НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри
Інформаційних систем та технологій
(назва кафедри)

_____ / **Швиденко М.З** /
(підпис) (ПІБ)

“___” _____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

«Автоматизована інформаційна система формування індивідуальної освітньої траєкторії здобувача вищої освіти»

Спеціальність 126 – «Інформаційні системи та технології»

Гарант освітньої програми

_____ _____ Мокрієв М.В.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

_____ _____ Мокрієв М.В.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Грабарчук Максим Богданович
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Завідувач кафедри інформаційних систем і технологій

_____ _Швиденко М.З.___
підпис ініціали та прізвище

__ _____ 2025 р.

ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

студенту(ці) _____Грабарчуку М.Б._____

Спеціальності 126 “Інформаційні системи та технології”

1. Тема роботи: **«Автоматизована інформаційна система формування індивідуальної освітньої траєкторії здобувача вищої освіти»**

Затверджена наказом ректора від 16.12.2024р. №2245С

2. Термін подання завершеної роботи на кафедру – ___06.2025р.

3. Вихідні дані отримуються з переліку методик оцінки здобувачів за когнітивними здібностями, перелік тестових запитань береться з відкритих джерел аналогічних тестів.

4. Перелік питань, що розглядаються:

1. Теоретико-методологічні засади дослідження систем визначення когнітивних здібностей здобувачів освіти

2. Архітектурно-технологічні аспекти проєктування системи формування індивідуальної освітньої траєкторії

3. Реалізація тестових програмних компонентів з аналізом результатів здобувачів освіти

5. Календарний план

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Вибір теми, планування та підготовка	25.02.2025	Завдання виконано
2	Дослідження та збір матеріалів	30.03.2025	Завдання виконано
3	Аналіз та написання	20.05.2025	Завдання виконано
4	Попередній перегляд та оцінка	27.05.2025	Завдання виконано
5	Захист бакалаврської роботи	06.2025	Завдання виконано

Керівник кваліфікаційної роботи _____ / __Мокрієв М.В., к.е.н.____
підпис ПІБ, вчене звання та ступінь

Завдання прийняв до виконання _____ / _____Грабарчук М.Б._____
підпис

Дата отримання завдання 15.02.2025

РЕФЕРАТ

Тема бакалаврської кваліфікаційної роботи: “Автоматизована інформаційна система формування індивідуальної освітньої траєкторії здобувача вищої освіти”.

Автор роботи: Грабарчук Максим Богданович.

Керівник роботи: Мокрієв М.В.

Пояснювальна записка: 74 с., 20 рис., 2 дод., 33 джерел.

Графічна частина: 15 презентаційних слайдів.

ОСВІТНЯ ТРАЄКТОРІЯ, КАР'ЄРНА ОРІЄНТАЦІЯ, ІНФОРМАЦІЙНА СИСТЕМА, АВТОМАТИЗАЦІЯ, МОДУЛЬ ТЕСТУВАННЯ.

Метою цієї бакалаврської роботи є покращення якості освітнього планування шляхом надання студентам науково обґрунтованих, індивідуалізованих рекомендацій, які сприятимуть їхньому академічному та кар'єрному успіху. Розроблена система пропонує діагностичний інструмент у вигляді тесту з профорієнтації, який оцінює відповіді користувачів та пропонує відповідні напрямки академічного розвитку.

Впроваджене програмне забезпечення надає користувачам інтерактивний модуль тестування, генерує індивідуальні рекомендації та підтримує адміністративні функції для управління вмістом тестів.

Практичне значення роботи полягає у створенні гнучкого інструменту для студентів, освітніх консультантів та установ, що дозволяє краще приймати рішення щодо академічних шляхів. Система підвищує залученість користувачів, пропонуючи індивідуальні поради, та спрощує процес профорієнтації завдяки автоматизації. Розроблене рішення може бути застосоване в навчальних закладах або центрах профорієнтації для покращення послуг підтримки студентів та академічного планування.

АНОТАЦІЯ

У цій бакалаврській роботі представлено проєктування та впровадження автоматизованої інформаційної системи для формування індивідуальної освітньої траєкторії здобувачів вищої освіти.

Система розроблена як веб-застосунок з використанням PHP, фреймворку Symfony та MySQL як бази даних. Її основний функціонал включає тест на професійну орієнтацію на основі психологічних моделей, які допомагають визначити схильності та інтереси студентів. Після завершення тесту система аналізує результати та рекомендує відповідні кар'єрні напрямки та освітні шляхи.

Розроблена система має на меті допомогти студентам у прийнятті обґрунтованих рішень щодо їхнього академічного та професійного розвитку, тим самим підвищуючи залученість та успіх у вищій освіті.

ABSTRACT

This bachelor's thesis presents the design and implementation of an automated information system for the formation of an individual educational trajectory for recipients of higher education.

The system is developed as a web application using PHP, the Symfony framework, and MySQL as the database. Its core functionality includes a career orientation test based on psychological models that help determine students' predispositions and interests. Upon completion of the test, the system analyzes the results and recommends suitable career directions and corresponding educational paths.

The developed system aims to support students in making informed decisions regarding their academic and professional development, thereby increasing engagement and success in higher education.

ЗМІСТ

ВСТУП

У контексті сучасних освітніх реформ та переходу до студентоцентрованого навчання дедалі більше уваги приділяється необхідності забезпечення персоналізованих підходів до освітнього процесу. Традиційна універсальна модель вищої освіти часто не враховує унікальні інтереси, здібності та кар'єрні цілі окремих студентів. Як наслідок, багато студентів стикаються з труднощами у виборі академічного шляху, який відповідає їхнім сильним сторонам та професійним прагненням.

Концепція індивідуальної освітньої траєкторії (ІОТ) виникла як рішення цієї проблеми. ІОТ – це персоналізований план освітньої та розвивальної діяльності, який допомагає студенту досягти конкретних академічних та професійних цілей. Формування такої траєкторії вимагає аналізу багатьох факторів, включаючи інтереси, здібності та кар'єрні вподобання студента. Однак ручна розробка ІОТ є трудомісткою та часто не має об'єктивності.

У цій роботі пропонується розробка автоматизованої інформаційної системи, призначеної для допомоги студентам у формуванні їхньої індивідуальної освітньої траєкторії. Система дозволяє користувачам пройти тест на профорієнтацію, який визначає їхні домінуючі інтереси та психологічні типи на основі встановлених теоретичних моделей (наприклад, моделі Голланда RIASEC). На основі результатів тестування система генерує персоналізовані рекомендації щодо кар'єрних напрямків та відповідних освітніх напрямків.

Актуальність цієї системи здійснюється у вигляді веб-застосунку з використанням PHP, фреймворку Symfony та MySQL. Застосунок включає такі функції, як реєстрація та автентифікація користувачів, управління тестами, аналіз результатів та генерація індивідуальних рекомендацій. Також

включено адміністративну панель для управління змістом тестів та даними про кар'єру.

Більше того, інтеграція методів психологічного тестування, таких як оцінювання професійної орієнтації, в автоматизовані системи дозволяє ефективніше підтримувати професійне самовизначення. У цьому контексті створення автоматизованої інформаційної системи для формування ініціативних навчальних курсів безпосередньо відповідає на потребу в інноваціях у сфері професійної орієнтації та освітнього планування.

Таким чином, розробка є дуже актуальною, оскільки вона поєднує сучасні ІТ-рішення з педагогічною метою покращення прийняття рішень у вищій освіті шляхом персоналізації та автоматизації.

Метою цієї дослідницької та розробницької роботи є покращення якості освітнього планування шляхом надання студентам науково обґрунтованих, індивідуалізованих рекомендацій, які сприятимуть їхньому академічному та кар'єрному успіху.

Об’єктом цього дослідження є освітній процес у вищих навчальних закладах, зокрема механізми, за допомогою яких студенти обирають академічні напрямки та формують свої навчальні шляхи.

Предметом дослідження є автоматизована інформаційна система, призначена для підтримки формування індивідуальних освітніх траєкторій на основі результатів тестування з профорієнтації.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Опис предметної області

Формування індивідуальної освітньої траєкторії – це складний процес, який вимагає ретельного аналізу особистих якостей, інтересів, здібностей та довгострокових кар'єрних цілей студента. У традиційній освітній моделі академічне та кар'єрне керівництво часто надається через загальні консультаційні сесії або стандартні анкети, яким може бракувати глибини, персоналізації та ефективності, необхідних для підтримки прийняття обґрунтованих рішень.

Зі зростанням різноманітності освітніх програм та кар'єрних сфер студенти все частіше потребують інструментів підтримки, які можуть допомогти їм у виборі шляхів, що найкраще відповідають їхнім психологічним профілям та професійним інтересам. Центри профорієнтації та психологи пропонують цінну підтримку, але їхні послуги можуть бути обмеженими в доступності, масштабованості або економічній ефективності, особливо у великих закладах [1].

Нещодавні досягнення в інформаційних технологіях надають можливість автоматизувати та оптимізувати цей процес. Автоматизовані інформаційні системи можуть допомогти аналізувати великі обсяги даних, застосовувати перевірені психологічні моделі (такі як модель Holland RIASEC) та надавати персоналізовані рекомендації в режимі реального часу. Такі системи здатні запропонувати студентам чітке розуміння їхніх кар'єрних схильностей та пропонувати відповідні освітні шляхи на основі результатів їхніх тестів.

Низка існуючих платформ спрямована на надання послуг з професійної орієнтації або планування освіти, такі як CareerExplorer, 123Test, Truity – онлайн-платформи, що пропонують психометричні тести для кар'єри.

Системи управління навчанням (LMS) – використовуються в університетах, але зазвичай зосереджені на проведенні курсів, а не на плануванні траєкторії розвитку;

Студентські портали – включають академічне відстеження, але не мають вбудованого тестування або механізмів консультування з професійної орієнтації [2].

Незважаючи на свою корисність, багато з цих платформ є або занадто узагальненими, не інтегрованими в університетські системи, або вимагають преміум-доступу. Більше того, більшість не надають адаптивних інструментів планування освіти, які динамічно реагують на результати тестування студентів.

На ринку існує явна прогалина в єдиній веб-системі, яка дозволяє студентам:

- проходити тест з професійної орієнтації;
- автоматично отримувати рекомендації щодо кар'єрних та освітніх шляхів;
- відстежувати свій прогрес та оновлювати свою траєкторію розвитку з часом.

Ця прогалина обґрунтовує необхідність розробки автоматизованої інформаційної системи, спрямованої на підтримку студентів у формуванні їхньої індивідуальної освітньої траєкторії. Така система сприятиме не лише підвищенню академічної задоволеності та успішності, але й підвищенню ефективності навчальних закладів у наданні послуг з професійної орієнтації.

Сучасна вища освіта переживає значну трансформацію, зі зростаючим акцентом на персоналізації навчального досвіду відповідно до унікальних здібностей, інтересів та прагнень кожного студента. У цьому мінливому ландшафті концепція індивідуальної освітньої траєкторії (IOT) стає дедалі

актуальнішою. ІОТ являє собою індивідуальний шлях академічним шляхом студента, розроблений для тісної узгодженості з його кар'єрними цілями, особистими сильними сторонами та освітніми вподобаннями. Такий підхід зміщує фокус зі стандартизованих навчальних програм на більш гнучкі та адаптивні плани навчання [3].

Формулювання такої траєкторії вимагає глибокого розуміння особистості студента, його академічного потенціалу та професійних схильностей. Традиційно цей процес спирався на консультантів з профорієнтації або педагогічних психологів, але ручні методи часто є неефективними, суб'єктивними та обмеженими в масштабованості. Як наслідок, багатьом студентам важко визначити відповідний академічний шлях, що призводить до відсутності мотивації, низької академічної успішності або навіть відмови від навчання.

Для вирішення цих проблем існує очевидна потреба в інтелектуальних інструментах, здатних аналізувати дані студентів та надавати індивідуальні рекомендації. Одна з найефективніших стратегій передбачає використання оцінювання професійної орієнтації, заснованого на психологічних теоріях. Яскравим прикладом є модель RIASEC Голланда, яка класифікує людей та робоче середовище на шість типів особистості: реалістичний, дослідницький, художній, соціальний, підприємницький та традиційний. Визначаючи домінуючі риси студента, стає можливим запропонувати професійні сфери та академічні напрямки, що відповідають його профілю.

Саме тут автоматизовані інформаційні системи демонструють значну цінність. Ці системи призначені для обробки великих обсягів вхідних даних, оцінки психологічних профілів за допомогою структурованого тестування та створення персоналізованих освітніх та кар'єрних рекомендацій. При впровадженні у вигляді веб-платформи така система стає широкодоступною та зручною для користувачів. Вона дозволяє студентам реєструватися, проходити орієнтаційні оцінювання, переглядати детальні відгуки та отримувати індивідуальні пропозиції, які можуть допомогти їм у виборі

курсів або рішеннях щодо спеціалізації. Крім того, адміністратори можуть легко керувати контентом та оновлювати базу знань системи, щоб відображати змінні академічні пропозиції або потреби ринку праці.

У вищих навчальних закладах інтеграція таких технологій не лише покращує послуги підтримки студентів, але й сприяє більшому академічному успіху та професійній самореалізації. Надаючи студентам точні рекомендації на основі даних, університети можуть краще узгодити освітні програми з реаліями ринку праці, одночасно задовольняючи індивідуальні потреби своїх учнів. Таким чином, розробка автоматизованої системи формування індивідуальної освітньої траєкторії є одночасно своєчасною та необхідною для підвищення якості та актуальності вищої освіти.

1.2 Огляд існуючих рішень

В останні роки було розроблено дедалі більшу кількість цифрових платформ та програмних інструментів для підтримки студентів у прийнятті академічних та кар'єрних рішень. Ці рішення значно різняться за призначенням, складністю та цільовою аудиторією. Однак не всі вони повністю враховують потребу в інтегрованому та автоматизованому підході до формування індивідуальної освітньої траєкторії, особливо в контексті вищих навчальних закладів.

123Test – це онлайн-платформа, яка отримала світове визнання завдяки наданню доступних, науково обґрунтованих інструментів для самооцінки в галузі психології та профорієнтації. Заснована в Нідерландах у 2003 році, вона перетворилася на широко використовуваний ресурс, який щорічно обслуговує мільйони користувачів у різних країнах. Її основна місія – допомогти людям краще зрозуміти свої особистісні характеристики, когнітивні здібності та кар'єрні вподобання за допомогою науково обґрунтованого тестування (Рис. 1.1) [4].

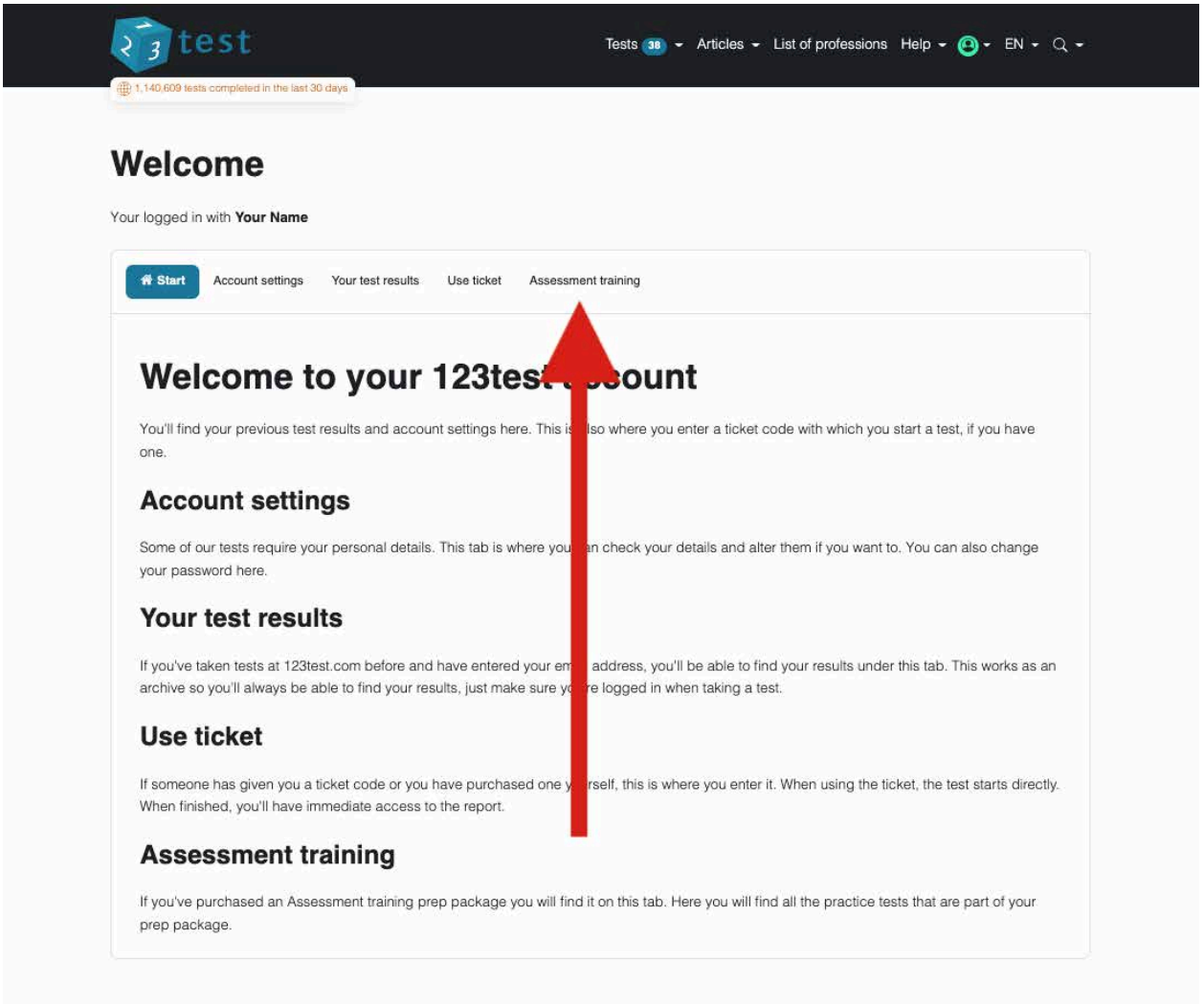

Рис. 1.1 Інформаційна система “123Test”

Серед найпопулярніших функцій – безкоштовний тест на професійну придатність, розроблений на основі моделі Holland Codes або RIASEC. Цей підхід класифікує людей на шість типів особистості – реалістичний, дослідницький, художній, соціальний, підприємницький та традиційний – на основі їхніх інтересів та вподобань. Тест представляє візуальні сценарії, а не текстові запитання, що робить його особливо зручним та ефективним для тих, хто навчається за допомогою візуальних навичок. Це не лише підвищує залученість, але й забезпечує чіткіше розуміння потенційних кар'єрних шляхів користувачів.

