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 2. Клієнтська частина»

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

(Підпис) Євгеній ЧЕГА
(Ім'я та ПРІЗВИЩЕ)

Керівник: к.т.н., доцент каф. АІІТ
(Підпис) Володимир КОЦЮБИНСЬКИЙ
(Ім'я та ПРІЗВИЩЕ)

«4» 12 2024 р.

Опонент: д.т.н., проф каф. КН
(Підпис) Ярослав ІВАНЧУК
(Ім'я та ПРІЗВИЩЕ)

«4» 12 2024 р.

Допущено до захисту
Зав. кафедри КСУ ВНТУ, 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

З А В Д А Н Н Я
НА МАГІСТРЕСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту ЗАКІТР – 23м Чезі Євгеній Івановичу
(прізвище, ім'я, по батькові)

1. Тема магістерської кваліфікаційної роботи: Розробка клієнт-серверної системи для автоматизації процесу формування команд учнів для навчання онлайн. Частина 2. Клієнтська частина.
   керівник роботи Коцюбинський Володимир Юрійович, доцент кафедри АІІТ
   затверджені наказам ВНТУ від «17» вересня 2024 року №310
2. Термін подання студентом роботи « 1 » грудня 2024 року
3. Вихідні дані до роботи:
   1. Дубовой В. М., Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спеціальності 174 «Автоматизація, комп'ютерно- інтегровані технології та робототехніка» (кафедра комп'ютерних систем управління) [Електронний ресурс] / Вінниця : ВНТУ, 2023 – (PDF, 45 с.);
   2. Олена Амеліна, Олександр Цуркан. Дистанційне та змішане навчання. Досвід, поради, інструменти / Київ «Основа», 2022 – 128с
   3. Saddleback Educational Saddleback Educational Publishing. Science and Technology Words. Saddleback Educational Publishing, Incorporated, 2010. 125 с.
4. Зміст текстової частини: Вступ. Аналіз предметної області та огляд існуючих аналогів та систем. Вибір та огляд технологій для розробки даної системи. Вибір архітектури системи. Розробка структури системи. Розробка та тестування розробленої клієнтської частини системи. Висновки, Список використаних джерел. Додатки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): UML Deployment діаграма. UML діаграма класів. UML діаграма об'єктів. UML діаграма компонентів. UML usecase діаграма. Результати тестування системи.

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

2. 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	17.09.2024	25.09.2024	Виконав
2	Огляд існуючих систем та рішень	25.09.2024	29.09.2024	Виконав
3	Постановка задачі та визначення основних функцій системи	30.09.2024	18.10.2024	Виконав
4	Проектування архітектури серверної частини	19.10.2024	24.10.2024	Виконав
5	Аналіз та вибір технологій розробки	25.10.2024	20.10.2024	Виконав
6	Розробка системи	21.10.2024	26.10.2024	Виконав
7	Тестування розробленої системи	26.10.204	28.10.2024	Виконав
8	Оформлення пояснювальної записки,	29.10.2024	08.11.2024	Виконав
9	Захист МКР	09.12.2024	12.12.2024	

Студент _____ Євгеній ЧЕГА
(підпис) (ім'я і ПРІЗВИЩЕ)

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

АНОТАЦІЯ

УДК 004.75:371.3

Чега Є. І. Розробка клієнт-серверної системи для автоматизації процесу формування команд учнів для навчання онлайн. Частина 2. Клієнтська частина. Магістерська кваліфікаційна робота зі спеціальності 171 - Автоматизація та комп'ютерно-інтегровані технології та роботехніка – Інформаційні системи та інтернет речей. Вінниця: ВНТУ, 2024. 97 с.

На укр. мові. Бібліогр.: 34 назв; рис.: 31; дод.: 3

Метою роботи є підвищення ефективності онлайн-навчання через автоматизацію процесу формування команд учнів за допомогою клієнт-серверної системи, зокрема вдосконалення клієнтської частини.

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, клієнська частина.

ABSTRACT

UDC 004.75:371.3

Development of a client-server system for automating the process of forming teams of students for online learning. Part 2. Client part. Master's qualification work in speciality 171 - Automation and computer-integrated technologies and robotics - Information systems and the Internet of things. Vinnytsia: VNTU, 2024. 70 c.

In Ukrainian. Bibliography: 34 titles; Figures: 31; Supplement: 3.

The aim of the work is to improve the effectiveness of online learning by automating the process of forming teams of students using a client-server system, in particular, improving the client side.

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 teams of students for online learning, the basic principles, elements of the automated system are considered and analysed. 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 to be performed by the system were set, the architecture was modelled, and the technologies required for development were selected. The software was developed and tested.

Keywords: automation, team, online learning, system, education, client side.

ЗМІСТ

ВСТУП

Актуальність. У сучасному світі навчання, поєднання інноваційних технологій та педагогічних підходів є ключовим фактором для підвищення ефективності освітнього процесу. Одним з важливих аспектів автоматизації навчання в онлайн-середовищі є процес формування команд учнів. Це дозволяє підвищити рівень взаємодії, співпраці та залученості учнів у навчальний процес. Особливо актуальним є розробка клієнт-серверної системи, яка автоматизує процес формування таких команд.

Технології дозволяють створювати гнучкі рішення, що враховують особливості кожного учня та сприяють побудові ефективної комунікації між учасниками навчального процесу. Клієнтська частина такої системи відіграє важливу роль, забезпечуючи користувачам (учням і вчителям) зручний та ефективний інтерфейс для участі в процесі формування команд та спільного навчання.

Метою даного дослідження є підвищення ефективності онлайн-навчання через автоматизацію процесу формування команд учнів за допомогою клієнт-серверної системи, зокрема вдосконалення клієнтської частини.

Для досягнення мети були визначені такі завдання:

- здійснити аналіз потреб користувачів (учнів та вчителів) щодо функціональності та графічного дизайну системи;
- здійснити аналіз можливих технічних складових графічного дизайну та можливих засобів його реалізацій;
- розробити систему, яка матиме зручний інтерфейс та широку функціональність для користувачів;
- забезпечити коректну взаємодію клієнтської та серверної частини.

Методи дослідження.

У процесі дослідження були застосовані такі методи:

- літературний аналіз (аналіз наукових публікацій і статей з тематики клієнт-серверних систем та автоматизації навчальних процесів);

- аналіз існуючих рішень (порівняння вже розроблених систем, що автоматизують процеси в освіті, з акцентом на їх клієнтську частину);
- прототипування (розробка прототипу клієнтської частини для проведення тестування на вибраній групі користувачів).

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

Предметом дослідження є автоматизація процесу формування команд за рахунок ефектиного використання UI/UX.

Практична цінність полягає у створенні алгоретмічного програмного забезпечення, яке дозволить автоматизувати процес формування команд учнів для спільного навчання в онлайн-середовищі, забезпечуючи зручність користування.

Інноваційність розробки полягає у впровадженні інтуїтивного інтерфейсу та інтеграції сучасних технологій та підвищеню ефективності формування команд для побудови ефективного навчального середовища.

Практична цінність полягає в автоматизації процесу формування команд, що дозволяє ефективно використовувати ресурси учасників, точно відстежувати їхні навички та прогрес, а також забезпечувати зручний моніторинг і контроль за виконанням завдань, що сприяє підвищенню ефективності та організації навчального процесу.

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

Публікації: Чега Євгеній Іванович «Аавтоматизація процесу формування команд учнів для підвищення ефектиності навчання онлайн», «Науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2024)», 2024 [Електронний ресурс] – Режим доступу: http://surl.li/gxihfs

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ СИСТЕМ ІЗ АВТОМАТИЗАЦІЇ ПРОЦЕСУ ФОРМУВАННЯ КОМАНД УЧНІВ ДЛЯ НАВЧАННЯ ОНЛАЙН

1.1 Огляд процесу онлайн-навчання та можливостей його покращення

Онлайн-навчання стало невід'ємною складовою сучасної освіти[4], додаючи нові можливості й задачі які не обіхдно виріщувати. Із розвитком цифрових технологій та широким доступом до Інтернету, навчальні формати набули гнучкості, що дозволяє учням здобувати знання в зручний для них час і просторі. Онлайн-освіта пропонує широкий спектр ресурсів для самостійного навчання, інструменти для інтерактивної взаємодії з викладачами та однокласниками, а також доступ до сучасних мультимедійних матеріалів[5].

Проте, незважаючи на численні переваги, онлайн-навчання стикається з низкою викликів, які можуть впливати на ефективність навчального процесу. Серед основних проблем — підтримка мотивації учнів, організація ефективної групової роботи та забезпечення належного рівня комунікації між учасниками. В умовах віддаленого навчання учні можуть відчувати ізоляцію, що може призвести до зниження їхнього інтересу до навчання. Ефективна організація командної роботи є ключовим аспектом для розвитку соціальних навичок та досягнення спільних цілей.

Для подолання вище згаданих проблем варто розгяладати автоматизації процесу формування команд для онлайн-навчання. Використання сучасних технологій[6], таких як штучний інтелект і аналіз великих даних, відкриває нові можливості для обробки інформації про учнів. Завдяки цим технологіям, системи можуть враховувати не лише академічні результати, а й індивідуальні стилі навчання, інтереси та соціально-емоційні характеристики, що суттєво підвищує ефективність формування команд. Автоматизація процесу формування команд дозволяє створювати оптимальні групи учнів, які здатні ефективно співпрацювати, використовуючи свої сильні сторони. Це не лише полегшує навчальний процес, але й забезпечує динамічну та інклюзивну атмосферу, де кожен учасник може внести

свій вклад. Наприклад, учні, які мають різні стилі навчання, можуть об'єднувати свої зусилля для досягнення спільних цілей, що позитивно впливає на їхню мотивацію та залученість.

Автоматизація формування команд дозволяє забезпечити інтеграцію процесу оцінювання ефективності роботи групх[7]. Інформаційні системи здатні проводити регулярні опитування учасників команд щодо рівня їхнього задоволення співпрацею, виявляти слабкі місця в комунікації чи розподілі завдань і надавати викладачам рекомендації для коригування процесу. Такий підхід сприяє динамічному покращенню якості навчання та ефективності командної роботи.

Завдяки всеохоплючим можливостям штучного інтелекту, система здатна в режимі реального часу адаптувати критерії формування команд залежно від успішності учнів або змін у навчальній програмі. Наприклад, якщо певна команда демонструє стабільно високі результати, система може запропонувати розділити її учасників і залучити до інших груп як менторів. Це сприяє поширенню знань та навичок між різними командами, створюючи більш рівномірний розподіл ресурсів і досвіду.

Варто додати, що автоматизовані системи можуть підтримувати багатомовність та адаптацію до різних культурних контекстів, що особливо актуально для міжнародних програм онлайн-навчання. Вони можуть враховувати різні часові пояси, мовні переваги та культурні особливості, формуючи команди так, щоб учасники могли ефективно взаємодіяти. Ще одним напрямом розвитку є інтеграція систем із платформами віртуальної та доповненої реальності (VR/AR). Наприклад, формування команд для роботи в віртуальних лабораторіях чи симуляційних середовищах дозволяє студентам отримувати практичний досвід навіть у віддаленому форматі навчання. Це відкриває нові можливості для організації практичних занять, які раніше були доступні лише в офлайн-режимі.

Автоматизація формування команд може бути корисною і для побудови довгострокових взаємозв’язків між учнями. Наприклад, після завершення курсу система може запропонувати зберегти контакти учасників успішних команд або навіть сформувати мережі випускників для подальшої співпраці. Це підсилює не

лише навчальні, але й соціальні аспекти онлайн-освіти, роблячи її більш цінною для учнів і викладачів.

