="body1"

paragraph

align="center"

className="specialties-intro"

>

Ознайомтеся з описами наших спеціальностей. Розкрийте будь-яку назву, щоб побачити докладний опис.

</Typography>

{categories.map(cat => (

<Accordion key={cat.id} className="specialties-accordion">

<AccordionSummary

expandIcon={<ExpandMoreIcon />}

aria-controls={`cat-${cat.id}-content`}

id={`cat-${cat.id}-header`}

>

<Box className="specialties-accordion-summary">

<Typography

variant="h6"

className="specialties-accordion-title"

>

{cat.name}

</Typography>

{cat.shortDescription && (

<Typography

variant="body2"

color="textSecondary"

className="specialties-short"

>

{cat.shortDescription}

</Typography>

)}

44165850.1507-01 12 01

19

</Box>

</AccordionSummary>

<AccordionDetails className="specialties-details">

<Box className="specialties-full">

<ReactMarkdown remarkPlugins={[remarkGfm]}>

{cat.fullDescription}

</ReactMarkdown>

</Box>

</AccordionDetails>

</Accordion>

))}

</Box>

);

}

Файл components\TestPage.tsx

// src/components/TestPage.tsx

import React, { useState, useEffect } from 'react';

import {

Box,

Typography,

Slider,

Button,

Accordion,

AccordionSummary,

AccordionDetails,

Paper,

} from '@mui/material';

import ExpandMoreIcon from '@mui/icons-material/ExpandMore';

import ReactMarkdown from 'react-markdown';

import remarkGfm from 'remark-gfm';

import './TestPage.css';

interface Question {

id: number;

text: string;

order: number;

}

interface Category {

id: number;

name: string;

vector: number[];

shortDescription: string;

fullDescription: string;

}

const API\_URL = process.env.REACT\_APP\_API\_URL || 'http://localhost:1337';

const TestPage: React.FC = () => {

const [questions, setQuestions] = useState<Question[]>([]);

const [answers, setAnswers] = useState<number[]>([]);

const [results, setResults] = useState<

{ name: string; score: number; short: string; full: string }[] | null

>(null);

const [loading, setLoading] = useState(true);

useEffect(() => {

const loadQuestions = async () => {

44165850.1507-01 12 01

20

setLoading(true);

try {

const res = await fetch(`${API\_URL}/api/questions?sort=order:asc`);

if (!res.ok) {

console.error('❌ QUESTIONS fetch failed:', res.status, await res.text());

setLoading(false);

return;

}

const json = await res.json();

const items = Array.isArray(json.data) ? json.data : [];

const loaded: Question[] = items.map((q: any) => {

const attrs = q.attributes ?? q;

return {

id: q.id,

text: attrs.text ?? '(без тексту)',

order: attrs.order ?? 0,

};

});

loaded.sort((a, b) => a.order - b.order);

setQuestions(loaded);

setAnswers(Array(loaded.length).fill(3)); // початкові значення «3» (середній рівень)

} catch (err) {

console.error('❌ Помилка завантаження питань:', err);

} finally {

setLoading(false);

}

};

loadQuestions();

}, []);

const handleAnswerChange = (index: number, value: number) => {

const updated = [...answers];

updated[index] = value;

setAnswers(updated);

};

const handleSubmit = async () => {

const ansArray = [...answers];

const jwt = localStorage.getItem('jwt');

// Якщо є JWT → зберігаємо результат у колекцію ExpertAnswer

if (jwt) {

const stored = localStorage.getItem('user');

const parsed = stored ? JSON.parse(stored) : null;

const userId = parsed?.id;

if (!userId) {

console.error('Не знайдено user у localStorage');

return;

}

const postBody = {

data: {

answers: ansArray,

isConfirmed: false,

user: { connect: userId },

},

};

try {

const response = await fetch(`${API\_URL}/api/expert-answers`, {

44165850.1507-01 12 01

21

method: 'POST',

headers: {

'Content-Type': 'application/json',

Authorization: `Bearer ${jwt}`,

},

body: JSON.stringify(postBody),

});

if (!response.ok) {

const text = await response.text();

console.error('❌ ExpertAnswer failed:', response.status, text);

return;

}

alert('✅ Результат тесту успішно збережено');

} catch (err) {

console.error('❌ Помилка відправки ExpertAnswer:', err);

}

return;

}

// Якщо користувач не авторизований → рахуємо топ-3 спеціальності по вектору

try {

const res = await fetch(

`${API\_URL}/api/categories?fields=name,vector,shortDescription,fullDescription&pagination[pageSize]=100`

);

if (!res.ok) {

console.error('❌ CATEGORIES fetch failed:', res.status, await res.text());

return;

}

const catsRaw = await res.json();

const cats: Category[] = Array.isArray(catsRaw.data)

? catsRaw.data.map((c: any) => {

const attrs = c.attributes ?? c;

// Strapi інколи повертає fullDescription із рядками "\n\n" → замінимо на справжні переводи рядків

const rawFull = attrs.fullDescription || '';

const convertedFull = rawFull.replace(/\\n\\n/g, '\n\n');

return {

id: c.id,

name: attrs.name,

vector: attrs.vector ?? [],

shortDescription: attrs.shortDescription ?? '',

fullDescription: convertedFull,

};

})

: [];

// Функція для косинусної міри схожості

const cosine = (a: number[], b: number[]) => {

let dot = 0,

na = 0,

nb = 0;

const len = Math.min(a.length, b.length);

for (let i = 0; i < len; i++) {

dot += a[i] \* b[i];

na += a[i] \*\* 2;

nb += b[i] \*\* 2;

}

44165850.1507-01 12 01

22

return na && nb ? dot / Math.sqrt(na \* nb) : 0;

};

const scored = cats.map((c) => {

let score = 0;

if (Array.isArray(c.vector) && c.vector.length === ansArray.length) {

score = cosine(ansArray, c.vector);

}

return {

name: c.name,

score,

short: c.shortDescription,

full: c.fullDescription,

};

});

setResults(scored);

} catch (err) {

console.error('❌ Помилка обчислення результатів:', err);

}

};

if (loading) {

return (

<Box className="testpage-loading">

<Typography>Завантаження тесту…</Typography>

</Box>

);

}

if (!loading && questions.length === 0) {

return (

<Box className="testpage-no-questions">

<Typography color="error">

Питань не знайдено. Перевірте CMS та права Public → find/findOne.