Окрім профорієнтації, 123Test містить велику бібліотеку з понад 140 оцінок, які охоплюють різні сфери, включаючи IQ, особистість, цінності та м'які навички. Ці тести розроблені з використанням перевірених

психометричних методологій та постійно вдосконалюються за допомогою емпіричних даних. Команда фахівців зі штучного інтелекту та психології гарантує, що кожен інструмент є одночасно точним та практичним [4].

Важливою перевагою 123Test є його тверда відданість конфіденційності користувачів. На відміну від багатьох платформ, він вимагає мінімальної особистої інформації та уникає зберігання конфіденційних даних. Зібрані результати використовуються анонімно для покращення алгоритмів тестування, зміцнюючи як етичні стандарти, так і довіру користувачів.

Платформа також добре підходить для інституційного використання. Вона надає варіанти інтеграції через API та комплекти розробки програмного забезпечення, що дозволяє школам, університетам або компаніям вбудовувати модулі тестування у свої системи. Ці інтеграції можна налаштовувати, вони підтримують потреби брендингу та безпечну автентифікацію користувачів.

Хоча багато тестів доступні безкоштовно, деякі розширені функції вимагають оплати. Тим не менш, безкоштовних версій достатньо для користувачів, які шукають початкового керівництва або самопізнання. 123Test не пропонує послуг консультування чи коучингу в реальному часі, але залишається ефективним інструментом для незалежного дослідження кар'єри та самооцінки.

Загалом, 123Test усуває розрив між науковою психологією та повсякденним прийняттям рішень. Поєднуючи доступні технології з академічною ретельністю, він допомагає користувачам робити більш обґрунтований освітній та професійний вибір.

Розглянемо інше програмне рішення на ринку.

Truity — це платформа для оцінки особистості, що базується в Каліфорнії, створена для того, щоб пропонувати широкій аудиторії надійні та науково обґрунтовані інструменти самооцінки. Заснована у 2012 році Моллі Оуенс, психологом-консультантом, Truity була створена для того, щоб заповнити прогалину між дорогими тестами особистості корпоративного

рівня та часто ненадійними безкоштовними тестами, які можна знайти в Інтернеті. Протягом багатьох років вона обслуговувала понад 60 мільйонів користувачів по всьому світу, допомагаючи людям досліджувати свої риси особистості, кар'єрні схильності та міжособистісні стосунки науково обґрунтованим способом (Рис. 1.2) [5].

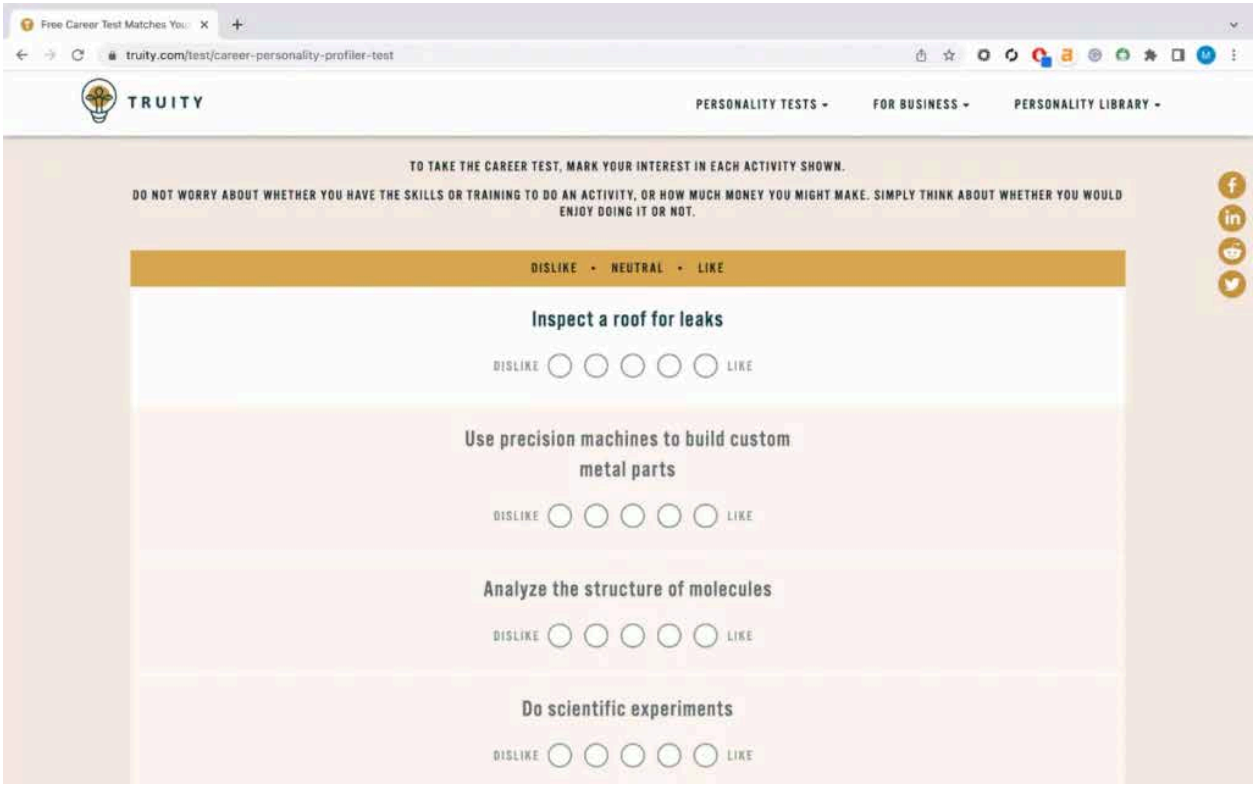

Рис. 1.2 Інформаційна система “Truity”

Асортимент оцінок платформи задовольняє різноманітні потреби, використовуючи добре відомі психологічні моделі. Один з її флагманських інструментів, тест особистості TypeFinder®, базується на теорії Ізабель Бріггс Майєрс про 16 типів особистості, надаючи детальне розуміння 23 різних аспектів особистості. Така глибина дозволяє користувачам отримати тонке розуміння своїх моделей поведінки та уподобань. Доповнює його тест особистості Еннеаграма, який класифікує людей на дев'ять типів на основі основних мотивацій та страхів, широко визнаних за сприяння особистісному зростанню та підвищення соціальної свідомості [5].

Крім того, Truity пропонує тест особистості «Велику п’ять», який оцінює користувачів за п’ятьма фундаментальними рисами – відкритістю,

сумлінністю, екстраверсією, доброзичливістю та невротизмом – надаючи широке уявлення про структуру особистості. Оцінка особистості DISC зосереджена саме на динаміці робочого місця, сортуючи людей за чотирма типами, що підкреслюють їхні поведінкові тенденції в професійному середовищі. Для тих, хто шукає поради щодо кар'єрних шляхів, Career Personality Profiler™ поєднує дані про особистість з професійними інтересами, щоб рекомендувати відповідні професії, що відповідають сильним сторонам та вподобанням людини.

Розроблений з урахуванням зручності користувача, Truity дозволяє завершити більшість оцінок протягом 10-15 хвилин. Хоча базові звіти доступні безкоштовно, більш повні аналізи можна отримати через платні оновлення, які зазвичай коштують близько 29 доларів. Цей баланс пропонує гнучкість, що робить його практичним для звичайних користувачів та тих, хто шукає глибокого зворотного зв'язку.

Відданість Truity науковій точності очевидна в ретельній розробці та валідації її тестів. Компанія дотримується надійних дослідницьких протоколів, щоб гарантувати, що кожна оцінка є одночасно надійною та змістовною. Це включає чітке визначення конструктів особистості, ретельно розроблені тестові завдання та ретельну статистичну оцінку, що сприяє достовірності та корисності наданих результатів.

Окрім комерційних пропозицій, Truity бере участь у соціальних ініціативах через свою некомерційну програму, яка надає послуги оцінювання зі знижкою вразливим групам, таким як діти-сироти, ув'язнені та ветерани. Це відображає ширшу місію платформи – зробити психологічні знання доступними для тих, хто може отримати від них найбільшу користь [5].

Підсумовуючи, Truity є провідним постачальником доступних та науково обґрунтованих оцінок особистості. Її різноманітний набір тестів дозволяє користувачам краще розуміти себе та приймати обґрунтовані рішення в таких сферах, як особистісний розвиток, планування кар'єри та міжособистісні стосунки.

1.3 Постановка завдання

У контексті автоматизованої інформаційної системи для формування індивідуальної освітньої траєкторії студентів вищих навчальних закладів, налаштування передбачає створення системи, яка збирає та обробляє ключові дані для надання персоналізованих рекомендацій кожному користувачеві. Наприклад, система може бути розроблена аналогічно до трекера здоров'я, який відстежує різні особисті показники, такі як споживання калорій, споживання води, фізична активність та інші відповідні показники. Ось приклад того, як можна визначити налаштування такої системи:

Завідувач навчального відділу відповідає за контроль відстеження різних показників навчального прогресу кожного студента. До них належать такі показники, як виконані курсові роботи, участь у позакласних заходах, результати тестів та інші ключові показники ефективності (KPI). Для досягнення цієї мети програма повинна включати комплексну базу даних, яка зберігає та обробляє всі необхідні дані про студентів. Ця база даних допомагатиме системі надсилати персоналізовані сповіщення, попереджаючи студентів про їхній прогрес або будь-які розбіжності в їхній академічній траєкторії.

Програмне забезпечення призначене для надання користувачам таких можливостей:

- моніторинг ефективності різних навчальних заходів;
- відстеження послідовності та правильності навчальних звичок (наприклад, годин навчання, часу зосередження уваги);
- фіксація прогресу студента з плином часу, включаючи виконані завдання, проєкти та іспити;
- визначення області, де студенти перевищують або не досягають необхідних стандартів успішності.

Програма працює в діалоговому режимі, що дозволяє користувачам взаємодіяти з системою через інтуїтивно зрозуміле меню. Інтерфейс

дозволить користувачам (наприклад, студентам, викладачам або академічним радникам) переглядати інформацію про академічну успішність у режимі реального часу та відстежувати свої освітні цілі. Крім того, користувачі зможуть змінювати або скасовувати раніше зареєстровані дані, такі як зміни в навчальному плані або запис на курс.

Ця система забезпечує основу, де система не лише зберігає та обробляє освітні дані, але й діє як інтерактивний інструмент для керівництва студентами в оптимізації їхніх освітніх шляхів на основі показників успішності в режимі реального часу.

1.4 Функціональні та нефункціональні вимоги

Функціональні вимоги:

1. забезпечити реєстрацію, автентифікацію та контроль доступу на основі ролей для студентів, академічних радників та адміністраторів;
2. дозволити введення та зберігання важливих освітніх даних, включаючи завершення курсів, результати тестів, позакласну діяльність та вподобання студентів;
3. сприяти проведенню тестів з професійної орієнтації та оцінювання академічних навичок, щоб допомогти визначити найкращі індивідуальні шляхи розвитку;
4. автоматично генерувати та оновлювати персоналізовані освітні плани на основі результатів оцінювання, успішності студентів та академічних норм;
5. відстежувати та відображати успішність студентів відповідно до запланованої траєкторії, виділяючи завершені модулі, майбутні вимоги та потенційні прогалини;

6. надсилати персоналізовані нагадування та сповіщення про терміни, зарахування на курси, результати оцінювання та рекомендовані дії;
7. надавати детальні звіти та візуалізацію академічного прогресу та ефективності траєкторії як для студентів, так і для консультантів;
8. дозволити авторизованим користувачам оновлювати або виправляти введені дані та коригувати освітні траєкторії за потреби;
9. працювати через інтуїтивно зрозумілий діалоговий інтерфейс з меню, що забезпечує легку навігацію та введення даних для всіх ролей користувачів.

Нефункціональні вимоги:

1. інтерфейс має бути зручним для користувача та доступним для студентів та співробітників з різним рівнем технічної підготовки;
2. система повинна обробляти вхідні дані та швидко генерувати персоналізовані траєкторії, забезпечуючи мінімальний час відгуку;
3. вона повинна підтримувати зростаючу кількість користувачів та зростаючі обсяги даних без погіршення продуктивності [6];
4. дані користувачів, особливо особиста та академічна інформація, повинні бути захищені за допомогою безпечної автентифікації, авторизації та шифрування даних;
5. система повинна бути стабільною, забезпечуючи цілісність та доступність даних з мінімальним часом простою;
6. архітектура програмного забезпечення повинна забезпечувати легке оновлення, налагодження та додавання нових функцій.

1.5 Вимоги до інтерфейсу користувача

Інтерфейс користувача автоматизованої інформаційної системи повинен пріоритезувати простоту та зрозумілість, щоб задовольнити потреби користувачів з різним технічним досвідом. Він повинен пропонувати опції та дані, адаптовані до ролі користувача — студента, академічного радника чи адміністратора — гарантуючи, що кожна особа матиме доступ лише до відповідних функцій та інформації, необхідних для виконання своїх завдань. Дизайн має бути адаптивним, безперешкодно адаптуватися до різних пристроїв, таких як настільні комп'ютери, планшети та смартфони, зберігаючи при цьому стабільну зручність використання.

Візуалізація даних відіграє вирішальну роль, тому академічний прогрес, результати оцінювання та персоналізовані освітні траєкторії повинні відображатися за допомогою інтуїтивно зрозумілих елементів, таких як діаграми, індикатори прогресу та добре організовані таблиці. Інтерактивні форми є важливими для введення даних, і вони повинні включати корисні функції, такі як перевірка, підказки та контекстні вказівки, щоб мінімізувати помилки та покращити взаємодію з користувачем.

Сповіщення та оповіщення повинні бути чітко видимими в інтерфейсі, щоб інформувати користувачів про важливі терміни, майбутні оцінювання або оновлення їхніх навчальних планів, тим самим заохочуючи до своєчасних дій. Для забезпечення ефективної навігації системою слід реалізувати опції пошуку та фільтрації, що дозволять користувачам швидко знаходити курси, контент або конкретні записи про прогрес [7].

Доступність є фундаментальним фактором, а інтерфейс розроблений для підтримки користувачів з інвалідністю шляхом дотримання визнаних стандартів, забезпечення сумісності з програмами зчитування з екрана, навігації за допомогою клавіатури та достатньої контрастності. Якщо база користувачів різноманітна за мовами, система повинна пропонувати багатомовні опції для сприяння інклюзивності.

Візуальна узгодженість важлива в усьому застосунку, з єдиними колірними схемами, шрифтами та шаблонами макета, що забезпечують цілісний та професійний вигляд. Коли користувачі стикаються з помилками або вводять недійсні дані, система повинна реагувати чітким, конструктивним зворотним зв'язком, який допоможе їм вирішити проблему без розчарувань.

Загалом, інтерфейс користувача повинен поєднувати простоту використання, адаптивність та продуманий дизайн, щоб створити сприятливе середовище, де користувачі можуть ефективно керувати та контролювати свої індивідуальні освітні траєкторії.

2 МОДЕЛЮВАННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Загальні відомості

Уніфікована мова моделювання (UML) – це універсальна та широко поширена платформа моделювання, яка використовується в розробці програмного забезпечення для візуального представлення структури та поведінки систем. Вона служить спільною мовою як для технічних команд, так і для зацікавлених сторін, сприяючи чіткому та спільному розумінню проєктування системи. Надаючи стандартизовані символи та позначення, UML допомагає подолати розрив у комунікації, який часто існує між розробниками, аналітиками та клієнтами [8].