1.2 Аналіз предметної області

Інтеграція автоматизованих систем у процес онлайн-навчання[33] дозволяє створити умови для гнучкого поєднання різних навичок і стилів навчання, що позитивно впливає на якість групової роботи, результативність проєктів та мотивацію учасників. Завдяки цим технологіям учні отримують можливість працювати в командах, що оптимально налаштовані для досягнення спільних цілей, що суттєво підвищує ефективність навчального процесу та сприяє розвитку комунікативних і соціальних навичок. Таким чином, автоматизація формування команд у процесі онлайн-навчання не лише спрощує організаційні моменти, але й створює нові горизонти для розвитку освітніх практик, що важливо для забезпечення якісної освіти в умовах сучасного світу.

Автоматизація формування команд у процесі онлайн-навчання відкриває нові можливості для створення ефективних освітніх середовищ[8] та підносить навчальну сферу на новий рівень. Використання сучасних технологій, таких як штучний інтелект та аналіз великих даних, дозволяє подібним системам швидко обробляти інформацію про учнів і формувати команди, що максимально відповідають конкретним завданням. Інтерграція таких систем дозволяє створити умови для гнучкого поєднання різних навичок і стилів навчання, що позитивно впливає на якість групової роботи, результативність проєктів та мотивацію учасників.

Одним із основних аспектів автоматизації є можливість створення команд на основі глибокого аналізу даних про кожного учасника. Системи враховують індивідуальні переваги у способах навчання, рівень володіння темами та особистісні риси, що можуть впливати на взаємодію в команді. Такий підхід дозволяє збалансувати знання, навички та стилі комунікації в команді, що сприяє продуктивній співпраці. Ці технології також скорочують час на організацію

групової роботи, надаючи можливість учням швидше приступати до завдань та ефективніше розподіляти обов'язки, зосереджуючись на своїх сильних сторонах. Автоматизовані системи не лише забезпечують організацію, але й підвищують залученість учнів[9]. Створюючи групи, які базуються на індивідуальних інтересах та здібностях, такі системи сприяють обміну знаннями між учнями. Учасники груп не лише навчаються від викладача, а й активно взаємодіють між собою, що сприяє креативності та розвитку критичного мислення. У свою чергу, це підсилює принципи колаборативного навчання, де учні активно беруть участь у спільному процесі вирішення проблем, отримуючи практичні навички співпраці.

Важливим елементом є також можливість безперервного відслідковування прогресу команд завдяки зворотному зв'язку в режимі реального часу. Системи можуть автоматично аналізувати ефективність співпраці та рекомендувати корективи, що робить навчальний процес більш адаптивним та гнучким. Такий підхід дозволяє викладачам зосередитися на стратегічному управлінні та індивідуальній підтримці учнів, створюючи сприятливі умови для навчання.

Центральним компонентом автоматизованого формування команд є система управління навчанням (Learning Management System, LMS)[10]. Вона збирає й аналізує великий обсяг даних про учнів, включаючи їхні академічні результати, стилі навчання, інтереси та рівень залученості в процес. Завдяки цьому LMS здатна створювати індивідуальні профілі учнів, враховуючи їхні сильні й слабкі сторони. На рис. 1.1 зображено структурну схему системи управління навчанням:

Рисунок 1.1 – структурна схема управління навчанням

LMS також відстежує взаємодію між учнями у попередніх групах, оцінюючи їхню активність, ролі та рівень співпраці. Це дозволяє системі формувати збалансовані команди, у яких кожен учасник може ефективно використовувати свої навички й взаємодіяти з іншими. Алгоритми машинного навчання обробляють ці дані, щоб створювати команди, що максимально відповідають завданням і забезпечують ефективну групову роботу. Після створення команд, LMS може пропонувати завдання та ресурси, адаптовані до специфічних потреб групи. Це може включати в себе матеріали, які сприяють розвитку тих навичок, що вже були виявлені під час аналізу даних. Наприклад, якщо команда потребує посилення аналітичних навичок, система може запропонувати відповідні навчальні модулі або активності.

Загальні освітні концепції, такі як персоналізоване навчання, колаборативне навчання та конструктивізм, гармонійно інтегруються в автоматизовані системи. Персоналізація дозволяє кожному учню реалізувати свій потенціал у командній роботі, тоді як колаборативне навчання підкреслює важливість взаємодії між учнями для досягнення спільних цілей. Конструктивістський підхід до навчання через діяльність реалізується через практичні завдання, які допомагають учням краще засвоювати знання шляхом активного застосування в реальних ситуаціях.

Таке поєднання технологій і педагогічних підходів[34] відкриває нові горизонти для освітнього процесу, роблячи його більш інтерактивним і ефективним. Важливо, що автоматизовані системи не лише забезпечують формування команд, але й активно підтримують процес навчання на всіх етапах. Додатково, LMS може автоматично надавати зворотний зв'язок учням, оцінюючи їхню участь і результати в командній роботі. Це забезпечує можливість для саморефлексії, дозволяючи учням усвідомлювати свої сильні та слабкі сторони, а також те, як їхня участь вплинула на загальний результат команди. Таким чином, учні отримують не лише академічні знання, а й розвивають навички самооцінки та критичного мислення

Таким чином, автоматизація формування команд не лише оптимізує організаційні аспекти онлайн-освіти, але й створює середовище для розвитку

соціальних і професійних навичок учнів. Це значно підвищує ефективність навчального процесу та забезпечує досягнення учнями вищих результатів через інтеграцію сучасних технологій та освітніх концепцій.

1.3 Аналіз об’єкта автоматизації

Об'єктом автоматизації є процес формування команд учнів для групової роботи в онлайн-навчанні. Цей процес охоплює декілька важливих етапів, які забезпечують ефективність комунікації та співпраці серед учнів.

Першим етапом є збір даних[11] про учнів, що включає не лише оцінку їхніх академічних результатів, але й дослідження особистісних характеристик, таких як соціальні навички, мотивація, стиль навчання і інтереси. Наприклад, система може враховувати, чи віддає перевагу учень самостійній роботі або ж йому комфортніше в команді. Ця інформація може бути отримана через опитування, анкетування або аналіз історії навчальної активності на платформі.

Другим етапом є аналіз зібраних даних. Тут важливу роль відіграють алгоритми машинного навчання[12], які здатні виявляти закономірності та зв'язки між різними параметрами. Наприклад, система може визначити, які комбінації учнів у попередніх проектах призводили до успішних результатів, а які – до проблем. Це дозволяє врахувати не лише академічні здібності, але й соціальну динаміку, що формується в групі.

На третьому етапі відбувається формування команд. Автоматизовані системи використовують алгоритми для створення збалансованих груп, у яких кожен учасник має можливість внести свій внесок, виходячи з своїх сильних сторін. Наприклад, у команду можуть бути включені учні з сильними аналітичними навичками, креативним мисленням і відмінними комунікаційними здібностями. Це забезпечує різноманітність у підходах до вирішення завдань, що є критично важливим для досягнення спільних цілей.

Крім того, автоматизація формування команд[13] знижує навантаження на вчителя. Оскільки система самостійно аналізує дані і формує групи, вчитель може зосередитися на інших аспектах навчання, таких як розробка навчальних матеріалів, надання зворотного зв'язку та індивідуальна робота з учнями. Це дозволяє створити більш ефективне навчальне середовище, де вчитель стає фасилітатором, а не просто джерелом інформації.

Також важливо, що автоматизовані системи забезпечують постійний моніторинг процесу групової роботи. Вони можуть відстежувати активність учнів у команді, їхній внесок у проект та взаємодію між учасниками. Це дозволяє не лише оцінювати результати роботи, але й своєчасно виявляти проблеми, які можуть виникнути в процесі співпраці. Наприклад, якщо один з учасників неактивний, система може надати вчителю інформацію про це, що дасть змогу вжити заходів для покращення динаміки групи.

Кінцевою метою автоматизації формування команд є створення оптимальних умов для співпраці учнів, що сприяє їхньому особистісному і професійному розвитку. Вона не лише полегшує організаційні моменти, але й дозволяє учням працювати в різноманітних командах, розвивати навички спілкування, критичного мислення та колективного вирішення проблем. Таким чином, автоматизація формування команд стає важливим елементом сучасної освіти, який допомагає підготувати учнів до викликів у їхньому майбутньому.

1.4 Сучасні методи автоматизації генерації команд для навчання

Сучасні методи автоматизації формування команд учнів активно використовують алгоритми штучного інтелекту[14], які дозволяють з високою точністю враховувати різноманітні параметри для створення збалансованих та ефективних груп. Ці алгоритми функціонують на основі аналізу великих обсягів даних, що надаються навчальними платформами, і здатні виявляти закономірності

в поведінці учнів, їхніх навчальних результатах, стилях навчання та особистісних уподобаннях.

Перш за все алгоритми можуть використовувати методи класифікації та кластеризації для групування учнів за певними критеріями. Наприклад, класифікація може допомогти визначити, чи підходить учень для роботи в команді на основі його успіхів у певних предметах або особистісних характеристик, таких як комунікабельність чи здатність до співпраці. Кластеризація, в свою чергу, дозволяє об'єднувати учнів із подібними навчальними стилями або інтересами, що забезпечує створення більш однорідних груп.

Сучасні системи застосовують аналітику поведінки учнів, яка є ключовим елементом автоматизації формування команд. Платформи можуть стежити за активністю учнів у реальному часі, фіксуючи їхню участь у дискусіях, виконанні завдань та взаємодії з іншими учасниками. Наприклад, якщо система помічає, що учень виявляє ініціативу в обговореннях або активно підтримує інших, вона може включити його в команду, де його лідерські якості та готовність до співпраці принесуть найбільшу користь. Це дозволяє формувати динамічні команди, здатні адаптуватися до змін у навчальному процесі, а також реагувати на нові завдання та виклики. Важливою складовою автоматизації є персоналізація навчального досвіду.

Системи можуть враховувати особисті вподобання учнів щодо типів завдань і методів навчання. Наприклад, учень, який віддає перевагу проектній роботі, може бути об'єднаний із тими, хто має схожі інтереси, що дозволяє створювати команди, в яких учасники працюють у найбільш комфортному для себе середовищі. Це підвищує мотивацію та зацікавленість учнів, оскільки вони можуть працювати над завданнями, які їх дійсно цікавлять. Автоматизовані системи формування команд можуть використовувати прогнозну аналітику[15], яка дозволяє оцінювати ймовірність успіху різних комбінацій учнів. Завдяки аналізу попередніх даних про результати групової роботи, система може визначити, які склади команд найчастіше призводять до успіху в конкретних проектах. Це забезпечує додатковий рівень обґрунтованості у формуванні команд, зменшуючи ризик невдач.

Варто занзачити, що зворотний зв'язок є важливою частиною процесу автоматизованого формування команд. Системи, що інтегрують функцію моніторингу прогресу команд, можуть автоматично відстежувати виконання завдань, надаючи як учням, так і вчителям миттєвий доступ до результатів роботи. Це дозволяє оперативно реагувати на будь-які труднощі чи невдачі, що можуть виникнути в процесі групової діяльності. В результаті, зворотний зв'язок стає не лише інструментом для оцінки, але й важливим елементом для розвитку та вдосконалення навчального процесу.

Завдяки автоматизації учні можуть бачити конкретні аспекти своєї роботи, які потребують покращення, а також розуміти, які підходи працюють найкраще для досягнення цілей. Такий зворотний зв'язок стимулює їх до більш активної участі, самокритики та рефлексії, що підвищує рівень їхньої мотивації до навчання. Водночас вчителі отримують можливість швидко виявляти проблеми у командній роботі або незадовільні результати, що дозволяє оперативно вжити коригувальних заходів, наприклад, у вигляді додаткових консультацій або зміни підходів до організації навчання.