</Typography>

</Box>

);

}

return (

<Box className="testpage-container">

<Typography variant="h4" gutterBottom className="testpage-title">

Визначся зі своєю майбутньою спеціальністю

</Typography>

<Box className="testpage-questions">

{questions.map((q, idx) => (

<Box key={q.id} className="testpage-question-block">

<Typography className="testpage-question-text">

{q.order}. {q.text}

</Typography>

<Slider

value={answers[idx]}

min={1}

max={5}

step={1}

marks

onChange={(\_, value) => handleAnswerChange(idx, value as number)}

valueLabelDisplay="auto"

/>

44165850.1507-01 12 01

23

</Box>

))}

</Box>

<Box className="testpage-submit-container">

<Button variant="contained" onClick={handleSubmit} className="testpage-submit-button">

Завершити тест

</Button>

</Box>

{results && (

<Box className="testpage-results-wrapper">

<Paper className="testpage-results-paper" elevation={2}>

<Typography variant="h6" className="testpage-results-title" gutterBottom>

Топ-3 рекомендовані спеціальності

</Typography>

{results

.slice()

.sort((a, b) => b.score - a.score)

.slice(0, 3)

.map((r, index) => (

<Accordion key={r.name} className="testpage-accordion">

<AccordionSummary

expandIcon={<ExpandMoreIcon />}

aria-controls={`res-${r.name}-content`}

id={`res-${r.name}-header`}

>

<Box className="testpage-accordion-header">

<Typography variant="subtitle1" className="testpage-accordion-title">

{`${index + 1}. ${r.name} — ${(r.score \* 100).toFixed(1)}%`}

</Typography>

{r.short && (

<Typography variant="body1" className="testpage-accordion-short">

{r.short}

</Typography>

)}

</Box>

</AccordionSummary>

<AccordionDetails className="testpage-accordion-details">

<ReactMarkdown

remarkPlugins={[remarkGfm]}

components={{

p: ({ node, ...props }) => (

<Typography

component="p"

variant="body2"

className="testpage-full-text"

{...props}

/>

),

li: ({ node, ...props }) => (

<Typography

component="li"

variant="body2"

44165850.1507-01 12 01

24

className="testpage-full-list-item"

{...props}

/>

),

h2: ({ node, ...props }) => (

<Typography

component="h2"

variant="subtitle2"

className="testpage-full-heading"

{...props}

/>

),

}}

>

{r.full}

</ReactMarkdown>

</AccordionDetails>

</Accordion>

))}

</Paper>

</Box>

)}

</Box>

);

};

export default TestPage;

Файл components\University.js

// src/components/University.jsx

import React from 'react';

import { Box, Container, Typography, Grid } from '@mui/material';

import './University.css'; // Правильний шлях до CSS

export default function University() {

return (

<Box component="main" className="university-root">

<Container maxWidth="lg" className="university-container">

{/\* Заголовок \*/}

<Typography variant="h3" className="university-title" gutterBottom>

Про Український державний університет науки і технологій

</Typography>

{/\* Короткий вступ \*/}

<Typography variant="body1" className="university-intro" paragraph>

Український державний університет науки і технологій (УДУНТ) — це сучасний

заклад вищої освіти у місті Дніпро, що поєднує наукові досягнення, потужну

практичну підготовку та міжнародні можливості. Наша мета — формувати конкурентоспроможних

фахівців, які здатні вирішувати захоплюючі виклики сьогодення.

</Typography>

{/\* Секції \*/}

<Grid container spacing={4} className="university-sections">

{/\* 1. Історія \*/}

44165850.1507-01 12 01

25

<Grid item xs={12} sm={6} className="university-section">

<Typography variant="h5" className="university-section-title" gutterBottom>

Історія

</Typography>

<Typography variant="body2" className="university-section-text" paragraph>

УДУНТ засновано у 1964 році як Інститут машинобудування та автоматизації.

За понад півстолітню історію ми перетворилися на багатогалузевий університет

з понад 15 000 студентів і 950 викладачів.

</Typography>

<Typography variant="body2" className="university-section-text">

Сьогодні ми працюємо над інноваційними проєктами разом із провідними

українськими та міжнародними партнерами.

</Typography>

</Grid>

{/\* 2. Факультети та інститути \*/}

<Grid item xs={12} sm={6} className="university-section">

<Typography variant="h5" className="university-section-title" gutterBottom>

Факультети та інститути

</Typography>

<Typography variant="body2" className="university-section-text" paragraph>

У складі УДУНТ:

</Typography>

<ul className="university-faculty-list">

<li>Інженерно-механічний інститут</li>

<li>Інститут інформаційних технологій</li>

<li>Інститут хімії та біоінженерії</li>

<li>Економічний факультет</li>

<li>Юридичний факультет</li>

<li>Факультет міжнародних відносин</li>

</ul>

<Typography variant="body2" className="university-section-text">

Кожен підрозділ має сучасні лабораторії, аудиторії та програми стажувань.

</Typography>

</Grid>

{/\* 3. Наукові досягнення \*/}

<Grid item xs={12} sm={6} className="university-section">

<Typography variant="h5" className="university-section-title" gutterBottom>

Наукові досягнення

</Typography>

<Typography variant="body2" className="university-section-text" paragraph>

Ми займаємо провідні позиції в рейтингах:

</Typography>

<ul className="university-bullet-list">

<li>Топ-5 у Scopus серед ЗВО Дніпра (2023)</li>

<li>Топ-15 серед усіх ЗВО України за ефективністю участі у конкурсі</li>

<li>Понад 2000 публікацій за останні 5 років</li>

<li>Участь у програмах Horizon та Erasmus+</li>

</ul>

<Typography variant="body2" className="university-section-text">

44165850.1507-01 12 01

26

Наші студенти та викладачі регулярно отримують гранти та нагороди.

</Typography>

</Grid>

{/\* 4. Кампус та студентське життя \*/}

<Grid item xs={12} sm={6} className="university-section">

<Typography variant="h5" className="university-section-title" gutterBottom>

Кампус та студентське життя

</Typography>

<Typography variant="body2" className="university-section-text" paragraph>

Кампус УДУНТ включає:

</Typography>

<ul className="university-faculty-list">

<li>21 гуртожиток із 100 % забезпеченням</li>

<li>2 спорткомплекси (4 зали) та 3 басейни</li>

<li>Легкоатлетичний манеж, футбольні поля</li>

<li>Ботанічний сад і геологічний полігон</li>

</ul>

<Typography variant="body2" className="university-section-text">

Студентський клуб, Палац культури, заходи та гуртки підтримують активне життя.

</Typography>

</Grid>

</Grid>

</Container>

</Box>

);

}

# **ДОДАТОК Г**

**Керівництво користувача**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ

Перший проректор Українського

державного університету

науки і технологій

Анатолій РАДКЕВИЧ

ВЕБ-ДОДАТОК ДЛЯ ПРОФОРІЄНТАЦІЇ АБІТУРІЄНТІВ

Керівництво користувача

ЛИСТ ЗАТВЕРДЖЕННЯ

44165850.1507-01 I3 01-ЛЗ

Завідувач кафедри КІТ

Вадим ГОРЯЧКІН

Керівник розробки

Олександр ЖЕВАГО

Виконавець

Артем БЄЛІКОВ

Нормоконтролер

Світлана ВОЛКОВА

2025

ЗАТВЕРДЖЕНО

44165850.1507-01 I3 01-ЛЗ

ВЕБ-ДОДАТОК ДЛЯ ПРОФОРІЄНТАЦІЇ АБІТУРІЄНТІВ

Керівництво користувача

Листів 14

2025

44165850.1507-01 I3 01

2

АНОТАЦІЯ

Документ 44165850.1507-01 12 01 «Веб-додаток для профорієнтації абітурієнтів. Керівництво користувача» входить до складу програмної документації на програму, що реалізує систему профорієнтації для абітурієнтів, яка дозволяє проводити тестування, аналіз результатів та надання рекомендацій щодо вибору професії та освітнього напряму.

