Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

Кафедра комп'ютерних систем управління

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка клієнт-серверної системи для автоматизації процесу формування команд учнів для навчання онлайн. Частина 1. Серверна частина»

Виконав: студент 2-го курсу, групи ЗАКІТР-23м спеціальності 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
(шифр і назва напряму підготовки, спеціальності)

Павло ІЩЕНКО
(ім'я та прізвище)

Керівник: к.т.н., доцент каф. АІІТ
Володимир КОЦЮБИНСЬКИЙ
(ім'я та прізвище)

«4» 12 2024 р.

Опонент: д.т.н., професор каф. КН
Ярослав ІВАНЧУК
(ім'я та прізвище)

«4» 12 2024 р.

Допущено до захисту
Зав. кафедри КСУ, д.т.н., проф.
В'ячеслав КОВТУН
(ім'я та прізвище)

«4» 12 2024 р.

Вінниця ВНТУ - 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп`ютерних систем управління
Рівень вищої освіти другий (магістерський)
Галузь знань – 17 – Електроніка, автоматизація та електронні комунікації
Спеціальність – 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Освітньо - професійна програма – Інформаційні системи і Інтернет речей

ЗАТВЕРДЖУЮ
Зав.кафедри КСУ, д.т.н., проф.
_____В'ячеслав КОВТУН
"_14_" ___жовтня____2024 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту _____Іщенку Павлу Сергійовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи _Розробка клієнт-серверної системи для автоматизації процесу формування команд учнів для навчання онлайн. Частина 1. Серверна частина.__
керівник роботи _Коцюбинський Володимир Юрійович, доцент кафедри АІІТ
затверджені наказом ВНТУ від "17" вересня 2024 року №310
2. Термін подання студентом роботи "1" грудня_ 2024 року
3. Вихідні дані до роботи: _____
_____1.Шарова Т. М. Освітній портал як ефективний засіб забезпечення дистанційного навчання здобувачів вищої освіти. Українські студії в європейському контексті. 2022. № 5. С. 237-244_____
_____2.Триус Ю.В., Герасименко І.В. Комбіноване навчання як інноваційна освітня технологія у вищій школі. Теорія та методика електронного навчання : зб. наук. праць. Випуск III. Кривий Ріг, 2012. С. 299–308._____
_____3. Горбунова В. В. Психологія командотворення: Ціннісно-рольовий підхід до формування та розвитку команд : монографія. – Житомир : Вид-во ЖДУ ім. І. Франка, 2014. – 380 с., іл. ISBN 978-966-485-162-3_____
_____
4. Зміст текстової частини: _Вступ. Аналіз предметної області та існуючих систем. Проекутвання системи.Розробка і тестування програмного забезпечення.Висновки. Перелік посилань. Додатки_____
_____
_____
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень) UML діаграма прецедентів, ER-моделі бази даних, UML діаграма розгортання, UML діаграма класів, UML діаграма об'єктів, UML діаграма компонентів, екранні форми системи, презентація_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв

7. Дата видачі завдання "14" жовтня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	02.09.2024	18.09.2024	Виконав
2	Огляд існуючих систем та рішень	19.09.2024	30.09.2024	Виконав
3	Постановка задачі та визначення основних функцій системи	01.10.2024	06.10.2024	Виконав
4	Проектування архітектури серверної	07.10.2024	13.10.2024	Виконав
5	Аналіз та вибір технологій розробки	14.10.2024	20.10.2024	Виконав
6	Розробка системи	21.10.2024	17.11.2024	Виконав
7	Тестування розробленої системи	18.11.2024	24.11.2024	Виконав
8	Оформлення пояснювальної записки, графічного матеріалу	25.11.2024	01.12.2024	Виконав
9	Захист МКР	12.12.2024	12.12.2024	

Студент _____ Павло ІЩЕНКО
(підпис) (ім'я і ПРІЗВИЩЕ)

Керівник роботи _____ Володимир КОЦЮБИНСЬКИЙ
(підпис) (ім'я і ПРІЗВИЩЕ)

АНОТАЦІЯ

УДК 004.75:371.3

Іщенко П.С. Розробка клієнт-серверної системи для автоматизації процесу формування команд учнів для навчання онлайн. Частина 1. Серверна частина. Магістерська кваліфікаційна робота зі спеціальності 174 - Автоматизація, комп'ютерно-інтегровані технології та робототехніка, освітньо-професійна програма - Інформаційні системи і Інтернет речей. Вінниця: ВНТУ, 2024, 112с.

На укр. мові. Бібліогр.: 40 назв; рис.: 55; табл. 1.

У теоретично-методичній частині роботи розглянуто та проаналізовано предметну область, тобто онлайн навчання та об'єкт автоматизації - процес автоматизації формування команд учнів для онлайн-навчання, основні принципи, елементи автоматизованої системи. Визначено основні проблеми, переваги, недоліки та перспективи розвитку системи в майбутньому.

У практичній частині було поставлено основні задачі, які повинна виконувати система, промодельована архітектура та підібрані технології, які потрібні для розробки. Виконана розробка та тестування програмного забезпечення.

Ключові слова: автоматизація, команда, онлайн навчання, система, освіта, сервер.

ABSTRACT

UDC 004.75:371.3

Ishchenko P.S. Development of a client-server system for automating the process of forming student teams for online learning. Part 1. Server part. Master's qualification work in the specialty 174 - Automation, computer-integrated technologies and robotics, educational and professional program - Information systems and the Internet of Things. Vinnytsia: VNTU, 2024, 112p.

In Ukrainian. Bibliography: 40 titles; fig.: 55; table. 1.

In the theoretical and methodological part of the work, the subject area, i.e. online learning and the object of automation - the process of automating the formation of student teams for online learning, the basic principles, elements of the automated system, are considered and analyzed. The main problems, advantages, disadvantages and prospects for the development of the system in the future are identified.

In the practical part, the main tasks that the system should perform, the modeled architecture and the selected technologies required for development were set. The software was developed and tested.

Keywords: automation, team, online learning, system, education, server.

ЗМІСТ

ВСТУП

Актуальність. В умовах сучасного освітнього процесу, який все більше переходить в онлайн-формат через глобальні виклики, як-от пандемія COVID-19 та війна в країні, постає необхідність у впровадженні ефективних систем для організації навчання. Особливо актуальним є автоматизація процесу формування команд учнів для онлайн-навчання, що дозволяє забезпечити зручність, швидкість та ефективність організації роботи в групах. У сучасному цифровому світі, де освітній процес відбувається в умовах активного використання технологій, особлива увага приділяється розробці клієнт-серверних систем, які можуть автоматизувати цей процес.

Сучасні технології дозволяють створювати інструменти для гнучкого й ефективного управління командами учнів в онлайн-середовищі. Автоматизована система для формування команд не лише оптимізує час викладачів і координаторів, але й допомагає створити більш справедливі, збалансовані групи для співпраці учнів. Такі системи сприяють підвищенню якості освітнього процесу, враховуючи індивідуальні особливості учнів, їхні здібності та рівень підготовки.

Актуальність теми обумовлена необхідністю інтеграції технологій автоматизації в освітні процеси, що дозволить підвищити ефективність навчання та поліпшити взаємодію між учасниками освітнього процесу. Тема "Розробка клієнт-серверної системи для автоматизації процесу формування команд учнів для навчання онлайн" є важливим напрямом для подальших досліджень і впровадження в освітню практику.

Метою дослідження є підвищення ефективності організації онлайн-навчання шляхом розробки серверної частини клієнт-серверної системи для автоматизації процесу формування команд учнів.

Для досягнення мети визначено такі основні завдання:

- здійснити аналіз предметної області та об’єкту;
- провести огляд існуючих рішень і визначити їх переваги та недоліки;
- сформулювати основні вимоги до серверної частини системи;

- розробити архітектуру системи;
- розробити необхідні UML-діаграми;
- визначити стек технологій для розробки серверної частини системи;
- реалізувати алгоритмічне та програмне забезпечення для реалізації серверної частини системи;
- провести тестування та оцінку ефективності роботи системи.

Об’єктом дослідження є процес автоматизації формування команд учнів для онлайн-навчання.

Предметом дослідження є автоматизація процесу формування команд за рахунок реалізації більш ефективної взаємодії серверної та клієнтської частини з базою даних.

Практична цінність полягає у розробці алгоритмічного та програмного забезпечення, яке робить процес навчання онлайн більш ефективним.

Інноваційність розробки полягає в застосуванні сучасних підходів до автоматизації навчального процесу, що сприяє покращенню організації роботи учнів у групах.

Апробація. Представлені в роботі результати апробовані в результаті участі в конференції ВНТУ: «Науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2024)».

Публікації. Іщенко Павло Сергійович, Коцюбинський Володимир Юрійович «Автоматизація інтеграції ігрових елементів у системах навчання в онлайн середовищі», «Науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2024)», 2024. URL: http://surl.li/sibwv

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ СИСТЕМ ДЛЯ НАВЧАННЯ ОНЛАЙН

1.1 Огляд процесу онлайн-навчання і його потреб

Онлайн-навчання стало невід'ємною частиною сучасної освітньої системи, особливо в умовах глобальних криз, таких як пандемія COVID-19 та війна в Україні. Впровадження дистанційного навчання[1] дозволило забезпечити безперервність освітнього процесу та адаптуватися до нових реалій. Однак, дистанційне навчання також створює низку викликів. Одним із них є організація командної роботи між учнями, що є важливою частиною багатьох навчальних курсів. В умовах відсутності фізичної присутності складніше координувати діяльність груп учнів та оцінювати їхню взаємодію. У зв'язку з цим виникає потреба в автоматизованих системах[2], які могли б ефективно формувати навчальні команди з урахуванням індивідуальних особливостей кожного учасника.

Онлайн-навчання має ряд переваг[3], які сприяють його популярності, деякі з них включають (за даними ChatGPT):

- Доступність. Учні можуть навчатися з будь-якої точки світу, маючи доступ до інтернету та відповідного пристрою;
- Гнучкість. Онлайн-навчання дозволяє адаптувати освітній процес до індивідуальних потреб учнів, надаючи можливість вибору зручного часу для занять;
- Широкий вибір навчальних ресурсів. Використання відео-лекцій, інтерактивних завдань, вебінарів та інших цифрових інструментів робить навчання більш різноманітним і цікавим;
- Автоматизація процесів. Технології дозволяють автоматизувати деякі процеси, такі як перевірка тестів, реєстрація на курси, управління навчальними ресурсами тощо.

Незважаючи на переваги, онлайн-навчання стикається з рядом викликів:

- Відсутність живої взаємодії. Один з основних недоліків онлайн-навчання — відсутність безпосереднього контакту між учнями та викладачем, що може негативно впливати на мотивацію та ефективність навчання;
- Технічні проблеми. Нерівномірний доступ до інтернету, застарілі пристрої або програмне забезпечення можуть створювати перешкоди для якісного навчання;
- Проблеми з організацією групової роботи. У онлайн-середовищі важче організувати взаємодію між учнями в межах групових проєктів. Це включає розподіл ролей, комунікацію та контроль за виконанням завдань.

Для ефективної організації онлайн-навчання необхідно врахувати наступні ключові потреби[4]:

- Платформа для взаємодії. Онлайн-навчання вимагає надійної та функціональної платформи для комунікації між учнями та викладачами. Така платформа повинна підтримувати обмін навчальними матеріалами, можливість проведення онлайн-конференцій та групових обговорень;
- Інтерактивність і зворотний зв’язок. Для підтримки мотивації учнів важливо забезпечити інтерактивність під час занять. Це можуть бути вікторини, дискусії в реальному часі, інтерактивні вправи та можливість швидкого отримання зворотного зв’язку від викладача;
- Автоматизація організаційних процесів. Потреба в автоматизації таких процесів, як формування команд для групових завдань, розподіл матеріалів, планування графіку занять, є критичною для забезпечення безперебійної роботи системи. Автоматизація дозволяє знизити навантаження на викладачів і забезпечити справедливий розподіл завдань між учнями;

- Індивідуальний підхід. Онлайн-навчання повинно враховувати індивідуальні потреби учнів, надавати адаптивний контент та можливість працювати у власному темпі. Це особливо важливо при формуванні груп для спільних проектів, де слід враховувати різні рівні знань і навичок учасників;
- Забезпечення безпеки даних. У зв'язку з масовим використанням онлайн-систем для навчання важливим аспектом стає захист персональних даних учнів та викладачів, що зберігаються у системах. Безпека інформації та конфіденційність мають бути пріоритетом при розробці таких систем.

Онлайн-навчання має значний потенціал[5], але його ефективність багато в чому залежить від правильного підходу до автоматизації процесів та організації групової роботи. Одним із ключових інструментів, який допоможе поліпшити якість навчання, є автоматизовані системи для формування команд учнів. Розробка таких систем, особливо їх серверної частини, є актуальним завданням для покращення організації навчання та взаємодії між учасниками освітнього процесу.

1.2 Аналіз об’єкта автоматизації

В контексті автоматизації процесу формування команд учнів для навчання онлайн, аналіз об'єкта автоматизації охоплює багато ключових аспектів.

Основна мета аналізу - визначення потреб та можливостей для впровадження автоматизаційних рішень, а також ідентифікація переваг, ризиків та вимог, пов'язаних з автоматизацією.

Під час аналізу об'єкта можна використовувати різні підходи та методи, спрямовані на комплексне вивчення та оптимізацію цього процесу[6]. Деякі з них включають:

1. Системний аналіз:
    - Опис процесів і взаємодії. Розгляд ідентифікації ключових етапів і взаємодії між елементами системи;

- Визначення вимог. Аналіз та визначення функціональних і нефункціональних вимог для досягнення поставлених цілей.

2. Педагогічний аналіз:
    - Визначення освітніх цілей. Оцінка того, як онлайн елементи відповідають освітнім цілям та завданням навчальної програми;
    - Адаптація для різних вікових груп. Розгляд можливостей адаптації для різних вікових категорій учнів.
3. Технічний аналіз:
    - Вибір технологій та платформ. Визначення оптимальних технічних рішень для інтеграції, враховуючи доступні технології та платформи;
    - Забезпечення безпеки та конфіденційності. Аналіз та визначення заходів для забезпечення безпеки та конфіденційності.
4. Економічний аналіз:
    - Оцінка вартості інтеграції. Розгляд витрат на впровадження і підтримку онлайн системи;
    - Визначення ефективності від інтеграції. Аналіз ефективності інтеграції онлайн навчання з точки зору покращення навчального процесу та досягнення освітніх цілей.
5. Оцінка впливу на навчальний процес:
    - Збір та аналіз статистики. Використання даних та статистики для оцінки впливу онлайн навчання на навчальний процес;
    - Оцінка залученості та мотивації. Аналіз, як онлайн навчання впливає на рівень залученості та мотивації учнів.

Автоматизація процесу формування команд передбачає використання спеціалізованих алгоритмів і програмного забезпечення для ефективного та швидкого розподілу учнів на групи, з урахуванням різноманітних критеріїв, таких як рівень знань, навички, інтереси, особистісні риси та технічні можливості[7]. Автоматизовані системи забезпечують масштабованість і об’єктивність при створенні команд, що особливо важливо для великих груп студентів, де ручне формування стає надто складним та часозатратним.

Особливості об'єкта автоматизації. Процес формування команд учнів є багатокомпонентним завданням, яке включає в себе аналіз різних факторів для досягнення збалансованого та ефективного розподілу учасників. У контексті онлайн-навчання цей процес стає ще складнішим через обмежену можливість особистого спостереження за учнями з боку викладача, а також через технічні обмеження, пов'язані з доступом до інтернету та ресурсів.

Основні етапи процесу формування команд:

- Збір даних про учнів – анкета (успішність, соціальні навички, попередні результати);
- Визначення критеріїв для розподілу на команди (баланс знань, умінь, інтересів, особистісних характеристик);
- Побудова команд з урахуванням заданих параметрів;
- Перегляд і корекція команд за потреби.

Автоматизована система повинна не лише аналізувати дані про учнів, але й пропонувати рішення, які сприяють ефективному виконанню завдань у команді та максимальному залученню всіх її учасників.

Складові об'єкта автоматизації. Процес формування команд включає в себе такі ключові елементи, що потребують автоматизації:

- Збір та аналіз даних про учнів. Цей етап включає отримання інформації про академічну успішність, поведінкові риси, рівень мотивації та інші фактори, які можуть впливати на якість командної роботи. Автоматизована система повинна вміти інтегрувати дані з різних джерел, таких як електронні журнали успішності, онлайн-тести та анкети учнів;
- Алгоритми розподілу. Ключовою частиною автоматизації є алгоритми, які обробляють вхідні дані та пропонують оптимальний варіант розподілу учнів на команди. Алгоритми можуть базуватися на різних критеріях, включаючи академічну успішність, особистісні риси, інтереси та здатність до співпраці;

- Баланс навичок та інтересів у команді. Формування команд має враховувати різноманітність навичок і здібностей учнів. Автоматизована система повинна прагнути до створення команд, де буде досягнуто балансу між сильними та слабкими сторонами учасників, що сприятиме більш ефективному виконанню завдань;
- Можливість корекції та адаптації. Система повинна надавати викладачу можливість вносити корективи в розподіл команд. Це важливо, оскільки викладач, на основі свого досвіду, може краще розуміти взаємовідносини між учнями або їхні індивідуальні потреби, які алгоритм не завжди здатен повністю врахувати.

Критерії ефективності автоматизації. Для оцінки ефективності автоматизованої системи формування команд необхідно враховувати кілька ключових критеріїв:

- Час, який затрачений на формування команд. Однією з основних переваг автоматизації є скорочення часу на процес формування груп. Система повинна забезпечувати швидке виконання цього завдання, навіть для великих класів або курсів;
- Об'єктивність розподілу. Автоматизовані системи повинні забезпечувати рівноправний розподіл учнів, зменшуючи вплив суб'єктивних чинників і забезпечуючи справедливість;
- Якість роботи команд. Ефективність роботи автоматизованої системи можна оцінювати за результатами командних завдань. Команди, створені за допомогою системи, мають бути здатні до ефективної співпраці та досягнення позитивних результатів у груповій роботі;
- Задоволеність учнів і викладачів. Автоматизація повинна сприяти не лише покращенню технічних аспектів навчального процесу, але й підвищенню мотивації та задоволеності учасників процесу. Важливо, щоб учні почували себе комфортно в своїх командах, а викладачі мали змогу легко керувати цим процесом.

Виклики при автоматизації формування команд. Незважаючи на всі переваги автоматизації, є й певні виклики, з якими можна стикнутися в процесі впровадження:

- Складність збору даних. Збір даних про учнів може бути ускладнений технічними обмеженнями або нестачею інформації. Деякі важливі аспекти, такі як особисті взаємостосунки між учнями, можуть бути важко вимірюваними та не завжди доступними для автоматизованих систем;
- Інтеграція з іншими системами. Для ефективного функціонування, автоматизована система повинна бути інтегрована з іншими освітніми платформами та базами даних, що може вимагати значних зусиль і ресурсів;
- Необхідність адаптації під конкретні освітні програми. Система повинна бути гнучкою, щоб відповідати різним навчальним програмам та методикам викладання. Різні курси можуть вимагати різних підходів до формування команд, що вимагає налаштування алгоритмів.

1.3 Проблеми при формуванні команд учнів для групових завдань

Одним з ключових елементів сучасного онлайн-навчання є організація групової роботи, яка сприяє розвитку комунікативних навичок, співпраці та критичного мислення[8]. В умовах онлайн-навчання виникають труднощі з формуванням збалансованих і ефективних команд. Цей процес включає:

- Оцінку навичок та рівня підготовки. Необхідно враховувати рівень знань учнів для створення команд, які зможуть працювати продуктивно;
- Розподіл ролей. Важливо автоматизувати розподіл ролей у групах на основі здібностей та уподобань учнів;

- Збалансованість команд. Формування збалансованих команд, де учасники мають доповнювати одне одного за навичками та досвідом, сприяє більш успішному виконанню завдань.

Формування команд для групових завдань в освітньому процесі є критичним елементом, який значно впливає на результативність навчання та розвиток ключових компетенцій учнів[9], таких як комунікація, співпраця, лідерські якості та вирішення проблем. Проте в умовах як традиційного, так і онлайн-навчання цей процес супроводжується рядом складнощів і викликів, які можуть перешкоджати успішній реалізації групових проєктів.

Розберемо основні проблеми при формуванні команд учнів для групових завдань[10], деякі з них включають (за даними ChatGPT):

- Нерівномірний розподіл навичок і знань між учнями. З найбільших проблем при формуванні команд є нерівномірний розподіл навичок і знань між учасниками. Кожен учень має різний рівень підготовки, що може призвести до того, що сильніші учні беруть на себе більшу частину роботи, а слабші залишаються пасивними учасниками. Це, в свою чергу, знижує ефективність навчального процесу, оскільки всі учасники не отримують рівноцінного досвіду;
- Відсутність інтересу або мотивації до групової роботи. Учні часто не виявляють інтересу до роботи в команді, особливо якщо вони не мають змоги вибирати своїх партнерів або тематику проєкту. Це може бути пов'язано з недостатньою мотивацією до взаємодії з іншими або небажанням працювати з певними однокласниками. У результаті така команда не функціонує ефективно, і завдання виконується на низькому рівні;
- Проблеми комунікації в команді. Навіть за наявності добре сформованих груп, учні можуть стикатися з проблемами комунікації, особливо в онлайн-середовищі. Недостатня взаємодія між учасниками команди може ускладнити координацію завдань, що призводить до затримок у виконанні роботи або до неякісних результатів. Онлайн-

формат додає додаткових бар'єрів, таких як технічні проблеми, різні графіки учасників або недостатнє володіння технологіями;

- Невизначеність або конфлікт ролей у команді. Під час групової роботи часто виникають ситуації, коли ролі учасників не чітко визначені. Це може викликати конфлікти, особливо якщо всі або жоден з учнів не хоче взяти на себе лідерські функції. Відсутність чітко розподілених завдань ускладнює координацію та відповідальність за результати роботи. Конфлікти ролей також можуть призвести до домінування одних учасників і пасивності інших, що погіршує загальний результат;
- Суб'єктивність викладача при формуванні команд. Часто викладач формує команди на власний розсуд, базуючись на особистих спостереженнях або випадковому виборі, що може призвести до непередбачуваних наслідків. Такий підхід може не враховувати реальні знання та навички учнів, їхні інтереси та особистісні характеристики. Як наслідок, деякі команди можуть виявитися надто сильними або слабкими, що створює нерівні умови для всіх учасників;
- Відсутність індивідуального підходу. У процесі формування команд важливо враховувати індивідуальні особливості кожного учня, такі як їхні сильні та слабкі сторони, інтереси та рівень залученості до навчання. Відсутність такого підходу може призвести до того, що учні з високим рівнем підготовки опиняються в одній команді, а ті, хто потребує більше допомоги, залишаються в іншій. Це може порушити баланс у навчанні та знизити мотивацію слабших учнів;
- Проблеми з доступністю технологій та інструментів. Онлайн-навчання часто стикається з технічними проблемами, які можуть ускладнювати роботу в командах. Не всі учні мають однаковий доступ до необхідних технологій, таких як швидкісний інтернет або сучасні пристрої для участі в онлайн-зустрічах. Це може створювати нерівні умови для виконання завдань і гальмувати загальний процес;

- Часові обмеження та проблеми з розкладом. Різні графіки учнів, особливо в умовах онлайн-навчання, можуть створювати додаткові труднощі при координації групових завдань. Якщо учасники команди живуть у різних часових поясах або мають різний доступ до інтернету протягом дня, це може ускладнити проведення спільних сесій для обговорення або виконання проєктів;
- Невміння працювати в команді. Деякі учні не мають достатнього досвіду роботи в команді або не розуміють важливості співпраці для досягнення спільної мети. Вони можуть намагатися працювати самостійно або ігнорувати інших учасників, що негативно впливає на динаміку роботи групи та результативність виконання завдання;
- Нестача часу на зворотний зв'язок і рефлексію. Після завершення групового завдання важливо надати учням зворотний зв’язок та можливість рефлексії щодо роботи команди. Проте часто цей етап ігнорується через брак часу або ресурсів. Відсутність аналізу процесу може призвести до повторення тих самих помилок у майбутньому, що знижує ефективність навчання.

Процес інтеграції у системи навчання - це комплексна діяльність, спрямована на впровадження покращення навчання та залучення учнів.

1.4 Порівняння ручного та автоматизованого формування команд

Процес формування команд учнів для групових завдань є важливою складовою сучасного навчального процесу, особливо в умовах онлайн-освіти. Однак ручний підхід до цього процесу часто супроводжується низкою проблем, таких як суб'єктивність вибору, нерівномірний розподіл навичок і знань між учасниками, та складнощі в управлінні великими групами учнів. У таких умовах автоматизація цього процесу стає необхідною, щоб забезпечити ефективність і справедливість у створенні команд.

Проблеми ручного формування команд. Ручне формування команд, хоча й дозволяє викладачам враховувати певні аспекти особистих знань та відносин між учнями, має ряд недоліків:

- Суб'єктивність та упередженість. Вибір команд може залежати від особистих вражень викладача, які не завжди точно відображають реальний рівень знань та навичок учнів. Це може призводити до того, що одні учні отримують переваги, а інші опиняються в невигідному становищі;
- Трудомісткість і часозатратність. Для формування команд необхідно витратити значну кількість часу на аналіз рівня знань, інтересів, сильних і слабких сторін кожного учня. Це стає особливо складним завданням при великій кількості учасників;
- Нерівномірний розподіл. Вручну важко досягти збалансованого розподілу учнів за рівнем підготовки. Це може призвести до того, що сильніші учні виконуватимуть основну частину роботи, а слабші залишатимуться в пасивній ролі, що знижує ефективність командної роботи.

Переваги автоматизованого підходу. Автоматизація формування команд може значно ускладнити систему[11], а також може полегшити процес, зробивши його більш ефективним та прозорим. Основні переваги автоматизованого підходу включають:

- Об'єктивність. Використання алгоритмів для автоматичного формування команд дозволяє усунути суб'єктивність і упередженість. Автоматизована система може аналізувати великий обсяг даних про учнів, враховуючи їхні оцінки, навички, стиль роботи та інші параметри, що забезпечує справедливий розподіл;
- Швидкість і ефективність. Алгоритми можуть формувати команди набагато швидше, ніж це робиться вручну. Це особливо важливо в умовах онлайн-навчання, де час викладача обмежений, а швидкість адаптації процесу є критичною;

- Індивідуалізація. Автоматизовані системи можуть враховувати індивідуальні особливості учнів, такі як їхні інтереси, рівень знань, поведінкові особливості тощо. Це дозволяє створювати команди, які не тільки збалансовані за рівнем підготовки, але й мають високу мотивацію до співпраці;
- Гнучкість. Автоматизовані системи дозволяють швидко змінювати команди у випадку необхідності, наприклад, при зміні складу учнів або їхніх інтересів. Це важливо для динамічних освітніх середовищ, де вимоги до команди можуть змінюватися в процесі виконання завдань.

Більш детальне порівняння ручного та автоматизованого підходу зображено в таблиці 1.1:

Таблиця 1.1 – Порівняння ручного та автоматизованого підходу

Критерій	**Ручний підхід**	**Автоматизований підхід**
Швидкість	Повільний, вимагає часу на аналіз учнів та формування команд	Миттєве формування команд на основі заздалегідь визначених параметрів
Об'єктивність	Може бути упередженим через особисті враження викладача	Об'єктивний, заснований на даних та алгоритмах
Гнучкість	Зміни у складі команд потребують часу та зусиль	Легко адаптується до змін
Індивідуальний підхід	Часто не враховує всі особливості учнів	Може враховувати різні параметри для кожного учня
Збалансованість команд	Нерівномірний розподіл навичок і знань	Збалансовані команди за рівнем підготовки та інтересами

Мотивація учнів	Може знижувати мотивацію через неправильний підбір команд	Підвищує мотивацію завдяки врахуванню інтересів і можливостей

Автоматизація процесу формування команд допомагає підвищити ефективність групової роботи, роблячи її більш структурованою та справедливою. Завдяки об'єктивності алгоритмів, учні отримують рівні можливості для самовираження та розвитку. Крім того, автоматизовані системи можуть легко інтегруватися з іншими освітніми інструментами, що сприяє подальшій оптимізації навчального процесу.

1.5 Можливості, способи та перспективи розвитку системи у майбутньому

Зростання можливостей та доступності різноманітних пристроїв, включаючи віртуальну (VR) та доповнену реальність (AR)[12], роблять систему "Автоматизація інтеграції ігрових елементів у системах навчання в онлайн середовищі" ще більш гнучкою та ефективною.

Збільшенням можливостей девайсів, таких як смартфони, планшети та персональні комп'ютери, користувачі отримають більше можливостей для взаємодії з навчальною платформою. Використання віртуальної та доповненої реальності може розширити можливості онлайн системи навчання та надати учням інтерактивні та захоплюючі навчальні досвіди.

Технології штучного інтелекту (ШІ)[13] можуть створити індивідуалізовані та адаптивні навчальні програми, які враховують потреби кожного учня. Такий підхід може значно поліпшити ефективність освітнього процесу та зробити його більш цікавим для учнів.

Інтеграція автоматизації у навчальні програми може відбуватися різними способами, залежно від конкретних цілей, предметів навчання та доступних технологічних можливостей:

1. Використання навчального програмного забезпечення:
    - Електронні підручники та ресурси;
    - Віртуальні лабораторії.
2. Ігрові технології в навчанні:
    - Гейміфікація;
    - Симулятори та серйозні ігри.
3. Використання інтерактивних технологій:
    - Відеолекції та відеоматеріали;
    - Вебінари та віддалені заняття.
4. Системи автоматизації та управління навчальним процесом:
    - Електронні системи оцінювання;
    - Інтегровані платформи.
5. Розробка індивідуалізованих програм:
    - Системи адаптивного навчання.
6. Використання ШІ в навчанні:
    - Машинне навчання та аналітика.

Інтеграція автоматизації в навчальні програми може бути етапним процесом, що включає в себе вивчення та впровадження нових технологій з метою покращення ефективності та доступності навчання.

Майбутній розвиток системи "Автоматизація інтеграції ігрових елементів у системах навчання в онлайн середовищі" буде залежати від вдосконалення технологічних засобів, а саме, розширення їх можливостей.

1.6 Аналіз існуючих систем для інтеграції ігрових елементів у навчальні програми

З появою комп'ютерів і цифрових технологій гейміфікація отримала новий поштовх. Перші спроби впровадження ігрових елементів, онлайн-систем у навчання були відзначені в ранніх комп'ютерних програмах для навчання, таких як "Oregon Trail" та "Lemonade Stand". Проте справжній прорив стався в останні десятиріччя.