UML включає різноманітний набір типів діаграм, які можна загалом згрупувати в структурні та поведінкові категорії. Структурні діаграми зосереджені на статичних компонентах системи. Наприклад, діаграми класів визначають ключові сутності в системі, їхні атрибути, методи та зв'язки між ними. Ці діаграми пропонують план архітектури системи, показуючи, як різні частини пов'язані та залежать одна від одної.

З іншого боку, поведінкові діаграми фіксують динамічні аспекти системи, ілюструючи, як компоненти взаємодіють з часом для виконання певних завдань. Діаграми варіантів використання виділяють різних учасників (користувачів чи інші системи) та їхню взаємодію з функціональністю системи, ефективно моделюючи вимоги користувачів. Діаграми послідовностей та діаграми діяльності детальніше описують потік процесів та обмін повідомленнями, що полегшує аналіз операцій системи та виявлення потенційних покращень.

При розробці автоматизованої інформаційної системи, призначеної для створення індивідуалізованих освітніх траєкторій, UML є особливо цінним. Він дозволяє чітко візуалізувати такі об'єкти, як студенти, курси, оцінювання та шляхи навчання, а також взаємодію між цими елементами.

Таке моделювання допомагає розробникам забезпечити відповідність системи потребам користувачів та академічним нормам, забезпечуючи основу для створення надійного та масштабованого застосунку.

Загалом, використання UML сприяє ефективнішому проєктуванню системи, кращій документації та більш плавній співпраці між членами команди. Він підтримує ітеративну розробку, дозволяючи часті оновлення та вдосконалення моделі в міру розвитку проєкту, що зрештою призводить до системи, яка точно відображає як технічні вимоги, так і очікування користувачів.

2.2 Об’єктне та функціональне моделювання

2.2.1 Діаграма прецедентів. Діаграма варіантів використання – це фундаментальний інструмент уніфікованої мови моделювання (UML), який візуально представляє функціональні вимоги до системи з точки зору користувача. Вона зосереджена на ілюструванні взаємодії між користувачами, відомими як акторами, та самою системою, виділяючи, що система повинна робити, не уточнюючи, як це буде зроблено внутрішньо [9].

Діаграма визначає різні типи користувачів, які взаємодіятимуть із системою, такі як студенти, академічні консультанти та адміністратори, та зображує ключові дії або послуги, які система пропонує цим користувачам. Кожен варіант використання представляє певну функціональність або мету, якої актор хоче досягти, наприклад, проходження тесту з професійної орієнтації, створення освітнього плану або моніторинг академічного прогресу.

Відображаючи ці взаємодії, діаграми варіантів використання допомагають уточнити системні вимоги на ранніх етапах процесу розробки, що полегшує зацікавленим сторонам перевірку того, що система відповідатиме їхнім потребам. Вони також надають розробникам загальний

огляд функціональності системи, який спрямовує детальне проєктування та впровадження.

У контексті автоматизованої інформаційної системи для формування індивідуальних освітніх траєкторій діаграми варіантів використання фіксують основні процеси, включаючи реєстрацію, адміністрування тестів, формування траєкторій та відстеження прогресу. Ця чітка візуалізація підтримує ефективну комунікацію між користувачами та розробниками, гарантуючи, що дизайн системи тісно відповідає очікуванням користувачів та освітнім цілям.

Розроблена діаграма прецедентів використання представлена на рис. 2.1.

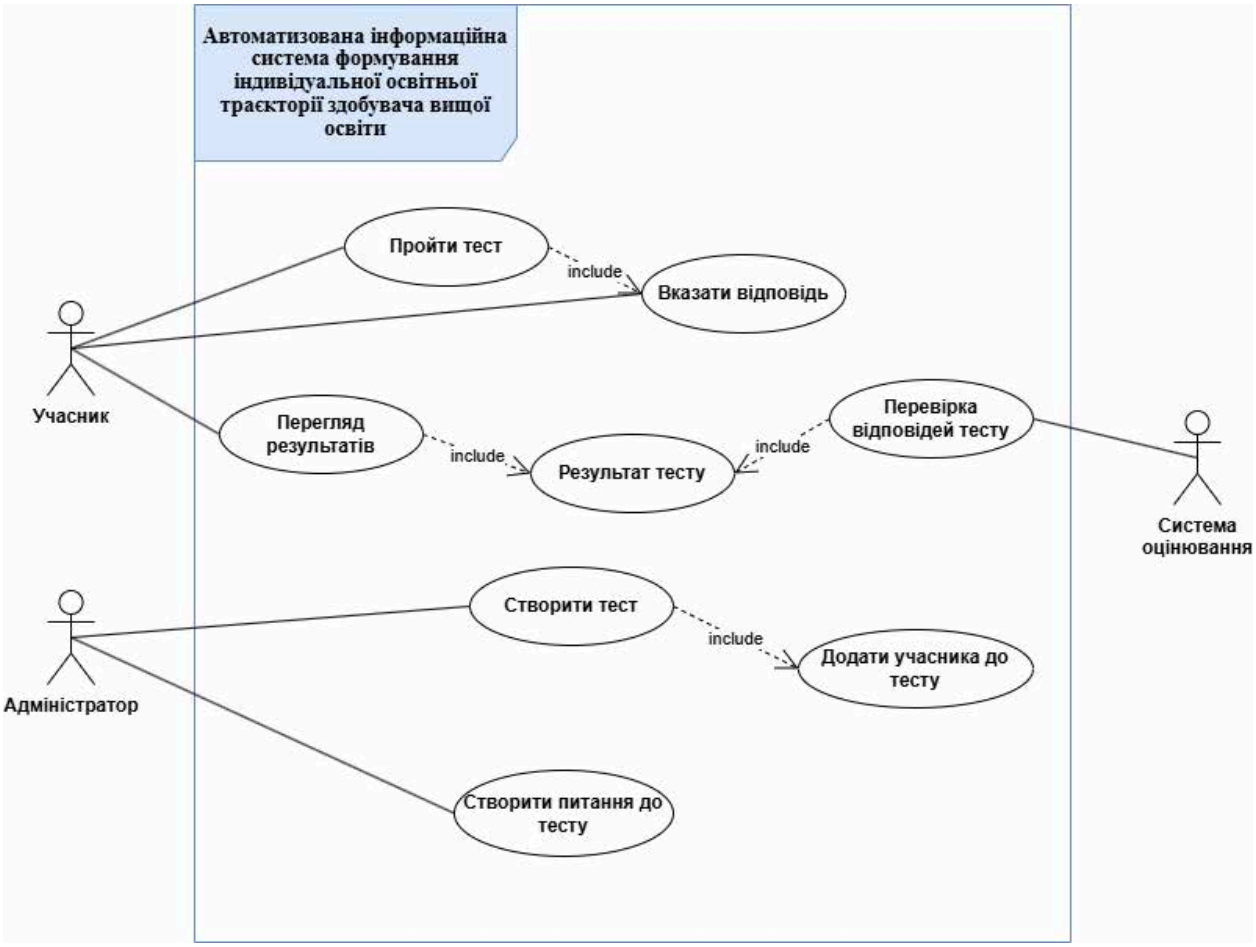

Рис. 2.1 Діаграма прецедентів

Створена діаграма прецедентів містить акторів:

- “Учасник”;

- “Адміністратор”;
- “Система оцінювання”.

Актор «Учасник» включає такі прецеденти:

- пройти тест;
- вказати відповідь;
- перегляд результатів.

Актор «Адміністратор» включає такі прецеденти:

- створити тест;
- додати учасника до тесту;
- створити питання до тесту.

Актор «Система оцінювання» включає такі прецеденти:

- перевірка відповідей до тесту.

Прецеденти певним чином залежать одне від одного.

Розглянемо детальніше вищеописані прецеденти.

Назва прецеденту: Пройти тест

Актор: Учасник

Опис: Цей варіант використання описує процес, за допомогою якого учасник проходить тест на професійну орієнтацію або здібності в автоматизованій інформаційній системі. Мета тесту полягає в оцінці інтересів, сильних сторін та вподобань користувача для підтримки формування персоналізованої освітньої траєкторії.

Основний потік:

1. учасник входить до системи, використовуючи дійсні облікові дані;
2. з панелі інструментів учасник переходить до розділу «Тест на професійну орієнтацію»;
3. система відображає вступ до тесту, пояснюючи його мету та орієнтовну тривалість;
4. учасник натискає кнопку «Розпочати тест», щоб розпочати;

5. система послідовно відображає серію питань, кожне з яких має кілька варіантів відповідей;
6. учасник вибирає відповіді на кожне питання та натискає «Далі», щоб продовжити;
7. після того, як на всі питання буде дано відповіді, учасник надсилає тест;
8. система обробляє відповіді за допомогою заздалегідь визначеного алгоритму;
9. учаснику надаються результати тесту, включаючи рекомендовані професійні галузі та сфери інтересів;
10. результати зберігаються в профілі користувача та можуть бути використані для створення індивідуального навчального плану.

Альтернативні потоки:

1. якщо учасник завершує тест до його завершення, система пропонує йому зберегти прогрес. Якщо учасник підтверджується, система зберігає поточний стан, що дозволяє продовжити тест пізніше;
2. якщо учасник неактивний протягом встановленого періоду, система автоматично виводить його з системи. Після повторного входу йому надається можливість продовжити тест з того місця, де він зупинився;
3. якщо учасник намагається здати тест без відповідей на запитання, система виділяє незаповнені розділи та пропонує користувачеві виконати всі пункти перед здачею.

Випадок використання «Пройти тест» є основною функцією інформаційної системи. Він дозволяє учасникам розмірковувати над своїми особистими схильностями за допомогою структурованого процесу оцінювання. Інформація, отримана під час тестування, служить основою для побудови індивідуальної освітньої траєкторії, що робить цю

функціональність необхідною для персоналізованого академічного планування.

Розглянемо ще один сценарій використання.

Назва прецеденту: Створення тесту

Актор: Адміністратор

Опис: Цей варіант використання описує процес, за допомогою якого адміністратор (наприклад, модератор системи, методист або психолог) створює новий тест в автоматизованій інформаційній системі. Тест призначений для оцінки освітніх уподобань, професійної орієнтації або когнітивних характеристик користувача з метою підтримки формування персоналізованої освітньої траєкторії.

Основний потік:

1. адміністратор входить у систему через безпечний інтерфейс автентифікації;
2. з головної панелі інструментів адміністратор переходить до розділу «Керування тестами» або «Створити тест»;
3. система відображає форму для введення загальної інформації про тест, включаючи назву тесту, опис, цільову категорію (наприклад, професійна орієнтація, здібності, інтереси) та орієнтовний час виконання;
4. адміністратор додає до тесту серію запитань;
5. адміністратор повторює крок 4, доки не будуть створені всі тестові елементи;
6. адміністратор встановлює параметри для оцінювання тесту (наприклад, логіку оцінювання, порогові значення, категорії результатів);
7. система перевіряє введені дані, щоб переконатися, що всі обов'язкові поля заповнені, а структура правильна;
8. адміністратор натискає кнопку «Зберегти» або «Опублікувати тест»;

9. система зберігає тест у базі даних і позначає його як доступний для учасників;
10. відображається повідомлення з підтвердженням, і тест стає доступним для користувачів у розділі «Пройти тест».

Альтернативні потоки:

1. якщо адміністратор намагається зберегти тест з відсутніми обов'язковими полями або невідповідними конфігураціями, система відображає помилки перевірки та запитує на завершення перед продовженням;
2. якщо адміністратор вирішує змінити існуючий тест, а не створити новий, система завантажує тестові дані у форму. Після внесення змін адміністратор може зберегти оновлену версію або повернутися до оригіналу;
3. адміністратор може зберегти тест як чернетку, а не публікувати його негайно. У цьому випадку тест залишається недоступним для користувачів, доки його не буде опубліковано пізніше.

Варіант використання «Створення тесту» – це критично важлива адміністративна функція, яка дозволяє постійно вдосконалювати та розширювати інструменти оцінювання системи. Дозволяючи адміністраторам розробляти та керувати тестами, система зберігає гнучкість та забезпечує відповідність освітнім потребам, що змінюються. Якість та структура цих тестів безпосередньо впливають на точність отриманих освітніх траєкторій.

2.2.2 Діаграма послідовності. Діаграма послідовності — це тип діаграми взаємодії в уніфікованій мові моделювання (UML), яка моделює потік логіки в системі в упорядкованій за часом послідовності. Вона візуально представляє, як об'єкти та компоненти системи взаємодіють один з одним для виконання певного випадку використання або процесу [10].

У контексті автоматизованої інформаційної системи для побудови персоналізованих освітніх траєкторій діаграма послідовності ілюструє динамічну взаємодію між компонентами системи, такими як інтерфейс

користувача, логіка програми, база даних та користувач (наприклад, учасник, адміністратор). Вона підкреслює хронологічний порядок повідомлень, якими обмінюються ці компоненти під час виконання таких завдань, як проходження тесту з професійної орієнтації, створення рекомендацій або перегляд академічного плану.

Наприклад, у сценарії «Пройти тест» діаграма послідовності може починатися з того, що учасник ініціює тест через інтерфейс. Потім система отримує тестові дані з бази даних, представляє запитання, збирає відповіді користувачів, обробляє дані через логічний модуль і, нарешті, відображає результати. Кожна взаємодія представлена як повідомлення зі стрілками, що вказують від відправника до одержувача, вирівняними вертикально відповідно до їхньої послідовності в часі.

Ці діаграми особливо корисні для визначення точного порядку операцій, виявлення залежностей та допомоги розробникам і аналітикам у забезпеченні того, щоб система виконувала дії очікуваним чином. Моделюючи потік повідомлень, діаграма також допомагає у налагодженні логіки, оптимізації продуктивності системи та уточненні функціональності для всіх зацікавлених сторін.

Зрештою, діаграма послідовності доповнює діаграми варіантів використання та класів, зосереджуючись на тому, як відбуваються взаємодії, забезпечуючи важливий план як на етапах проєктування, так і на етапах розробки системи [10].

Діаграма послідовності, зображена на рис. 2.2, включає наступні об'єкти:

- “Учасник”;
- “Адміністратор”;
- “Тест”;
- “Система оцінення”.

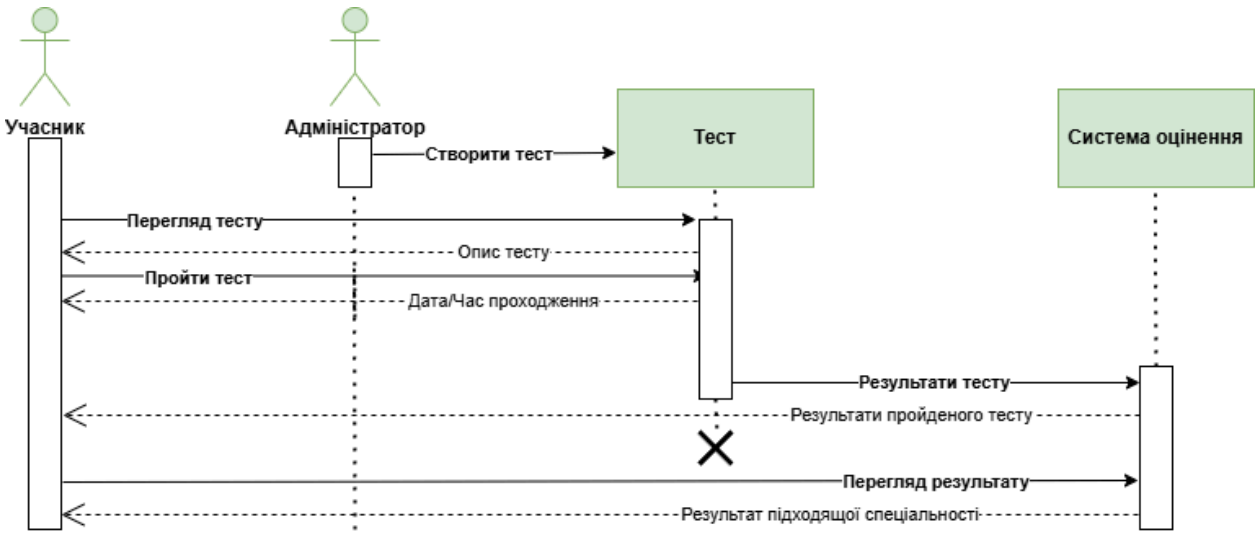

Рис. 2.2 Діаграма послідовності

Ця діаграма послідовності ілюструє процес проходження учасником тесту, створеного адміністратором, та подальшої оцінки його результатів системою оцінювання. Все починається з адміністратора, який відповідає за створення тесту. Щойно тест доступний, учасник переглядає його перед тим, як перейти до його заповнення, надаючи необхідну дату та час подання. Після завершення його відповіді передаються до системи оцінювання для подальшого аналізу. Потім система обробляє дані, оцінює результати учасника та визначає для нього найбільш підходящу спеціалізацію. Після завершення оцінювання учасник отримує доступ до своїх результатів, що дозволяє йому побачити свої результати та отримати рекомендації щодо потенційних кар'єрних шляхів. Ця діаграма ефективно демонструє взаємодію між різними елементами, задіяними в процесі тестування, пропонуючи чітке уявлення про робочий процес системи.