Окрім цього, автоматизовані системи формування команд створюють умови для розвитку комунікаційних та соціальних навичок учнів, що є важливими складовими їхнього особистісного та професійного розвитку. Коли учні взаємодіють у різних командах, вони не лише набувають практичних знань від викладачів, але й обмінюються ідеями, досвідом та навичками між собою. Така співпраця дозволяє розвивати критичне мислення, креативність і вміння працювати в колективі, що є важливими навичками для майбутньої кар'єри.

Створюючи таке середовище, де учні активно залучені до процесу навчання та розвитку, автоматизація сприяє формуванню активного навчального середовища, яке підвищує ефективність як самого процесу навчання, так і взаємодії між учасниками. Зворотний зв'язок, наданий автоматизованою системою, забезпечує безперервне вдосконалення не тільки знань учнів, але й їхніх умінь працювати разом у команді, що має величезне значення в умовах сучасного освітнього процесу.

Підсумовуючи можна занзанчити, що сучасні методи автоматизації формування команд сприяють створенню ефективних освітніх середовищ, де учні можуть досягати кращих результатів у навчанні. Вони забезпечують можливість реалізувати потенціал кожного учня в командній роботі, що є важливим аспектом для їхнього особистісного та професійного розвитку.

1.5 Порівняльна характеристика існуючих систем для навчання онлайн

Автоматизація формування команд у сучасних освітніх системах є важливим інструментом для ефективної організації навчального процесу та покращення групової динаміки. У рамках навчальних платформ, таких як Moodle, Google Classroom, Blackboard та Canvas, інтегровані функції для розподілу учнів у групи, що дозволяє полегшити роботу викладачам та підвищити залученість студентів до спільної роботи. Проте, попри наявність цих можливостей, викладачі часто стикаються з обмеженнями гнучкості налаштувань або браком глибокого аналізу ефективності команд.

Сучасні системи автоматизації формування команд дозволяють викладачам швидко та зручно організовувати команди на основі різних критеріїв. Однак, існують значні відмінності у функціоналі, аналітичних можливостях та адаптивності між різними платформами. Деякі з них надають базові можливості випадкового або рівномірного розподілу учасників, в той час як інші дозволяють більш глибоку персоналізацію, використовуючи такі параметри, як академічні показники, інтереси чи соціальні взаємодії. Проте, у багатьох випадках такі системи не забезпечують гнучкості, необхідної для складніших підходів до формування команд, особливо якщо мова йде про аналіз ефективності та адаптацію до конкретних навчальних середовищ.

1.5.1 Moodle

Moodle[16] є одною із з найпоширеніших систем управління навчанням (Learning Management System, LMS), яка використовується у навчальних ціллях, зокрема як дистанційного, так і очного навчання. Основною перевагою цієї платформи є її гнучкість, можливість інтеграції плагінів та відкритий код, що дозволяє адаптувати систему під специфічні потреби закладу або курсу.

Moodle надає широкий спектр інструментів для викладачів та учнів, серед яких модулі для створення та управління навчальними курсами, інтерактивні завдання, оцінювання та форуми для обговорень. Користувачі можуть створювати курси з різними типами контенту, такими як текстові матеріали, відео, презентації, а також інтерактивні елементи, що залучають студентів до активного навчання. Крім того, Moodle дозволяє налаштовувати гнучкі правила оцінювання, включаючи автоматизоване тестування, яке дає можливість оцінювати знання учнів безпосередньо на платформі.

Платформа підтримує багатомовність, що дає змогу використовувати Moodle для навчання в міжнародних або багатонаціональних середовищах, де учасники можуть працювати на своїй рідній мові. Moodle також має вбудовану систему сповіщень і повідомлень, яка дозволяє викладачам і студентам оперативно обмінюватися інформацією. Система дозволяє реалізувати різноманітні форми взаємодії, такі як форуми, чати, завдання та опитування, що сприяє розвитку співпраці і комунікації між учасниками навчального процесу.

Однією з важливих функцій Moodle є її можливість інтеграції з іншими навчальними ресурсами, що дозволяє створювати єдину платформу для всіх навчальних матеріалів та інструментів. Платформа підтримує інтеграцію з різними зовнішніми системами, такими як платіжні системи, відеоконференцсистеми, системи для ведення оцінок та інші освітні технології, що розширює її можливості та робить навчальний процес більш зручним і ефективним. Наприклад, інтеграція з відеоконференцсистемами дає можливість проводити онлайн-лекції, семінари та групові обговорення без необхідності в сторонніх програмах.

Moodle також має великий вибір готових плагінів, що дозволяють розширювати її функціональність, включаючи інтеграцію з різними типами мультимедійних ресурсів, проведення онлайн-опитувань та тестів, а також підтримку різних мов, що робить її доступною для користувачів з усього світу. Це забезпечує універсальність та зручність використання Moodle у різноманітних навчальних контекстах, від шкіл до університетів, і є однією з основних причин її популярності. Важливою перевагою є також те, що Moodle — це відкритий програмний продукт, що дозволяє навчальним закладам налаштовувати платформу під свої індивідуальні вимоги без необхідності закупівлі дорогих ліцензій.

Окрім цього, Moodle підтримує різноманітні форми оцінювання, зокрема автоматизовані тести, завдання з перевіркою результатів у реальному часі та зворотний зв’язок для студентів, що дозволяє викладачам ефективно відстежувати прогрес кожного учня та вчасно коригувати навчальний процес. Платформа підтримує також можливість створення звітів про успішність студентів, що дозволяє здійснювати моніторинг та аналіз навчальних досягнень учнів на різних етапах їхнього навчання.

Таким чином, Moodle є потужним інструментом для управління навчанням, який допомагає створювати ефективне, інтерактивне та доступне навчальне середовище. Завдяки своїй гнучкості, відкритому коду та широкому набору інструментів, Moodle залишається одним із найбільш популярних виборів для онлайн-освіти у всьому світі.

1.5.2 Google Classroom

Google Classroom[17] — це хмарна платформа для організації навчального процесу, яка є частиною екосистеми Google. Завдяки інтеграції з іншими сервісами Google, такими як Google Docs, Sheets, Drive, вона забезпечує зручний інструментарій для організації як індивідуальних, так і групових завдань.

Платформа стала популярною завдяки своїй простоті у використанні, доступності та широкій інтеграції з іншими продуктами Google.

Google Classroom дозволяє викладачам створювати групові завдання та автоматично розподіляти студентів по командах. Однак можливості для налаштування цього процесу досить обмежені:

1. Основний метод розподілу студентів у Google Classroom — це випадковий розподіл, де студенти рівномірно розподіляються по групах. Викладач може вказати кількість команд або кількість студентів у кожній групі, а система автоматично організує учасників. Цей процес підходить для простих групових завдань, де немає потреби враховувати індивідуальні особливості студентів.
2. Google Classroom не надає можливості глибокого налаштування критеріїв для розподілу учасників. Наприклад, система не враховує академічні показники студентів, їхні інтереси, або інші персоналізовані фактори під час створення команд.
3. Викладач може самостійно створювати групи і вручну призначати студентів до певних команд. Це дозволяє врахувати певні критерії, проте цей процес є більш трудомістким і може зайняти більше часу, особливо в умовах великих груп.

На додачу до вище згадних особливостей розподілу варто уточнити певні обмеження, які можуть бути викликом для організації складніших командних завдань:

1. Відсутність глибокого аналізу роботи груп, тобто Google Classroom не надає інструментів для детального аналізу ефективності роботи команд. Наприклад, викладач не може відстежувати рівень активності кожного учасника, їхній вклад у спільну роботу чи інші аспекти взаємодії. Це обмежує можливості для контролю якості командної роботи.
2. Обмежені можливості автоматизації при якій, розподіл студентів у групи не враховує такі важливі фактори, як академічні результати або взаємодію між студентами. У порівнянні з іншими платформами, Google Classroom не

пропонує додаткових алгоритмів або плагінів для більш гнучкого розподілу команд.

3. Брак інструментів для відстеження внеску окремих учасників, що не дозволяє викладачам детально аналізувати індивідуальний вклад кожного учасника в групове завдання. Це ускладнює процес оцінювання, особливо коли потрібно відзначити роботу окремих студентів у команді.

На противагу недолікам слід занзачати переваги Google Classroom, і його інтеграцію з іншими продуктами Google, що значно полегшує роботу над груповими завданнями:

1. Google Docs та Sheets, де студенти можуть одночасно працювати над документами, таблицями або презентаціями в режимі реального часу. Кожен учасник команди може редагувати або коментувати спільний документ, що сприяє ефективній співпраці.
2. Google Drive, який розміщує в собі всі необіхдні матеріали, над якими працює група, автоматично зберігаються у Google Drive. Це спрощує доступ до файлів та організацію роботи в команді, адже викладач і студенти мають централізоване сховище для своїх документів.
3. Google Meet, який дозволяє студентам легко організовувати відеоконференції для обговорення проєктів. Викладачі можуть також організовувати відеозустрічі з командами для моніторингу прогресу.

Google Classroom відмінно підходить для простих групових завдань, коли немає потреби у складних налаштуваннях або глибокому аналізі командної роботи. Викладачі можуть швидко створювати групові проєкти, організовувати співпрацю через Google Docs та інші інструменти, а студенти — мати легкий доступ до всіх необхідних ресурсів у рамках одного інтегрованого середовища.

Тому Google Classroom є зручним інструментом для автоматизації простих групових завдань завдяки інтеграції з іншими сервісами Google, такими як Docs, Sheets та Drive. Однак, його можливості для гнучкого налаштування командного розподілу та глибокого аналізу командної роботи є обмеженими.

1.5.3 Blackboard

Blackboard[18] — це одна з найпотужніших і найбільш функціонально насичених систем управління навчанням (Learning Management System, LMS), яка широко використовується в освітніх установах по всьому світу. Вона надає викладачам та адміністраторам значний набір інструментів для організації навчальних процесів, включно з можливостями для створення та управління командами студентів для групових завдань. Blackboard пропонує гнучкі варіанти розподілу учасників у команди на основі різних критеріїв і забезпечує інструменти для аналізу ефективності їхньої роботи.

Формування команд у Blackboard може відбуватися настпними способами:

1. Випадковий розподіл студентів дозволяє викладачам автоматично розподіляти студентів по командах на випадковій основі. Це зручний і швидкий спосіб організувати роботу команд, коли немає необхідності враховувати будь-які специфічні критерії, такі як рівень знань або інтереси студентів. Випадковий розподіл часто використовується для швидкого налаштування групових завдань, особливо в умовах великих класів.
2. Розподіл за академічними критеріями пропонує можливість формувати команди на основі певних критеріїв, таких як академічні показники студентів. Викладач може налаштувати систему таким чином, щоб у кожній команді був рівномірно розподілений рівень знань або успішності. Це дозволяє створювати збалансовані команди, де учасники з різним рівнем підготовки можуть працювати разом для досягнення спільних цілей.
3. Соціальні та поведінкові критерії дозволяють проводити формування команд на основі соціальної взаємодії або попереднього досвіду роботи студентів у групах. Це може бути корисно для викладачів, які прагнуть врахувати динаміку комунікації між студентами або їхні навички роботи в команді.
4. Ручне налаштування команд дозволяє вручну формувати команди, що дозволяє їм призначати студентів до конкретних груп за власним бажанням або на основі специфічних критеріїв, які не можна автоматизувати. Це

корисно у випадках, коли важливо врахувати індивідуальні особливості студентів або їхні попередні досягнення.

Однією з важливих переваг Blackboard є наявність вбудованих аналітичних інструментів, які дозволяють викладачам детально аналізувати прогрес команд та ефективність їхньої роботи:

1. Моніторинг активності системи Blackboard дозволяє викладачам відслідковувати рівень участі кожного члена команди у групових завданнях. Викладачі можуть бачити, скільки часу кожен студент витратив на виконання завдань, як часто він взаємодіє з іншими учасниками команди та які ресурси він використовує.
2. Інструменти Blackboard дозволяють детально аналізувати внесок кожного студента у загальний результат роботи команди. Це корисно для оцінювання індивідуального внеску учасників і коригування підходів до роботи в команді, якщо деякі студенти проявляють пасивність.
3. Звіти про прогрес команд відбуваються автоматично. Система генерує звіти про роботу команд, що дозволяє викладачам отримувати загальну картину про прогрес кожної команди. Ці звіти можуть містити інформацію про дотримання дедлайнів, рівень комунікації між учасниками та якість виконаних завдань.