У даному документі представлений текст програми. Програма написана на мовах JavaScript, TypeScript, CSS, SQL. Об’єм пам’яті, що займає програма комплексу складає 1,34 ГБ. Конфігурація комп’ютера стандартна. Комплекс функціонує в середовищі MS WINDOWS 11

44165850.1507-01 I3 0

3

ЗМІСТ

1. Введення4

2. Призначення та умови застосування5

1. Підготовка до роботи6
2. Опис операцій7
3. Аварійні ситуації14

44165850.1507-01 I3 0

4

1. ВВЕДЕННЯ

Програмний продукт, розроблений за темою «Веб-додаток для профорієнтації абітурієнтів», може бути застосований абітурієнтами, студентами, адміністрацією навчальних закладів.

Абітурієнти проходять у додатку профорієнтаційний тест, відповіді на який порівнюються з «еталонними» відповідями експертів; на основі цього формується вектор і генеруються рекомендації щодо найбільш відповідних спеціальностей.

Адміністрація університету отримує можливість через Strapi CMS створювати та редагувати питання, керувати експертними відповідями й автоматично оновлювати векторні представлення спеціальностей для точнішого підбору рекомендацій.

Для роботи з програмою від користувача достатньо базових навичок користування персональним комп’ютером (прості операції в ОС Windows чи іншій) та вміння працювати в веб-браузері.

44165850.1507-01 I3 0

5

1. ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

Функціональне призначення — можливість абітурієнтам проходити профорієнтаційний тест, отримувати рекомендації щодо вибору спеціальності на основі обчислення косинусної подібності їхніх відповідей з еталонними профілями експертів; адміністратори можуть формувати, редагувати та впорядковувати тестові питання й категорії, зберігати та обробляти відповіді користувачів; студенти-експерти долучаються до вдосконалення системи, надаючи «еталонні» відповіді для оновлення векторних представлень спеціальностей.

Експлуатаційне призначення — абітурієнти використовують додаток для проходження тесту й ознайомлення з освітніми напрямами; студенти-експерти поповнюють базу «еталонних» даних; адміністратори можуть змінювати питання, спеціальності, «еталонні» дані.

Програма розроблена для використання в операційних системах Windows 10/11, macOS і Linux через сучасні веб-браузери (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari). Для експлуатації на стороні клієнта необхідні мінімальні характеристики:

— процесор: Intel Core i3 / AMD Ryzen 3 або вище

— оперативна пам’ять: 4 ГБ

— графічний адаптер: інтегроване відеоядро з підтримкою OpenGL 3.0 або вище

— дисплей: роздільна здатність від 1920×1080

— підключення до Інтернету: не менше 15 Мбіт/с

— веб-браузер: останні версії Google Chrome, Mozilla Firefox, Edge або Safari

— вхідні пристрої: клавіатура, миша або сенсорний екран.

44165850.1507-01 I3 0

6

1. ПІДГОТОВКА ДО РОБОТИ

Для користування програмним продуктом необхідно:

— мати папку career-guidance з усіма файлами проєкту;

— встановити Node.js (версія ≥ 14) та менеджер пакетів npm або yarn;

— у терміналі перейти в кореневу директорію career-guidance.

Запуск Strapi з директорії cms.

— npm run develop або yarn develop — старт Strapi із автоперезавантаженням (autoReload);

— npm run start або yarn start — старт Strapi без автоперезавантаження;

— npm run build або yarn build — збірка адмін-панелі Strapi.

Після успішного старту відкрити в браузері адресу http://localhost:1337/admin для доступу до панелі адміністратора.

Запуск фронтенду (React) з директорії frontend.

— npm start — запуск додатку в режимі розробки; відкрити http://localhost:3000;

— npm test — інтерактивний запуск тестів;

— npm run build — збірка для продакшен у папку build;

— npm run eject — одноразова операція вивантаження конфігурації Create React App (необоротно).

44165850.1507-01 I3 0

7

1. ОПИС ОПЕРАЦІЙ

Неавторизований користувач — може переглядати головну сторінку та сторінку університету, ознайомлюватися зі спеціальностями, проходити тест.

Авторизований користувач — має всі можливості неавторизованого; додатково може проходити профорієнтаційний тест і надсилати свої «еталонні» відповіді на бекенд.

Адміністратор (через адмін-панель Strapi) — має повний доступ до контенту сайту: створює, редагує та видаляє спеціальності (категорії), тестові питання й «еталонні» відповіді.

При запуску панелі адміністратора можна побачити вікно для авторизації (рис. Г.1).

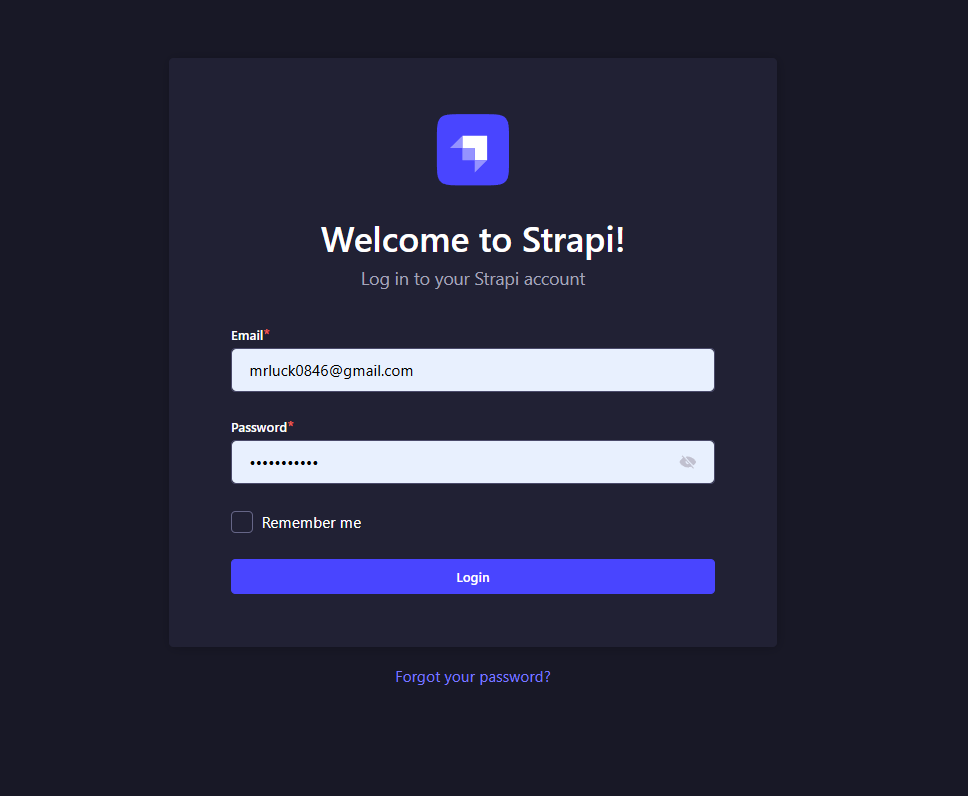


Рис. Г.1. – Вікно авторизації

44165850.1507-01 I3 0

8

При вході у систему можемо побачити головне меню з боковою панеллю, яке містить основні кнопки для керування колекціями: категорій, відповідей, «еталонних» даних, відповідей експертів та користувачів (рис. Г.2)