2.2.3 Діаграма активності. Діаграма діяльності — це поведінкова діаграма в уніфікованій мові моделювання (UML), яка представляє потік дій у системі або процесі. Вона візуалізує, як певна операція або варіант використання виконується крок за кроком, підкреслюючи динамічні аспекти системи. Кожна дія представляє завдання або функцію, а потік між ними показує, як користувачі або система переходять від однієї дії до іншої [11].

У контексті автоматизованої системи, призначеної для керівництва здобувачами вищої освіти у формуванні їхніх персоналізованих шляхів навчання, діаграма діяльності відіграє вирішальну роль у відображенні логічної послідовності дій, що виконуються користувачем або системою під час такого процесу, як проходження тесту з професійної орієнтації, генерування рекомендацій або створення профілю користувача.

Наприклад, у дії «Пройти тест» діаграма може починатися зі входу користувача, потім вибір тесту, представлення питання, подання відповіді, розрахунок результату і, нарешті, генерування рекомендації. Потік може включати гілки рішень (наприклад, «Чи завершено тест?»), які визначають наступний шлях процесу.

Діаграми діяльності особливо цінні на ранніх етапах розробки, щоб чітко показати, як розгортається процес, виділити точки взаємодії з користувачем та виявити потенційні винятки або неефективність у логіці системи. Вони допомагають як технічним, так і нетехнічним зацікавленим сторонам зрозуміти поведінку програми.

Загалом, діаграма діяльності забезпечує загальне уявлення про робочі процеси в системі, що спрощує розробку інтуїтивно зрозумілих процесів, управління складністю та забезпечення безперебійної взаємодії з користувачем у вашій системі планування освітньої траєкторії.

Розроблена діаграма активності представлена на рис. 2.3

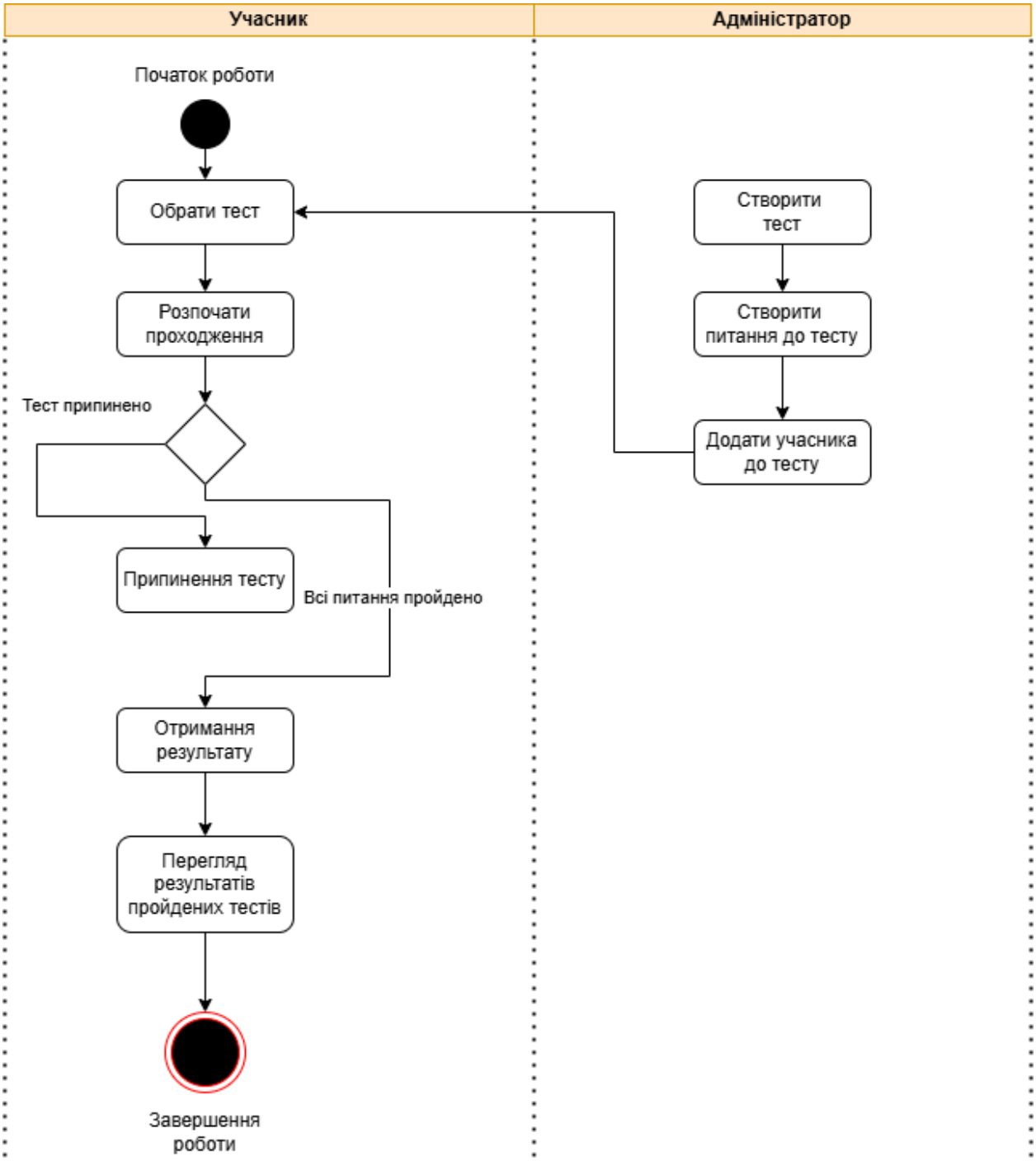

Рис. 2.3 Діаграма активності

Ця діаграма активності ілюструє процес проходження тесту та управління ним з боку адміністратора. Вона розділена на два основні блоки: дії учасника та дії адміністратора.

З боку учасника процес починається з вибору тесту, після чого він приступає до його виконання. У певних умовах тест може бути припинений, і якщо це стається, учасник завершує його достроково. Якщо ж він проходить

усі питання, система формує результати тестування, які учасник може переглянути та ознайомитися зі своїми успіхами.

Адміністратор, у свою чергу, керує тестом на більш глибокому рівні: створює тест, додає до нього питання та реєструє учасників. Це гарантує, що тест проходить у правильному форматі, а учасники отримують повний доступ до нього.

Загалом, діаграма демонструє чіткий алгоритм взаємодії між учасником та адміністратором, допомагаючи візуалізувати процес тестування від початку до кінця.

2.3 Абстракції предметної області

У розробці програмного забезпечення абстракції предметної області допомагають спростити складні реальні системи, визначаючи основні елементи, взаємозв'язки та процеси, що стосуються розв'язуваної проблеми. У контексті автоматизованої інформаційної системи для формування індивідуальної освітньої траєкторії абстракція є вирішальною для моделювання освітнього середовища та його компонентів у структурованій, зрозумілій та реалізованій формі [11].

Предметна область охоплює такі сутності, як студенти (одержувачі освіти), тести, кар'єрні профілі, рекомендації, освітні плани та адміністратори. Кожна з них представлена як логічна абстракція, часто моделюється через класи або об'єкти в системі, фіксуючи лише необхідні характеристики та поведінку для функціональності системи.

Наприклад, абстракція студента може включати такі атрибути, як ім'я, вік, академічна історія, результати тестів та вподобання, тоді як абстракція тесту представляє структуру та логіку оцінювання, що використовуються для визначення сильних сторін студента або його кар'єрної орієнтації. Абстракція профілю кар'єри фіксує категоризовані результати цих оцінювань, які, у свою

чергу, впливають на створення індивідуального освітнього плану — ще однієї ключової абстракції, що містить запропоновані курси, терміни та цілі.

Абстракція також включає ідентифікацію процесів, таких як реєстрація користувача, проведення тесту, аналіз результатів та генерування рекомендацій. Ці робочі процеси спрощуються до системних функцій або модулів, які інкапсулюють основну складність, водночас надаючи користувачам інтуїтивно зрозумілий інтерфейс.

Крім того, такі ролі, як Адміністратор та Учасник, абстрагуються до учасників з певними дозволами та обов'язками. Наприклад, адміністратор може створювати або редагувати тестові питання, керувати базою даних кар'єрних напрямків або переглядати аналітику, тоді як учасники взаємодіють із системою, щоб отримувати персоналізовані академічні рекомендації.

Абстрагуючи предметну область, система може ефективно керувати різноманітними типами даних та взаємодією з користувачами, підтримувати масштабованість та забезпечувати адаптивність до майбутніх змін в освітніх стратегіях чи технологіях.

Підсумовуючи, абстракції слугують концептуальною основою інформаційної системи, забезпечуючи чіткий, організований підхід до перетворення реальних освітніх процесів на цифрове рішення, яке допомагає студентам формувати їхній індивідуальний навчальний шлях.

Рис. 2.4 Абстракції предметної області

Ця діаграма абстракцій предметної області відображає ключові концепції та взаємозв’язки, що формують предметну область тестування. Вона структурована таким чином, щоб показати основні об'єкти та їхню взаємодію.

Основним елементом є тест, який має набір питань, створених адміністратором. Адміністратор відповідає за формування тесту, додаючи необхідні питання та реєструючи учасників, які проходитимуть тест. Учасник отримує доступ до тесту, проходить його та отримує результати після оцінювання. Система оцінювання здійснює аналіз відповідей учасника та визначає, чи задовольняє він критеріям певної спеціальності чи рівня знань.

Крім цього, взаємозв’язок між об’єктами може включати залежності, такі як доступ учасника до тесту через його реєстрацію або автоматичне оцінювання після завершення тесту. Діаграма допомагає зрозуміти логіку тестування та взаємодію між користувачами та системою, що може бути корисним для розробки навчальних платформ чи програмних рішень для тестування.

2.4 Діаграма класів

Діаграма класів відіграє вирішальну роль у моделюванні архітектури об'єктно-орієнтованої системи, представляючи загальний вигляд структури, включаючи її класи, їхні атрибути, поведінку та зв'язки між ними. Під час розробки автоматизованої інформаційної системи для керівництва студентів персоналізованими освітніми шляхами діаграма класів стає структурною основою, яка фіксує ключові елементи предметної області [12].

Ядро системи обертається навколо користувачів, а загальний клас User служить основою для більш специфічних ролей, таких як студенти та адміністратори. Цей батьківський клас зазвичай містить спільні атрибути, такі як ім'я, електронна пошта, роль та пароль, а також загальні методи, такі як управління профілями або доступ до системи. Підклас Student розширює

це додатковими полями, адаптованими до навчальної діяльності, включаючи дані про продуктивність, результати тестів та персоналізовані освітні рекомендації. На противагу цьому, підклас Administrator відповідає за завдання конфігурації та обслуговування системи, такі як створення тестів, управління користувачами та аналіз даних.

Тести утворюють ще одну центральну абстракцію в системі. Кожен тест містить назву, опис та набір пов'язаних питань, що дозволяє програмі оцінювати кар'єрні інтереси та здібності студента. Клас Question (Питання) інкапсулює зміст і структуру кожного окремого питання, включаючи можливі відповіді та, де це можливо, логіку оцінювання.

Після завершення тесту результат записується як об'єкт класу Result (Результат), який пов'язує особу студента з конкретним складеним тестом і включає кінцевий бал і рекомендації. Ці результати допомагають зіставити користувача з попередньо визначеним CareerProfile (Профіль кар'єри), який окреслює потенційні кар'єрні напрямки на основі навичок, рис особистості та уподобань. Кожен кар'єрний профіль пов'язаний з одним або кількома курсами, що входять до складу EducationalPlan (Освітнього плану), що надає студентам структурований, але гнучкий шлях для досягнення їхніх цілей. Сам план містить кураторський вибір об'єктів Course (Курс), кожен з яких представляє предмет з такими атрибутами, як назва курсу, тривалість та академічні кредити.

Зв'язки між цими елементами ілюструють логіку системи. Студенти складають тести, які генерують результати; результати визначають кар'єрні профілі; ці профілі керують створенням освітнього плану; і кожен план містить кілька курсів, що відповідають цілям студента. Ці взаємозв'язки підтримують високо персоналізований навчальний шлях.

Представляючи ці концепції на діаграмі класів, розробники можуть візуалізувати, як взаємодіють різні частини системи, забезпечити модульну структуру та підтримувати ефективне впровадження. Діаграма пропонує не лише знімок статичної структури, але й чіткий посібник з розробки,

розширення та обслуговування системи. Вона інкапсулює освітні процеси в цифровому форматі, роблячи персоналізоване навчання більш доступним та керованим даними.

Для цього проєкту було створено спеціальну діаграму класів з використанням об’єктно-орієнтованого підходу (рис. 2.5).

Рис. 2.5 Діаграма класів

Ця діаграма класів представляє структуру об’єктів у предметній області тестування, відображаючи основні класи та їхні зв’язки.

Ключовими класами є Тест, Учасник, Питання та Система оцінювання. Адміністратор виступає як об’єкт, який створює тест і додає питання. Учасник отримує доступ до тесту, проходить його та взаємодіє з Системою оцінювання, яка аналізує результати та надає рекомендації. Клас Питання містить необхідні атрибути, такі як текст питання та можливі варіанти відповідей.

Діаграма демонструє спадковість, асоціації та агрегування між класами, що допомагає візуалізувати логіку взаємодії між користувачами та тестовою системою. Вона слугує важливим етапом для розробки програмного забезпечення, оскільки дозволяє структурувати дані та визначати їхні функції.

3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Логічна модель даних

Логічна модель даних є критично важливою частиною проєктування системи, надаючи абстрактне представлення того, як організовані дані та як різні сутності пов'язані одна з одною. На відміну від фізичної моделі даних, яка пов'язана з конкретними реалізаціями бази даних, логічна модель зосереджується на семантиці даних, забезпечуючи чіткість структури та зв'язків. Цей рівень моделювання особливо важливий під час розробки персоналізованої системи освіти, оскільки гарантує чисте та нормалізоване представлення всіх ключових концепцій [12].

У цій системі центральною сутністю є Користувач, який включає як студентів, так і адміністраторів. Сутність користувача зберігає фундаментальні атрибути, такі як повне ім'я, електронна пошта, пароль, дата реєстрації та роль. Вона служить основою для взаємодії інших сутностей і посилається на неї в моделі для пов'язування дій або власності.

Для студентів взаємодія починається з сутності Тест, яка представляє професійну орієнтацію або діагностичну оцінку. Кожен тест складається з набору питань, кожне з яких включає текст питання, визначений набір можливих відповідей та метод оцінювання. Сутність Результат фіксує відповіді студента та зберігає результати, такі як результати тестів, зіставлені профілі кар'єри та дані позначок часу.

На основі результатів тестування система зіставляє студента з CareerProfile (Профілем кар'єри), який окреслює характеристики певного кар'єрного шляху, такі як необхідні компетенції, галузь навчання та відповідні курси. Кожен профіль пов'язаний з одним або кількома записами курсів, формуючи освітні варіанти, доступні для цієї конкретної кар'єрної траєкторії.

Ці курси згруповані в сутність EducationalTrajectory (Освітня траєкторія), яка служить персоналізованим навчальним шляхом студента.

Вона будується динамічно, на основі перетину вподобань студента, результатів тестування та доступних освітніх ресурсів. Траєкторія відстежує прогрес, включає статус завершення курсу та дозволяє адаптацію з часом.

Логічна модель даних також включає адміністративні функції. Сутність Адміністратор, хоча структурно подібна до користувача, має повноваження керувати вмістом тесту, перевіряти використання системи та модерувати загальну структуру процесу побудови траєкторії.

Зв'язки між сутностями ретельно розроблені, щоб відображати реальні зв'язки. Між тестами та питаннями існує зв'язок "один до багатьох", тоді як результати підтримують зв'язок "багато до одного" як з користувачами, так і з тестами. Аналогічно, кар'єрні профілі пов'язані з кількома курсами, і кожен студент має рівно одну пов'язану освітню траєкторію в будь-який момент часу, яка складається з кількох окремих курсів. Ця структурована, нормалізована модель підтримує масштабованість, зберігає узгодженість даних та спрощує інтеграцію з користувацькими інтерфейсами та серверними службами. Вона гарантує, що система може обробляти складну освітню логіку без надмірностей, тим самим забезпечуючи більш надійний та персоналізований досвід для кожного здобувача вищої освіти.

Логічна модель системи представлена на рисунку 3.1.