Аналіз існуючих систем для інтеграції ігрових елементів у навчальні програми може включати в себе огляд різних платформ[14], інструментів та підходів, що дозволяють вчителям і розробникам впроваджувати групові та індивідуальні ігрові елементи в процес навчання.

Аналіз існуючих систем для інтеграції ігрових елементів у навчальні програми допомагає виявити сильні та слабкі сторони, ідентифікувати проблемні зони та визначити напрямки для подальшого вдосконалення. Результати аналізу служать основою для розробки стратегій оптимізації, впровадження нових технологій та вдосконалення вже існуючих процесів.

Тому проведемо аналіз існуючих систем для інтеграції ігрових елементів у навчальні програми:

Quizizz[15] - це онлайн-платформа для створення та проведення інтерактивних вікторин і тестів. Основна мета сайту полягає в забезпеченні вчителям інструменту для створення цікавих та ефективних занять для учнів.

Основні функції системи "Quizizz" можуть включати:

- Створення тестів. Вчителі можуть створювати свої власні тести з питань різного типу, таких як багатовибірні, відкриті питання тощо;
- Проведення вікторин. Учителі можуть використовувати створені тести для проведення інтерактивних вікторин на уроках або вдома;
- Участь у готових тестах. Користувачі можуть брати участь у готових вікторинах та тестах, які були створені іншими користувачами;

- Оцінювання та статистика. Система надає звіти та статистику про відповіді учнів, що дозволяє вчителям відстежувати прогрес та розуміти, на яких темах потрібно зосереджувати увагу.

Аналізуючи систему "Quizizz", можна виділити наступні переваги і недоліки:

Переваги:

- Інтерактивність та зацікавленість учнів через групову конкуренцію;
- Зручний інтерфейс для створення та використання тестів;
- Можливість доступу до готових тестів від інших вчителів;
- Звітність та статистика для вчителів.

Недоліки:

- Обмежена можливість налаштування тестового середовища;
- Потреба у Інтернет-підключенні для користувачів;
- Залежність від інфраструктури платформи;
- Деякі функції можуть бути обмеженими в безкоштовній версії.

На рис. 1.1 відображена домашня сторінка веб-сервісу “Quizizz”, який пропонує широкий спектр інструментів для створення та проведення інтерактивних занять. Платформа дозволяє швидко знаходити або створювати власні навчальні матеріали, а також отримувати детальну статистику про результати учнів.

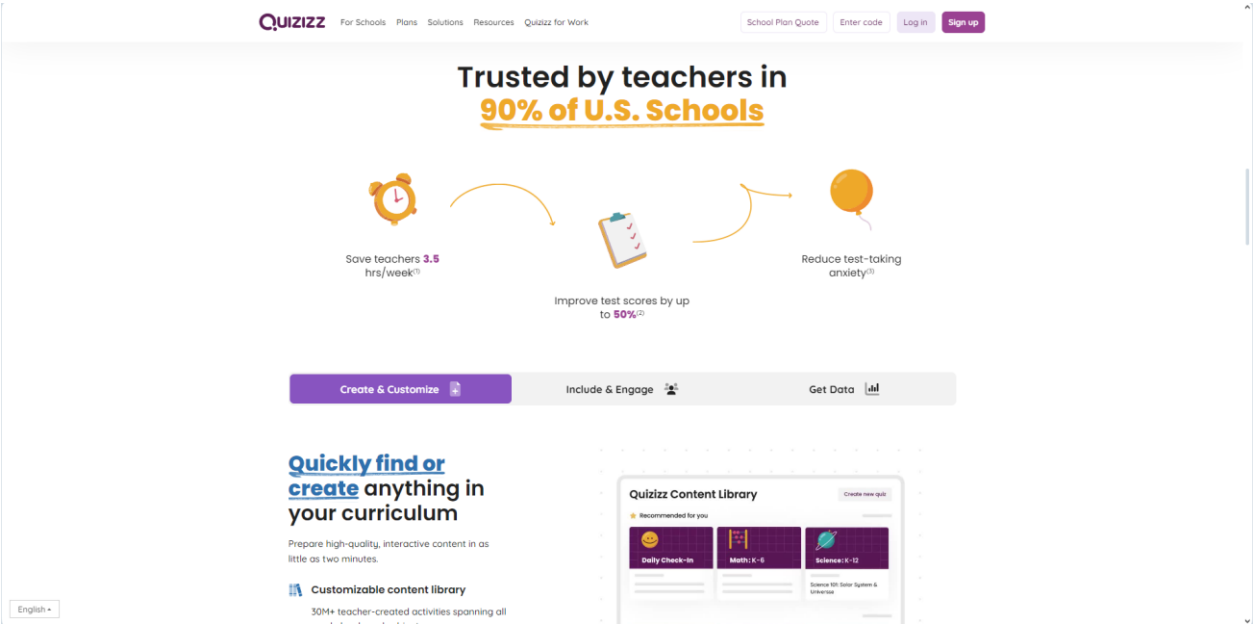

Рисунок 1.1 - Головна сторінка сервісу «Quizizz»

На рис. 1.2 зображена сторінка з актуальними та минулими темами для самонавчання з різних предметів:

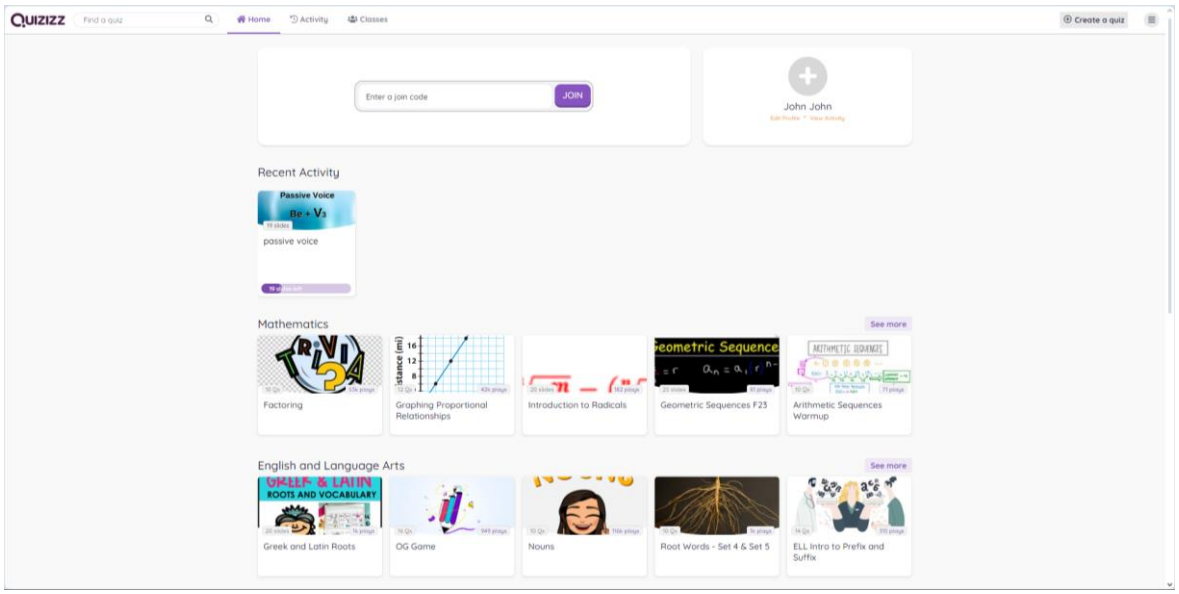

Рисунок 1.2 – Сторінка з темами для самонавчання

Сторінка огляду класу з діаграмами оцінок, виконанням плану, питаннями, списком учнів зображена на рис. 1.3:

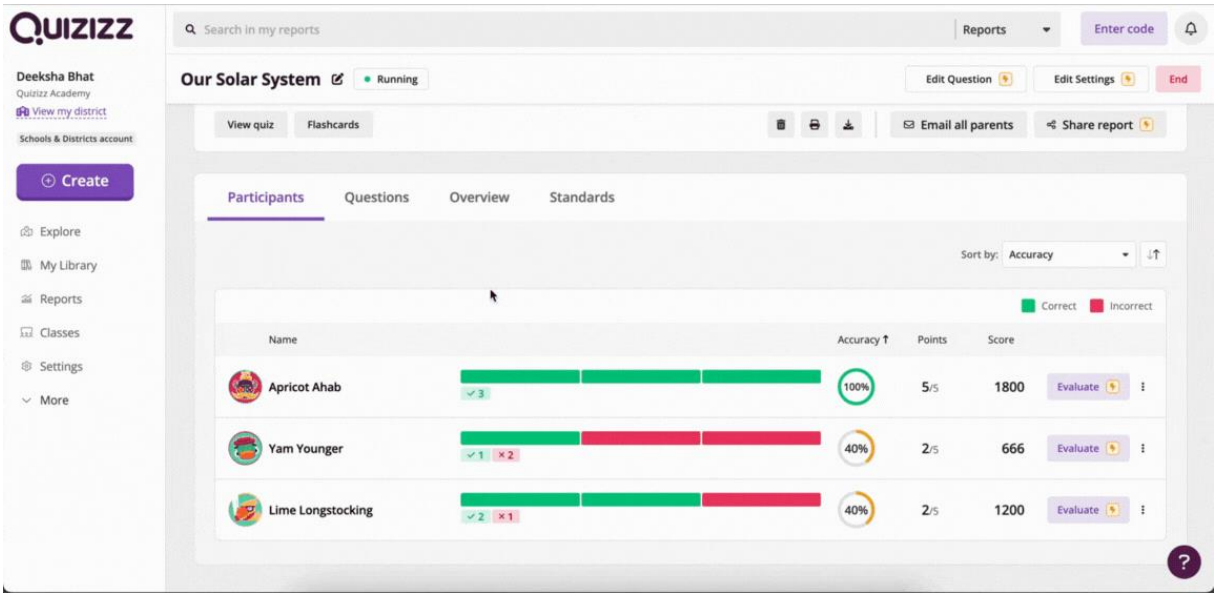

Рисунок 1.3 – Сторінка огляду класу

Classcraft[16] - це онлайн-платформа, яка створена для вчителів з метою покращення процесу навчання через використання ігрових елементів та гейміфікації в класі. Основна ідея полягає в тому, щоб залучити учнів до навчання, створюючи відчуття спільноти та зацікавленості.

Основні функції системи "Classcraft" можуть включати:

- Гейміфікація. Платформа пропонує інструменти для створення гри в класі, де учні можуть заробляти бали та нагороди за свої досягнення та активність;
- Створення персональних персонажів. Учні можуть створювати власних персонажів, а вчителі можуть використовувати це для взаємодії зі студентами в грі;
- Моніторинг поведінки та навчального прогресу. Вчителі можуть відслідковувати поведінку та академічний прогрес учнів через спеціальні функції платформи;
- Спільнота класу. Classcraft створює віртуальну спільноту, де учні можуть взаємодіяти один з одним та вчителями;
- Співпраця та командна робота. Гра стимулює співпрацю та командну роботу, що може сприяти соціальному взаємодії в класі;

- Батьківський Звіт. Платформа може надавати звіти батькам про активність та прогрес їхніх дітей;
- Інтеграція з Освітніми Платформами. Classcraft може інтегруватися з іншими платформами для полегшення використання у навчальному процесі.

Аналізуючи систему "Classcraft", можна виділити наступні переваги і недоліки:

Переваги:

- Залучення учнів до навчання через ігрові елементи;
- Покращення дисципліни та мотивації учнів;
- Відстеження навчального прогресу та поведінки;
- Наявність мобільного додатку.

Недоліки:

- Може вимагати часу на вивчення та налаштування платформи;
- Залежність від доступу до Інтернету та технічних засобів;
- Можливі технічні або логістичні труднощі.

На рис. 1.4 відображена домашня сторінка веб-сервісу “Classcraft”, який пропонує. На ній розміщена основна інформація про сервіс та функціональній кнопки.

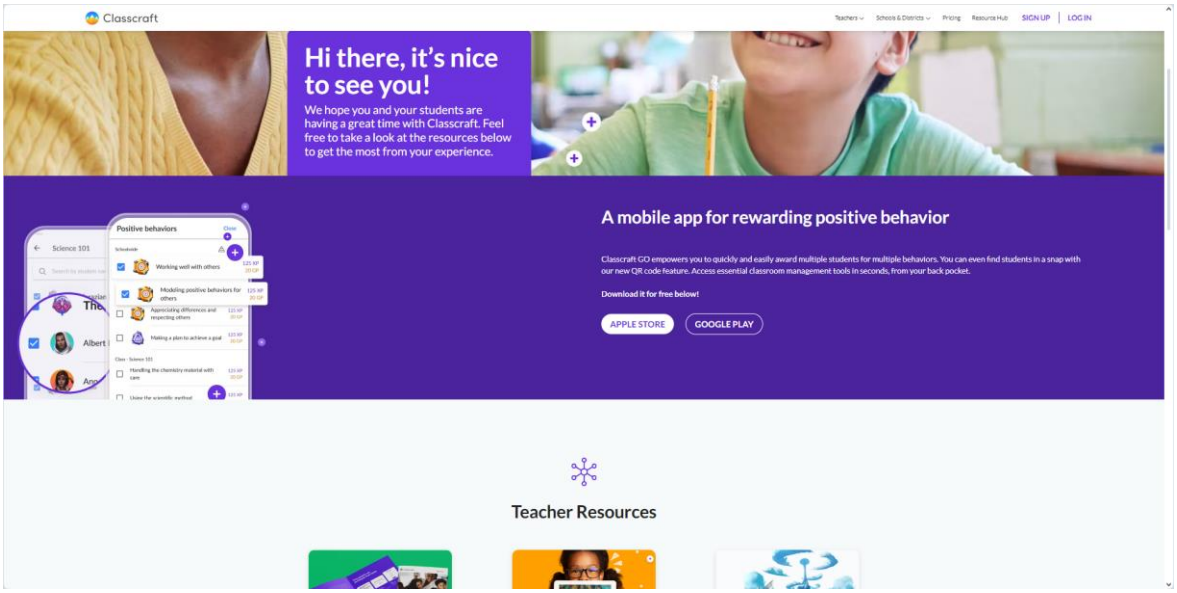

Рисунок 1.4 - Головна сторінка сервісу "Classcraft"

Сторінка з актуальними та найбільш популярними квестами зображена на рис. 1.5.

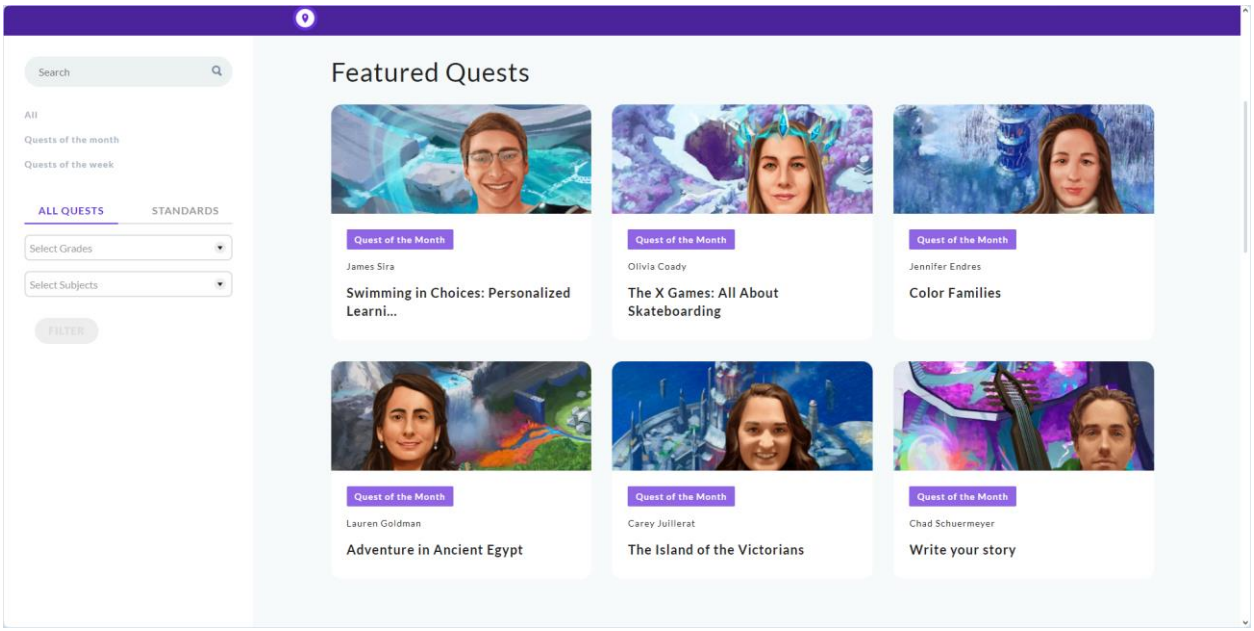

Рисунок 1.5 – Сторінка з квестами

Сторінка огляду класу, де розміщені всі учні з їх статусами(здоров’я, навчання), оцінками та рівнями зображена на рис. 1.6. На цій сторінці можна керувати як і самими учнями, цілим класом, так і їх програмою.

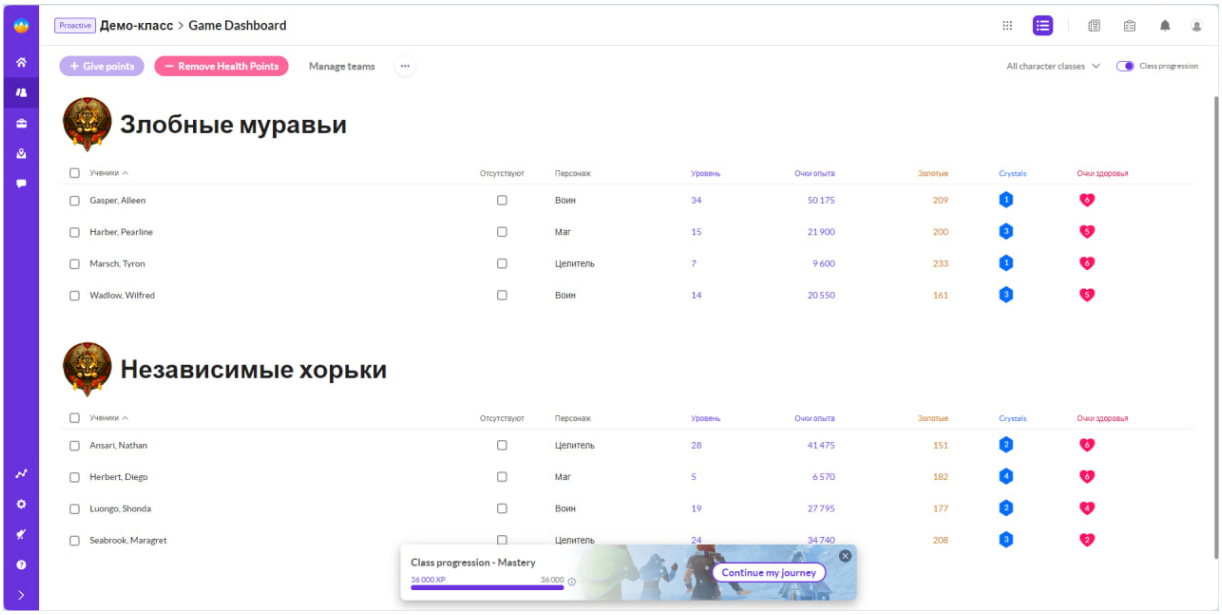

Рисунок 1.6 – Сторінка огляду класу

На основі нашого аналізу можна сказати, що автоматизація інтеграції ігрових елементів у навчальні програми є важливою складовою конкурентоспроможності. Для того, щоб залишатися конкурентоспроможними на ринку, необхідно впроваджувати передові технології, покращувати ефективність та постійно вдосконалювати вже реалізовані процеси.

Аналіз конкурентів дає цінні висновки та рекомендації для подальшого розвитку власної освітньої системи. Ці знання можна використати для вдосконалення різних процесів, впровадження нових технологій та пошуку конкурентних переваг у сфері освіти.

2 ВИБІР АРХІТЕКТУРИ ТА ТЕХНОЛОГІЙ ДЛЯ ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Постановка задачі та визначення основних функцій системи

Визначення завдань є критичним етапом у розробці будь-якого проекту. Цей процес встановлює ключові цілі, вимоги та обмеження проекту, створюючи основу для подальших кроків. Правильна постановка завдань сприяє уточненню вимог та очікувань, а також визначає основні критерії успіху для майбутньої розробки та оцінки її результатів. Цей процес дозволяє команді проекту зосередитися на важливих аспектах, забезпечуючи узгодженість між всіма учасниками та ефективне використання ресурсів для досягнення поставлених цілей[17].

В сучасних умовах дистанційного навчання, коли студенти і викладачі взаємодіють через інтернет, процес організації команд для групових проєктів часто стає складним і непродуктивним. Відсутність автоматизованих інструментів для формування команд ускладнює роботу викладачів, особливо якщо мова йде про велику кількість учнів з різним рівнем підготовки, інтересами та знаннями.

Однак проблема не обмежується лише необхідністю створення команд. Система, що займається тільки формуванням команд для одного проєкту, не є життєздатною в довгостроковій перспективі. Вона потребує розширення функціональності, щоб обслуговувати не тільки один процес створення команд, але й цілу низку завдань, пов'язаних із дистанційним навчанням.

Метою даного проєкту є розробка серверної частини клієнт-серверної системи, яка забезпечує автоматизацію процесу формування команд учнів для спільних навчальних проєктів. Проте для забезпечення стабільної та повноцінної роботи цієї системи, необхідно впровадити більш загальну інфраструктуру, яка підтримуватиме усі етапи онлайн-навчання. Таким чином, система стане інформаційною онлайн-платформою для навчання, в рамках якої не лише будуть формуватися команди, але й буде забезпечено управління навчальними

матеріалами, взаємодію між студентами та викладачами, оцінювання проєктів тощо.

Щоб забезпечити повноцінну роботу клієнт-серверної системи для автоматизації навчання, серверна частина має виконувати низку важливих функцій:

1. Управління користувачами:
    - Реєстрація та аутентифікація. Кожен користувач системи (учень, викладач, адміністратор) матиме свій обліковий запис, що забезпечить доступ до ресурсів системи відповідно до його ролі;
    - Авторизація. Після аутентифікації система надаватиме доступ до функцій відповідно до рівня доступу. Наприклад, учні зможуть переглядати матеріали курсів і брати участь у командах, тоді як викладачі зможуть керувати курсами та командами.
2. Автоматизація формування команд:
    - Алгоритмічне формування команд. Система автоматично створюватиме команди учнів на основі таких критеріїв, як рівень знань, інтереси, вільний час, участь у попередніх проєктах;
    - Збалансованість команд. Алгоритм повинен враховувати різні аспекти, щоб сформувати команди, де рівень учнів є оптимально збалансованим для спільної продуктивної роботи.
3. Управління курсами та проєктами:
    - Створення курсів. Викладачі зможуть створювати курси, додавати навчальні матеріали та завдання;
    - Управління проєктами. Кожен курс може мати один або кілька проєктів, для яких автоматично формуються команди учнів;
    - Моніторинг успішності. Система дозволяє відслідковувати успіхи кожної команди в рамках проєкту, а також надає інструменти для оцінювання.

4. Управління навчальними матеріалами:
    - Завантаження та зберігання матеріалів. Система повинна забезпечувати зручний інтерфейс для зберігання і доступу до навчальних матеріалів (відео, документи, презентації);
    - Доступ до матеріалів. Викладачі можуть надавати доступ до навчальних матеріалів відповідно до етапів навчання або досягнень учнів.
5. Обмін інформацією:
    - Чат та повідомлення. Для ефективної взаємодії між учасниками курсу та командами потрібні інструменти комунікації (чат, внутрішні повідомлення);
    - Повідомлення та нагадування. Система буде надсилати автоматичні сповіщення про нові завдання, зміни у курсі, дедлайни тощо.
6. Безпека та збереження даних:
    - Аутентифікація за допомогою JWT (JSON Web Tokens). Забезпечить безпечну авторизацію користувачів у системі;
    - Шифрування даних. Всі дані, що передаються між клієнтом і сервером, будуть зашифровані для запобігання несанкціонованого доступу;
    - Захист від атак. Система повинна бути захищена від поширених видів атак, таких як SQL-ін'єкції, XSS, CSRF та інші.

Хоча основним завданням є автоматизація процесу формування команд, система не може існувати лише для одного проєкту. Щоб система була життєздатною, вона повинна бути здатною обслуговувати кілька курсів, проектів і команд одночасно[18].

Це означає, що серверна частина має бути спроектована з урахуванням наступних розширень, таких як:

- Додавання нових функцій для підтримки взаємодії учнів та викладачів (форум, система оцінювання, статистика успіхів);
- Розширення функціоналу для інтеграції із зовнішніми сервісами (наприклад, системи відеоконференцій для онлайн-занять);

- Підтримка додаткових ролей користувачів (наприклад, модератори, координатори).

Таким чином, кінцевий продукт не лише полегшить процес організації навчальних проєктів, але й стане багатофункціональною онлайн-системою для управління навчальним процесом загалом. Це дозволить автоматизувати різні аспекти навчання, зокрема, комунікацію, управління курсами та оцінювання успішності учнів.

Визначимо основні функції, які буде виконувати розроблена система:

- реєстрація на сайті;
- персональний кабінет користувача з можливістю зміни персональних даних;
- генерація команди;
- сторінка завдань класу з можливістю створення, редагування, видалення, оновлення завдань в процесі навчання.

В процесі виконання вказаних завдань маємо намір розробити інноваційний проект, який допоможе автоматизувати та оптимізувати роботу системи навчання в онлайн середовищі. Він сприятиме зручності та задоволенню користувачів, дозволяючи їм зручно використовувати інтерактивні ігрові елементи, вільно взаємодіяти з ними, а також отримувати корисні рекомендації та статистичні дані для оптимального процесу навчання.

Визначившися із постановкою задачі, складемо UML-діаграму варіантів використання (UML Use Case діаграма).

Ця діаграма описує, як різні актори (користувачі або інші системи) взаємодіють з системою, щоб виконати певні завдання або досягти конкретних цілей. Кожен варіант використання, представлений у вигляді овала, позначає конкретний сценарій або функцію, яку система надає актору. Актори з'єднуються з варіантами використання за допомогою ліній або стрілок, що відображає взаємодію між ними.

На рис. 2.1, зображена UML Use Case діаграма взаємодії користувача з системою навчання онлайн:

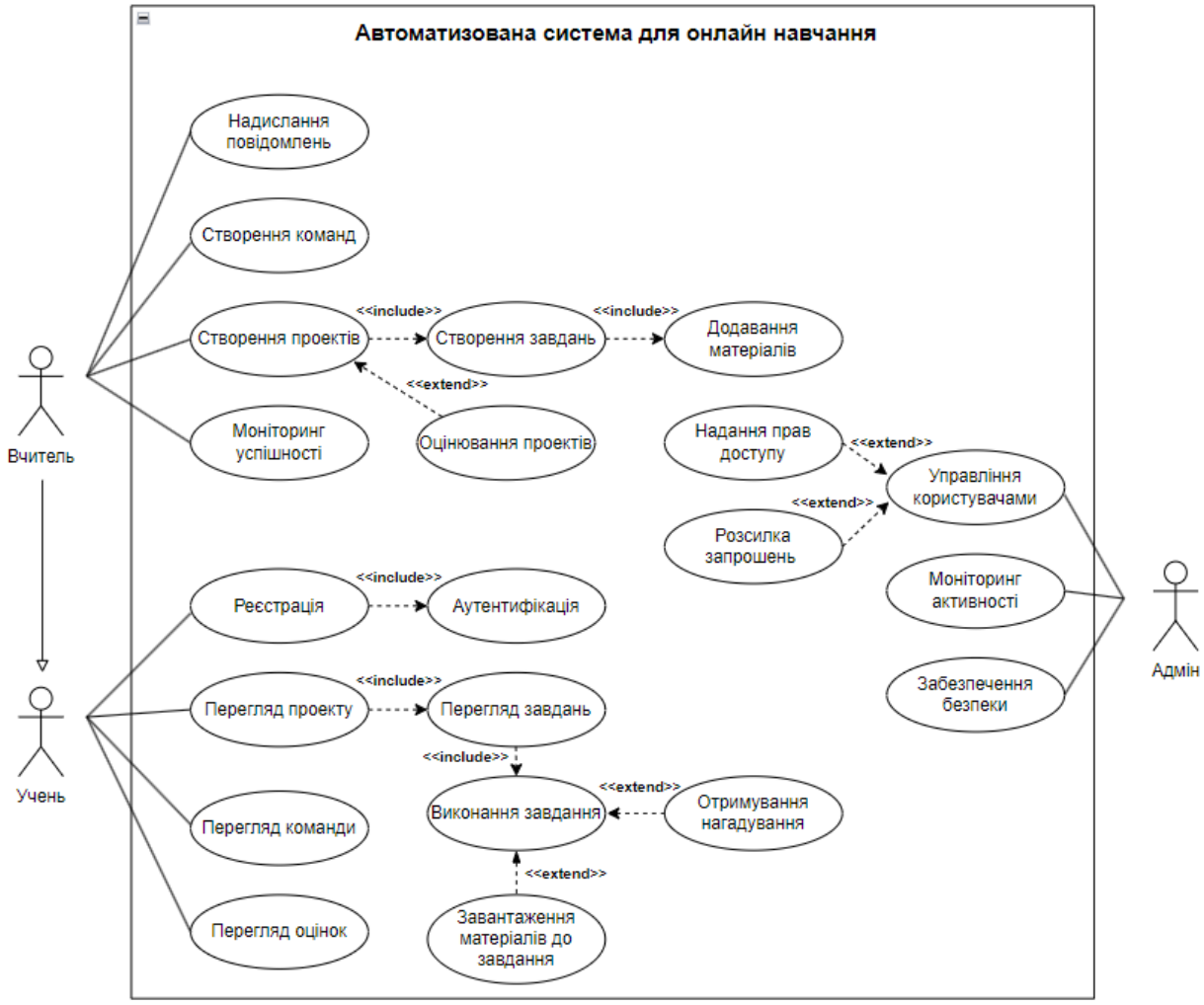

Рисунок 2.1 – Use Case діаграма взаємодії користувача з системою

Дана діаграма показує функціональні можливості автоматизованої системи для онлайн навчання, в якій беруть участь три основних актора: вчитель, учень та адміністратор. Кожен актор має доступ до своїх прецедентів і як пов’язані між собою функції через include і extend, забезпечуючи послідовність та логіку в системі.