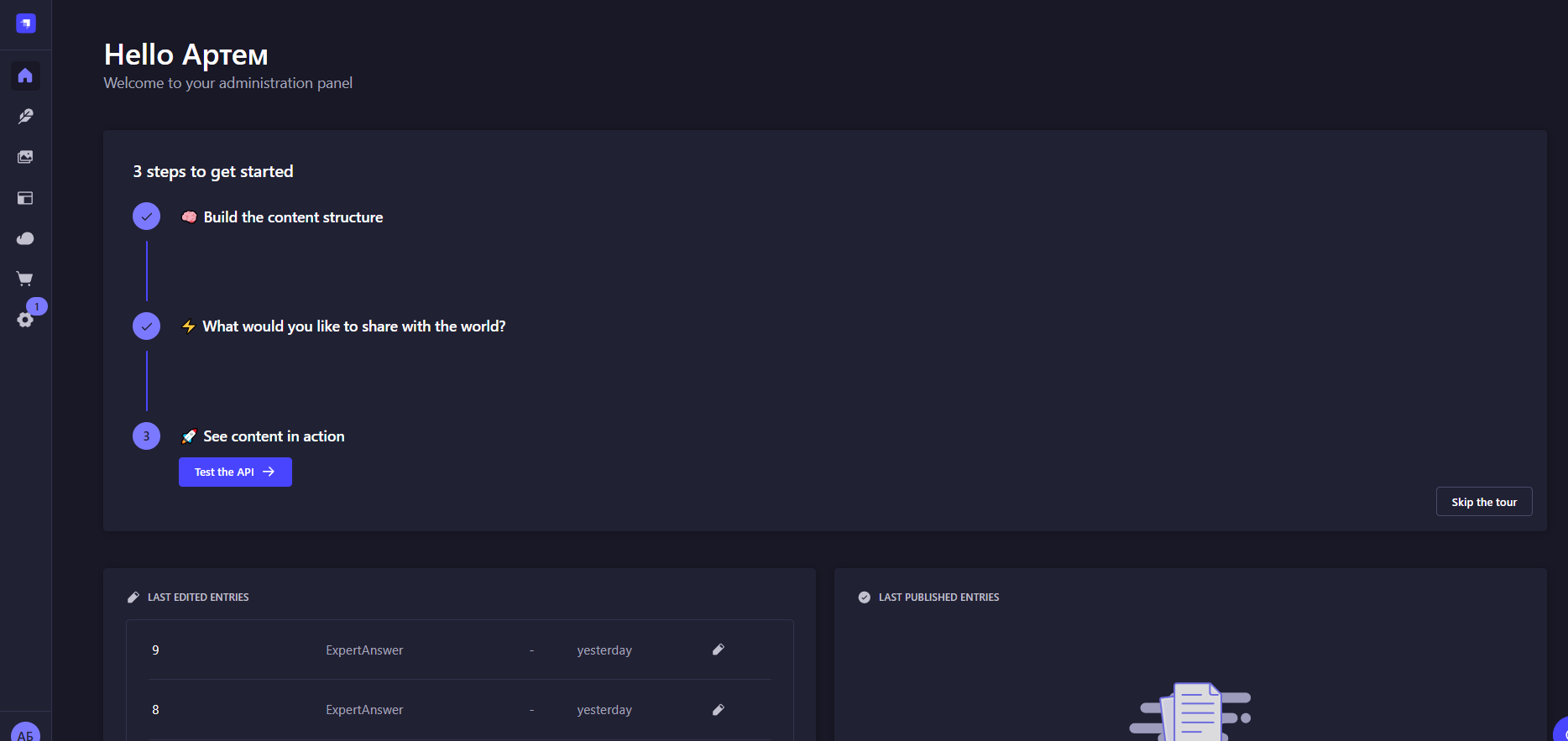


Рис. Г.2. – Головне меню

При натисканні на кнопку Content Manager можна побачити всі колекції та панель керування вмістом. Дизайн інтуїтивно зрозумілий є відповідні кнопки для створення запису «Create new entry», фільтрації пошуку. Щоб відредагувати потрібний запис потрібно натиснути на рядок (рис Г.3.).