Попри потужний функціонал, Blackboard має кілька обмежень, які можуть вплинути на ефективність його використання в певних освітніх середовищах:

1. Складність використання Blackboard для нових користувачів. Багато функцій і інструментів можуть здатися надмірно заплутаними, особливо для викладачів або студентів, які не мають досвіду роботи з подібними системами. Це може потребувати значних зусиль для навчання та адаптації користувачів до роботи з системою.
2. Висока ціна платформи Blackboard , і її впровадження в освітні заклади може бути досить дорогим. Вартість ліцензій і підтримки може стати бар'єром для деяких закладів, особливо тих, що працюють з обмеженим бюджетом.

Підусумовуючи, Blackboard — це потужна LMS, яка пропонує гнучкі можливості для автоматизації формування команд і детальний аналіз їхньої роботи. Вбудовані аналітичні інструменти допомагають викладачам відслідковувати прогрес команд, а також аналізувати внесок кожного учасника в спільну роботу. Однак складність у використанні та висока ціна можуть бути суттєвими недоліками для деяких освітніх установ. Незважаючи на це, Blackboard є одним із провідних рішень для тих, хто шукає багатофункціональну платформу для управління навчальним процесом і командною роботою.

1.5.4 Canvans

Canvas[19] — це сучасна система управління навчанням (Learning Management System, LMS), яка використовується в багатьох освітніх закладах для організації дистанційного та очного навчання. Вона відома своєю простотою використання, інтуїтивним інтерфейсом і широкими можливостями для автоматизації навчальних процесів. Однією з основних функцій платформи є можливість автоматизованого розподілу студентів у групи для групових завдань. Canvas пропонує кілька варіантів для створення та управління групами студентів:

1. Автоматичний розподіл студентів в системі Canvas дозволяє автоматично розподіляти студентів у групи на основі кількох критеріїв. Основний варіант — випадковий розподіл, коли студенти рівномірно розподіляються по групах. Це підходить для швидкого організування команд у випадках, коли немає необхідності враховувати особливі характеристики учасників.
2. Розподіл на основі інтересів в системі Canvas — це створення груп на основі інтересів студентів. Викладач може надати студентам можливість обрати групу, яка відповідає їхнім навчальним інтересам або тематиці проєкту. Це дає змогу студентам працювати над завданнями, які їм дійсно цікаві, підвищуючи мотивацію та рівень залученості.

3. Розподіл за результатами навчання підтримує можливість створення команд на основі академічних результатів студентів. Викладачі можуть формувати збалансовані команди, враховуючи рівень знань учасників, що допомагає створювати більш рівноправні умови для виконання складних завдань.
4. Самостійний вибір групи студентами надає студентам можливість самостійно обирати команду, до якої вони хочуть приєднатися, або створювати власні групи. Це дозволяє студентам працювати з тими, кого вони вважають найбільш підходящими партнерами для виконання завдань.

Однією з основних переваг Canvas є її простота та інтуїтивний інтерфейс, що робить процес налаштування команд легким навіть для користувачів, які не мають досвіду роботи з подібними системами:

1. Інтуїтивний інтерфейс, який спрощує процес створення груп та управління ними. Викладачі можуть легко налаштовувати групові завдання, бачити склад команд і керувати процесом роботи студентів.
2. Легкий доступ до групових завдань у Canvas пропонує зручні інструменти для спільної роботи студентів. Всі учасники команди мають доступ до спільних ресурсів, таких як документи або презентації, а також можуть комунікувати через вбудовані інструменти для обміну повідомленнями або обговорень.
3. Підтримка мобільних пристроїв, Canvas є повністю адаптованим під використання на мобільних пристроях, що дозволяє студентам і викладачам організовувати групову роботу та співпрацювати незалежно від того, де вони знаходяться.

Незважаючи на загальну зручність і гнучкість у використанні, Canvas має певні обмеження щодо можливостей формування команд і аналізу їхньої роботи:

1. Обмежені можливості налаштування команд, тобто лише базові варіанти для автоматизації розподілу студентів у команди. Платформа не підтримує складні алгоритми для формування груп, що враховують більше параметрів, таких як особистісні якості студентів, соціальні взаємодії чи інші глибокі показники. Це робить її менш гнучкою для складних освітніх сценаріїв.

2. Відсутність глибокого аналізу командної роботи, інструментів для детального аналізу ефективності командної роботи. В платформі немає інструментів для відстеження внеску кожного учасника або моніторингу взаємодії всередині груп. Це може стати викликом у випадках, коли потрібно оцінити індивідуальний вклад кожного студента у спільну роботу.
3. Брак аналітичних інструментів для команд, обмежені можливості для генерації звітів про роботу команд. В той час, як викладачі можуть бачити результати завершених завдань, вони не мають доступу до детальних даних про процес роботи кожної команди чи рівень комунікації між учасниками.

Canvas є зручною та інтуїтивною платформою для організації навчального процесу і формування команд студентів. Вона пропонує базові можливості для автоматизації розподілу студентів у групи на основі інтересів або результатів навчання, а також дозволяє викладачам вручну налаштовувати склад команд. Проте, платформа має певні обмеження у гнучкості налаштувань та аналізі роботи команд, що може стати викликом для більш складних освітніх сценаріїв.

1.5.5 Узагальнена оцінка існуючих систем

Варто зазначити, що наявність широкого вибору платформ для автоматизації формування команд, нівелюється обмеженими можливостями для глибокого налаштування та аналізу. Ці системи часто дозволяють базове формування груп на основі загальних критеріїв, таких як академічні результати або інтереси учнів, але не завжди здатні враховувати індивідуальні особливості кожного учасника, соціально-емоційні характеристики або стилі навчання, що може значно впливати на ефективність командної роботи.

Інтеграція з іншими освітніми платформаи та можливість кастомізації критеріїв формування команд часто слугуюь ключовими факторомі під час вибору системи, оскільки ці критерії впливають на персоналізовані та адаптовані групи для учнів.

Сучасні освітні технології вимагають інтеграції з іншими навчальними ресурсами, такими як платформи для відеоконференцій, управління оцінками або системи для збору та аналізу результатів навчання. Це дозволяє не лише зручніше організовувати навчальний процес, але й покращувати взаємодію між учасниками команд, здійснювати моніторинг їхнього прогресу в реальному часі та отримувати зворотний зв'язок, що є важливим для коригування процесу навчання. Проте навіть за наявності таких можливостей багато платформ не дають можливості детально налаштовувати параметри формування команд під конкретні цілі курсу чи потреби навчальної програми.

Задля досягнення більш ефективного управління груповою роботою може знадобитися використання сторонніх програмних рішень або розробка власних алгоритмів для персоналізації процесу формування команд. Це дозволяє враховувати не тільки академічні досягнення, але й соціальні взаємодії, мотивацію, психологічний клімат у групах та інші фактори, що можуть впливати на результати роботи. Наприклад, спеціалізовані алгоритми можуть автоматично адаптувати групи на основі перформансу учнів у попередніх проектах, рівня їхнього стресу або готовності до спільної роботи.

Таке рішення дозволяє більш точно налаштувати процеси, забезпечуючи створення команд, які здатні досягати високих результатів завдяки більш глибокому розумінню потреб і можливостей кожного учасника.

Власні алгоритми або сторонні інтегровані рішення можуть також покращити механізми збору та аналізу даних про учнів, що дозволяє створювати більш динамічне та інклюзивне навчальне середовище. Це дає можливість виявляти сильні та слабкі сторони учнів, коригувати складність завдань в залежності від їхнього рівня підготовки, а також персоналізувати навчальні шляхи для кожного учасника. Автоматизовані системи можуть навіть виявити потенційних лідерів або менторів у групах, допомагаючи створювати більш збалансовані та продуктивні команди для досягнення максимальних результатів.

Тому хоча існуючі платформи для автоматизації формування команд можуть бути корисними для загальних завдань, для створення більш персоналізованих і

ефективних групових робіт часто необхідно інтегрувати їх з іншими інструментами або розробляти індивідуальні програмні рішення.

1.6 Висновки до розділу

В цьому розділі було детально розглянуто питання автоматизації формування команд в онлайн-освіті та її вплив на якість навчального процесу. Сучасні реалії вимагають адаптації традиційних підходів до нових умов, де інтерактивність та залученість учасників навчання відіграють ключову роль. Інноваційні технології, інтегровані в освітній процес, стають критичним фактором для підвищення його ефективності.

Командна робота в освітньому середовищі набуває все більшої актуальності, оскільки сприяє розвитку соціальних і професійних навичок учнів. Процес створення ефективних навчальних груп стає важливою складовою успішного навчання, де автоматизовані системи формування команд займають провідне місце. Такі системи не лише забезпечують автоматичний розподіл учасників у групи, але й пропонують аналітичні інструменти для моніторингу та вдосконалення навчального процесу.

Сучасні підходи до формування груп варіюються від випадкового розподілу до більш складних алгоритмів, що враховують академічні показники, особисті інтереси або навіть поведінкові особливості студентів. Це дозволяє створювати групи, які максимально відповідають індивідуальним потребам учасників.

Платформи як Moodle, Google Classroom, Blackboard та Canvas пропонують різні можливості для організації командної роботи, кожна з яких має свої переваги та обмеження. Так, Moodle вирізняється широким функціоналом, але обмежується аналітичними інструментами. Google Classroom є простим у використанні, однак не підтримує глибокого аналізу. Blackboard забезпечує потужні можливості для аналітики, хоча може бути складним для освоєння. Canvas пропонує інтуїтивний інтерфейс, але має обмежені опції для налаштування.

Важливим аспектом автоматизації формування команд є її здатність не лише оптимізувати розподіл учасників, але й створювати умови для адаптивного навчання. Завдяки використанню алгоритмів, що враховують різноманітні параметри, як-от рівень знань, інтереси або навіть психологічні типи студентів, забезпечується індивідуалізація підходу до кожного учасника. Це сприяє підвищенню мотивації, залученості та загальної продуктивності команди, що є основою для досягнення високих результатів у навчальному процесі.

Вибір системи для автоматизації формування команд залежить від специфіки навчального процесу та вимог користувачів. Ефективна система повинна не лише сприяти створенню команд, а й забезпечувати засоби для їхнього моніторингу, аналізу і вдосконалення. Таким чином, автоматизація формування команд стає важливим елементом, який сприяє оптимізації навчального процесу в онлайн-освіті. Подальше дослідження нових технологій є необхідним для задоволення потреб студентів і закладів освіти в умовах динамічного розвитку цієї сфери.

2 ВИБІР АРХІТЕКТУРИ ТА ТЕХНОЛОГІЙ ДЛЯ ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Постановка задачі та визначення основних складових системи

Аналіз існуючих рішень для формування команд, а саме Moodle, Google Classroom, Blackboard, Canvans виявив, що вони з них не враховують усі аспекти, необхідні для оптимального підбору учасників, включаючи психологічну сумісність та індивідуальні характеристики. У зв'язку з цим було прийнято рішення про створення власного продукту, що автоматизує процес формування команд, забезпечуючи більш точний підбір учасників на основі їхніх навичок, досвіду та особистих якостей.