2.2 Проектування архітектури системи

Клієнт-серверна система[19] для автоматизації процесу формування команд учнів повинна мати гнучку та масштабовану архітектуру, що забезпечить стабільну роботу, надійне зберігання даних і можливість розширення функціоналу. Система

повинна підтримувати кілька ролей користувачів (учень, викладач, адміністратор), обробляти великий обсяг даних про учнів, курси, команди та забезпечувати інтерактивну взаємодію між учасниками процесу.

Для реалізації функціоналу системи обрана трирівнева архітектура, яка розділяє систему на окремі компоненти, кожен з яких виконує свою функцію.

Основними компонентами є:

1. Клієнтський шар (Presentation Layer);
2. Шар бізнес-логіки (Business Logic Layer);
3. Шар зберігання даних (Data Layer).

На рис. 2.2 зображена архітектурна схема системи:

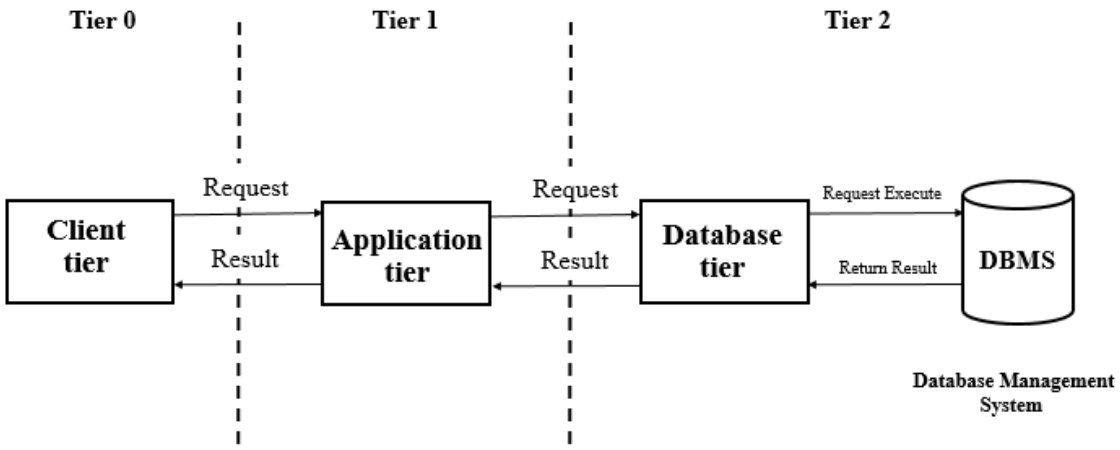

Рисунок 2.2 – Архітектурна схема системи

Клієнтський шар.

Цей шар взаємодіє безпосередньо з користувачами системи через графічний інтерфейс, надаючи їм можливість виконувати певні дії. Клієнт надсилає запити до сервера через інтерфейс API і отримує відповідні дані. Клієнт може бути реалізований як веб-інтерфейс (веб-додаток), мобільний додаток або навіть інші зовнішні системи, які можуть інтегруватися через API.

Основні функції:

- Відображення даних користувачу (наприклад, список доступних курсів, інформація про команди);

- Надсилання запитів до серверу (реєстрація, входження до системи, створення курсів, управління командами);
- Обробка отриманих відповідей та оновлення інтерфейсу відповідно до результатів дій.

Шар бізнес-логіки.

Цей шар відповідає за реалізацію всіх функціональних можливостей системи. Він обробляє запити, отримані від клієнтського шару, виконує необхідну логіку та взаємодіє з шаром зберігання даних для отримання або запису інформації.

Основні функції:

- Формування команд. На основі критеріїв, таких як рівень знань учнів, їхні інтереси, попередній досвід тощо, система автоматично формує команди;
- Управління курсами та проектами. Викладачі можуть створювати курси та керувати ними, додавати завдання та формувати проекти для учнів;
- Управління користувачами. Система дозволяє реєструвати нових користувачів, надавати їм відповідні права доступу та керувати їхніми профілями;
- Забезпечення безпеки. Система перевіряє автентичність запитів користувачів і контролює доступ до певних функцій відповідно до ролей користувачів.

Шар зберігання даних.

Цей шар забезпечує надійне зберігання інформації та ефективну роботу з великими обсягами даних. Дані зберігаються в базі даних, яка включає такі основні сутності:

- Користувачі (учні, викладачі, адміністратори). Інформація про реєстрацію, профілі, ролі та права доступу;
- Курси та проекти. Дані про створені курси, навчальні матеріали, завдання та прогрес учнів;
- Команди. Інформація про сформовані команди, їхній склад, успішність та поточний стан;

- Результати навчання. Статистичні дані про виконання завдань, оцінки та досягнення учнів.

Шар даних відповідає за:

- Запити на читання. Надання даних за запитами бізнес-логіки, наприклад, отримання списку курсів або складу команд;
- Запити на запис. Збереження нових даних, таких як нові курси, команди або оновлення інформації про користувачів;
- Обробку складних запитів. Виконання операцій, які потребують складних обчислень або агрегації даних (наприклад, підрахунок успішності учнів за результатами курсу).

Взаємодія між компонентами.

Взаємодія між компонентами архітектури відбувається за допомогою чітко визначених інтерфейсів і протоколів. Основний потік взаємодії включає наступні етапи:

1. Запит від клієнта. Користувач через інтерфейс (веб-додаток або мобільний додаток) надсилає запит до сервера. Наприклад, запит на формування нової команди;
2. Обробка запиту на рівні бізнес-логіки. Сервер обробляє отриманий запит, застосовуючи бізнес-логіку для визначення дій, які необхідно виконати. Якщо потрібні дані з бази даних, система звертається до шару зберігання даних;
3. Отримання/запис даних. Шар зберігання даних виконує запити на отримання або запис даних. Наприклад, зберігає нову команду або надає інформацію про користувача;
4. Формування відповіді. Після виконання всіх дій, сервер формує відповідь і відправляє її клієнту;
5. Оновлення інтерфейсу. Клієнт отримує відповідь від сервера і оновлює користувацький інтерфейс на основі результатів.

2.3 Аналіз та обґрунтування технологій розробки

Розробка клієнт-серверної системи для автоматизації процесу формування команд учнів вимагає використання сучасних технологій, які забезпечують надійність, масштабованість та ефективність функціонування системи. Для кожного компонента архітектури необхідно вибрати відповідні технологічні засоби, що відповідають вимогам проекту[20]. Нижче наводиться аналіз та обґрунтування вибору ключових технологій для розробки серверної частини системи.

Мова програмування.

Мова програмування[21] є визначальним фактором у створенні надійної серверної архітектури веб-додатків. Правильний вибір мови дозволяє підвищити продуктивність, забезпечити зручність підтримки та оптимізувати розвиток проекту в довгостроковій перспективі. Для цього проекту, враховуючи актуальні тенденції та вимоги до сучасних веб-рішень, найбільш підходящими є JavaScript/TypeScript (через платформу Node.js) і Python. Кожна з цих мов має потужні інструменти, великий набір бібліотек і унікальні переваги, які підвищують ефективність розробки та підтримки коду.

Node.js (JavaScript/TypeScript).

Node.js[22] є однією з найбільш затребуваних платформ для розробки серверної частини веб-додатків і мікросервісів. Платформа базується на движку V8 від Google, що забезпечує високу швидкодію JavaScript-коду. Це дозволяє працювати з асинхронними запитами, які обробляються в однопоточному режимі з використанням подій, що значно підвищує продуктивність і масштабованість систем. Основні переваги Node.js включають:

- Швидкодія. Архітектура Node.js дозволяє обробляти одночасно десятки тисяч запитів без значних затримок. Асинхронна модель обробки запитів зменшує навантаження на сервер, оптимізує роботу з мережевими ресурсами та забезпечує швидкий час відгуку;

- Універсальність. JavaScript (та його строгий надбудовний варіант TypeScript) підтримується як на клієнтській, так і на серверній стороні. Це забезпечує повну інтеграцію логіки додатка, скорочуючи розрив між фронтендом і бекендом. Така єдність коду дозволяє зменшити кількість контекстних перемикань між мовами та забезпечує більш послідовну підтримку проєкту;
- Модульність та розширюваність. Одним із сильних аспектів Node.js є екосистема npm, що містить мільйони готових до використання модулів та бібліотек. Це значно скорочує час на розробку, оскільки більшість необхідної функціональності вже реалізована. Модульний підхід дозволяє легко підключати і видаляти різні компоненти в залежності від потреб проекту;
- TypeScript для надійності. TypeScript як надбудова над JavaScript вводить строгий контроль типів і робить код більш структурованим, що особливо важливо для великих і складних проектів із численними залежностями та компонентами. TypeScript сприяє кращій читабельності коду, спрощує автодоповнення та забезпечує ефективне виявлення помилок на ранніх стадіях розробки.

Python.

Python, відомий своїм простим і інтуїтивно зрозумілим синтаксисом, є потужним інструментом для розробки серверної логіки, особливо у випадках, коли проєкт потребує складних обчислень, обробки даних або автоматизації. Він активно використовується у різних сферах, включаючи розробку алгоритмів, машинне навчання та аналіз даних. Основні переваги Python:

- Простота і читабельність. Один з найбільших плюсів Python — це зрозумілий, інтуїтивний синтаксис, який робить код легким для розуміння і підтримки. Це знижує ймовірність помилок, сприяє швидкому освоєнню для нових учасників команди та полегшує рев'ю коду;
- Широкий набір бібліотек і фреймворків. Python має потужну екосистему, включаючи популярні фреймворки, такі як Django і Flask, які прискорюють створення REST API та обробку HTTP-запитів. Існує також великий вибір бібліотек для обробки даних (Pandas, NumPy), що забезпечують ефективне

виконання складних операцій і дозволяють швидко створювати додатки для аналізу та обробки даних;

- Підтримка обчислювальних і наукових завдань. Python ідеально підходить для автоматизації та обчислень. Використання бібліотек, таких як SciPy, TensorFlow або Keras, забезпечує реалізацію складних алгоритмів і моделей машинного навчання, що дозволяє ефективно працювати з великим обсягом даних, навчанням моделей та розробкою AI-інструментів;
- Активна підтримка та розширення. Python має одну з найбільших спільнот розробників, що означає постійну підтримку і розширення можливостей мови. Це забезпечує стабільність та доступ до великої кількості ресурсів, документації та навчальних матеріалів.

Таким чином, при проектуванні серверної частини сучасного веб-додатка була обрана мова програмування Python.

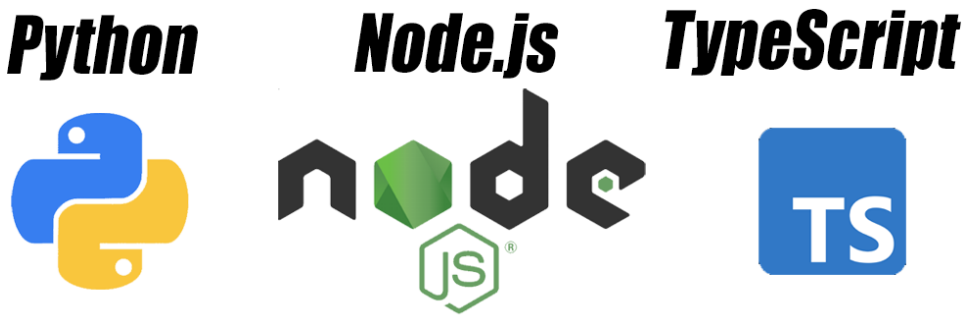

Рисунок 2.3 - Мови програмування

Вибір фреймворку[23] для серверної частини веб-додатка є важливим етапом, оскільки від нього залежить масштабованість, продуктивність і зручність роботи з API. При створенні ефективної бекенд-архітектури слід враховувати специфічні вимоги проекту, зокрема необхідність швидкої обробки запитів, безпеки, підтримки складних баз даних та можливість додавання нових функцій. У цьому контексті для даного проекту найкраще підійдуть два фреймворки — Express.js для Node.js і Django для Python. Кожен із них має унікальні переваги, які роблять їх зручними для побудови масштабованих, безпечних і продуктивних серверних рішень.

Express.js.

Є легким та гнучким веб-фреймворком для Node.js, орієнтованим на побудову RESTful API. Завдяки своїй мінімалістичній структурі він забезпечує швидкий старт розробки і дозволяє легко адаптуватися до зростаючих потреб проєкту. Основні переваги Express.js включають:

- Швидкість і легкість у використанні. Express.js має простий, інтуїтивно зрозумілий інтерфейс і мінімалістичний набір функцій, що дозволяє швидко почати роботу. Розробники можуть швидко налаштувати сервер, визначити маршрути та обробники запитів, що спрощує процес розробки і скорочує час виведення продукту на ринок;
- Масштабованість. Завдяки підтримці модульності Express.js добре підходить для створення великих веб-додатків, у яких потрібно поступово додавати нові функції. Фреймворк дозволяє легко інтегрувати численні сторонні модулі, зокрема для роботи з базами даних, автентифікації, авторизації та інших завдань, що розширює можливості програми і дозволяє швидко масштабувати додаток у разі зростання навантаження;
- Широка спільнота. Express.js є одним із найпопулярніших фреймворків для Node.js, що означає наявність великої кількості готових рішень і прикладів для найрізноманітніших задач. Спільнота розробників надає підтримку і швидко реагує на проблеми, що виникають, що значно полегшує розробку, оновлення і підтримку коду.

Express.js підходить для проектів, де важлива швидкість розробки, мінімалістичність коду і гнучкість у додаванні нових функцій. Це оптимальний вибір для веб-додатків, що потребують швидкого відгуку та мають велику кількість одночасних запитів.

Django.

Django є повноцінним веб-фреймворком для Python, що надає безліч готових компонентів і дозволяє швидко створювати складні додатки з високим рівнем безпеки та ефективності. Він побудований за принципом «все включено», що

означає наявність багатьох вбудованих рішень для роботи з користувачами, сесіями, безпекою та базами даних. Основні переваги Django:

- Швидка розробка. Django містить вбудовані модулі для основних функцій, зокрема автентифікації користувачів, управління сесіями, роботи з ORM (Object-Relational Mapping) для взаємодії з базами даних, а також компонентами для побудови REST API. Завдяки цьому розробники можуть зосередитися на логіці програми, а не на технічних деталях, що суттєво пришвидшує процес розробки;
- Безпека. Django відомий своєю увагою до безпеки, що робить його надійним вибором для проєктів, у яких потрібен високий рівень захисту даних. Фреймворк включає вбудовані механізми захисту від поширених атак, таких як XSS (міжсайтове виконання сценаріїв), CSRF (міжсайтовий запит підробки), SQL-ін'єкції та Clickjacking. Це дозволяє розробникам автоматично впроваджувати надійні рішення безпеки і мінімізувати ризик втрати даних;
- Висока продуктивність. Django оптимізований для обробки великих обсягів даних, що робить його підходящим для клієнт-серверних систем з великою кількістю користувачів. Його ORM дозволяє працювати з базами даних високого рівня, а кешування і інші інструменти оптимізації забезпечують швидкий відгук сервера навіть при великому навантаженні;
- Активна спільнота та документація. Django підтримується широкою спільнотою розробників і має детальну документацію, що спрощує навчання та пошук інформації для вирішення специфічних задач. Крім того, фреймворк постійно оновлюється та вдосконалюється, що гарантує його актуальність і сумісність з останніми стандартами веб-розробки.

Flask.

Flask — це легкий та мінімалістичний веб-фреймворк для Python, розроблений для створення простих і масштабованих веб-додатків. Його філософія полягає у забезпеченні максимального контролю для розробників із мінімальними

обмеженнями. Flask добре підходить для проєктів, що потребують гнучкості, і дозволяє інтегрувати сторонні бібліотеки за потреби. Основні переваги Flask:

- Простота і легкість. Flask надає мінімалістичний каркас для розробки, що дозволяє розробникам почати створювати додатки без зайвих складнощів. Завдяки його простій архітектурі можна легко налаштовувати додаток відповідно до специфічних вимог;
- Гнучкість і модульність. Фреймворк надає можливість вибору компонентів і модулів, необхідних для додатка, не нав’язуючи жодних стандартів. Це забезпечує гнучкість у побудові додатків різної складності, від простих API до складних систем;
- Підтримка розширень. Flask має безліч розширень, які додають підтримку таких функцій, як автентифікація, управління базами даних, обробка форм та інше. Розширення легко інтегруються, дозволяючи розширювати функціональність програми без зайвих зусиль;
- Масштабованість. Flask дозволяє створювати як невеликі додатки, так і масштабовані проєкти. Розробники можуть почати з простого додатка і поступово додавати нові функції та компоненти, не змінюючи основну структуру;
- Активна спільнота та документація. Flask має широку спільноту користувачів, яка надає підтримку у вигляді готових рішень, порад і прикладів. Його документація проста та доступна, що полегшує навчання та роботу з фреймворком.

Всі три фреймворки надають потужні інструменти для створення продуктивних веб-додатків, проте їхні можливості відрізняються залежно від потреб проекту, тому в роботі ми будемо використовувати Flask.

Рисунок 2.4 - Фреймворки

Для надійного та ефективного зберігання великого обсягу даних, включаючи інформацію про користувачів, курси, команди та результати навчання, важливим є вибір правильної бази даних[24]. Враховуючи особливості проекту, зокрема потребу в швидкому доступі до даних, цілісності та гнучкості в роботі з різними типами даних, найкращими кандидатами для цього проекту є реляційна база даних MariaDB та NoSQL база даних MongoDB. Кожна з них пропонує унікальні можливості для управління даними, які можуть відповідати вимогам системи.

MariaDB(Реляційна база даних)

MariaDB — це потужна і швидка система керування базами даних із відкритим кодом, яка є форком MySQL. Вона розроблена для забезпечення високої продуктивності, безпеки та сумісності з сучасними вимогами до зберігання та обробки даних. Основні переваги MariaDB:

- Висока продуктивність. MariaDB оптимізована для роботи з великими обсягами даних і високими навантаженнями. Завдяки покращеним механізмам індексації, кешування і обробки запитів вона забезпечує швидкий доступ до даних навіть у складних системах;
- Сумісність із MySQL. MariaDB підтримує синтаксис і функції MySQL, що дозволяє безболісно мігрувати з MySQL на MariaDB або використовувати їх паралельно. Це також гарантує легкість інтеграції з додатками, розробленими для MySQL;
- Безпека і надійність. У MariaDB реалізовано широкий набір інструментів для захисту даних, таких як шифрування таблиць і з'єднань, а також покращені

механізми автентифікації. Це робить її надійним вибором для критичних систем, які потребують високого рівня безпеки;

- Розширюваність і модульність. Завдяки підтримці плагінів і модулів MariaDB легко налаштовується під конкретні потреби проєкту. Наприклад, вона підтримує декілька механізмів зберігання даних, включаючи InnoDB, Aria та MyRocks, що дозволяє оптимізувати роботу бази під різні типи навантаження;
- Активна спільнота і розвиток. Проєкт MariaDB підтримується великою міжнародною спільнотою, яка постійно оновлює і вдосконалює систему. Це гарантує її відповідність сучасним стандартам і сумісність із новими технологіями.

MariaDB підходить для широкого спектра проєктів — від невеликих веб-додатків до великих корпоративних систем, які потребують масштабованої та надійної бази даних. Її відкритий код і активний розвиток роблять її доступною та перспективною альтернативою іншим системам керування базами даних.

MongoDB (NoSQL база даних).

MongoDB є NoSQL базою даних, яка забезпечує високу гнучкість і швидкість доступу до даних, що робить її зручним варіантом для систем, де структура даних може бути змінною або де важливий швидкий доступ до великих масивів інформації. MongoDB зберігає дані у форматі документів, що забезпечує просту роботу з неструктурованими або динамічними даними, які важко представити у традиційних таблицях. Основні переваги MongoDB:

- Гнучкість. MongoDB не вимагає жорсткої схеми даних, що дозволяє зберігати інформацію у динамічному форматі. Це забезпечує швидке оновлення даних без необхідності змінювати структуру бази при зміні вимог. Такий підхід дозволяє зберігати різні типи даних в одному полі, що полегшує роботу зі складними даними і скорочує час на їх підготовку;
- Швидкість обробки даних. MongoDB оптимізована для швидкого доступу до великих масивів даних, що є особливо корисним для масштабованих систем, які повинні обробляти великий потік запитів в реальному часі. Завдяки

горизонтальному масштабуванню MongoDB забезпечує ефективне оброблення даних, що підходить для систем із великою кількістю одночасних користувачів.

MongoDB є зручним варіантом для проектів, де обробка неструктурованих даних є пріоритетом, або в системах, де дані можуть змінювати свою структуру з часом. Вона підходить для додатків, які вимагають високої гнучкості і адаптивності, особливо при роботі з великими обсягами динамічних даних.

Обидві бази даних підходять для роботи з великими обсягами даних, проте кожна з них має свої особливості. MariaDB є оптимальним рішенням для системи.

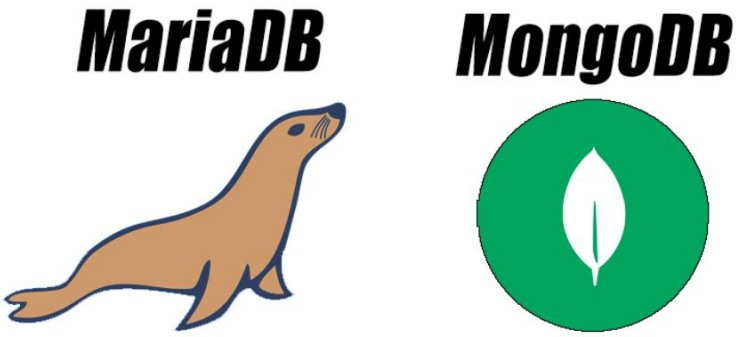

Рисунок 2.5 - Бази даних

Для забезпечення ефективної та надійної взаємодії між клієнтською та серверною частинами веб-додатка обирається протокол HTTP(S). Цей протокол є загальноприйнятим стандартом для веб-додатків і гарантує безпечну та стабільну передачу даних через мережу. Використання HTTP(S) дозволяє передавати дані у зручному текстовому форматі, що підтримується більшістю клієнтських платформ, забезпечуючи сумісність і легкість інтеграції з різними типами пристроїв і додатків.

REST API (Representational State Transfer API).

Для побудови інтерфейсу між клієнтом і сервером використовується архітектурний стиль REST API, який є одним із найпоширеніших і гнучких підходів для створення веб-сервісів. REST API базується на простих, стандартизованих HTTP методах, таких як GET, POST, PUT, DELETE, що

дозволяє легко керувати ресурсами на сервері. Завдяки використанню цих методів REST API є інтуїтивно зрозумілим для розробників і забезпечує високу продуктивність, простоту інтеграції та гнучкість. REST API забезпечує такі переваги:

- Простота у використанні та розгортанні. REST API легко налаштовується і забезпечує простий доступ до ресурсів через уніфіковані URI-адреси. Завдяки цьому розробники можуть легко інтегрувати його з клієнтськими додатками (як веб, так і мобільними), не витрачаючи багато часу на додаткові налаштування та конфігурацію. REST API дозволяє швидко створювати прості та зрозумілі структури запитів і відповіді, що спрощує роботу як для бекенд-розробників, так і для фронтенд-розробників;
- Масштабованість. Завдяки стандартизованим методам взаємодії REST API здатний обробляти велику кількість одночасних запитів, що важливо для масштабованих веб-додатків. Розробка веб-сервісу, що підтримує масштабованість, забезпечує ефективну роботу навіть при зростанні кількості користувачів або навантаження на сервер. REST API дозволяє швидко адаптувати систему до нових вимог або додавати нові ресурси без значних змін в архітектурі, що робить його оптимальним вибором для розподілених систем;
- Відокремлення клієнта та сервера. REST API забезпечує повне розділення клієнтської і серверної частини, що дозволяє легко підтримувати різні клієнтські платформи (наприклад, веб-браузери, мобільні додатки, інші сервери). Завдяки цьому серверна частина може обробляти запити незалежно від клієнтського додатка, що робить систему більш гнучкою та легко адаптованою до оновлень на будь-якій зі сторін. Це також полегшує підтримку й оновлення, оскільки можна модернізувати сервер без зміни клієнтської частини;
- Мінімізація трафіку. REST API підтримує кешування, що дозволяє значно зменшити кількість запитів до сервера і знизити навантаження на нього. Використання кешованих даних є особливо корисним для систем із великим

обсягом запитів, що дозволяє підвищити швидкість доступу до інформації і покращити загальну продуктивність;

- Відповідність сучасним стандартам. REST API широко підтримується і має стандартизований підхід, що робить його зручним для роботи з більшістю сучасних фреймворків і клієнтських платформ. Це знижує ризик несумісності та дозволяє використовувати REST API в поєднанні з популярними технологіями, такими як React, Angular, iOS або Android.

Використання REST API забезпечує низку переваг для даного проекту, особливо враховуючи необхідність швидкого та надійного доступу до серверних даних. REST API дозволяє легко керувати ресурсами (наприклад, інформацією про користувачів, курси та команди), забезпечуючи простий і зрозумілий інтерфейс для клієнтських додатків. Він забезпечує високу гнучкість і масштабованість системи, що дозволяє проекту залишатися ефективним і функціональним при зміні вимог або зростанні навантаження на сервер.

2.4 Проектування структури бази даних

ER-діаграма (Entity-Relationship діаграма, діаграма «Сутність-Зв'язок») — це графічне представлення структури бази даних, яке показує, як основні об'єкти (сутності) системи пов'язані між собою. Вона є важливим інструментом для розробників, аналітиків та архітекторів баз даних, оскільки дозволяє наочно побачити та зрозуміти взаємозв'язки між даними в системі ще до того, як буде написаний код або створена сама база даних.

Використовується дана діаграма для опису структури даних, визначення зв’язків між об'єктами і можуть бути як основою для проектування нових систем, так і для документування існуючих. Це основний етап у розробці баз даних, що допомагає запобігти помилкам у побудові структури і забезпечити, щоб система була логічно узгодженою та ефективною.

ER-діаграма описує структуру бази даних для системи управління проєктами, яка призначена для підтримки взаємодії між користувачами, командами, проєктами

та завданнями. Система побудована для того, щоб користувачі могли працювати над проєктами у складі команд, отримувати доступ до навчальних матеріалів, а також отримувати оцінки за виконану роботу. Різні ролі користувачів надають різні права доступу до даних і функціоналу системи.

Нижче наведено детальний опис кожної сутності, її атрибутів і зв’язків у системі:

1. Users (Користувачі) - зберігає інформацію про кожного користувача, його облікові дані, роль та анкетні дані.

Атрибути:

- user_id: Унікальний ідентифікатор користувача;
- role_id: Зовнішній ключ, що посилається на role_id у Roles (визначає роль користувача);
- questionnaire_id: Зовнішній ключ, що посилається на questionnaire_id у Questionnaires (пов'язує з анкетними даними);
- email: Адреса електронної пошти користувача;
- password: Пароль користувача для входу в систему.

Зв’язки:

- Зв'язок M:1 з Roles через role_id, що визначає права доступу користувача;
- Зв'язок 1:1 з Questionnaires через questionnaire_id для зберігання особистих даних користувача;
- Зв'язок 1:M з Grades для зберігання оцінок, прив'язаних до учасників.

2. Roles (Ролі) - визначає права доступу для кожної ролі користувача в системі.

Атрибути:

- role_id: Унікальний ідентифікатор ролі;
- role_name: Назва ролі ("учень"/"учитель"/"адміністратор");
- rights: Права доступу, що визначають можливості користувача з цією роллю.

Зв’язки:

- Зв'язок 1:1 з Users для призначення ролей різним користувачам.

3. Questionnaires (Анкети) - містить додаткову інформацію про користувачів, таку як дата народження, зображення, оцінки та тип темпераменту.

Атрибути:

- questionnaire_id: Унікальний ідентифікатор анкети;
- name: Ім'я користувача;
- birth: Дата народження користувача;
- image: Посилання на зображення користувача;
- gender: Стать користувача;
- input_grades: Оцінки, введені користувачем;
- temperament_type: Тип темпераменту користувача (визначається на основі відповідей на питання).

Зв’язки:

- Зв'язок 1:1 з Users, що дозволяє зберігати особисті дані про користувача.

4. Teams (Команди) - дозволяє організовувати користувачів у команди для спільної роботи над проєктами.

Атрибути:

- team_id: Унікальний ідентифікатор команди;
- project_id: Зовнішній ключ, що посилається на project_id у Projects (визначає проєкт, над яким працює команда);
- team_name: Назва команди.

Зв’язки:

- Зв'язок M:1 з Projects через project_id для зв’язку команди з проєктом;
- Зв'язок 1:M з Users для відображення учасників команди через зв’язуючу таблицю;
- Зв'язок 1:M з Grades для прив'язки оцінок до команди в рамках проєкту.

5. Projects (Проєкти) - є основною одиницею роботи в системі та включає назву, опис, статус та завдання.

Атрибути:

- project_id: Унікальний ідентифікатор проєкту;

- task_id: Зовнішній ключ, що посилається на task_id у Tasks (визначає завдання, пов'язані з проєктом);
- project_name: Назва проєкту;
- description: Опис проєкту;
- status: Статус проєкту.

Зв’язки:

- Зв'язок 1:M з Teams для визначення команд, що працюють над проєктом;
- Зв'язок 1:M з Tasks через task_id, що дозволяє прив’язати кілька завдань до проєкту.

6. Tasks (Завдання) - конкретні завдання, пов'язані з проєктами, які мають опис, матеріали та кінцеву дату виконання.

Атрибути:

- task_id: Унікальний ідентифікатор завдання;
- material_id: Зовнішній ключ, що посилається на material_id у Materials (визначає матеріали для завдання);
- task_name: Назва завдання;
- description: Опис завдання;
- deadline: Кінцева дата виконання завдання.

Зв’язки:

- Зв'язок M:1 з Projects через task_id для прив’язки завдання до проєкту;
- Зв'язок M:1 з Materials для прив’язки матеріалів до завдання.

7. Materials (Матеріали) - навчальні або робочі матеріали, що використовуються у завданнях.

Атрибути:

- material_id (PK): Унікальний ідентифікатор матеріалу;
- type: Тип матеріалу (наприклад, документ, відео);
- file_url: URL-адреса матеріалу.

Зв’язки:

- Зв'язок M:1 з Tasks для надання необхідних матеріалів для завдання.

8. Grades (Оцінки) - зберігає оцінки, які користувачі отримують за участь у проєктах та завданнях.

Атрибути:

- grade_id: Унікальний ідентифікатор оцінки;
- user_id: Зовнішній ключ, що посилається на user_id у Users (визначає користувача, якому належить оцінка);
- team_id: Зовнішній ключ, що посилається на team_id у Teams (визначає команду, яку оцінено);
- project_id: Зовнішній ключ, що посилається на project_id у Projects (визначає проєкт для оцінки);
- grade_value: Значення оцінки;
- comments: Коментар до оцінки.