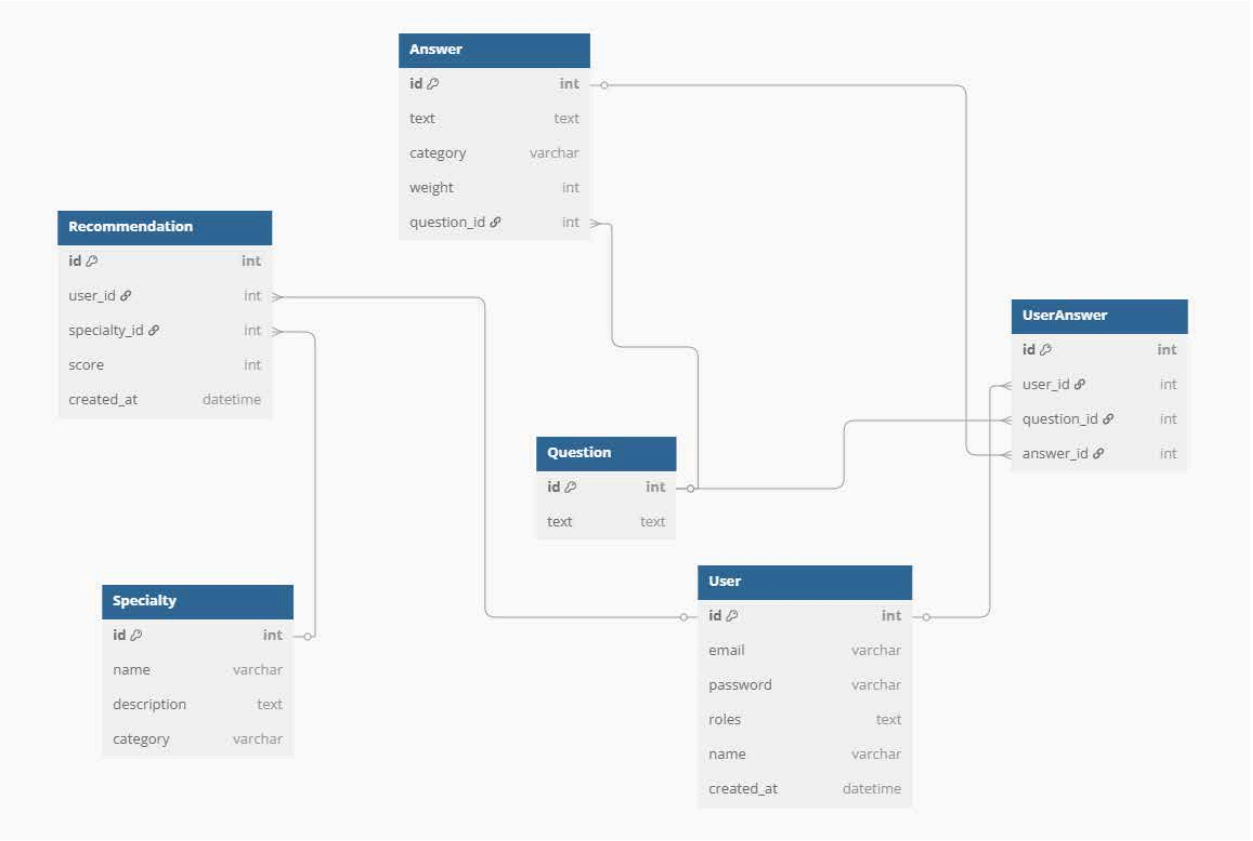

Рис. 3.1 ER-діаграма

User

У цій таблиці зберігаються дані про користувачів системи (учасників тестування).

Поля:

1. id: Первинний ключ, автоінкремент;
2. email: Електронна адреса користувача (використовується для входу/зв'язку);
3. password: Хешований пароль;
4. roles: Ролі користувачів (можуть бути серіалізованим JSON або ролями, розділеними комами, наприклад ROLE_USER, ROLE_ADMIN);
5. name: Повне ім'я користувача;
6. created_at: Мітка часу створення облікового запису.

Зв'язки:

- один-до-багатьох з UserAnswer — один користувач може відповісти на багато запитань;
- один-до-багатьох з Recommendation — один користувач може мати багато рекомендацій.

Question

Зберігає питання, що використовуються в кар'єрному тесті.

Поля:

1. id: Первинний ключ;
2. text: Текст питання.

Зв'язки:

- один-до-багатьох з Answer — одне питання має кілька можливих відповідей;
- один-до-багатьох з UserAnswer — відстежує, на які питання відповів користувач.

Answer

Представляє варіант відповіді на питання.

Поля:

1. id: Первинний ключ;
2. text: Текст відповіді;
3. category: Рядок, що представляє категорію професії (наприклад, бізнес, лікар, творчість тощо);
4. weight: Числова вага, яка впливає на бал у категорії;
5. question_id: Зовнішній ключ до питання.

Зв'язки:

- багато до одного до питання — кожна відповідь належить одному питанню;
- один до багатьох з UserAnswer — відповідь, вибрана користувачем.

UserAnswer

Представляє відповідь, вибрану користувачем для певного питання.

Поля:

1. id: Первинний ключ;
2. user_id: Зовнішній ключ до користувача;
3. question_id: Зовнішній ключ до питання;
4. answer_id: Зовнішній ключ до відповіді.

Зв'язки:

- багато до одного до користувача, питання та відповіді.

Speciality

Представляє професію або кар'єрний шлях.

Поля:

1. id: Первинний ключ;
2. name: Назва спеціальності (наприклад, "Інженер", "Лікар");
3. description: Короткий опис професії;
4. category: Назва категорії (повинна збігатися зі значеннями Answer.category для логічного зв'язку).

Зв'язки:

- один-до-багатьох з Recommendation.

Recommendation

Зберігає розраховані рекомендації для користувачів після завершення тесту.

Поля:

1. id: Первинний ключ;
2. user_id: Зовнішній ключ для User;
3. specialty_id: Зовнішній ключ для Speciality;
4. score: Остаточний розрахований бал у категорії цієї спеціальності;
5. created_at: Мітка часу, коли було створено рекомендацію.

Зв'язки:

- багато-до-одного для User;
- багато-до-одного для Speciality.

Ця діаграма ілюструє систему тестування з професійної орієнтації, розроблену, щоб допомогти користувачам визначити професійну спеціальність, яка найкраще відповідає їхнім здібностям та інтересам, на основі їхніх відповідей на серію запитань. Основні сутності складаються з користувачів, які складають тест, запитань, які пропонують кілька варіантів відповідей, та відповідей, що відповідають різним професійним категоріям. Кожна відповідь має певну вагу, що впливає на бал користувача в пов'язаній з нею категорії.

Під час взаємодії користувачів з тестом вибрані ними відповіді записуються, що дозволяє системі розраховувати сукупні бали за різними категоріями. Ці бали відображають ступінь відповідності користувача певним професійним сферам. Категорії пов'язані зі спеціальностями, що представляють різні кар'єрні шляхи, такі як інженерія, медицина, бізнес або творчі професії. На основі категорії з найвищим балом система генерує персоналізовані рекомендації, пропонуючи спеціальність, яка найбільше відповідає профілю користувача.

Діаграма ефективно фіксує зв'язки між сутностями, наприклад, як запитання пов'язані з кількома відповідями, і як відповіді користувачів пов'язані як із запитаннями, так і з відповідями. Використання проміжної сутності для реєстрації відповідей користувачів дозволяє детально відстежувати окремі взаємодії з тестом. Така структура забезпечує безперебійний перехід від відповідей на запитання до оцінювання та, зрештою, до індивідуальних консультацій з питань кар'єри, організовуючи дані чітким та реляційним чином, що сприяє функціональності системи.

3.2 Вибір системи управління базою даних та її реалізація

Вибір відповідної системи керування базами даних (СКБД) є фундаментальним кроком у розробці автоматизованої інформаційної системи, призначеної для створення індивідуальних освітніх траєкторій для здобувачів

вищої освіти. СКБД повинна бути здатною ефективно обробляти структуровані дані, підтримувати складні запити, забезпечувати цілісність даних та забезпечувати масштабованість у міру зростання системи.

Реляційна система керування базами даних (РСБД) – це тип системи баз даних, яка організовує дані в таблиці, що складаються з рядків і стовпців, де кожна таблиця представляє собою сутність, а кожен рядок відповідає запису. Зв'язки між цими таблицями встановлюються за допомогою ключів, що забезпечує ефективне зберігання, пошук та маніпулювання структурованими даними [13].

РСБД широко використовуються завдяки своїй надійності, гнучкості та дотриманню принципів нормалізації даних, що мінімізує надмірність та забезпечує цілісність даних. Вони підтримують можливості складних запитів за допомогою структурованої мови запитів (SQL), що дозволяє розробникам та користувачам ефективно витягувати та аналізувати інформацію.

У контексті автоматизованої інформаційної системи для формування індивідуалізованих освітніх траєкторій РСБД забезпечує ідеальну основу. Він підтримує представлення різноманітних сутностей даних, таких як користувачі, тести, запитання, результати, профілі кар'єри та освітні плани, а також підтримує чіткі зв'язки між ними. Така структура сприяє реалізації персоналізованих навчальних шляхів шляхом ефективного управління залежностями даних та взаємодією.

Ключові особливості СУБД включають транзакційну підтримку, яка гарантує атомарність, узгодженість, ізоляцію та довговічність (ACID) операцій. Це гарантує, що освітні дані, такі як результати тестів та успішність студентів, залишаються точними та узгодженими, навіть у разі збоїв системи або одночасного доступу. Крім того, СУБД забезпечують механізми безпеки, інструменти резервного копіювання та відновлення даних, а також варіанти масштабованості для обробки зростаючих обсягів даних у міру розширення системи [14].

Використання СУБД також спрощує інтеграцію із сучасними фреймворками веб-розробки, такими як Symfony, за допомогою інструментів об'єктно-реляційного відображення (ORM). Ці інструменти абстрагують прямі взаємодії з базами даних, роблячи процес розробки ефективнішим та менш схильним до помилок.

Загалом, РСКБД – це надійний, зрілий та добре підтримуваний технологічний вибір, який лежить в основі ефективного зберігання та управління структурованими освітніми даними, необхідними для успішного функціонування індивідуалізованої системи освітніх траєкторій.

Для цього проєкту було обрано реляційну систему керування базами даних (РСБД) MySQL завдяки її перевіреній надійності, широкому використанню та сумісності з PHP та фреймворком Symfony. MySQL пропонує надійне середовище для управління структурованими освітніми даними, такими як профілі користувачів, результати тестів, кар'єрні профілі та персоналізовані навчальні шляхи.

Однією з основних переваг MySQL є підтримка транзакцій ACID (атомірність, консистентність, ізоляція, довговічність), що гарантує надійне та послідовне виконання всіх операцій з базою даних. Це важливо для збереження точності конфіденційної інформації, такої як результати тестів та навчальний прогрес, де будь-яка втрата даних або невідповідність можуть негативно вплинути на навчальний процес користувача.

Впровадження почалося з проєктування логічної моделі даних, яка потім була перетворена на фізичну схему, що складається з нормалізованих таблиць. Ці таблиці містять такі сутності, як «Користувачі», «Тести», «Запитання», «Результати», «Профілі кар'єри», «Курси» та «Освітні траєкторії», кожна з яких має чітко визначені атрибути та обмеження зовнішніх ключів для підтримки цілісності посилань.

Індекси були додані до часто запитуваних стовпців для оптимізації швидкості отримання даних, особливо для операцій, пов'язаних з оцінюванням студентів та динамічною генерацією навчальних планів. Крім

того, там, де це необхідно, були реалізовані збережені процедури та тригери для автоматизації повторюваних завдань, таких як оновлення зведених результатів тестів або повідомлення користувачів про нові рекомендації щодо курсів.

Для взаємодії з базою даних MySQL було використано шар Doctrine ORM (об'єктно-реляційне відображення) фреймворку Symfony, що забезпечує абстракцію, яка спрощує маніпулювання даними та зменшує ймовірність вразливостей SQL-ін'єкцій. Ця інтеграція сприяє безперебійній комунікації між бізнес-логікою застосунку та постійним сховищем даних.

На завершення, вибір MySQL як СУБД у поєднанні з фреймворком Symfony забезпечує масштабовану, безпечну та ефективну основу для автоматизованого формування індивідуалізованих освітніх траєкторій. Така конфігурація забезпечує узгодженість даних та швидкість реагування, що є критично важливим для забезпечення надійного та зручного досвіду як для студентів, так і для адміністраторів.

Таким чином було створено базу даних в середовищі MySQL Workbench (Рис. 3.2).

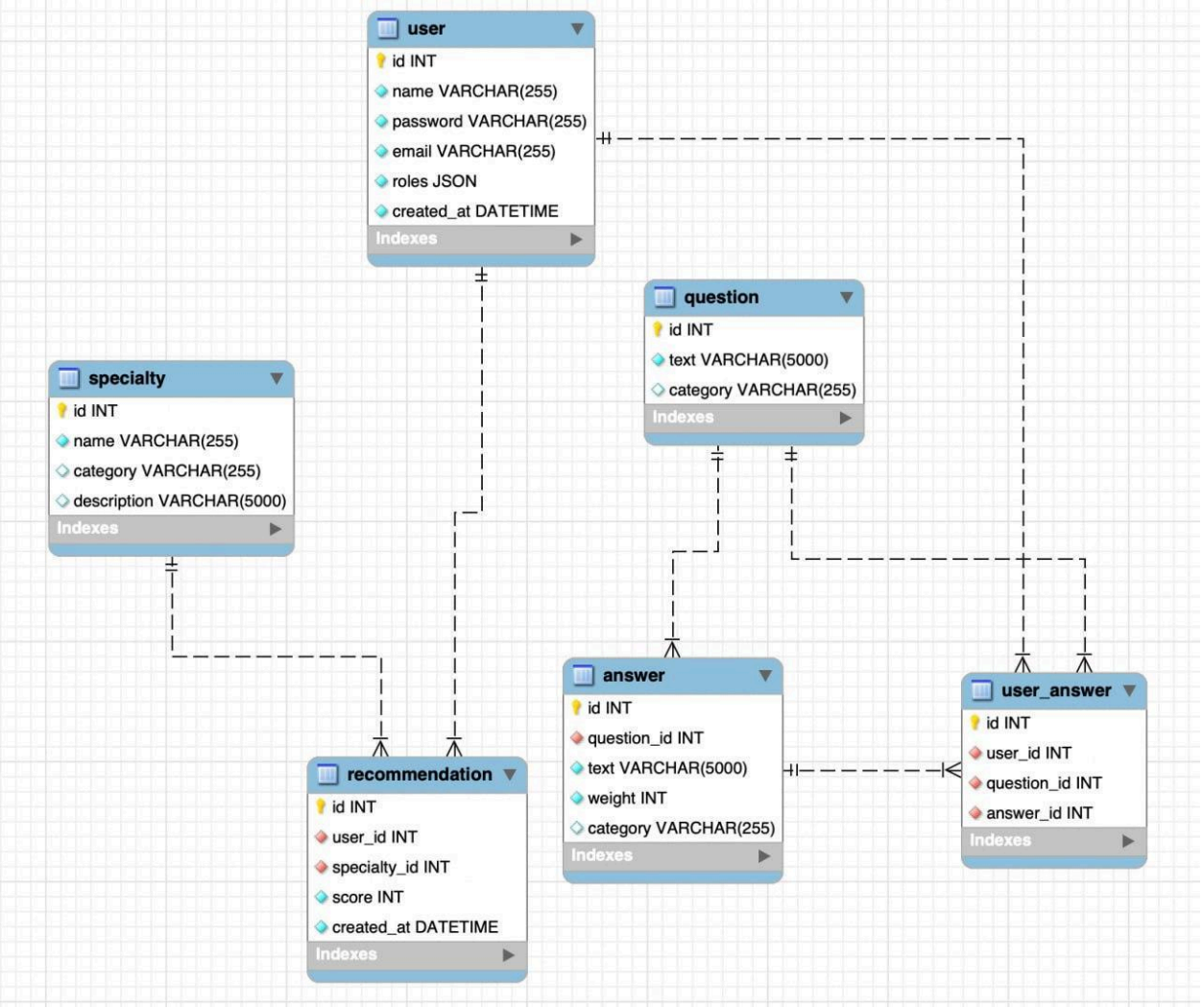

Рис. 3.2 База даних

Ця діаграма представляє структуру бази даних, яка використовується для зберігання та обробки інформації, пов’язаної з тестуванням. Вона включає основні таблиці та їх зв’язки, що дозволяє системі ефективно керувати тестами, учасниками та результатами.

У базі даних є таблиця Question, що містить інформацію про кожен створений тест, його назву та опис. Питання пов’язані з тестом, і кожне питання містить текст та можливі варіанти відповідей. Учасник реєструється у системі та проходить тест, після чого його відповіді передаються в таблицю Recommendation. Окремо працює система оцінювання, яка аналізує відповіді та зберігає оцінку учасника.

Зв’язки між таблицями відображають логіку взаємодії – наприклад, один тест може містити кілька питань, а кожен учасник може пройти кілька

тестів. Це робить структуру бази даних гнучкою і ефективною для зберігання та аналізу даних.

3.3 Архітектура програмного забезпечення

Архітектура програмного забезпечення автоматизованої інформаційної системи для формування індивідуальної освітньої траєкторії розроблена для забезпечення модульності, масштабованості та зручності обслуговування. Вона дотримується багаторівневої структури, яка розділяє завдання, дозволяючи незалежну розробку та легше управління компонентами системи [15].

В основі архітектури лежить шаблон Model-View-Controller (MVC), який розділяє застосунок на три взаємопов'язані шари. Рівень моделі представляє структуру даних та бізнес-логіку, керуючи такими сутностями, як користувачі, тести, кар'єрні профілі та освітні траєкторії. Рівень View обробляє логіку представлення, забезпечуючи зручний інтерфейс, через який здобуті вищої освіти та адміністратори взаємодіють із системою. Рівень Controller діє як посередник, обробляючи вхідні дані користувача, викликаючи бізнес-логіку та відповідно оновлюючи представлення.