Ця система повинна полегшити процес підбору команд, а й забезпечити зручність для кінцевих користувачів. Інтуїтивний інтерфейс, можливість швидкого створення команд та моніторинг їх прогресу — це ключові компоненти, які покращують користувацький досвід. Система повинна забезпечувати ефективну взаємодію між менеджерами проектів та учасниками, а також мати надійну серверну частину для обробки даних і виконання складних алгоритмів. У цій роботі розглядається створення автоматизованої системи формування команд з акцентом на клієнтській частині, яка забезпечить високу продуктивність та адаптивність.

Постановка задачі полягає у розробці автоматизованої системи формування ефективних команд на основі аналізу навичок, досвіду та інтересів учасників, яка забезпечить гнучкість та швидкість підбору, а також покращення якості проектних результатів через оптимальний розподіл ролей та обов'язків. Інтелектуальна система повинна автоматично збирати та аналізувати інформацію про потенційних учасників, формувати команди з урахуванням складних багатокритеріальних вимог, таких як кваліфікація, досвід роботи, наявність вільного часу, готовність до співпраці та психологічна сумісність. Система також повинна забезпечувати можливість керування процесом роботи команди та надавати аналітичні дані для прийняття управлінських рішень.

Основним задачами для такої системи будуть:

- Автоматизація процесу формування команд на основі навичок студентів.
- Моніторинг прогресу проектів і виконання завдань.
- Надання зручного інструментарію для адміністратора, вчителя та студентів.

Клієнтська частина[20] є ключовим компонентом для взаємодії користувачів з системою. Вона повинна бути інтуїтивною, швидкою та адаптивною до різних платформ (настільних ПК, планшетів, смартфонів). Основний акцент робиться на зручність використання та доступність основних функцій у кілька кліків. Зазначимо наступні вимоги до клієнтської частини:

- Інтуїтивний інтерфейс із мінімалістичним та чітким розподілом функціональних блоків, що забезпечує легку навігацію та швидкий доступ до основних функцій. Для користувачів повинен бути доступ дашборд, щоб спостерігати загальним прогресом проектів, які виконують автоматично згенеровані команди. Також повинен бути доступний перегляд статусів завдань та загальну аналітику.
- Реєстраційна форма повинна надсилатися андміністратором, ісля чого користувачі переходять за лінком і заповнюють форму про свої soft і hard skills. Введені студентами дані зберігаються на сервері та використовуються для автоматизованого формування команд.
- Створення команд повинно відбуватися на основі зібрани даних, система генерує оптимальні команди з урахуванням навичок та критеріїв проекту (кількість учасників, потрібні навички, терміни). Команди повинні бути підваласні редагуванню зі сторони адміністратора, додаючи або виключаючи учасників, а також призначати роль лідера певному студенту.
- Доступність моніторингу та звітність повинна відбуватися за рахунок дашборду, який повинен відображати статуси виконання завдань, прогрес команд та основні аналітичні показники. Інформація також додатково повинна подаватися у вигляді графіків та діаграм у реальному часу.
- Система повинна бути адаптивна, коректно працювати на різних типах девайсів.

Персоналізовані сповіщення повинні інформувати користувачів про зміни в проектах та завданнях.

2.2 Проектування архітектури автомазованої системи

Процес проектування архітектури системи є важливим етапом у створенні алгоретмічного та програмного забезпечення, що забезпечує ефективну взаємодію між клієнтською та серверної частиною для кінцевого. Від правильного вибору архітектури залежаить продуктивність, масштабованість, надійність, зручність експлуатації та підтримка системи.

Автоматизовані системи вимагають адаптивних рішень, які враховують постійно змінювані потреби бізнесу та користувачів. Архітектура системи визначає її загальну структуру, взаємозв’язки між компонентами та спосіб обміну даними. Крім того, добре продумана архітектура дозволяє легко впроваджувати нові функціональні можливості, підтримувати багатокористувацький режим роботи, забезпечувати крос-платформеність та оптимальну швидкодію. Враховуючи різноманітність завдань, які ставляться перед програмним забезпеченням, архітектура системи має бути гнучкою та модульною, що дозволяє адаптуватися до майбутніх змін і вимог.

У випадку автоматизованої системи формування команд архітектура повинна враховувати специфіку аналізу даних про користувачів, таких як їхні навички, досвід, особистісні характеристики та психологічну сумісність. Це вимагає інтеграції інструментів для збору та обробки інформації, алгоритмів машинного навчання або спеціалізованих евристичних методів для ефективного підбору учасників у команди.

Існує багато різних типів архітектури систем[21] які можуть бути використані в залежності від конкретних потреб і вимог проекту. Основні типи архітектури систем включають:

- Монолітна архітектура, яка застосовується у систем, які розгорнуті як єдиний монолітний блок. Усі компоненти та функціональності вбудовані в одну програму або платформу. Цей тип архітектури простий у використанні, але може стати обмеженням для масштабованості та змін;
- У клієнт-серверні архітектурі система[22] розділяється на клієнтську і серверну частини. Клієнти звертаються до серверів для отримання ресурсів та послуг. Цей підхід дозволяє розподіляти обробку даних та ресурси між багатьма серверами, що полегшує масштабування та підтримку більшої кількості клієнтів;
- Розподілена архітектура, при якій компоненти системи розташовані на різних вузлах мережі і взаємодіють через мережу передачі даних. Розподілені системи зазвичай мають велику масштабованість і надійність, оскільки вони можуть бути розгорнуті на багатьох фізичних або віртуальних серверах;
- Мікро сервісна архітектура, при якій системи будується як набір окремих мікро сервісів, які функціонують незалежно один від одного та проводять комунікацію між собою через API. Кожен мікро сервіс відповідає за виконання конкретної функції і може бути розгорнутий, масштабований та оновлюваний незалежно від інших.

При обранні архітектури автоматизованої системи формування команд учнів для онлайн-навчання, важливо обрати такий підхід до вибору, який дозволить не тільки ефективно реалізувати бізнес-логіку, але й вирішити специфічні завдання, визначені аналізом існуючих рішень та вимогами до системи. Зокрема, важливо забезпечити гнучкість та адаптивність, необхідні для оптимального підбору учасників, моніторингу їхньої роботи та інтеграції різних ролей у проєкти.

Основні критерії вибору архітектури:

1. Масштабованість[23], яка дозволить система обробляти великі обсяги даних і аналізувати інформацію про навички, досвід і психологічну сумісність учасників. Клієнт-серверна архітектура здатна централізовано контролювати процеси на сервері, де відбувається обробка та зберігання всіх даних. Це дозволяє забезпечувати гнучке масштабування серверної частини залежно від зростання кількості користувачів і проектів.

2. Гнучкість та адаптивність системи означають, що вона повинна враховувати багатокритеріальні вимоги для підбору команд, такі як навички, досвід роботи та готовність до співпраці. У клієнт-серверній архітектурі серверна частина відповідатиме за збирання та обробку даних студентів, аналізуючи їхні профілі та автоматично формуючи оптимальні команди. Це дозволяє забезпечити централізовану логіку прийняття рішень щодо формування команд, з можливістю ручного коригування з боку адміністратора.
3. Моніторинг та звітність у реальному часі, тобто серверна частина повинна забезпечувати збір інформації про прогрес проєктів та виконання завдань команд. Клієнти (учасники, адміністратори, викладачі) матимуть доступ до цих даних через дашборди на клієнтській частині, що дозволить легко спостерігати за станом завдань та проектів. У клієнт-серверній архітектурі ця взаємодія реалізована за рахунок обробки даних на сервері і надання клієнтам оновлених звітів у реальному часі.
4. Безпека даних повинна відбуватися за рахунок операцій з обробки конфіденційної інформації (навички, досвід, особисті дані учасників) будуть виконуватись на сервері, що дозволить централізовано контролювати доступ до даних та забезпечити належний рівень захисту. Сервер повинен використовувати різні методи шифрування для збереження інформації, а клієнтська частина отримає лише необхідні дані для відображення.
5. Інтуїтивний інтерфейс та багатоплатформність, тобто клієнтська частина повинна бути простою у використанні та адаптованою для різних пристроїв (настільні ПК, мобільні телефони, планшети). Клієнт-серверна архітектура повинна відокремлюватт логіку обробки даних на сервері від інтерфейсу користувача.

Клієнт-серверна архітектура є ефективним вибором для автоматизованої системи формування команд учнів для онлайн-навчання. Вона дозволяє централізовано обробляти складні операції на сервері, забезпечуючи гнучкість, масштабованість та безпеку системи, а також інтуїтивний користувацький інтерфейс, який спрощує взаємодію з кінцевими користувачами.

2.3 Аналіз принципів використання методології IDEF4 для проктування та функціонування системи

Методологія IDEF4[24] (Integration DEFinition for Information and Function Modeling) спеціалізується на проектуванні об’єктно-орієнтованих систем. Вона є однією з родини методологій IDEF, розроблених для сприяння аналізу та проектуванню різних аспектів систем. IDEF4 зокрема орієнтована на проектування та розробку програмного забезпечення. Основні принципи даної методології включають об'єктно-орієнтоване проектування, яке підтримує концепцію класів і об'єктів, що дозволяє створювати абстракції реальних сутностей та їх поведінки. Використання спадкування дозволяє повторно використовувати код та спрощувати архітектуру програмного забезпечення, а інкапсуляція приховує внутрішню реалізацію об'єктів, що сприяє зменшенню взаємозалежностей між різними частинами системи. Проектування в IDEF4 базується на використанні діаграм, які є основним засобом моделювання системи.

UML діаграми класів[25] слугують інструментом моделювання структури програмного забезпечення. Даний тип діаграм призначений щоб візуалізувати класи, їхні атрибути, методи та взаємозв’язки, які описують, як окремі сутності предметної області співпрацюють між собою. Класи відображають об'єкти реального світу або абстракції, їхні властивості (атрибути) та дії (методи). Зв’язки між класами деталізують залежності, взаємодії чи ієрархії, що існують у системі. Ці зв’язки можуть бути асоціаціями (наприклад, автомобіль належить власнику), агрегаціями (автомобіль складається з частин, таких як двигун або колеса), композиціями (сильніша форма агрегації, наприклад, двигун є невід'ємною частиною автомобіля) або залежностями (тимчасова взаємодія між об'єктами, наприклад, автомобіль звертається до сервісного центру).

Даний тип діаграма дає змогу наочно відобразити складність системи, що дозволяє виявити потенційні проблеми з проектуванням на ранніх стадіях розробки. Кожен клас на діаграмі може містити атрибути, які описують його характеристики, а також методи, що визначають поведінку цього класу в рамках

програми. Зв’язки між класами, такі як асоціації, агрегації, композиції та залежності, дозволяють точніше моделювати взаємодії між різними об'єктами в системі, а також визначити типи взаємодій і їхній вплив на функціонування програми. Така візуалізація допомагає в подальшому розвитку та обслуговуванні програмного забезпечення, а також полегшує спільну роботу команди розробників.

UML діаграми об'єктів дозволяють моделювати конкретні стани системи в певний момент часу. Цей тип діаграм дає можливість візуалізувати екземпляри класів, значення їхніх атрибутів та зв'язки, що описують фактичну взаємодію між об'єктами в системі. Об'єктні діаграми показують не загальну структуру, як діаграми класів, а конкретну конфігурацію об'єктів під час виконання. Об'єкти представляють екземпляри класів, які мають конкретні значення своїх атрибутів і реалізують конкретні методи. Зв'язки між об'єктами показують фактичні взаємодії, залежності або асоціації, які формуються під час роботи системи.

Діаграми класів можуть бути використані для відстеження змін у системі, оскільки вона демонструє, як об'єкти створюються, змінюються та взаємодіють один з одним протягом виконання програми. Важливим аспектом є те, що об'єктні діаграми дозволяють візуалізувати конкретні значення атрибутів об'єктів, що забезпечує точніше уявлення про внутрішні процеси програми та взаємодію між її частинами. Ці діаграми допомагають не тільки на етапі розробки, але й у процесі налагодження та тестування, оскільки надають розробникам чітке уявлення про поточний стан системи в будь-який момент її роботи.