Зв’язки:

- Зв'язок M:1 з Users для прив’язки оцінки до користувача;
- Зв'язок M:1 з Teams для прив’язки оцінки до команди;
- Зв'язок M:1 з Projects для прив’язки оцінки до конкретного проєкту.

ER-діаграма зображена на рис. 2.6

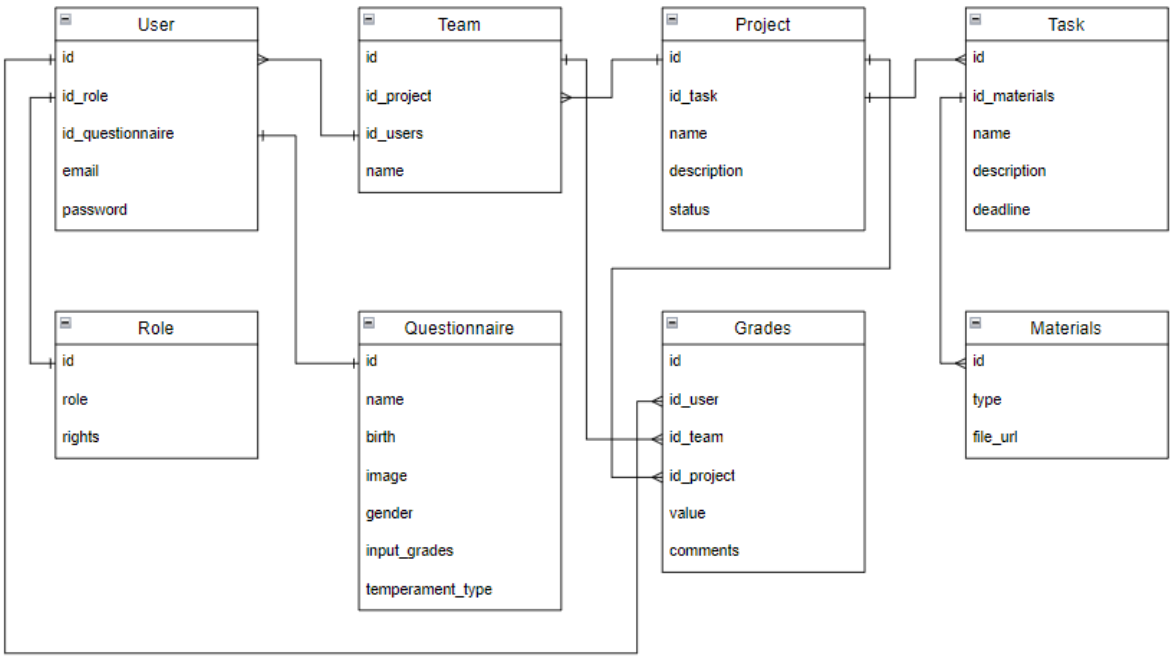

Рисунок 2.6 - ER-діаграма

Кожна з цих сутностей має унікальні атрибути та зв'язки з іншими сутностями, що дозволяє ефективно управляти всіма аспектами роботи над проєктами. Ця схема описує структуру даних для управління проєктами, командами, завданнями, матеріалами і оцінками, що дозволяє ефективно керувати навчальними чи робочими процесами в системі.

3 РОЗРОБКА АЛГОРИТМІЧНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ НАВЧАННЯ ОНЛАЙН

3.1 Моделювання архітектури системи

Моделювання архітектури — це процес створення абстрактного представлення структури програмної системи, її компонентів та взаємодії між ними. Моделювання архітектури дозволяє зрозуміти, як система буде реалізована на практиці, визначити її основні складові, взаємозв'язки та відповідність вимогам користувачів і бізнесу.

Основна мета моделювання архітектури полягає в тому, щоб забезпечити чітке бачення всіх аспектів системи ще до початку її реалізації, а також спростити процеси обговорення, аналізу та прийняття рішень. Воно також допомагає визначити потенційні ризики, оптимізувати витрати ресурсів і забезпечити масштабованість, продуктивність та надійність майбутньої системи.

Діаграма розгортання (UML Deployment diagram) — це тип діаграми, яка використовується для моделювання фізичного розгортання компонентів програмної системи на апаратному забезпеченні. Вона показує, як програмні компоненти взаємодіють з апаратними вузлами (сервери, клієнтські машини, бази даних тощо) у робочому середовищі.

Основні елементи діаграми розгортання:

- Вузли (Nodes):
  - Це апаратні чи програмні ресурси, які виконують певні завдання;
  - В UML вузол позначається у вигляді тривимірного куба;
  - Наприклад: сервери, пристрої користувачів, мережеве обладнання.
- Артефакти (Artifacts):
  - Це програмні компоненти, які розгортаються на вузлах, наприклад, виконувані файли, скрипти, бази даних;
  - В UML вони позначаються прямокутниками з написом типу артефакта.
- З'єднання (Communication Paths):

- Лінії, які показують взаємодію між вузлами;
- Вказують на те, як дані передаються між вузлами (наприклад, через мережу).

- Стереотипи:
  - Використовуються для уточнення ролей вузлів чи артефактів (наприклад, «сервер», «клієнт», «база даних»).

Діаграма розгортання зображена на рис. 3.1:

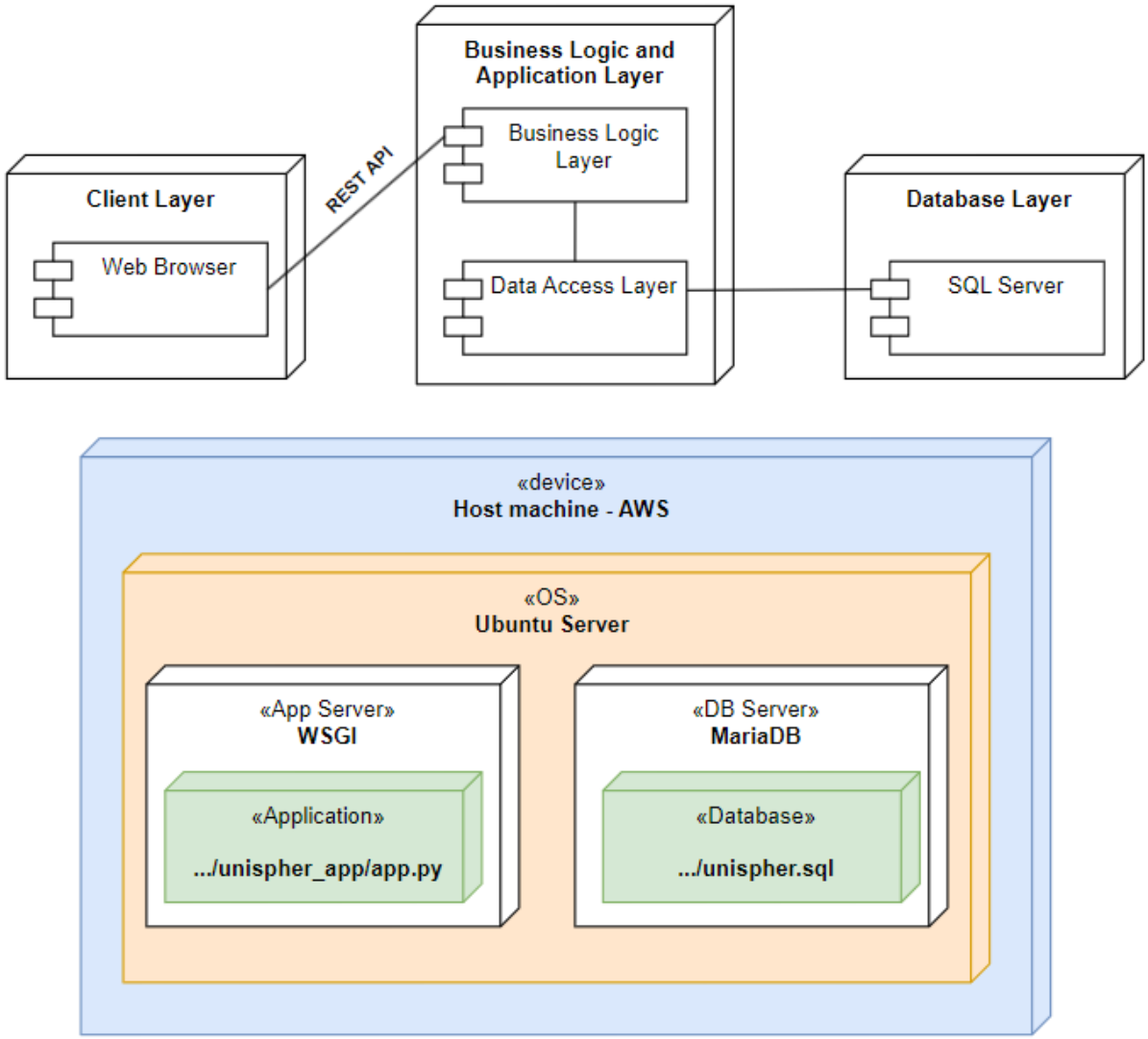

Рисунок 3.1 – UML Діаграма розгортання

На цій діаграмі зображена система, яка складається з кількох частин, що працюють разом:

- Клієнтський шар (Client Layer) - це комп'ютер, де користувач використовує веб-браузер для доступу до програми;
- Прикладний шар (Business Logic and Application Layer) - виконує бізнес-логіку (основні обчислення та правила програми). Зв'язується з базою даних через шар доступу до даних (Data Access Layer);
- Шар бази даних (Database Layer) - тут зберігається інформація в базі даних (SQL Server);
- Хост-машина в AWS - уся система розгорнута на сервері в хмарі (AWS - Amazon Web Services). Встановлена операційна система Ubuntu;
- Складові хост-машини – це App Server (сервер додатків): Використовує WSGI для запуску Python-додатку (app.py). DB Server (сервер бази даних): Використовує MariaDB для роботи з файлом бази даних (unisphere.sql).

3.2 Моделювання класів та об'єктів

Моделювання класів і об'єктів є важливим етапом у розробці серверної частини клієнт-серверної системи. На цьому етапі розробляються UML (діаграми класів та об'єктів, які дають змогу візуалізувати структуру системи, взаємозв'язки між її елементами та визначити основні компоненти, що будуть реалізовані у вигляді класів у програмному коді.

UML (Unified Modeling Language) — це уніфікована мова моделювання, яка дозволяє візуально представляти структуру та поведінку програмних систем. Одним з основних інструментів UML є діаграми класів і об'єктів, які допомагають розробникам проєктувати системи, моделювати зв'язки між компонентами, планувати інтеграцію та забезпечувати кращу підтримку коду.

Моделювання за допомогою UML-діаграм класів і об'єктів надає детальне бачення архітектури програмного забезпечення. Це забезпечує можливість:

- визначити необхідні класи та їхні властивості;

- зрозуміти зв’язки та взаємодії між компонентами;
- уникнути надмірності та ускладнень у структурі;
- забезпечити узгодженість системи на всіх рівнях розробки та впровадження.

Таким чином, UML-діаграми класів і об'єктів дозволяють не лише оптимально спланувати структуру серверної частини, але й спрощують наступні етапи тестування та інтеграції.

Діаграма класів (UML Class Diagram) - є статичною моделлю, яка фокусується на структурі системи. Вона показує класи, їх атрибути, методи та зв'язки між ними, забезпечуючи загальне уявлення про те, з яких компонентів складається система.

Основні елементи діаграми класів:

- Клас — основний елемент, який відображається у вигляді прямокутника з трьома секціями;
- Ім'я класу — розташовується у верхній частині;
- Атрибути — розміщуються в середній частині та представляють властивості або змінні класу;
- Методи — у нижній частині, представляючи функціонал або поведінку класу;
- Взаємозв’язки між класами — ключовий аспект діаграми, який відображає типи зв’язків між класами, як-от асоціація, агрегація, композиція та наслідування.

Типи зв'язків між класами:

- Асоціація — відображає зв’язок між двома класами, де один клас може використовувати функціонал іншого. Зазвичай представлений суцільною лінією;
- Агрегація — показує, що один клас складається з кількох інших, які можуть існувати окремо від нього. На діаграмі відображається порожньою ромбоподібною позначкою на боці класу, що містить інші;

- Композиція — вказує на те, що один клас є складовою частиною іншого і не може існувати окремо. Позначається суцільним ромбом на боці класу-власника;
- Успадкування (або генералізація) — показує ієрархічний зв’язок між класами, де один клас успадковує властивості іншого. Відображається порожньою стрілкою, спрямованою до батьківського класу.

Створимо UML діаграму класів для нашої системи, яка складається з наступних атрибутів, методів та зв’язків між класами:

Клас «Admin»:

- Атрибути: Немає.
- Методи:
  - monitorActivity(): відстежує активність користувачів на платформі;
  - manageUsers(): дозволяє адміністратору керувати списком користувачів;
  - grantAccessRights(): забезпечує надання прав доступу іншим користувачам;
  - manageSecurity(): відповідає за управління безпекою системи;
  - sendInvitations(): надсилає запрошення новим користувачам.
- Зв'язок:
  - Наслідування з класом User.

Клас «Teacher»:

- Атрибути: Немає.
- Методи:
  - createTeam(): створює команду;
  - deleteTeam(): видаляє команду;
  - createProject(): створює проєкт;
  - evaluateProject(): оцінює проєкт;
  - monitorProgress(): відстежує прогрес виконання проєкту;
  - assignTasks(): призначає завдання студентам або командам;

- assignGrade(): присвоює оцінки студентам.

- Зв'язок:
  - Наслідування з класом User.

Клас «User»:

- Атрибути:
  - id: унікальний ідентифікатор користувача;
  - id_questionnaire: посилання на анкету користувача;
  - email: електронна адреса користувача;
  - password: пароль для доступу до системи;
  - role: роль користувача (наприклад, студент, вчитель, адміністратор).
- Методи:
  - register(): проходження анкети нового користувача;
  - authenticate(): аутентифікує користувача при вході в систему;
  - editProfile(): редагує профіль користувача;
  - viewProject(): переглядає інформацію про проєкти;
  - viewTasks(): переглядає призначені завдання;
  - viewGrades(): переглядає свої оцінки;
  - checkAccess(): перевіряє права доступу користувача.
- Зв'язки:
  - Агрегація з класом Team.

Клас «Questionnaire»

- Атрибути:
  - id: унікальний ідентифікатор анкети;
  - name: ім'я користувача, якого описує анкета;
  - birth: дата народження користувача;
  - image: зображення профілю користувача;
  - gender: стать користувача;
  - input_grades: масив оцінок, які вносить користувач;
  - temperament_type: тип темпераменту користувача.

- Методи:
    - viewQuestionnaire(): переглядає анкету;
    - updateQuestionnaire(): оновлює інформацію в анкеті.
- Зв'язок:
    - Композиція з User (один до одного) - кожен користувач має одну анкету.

Клас «Team»:

- Атрибути:
    - id: унікальний ідентифікатор команди;
    - id_project: посилання на проєкт, який виконує команда;
    - id_user: список користувачів, які належать до команди;
    - name: назва команди.
- Методи:
    - viewMembers(): переглядає список учасників команди;
    - editTeam(): редагує інформацію про команду;
    - editMembers(): редагує склад команди.
- Зв'язки:
    - Асоціація з класом Project (1 до 1).

Клас «Project»:

- Атрибути:
    - id: унікальний ідентифікатор проєкту;
    - name: назва проєкту;
    - description: опис проєкту;
    - status: поточний статус проєкту.
- Методи:
    - editProject(): редагує інформацію про проєкт;
    - evaluateProject(): оцінює проєкт;
    - monitorProject(): відстежує прогрес виконання проєкту.
- Зв'язки:
    - Композиція з класом Task.

Клас «Task»

- Атрибути:
  - id: унікальний ідентифікатор завдання;
  - name: назва завдання;
  - description: опис завдання;
  - deadline: кінцевий термін виконання завдання;
  - id_materials: посилання на матеріали, необхідні для виконання завдання.
- Методи:
  - createTask(): створює нове завдання;
  - editTask(): редагує існуюче завдання;
  - deleteTask(): видаляє завдання;
  - assignTask(): призначає завдання студентам або командам;
  - viewMaterials(): переглядає матеріали, прикріплені до завдання.
- Зв'язок:
  - Агрегація з класом Materials.

Клас «Materials»

- Атрибути:
  - id: унікальний ідентифікатор матеріалу;
  - type: тип матеріалу (наприклад, документ, відео тощо);
  - file_url: URL-адреса або шлях до файлу.
- Методи:
  - addMaterial(): додає новий матеріал до системи;
  - deleteMaterial(): видаляє матеріал;
  - downloadMaterial(): завантажує матеріал.

Клас «Grades»

- Атрибути:
  - id: унікальний ідентифікатор оцінки;
  - id_user: ідентифікатор користувача, якому призначена оцінка;
  - id_team: ідентифікатор команди, яка пов'язана з оцінкою;

- id_project: ідентифікатор проєкту, за який виставлена оцінка;
- value: значення оцінки;
- comment: коментар до оцінки.

- Методи:
  - viewGrade(): переглядає виставлені оцінки;
  - addComment(): додає коментар до оцінки.
- Зв'язок:
  - Асоціація з User, Team, і Project (багато до одного) - одна оцінка може відноситись до конкретного студента, команди і проєкту.

UML діаграма класів зображена на рис. 3.2:

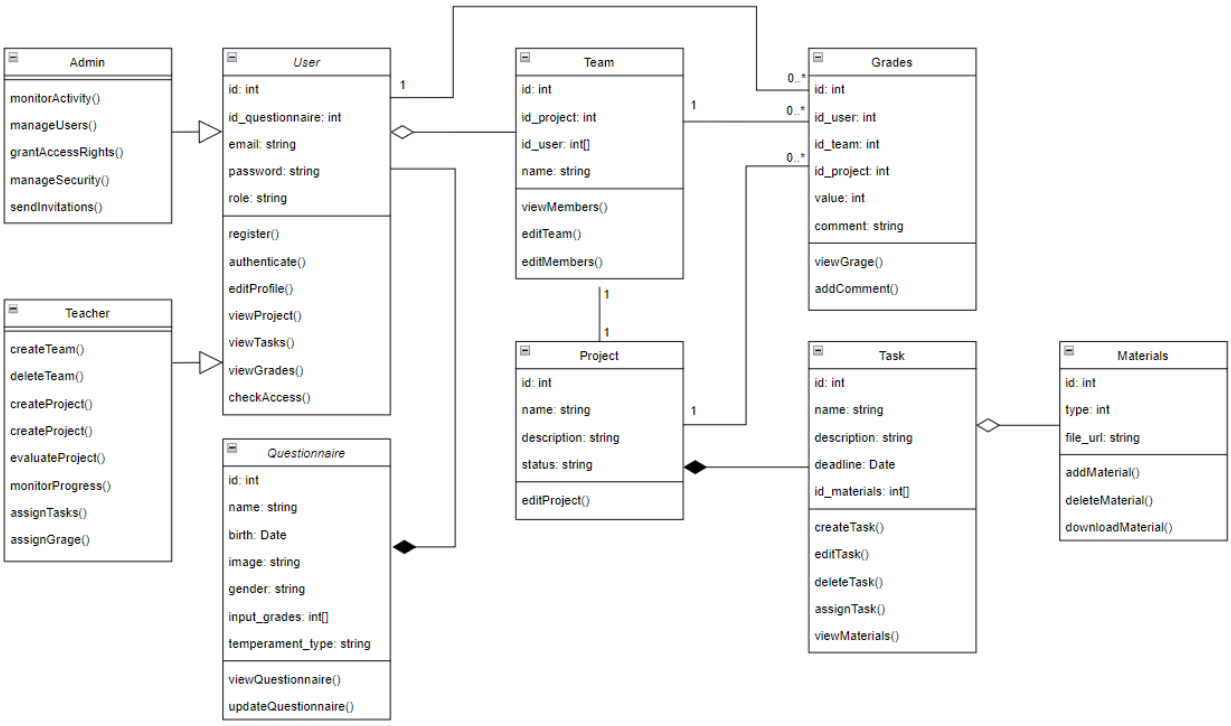

Рисунок 3.2 – UML діаграма класів

Діаграма об'єктів (UML Object Diagram) є конкретною інстанцією діаграми класів, яка показує реальні об'єкти, що існують у певний момент часу, і їхній стан. Ця діаграма дозволяє краще зрозуміти, як класи взаємодіють у реальному контексті і як виглядає структура об'єктів у системі.

Елементи діаграми об'єктів:

- Об'єкти — представлені прямокутниками, які подібні до класів, але містять реальні значення атрибутів;
- Атрибути об'єктів — містять конкретні значення, що допомагає зрозуміти стан об'єкта у певний момент;
- Зв'язки: Зв'язки на діаграмі об'єктів відображають конкретні асоціації між екземплярами класів, що існують у цей момент. Кожен зв'язок показує реальні екземпляри об'єктів, які взаємодіють між собою відповідно до класової структури.

Створимо UML діаграму об’єктів для нашої системи, використовуючи уже раніше створенну UML діаграму класів (рис.3.1).

Об’єкт Admin1:

- id: 1;
- id_questionnaire: nan (без анкети);
- email: admin@example.com;
- password: Admin1;
- role: Admin.

Об’єкт Teacher1:

- id: 2;
- id_questionnaire: 1 (посилання на об'єкт Questionnaire1:Questionnaire);
- email: teacher@example.com;
- password: Teacher2;
- role: Teacher.

Об’єкт Questionnaire1:

- id: 1;
- name: John Doe;
- birth: 2000-05-15;
- image: images/johndoe.jpg;
- gender: male;

- input_grades: [] (порожній масив оцінок);
- temperament_type: Choleric.

Об’єкт Student1:

- id: 3;
- id_questionnaire: 2 (посилання на об'єкт Questionnaire2:Questionnaire);
- email: student@example.com;
- password: Student3;
- role: Student.

Об’єкт Questionnaire2:

- id: 2;
- name: Billy Jean;
- birth: 2010-12-12;
- image: images/billyjean.jpg;
- gender: male;
- input_grades: [5, 4, 5, 3, 4] (масив оцінок, отриманих студентом);
- temperament_type: Sanguine.

Об’єкт Team1:

- id: 101;
- id_project: 201 (посилання на Project1);
- id_user: int[2] (посилання на користувачів у команді);
- name: Team Alpha.

Об’єкт Project1:

- id: 201;
- name: Animals on our Planet;
- description: Students will get to know all kinds of animals on planet Earth;
- status: in Progress.

Об’єкт Task1:

- id: 301;
- name: Mammals;

- description: Study of the main types of mammals;
- deadline: 2024-12-01;
- id_materials: int[1001, 1002] (посилання на об'єкти матеріалів Material1 і Material2).

Об’єкт Material1:

- id: 1001;
- type: book;
- file_url: files/allMammals.pdf.

Об’єкт Material2:

- id: 1002;
- type: photo;
- file_url: files/Mammals.jpg.

Об’єкт Grade1:

- id: 501;
- id_user: 3 (посилання на Student1);
- id_team: 101 (посилання на Team1);
- id_project: 201 (посилання на Project1);
- value: 5;
- comment: Good research.

UML діаграма об’єктів зображена на рис. 3.3:

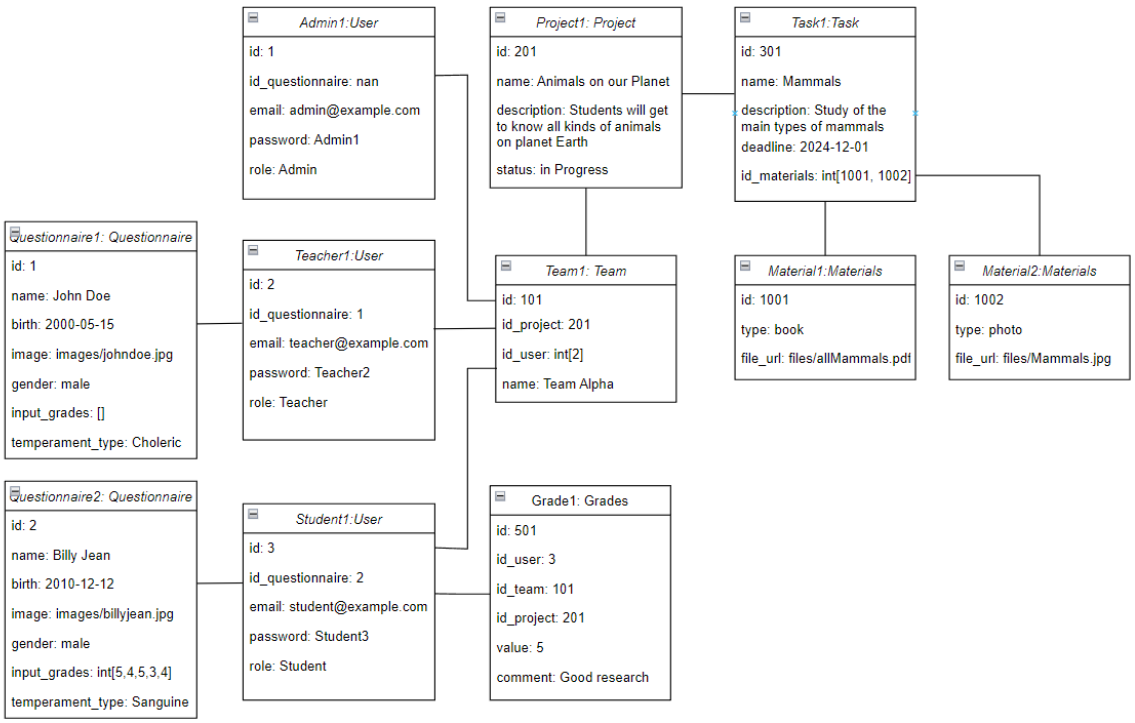

Рисунок 3.3 – UML діаграма об’єктів

3.3 Розробка програмного забезпечення

Розробка програмного забезпечення є ключовим етапом створення клієнт-серверної системи для автоматизації процесу формування команд учнів для навчання онлайн. Цей процес передбачає не лише технічну реалізацію функціоналу серверної частини, але й забезпечення її інтеграції з клієнтськими додатками, відповідність вимогам продуктивності, надійності та масштабованості.

Серверна частина системи виконує центральну роль, забезпечуючи зберігання, обробку та обмін даними між учасниками навчального процесу, що є критично важливим для реалізації цілей автоматизації.

Одним із принципів проєктування програмного забезпечення є розподілення файлів по чітко визначеним завданням кожного шару. Це розділення дозволяє створювати масштабовану, зрозумілу та легку в обслуговуванні систему.

В даному ПЗ буде використано три частини: **Routes/Services/Modules**.

Routes (Маршрути) - відповідають за обробку HTTP-запитів і передачу даних між клієнтом (користувачем API) та сервером. Це точка входу для клієнтських запитів.

Services (Сервіси) - містять бізнес-логіку додатка, наприклад, як саме обробляються запити, взаємодія з базою даних, обчислення або складні операції. Тут реалізується основний "мозок" системи.

Modules (Модулі) - відповідають за інфраструктуру програми: опис структури бази даних (ORM-моделі), взаємодію з таблицями, виконання CRUD-операцій (Create, Read, Update, Delete). Це основа системи, де описані всі дані.

Частина 1. Створення аккаунту та автентифікація користувача (Authentication). На даному етапі користувачі вводять основну (мінімальну) інформацію для створення аккаунту та переходу до анкети.

Реалізація маршрутів Authentication (User).

Цей файл відповідає за взаємодію клієнта із сервером у рамках операцій реєстрації, авторизації та управління обліковими записами.

Реєстрація користувача:

```
@auth.route('/api/register', methods=['POST'])
@swag_from(fr'{path}/docs/auth_docs/register_user.yaml')
@skip_login
def register_user():
    post_data = request.form.to_dict()
    discipline = post_data.pop('discipline_id', None)
    errors = user_schema.validate(post_data, partial=('questionnaire', 'school_id'))
    if errors:
        raise InputValidationError(errors)
    response_object = register_login_user_endpoint(post_data, discipline)
    return wrap_response(response_object).
```

Що в даному коді відбувається?

- @auth.route: Визначає URL /api/register та метод POST;
- @swag_from: Підключає документ з описом цього API до Swagger (документація генерується автоматично);
- request.form.to_dict(): Конвертує отримані дані у словник для зручної обробки;

- user_schema.validate: Перевіряє, чи всі дані відповідають очікуваному формату;
- Якщо валідація успішна, викликається логіка реєстрації через register_login_user_endpoint.

Авторизація користувача:

```
@auth.route('/api/login', methods=['POST'])
@swag_from(fr'{path}/docs/auth_docs/login_user.yaml')
@skip_login
def login():
    post_data = request.form  # Отримання email і пароля.
    resp = login_user_endpoint(post_data)  # Виклик функції бізнес-логіки.
    return wrap_response(resp)  # Формування відповіді.
```

Дані передаються до функції login_user_endpoint, яка перевіряє користувача та створює токен.

Видалення користувача:

```
@auth.route('/api/user/delete/<user_id>', methods=['POST'])
@swag_from(fr'{path}/docs/auth_docs/del_user_by_id.yaml')
def del_user_by_id(user_id):
    return wrap_response(del_user_by_id_endpoint(user_id))
```

Простий спосіб обробляти запити на видалення користувачів із бази даних, передаючи ідентифікатор.

Логіка авторизації.

Логін:

```
def login_user(post_data: dict):
    user = User.query.filter_by(email=post_data.get("email")).first()
    if not user:
        raise UnknownUserEmailError(user_email=post_data.get("email"))
    if bcrypt.check_password_hash(user.password, post_data.get("password")):
```

```
            auth_token = User.encode_auth_token(user.id)
            return {"auth_token": auth_token, "user": user.to_dict()}
        else:
            raise AppBackendError("Invalid credentials.")
```

В даній частині коду перевіряється чи існує користувач із вказаним email, порівнюється хеш пароля в базі з паролем у запиті. Якщо перевірка успішна, генерує токен і повертає його разом із даними користувача.

Реєстрація:

```
def register_login_user(post_data: dict, discipline):
        user = User.query.filter_by(email=post_data.get("email")).first()
           if user:
            return {"message": "User already exists."}

        new_user = User(
            email=post_data.get("email"),
            password=bcrypt.generate_password_hash(post_data.get("password")).decode("utf-8"),
            name=post_data.get("name"),
            surname=post_data.get("surname"),
            is_active=True,
        )
        db.session.add(new_user)
        db.session.commit()
        return {"message": "User registered successfully."}
```

В даній частині коду перевіряється чи немає в базі користувача з таким email. Якщо немає то створюється новий користувач, хешуючи пароль. Додається користувач до бази даних і зберігаються зміни.