Бекенд побудовано з використанням фреймворку Symfony, використовуючи його надійні функції для маршрутизації, безпеки та управління базами даних через Doctrine ORM. Цей рівень відповідає за обробку запитів, перевірку даних, керування сеансами та оркестрацію взаємодії між моделлю та представленням. Модульна система пакетів Symfony спрощує додавання нових функцій, таких як керування тестами, реєстрація користувачів та персоналізовані рекомендації.

Фронтенд використовує принципи адаптивного дизайну для забезпечення доступності на різних пристроях, включаючи настільні комп'ютери, планшети та смартфони. Він використовує механізми шаблонів та клієнтські технології для забезпечення плавного та інтерактивного

користувацького досвіду, необхідного для залучення студентів під час процесу складання тестів та перегляду їхніх персоналізованих освітніх шляхів.

Рівень бази даних реалізовано за допомогою MySQL, який зберігає постійні дані та підтримує складні запити, необхідні для створення індивідуальних траєкторій на основі результатів тестів та профілів кар'єри. Розділення логіки зберігання даних та застосування забезпечує кращу цілісність даних та легшу масштабованість.

Безпека інтегрована в усю архітектуру, а механізми автентифікації та авторизації контролюють доступ до конфіденційної інформації та адміністративних функцій. Перевірка даних та обробка помилок застосовуються на кількох рівнях для забезпечення стійкості та надійності системиv [15].

Підсумовуючи, обрана архітектура програмного забезпечення поєднує перевірені шаблони проєктування та сучасні технології для створення гнучкої, ефективної та безпечної системи. Ця структура не тільки відповідає поточним функціональним вимогам, але й забезпечує міцну основу для майбутніх удосконалень та масштабованості.

Рішення впровадити багаторівневу архітектуру для автоматизованої інформаційної системи, спрямованої на формування індивідуальних освітніх траєкторій, було зумовлене кількома ключовими міркуваннями, пов'язаними зі складністю та майбутньою масштабованістю програми. Багаторівнева архітектура поділяє систему на окремі шари, кожен з яких відповідає за певний аспект функціональності, що підвищує модульність та спрощує як розробку, так і обслуговування.

Однією з основних причин вибору цієї архітектури є чіткий розподіл завдань, який вона забезпечує. Виділяючи шар презентації (інтерфейс користувача), шар бізнес-логіки (процеси програми) та шар доступу до даних (взаємодія з базою даних), розробники можуть зосередитися на окремих компонентах незалежно. Це не тільки покращує організацію коду, але й

дозволяє командам працювати паралельно над різними частинами системи, не спричиняючи конфліктів чи залежностей.

Крім того, багаторівнева архітектура підвищує масштабованість системи. Зі зростанням кількості користувачів або впровадженням нових функцій кожен шар можна масштабувати або оновлювати незалежно. Наприклад, базу даних можна оптимізувати або розширювати, не змінюючи інтерфейс користувача, тоді як бізнес-логіка може розвиватися, щоб враховувати нові освітні моделі чи методи оцінювання, не порушуючи роботу інших шарів.

Безпека та цілісність даних також краще управляються в рамках багаторівневої структури. Такі чутливі операції, як автентифікація, авторизація та перевірка даних, централізовані на рівні бізнес-логіки, що забезпечує послідовне застосування політик безпеки в усьому застосунку. Такий багаторівневий підхід зменшує вразливості, які можуть виникнути через тісно пов'язані системи.

Більше того, ця архітектура підтримує легшу інтеграцію зі сторонніми сервісами та технологіями. Наприклад, рівень представлення може бути адаптований до різних клієнтських платформ або пристроїв, не впливаючи на основні бізнес-правила або управління даними, що робить систему більш гнучкою у задоволенні потреб користувачів.

На завершення, використання багаторівневої архітектури забезпечує надійну основу для системи, сприяючи ремонтопридатності, масштабованості, безпеці та адаптивності. Ці переваги узгоджуються з метою проєкту створити надійну, зручну для користувача та перспективну автоматизовану систему для керівництва студентів персоналізованими освітніми шляхами.

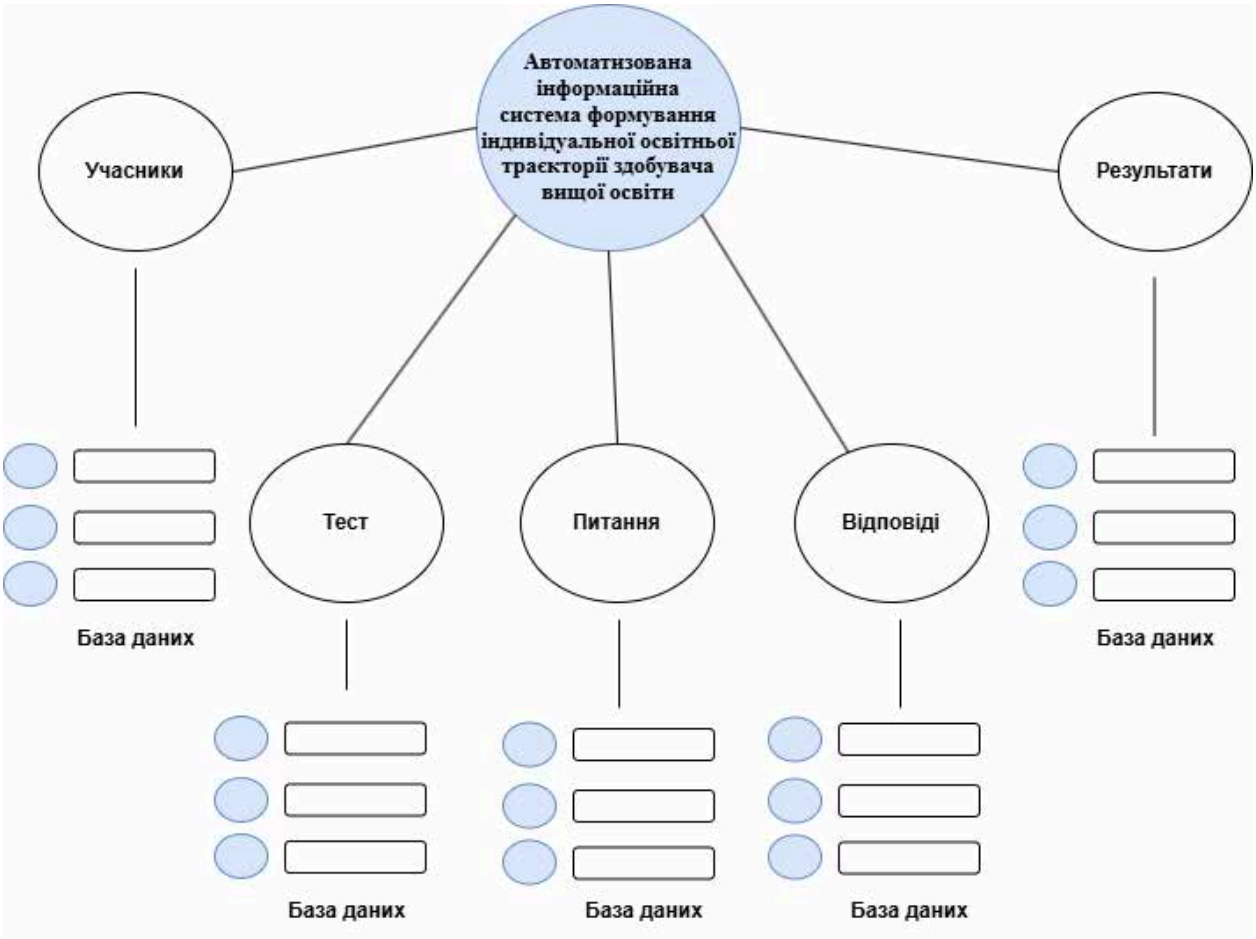

Рис. 3.3 Багатошарова архітектура

Ця діаграма демонструє багатошарову архітектуру програмного забезпечення, яка структурована за принципом розподілу функціональних компонентів між різними рівнями для забезпечення ефективності та масштабованості.

Архітектура складається з кількох основних шарів. Презентаційний рівень відповідає за взаємодію користувача із системою та містить інтерфейси для введення і отримання даних. Логічний (або бізнес) рівень обробляє логіку роботи застосунку, включаючи перевірку введених даних, виконання правил бізнес-процесів та взаємодію із сервісами. Рівень даних містить базу даних та механізми для її управління, забезпечуючи збереження та запит необхідної інформації.

Важливим елементом є взаємодія між рівнями – презентований інтерфейс передає запити до логічного рівня, який обробляє їх і надсилає

відповідні запити до рівня даних. Така структуризація дозволяє забезпечити модульність, що спрощує оновлення системи та підтримку її роботи.

3.4 Організаційна структура програмного забезпечення

3.4.1 Діаграма пакетів. Діаграма пакета — це структурна діаграма UML, яка організовує компоненти системи в групи вищого рівня, що називаються пакетами. Ці пакети допомагають керувати складністю програмного забезпечення, групуючи пов'язані класи, інтерфейси та інші елементи на основі їхньої функціональності або предметної області. Така організація сприяє модульності та покращує зручність обслуговування, чітко визначаючи залежності та взаємодії між різними частинами системи [16].

У контексті автоматизованої інформаційної системи для формування індивідуальної освітньої траєкторії, діаграма пакета візуально відображає, як організовані основні модулі програми. Типові пакети можуть включати керування користувачами, керування тестуванням, профілювання кар'єри, освітні шляхи та доступ до даних. Кожен пакет інкапсулює відповідні класи та компоненти, що відповідають за певні функції, такі як обробка реєстрації користувачів, створення та адміністрування тестів, аналіз результатів або генерування персоналізованих освітніх планів.

Діаграма також ілюструє залежності між пакетами, показуючи, як один модуль залежить від іншого або взаємодіє з ним. Наприклад, пакет «Освітні шляхи» може залежати від даних як з пакетів «Управління тестуванням», так і з пакетів «Профілювання кар'єри» для створення точних рекомендацій.

Використовуючи діаграму пакетів, розробники та зацікавлені сторони отримують чітке уявлення про структуру системи високого рівня, що полегшує комунікацію та керує процесом розробки. Вона слугує дорожньою картою, яка допомагає забезпечити організованість, масштабованість та легкість навігації системою в міру додавання або зміни нових функцій.

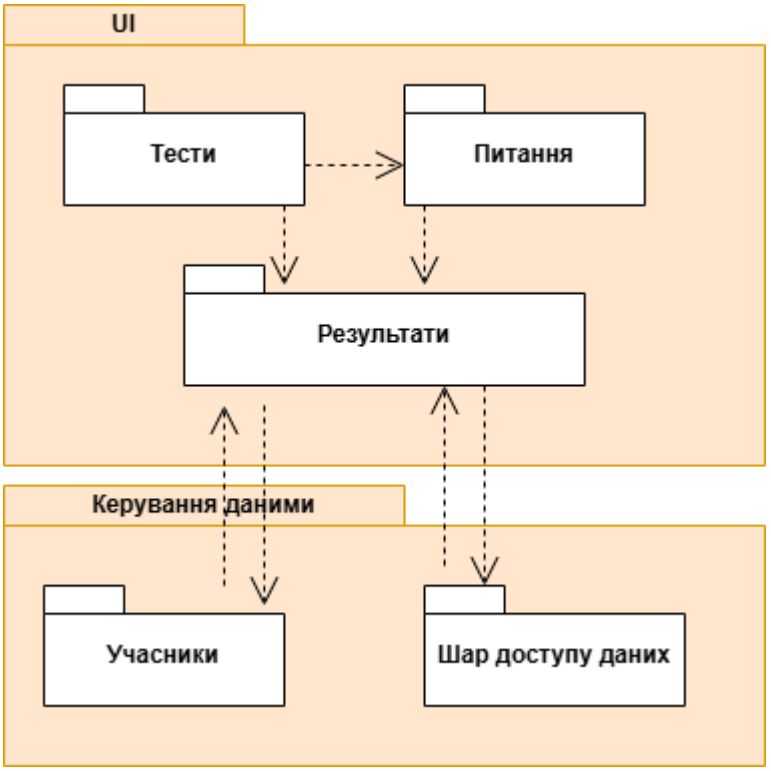

Рис. 3.4 Діаграма пакетів

Ця діаграма пакетів демонструє організацію модулів у системі тестування, розділяючи їх за функціональним призначенням. Вона охоплює дві основні ролі: учасник тестування та адміністратор, кожен з яких має свої відповідні пакети.

З боку учасника процес починається з доступу до системи, де він може обрати тест та розпочати його виконання. У певних умовах тест може бути припинений достроково, або ж, якщо всі питання пройдено, результати будуть автоматично сформовані та надані для перегляду. Учасник має можливість отримати оцінку та проаналізувати свої успіхи після завершення тесту.

Адміністратор, у свою чергу, керує тестами на системному рівні—він створює тести, додає до них питання та реєструє учасників, забезпечуючи правильний розподіл доступу та коректне функціонування тестового процесу. Важливим аспектом є можливість управління учасниками, що гарантує їх коректну реєстрацію та призначення до відповідних тестів.

Взаємозв'язки між пакетами чітко визначають потік даних та логіку взаємодії, що робить діаграму корисним інструментом для розробки програмного забезпечення тестування.

3.5 Вибір інструментарію для створення програмного забезпечення

Розробка автоматизованої інформаційної системи для формування індивідуальної освітньої траєкторії передбачає ретельний підбір інструментів та технологій для забезпечення ефективності, масштабованості та зручності обслуговування. Кожен обраний інструмент відповідає конкретним потребам проєкту, сприяючи створенню цілісного та надійного рішення.

Основною мовою програмування для цього проєкту є PHP, широко використовувана мова сценаріїв з відкритим кодом, яка добре підходить для веб-розробки. PHP пропонує сильну підтримку спільноти та сумісність з численними фреймворками, що робить його надійним вибором для створення динамічних веб-додатків.

Для структурування застосунку та оптимізації розробки використовується фреймворк Symfony. Symfony надає комплексний набір компонентів багаторазового використання та дотримується найкращих практик, таких як архітектурний шаблон Model-View-Controller (MVC). Його модульна конструкція та розгалужена екосистема забезпечують швидку розробку, зберігаючи при цьому високу якість коду та гнучкість.

Для зберігання та управління даними MySQL служить реляційною системою управління базами даних. Відомий своєю продуктивністю, надійністю та масштабованістю, MySQL ефективно обробляє структуровані дані, необхідні для системи, такі як профілі користувачів, результати тестів та освітні шляхи. Він бездоганно інтегрується з Symfony та підтримує складні запити, необхідні для персоналізованих рекомендацій.

Для полегшення взаємодії з базами даних використовується Doctrine ORM (Об'єктно-реляційний маппер). Doctrine абстрагує рівень бази даних, дозволяючи розробникам працювати з сутностями бази даних як з об'єктами PHP, спрощуючи маніпулювання даними та зменшуючи ризик помилок. Це підвищує продуктивність та забезпечує узгодженість операцій з даними.

В якості інтегрованого середовища розробки (IDE) було обрано PHPStorm, потужний інструмент, розроблений спеціально для розробки на PHP. PHPStorm пропонує інтелектуальне автодоповнення коду, можливості налагодження та інтеграцію з системами контролю версій, що разом покращує ефективність кодування та допомагає підтримувати якість коду протягом усього життєвого циклу проєкту.

Для забезпечення узгодженого та портативного середовища розробки для контейнеризації використовується Docker. Docker дозволяє упаковувати програму та її залежності в ізольовані контейнери, які можна легко розгортати на різних системах. Такий підхід спрощує процеси налаштування, тестування та розгортання, мінімізуючи проблеми, спричинені розбіжностями в середовищі.

Разом ці інструменти утворюють міцну технологічну основу, яка підтримує створення масштабованої, зручної в обслуговуванні та зручної автоматизованої системи для формування індивідуалізованої освітньої траєкторії. Їхнє поєднання балансує сучасні практики розробки з практичними міркуваннями, забезпечуючи успіх проєкту від розробки до розгортання.

4 ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ

4.1 Вимоги до апаратного та програмного забезпечення

Для забезпечення безперебійної роботи та ефективної роботи автоматизованої інформаційної системи, призначеної для формування індивідуальних освітніх траєкторій, необхідно дотримуватися певних вимог до апаратного та програмного забезпечення. Ці вимоги створюють основу як для середовищ розробки, так і для розгортання, гарантуючи ефективне та надійне функціонування системи.

Вимоги до апаратного забезпечення

Апаратна інфраструктура повинна підтримувати одночасно кількох користувачів, забезпечувати достатнє сховище для вмісту бази даних та забезпечувати швидку взаємодію. Мінімальна конфігурація сервера включає багатоядерний процесор, бажано Intel Xeon або еквівалент, з щонайменше 8 ГБ оперативної пам'яті для ефективного управління процесами програм та операціями з базою даних. Сховище має базуватися на SSD-накопичувачах, що пропонують щонайменше 100 ГБ вільного місця для розміщення системних файлів, даних користувачів, тестових матеріалів та резервних копій. Мережеве підключення має забезпечувати стабільний високошвидкісний доступ до Інтернету для забезпечення обміну даними в режимі реального часу та взаємодії з користувачами без значної затримки.