UML діаграми компонентів є інструментом моделювання, для відображення високорівневої архітектуру системи, програмні компоненти та їхні взаємозв’язки. Цей тип діаграм дає змогу візуалізувати, як окремі частини системи взаємодіють, щоб забезпнчувати функціональність системи. Компоненти є модульними елементами, які реалізують певний набір функцій, і можуть представляти собою бібліотеки, модулі, веб-сервіси або інші програмні одиниці. На діаграмі компонентів відображаються не лише самі компоненти, але й їхні інтерфейси, що вказують, які сервіси вони надають або використовують. Інтерфейси поділяються на реалізовані (надані) та використовувані, що показує, як компоненти взаємодіють

один із одним. Діаграми компонентів також допомагають візуалізувати залежності між компонентами, що дозволяє зрозуміти, як зміни в одному компоненті можуть вплинути на інші частини системи. Це є важливим аспектом при плануванні масштабування або оптимізації системи, оскільки дозволяє передбачити потенційні проблеми, що можуть виникнути через взаємозв'язки між компонентами.

UML діаграми розгортання представляють фізичну архітектуру системи, моделюючи умовне розміщення програмних компонентів на апаратних вузлах і представляючи їхню взаємодію. Даний тип діаграм зображує, як система працює в реальному середовищі, включаючи сервери, пристрої користувачів, мережеві з'єднання та інші елементи інфраструктури. На діаграмі вузли можуть бути фізичними або віртуальними і представляти пристрої, сервери або хмарні ресурси, в той час як компоненти показують програмні частини, розгорнуті на цих вузлах. Зв'язки між вузлами ілюструють передачу даних або задачі сервісів, відображаючи реальні механізми взаємодії між різними частинами системи.

Використання UML діаграми розгортання дозволяє деталізувати середовище виконання, забезпечити ефективну інтеграцію програмного забезпечення з апаратними ресурсами та полегшити планування, оптимізацію та обслуговування системи. Цей тип діаграм також є корисним для візуалізації масштабованості системи, оскільки показує, як окремі компоненти можуть бути перенесені або додані до інших вузлів для покращення продуктивності або балансування навантаження. Використання діаграм розгортання допомагає забезпечити надійність системи, даючи змогу визначити потенційні точки відмови та шляхи для резервного копіювання або відновлення.

UML діаграми варіантів використання моделюють функціональні аспекти системи, її взаємодію із зовнішніми користувачами або іншими системами. Даний тип діаграм дозволяє візуалізувати завдання або функції, які система може виконувати, згідно до потреб користувачів. Основними її елементами є актори, які представляють зовнішніх користувачів або системи, і варіанти використання, які описують дії або послуги, доступні під час взаємодії із певною системою. Зв'язки між акторами та варіантами використання дозволяють зобразити функції, доступні

для конкретного актора в конкретний момент часу. Діаграми варіантів використання корисні для аналізу вимог і визначення функціональності системи. Також ці діаграми дозводяють кразе зрозуміти поведінку системи з точки зору кінцевого користувача, визначити основні робочі сценарії та уточнити обмеження системи.

Ці діаграми є незамінним інструментом на етапі проектування, оскільки вони дозволяють розробникам зрозуміти, як система повинна функціонувати для кінцевого користувача, що полегшує подальше створення детальних специфікацій. Використання діаграм варіантів використання також допомагає забезпечити точність і повноту функціональних вимог, що є основою для успішної реалізації проекту.

Інструменти моделювання дозволяють дозволяють проводоити ефективну розробку та аналіз автоматизованої системи формування команд учнів. Застосування методології IDEF4 і UML-діаграм дозволяє створити комплексне уявлення про систему, її компоненти, взаємодії між ними та спосіб функціонування в реальному середовищі.

Методологія IDEF4 зосереджується на об’єктно-орієнтованому підході до проектування, що дозволяє будувати систему з урахуванням ключових принципів: інкапсуляції, спадкування та поліморфізму. Це спрощує реалізацію повторно використовуваних компонентів і підвищує масштабованість системи, що є критичним для освітньої платформи з великою кількістю користувачів.

UML-діаграми додають до процесу розробки універсальність і деталізацію. Діаграми класів надають структурний опис системи, а діаграми об’єктів демонструють, як ці класи взаємодіють під час виконання програми. Діаграми компонентів забезпечують чітке уявлення про модульну структуру системи, дозволяючи краще спланувати її розгортання та підтримку. Діаграми розгортання деталізують фізичне середовище, в якому система функціонує, а діаграми варіантів використання допомагають розкрити її функціональні можливості, орієнтуючись на користувача.

2.4 Вибір технологій для реалізації клієнтської частини системи

Архітектура автоматизованої системи формування команд учнів для онлайн-навчання базується на клієнт-серверній моделі, яка забезпечує розділення функцій між клієнтською частиною[26] (інтерфейсом для користувачів) та серверною частиною (обробкою даних і бізнес-логікою). У клієнт-серверній архітектурі користувачі звертаються до сервера для отримання інформації та виконання дій через зручний веб-інтерфейс. Важливим аспектом такої архітектури є вибір технологій для реалізації клієнтської частини, яка є критичною для забезпечення ефективної взаємодії з сервером та зручного користування системою.

При виборі технологій для реалізації клієнтської частини автоматизованої системи формування команд учнів для онлайн-навчання важливо розглянути не лише можливості створення привабливого інтерфейсу, а й забезпечити високу продуктивність, масштабованість та гнучкість для подальшого розвитку системи. Основними технологіями розробки, які обрані для виконання поставленого завдання, є HTML, CSS, та Vue.js. Кожна з цих технологій забезпечує специфічні функції, необхідні для побудови сучасного веб-додатка, що дозволяє ефективно управляти командами учнів та контролювати прогрес у навчанні.

HTML (HyperText Markup Language)[27] слугує інструментом створення структури веб-сторінок та веб-додатків. Цей інструмент дозволяє побудувати мапу сайту, з різноманітних елементів: заголовків, абзаців, списків, форм, таблиць, зображень, відео, аудіо та інтерактивних компонентів. HTML дозволяє розробникам створювати чітку структуру контенту. Кожен елемент визначається за допомогою конкретних тегів, які вказують браузеру, як відображати той чи інший контент.

Основні елементи HTML є заголовки від <h1> до <h6> для позначення рівнів важливості тексту; абзаци <p> для структуризації тексту; списки <ul> (марковані) та <ol> (нумеровані); гіперпосилання <a> для навігації між сторінками; зображення <img> для інтеграції графічного контенту; мультимедійні елементи <video> та <audio> для відтворення відео та аудіофайлів без додаткових

плагінів. Крім того, HTML дозволяє створювати форми для збору даних, реєстрації користувачів, пошуку та взаємодії з системою через елементи вводу <input>, випадаючі списки <select> та кнопки <button>.

HTML забезпечує кросплатформенність, оскільки підтримується всіма сучасними браузерами, включаючи Chrome, Firefox, Safari та Edge. Це робить веб-додатки доступними для різних пристроїв — від комп'ютерів до мобільних телефонів. HTML5, остання версія мови, значно розширила функціональність, запропонувавши нові семантичні теги (<header>, <footer>, <article>), вбудовані API для малювання на Canvas, роботи з геолокацією, локальним сховищем та багатопотоковістю через Web Workers.

Варто додати що HTML має змогу інтегруватися із CSS та JavaScript. CSS дозволяє стилізувати сторінки, змінюючи кольори, розміри, відступи та інші візуальні аспекти. JavaScript додає інтерактивність, наприклад, можливість обробки подій, створення анімацій або взаємодії з сервером у реальному часі. Разом ці технології формують основу для створення сучасних, адаптивних і функціональних веб-додатків. HTML також легко інтегрується з іншими веб-технологіями, такими як фреймворки Angular, React або Vue, що дозволяє створювати складні односторінкові додатки (SPA). Завдяки своїй простоті та потужності HTML є універсальним інструментом для розробки веб-рішень будь-якої складності.

CSS (Cascading Style Sheets)[28] є допоміжним іструментом, який відповідає за оформлення та стилізацію HTML-документів, додаючи до нього візуальну привабливість та забезпечуючи зручну навігацію для користувачів. Ця технологія дозволяє розділяти структуру веб-сторінок від її зовнішнього вигляду, що значно полегшує процес розробки, масштабування та підтримки веб-додатків. CSS дозволяє налаштовувати шрифти, кольори, розміри, позиціювання елементів, а також створювати складні візуальні ефекти, такі як градієнти, тіні та анімації.

Однією з ключових переваг CSS є його здатність забезпечувати адаптивний дизайн через використання медіа-запитів (@media), які дозволяють автоматично змінювати вигляд сторінки залежно від типу пристрою чи розміру екрану. Це

особливо важливо для таких систем, як автоматизоване формування команд для онлайн-навчання, де користувачі можуть працювати зі смартфонів, планшетів чи комп'ютерів.

Адаптивність гарантує, що система однаково зручно виглядає та функціонує на будь-якому пристрої — від мобільних телефонів учнів до десктопів викладачів. CSS також пропонує можливість створення динамічних візуальних ефектів. За допомогою властивостей анімацій (animation, @keyframes), перехідних ефектів (transition) та інших засобів стилізації розробники можуть оживляти інтерфейс, роблячи взаємодію користувачів із системою більш приємною та інтуїтивною. Крім того, сучасні інструменти CSS, такі як Flexbox і CSS Grid, дозволяють створювати гнучкі та складні макети з точним позиціонуванням елементів.

Інтеграція CSS із JavaScript дозволяє змінювати стилі в реальному часі, створюючи інтерактивні інтерфейси з динамічними елементами. Завдяки цьому CSS є незамінним інструментом у розробці веб-додатків, забезпечуючи покращений користувацький досвід (UI/UX) та роблячи веб-інтерфейси естетично привабливими та функціональними.

Vue.js[29] є одним із провідних JavaScript-фреймворків для створення користувацьких інтерфейсів, і його вибір для клієнтської частини системи автоматизованого формування команд був обумовлений кількома ключовими факторами. Vue.js забезпечує реактивність, що дозволяє автоматично оновлювати відповідні компоненти на сторінці при зміні даних, що важливо для динамічних систем, де інформація постійно змінюється, наприклад, при додаванні нових завдань або оновленні прогресу команд. Компонентна структура Vue.js дає змогу розбивати додаток на незалежні блоки, кожен з яких відповідає за конкретну функцію — це спрощує не лише розробку, але й підтримку системи, оскільки кожен компонент можна легко оновлювати й тестувати окремо.

Також фреймворк відомий своєю високою продуктивністю та масштабованістю завдяки оптимізованому рендерингу компонентів і ефективному управлінню станом за допомогою бібліотеки Vuex. Це дозволяє системі працювати плавно навіть при великій кількості користувачів і взаємодій між ними. Vue.js легко

інтегрується з іншими інструментами, такими як Axios для обробки HTTP-запитів, що значно спрощує побудову додатків, які активно взаємодіють із сервером, та Vue Router, що дозволяє зручно реалізовувати маршрутизацію сторінок у складних SPA (Single Page Applications). Також Vue.js підтримує безшовну інтеграцію з бекендом через API, що дозволяє легко оновлювати дані на клієнтській частині та взаємодіяти з серверною частиною, забезпечуючи ефективну та швидку роботу системи.

Слід додати, що використовуючи Vue.js у поєднанні з CSS, можна створювати інтерфейси, які автоматично підлаштовуються під різні екранні розміри та пристрої. Це забезпечує зручність користування як на мобільних пристроях учнів, так і на робочих станціях викладачів, що критично важливо для онлайн-системи формування команд, де учасники можуть мати різні типи пристроїв. Завдяки широким можливостям для інтеграції з різноманітними API та сторонніми бібліотеками, Vue.js дозволяє легко додавати в систему такі функції, як реєстрація, авторизація, система оцінок, повідомлення в реальному часі та багато інших. Це робить систему універсальним інструментом для освітніх закладів, що дозволяє автоматизувати не тільки процеси формування команд, але й значно покращити ефективність навчального процесу в цілому.