Визначення моделі:

```
class User(DbModelBase, db.Model, UserMixin):
            __tablename__ = "users"
```

```
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    email = db.Column(db.String(255), unique=True, nullable=False)
    password = db.Column(db.String(255), nullable=False)
    name = db.Column(db.String(200), nullable=False)
    surname = db.Column(db.String(200), nullable=False)
    is_active = db.Column(db.Boolean(), nullable=False, default=True)
```

В цій частині коду __tablename__ вказує назву таблиці в базі. Поля (email, password, is_active) відповідають колонкам таблиці. UserMixin додає корисні методи для роботи з авторизацією (Flask-Login).

Методи моделі.

Токени авторизації:

```
@staticmethod
def encode_auth_token(user_id):
    payload = {
        "exp": datetime.datetime.utcnow() + datetime.timedelta(days=7),
        "iat": datetime.datetime.utcnow(),
        "sub": user_id,
    }
    return jwt.encode(payload, app.config.get("SECRET_KEY"), algorithm="HS256")
@staticmethod
def decode_auth_token(auth_token):
    try:
        payload = jwt.decode(auth_token, app.config.get("SECRET_KEY"),
algorithms=["HS256"])
        return payload["sub"]
    except jwt.ExpiredSignatureError:
        return "Signature expired."
    except jwt.InvalidTokenError:
        return "Invalid token."
```

Метод encode_auth_token генерує JWT-токен для користувача. В свою чергу метод decode_auth_token розшифровує токен для отримання ID користувача.

Частина 2. Створення анкети (questionnaire), яку повинен пройти кожен користувач після реєстрації аккаунту. Користувачі можуть переглядати анкети, зберігати результати та працювати з анкетними даними.

Реалізація маршрутів Questionnaire.

Цей файл забезпечує взаємодію клієнтів із сервером у рамках роботи з анкетами: отримання блоків анкети, збереження результатів, перегляд даних.

Отримання всіх блоків анкети за ID користувача:

```
@api_questionnaire.route('/api/questionnaire/<user_id>', methods=['GET'])
@swag_from(f'{path}/docs/questionnaire_docs/get_by_user_id.yaml')
@allow_inactive_user
@roles_accepted_custom('student', 'teacher', 'admin', 'super_admin', 'school_admin')
def get_all_block_by_user_id(user_id):
        return wrap_response(get_all_block_by_user_id_endpoint(user_id))
```

Даний код отримує user_id із маршруту. Перевіряє доступ користувача (через roles_accepted_custom).

Викликає функцію логіки get_all_block_by_user_id_endpoint, яка повертає всі блоки анкети.

Збереження результатів анкети:

```
@api_questionnaire.route('/api/questionnaire/<user_id>', methods=['PUT'])
@swag_from(fr'{path}/docs/questionnaire_docs/save_result.yaml')
@roles_accepted_custom('student', 'teacher', 'admin', 'super_admin', 'school_admin')
def save_result(user_id):
        data = request.form  # Отримання даних анкети.
        return wrap_response(save_result_endpoint(user_id, data))
```

Даний код забезпечує збереження відповідей у базі через функцію save_result_endpoint.

Збереження результатів поточного користувача:

```
@api_questionnaire.route('/api/questionnaire/me', methods=['PUT'])
@swag_from(fr'{path}/docs/questionnaire_docs/save_user_result.yaml')
@allow_inactive_user
@roles_accepted_custom('student', 'teacher', 'admin', 'super_admin', 'school_admin')
def save_user_result():
    data = request.form.get('data')  # Отримує JSON-дані з форми.
    if g.user.roles[0].label == 'Student':
        data = {'data': ast.literal_eval(data)}  # Преобразує дані у словник.
        errors = student_schema.validate(data.get('data'))  # Валідація студентської анкети.
        if errors:
            raise InputValidationError(errors)
    return wrap_response(save_user_result_endpoint(None, data, True))
```

Дана частина коду дозволяє студентам і викладачам надсилати результати анкет, використовуючи свою роль.

Логіка для роботи з анкетами (Questionnaire).

Отримання анкети за ID користувача:

```
def get_by_user(user_id, inactive):
    if inactive:
        user_id = g.user.id  # Якщо користувач неактивний, береться його ID.
    role = UserRole.get_by_user_id_all(user_id)  # Отримання ролі користувача.

    student_role_id = Role.get_by_name("student").id
    teacher_role_id = Role.get_by_name("teacher").id
    roles = {'student': student_role_id, 'teacher': teacher_role_id}
    if role.role_id == roles['student']:
        questionnaire = StudentQuestionnaire.get_by_user_id(user_id)  # Анкета студента.
        return questionnaire.to_dict() if questionnaire else []
    if role.role_id == roles['teacher']:
        questionnaire = TeacherQuestionnaire.get_by_user_id(user_id)  # Анкета викладача.
        return questionnaire.to_dict() if questionnaire else []
```

Повертає анкету залежно від ролі (студент або викладач).

Збереження результатів анкети:

```
def save_user_result(user_id: int, data: dict, inactive: bool):
    if inactive:
        user_id = g.user.id
    role = UserRole.get_by_user_id_all(user_id)
    student_role_id = Role.get_by_name("student").id
    teacher_role_id = Role.get_by_name("teacher").id
    if role.role_id == student_role_id:
        errors = student_schema.validate(data.get('data'))
        if errors:
            raise InputValidationError(errors)
        questionnaire = StudentQuestionnaire.save_result(user_id, data)
        return questionnaire.to_dict()
```

Даний код перевіряє, чи активний користувач. Залежно від ролі, виконує валідацію даних анкети. Викликає метод save_result для збереження.

Моделі Questionnaire:

```
class Questionnaire(DbModelBase, db.Model):
    __tablename__ = 'questionnaire'

    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id', ondelete='CASCADE'), nullable=False)
    data = db.Column(db.JSON(), nullable=False, server_default='{}')
```

В даній частині коду __tablename__ визначає назву таблиці в базі, а саме «questionnaire». Поле data зберігає JSON-дані анкети.

Збереження результатів Questionnaire:

```
    @staticmethod
    def save_result(user_id, raw):
        obj = Questionnaire.get_by_user_id(user_id)
        if obj:
            for r in raw:
                if hasattr(obj, r):
                    data = raw[r]
                    if r == 'data':
                        data = ast.literal_eval(raw[r])
                    setattr(obj, r, data)
        try:
            db.session.commit()
        except SQLAlchemyError as ex:
            db.session.rollback()
            raise AppBackendError(ex)
        return obj
```

Даний код оновлює існуючу анкету, додаючи нові дані.

Частина 3. Створення команди. На даному етапі реалізується бізнес-логіка для операцій із командами: створення, оновлення, отримання списку учасників, видалення.

Реалізація маршрутів Teams.

Отримання списку команд:

```
@api_team.route("/api/teams", methods=["GET"])
@swag_from(f"{path}/docs/team_docs/get_all_teams.yaml")
@allow_inactive_user
@roles_accepted_custom("student", "teacher", "admin", "super_admin", "school_admin")
def get_team_list():
    return wrap_response(get_team_list_endpoint())
```

За допомогою запиту GET /api/teams можна отримати список всіх команд.

Оновлення списку команди:

```
@api_team.route("/api/teams/<team_id>", methods=["PUT"])
@swag_from(f"{path}/docs/team_docs/update_team_by_id.yaml", methods=["PUT"])
@roles_accepted_custom("student", "teacher", "admin", "super_admin", "school_admin")
def upd_team(team_id):
        form = request.form.copy()
        file = request.files.get("image_path", None)
        if form or file:
            errors = team_schema.validate(form, partial=True)
            if errors:
                raise InputValidationError(errors)
            return wrap_response(upd_team_endpoint(team_id, file, form))
        json: dict = request.json
        if json:
            errors = team_schema.validate(json, partial=True)
            if errors:
                raise InputValidationError(errors)
            file = request.files.get("image_path", None)
            return wrap_response(upd_team_endpoint(team_id, file, json))
```

За допомогою запиту PUT /api/teams/<team_id> обробляються запити для оновлення інформації про команду. У разі успішної валідації передаються у функцію upd_team_endpoint.

Створення команди:

```
@api_team.route("/api/teams", methods=["POST"])
@swag_from(f"{path}/docs/team_docs/add_team.yaml")
@roles_accepted_custom("student", "teacher", "admin", "super_admin", "school_admin")
def add_team():
        data = request.form  # Зчитує дані з тіла POST-запиту.
        errors = team_schema.validate(data)  # Валідує отримані дані через TeamSchema.
        if errors:
            raise InputValidationError(errors)  # Генерує виняток, якщо дані невалідні.
```

```
    return wrap_response(add_team_endpoint(data))  # Передає обробку бізнес-логіці.
```

За допомогою запиту POST /api/teams оброблюється запит для створення нової команди.

Видалення команди:

```
@api_team.route("/api/teams/<team_id>", methods=["DELETE"])
@swag_from(f"{path}/docs/team_docs/delete_team_by_id.yaml")
@roles_accepted_custom("student", "teacher", "admin", "super_admin", "school_admin")
def del_team(team_id):
    return wrap_response(del_team_endpoint(team_id))
```

За допомогою запиту DELETE /api/teams/<team_id> обробляється запит на видалення команди за її id.

Логіка для роботи з командами (Teams).

Отримання списку команд:

```
def get_team_list_endpoint():
    teams = Team.query.all()
    return {"teams": [team.to_dict() for team in teams]}
```

За допомогою Team.query.all() отримуються всі записи з таблиці teams. Результат перетворюється у список словників через метод моделі to_dict().

Перевірка та створення нової команди:

```
def generate_team(data):
    project_id = data.get("project_id")
    school_id = g.user.school_id
    teams = Team.query.filter_by(school_id=school_id, project_id=project_id).all()

    if teams:
       raise AppBackendError(f"Teams for school {school_id} already exist.")
```

```
    new_team = Team.add_new(data)
    return {"message": "Team successfully created.", "team": new_team}
```

Даний код перевіряє, чи існує команда для проекту в школі. Якщо команди немає, створює нову за допомогою моделі Team.

Оновлення команди:

```
def upd_team_endpoint(team_id, file, form_data):
    team = Team.upd_by_id(team_id, form_data)  # Оновлює атрибути.
    if file:
        team.image_path = Team.images_path(file, team_id, "image")  # Додає зображення.
    db.session.commit()  # Зберігає зміни в базі.
    return {"message": "Team successfully updated", "team": team.to_dict()}# Повертає результат у форматі словника.
```

Даний код за допомогою методу Team.upd_by_id, оновлює інформацію про команду.

Видалення команди:

```
def del_team_endpoint(team_id):
    Team.del_by_id(team_id)  # Викликає метод моделі для видалення.
    return {"message": "Team successfully deleted"}
```

Даний код викликає метод Team.del_by_id та видаляє команду відповідно її id.

Моделі Teams:

Модель комнди:

```
class Team(DbModelBase, db.Model):
    __tablename__ = "teams"

    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
```

```
    name = db.Column(db.String(255), unique=True, nullable=False)
    school_id = db.Column(db.Integer, db.ForeignKey("schools.id"), nullable=False)
    project_id = db.Column(db.Integer, db.ForeignKey("projects.id"), nullable=False)
```

Додавання нової команди:

```
@staticmethod
def add_new(data):
    team = Team(
        name=data.get("name"),
        school_id=data.get("school_id"),
        project_id=data.get("project_id"),
    )
    db.session.add(team)  # Додає команду в сесію.
    db.session.commit()  # Зберігає зміни в базі.
    return team.to_dict()  # Повертає дані у вигляді словника.
```

Оновлення команди:

```
@classmethod
def upd_by_id(cls, rec_id, raw):
    team = cls.get_by_id(rec_id)  # Отримує команду за ID.
    for key, value in raw.items():
        setattr(team, key, value)  # Оновлює атрибути.
    db.session.commit()  # Зберігає зміни.
    return team  # Повертає оновлений об'єкт.
```

Видалення команди:

```
@classmethod
def del_by_id(cls, rec_id):
    result = cls.query.filter_by(id=rec_id).delete()  # Видаляє запис.
    db.session.commit()  # Зберігає зміни.
    if result == 0:
        raise UnknownIdError("Team", rec_id)  # Генерує помилку, якщо команда не знайдена.
    return result
```

У підсумку можна сказати, що код розподілений за функціоналом, тому демонструє добре структуровану систему, яка легко підтримується. Чітке розділення між маршрутами, бізнес-логікою та моделями забезпечує масштабованість і зручність тестування. Коментарі пояснюють кожен крок, роблячи код зрозумілим навіть для стороннього розробника.

3.4 Тестування програмного забезпечення

Тестування програмного забезпечення є невід'ємною частиною життєвого циклу розробки будь-якої програмної системи, особливо у випадках створення клієнт-серверних систем, які мають забезпечувати стабільну та безпечну роботу в умовах великого навантаження. Метою тестування є виявлення помилок у функціональності, продуктивності, безпеці та сумісності, що дозволяє підвищити якість програмного продукту та мінімізувати ризики, пов’язані з його використанням.

У процесі тестування серверної частини клієнт-серверної системи для автоматизації процесу формування команд учнів особливу увагу приділяють перевірці правильності обробки запитів, відповідності API задекларованим специфікаціям, а також стійкості системи до некоректних даних чи збоїв. Даний етап дозволяє гарантувати, що створене програмне забезпечення відповідає вимогам користувачів, забезпечує належний рівень продуктивності та здатне функціонувати стабільно у реальному середовищі.

У процесі розробки та тестування серверної частини клієнт-серверної системи для автоматизації процесу формування команд учнів ми використовуватимемо Swagger як інструмент для документування та тестування API.

Swagger — це набір відкритих інструментів для створення, опису, документування та тестування REST API. Основна мета Swagger — зробити API зрозумілим як для розробників, так і для кінцевих користувачів, надаючи детальну специфікацію його функціоналу у зрозумілому форматі. Swagger

підтримує стандарт OpenAPI, що є широко розповсюдженим і прийнятим у сучасній індустрії розробки програмного забезпечення.

Отже, використання Swagger забезпечує ефективну взаємодію між розробниками та тестувальниками, дозволяючи перевіряти коректність роботи API у зручному середовищі.

Тому в межах цього пункту ми проведемо тестування основних функцій з метою знаходження та усунення багів.

Тест №1. Реєстрація аккаунту.

Для початку потрібно створити pre-user, та ввести початкову інформацію: email, role_id та mentor.

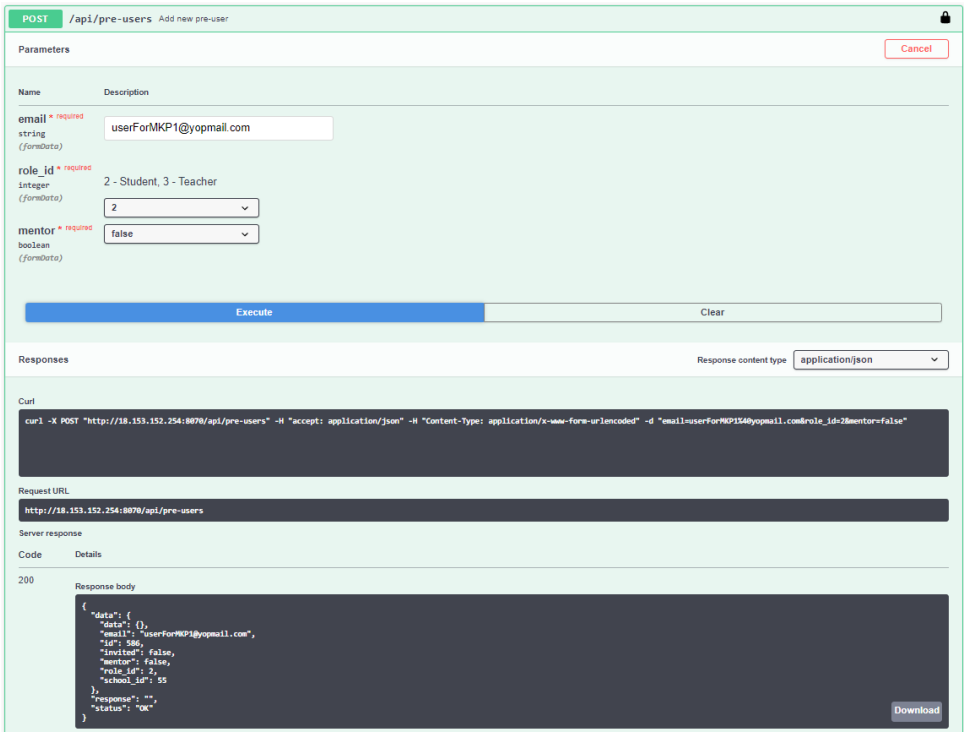

Рисунок 3.4 – Створення pre-user

Після створення pre-user, потрібно для нього створити запрошення (Invitations):

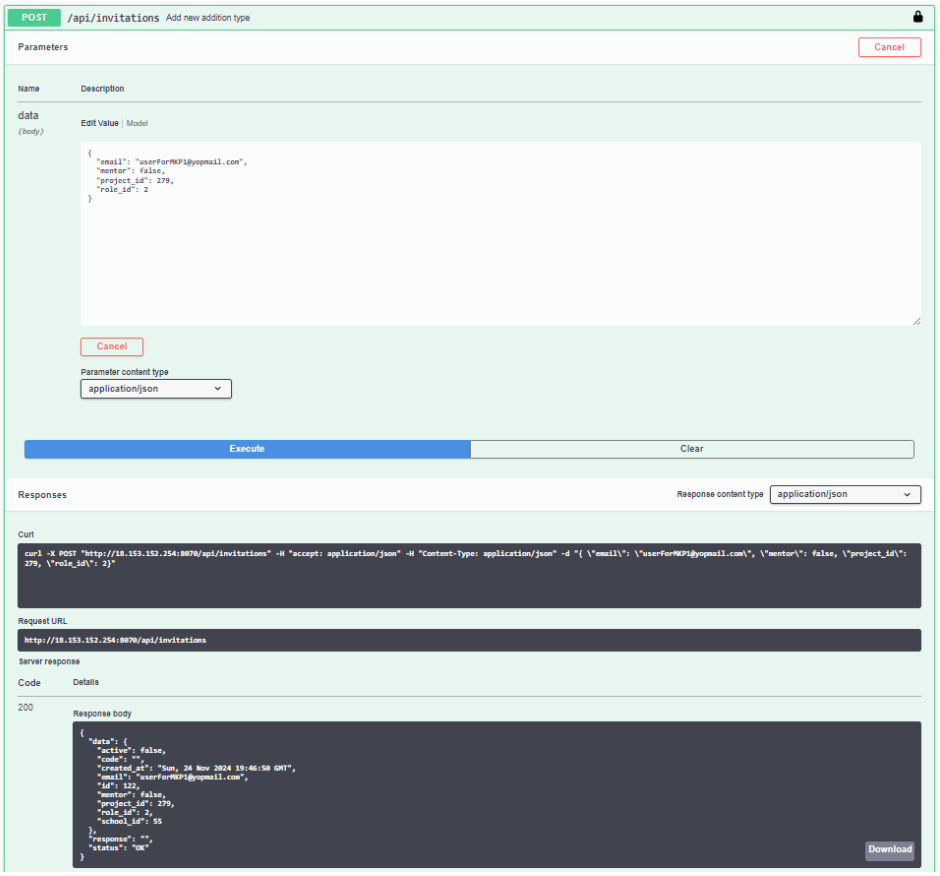

Рисунок 3.5 – Створення запрошення

Наступним етапом є відправлення цього запрошення:

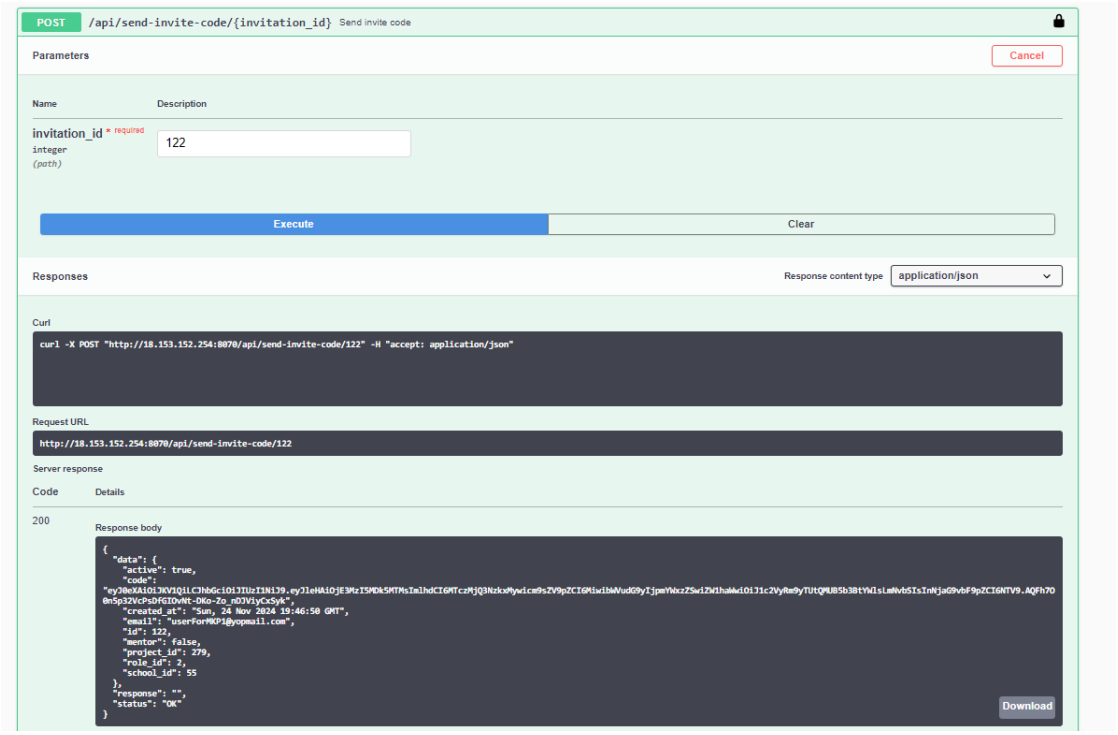

Рисунок 3.6 – Відправка запрошення

Останнім етапом є реєстрація аккаунту. Для цього потрібно ввести наступну інформацію: birthday, email, password, nickname, school_id, name, surname, grade, discipline id, language. Результат даного запиту зображений на рис. 3.7.

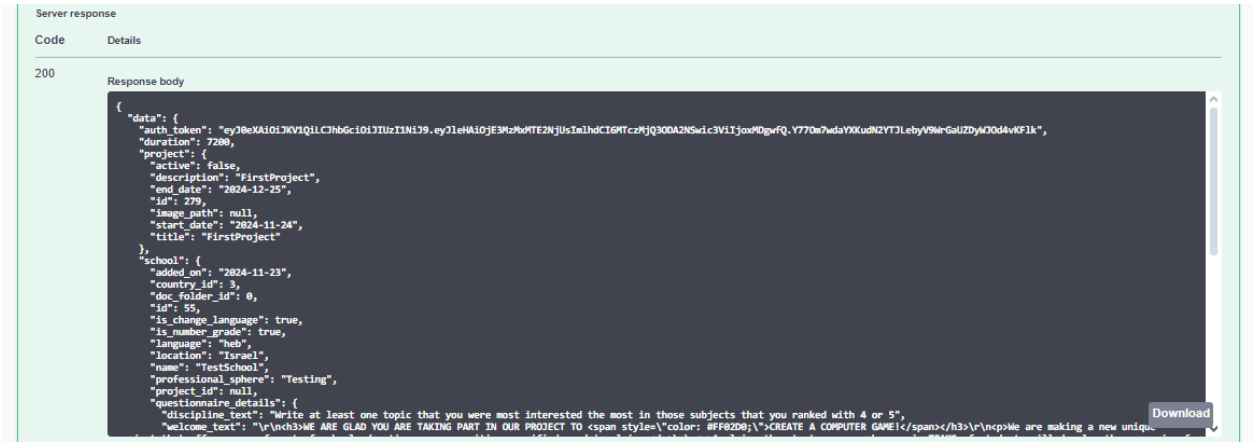

Рисунок 3.7 – Реєстрація аккаунту

Перевірити те, що user успішно зареєстрований можна за допомогою запиту GET/users/user_id на рис. 3.8.

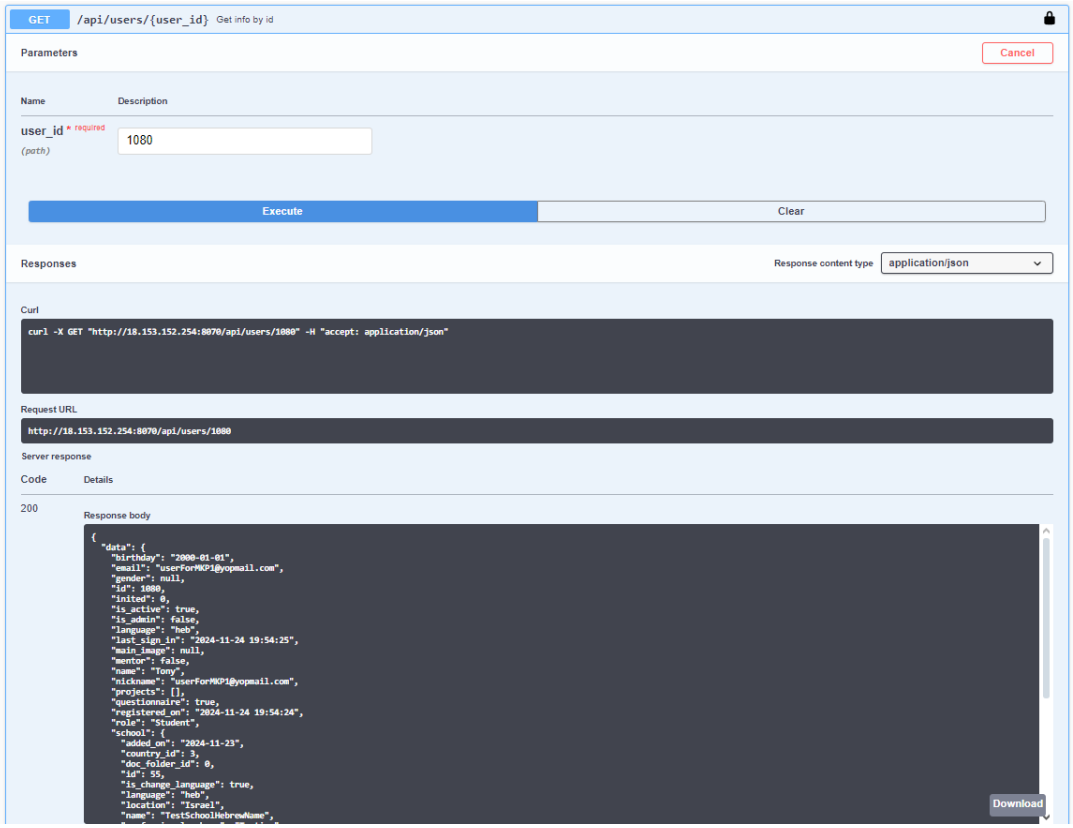

Рисунок 3.8 – Перевірка інформації про user

Наступним етапом реєстрації є проходження анкети користувача. Для цього нам потрібно виконати запит PUT/questionnaire/user_id та ввести відповідну інформацію.

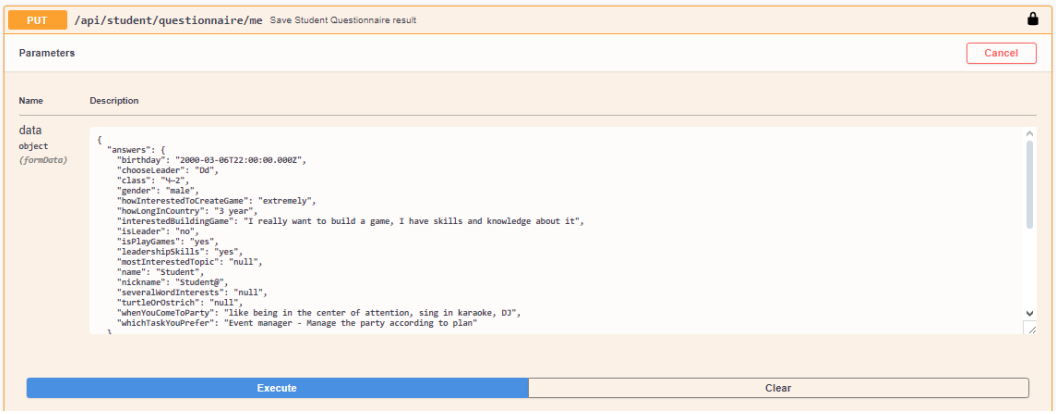

Рисунок 3.9 – Введення інформації в анкету

Відправляємо запит за допомогою кнопки «Execute» та дивимось на відгук сервера на рис. 3.10.

Рисунок 3.10 – Відповідь сервера

Анкета була успішно пройдена. Для перевірки можемо повернути масив даних за допомогою метода GET/questionnaire/me на рис. 3.11.

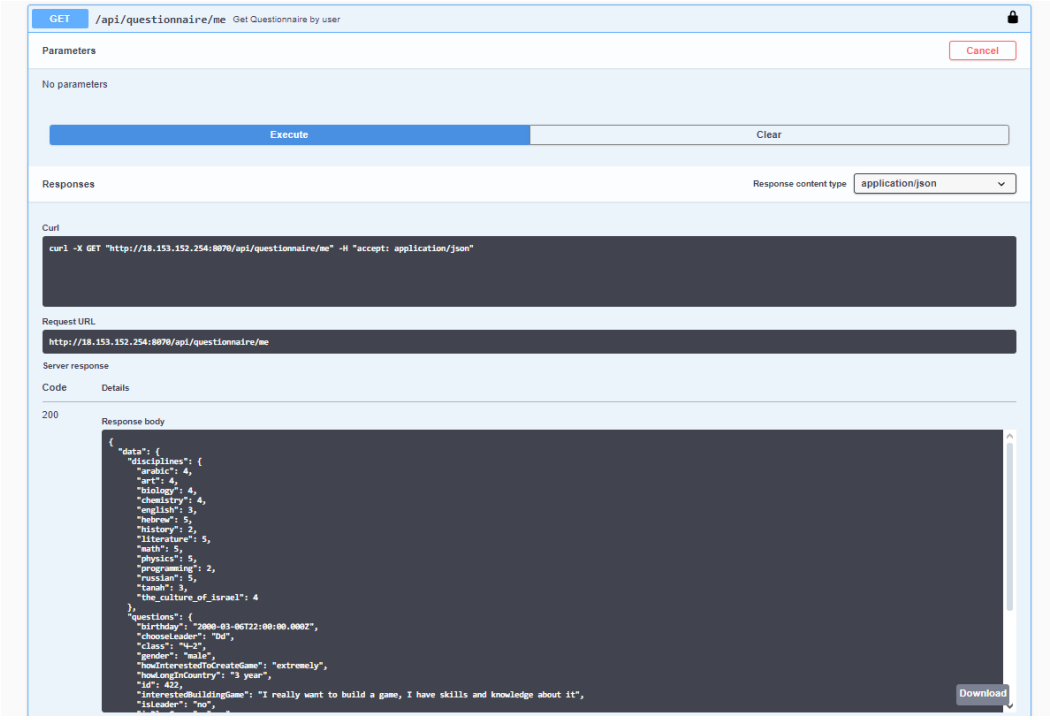

Рисунок 3.11 – Перевірка анкети

Тест №2. Генерація команди.