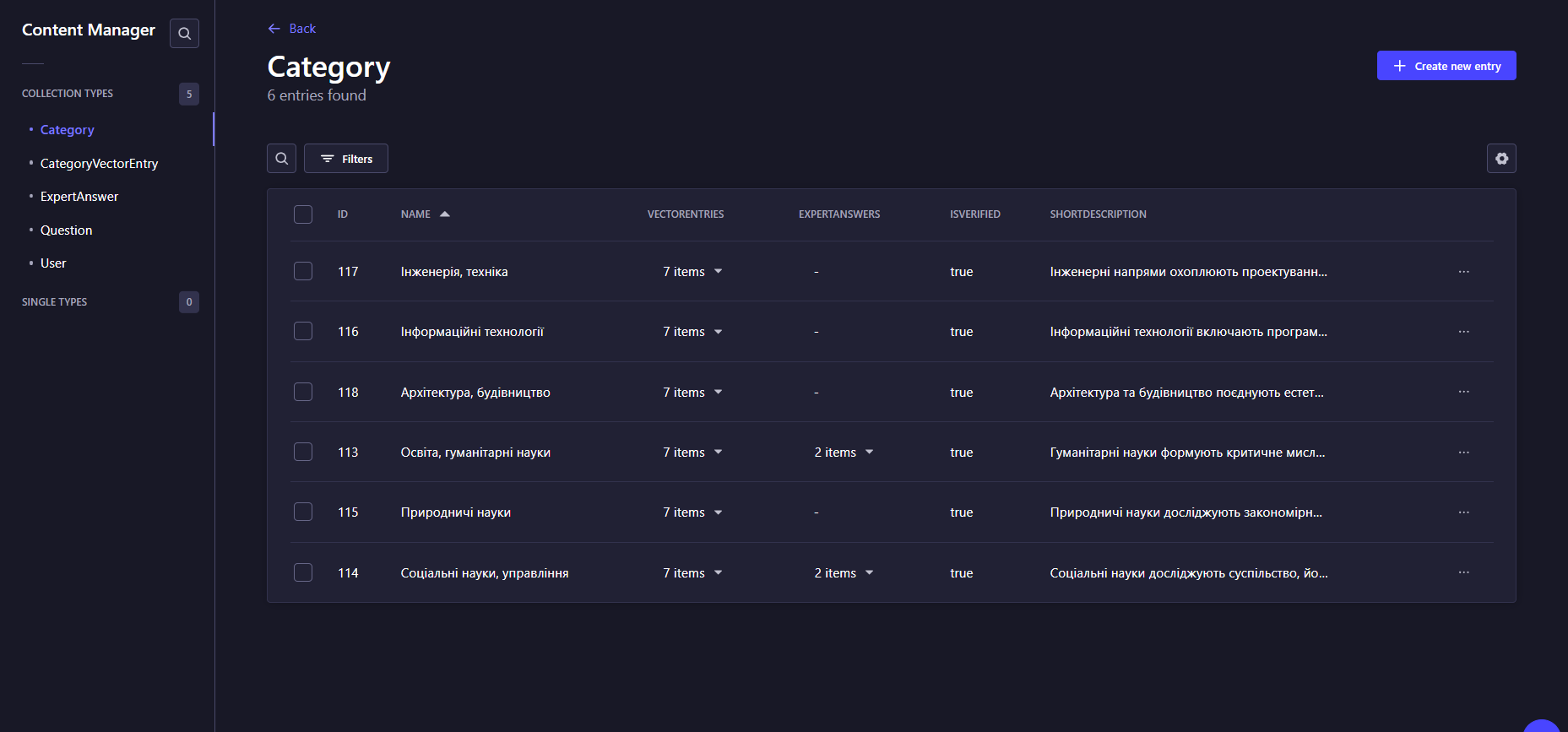


Рис. Г.3. – Content Manager

44165850.1507-01 I3 0

9

Приклад створення нового запису для колекції Category (рис. Г.4.). Вторинні ключі та вектор вказувати не потрібно. Ці поля заповнюються автоматично.

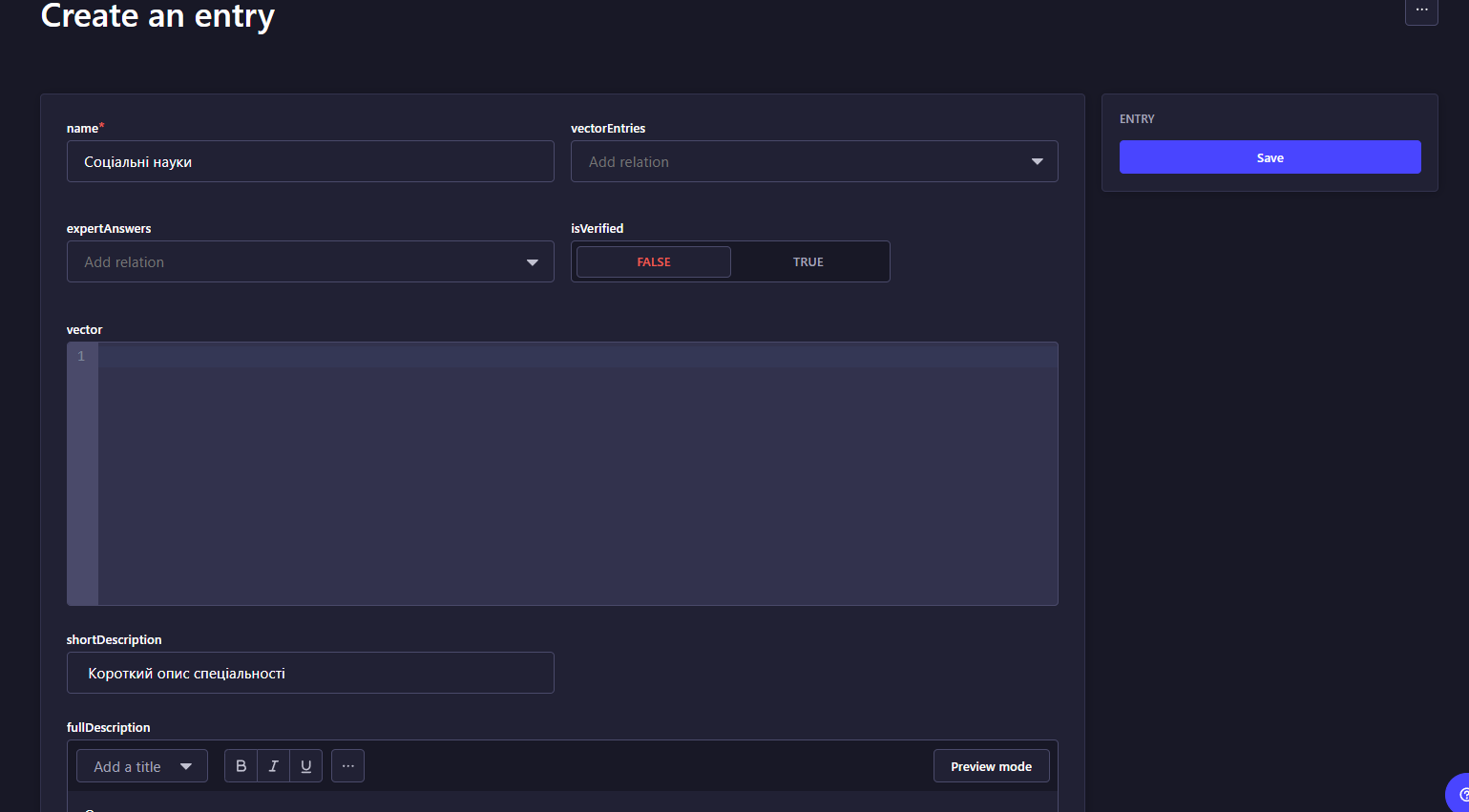


Рис. Г.4. – Створення нового запису

Також на боковій панелі доступні кнопки Content-Type Builder, Settings (рис. Г.5). Кнопка налаштування дозволяє: створювати нові ролі, керувати дозволами та API; кнопка Content-Type Builder дозволяє керувати структурую колекцій: додавати, видаляти поля.

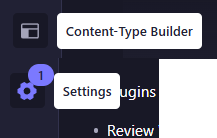


Рис. Г.5. – Кнопки на боковій панелі

44165850.1507-01 I3 0

10

При запуску фронтенду можна побачити головну сторінку сайту (рис. Г.6.).