Для кінцевих користувачів, таких як студенти та адміністратори, система доступна через веб-браузери на персональних комп'ютерах, планшетах або смартфонах. Отже, клієнтські пристрої повинні підтримувати сучасні браузери (наприклад, Chrome, Firefox, Edge) та мати щонайменше 4 ГБ оперативної пам'яті зі стабільним підключенням до Інтернету для забезпечення безперебійного користувацького досвіду.

Вимоги до програмного забезпечення

Серверне середовище вимагає сучасної операційної системи, здатної підтримувати вибраний стек розробки. Дистрибутиви Linux, такі як Ubuntu або CentOS, є кращими завдяки своїй стабільності, безпеці та сумісності із серверними програмами. Програмний стек включає PHP (версії 8.x або вище) для виконання серверних скриптів, а також фреймворк Symfony, який керує логікою програми та обробкою запитів.

MySQL (версії 8 або вище) служить системою керування реляційними базами даних, відповідальною за зберігання та отримання структурованих даних. Для забезпечення ефективної комунікації між програмою та базою даних Doctrine ORM інтегровано як об'єктно-реляційний маппер.

Розробка та розгортання спрощуються за допомогою Docker, який вимагає встановлення Docker Engine на хост-системі для послідовного керування контейнерними компонентами програми в різних середовищах.

Вимоги на стороні клієнта включають сучасні веб-браузери з підтримкою HTML5, CSS3 та JavaScript для забезпечення сумісності з адаптивним та інтерактивним інтерфейсом користувача.

Відповідаючи цим специфікаціям апаратного та програмного забезпечення, система може забезпечити надійну, ефективну та безпечну платформу для користувачів, щоб вони могли взаємодіяти з персоналізованим формуванням освітньої траєкторії. Забезпечення відповідності системних вимог наявним ресурсам має вирішальне значення для підтримки стандартів продуктивності та забезпечення майбутнього зростання.

Також було зроблено діаграму розгортання для візуального огляду деплою системи (Рис. 4.1).

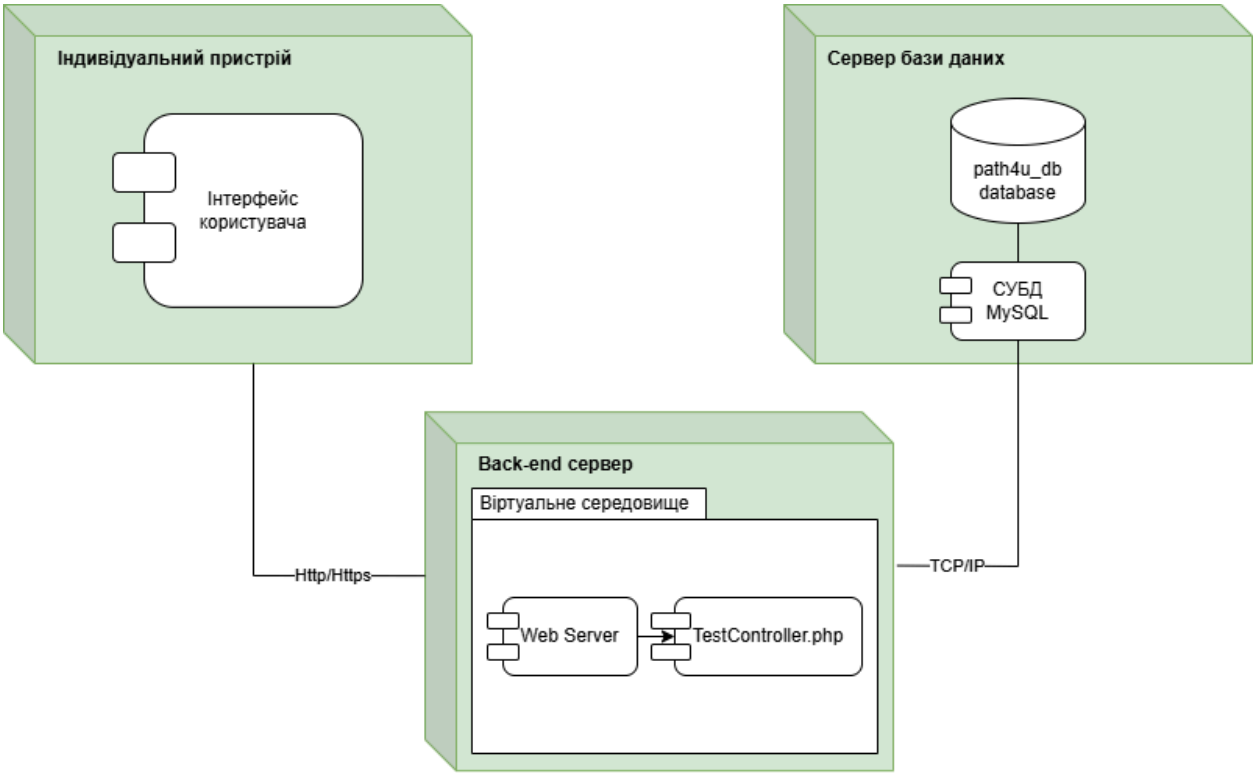

Рис. 4.1 Діаграма розгортання

4.2 Тестування системи

Запустивши систему, потрапляємо на форму реєстрації, тут нам треба ввести логін, email та пароль користувача (рис. 4.2).

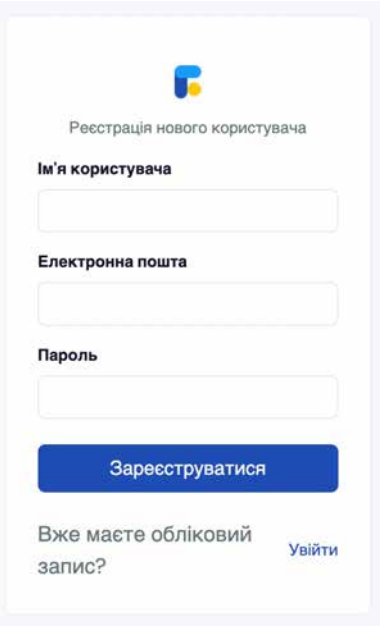

Рис. 4.2 Форма реєстрації

Після реєстрації та авторизації ми потрапляємо на основну сторінку даного сайту, на ній ми маємо кнопку для початку проходження тесту, який ми можемо почати і проходити відповідаючи на запитання (Рис. 4.3).

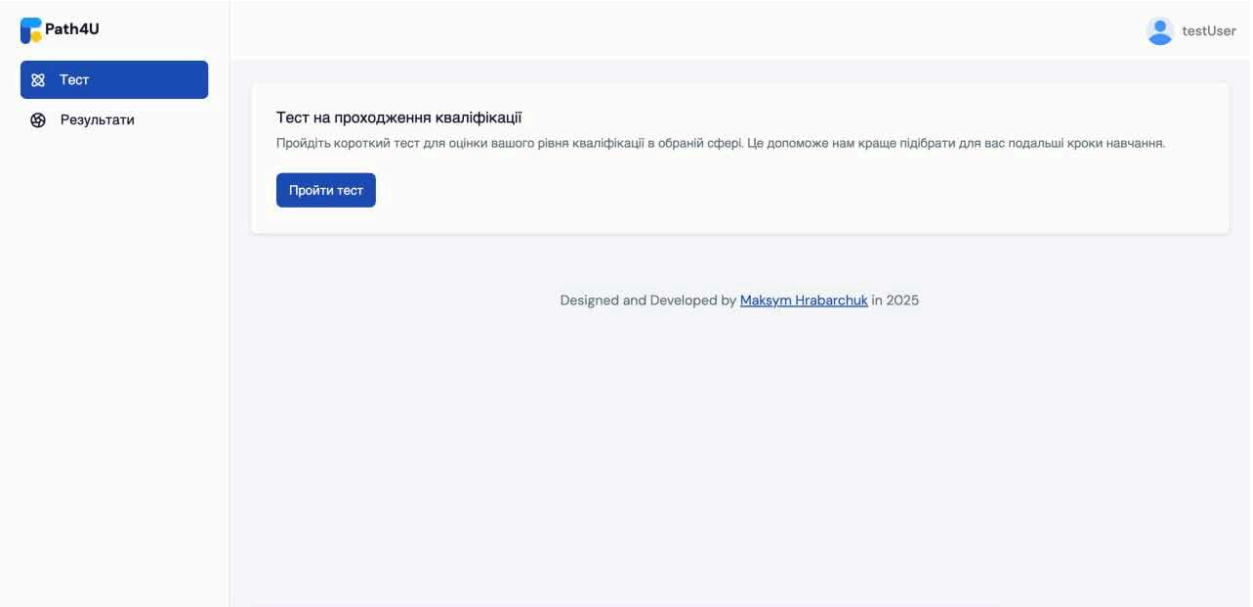

Рис. 4.3 Сторінка “Тест”

Розпочавши тест, потрапляємо на сторінку із запитаннями. Тут у нас буде виводиться по одному питанню на сторінку, на яке належить відповісти користувачеві. Як тільки користувач вибере відповідний варіант, він натискає на кнопку "Далі" і відкривається наступне питання, і так аж до всіх питань тесту (Рис. 4.4 - 4.6).

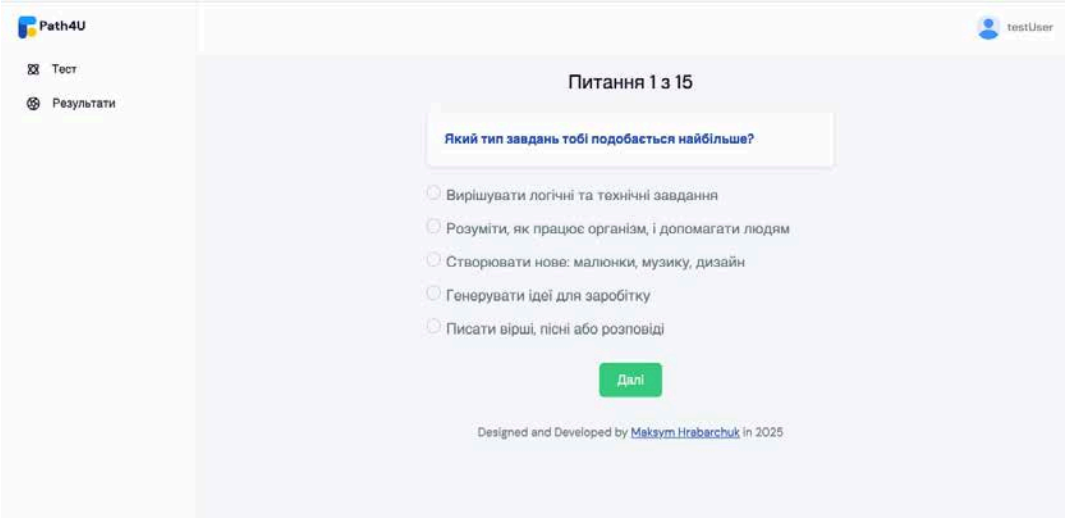

Рис. 4.4 Проходження тесту

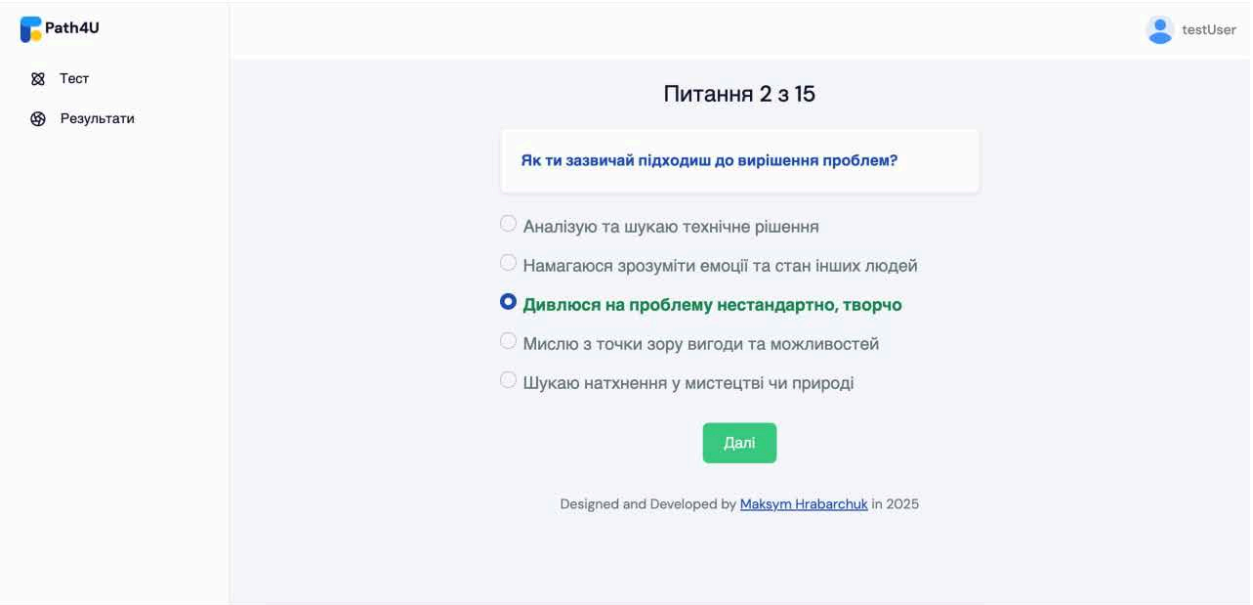

Рис. 4.5 Проходження тесту

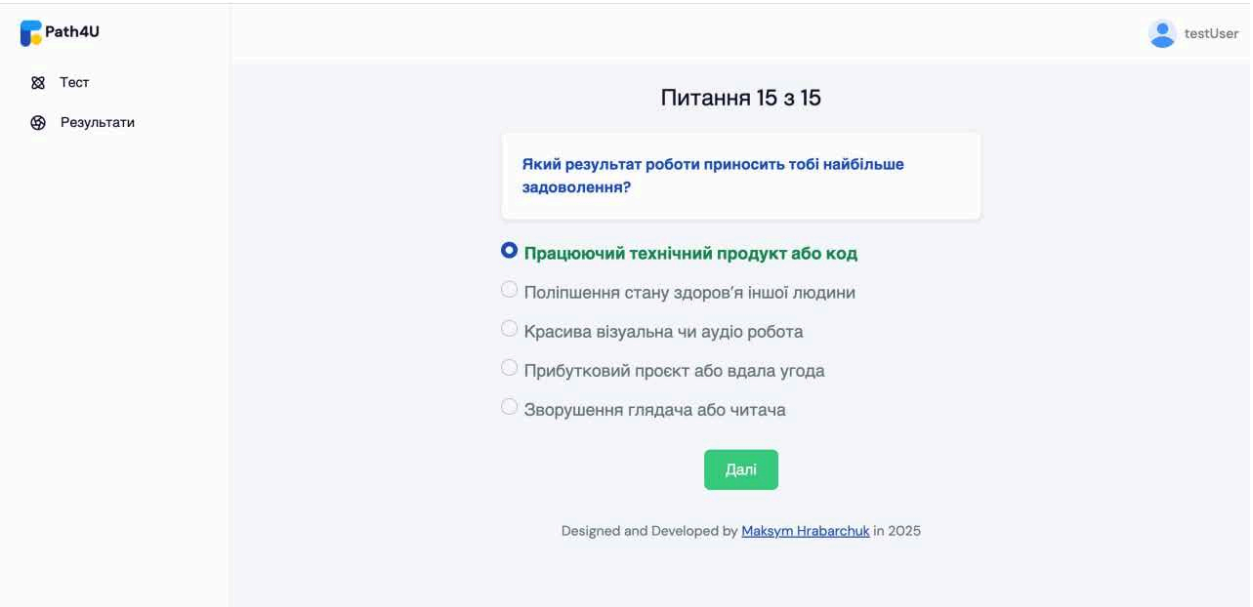

Рис. 4.6 Проходження тесту

Після проходження тесту, відповівши на всі запитання в ньому, у нас формується результат тесту, де система вважає внутрішню кількість балів за кожну відповідь за кожною категорією та видає користувачеві рекомендацію яка спеціальність йому більше підходить (Рис. 4.7).