По аналогії до тесту №1, створюємо ще декілька аккаунтів, аби мати змогу згенерувати команду. За допомогою запиту GET/school/students отримуємо інформацію про всіх студентів школи на рис. 3.12.

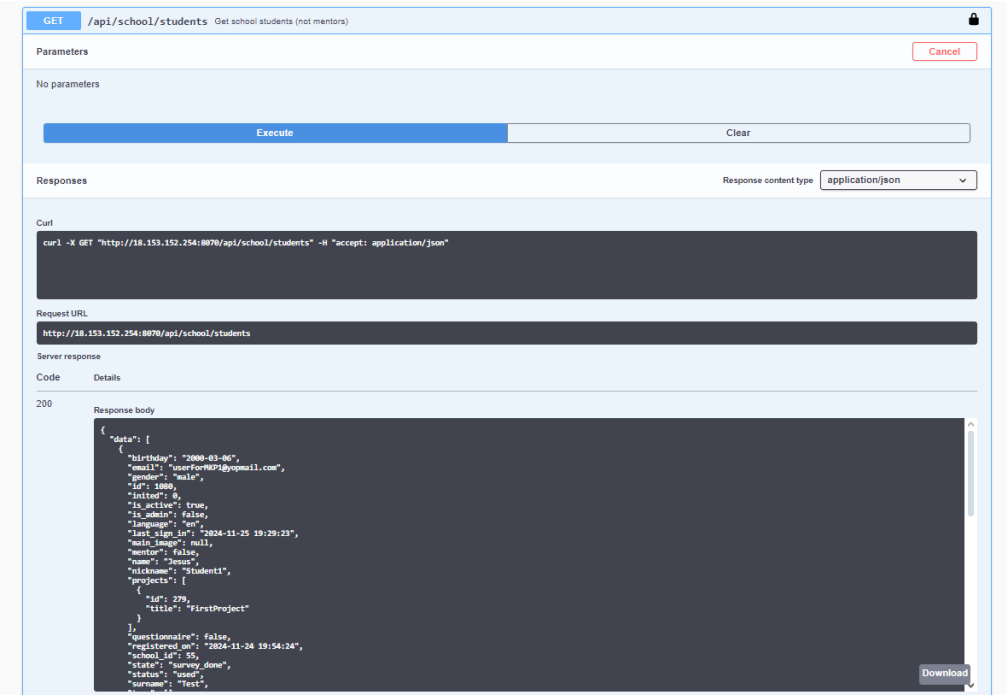

Рисунок 3.12– Всі користувачі школи

За допомогою запиту PUT/genereate-teams генеруємо наші команди вводячи project, id.

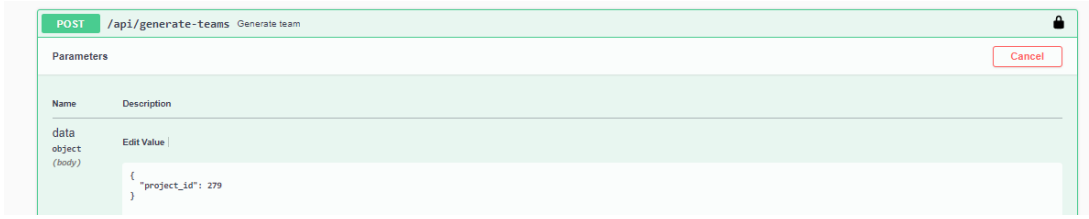

Рисунок 3.13– Генерація команди

На рис. 3.14 представлений результат генерації команди, а саме відгук сервера.

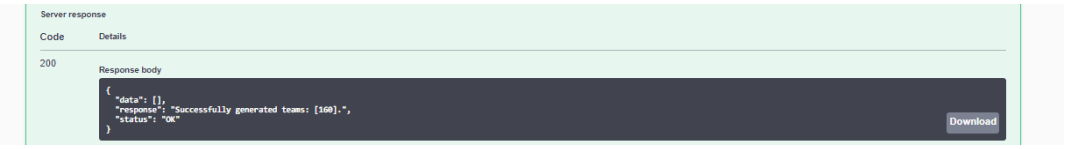

Рисунок 3.14– Результат генерація команди

За допомогою запиту GET/teams/teams отримуємо всю детальну інформацію по згенерованій команді на рис. 3.15.

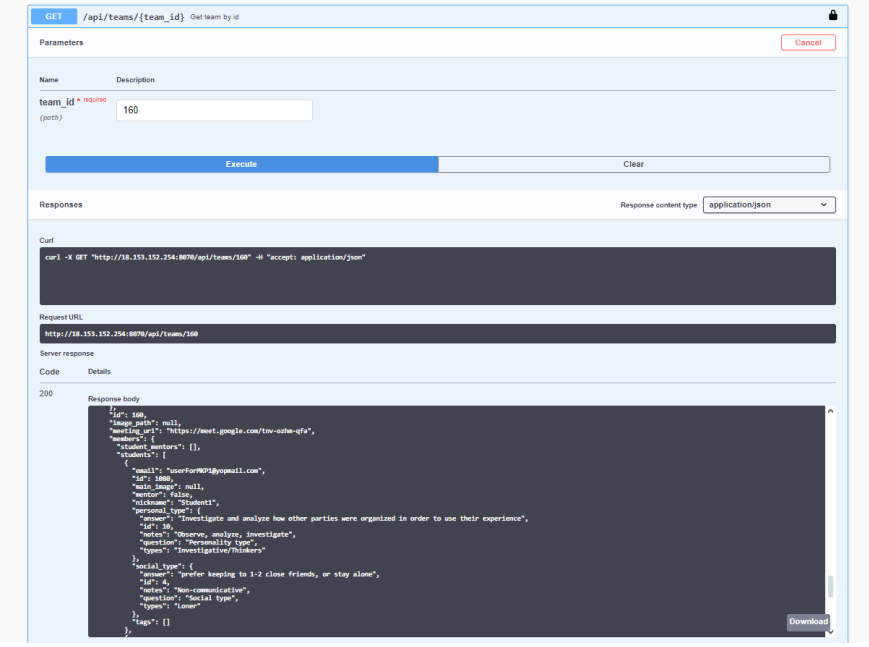

Рисунок 3.15– Інформація про згенеровану команду

Тест №3. Видалення користувача.

За допомогою запиту GET/school/students, отримав інформацію про всіх студентів школи на рис. 3.16.

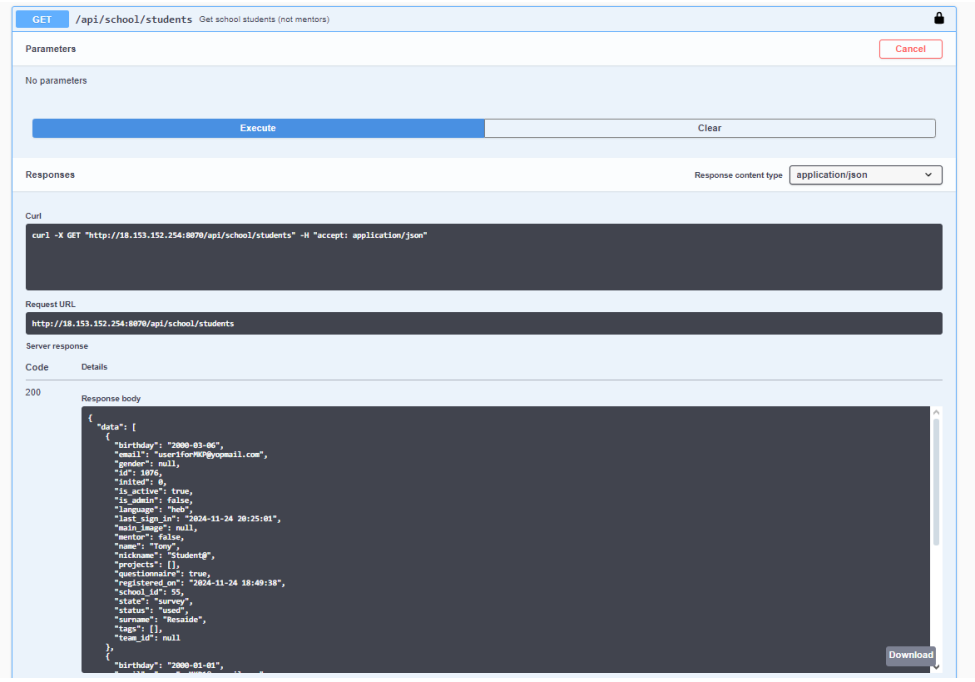

Рисунок 3.16 - Всі студенти школи

Обираємо студента, якого плануєм видалити із школи та запам’ятовуємо його email або його id.

Використовуючи запит DELETE/user_id, видаляємо профіль користувача.

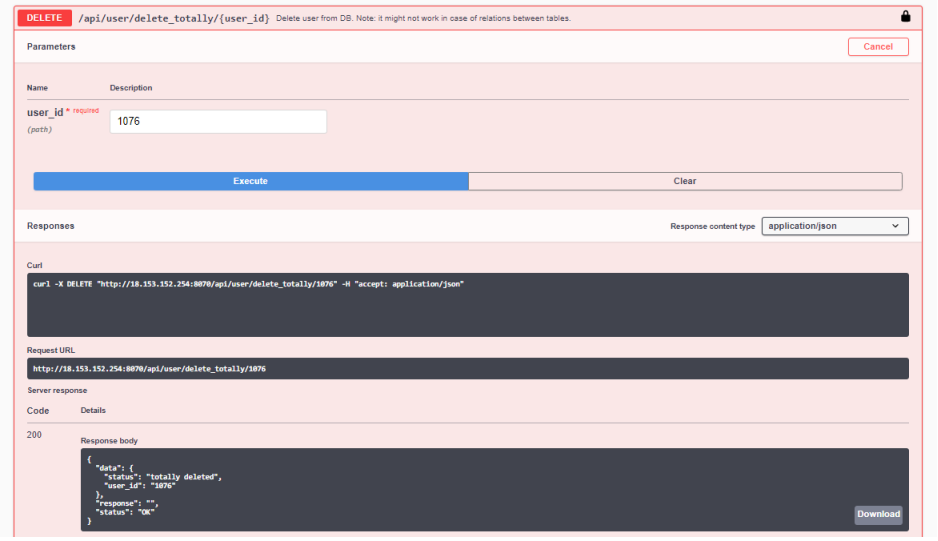

Рисунок 3.17 - Видалення студента школи

Тест №4. Зміна інформації користувача.

За допомогою запиту GET/users/user_id дізнаємось всю наявну інформацію про користувача на рис. 3.18.

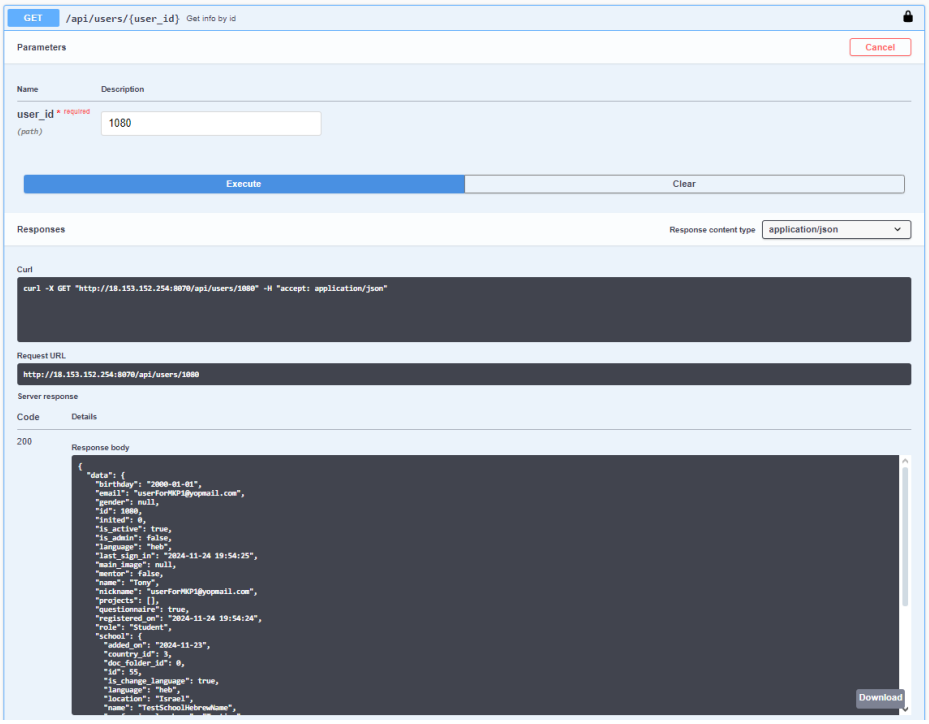

Рисунок 3.18 - Отримання інформації про користувача

Вибираємо відповідні дані, які ми хочимо змінити. Для прикладу візьмемо ім’я користувача «Tony», а ми хочемо його замінити на «Jesus». Для цього використаємо запит PUT/update/user_id та внесемо відповідні зміни.

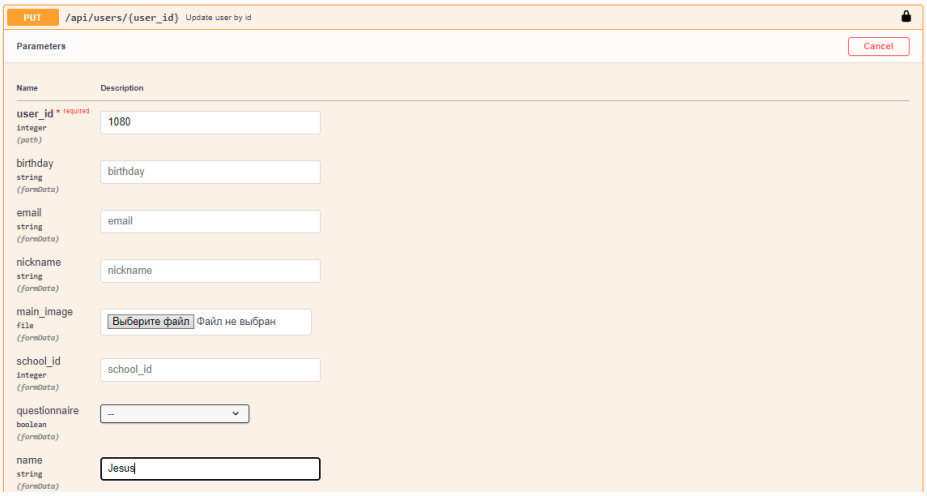

Рисунок 3.19 - Введення нової інформації

На рис. 3.20 представлений результат запиту на зміну ім’я:

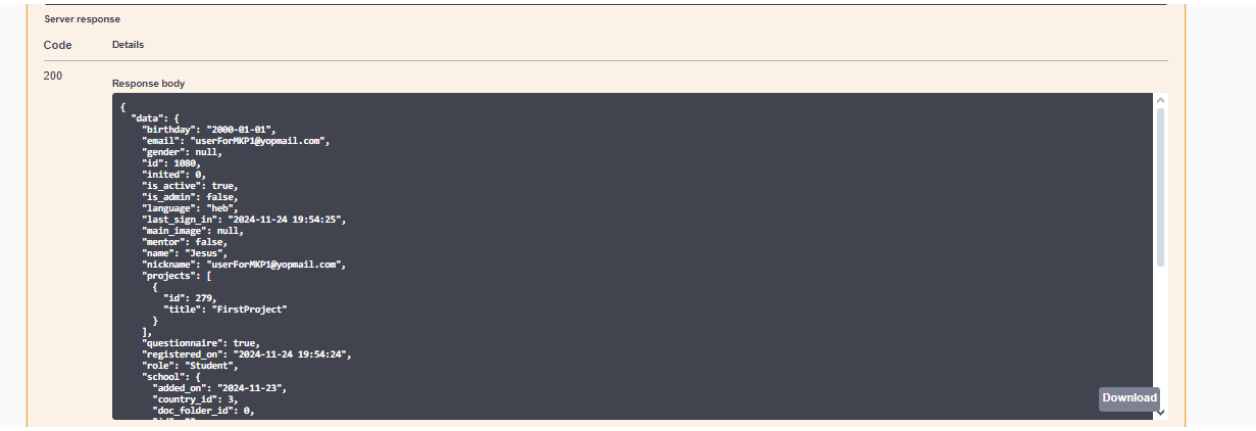

Рисунок 3.20 - Результат запиту на зміну ім’я

Перевіряємо інформацію ще раз за допомогою запиту GET/users/user_id на рис. 3.21.

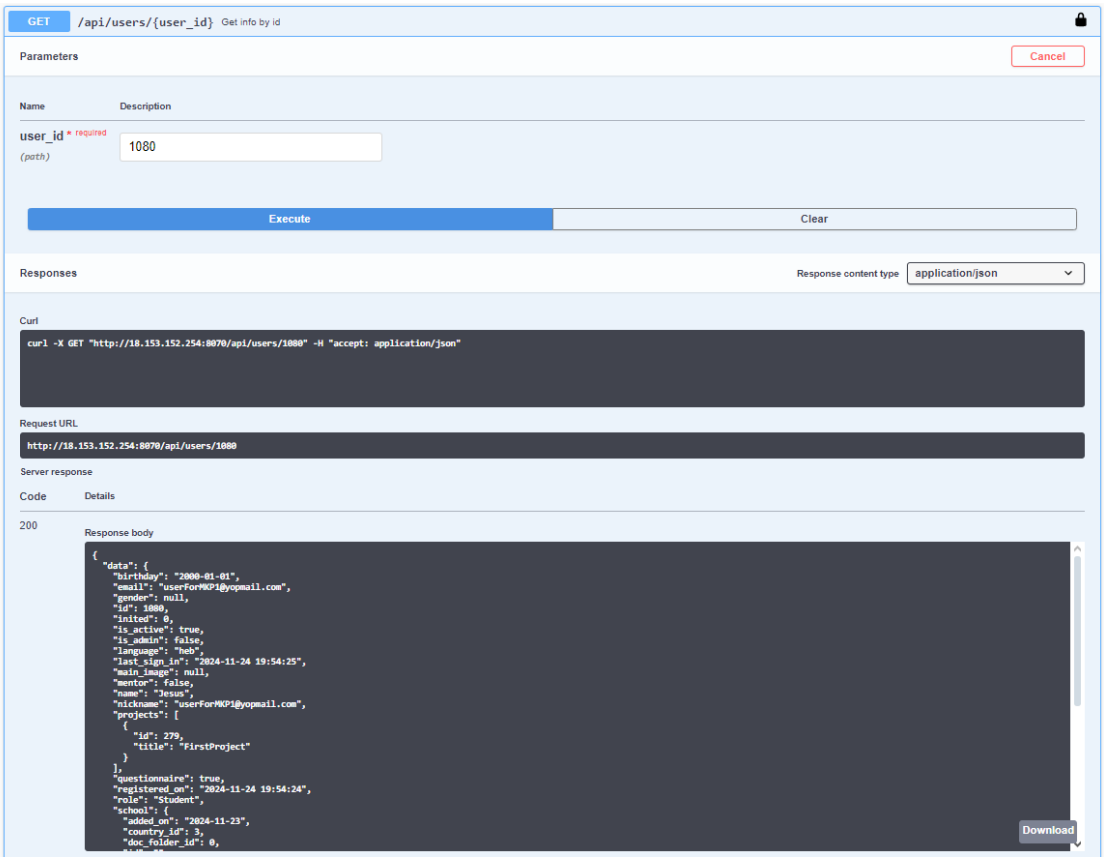

Рисунок 3.21 - Перевірка результату

Все правильно, ім’я змінилось на «Jesus».

Після завершення тестування серверної частини за допомогою Swagger та впевненості у коректній роботі API, наступним кроком є перевірка взаємодії бекенду з інтерфейсом користувача (UI). Це дозволить оцінити, наскільки успішно серверна логіка інтегрується з клієнтським додатком, забезпечуючи коректну передачу даних, виконання бізнес-логіки та належний користувацький досвід. Для перевірки візьмемо попередній набір тестів.

Тест 1. Реєстрація аккаунту

В першу чергу потрібно створити запрошення (invitation) на його email та відправити. Результати зображені на рис. 3.22.

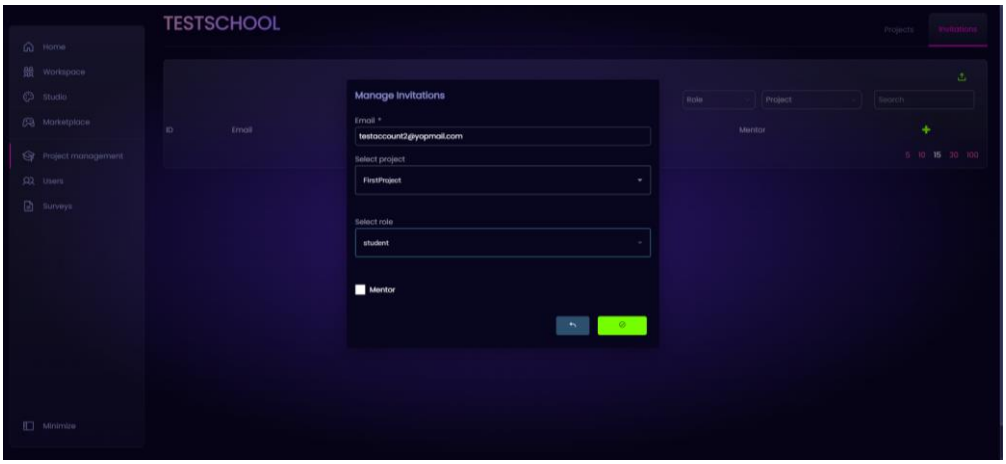

Рисунок 3.22 - Створення запрошення

Після переходу по запрошенню, яке прийшло нам на email, в нас з’являється наступна сторінка де потрібно придумати та підтвердити свій пароль. Результати зображені на рис. 3.23.

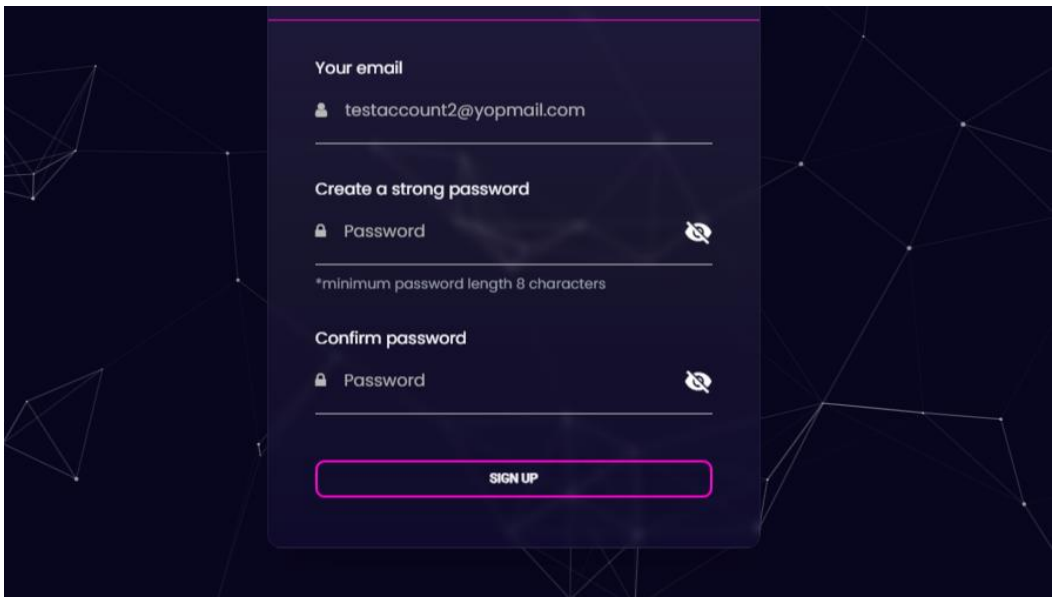

Рисунок 3.23 - Створення паролю

Далі потрібно ввести основну інформацію про користувача: nickname, class, birthday, sex. Також одразу можна завантажити аватар. Результати зображені на рис. 3.24.

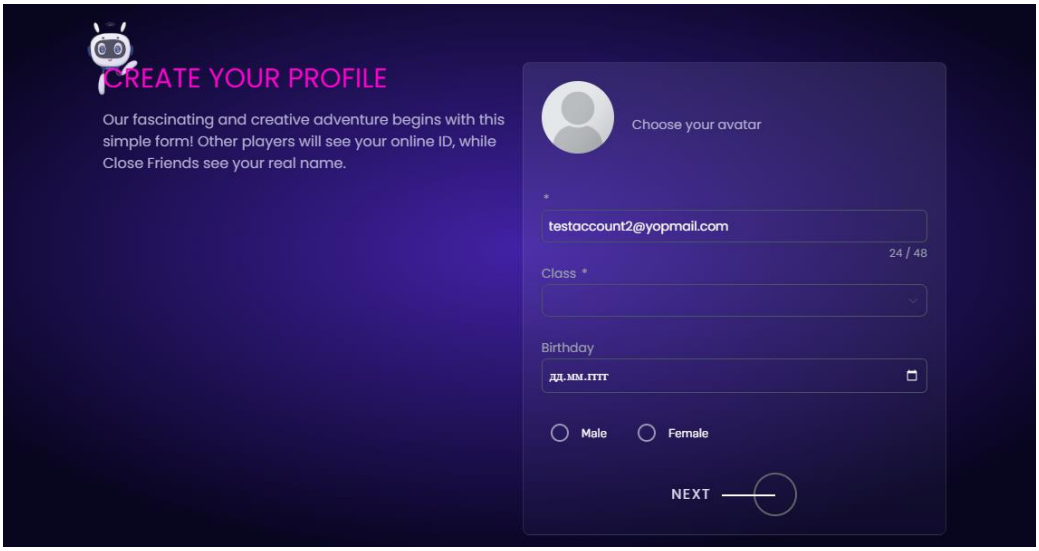

Рисунок 3.24 - Введення основної інформації

Наступним етапом у нас одразу відкривається анкета, яку потрібно обов’язково пройти, аби мати доступ до сервісу. Спочатку анкета починається із питань про оцінки, які зображені на рис. 3.25, а потім із питань, щоб визначити темперамент учня, які зображені на рис. 3.26.

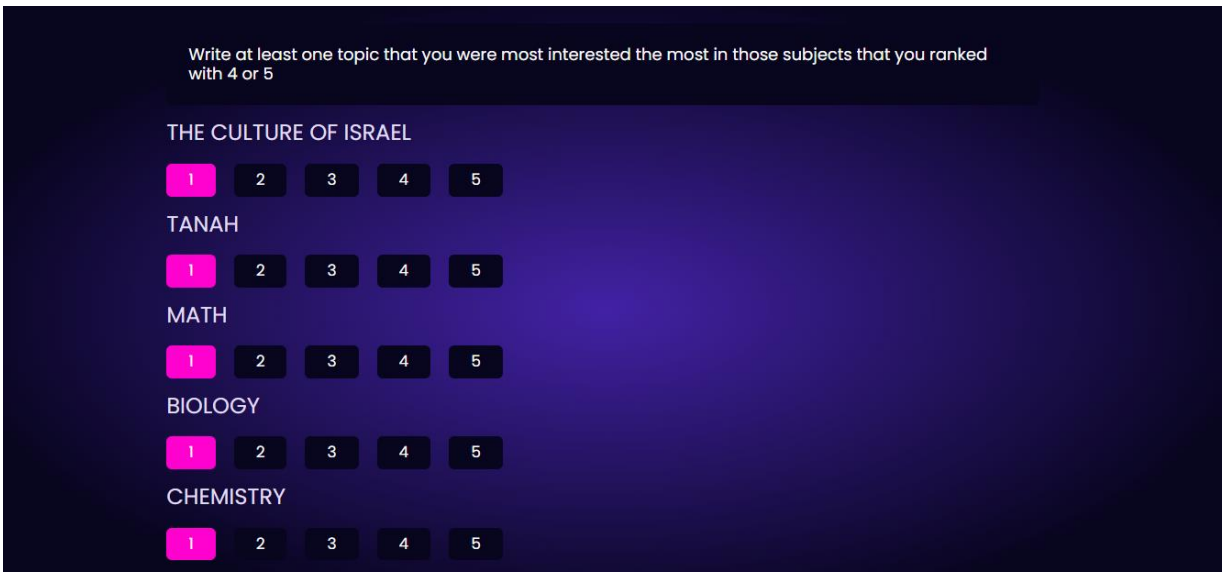

Рисунок 3.25 - Проходження анкети. Частина 1

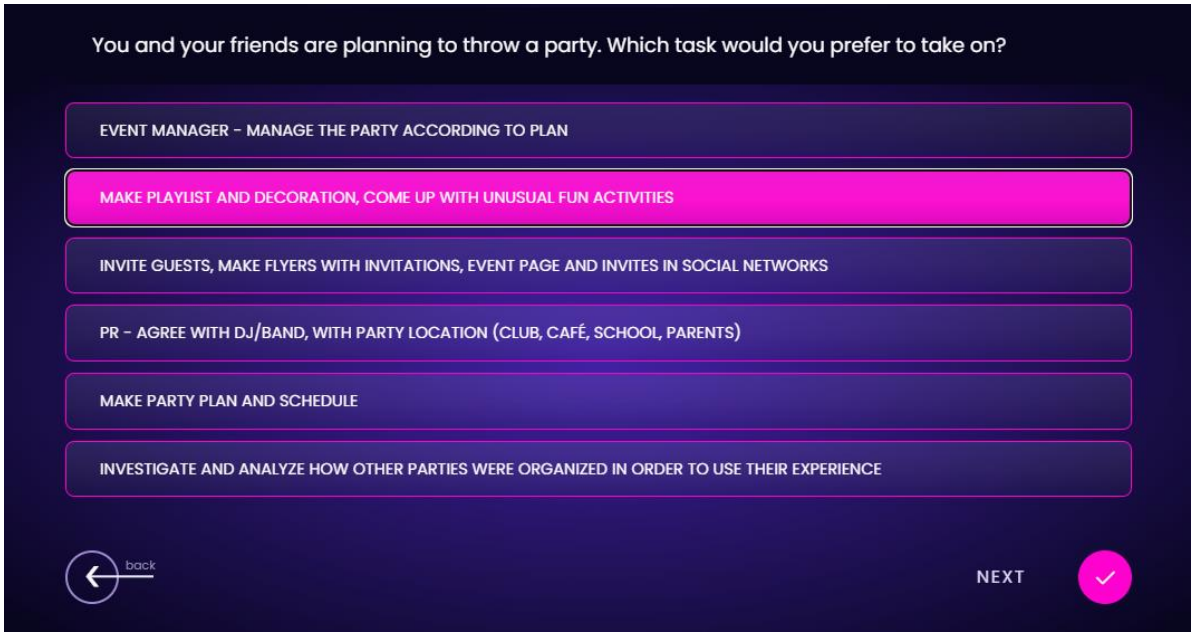

Рисунок 3.26 - Проходження анкети. Частина 2

В результаті ми отримуємо наступний екран з коротенькими висновками по анкеті, а також кнопкою, щоб перейти до основного функціоналу сайту, який зображений на рис. 3.27.