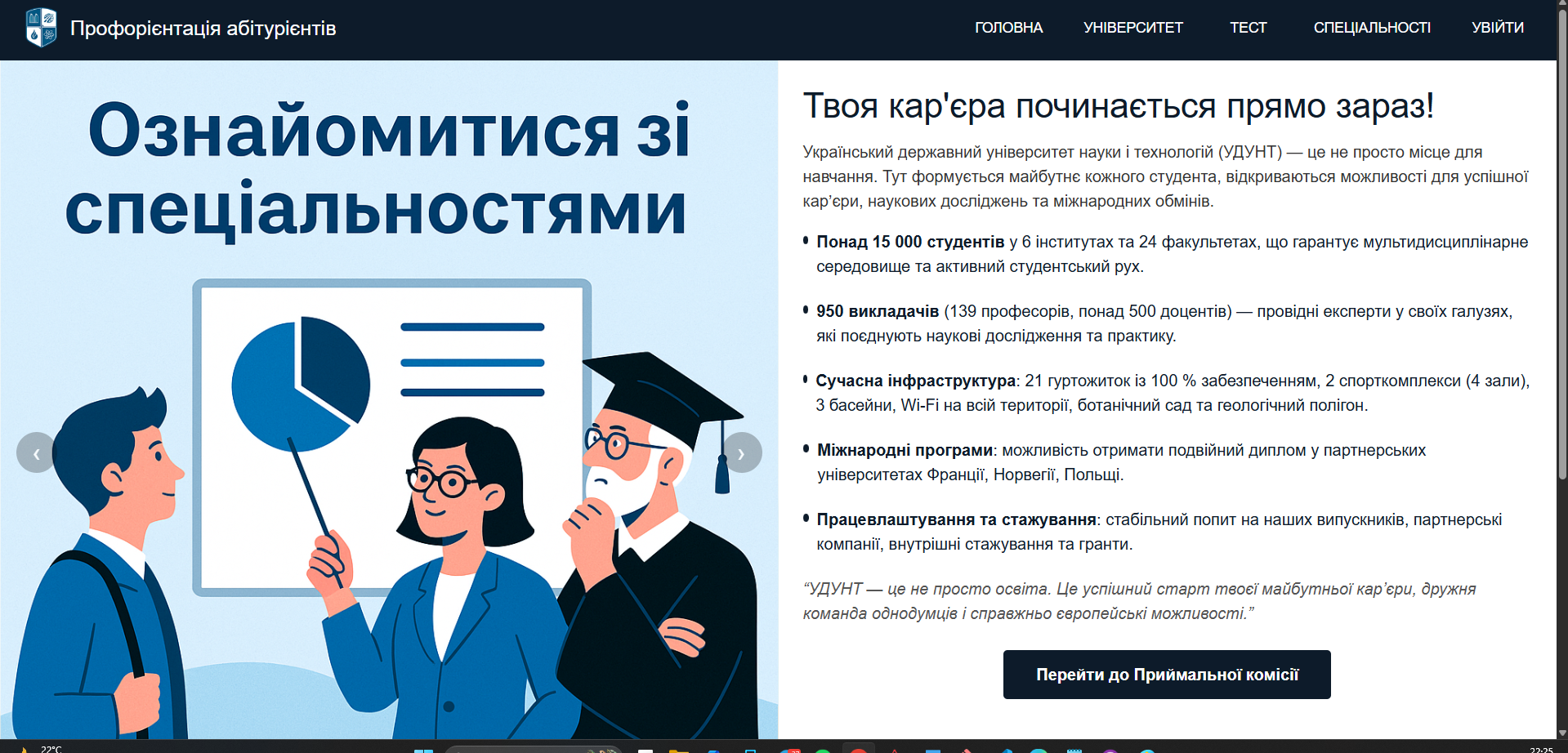


Рис. Г.6. – Головна сторінка

Також продемонстровано роботу основних сторінки сайту (рис. Г.7. ­– Г.12.).