Завдяки своїй гнучкості, високій продуктивності та простоті інтеграції з іншими технологіями, Vue.js є чудовим вибором для розробки динамічних та адаптивних веб-додатків, таких як система автоматизованого формування команд учнів. Це дозволяє створювати ефективні, масштабовані і функціональні інтерфейси, що відповідають вимогам сучасної онлайн-освіти.

2.5 Вибір бази даних та протоколу взаємодії між клієнтською та серверною частинами системи

У контексті розробки автоматизованої системи для формування команд учнів важливу роль відіграє правильний вибір бази даних, що забезпечить ефективне

зберігання, обробку та доступ до великого обсягу даних, з якими система працюватиме. Оскільки система буде реалізована в клієнт-серверній архітектурі, важливо враховувати, що база даних повинна не лише зберігати дані, але й бути здатною швидко взаємодіяти з іншими компонентами системи, надаючи клієнтам необхідну інформацію в реальному часі. Клієнт-серверна архітектура передбачає, що база даних виступатиме як центральний елемент, до якого звертаються всі компоненти системи — від вебінтерфейсу до серверної логіки, що обробляє запити.

У такій архітектурі база даних повинна підтримувати високий рівень доступності та масштабованості, що дозволяє системі легко обробляти великі обсяги інформації, яку генерують користувачі, а також бути готовою до змін у навантаженні, наприклад, під час пікових періодів використання системи. Клієнт-серверна архітектура, в свою чергу, дозволяє досягти чіткого розподілу ролей і відповідальностей, де сервер відповідає за обробку запитів до бази даних, а клієнт забезпечує зручний інтерфейс для користувачів. Для забезпечення ефективної роботи з великими даними, важливо обирати таку базу даних, яка дозволить забезпечити високу швидкість запитів, надійність і безпеку в умовах одночасної взаємодії багатьох користувачів.

Під час вибору баз даних для автоматизованої системи формування команд учнів вибір впав до MariaDB через її високі технічні можливості, масштабованість і стабільність, що відповідають сучасним вимогам до роботи з великими обсягами даних. При розробці освітніх платформ, важливо враховувати особливості учнів, їхні навички, інтереси та інші параметри для ефективного формування команд, обрана база даних пропонує ідеальні рішення для обробки та зберігання такої інформації.

Автоматизована система формування команд учнів потребує постійного безперебійного доступу до даних і забезпечення їхньої цілісності під час взаємодії багатьох користувачів одночасно. MariaDB має підримку ACID-транзакцій, що гарантує коректність даних у складних сценаріях, таких як динамічне оновлення даних про учасників або статус команд. Гнучкість у роботі з різними типами даних, включно з JSON, дозволяє ефективно обробляти напівструктуровану інформацію,

що є ключовою для адаптивного формування груп на основі персональних характеристик студентів.

MariaDB також забезпечує легку інтеграцію з сучасними технологіями та фреймворками, що сприяє більшій швидкості та зручності розробки системи. Її відкритий код і безкоштовна ліцензія роблять її доступним рішенням, яке забезпечує високу продуктивність і відповідає вимогам безпеки, що є критично важливими для освітніх платформ. Усе це робить MariaDB ідеальним вибором для розробки бази даних, яка стане основою ефективного управління навчальними процесами та командною роботою.

Процес вибору протоколу взаємодії між клієнтською та серверною частинами системи автоматизованого формування команд учнів для онлайн-навчання через API є важливим етапом у розробці системи. API (Application Programming Interface)[30] є набором правил і протоколів, які дозволяють різним програмним компонентам взаємодіяти між собою, і в рамках цієї системи він забезпечує зручний спосіб обміну даними між клієнтською та серверною частинами. Одним із найефективніших варіантів є вибір RESTful API на базі HTTP/HTTPS. RESTful API (Representational State Transfer) є стилем архітектури програмного забезпечення, який використовує HTTP/HTTPS для взаємодії між клієнтом і сервером. Цей протокол має кілька важливих переваг. По-перше, простота реалізації — RESTful API легко реалізувати за допомогою стандартних HTTP-методів (GET, POST, PUT, DELETE), що робить його доступним для розробників.

Використання JSON або XML для обміну даними дозволяє легко інтегрувати API з клієнтською частиною, особливо з такими технологіями, як Vue.js. По-друге, RESTful API забезпечує кросплатформену сумісність, адже працює на базі стандартних HTTP-протоколів, що дозволяє інтегрувати його з будь-якою платформою, яка підтримує HTTP, включаючи веб-додатки, мобільні додатки та серверні рішення.

Варто зазначити, що використання HTTPS забезпечує шифрування даних під час їх передачі. Це особливо важливо для системи, що обробляє особисті дані учнів

та викладачів. Гнучкість і масштабованість RESTful API дозволяють легко додавати нові функції та ендпоінти, що робить систему масштабованою і здатною підтримувати зростаючі вимоги. Можливість використовувати різні формати відповіді (наприклад, JSON) для задоволення специфічних потреб клієнтів також сприяє цьому. Кешування запитів у RESTful API зменшує навантаження на сервер і підвищує швидкість відповіді на запити клієнтів, що також є важливим аспектом для забезпечення ефективної роботи системи. Крім того, REST дозволяє чітко визначити ресурси (команди, учні, завдання) і їх представлення у формі ендпоінтів, що спрощує навігацію та використання API.

Таким чином, вибір RESTful API на базі HTTP/HTTPS[31] для взаємодії між клієнтською та серверною частинами автоматизованої системи формування команд учнів є обґрунтованим рішенням. Це забезпечує простоту, безпеку, гнучкість і масштабованість системи, що дозволяє ефективно управляти командами учнів, контролювати їхній прогрес і адаптуватися до змінюваних потреб користувачів.

2.6 Висновки до розділу

У даному розділі було визначено ключові вимоги до функціональності, зручності використання та технологічного стеку, які необхідні для коректної та якісної розробки автоматизованої системи формування команд учнів для онлайн-навчання. Було визначено, що клієнт-серверну архітектуру є оптимальним рішенням для забезпечення ефективної роботи системи. Ця модель дозволяє гнучко змінювати бізнес-логіку, легко масштабувати систему для підтримки великої кількості користувачів, забезпечувати централізовану безпеку даних та ефективний розподіл ресурсів між клієнтом та сервером.

Також було затверджено, що клієнтська частина повинна бути розроблена з використанням сучасних технологій, таких як HTML, CSS і Vue.js, що забезпечить створення інтуїтивно зрозумілого інтерфейсу, адаптивного дизайну та високої продуктивності. HTML надає структурну основу для веб-сторінок, CSS забезпечує коректне відображення на різних пристроях, а Vue.js дозволяє створювати

динамічні, реактивні веб-додатки завдяки своїй компонентній архітектурі, що спрощує управління станом додатка та його тестування.

Використання Vue.js забезпечує високу швидкість завантаження сторінок, інтерактивність та зручність інтеграції із серверною частиною через REST API або GraphQL. Такий підхід сприяє подальшому масштабуванню системи, включно з додаванням нових модулів. Обрана архітектура та стек технологій відповідають сучасним вимогам розробки веб-додатків і забезпечують стабільність, продуктивність та зручність для всіх користувачів.

Таким чином, система щоб буде розроблятися стане ефективним інструментом для організації онлайн-навчання та взаємодії між учасниками освітнього процесу.

3 РОЗРОБКА АЛГОРЕТМІЧНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ОНЛАЙН НАВЧАННЯ

3.1 Визначення структури та компонентів автоматизованої системи формування команд

Розробка клієнтської частини автоматизованої системи формування команд спрямована на створення адаптивного та інтуїтивно зрозумілого інтерфейсу, який надає користувачам легкий доступ до всіх необхідних функцій. Інтерфейс повинен підтримувати роботу на різноманітних пристроях, включаючи комп'ютери, планшети та мобільні телефони, що забезпечує безперешкодний та позитивний користувацький досвід. Ключові компоненти клієнтської частини включають модуль реєстрації користувачів[32], інтерфейс для автоматизованого формування команд на основі заданих параметрів, а також панель відображення актуальної інформації про створені команди, їхній прогрес та індивідуальні показники учасників.

Для реалізації клієнтської частини автоматизованої системи формування команд необхідно побудувати діаграми, які детально відображають структуру, взаємодію компонентів і функціональні можливості інтерфейсу. Ці діаграми служать основою для візуалізації та аналізу внутрішньої логіки системи, а також взаємодії між різними її частинами. Вони дозволяють не лише зрозуміти, як працює система в цілому, але й сприяють ефективному проектуванню, покращенню зручності користувача та оптимізації процесів розробки. Завдяки цим діаграмам можна наочно побачити, як компоненти клієнтської частини взаємодіють один з одним і з сервером, як дані передаються між різними елементами інтерфейсу, а також як забезпечується відображення інформації та виконання основних функцій системи.

Діаграма класів дозволяє відображати основні класи, що відповідають за побудову інтерфесу та організацію його компонентів та їх взаємозв'язки в системі автоматизованої генерації команд учів. Перед створення такої діаграми варто зазначити основні класи, їх атрибути та методи, а також зв'язки між класами.

1) Клас «UserProfile» — даному класу будуть присвоєні наступні атрибути:
   - userId: int — унікальний ідентифікатор користувача;
   - personalInfo: string — особиста інформація користувача (ім'я, прізвище тощо);
   - achievements: string — досягнення користувача;
   - skills: string — технічні навички користувача;
   - softSkills: string — м'які навички користувача (комунікація, лідерство, робота в команді тощо);
   - subjectInterests: array — інтереси до предметів (оціночна шкала для кожного предмету, від 1 до 5);
   - globalInterests: string — глобальні інтереси користувача, що включають технічні інтереси;

   Методи:
   - viewProfile(): відображає профіль користувача;
   - editProfile(): дозволяє змінити особисті дані користувача;
   - addSkill(): додає нову технічну навичку користувачу;
   - updateInterests(): оновлює предметні та глобальні інтереси користувача;
   - evaluateSoftSkills(): оцінює м'які навички користувача;
   - calculateInterestCompatibility(): обчислює сумісність інтересів із іншим користувачем;
2) Клас «UserRegistration» — даному класу будуть присвоєні наступні атрибути:
   - username: string — ім'я користувача;
   - email: string — електронна адреса користувача;
   - password: string — пароль користувача;
   - role: string — роль користувача (студент, вчитель, адміністратор);
   - birthDate: string — дата народження користувача;

- gender: string — стать користувача;
- leaderStatus: статус лідера;
- softSkills: string — м'які навички користувача;
- subjectInterests: string — інтереси до предметів;
- globalInterests: string — глобальні інтереси користувача (технічні, ігрові тощо);

Методи:

- registerUser(): реєструє нового користувача;
- validateInput(): перевіряє коректність введених даних;
- assignRole(): призначає роль користувачу;
- setLeaderStatus(): оновлює статус лідера;
- evaluateSoftSkills(): оцінює м'які навички користувача для командного розподілу;
- createUserProfile(): створює новий об’єкт UserProfile на основі введених даних;
- calculateTotal(): оцінює загальні навики користувача;

3) Клас «Team» — даному класу будуть присвоєні наступні атрибути:

- teamId: int — унікальний ідентифікатор команди;
- teamMembers: array — список учасників команди (список об’єктів UserProfile);
- skillsRequired: string — набір технічних навичок, необхідних для проєкту;
- softSkillsRequired: string — перелік м'яких навичок, необхідних для проєкту (лідерство, комунікація тощо);
- subjectInterests: string — інтереси команди по дисциплінах;
- globalInterests: string — глобальні інтереси команди;

Методи:

- createTeam(): формує команду на основі профілів студентів.
- assignMember(): додає учасника до команди.