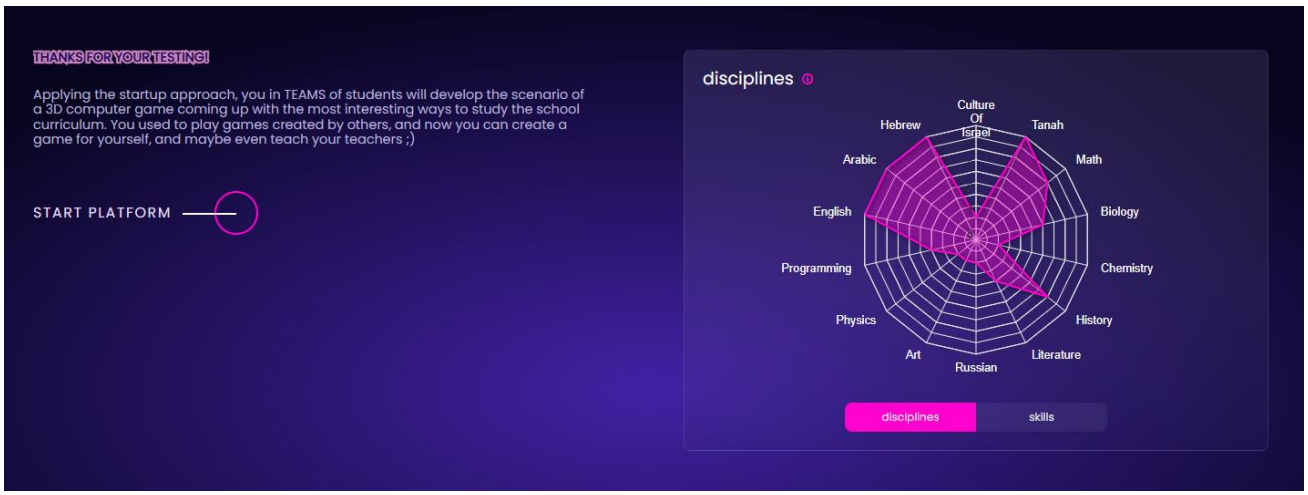

Рисунок 3.27 - Кінцевий екран реєстрації

Тест 2. Генерація команди

За аналогією з тестом №1 створюємо ще кілька облікових записів для користувачів, які зображені на рис. 3.28.

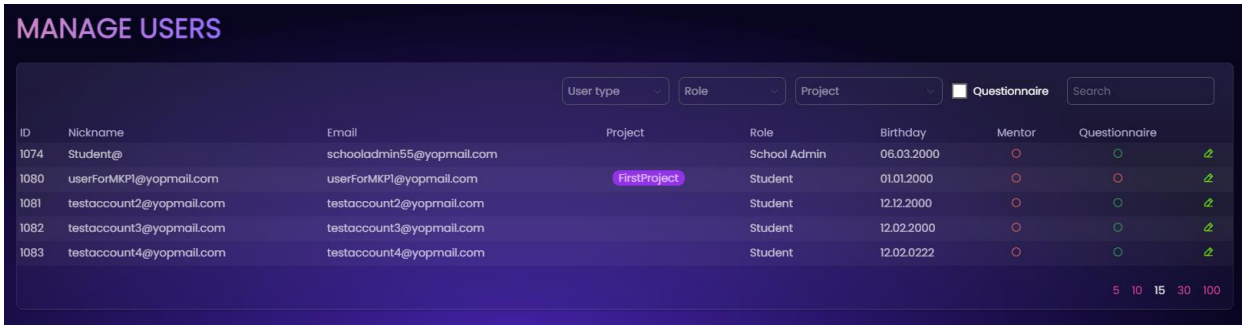

Рисунок 3.28 - Створення групи користувачів

Далі потрібно переміститись на сторінку «Workspace» і натиснути на кнопку генерації команд , тобто на «+». В модальному вікні потрібно обрати проект і натиснути «Submit», яка зображена на рис. 3.29.

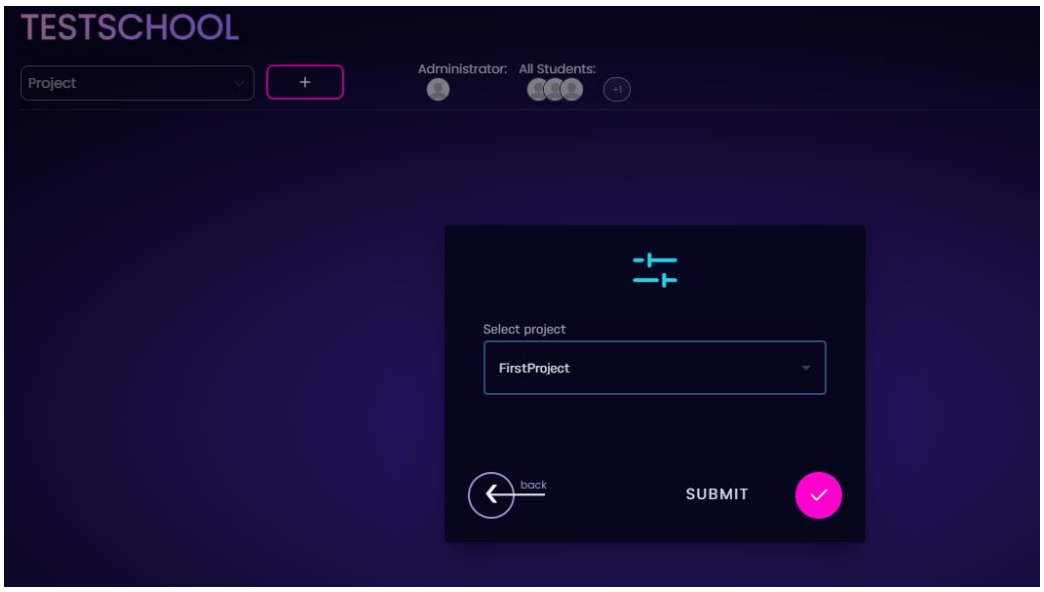

Рисунок 3.29 - Генерація команди

Результатом генерації є створена сторінка команди з її членами та основними характеристиками, логотипами та ролями, яка зображена на рис. 3.30.

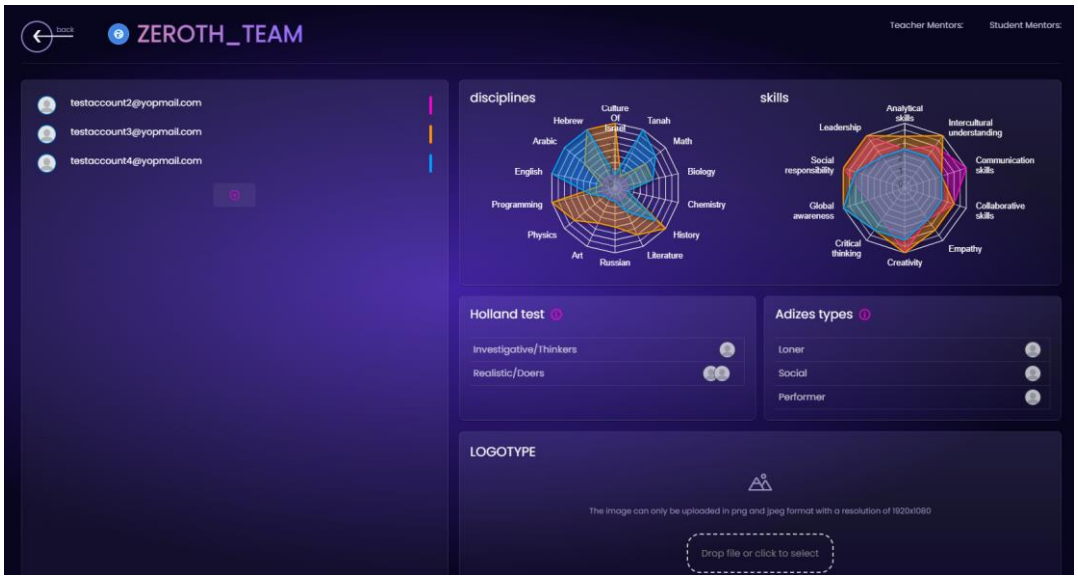

Рисунок 3.30 - Сторінка згенерованої команди

Результатом виконання даного підпункту є тестування функціональності API за допомогою Swagger, що дозволило побачити коректну взаємодію з базою даних і клієнтською частиною. Після цього виконано інтеграційне тестування серверної частини з користувацьким інтерфейсом (UI) для оцінки їх сумісності.

ВИСНОВКИ

У результаті виконання даної магістерської кваліфікаційної роботи було досягнуто основну мету — створення серверної частини клієнт-серверної системи, яка автоматизує процес формування команд учнів для онлайн-навчання.

На основі проведеного аналізу предметної області визначено основні проблеми, пов'язані з організацією групової роботи в освітньому процесі, та переваги автоматизованого підходу до формування команд.

Під час роботи було здійснено огляд існуючих систем і рішень, що дозволило чітко сформулювати вимоги до розробки. Вибір архітектури та технологій забезпечив модульність, масштабованість і зручність інтеграції системи в різні освітні процеси. Розробка програмного забезпечення включала створення функціоналу для автоматичного формування команд, управління курсами та проектами, а також забезпечення безпеки даних і зберігання навчальних матеріалів. Тестування системи підтвердило її працездатність і відповідність встановленим вимогам.

Практична цінність роботи полягає в тому, що створена система сприяє ефективності організації онлайн-навчання, автоматизуючи рутинні задачі викладачів і забезпечуючи збалансоване формування навчальних команд. Це дозволяє покращити взаємодію між учасниками освітнього процесу, враховуючи їхні індивідуальні особливості.

ПЕРЕЛІК ПОСИЛАНЬ

1. Амеліна О., Цуркан О. Дистанційне та змішане навчання. Досвід, поради, інструменти – Київ «Основа», 2022 – 128с.

2. Шарова Т. М. Освітній портал як ефективний засіб забезпечення дистанційного навчання здобувачів вищої освіти. Українські студії в європейському контексті. 2022. № 5. С. 237–244

3. Іщенко П.С., Коцюбинський В.Ю. Автоматизація інтеграції ігрових елементів у системах навчання в онлайн середовищі: доповідь [Електронний ресурс] – Вінниця: ВНТУ, 2024. URL: http://surl.li/sibwv

4. Цапар В.С., Жученко О.А. Сучасні програмні засоби автоматизації технологічних процесів / НТУУ «КПІ» - Київ : НТУУ «КПІ», 2012.

5. Триус Ю.В., Герасименко І.В. Комбіноване навчання як інноваційна освітня технологія у вищій школі. Теорія та методика електронного навчання : зб. наук. праць. Випуск ІІІ. Кривий Ріг, 2012. С. 299–308.

6. Трегуб В.Г. Проектування систем автоматизації: навчальний посібник – Київ: «Ліра К», 2019., ISBM: 978-966-2609-58-5

7. Розробка ПЗ для навчальної організації / Evergreen - web розробка і діджиталізація бізнесу за допомогою AI продуктів [Електронний ресурс] – URL: https://evergreens.com.ua/ua/articles/education-system.html

8. Зайченко І.В. Педагогіка [Електронний ресурс] – URL: https://pidru4niki.com/17000308/pedagogika/pedagogika

9. Концепція нової української школи. Офіційний сайт Міністерство освіти і науки України. [Електронний ресурс]. — URL: https://mon.gov.ua/storage/app/media/zagalna%20serednya/nova-ukrainska-shkola-compressed.pdf

10. Горбунова В. В. Психологія командотворення: Ціннісно-рольовий підхід до формування та розвитку команд : монографія. – Житомир : Вид-во ЖДУ ім. І. Франка, 2014. – 380 с., іл. ISBN 978-966-485-162-3

11. Байбуз О.Г. Актуальні проблеми автоматизації та інформаційних технологій: збірник наукових праць - Дніпро: ДНУ, 1997., ISSN 2312-119X.
12. Митрошина Н. Доповнена та віртуальна реальність [Електронний ресурс] – URL: http://surl.li/ncgww
13. Системи штучного інтелекту: навч. посіб. / Ю. В. Нікольський, В. В. Пасічник, Ю. М. Щербина; за наук. ред. В. В. Пасічника; М-во освіти і науки, молоді та спорту України. — 2-ге вид. — Львів: Магнолія-2006, 2013. — 279 с.: іл.— ISBN 978-617-57-40-11-4.
14. Системи дистанційного навчання: найкращі LMS у 2020 році / Evergreen - web розробка і діджиталізація бізнесу за допомогою AI продуктів [Електронний ресурс] – URL: https://evergreens.com.ua/ua/articles/best-lms-2020.html
15. Офіційний сайт Quizizz [Електронний ресурс] – URL: https://quizizz.com/home-v1.
16. Офіційний сайт Classcraft [Електронний ресурс] – URL: https://www.classcraft.com
17. Бевз О. М., Папінов В. М., Скидан Ю. А. Проектування програмних засобів систем управління: навчальний посібник – Вінниця : ВНТУ, 2010. – 125 с
18. Глобальні інформаційні системи та технології: моделі ефективного аналізу, опрацювання та захисту даних. Монографія / В.В.Пасічник, П. І. Жежнич, Р. Б. Кравець, А. М. Пелещишин, Д. О. Тарасов – Львів: Видавництво Львівської політехніки, 2006. – 348 с. ISBN: 966-553-578-1.
19. Семков Д. Розуміння Клієнт-Серверної Архітектури на прикладах [Електронний ресурс] – URL: https://dou.ua/forums/topic/44636/
20. Бунке О.С. Серверні Web-Технології: навчальний посібник – Київ: КПІ ім. Ігоря Сікорського, 2023. – 109 с.
21. Мови програмування: список сучасних мов, з якої мови почати вивчення програмування - Ukrainian Digital Community [Електронний ресурс] – URL: ukrainiandigital.com/strong-movy-prohramuvannia-dlia-pochatkivtsiv-ta-dosvidchenykh-ohliad-populiarnykh.

22. Колмогоров Н. Все, що вам потрібно знати про Node.js [Електронний ресурс] – URL: https://medium.com/webbdev/js-db3d35ffed7e

23. Колесніков Д. Що таке фреймворк: пояснюємо простими словами [Електронний ресурс] – URL: https://brainlab.com.ua/uk/blog-uk/shho-take-frejmvork-poyasnyuyemo-prostymy-slovamy

24. Лосєв М. Ю., Федько В. В. Бази даних : навчально-практичний посібник для самостійної роботи студентів [Електронний ресурс] – Харків : ХНЕУ ім. С. Кузнеця, 2018. – 233 с. ISBN 978-966-676-731-1

25. Заболоцький А. Ю. Сучасний стан дистанційного навчання у ВНЗ України. Вісник Дніпропетровського ун-ту ім. А. Нобеля. «Педагогіка і психологія». Педагогічні науки. 2016. № 2 (12). С. 19-23.

26. Центр дистанційного навчання Університет економіка та права «КРОК». URL: https://www.krok.edu.ua/ua/pro-krok/pidrozdili/navchalni/tsentr-distantsijnogo-navchannya.

27. Littlefield J. 10 Reasons to Choose an Online Education to Earn Your Degree. ThoughtCo. URL: https://www.thoughtco.com/reasons-to-choose-online-education-1098006

28. Ягупов В. В. Педагогіка: навч. посібник. - Київ: Либідь. 2002. 560 с.

29. Герасименко І. В. Використання технологій дистанційного навчання в підготовці майбутніх бакалаврів комп’ютерних наук. Інформаційні технології і засоби навчання. 2014. URL: http://journal.iitta.gov.ua/index.php/itlt/article/view

30. Спірін О. М., Лупаренко Л. А. Досвід використання програмної платформи Open Journal Systems для інформаційно-комунікаційної підтримки науково-освітньої діяльності. Інформаційні технології і засоби навчання. 2017. Т. 61. Вип.5. С. 196-218. URL : http://www.irbis-nbuv.gov.ua/cgi

31. Романовський О. Г., Квасник О. В., Мороз В.М., Підбуцька Н. В., Резнік С. М., Черкашин А. І., Шаповалова В. В. Фактори розвитку та напрями вдосконалення дистанційної освіти навчання в системі вищої освіти України. Інформаційні технології та засоби навчання. 2019. Т. 74. №6. 37 с.

32. Шарова Т. М. Освітній портал як ефективний засіб забезпечення дистанційного навчання здобувачів вищої освіти. Українські студії в європейському контексті. 2022. № 5. С. 237–244
33. Павлова Н.С. Підготовка вчителя до роботи з е-щоденниками та ежурналами. Українські студії в європейському контексті. 2022. № 5.С. 201–204.
34. Кривонос І. О. Особливості використання інформаційних технологій в освітній діяльності здобувачів освіти. Українські студії в європейському контексті. 2022. №5. С. 183–189.
35. Наливайко О. Дистанційне навчання: сутність та особливості. Педагогічний альманах : збірник наукових праць - Херсон : КВНЗ «Херсонська академія неперервної освіти», 2017. Випуск 36. С. 75–81
36. STEM-освіти в умовах інтеграції формальної і неформальної освіти обдарованих учнів: методичні рекомендації / Н. І. Поліхун, К. Г. Постова, І. А. Сліпухіна, Г. В. Онопченко, О. В. Онопченко. – Київ : Інститут обдарованої дитини НАПН України, 2019. – 80 с
37. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, Incorporated, 2022.
38. Saddleback Educational Saddleback Educational Publishing. Science and Technology Words. Saddleback Educational Publishing, Incorporated, 2010. 125 с.
39. Pais M., Skelton M. Team Topologies: Organizing Business and Technology Teams for Fast Flow. IT Revolution Press, 2019.
40. Дубовой В. М. Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спеціальності 151 – Автоматизація та комп’ютерно-інтегровані технології – Вінниця : ВНТУ, 2022 – 38с

ДОДАТКИ

ДОДАТОК А
(обов'язковий).

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: « Розробка клієнт-серверної системи для автоматизації процесу формування команд учнів для навчання онлайн. Частина 1. Серверна частина »

Тип роботи: магістерська кваліфікаційна робота

(кваліфікаційна робота, курсовий проект (робота), реферат, аналітичний огляд, інше (зазначити))

Підрозділ кафедра комп'ютерних систем управління

(кафедра, факультет (інститут), навчальна група)

Керівник Коцюбинський В.Ю., доцент кафедри АІІТ

(прізвище, ініціали, посада)

Показники звіту подібності

Turnitin	
Оригінальність	96%
Загальна схожість	4%

Аналіз звіту подібності (відмітити потрібне)

√ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений (-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ Павло ІЩЕНКО
(підпис) (прізвище, ініціали)

Опис прийнятого рішення

Особа, відповідальна за перевірку _____ Володимир ДУБОВОЙ
(підпис) (прізвище, ініціали)

Експерт _____ _____
(за потреби) (підпис) (прізвище, ініціали, посада)

ДОДАТОК Б

(обов'язковий).

Технічне завдання

ВНТУ

ЗАТВЕРДЖЕНО
Зав. кафедри КСУ ВНТУ,
д.т.н., професор
_____В'ячеслав КОВТУН
" 14 " 10 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

Розробка клієнт-серверної системи для автоматизації процесу формування команд учнів для навчання онлайн. Частина 1. Серверна частина.

08-33.МКР.15.03.000 ТЗ

Студент групи ЗАКІТР-23м
_____ Павло ІЩЕНКО

Керівник к.т.н., доцент каф. АІІТ
_____ Володимир КОЦЮБИНСЬКИЙ

Вінниця 2024

1. Назва та галузь застосування

1.1. Назва – Розробка клієнт-серверної системи для автоматизації процесу формування команд учнів для навчання онлайн. Частина 1. Серверна частина.

1.2. Галузь застосування – освітній процес.

2. Підстава для проведення розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ від №310 від 17.09.2024р.

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи підвищення ефективності організації онлайн-навчання.

4. Джерела розробки.

Магістерська кваліфікаційна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1. Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спеціальності 151 – Автоматизація та комп’ютерно-інтегровані технології / Уклад. В. М. Дубовой, – Вінниця : ВНТУ, 2022 – 38с.
2. Дистанційне та змішане навчання. Досвід, поради, інструменти / Олена Амеліна, Олександр Цуркан – Київ «Основа», 2022 – 128с.
3. STEM-освіти в умовах інтеграції формальної і неформальної освіти обдарованих учнів: методичні рекомендації / Н. І. Поліхун, К. Г. Постова, І. А. Сліпухіна, Г. В. Онопченко, О. В. Онопченко. – Київ : Інститут обдарованої дитини НАПН України, 2019. – 80 с

5. Вимоги до розробки.

5.1. Перелік головних функцій:

- реєстрація аккаунту;
- проходження анкети;
- створення команди.

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:

- Windows 10/11;
- Visual Studio IDE;
- Браузер (Chrome/Opera/Edge).

5.2.2. Умови експлуатації системи:

- робота на мобільних та веб-додатках;
- можливість цілодобового функціонування системи;
- дані оновлюються і є актуальними.

6. Стадії та етапи розробки.

6.1 Пояснювальна записка:

1. Аналіз методів, принципів, підходів і засобів реалізації задачі автоматизації процесами в об’єкті управління відповідно до теми кваліфікаційної роботи. Постановка задач дослідження «03»_09_ 2024 р.
2. Удосконалення технології прийняття рішень при автоматизації об’єкту управління «_01_»_10_ 2024 р.
3. Визначення технічних характеристик системи «_06_»_10_ 2024 р.
4. Розробка програмного забезпечення системи «_17_»_11_ 2024 р.

6.2 Графічні матеріали:

1. Розробка UML-діаграм системи «_24_»_10_ 2024 р.
2. Розробка моделі бази даних системи «_29_»_10_ 2024 р.
3. Тестування програмного забезпечення «_24_»_11_ 2024 р.

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до «_20_»__11__ 2024 р.

7.2. Атестація МКР здійснюється на попередньому захисті. Попередній захист магістерської кваліфікаційної роботи провести до «_1_»__12__ 2024 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК. Захист магістерської кваліфікаційної роботи провести до «7»_12_ 2024 р.

ДОДАТОК В

(обов'язковий).

Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА КЛІЄНТ-СЕРВЕРНОЇ СИСТЕМИ ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСУ ФОРМУВАННЯ КОМАНД УЧНІВ ДЛЯ НАВЧАННЯ ОНЛАЙН. ЧАСТИНА 1. СЕРВЕРНА ЧАСТИНА.