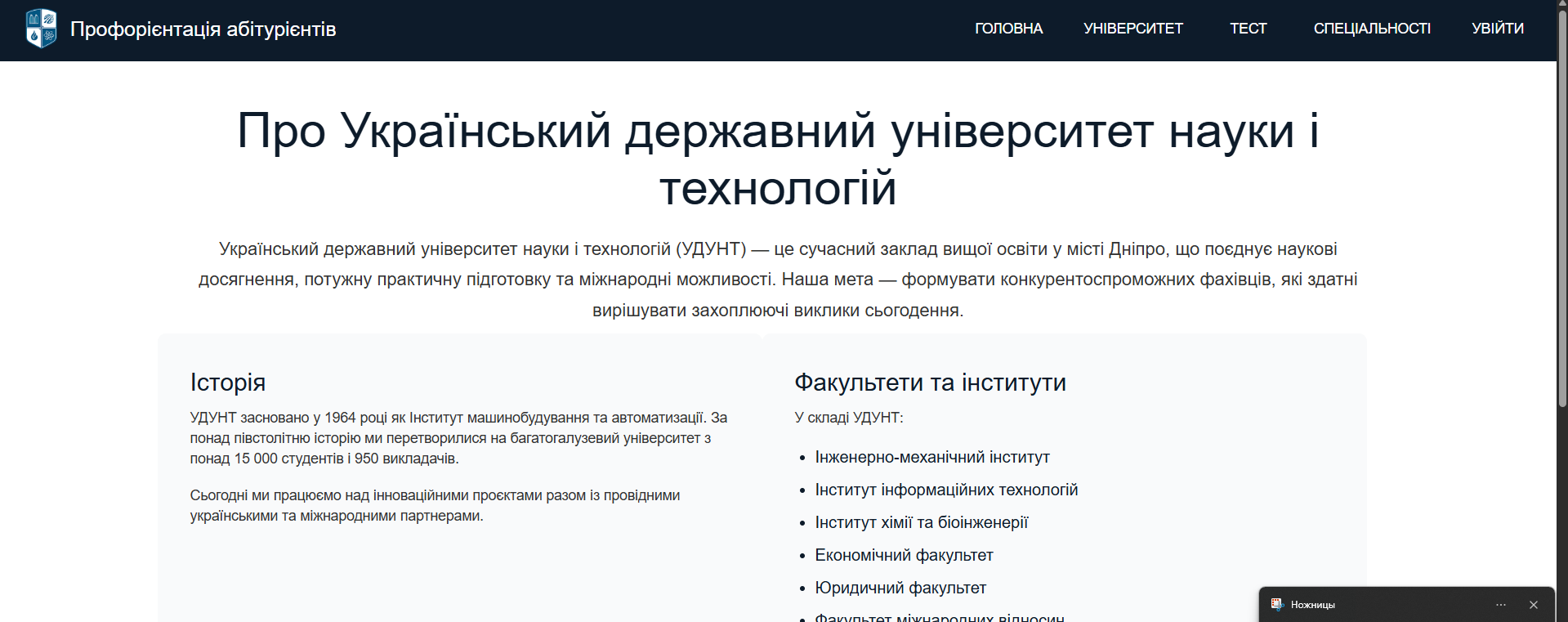


Рис. Г.7. – Сторінка університет

44165850.1507-01 I3 0

11

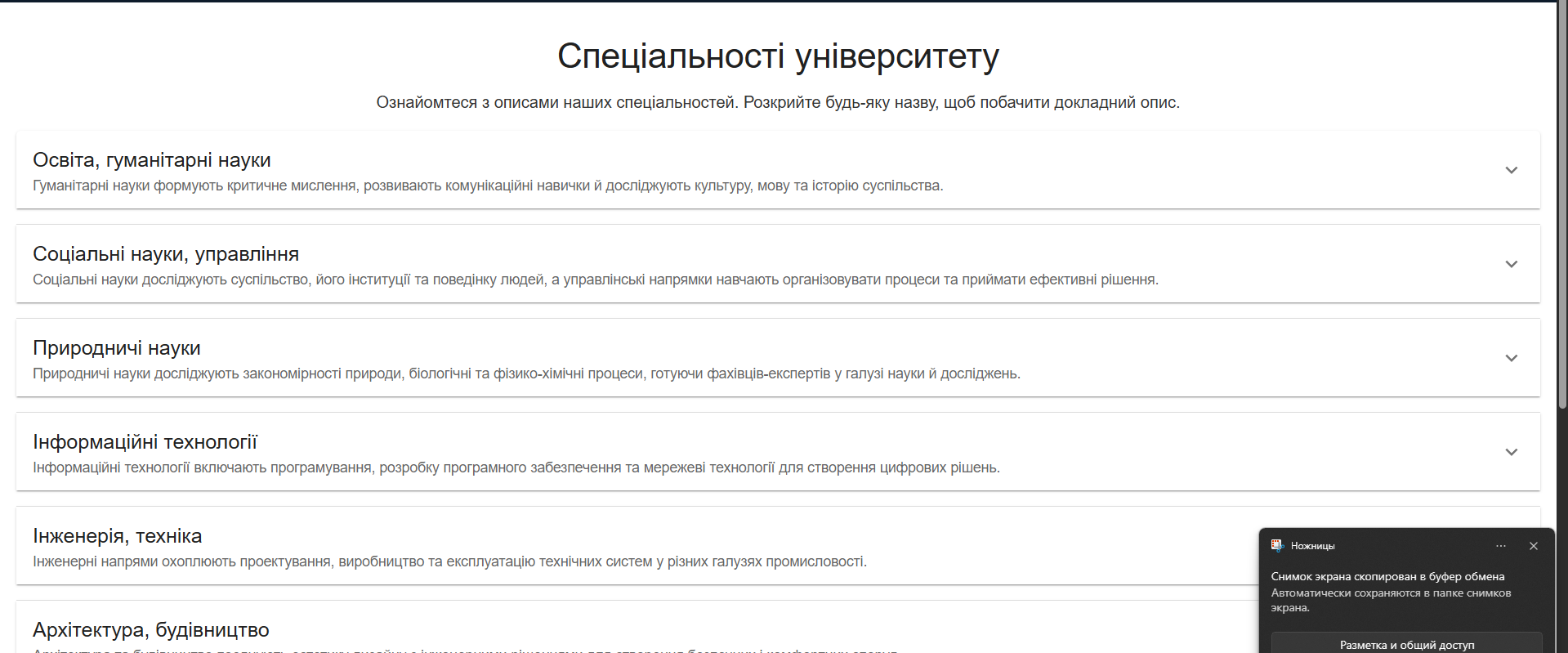


Рис. Г.8. ­– Сторінка спеціальності

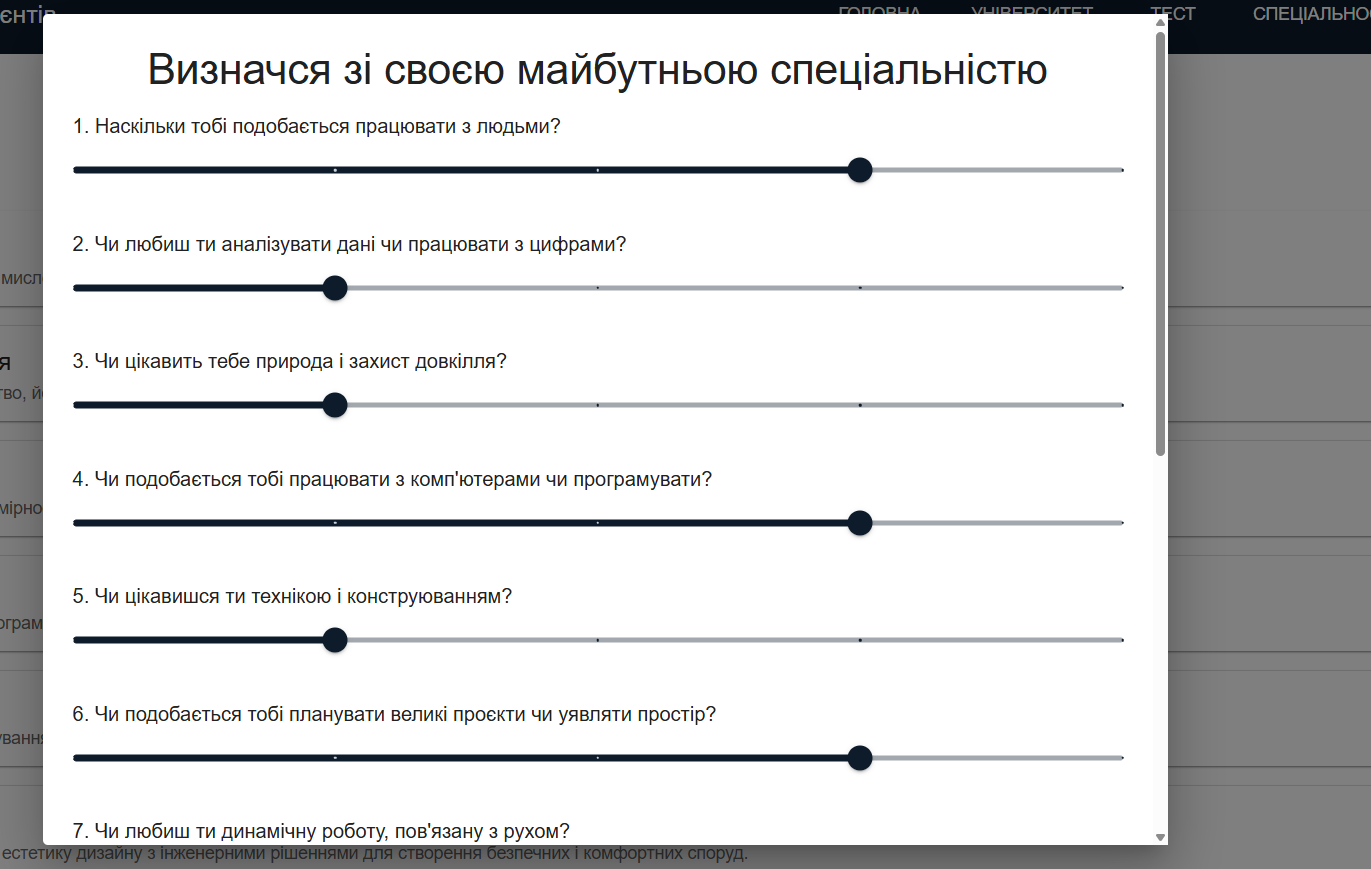


Рис. Г.9. – Сторінка тест

44165850.1507-01 I3 0

12

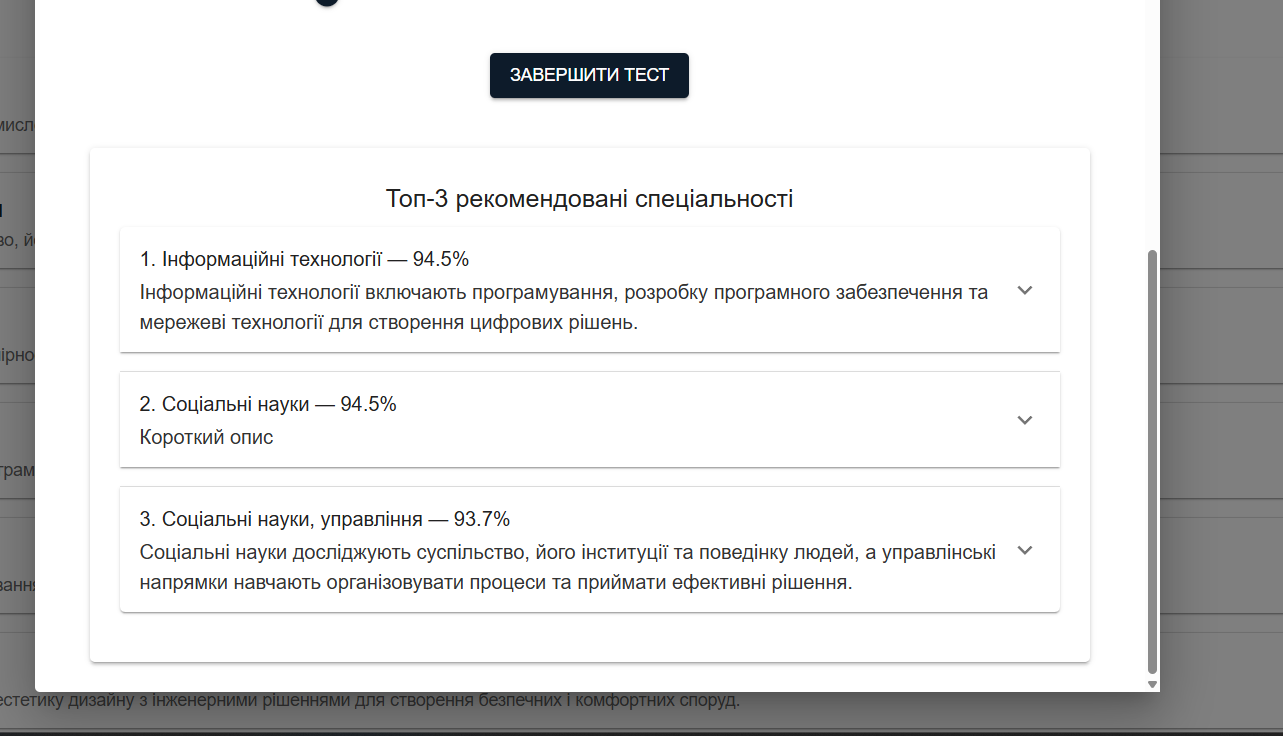


Рис. Г.10. – Результат тестування