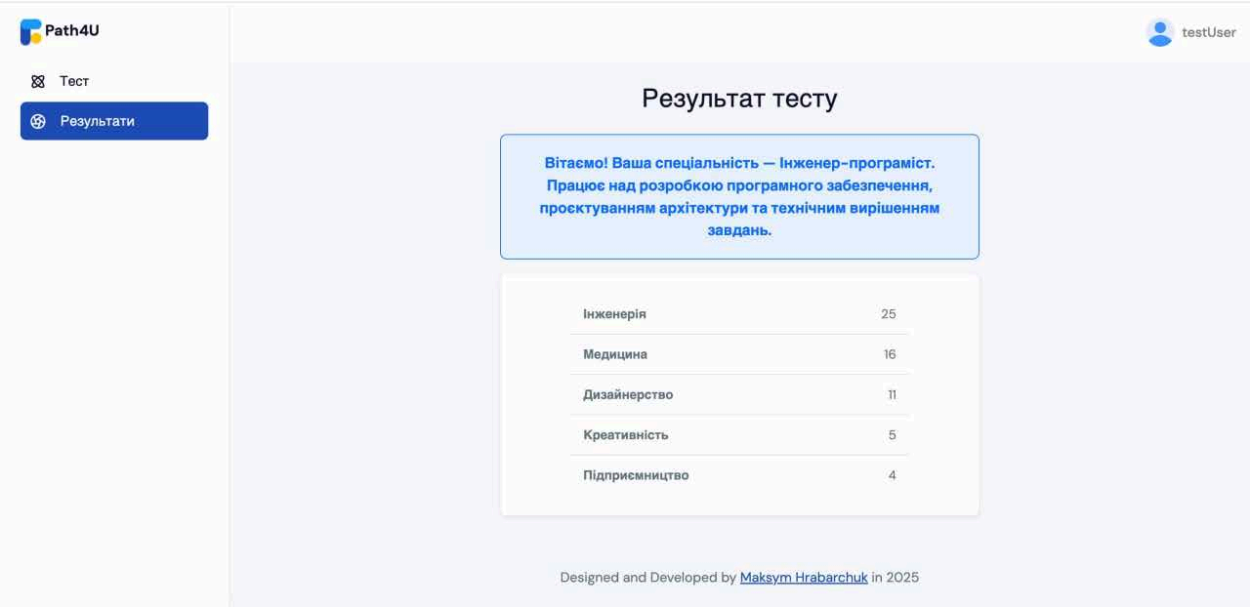

Рис. 4.7 Результати тесту

Кожен тест закріплюється на конкретному обліковому записі який його проходив, тому щоб поміняти обліковий запис, можна натиснути на іконку по верхньому правому кутку екрана (Рис. 4.8).

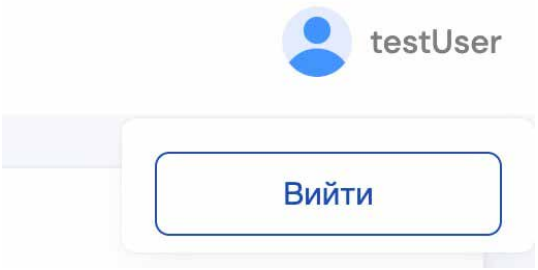

Рис. 4.8 Форма виходу з акаунту

Якщо користувач ще не проходив тест і спробує зайти на сторінку результатів, то в такому випадку нічого не буде виведено на сторінці, тільки сповіщення про те, що тест ще не пройдено (Рис. 4.9).

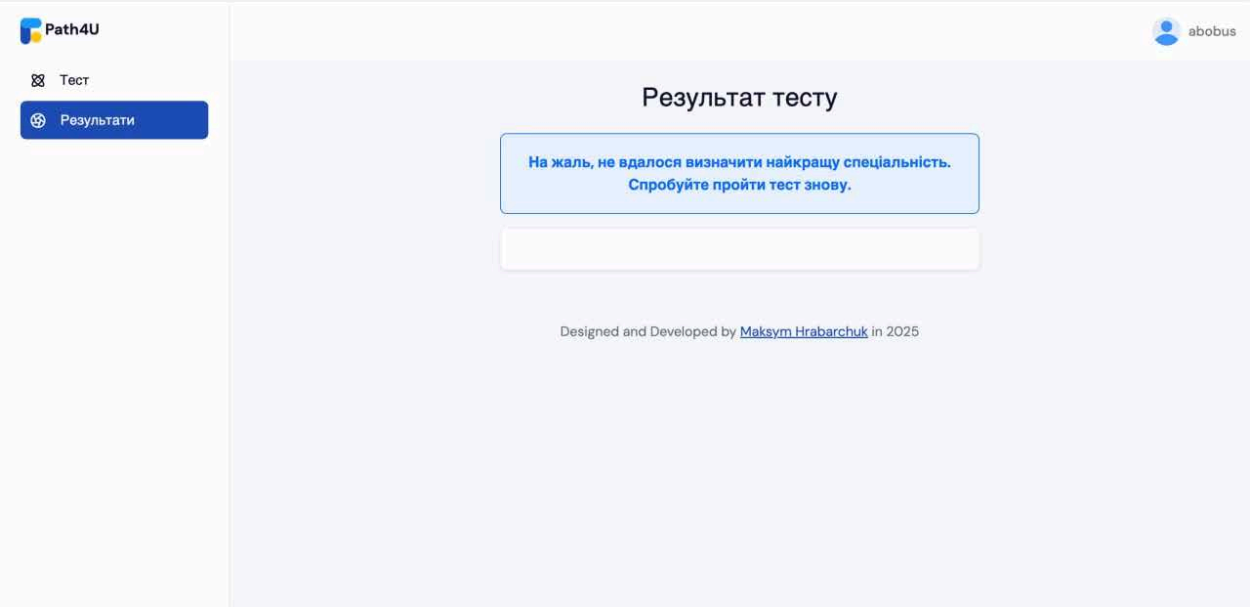

Рис. 4.9 Результати, якщо тест ще не пройдено

ВИСНОВКИ

Розробка автоматизованої інформаційної системи для формування індивідуальних освітніх траєкторій є значним кроком у підтримці студентів вищих навчальних закладів у їхньому академічному та професійному майбутньому. Використовуючи технології для оцінки кар'єрної орієнтації за допомогою персоналізованого тестування, система дозволяє студентам приймати більш обґрунтовані рішення щодо своїх освітніх шляхів. Такий цілеспрямований підхід допомагає зменшити невизначеність та сприяє проактивній участі в академічному розвитку.

Робота розпочалася з вивчення предметної області та формулювання проблеми дослідження. Існуючі рішення були переглянуті, щоб виявити прогалини та можливості для вдосконалення програмного забезпечення. Це послужило основою для наступних етапів роботи.

Було створено логічну модель даних, що представляє вимоги до інформації та зв’язки в програмному забезпеченні. Ця модель лягла в основу проєктування програмної системи. Розроблено фізичну модель даних, що забезпечує ефективне зберігання та пошук даних за допомогою вибраної системи керування базами даних.

Упродовж усього проєкту ретельний відбір сучасних та надійних технологій, таких як PHP, Symfony, MySQL та Doctrine ORM, забезпечив ефективність та масштабованість системи. Використання багаторівневої архітектури сприяє розділенню обов'язків, підвищує зручність обслуговування системи та дозволяє легше інтегрувати нові функції або модулі в майбутньому. Крім того, контейнеризація через Docker гарантує послідовне розгортання в різних середовищах, зменшуючи потенційні проблеми, пов'язані з конфігурацією програмного забезпечення.

Інтерфейс користувача системи був розроблений з урахуванням простоти та доступності, забезпечуючи зручний досвід як для студентів, так і

для адміністраторів. Ця простота використання є критично важливою для заохочення широкого впровадження та регулярного використання, що є важливим для успіху системи в практичних освітніх умовах. Автоматизуючи складний процес побудови персоналізованих освітніх траєкторій, система також зменшує адміністративне навантаження та дозволяє викладачам зосередитися на ефективнішій підтримці студентів.

Крім того, проєкт підкреслює потенціал інформаційних систем для трансформації традиційних освітніх моделей шляхом включення рекомендацій на основі даних та інтерактивних інструментів. Успішне впровадження цієї системи слугує основою для подальших удосконалень, таких як інтеграція з існуючими університетськими платформами, підтримка додаткових методів оцінювання або розширені аналітичні можливості для відстеження прогресу студентів з часом.

На завершення, автоматизована інформаційна система, розроблена в цій роботі, не лише відповідає початковим цілям, але й прокладає шлях до більш персоналізованого, ефективного та адаптивного підходу до вищої освіти. Вона є цінним інструментом, який відповідає потребам студентів та навчальних закладів, що постійно змінюються, зрештою сприяючи покращенню академічних результатів та готовності до кар'єри.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Рассел, М. Дж., Кан, Д. Аналітика навчання в освіті: прийняття рішень на основі даних. - Springer, 2017.
2. Бахші, Р., Шадаб, М., Каушік, А. Системи кар'єрної орієнтації у вищій освіті: огляд. - IGI Global, 2020.
3. Ларман, К. Застосування UML та шаблонів: вступ до об'єктно-орієнтованого аналізу та проєктування, а також ітеративної розробки. - Prentice Hall, 2004.
4. 123Test – [Електронний ресурс] – Режим доступу: https://www.123test.com
5. Truity – [Електронний ресурс] – Режим доступу: https://www.truity.com
6. Соммервіль, І. Програмна інженерія. - Pearson, 2016.
7. Прессман, Р. С., Максим, Б. Р. Програмна інженерія: підхід практикуючого спеціаліста. - McGraw-Hill, 2014.
8. Фаулер, М. UML Distilled: Короткий посібник зі стандартної мови моделювання об'єктів. - Addison-Wesley, 2012.
9. Амблер, С. В. Основи об'єктів: Гнучка розробка на основі моделей з UML 2.0. - Cambridge University Press, 2004.
10. Дейт, К. Дж. Вступ до систем баз даних. - Addison-Wesley, 2003.
11. Хартманн, С., Грох, Г. Проєктування інтерфейсів користувача для освітнього програмного забезпечення. - Wiley, 2015.
12. Мелл, П., Грейнс, Т. Визначення хмарних обчислень NIST. - Національний інститут стандартів і технологій, 2011.
13. Doctrine ORM – [Електронний ресурс] – Режим доступу: https://www.doctrine-project.org/projects/orm.html
14. Документація Symfony – [Електронний ресурс] – Режим доступу: https://symfony.com/doc/current/index.html

15. Документація MySQL – [Електронний ресурс] – Режим доступу: https://dev.mysql.com/doc/
16. Посібник з PHP – [Електронний ресурс] – Режим доступу: https://www.php.net/manual/en/
17. Документація Git – [Електронний ресурс] – Режим доступу: https://git-scm.com/doc
18. Посібник з HTML W3Schools – [Електронний ресурс] – Режим доступу: https://www.w3schools.com/html/
19. Посібник з CSS W3Schools – [Електронний ресурс] – Режим доступу: https://www.w3schools.com/css/
20. Посібник з JavaScript W3Schools – [Електронний ресурс] – Режим доступу: https://www.w3schools.com/js/
21. "«Важливість систем управління корпоративним навчанням» – [Електронний ресурс] – Режим доступу: https://elearningindustry.com/importance-corporate-learning-management-system
22. «Найкращі практики управління корпоративним навчанням» – [Електронний ресурс] – Режим доступу: https://www.simplilearn.com/corporate-training-best-practices-article
23. Інструмент для створення діаграм Draw.io – [Електронний ресурс] – Режим доступу: https://app.diagrams.net/
24. ПРОЄКТУВАННЯ ER-ДІАГРАМИ – [Електронний ресурс] – Режим доступу: https://nationalteam.worldskills.ua/skills/proektirovanie-er-diagrammy/
25. Діаграма варіантів використання (UseCase diagram) – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ua/fd_use-case-diagram.html
26. ПРОЄКТУВАННЯ USE CASE ДІАГРАМИ. ВИЗНАЧЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ СИСТЕМИ – [Електронний ресурс] – Режим доступу:

https://nationalteam.worldskills.ua/skills/proektirovanie-use-case-diagrammy-opredelenie-funktsionalnykh-vozmozhnostey-sistemy/

27. What is Sequence Diagram? – [Електронний ресурс] – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/
28. UML - Activity Diagrams – Tutorialspoint – [Електронний ресурс] – Режим доступу: https://www.tutorialspoint.com/uml/uml_activity_diagram.htm
29. Побудова діаграми класів – [Електронний ресурс] – Режим доступу: https://flexberry.github.io/ua/gpg_class-diagram.html
30. UML-діаграми класів – [Електронний ресурс] – Режим доступу: https://prog-cpp.ua/uml-classes/
31. Entity Relationship Diagram - Data Modeling – [Електронний ресурс] – Режим доступу: https://www.visualparadigm.com/VPGallery/datamodeling/EntityRelationshipDiagram.html
32. UML 2 Tutorial - Package Diagram - Sparx Systems – [Електронний ресурс] – Режим доступу: https://sparxsystems.com/resources/tutorials/uml2/package-diagram.html
33. Документація Bootstrap (офіційний сайт) – [Електронний ресурс] – Режим доступу: www.getbootstrap.com/documentation.

ДОДАТОК А

Фрагменти програмного коду. Функція проходження тесту

```
<?php

#[IsGranted('ROLE_USER')]
class TestController extends AbstractController
{
    #[Route('/test/{step}', name: 'app_test', requirements: ['step' => '\d+'])]
    public function test(
        int $step,
        QuestionRepository $questionRepo,
        AnswerRepository $answerRepo,
        Request $request,
        EntityManagerInterface $em
    ): Response {
        $questions = $questionRepo->findAll();
        $total = count($questions);

        if ($step > $total) {
            return $this->redirectToRoute('app_test_result');
        }

        $question = $questions[$step - 1];
        $answers = $question->getAnswers();

        if ($request->isMethod('POST')) {
            $answerId = $request->request->get('answer');
            $answer = $answerRepo->find($answerId);

            if ($answer && $this->getUser()) {
                $userAnswer = new UserAnswer();
                $userAnswer->setUser($this->getUser());
                $userAnswer->setQuestion($question);
                $userAnswer->setAnswer($answer);
                $em->persist($userAnswer);
                $em->flush();
            }

            return $this->redirectToRoute('app_test', ['step' => $step + 1]);
        }
```

```
        return $this->render('test/question.html.twig', [
            'question' => $question,
            'answers' => $answers,
            'step' => $step,
            'total' => $total
        ]);
    }

    #[Route('/test/result', name: 'app_test_result')]
    public function result(EntityManagerInterface $em): Response
    {
        $user = $this->getUser();

        $answers = $user->getUserAnswers();
        $scores = [];

        foreach ($answers as $userAnswer) {
            $answer = $userAnswer->getAnswer();
            $category = $answer->getCategory();
            $weight = $answer->getWeight();

            if (!isset($scores[$category])) {
                $scores[$category] = 0;
            }

            $scores[$category] += $weight;
        }

        foreach ($scores as $category => $score) {
                $specialty = $em->getRepository(Specialty::class)->findOneBy(['category' =>
$category]);
            if ($specialty) {
                $recommendation = new Recommendation();
                $recommendation->setUser($user);
                $recommendation->setSpecialty($specialty);
                $recommendation->setScore($score);
                $recommendation->setCreatedAt(new \DateTimeImmutable());
                $em->persist($recommendation);
            }
        }

        $em->flush();

        if (count($scores) > 0) {
            arsort($scores);
```

```
            $topCategory = key($scores);
                $topSpecialty = $em->getRepository(Specialty::class)->findOneBy(['category' =>
$topCategory]);
        } else {
            $topSpecialty = null;
        }

        return $this->render('test/result.html.twig', [
            'scores' => $scores,
            'topSpecialty' => $topSpecialty,
        ]);
    }
}
```

ДОДАТОК Б

Фрагменти програмного коду. Створення сутностей бази даних

```
<?php

namespace App\Entity;

use App\Repository\UserRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface;
use Symfony\Component\Security\Core\User\UserInterface;

#[ORM\Entity(repositoryClass: UserRepository::class)]
class User implements UserInterface, PasswordAuthenticatedUserInterface
{
    #[ORM\OneToMany(targetEntity: UserAnswer::class, mappedBy: 'user')]
    private Collection $userAnswers;

    public function __construct()
    {
        $this->userAnswers = new ArrayCollection();
    }

    public function getUserAnswers(): Collection
    {
        return $this->userAnswers;
```

```
}

#[ORM\Id]
#[ORM\GeneratedValue]
#[ORM\Column]
private ?int $id = null;

#[ORM\Column(length: 255)]
private ?string $name = null;

#[ORM\Column(length: 255)]
private ?string $password = null;

#[ORM\Column(length: 255)]
private ?string $email = null;

#[ORM\Column]
private array $roles = [];

#[ORM\Column]
private ?\DateTimeImmutable $createdAt = null;

public function getId(): ?int
{
    return $this->id;
}

public function getName(): ?string
{
    return $this->name;
}

public function setName(string $name): static
{
    $this->name = $name;

    return $this;
}

public function getPassword(): ?string
{
    return $this->password;
}

public function setPassword(string $password): static
```

```
{
    $this->password = $password;

    return $this;
}

public function getEmail(): ?string
{
    return $this->email;
}

public function setEmail(string $email): static
{
    $this->email = $email;

    return $this;
}

public function getRoles(): array
{
    return $this->roles;
}

public function setRoles(array $roles): static
{
    $this->roles = $roles;

    return $this;
}

public function getCreatedAt(): ?\DateTimeImmutable
{
    return $this->createdAt;
}

public function setCreatedAt(\DateTimeImmutable $createdAt): static
{
    $this->createdAt = $createdAt;

    return $this;
}

public function eraseCredentials(): void
{
}
```

```
    public function getUserIdentifier(): string
    {
        return $this->name;
    }
}
```