Студент групи ЗАКІТР-23м

_____ Павло ІЩЕНКО

Керівник к.т.н., доцент каф. АІІТ

_____ Володимир КОЦЮБИНСЬКИЙ

Вінниця 2024

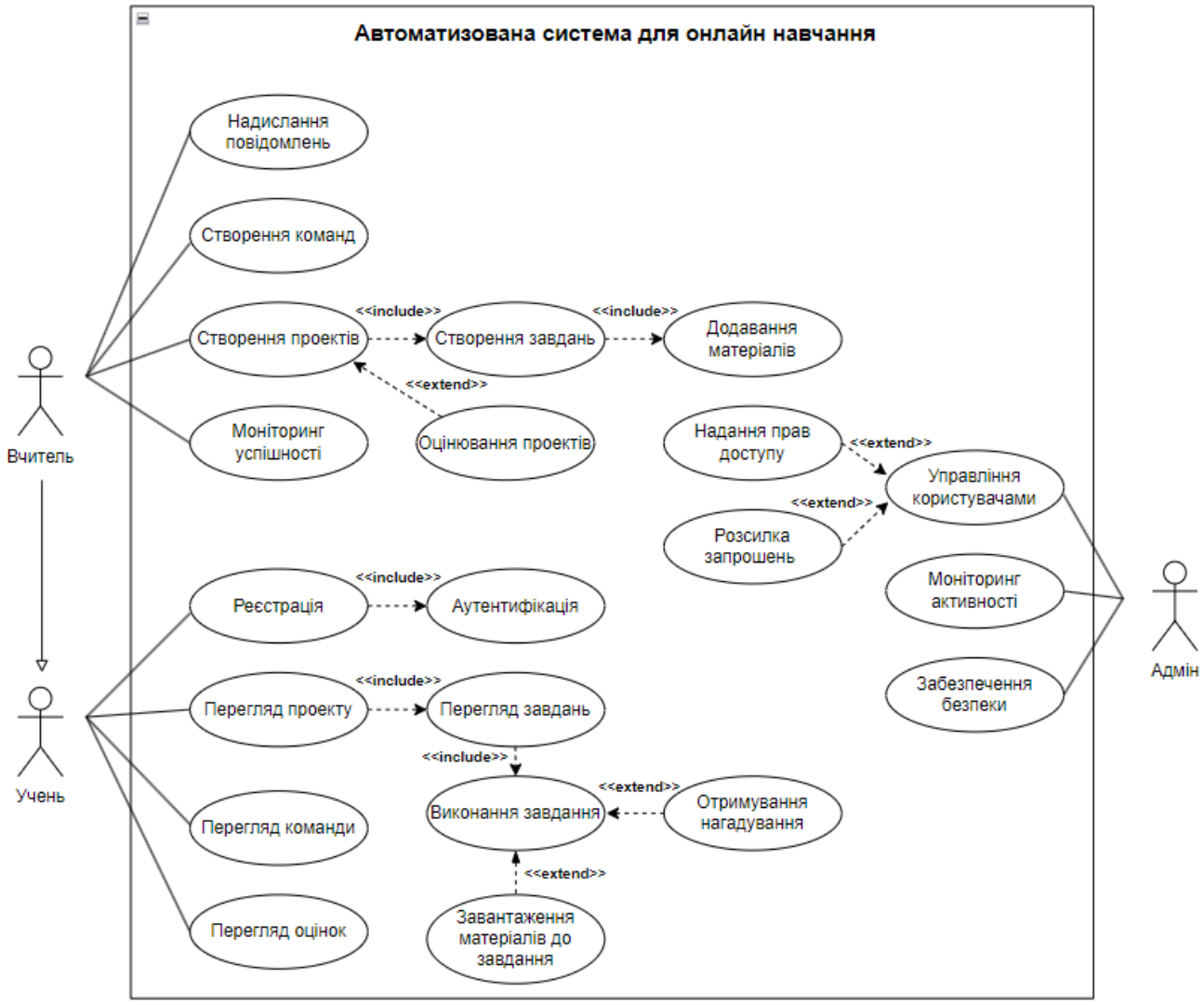

Рисунок В.1 – UML Use Case діаграма прецедентів

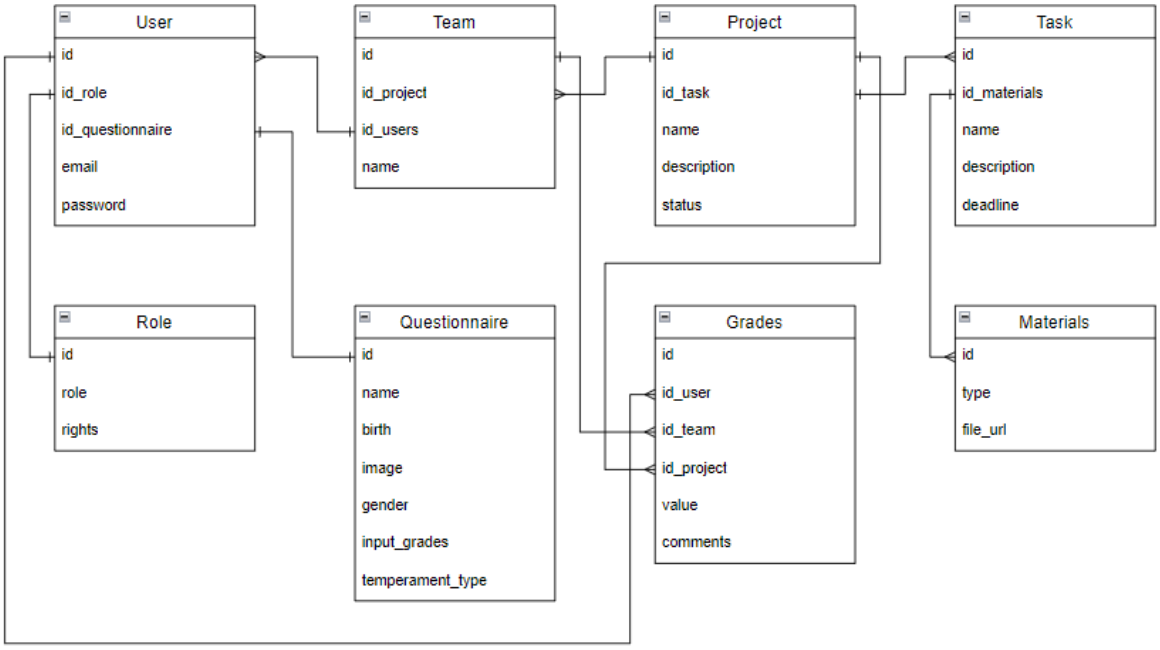

Рисунок В.2 - ER-діаграма

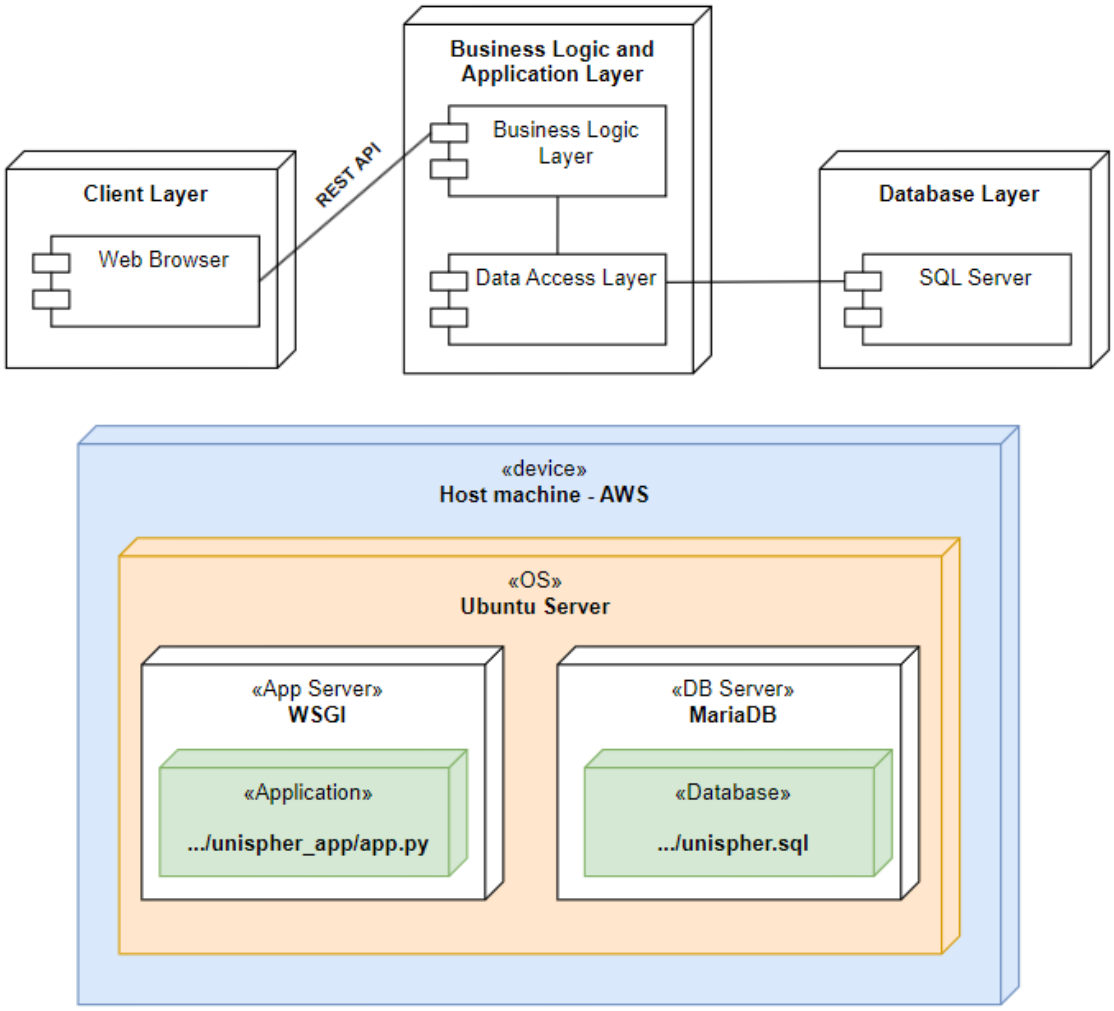

Рисунок В.3 – UML Діаграма розгортання

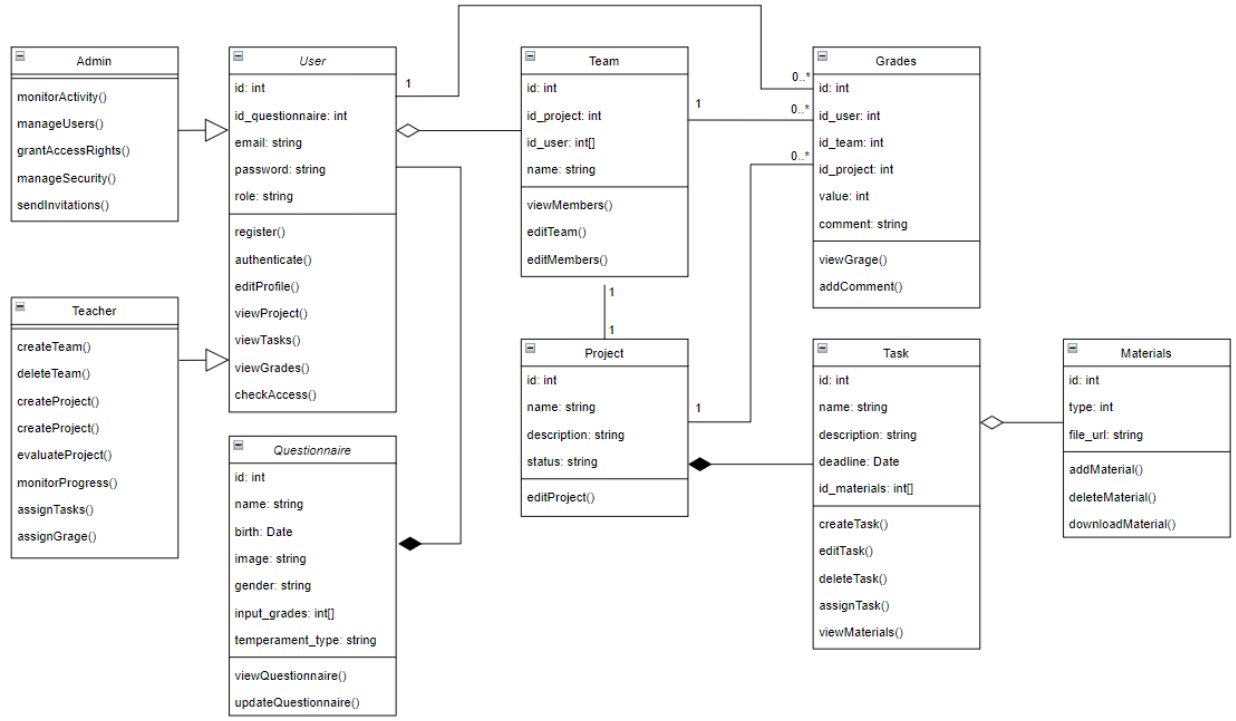

Рисунок В.4 – UML діаграма класів

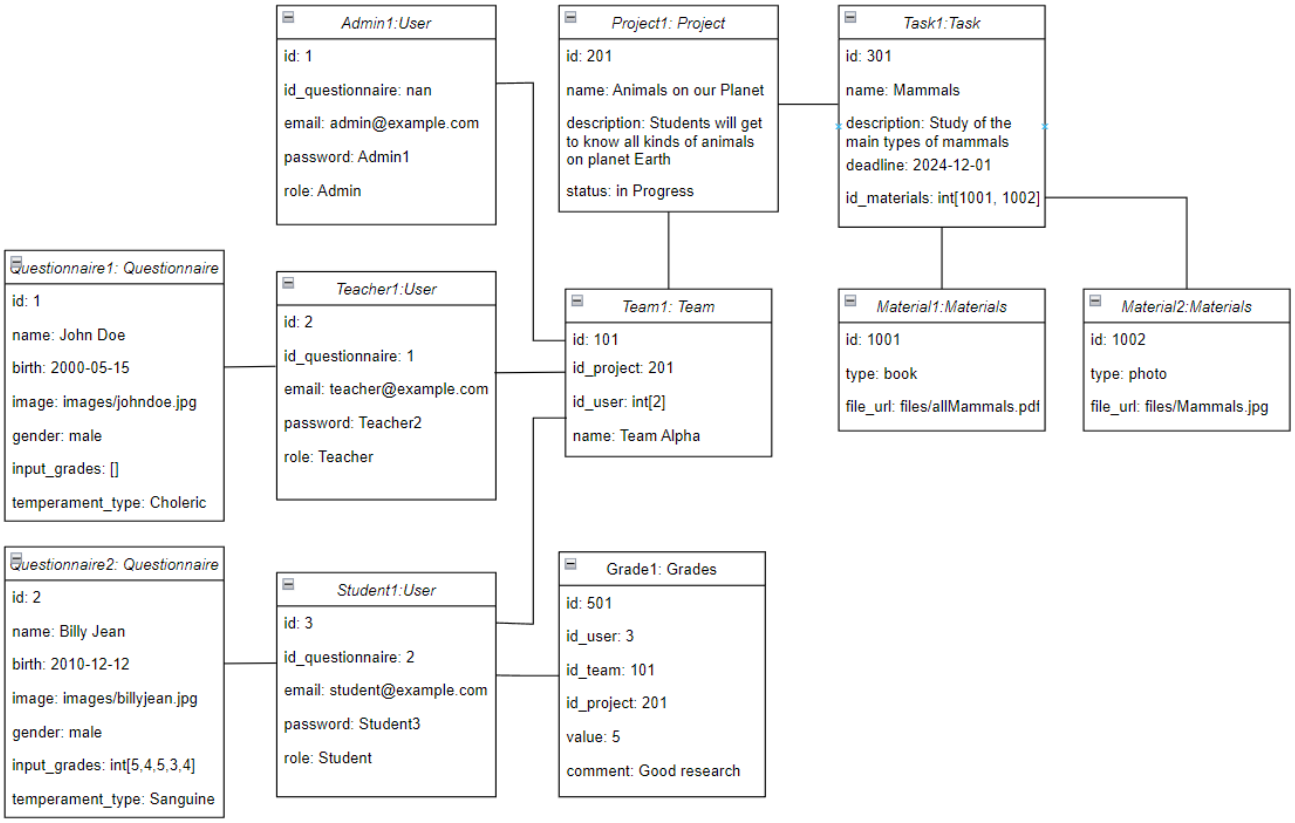

Рисунок В.5 – UML діаграма об’єктів

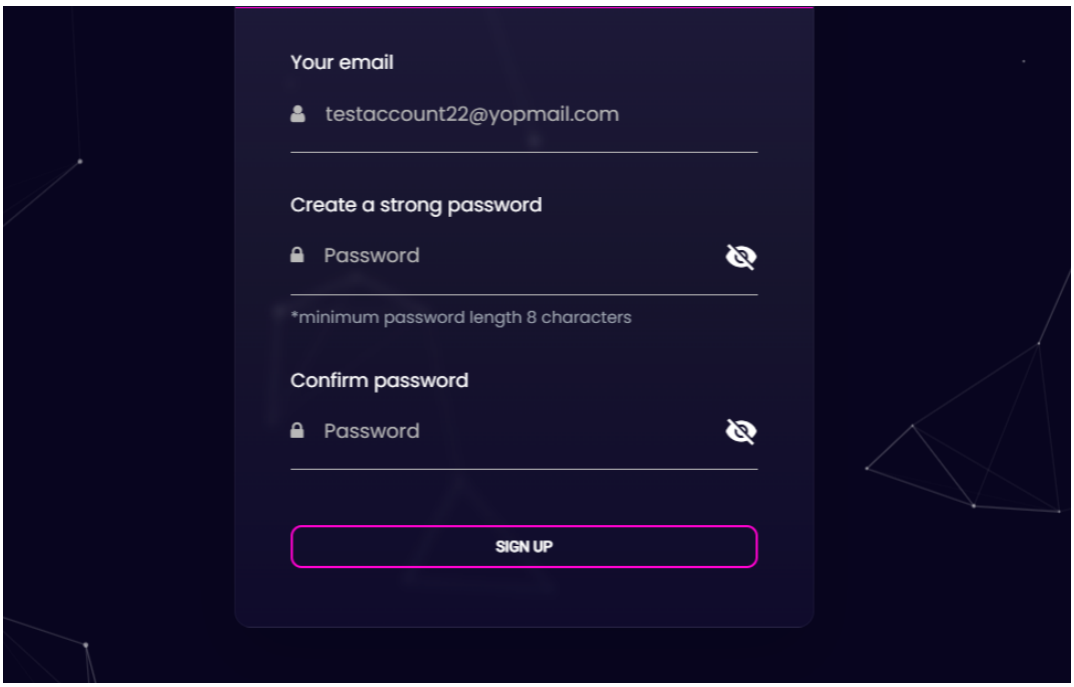

Рисунок В.6 – Початкова реєстрація акаунту

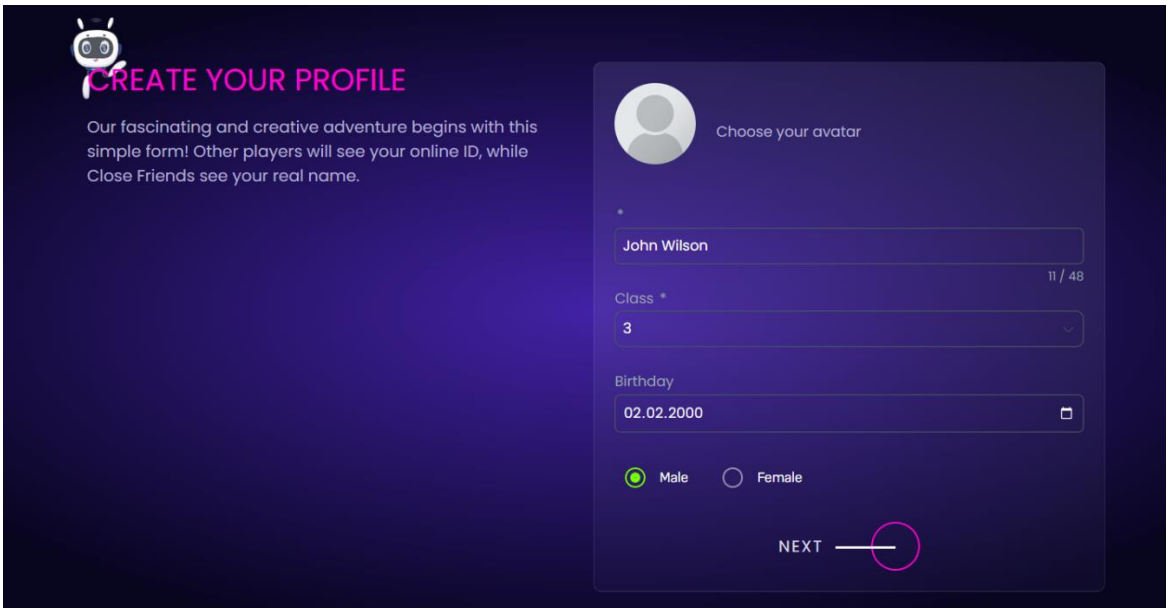

Рисунок В.7 – Введення основної інформації

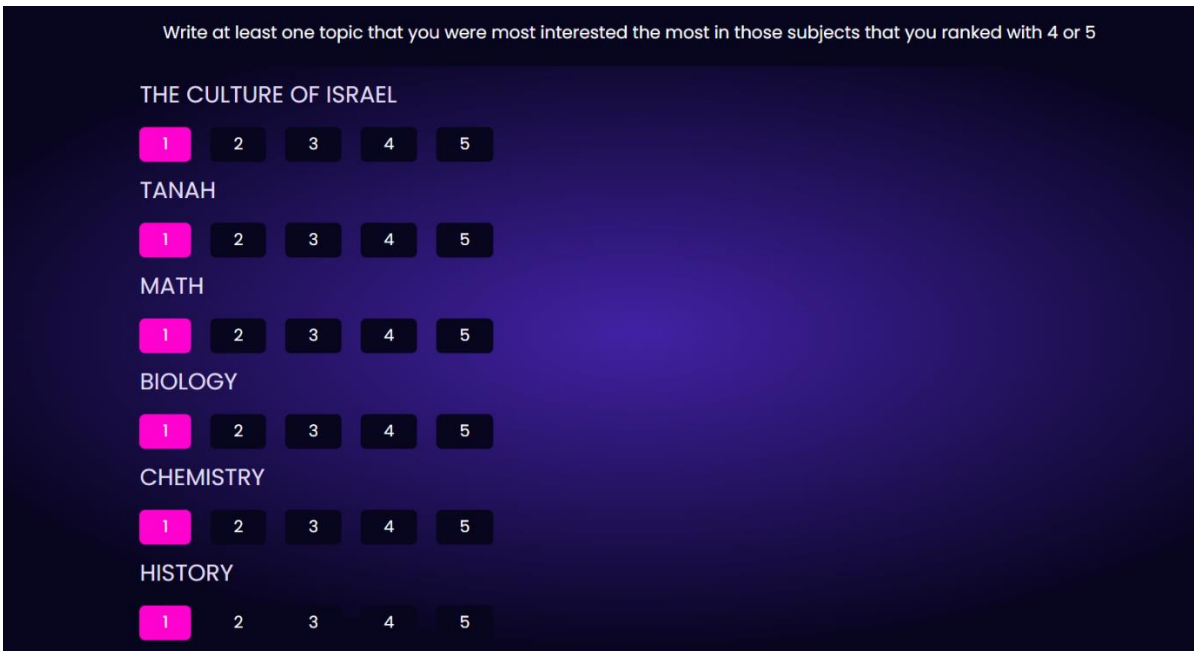

Рисунок В.8 – Проходження анкети. Частина 1

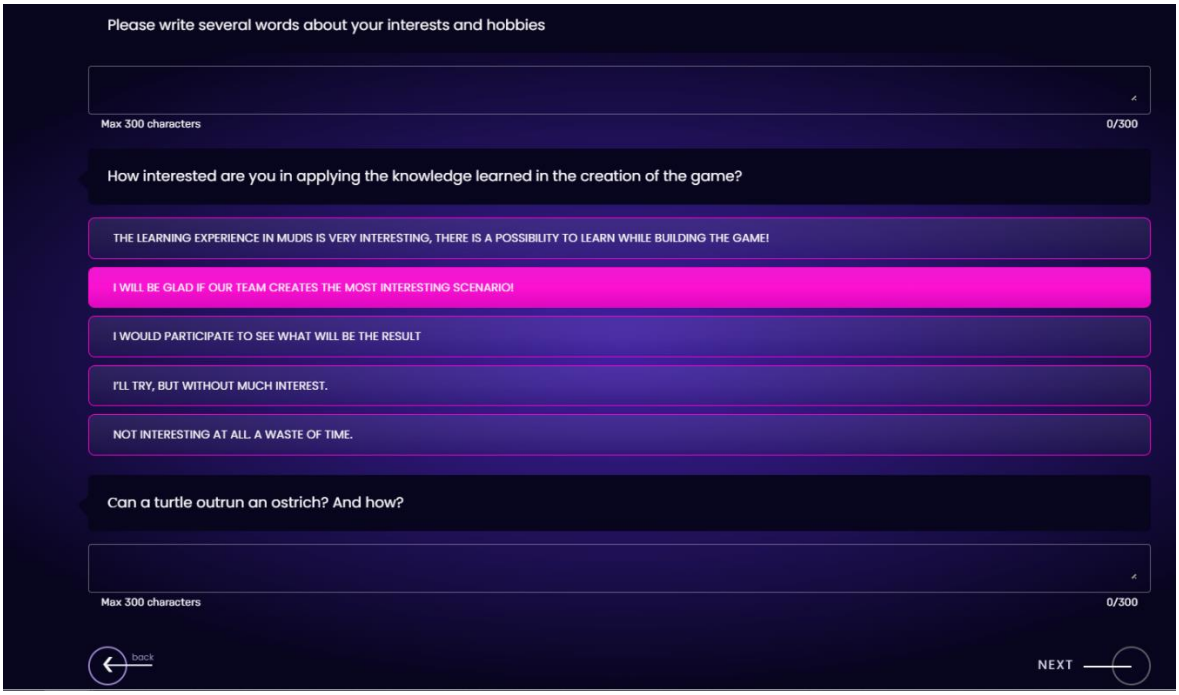

Рисунок В.9 – Проходження анкети. Частина 2

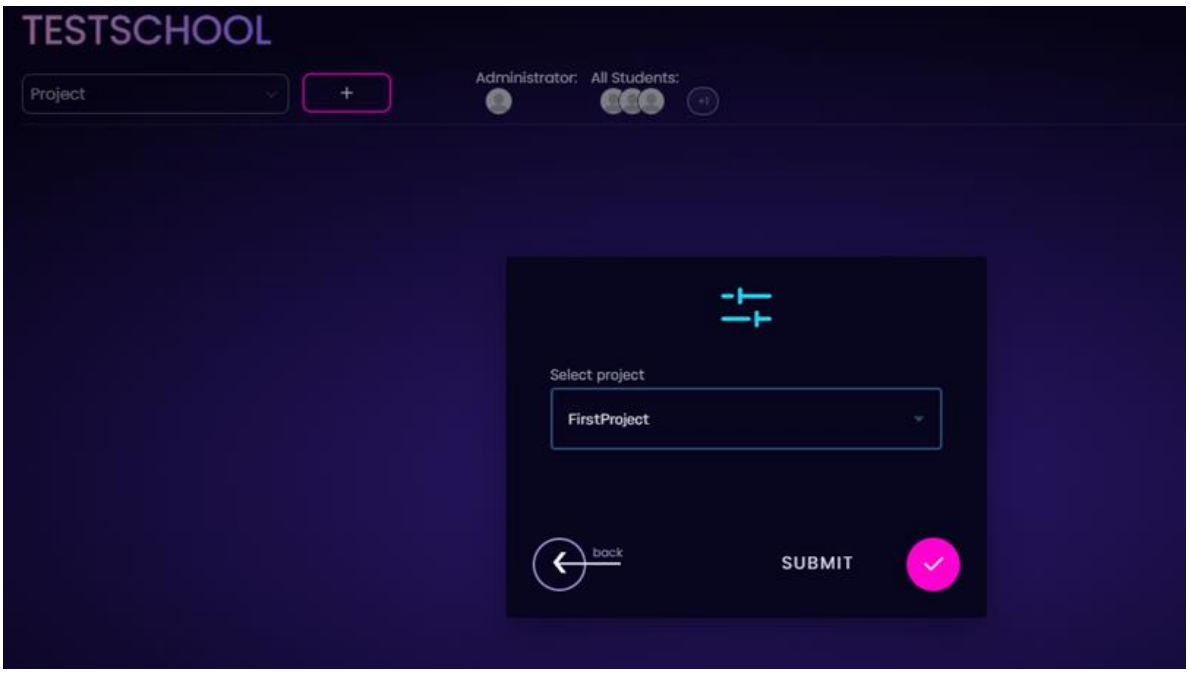

Рисунок В.10 – Генерація команди

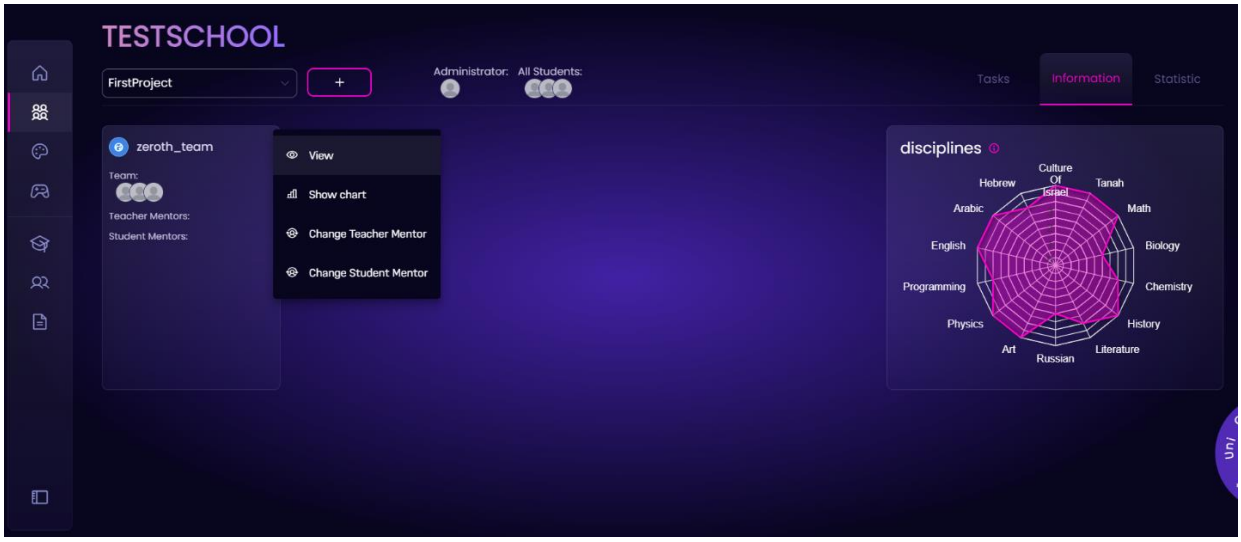

Рисунок В.11 – Сторінка усіх згенерованих команд школи

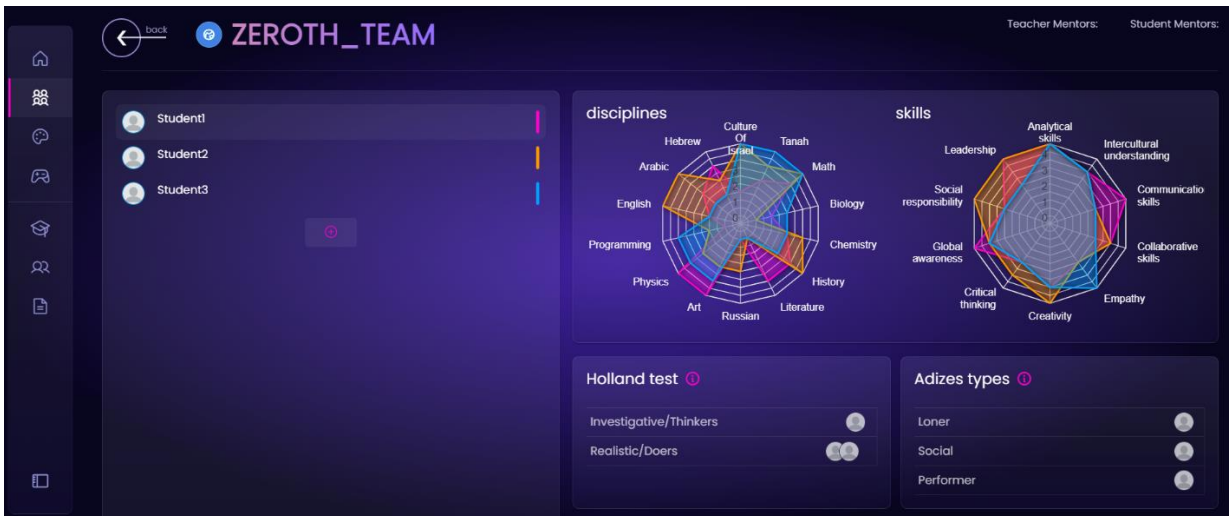

Рисунок В.12 – Сторінка згенерованої команди

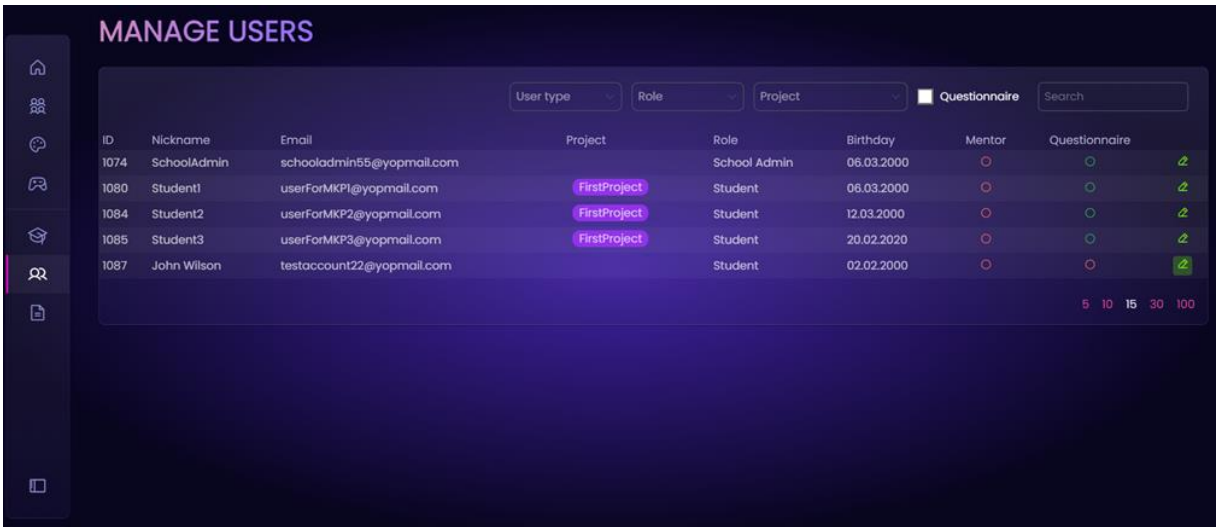

Рисунок В.13 – Список користувачів школи

Додаток Г
(довідниковий).

Науково-виробниче підприємство
“СПІЛЬНА СПРАВА” Товариство з обмеженою відповідальністю
Україна, 21000, м. Вінниця, вул. Магістратська, 36/3, тел. +38(096)-558-24-64
код ЄДРПОУ 31041670, код платників податків 310416702282, свідоцтво № 0183329
IBAN : UA 41 322313 0000026004000003226 в філії АТ «Укрексімбанк» в м. Вінниці

Затверджую

Директор

НВП «СПІЛЬНА СПРАВА», ТОВ

Малачковська Роксолана Ігорівна

«01» грудня 2024р.

АКТ

впровадження результатів магістерської роботи

Іщенка Павла Сергійовича «Розробка клієнт-серверної системи для автоматизації процесу формування команд учнів для навчання онлайн. Частина 1. Серверна частина»

Комісія у складі головного спеціаліста, керівника відділу обробки даних С. Житанського та керівника відділу розробки інформаційних систем, О. Кириленка склали цей акт про те, що у Науково-виробничому підприємстві “Спільна Справа”, ТОВ впроваджуються результати, які отримані магістром Іщенком П. С. при виконанні магістерської кваліфікаційної роботи мають практичну цінність. Інструменти для генерації команд учнів є досить актуальними та необхідними для освітніх систем в умовах сьогодення, при зростанні попиту на технології розвитку сфери онлайн навчання.

Таким чином, актуальність роботи П. С. Іщенка не викликає сумнівів, а низка методик та підходів та результати їх застосування можуть бути застосованими у практичній діяльності.

Науково-виробничому підприємству “Спільна Справа”, ТОВ магістром Іщенком П. С. передано програмне забезпечення, яке дозволяє підвищити ефективність організації онлайн навчання шляхом розробки серверної частини для автоматизації процесу формування команд учнів.

Члени комісії:

Головний спеціаліст. Сергій ЖИТАНСЬКИЙ

Керівник відділу Олександр КИРИЛЕНКО