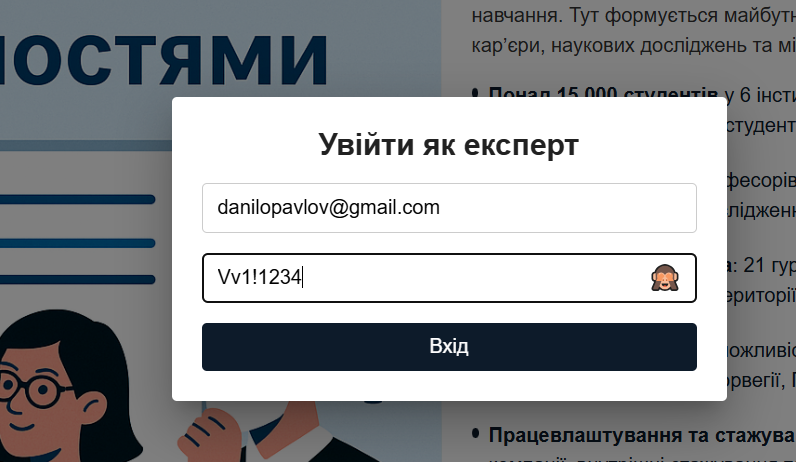


Рис. Г.11. – Вхід як експерт

44165850.1507-01 I3 0

13

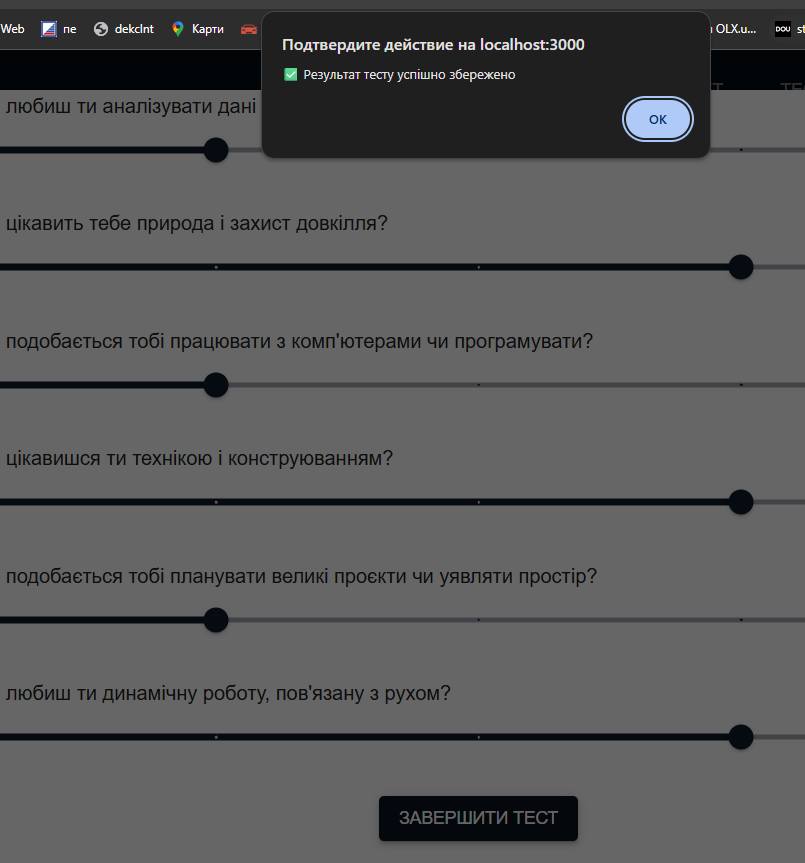


Рис. Г.12. – Відправка відповідей експерта

44165850.1507-01 I3 0

14

1. АВАРІЙНІ СИТУАЦІЇ

Якщо під час роботи програми сталася аварійна ситуація, рекомендується перезавантажити програму для відновлення її працездатності.