- matchSkills(): порівнює технічні навички учасників із вимогами проєкту.
- evaluateSoftSkillsMatch(): порівнює м'які навички учасників із вимогами.
- calculateTeamInterestProfile(): розраховує загальні інтереси команди на основі інтересів її членів.
- generateTeamReport(): створює звіт про поточний склад команди, її сильні сторони та інтереси.

4) Клас «Project» — даному класу будуть присвоєні наступні атрибути:
   - projectId: int — унікальний ідентифікатор проєкту;
   - projectName: string — назва проєкту;
   - description: string — опис проєкту;
   - teamAssigned: string — команда, призначена для виконання проєкту;
   - direction: string — напрямок проєкту;
   - directionSpecificRequirements: string — специфічні вимоги для обраного напрямку проєкту (певні навички, технічні знання або програмне забезпечення);

     Методи:
   - assignTeam(): призначає команду до проєкту;
   - trackProgress(): відстежує прогрес виконання завдань у проєкті;
   - evaluateCompletion(): оцінює завершення завдань проєкту;
   - setDirection(): встановлює напрямок проєкту;
   - checkDirectionRequirements(): перевіряє відповідність команди специфічним вимогам напрямку;
   - generateProjectSummary(): створює підсумковий звіт проєкту, включаючи прогрес і відповідність команди вимогам;
5) Клас «Dashboard» — даному класу будуть присвоєні наступні атрибути:
   - id: int — Унікальний ідентифікатор дашборду;

- associatedTeam: string — Посилання на команду (клас Team), для якої створено дашборд;
- associatedProject: string — Посилання на проєкт (клас Project), що пов’язаний із цим дашбордом;
- tasks: array — Колекція завдань (список об’єктів класу Task);
- description: string — Опис дашборду (загальний огляд призначення та функцій дашборду);

Методи:

- addTask(task: Task): Додає нове завдання до дашборду.
- removeTask(taskId: int): Видаляє завдання за його ідентифікатором.
- updateTask(taskId: int, newDetails: dict): Оновлює деталі завдання.
- editDashboard(newDetails: dict): Редагує основні параметри дашборду, такі як опис чи зв’язок із проєктом або командою.
- generateProgressReport(): Генерує звіт про виконання завдань.

6) Клас «Task» — даному класу будуть присвоєні наступні атрибути:

- taskId: int — Унікальний ідентифікатор завдання;
- title: string — Назва завдання;
- description: string — Опис завданнz;
- type: string — Тип завдання;
- status: string — Статус завдання;
- assignedTo: array — Список користувачів (клас UserProfile), яким призначено завдання;
- dueDate: date — Термін виконання завдання;
- priority: array — Пріоритет завдання;

Методи:

- assignUser(): Призначає користувача до завдання;
- updateStatus(): Оновлює статус завдання;
- edittask(): дозволяє редагувати інформацію в самому таску;

Зв'язок між класами UserRegistration та UserProfile визначається як агрегація, оскільки UserRegistration містить дані з UserProfile. Під час процесу реєстрації створюється профіль користувача, який включає основну інформацію, як-от особисті дані та інтереси. Цей зв’язок дозволяє забезпечити інтеграцію даних користувача для подальшого використання в системі.

Зв'язок між UserRegistration та Team є композицією, оскільки після реєстрації користувачів їхні дані передаються до Team для аналізу та формування команд. Композиція вказує на тісну залежність між цими класами, адже Team неможливий без даних, отриманих через UserRegistration.

Зв'язок між класами UserProfile та Project є асоціацією, що дозволяє одному користувачу бути задіяним у кількох проектах. Цей зв’язок демонструє залежність між навичками користувача, збереженими в його профілі, та проєктами, у яких він може брати участь. Це важливо для оптимального розподілу учасників між проєктами.

Зв'язок між Team та UserProfile визначається як багато до багатьох (асоціація), оскільки кожна команда складається з кількох користувачів, а один користувач може бути частиною кількох команд. Це дозволяє створювати динамічні групи для різних завдань і проєктів, забезпечуючи гнучкість у роботі команд.

Між класами Project та Team встановлено зв'язок один до багатьох, де один проєкт може бути розподілений між кількома командами, але кожна команда працює лише над одним проєктом. Це забезпечує чітке розмежування обов’язків між командами, які працюють над різними частинами або аспектами одного проєкту.

Зв'язок між Dashboard та Team є асоціацією один до одного, де кожен дашборд належить одній команді, а кожна команда має лише один дашборд. Це спрощує відстеження прогресу команди через інтерфейс дашборду, який об’єднує всі завдання та показники продуктивності.

Зв'язок між класами Dashboard та Project також є асоціацією один до одного, оскільки кожен дашборд відповідає одному проєкту, а проєкт має один дашборд. Це забезпечує структуроване управління проєктом, дозволяючи відображати всю ключову інформацію на єдиній платформі.

Зв’язок між класами Dashboard та Task встановлено зв'язок агрегації, де дашборд включає список завдань. Завдання можуть існувати незалежно від дашборду, але вони логічно об'єднані через нього для зручності управління командною роботою.

Зв'язок між Task та UserProfile є асоціацією, визначається як багато до багатьох, адже завдання можуть бути призначені кільком користувачам, а один користувач може виконувати кілька завдань. Такий підхід дозволяє врахувати навички кожного учасника та забезпечити рівномірний розподіл завдань між членами команди.

Діаграма класів демонструє чітку організацію та взаємодію компонентів системи для управління навчальними командами й проєктами. Клас UserRegistration відповідає за ініціалізацію користувачів, а UserProfile зберігає їхні дані для формування команд. Клас Team забезпечує гнучку організацію командної роботи, дозволяючи залучати користувачів з різними навичками. Dashboard відстежує прогрес команди, асоціюючись з конкретними проєктами та завданнями (Task), що полегшує управління й контроль виконання. Зв'язки між класами, зокрема асоціації, агрегації та композиції, створюють основу для масштабованої, інтегрованої й адаптивної системи, що підтримує ефективне управління навчальними командами. Діаграму класів візуалізована та продеменстровано у додатку Б.

Після створення діаграми класів для автоматизованої системи генерації команд учнів, важливо розробити діаграму об'єктів, оскільки це дасть змогу конкретизувати, як класи реалізуються в контексті конкретних об'єктів під час виконання програми. Діаграму об’єктів візуалізована та продеменстровано у додатку Б.

Також варто побудувати UML-діаграму компонентів, оскільки вона надає можливість візуалізувати основні функціональні модулі системи, їхню структуру та взаємодію між собою. Ця діаграма допомагає не лише виявити логічні зв’язки між компонентами, але й чітко показує ролі кожного з них у загальному процесі роботи системи. Для UML-діаграми компонентів виділимо наступні компоненти:

- UserRegistration — компонент, який відповідає за реєстрацію нових користувачів та створення їх профілів із конкртеною інформацію, яка була надана користувачем;
- Team — компонент, який відповідальний за збір обробку данних користувачів, які потрібні для аналізу та автоматичного генерування команд учнів;
- TaskManagement — компонент, що використовується для створення, редагування та моніторингу виконання завдань учасниками конкретної команди над проєктом;
- User — додатковий компонент, який взаємодіє із компонентом UserRegistration для реєстрації із TaskManagement для керування завданнями;

Варто також зазначити наступні інтерфейси для даних компонентів:

- UserDataInterface — інтерфейст, що створюється компонентом UserRegistration для передачі даних заповненим користувачем. Даний інтерфейс використовується компонентом Team для отримання цих даних;
- TeamDataInterface — інтерфейст, що створюється компонентом Team для передачі сформованих команд учнів в компонент TaskManagement для подальшого управління завданнями;

Діаграма компонентів візуалізована та продеменстровано на рис. 3.1

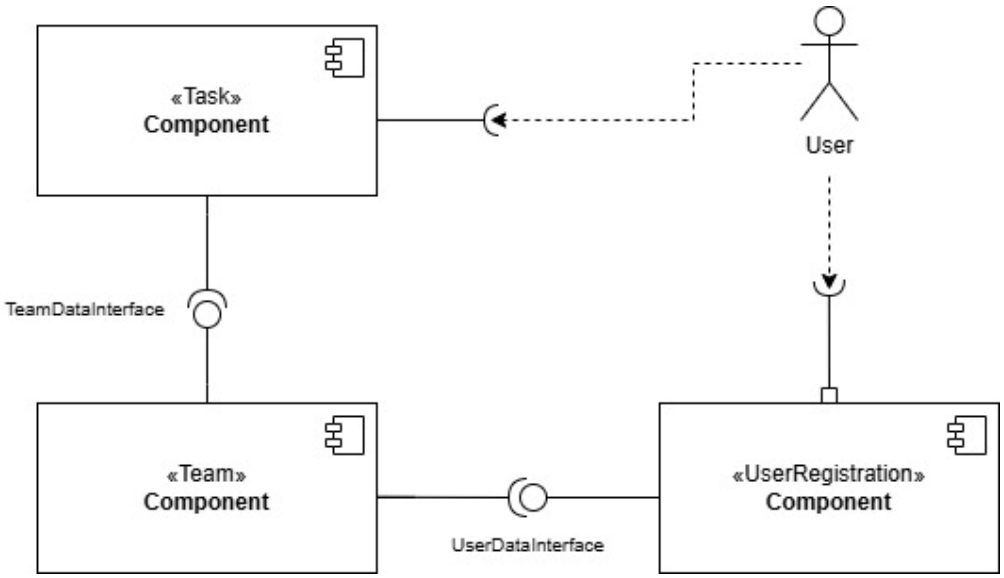


Рисунок 3.1 — Діаграма компонентів системи

Візуалізована вище діаграма компонентів ілюструє структуру автоматизованої системи формування команд учнів, демонструючи складові цієї системи та їх взаємодію. Компонент UserRegistration відповідає за процес реєстрації нових користувачів і передачу введених даних. Даний процес відбувається через інтерфейс UserDataInterface, взаємодіючи при цьому із актором User, який безпосередньо й вводить дані. Компонент Team проводить аналіз отриманих та проводить формування команд за отриманими результатами, і передає їх через інтерфейс TeamDataInterface до компонента Task, який у свою чергу відповідає за створення, редагування та моніторинг завдань. Зв'язки між компонентами забезпечують модульність і дозволяють розширювати систему.

Також для детального аналізу системи формування команд необхідно побудувати діаграму розгортання, яка дозволить зобразити фізичне розташування компонентів у середовищі виконання. На ній повинни бути продемонстровано серверну частину з основними компонентами системи, клієнтську частину для взаємодії користувача з системою, а також зв'язки між сервером, клієнтом і базою даних.

Діаграма розгортання повинна візуалізувати взаємодію користувача з системою через браузер, який візуалізується як компонент Client Tier. Запити від

користувача передаються до Web Tier через HTTP/HTTPS протоколи, які забезпечують зв'язок між клієнтом і сервером. У відповідь веб-сервер повинен формувати сторінки у форматі HTML або інших даних (JSON, XML), які потім відображаються в браузері для користувача.

Прооцес обробки даних, наприклад, створення нового завдання, відбувається заразунок передачі інформації від компоненту Web Tier через User Interface Layer до компоненту Application Tier, де вкладена Business Logic Layer та Data Access Layer. Business Logic Layer обробляє запит, виконує необхідну операцію і, у разі потреби, передає його до Data Access Layer, який здійснює запит до бази даних для отримання або зміни даних. Після цього дані передаються назад до Business Logic Layer, яка формує відповідь для користувача і передає її до Web Tier.

Data Access Layer в компоненті Application Tier взаємодіє з Data Tier через TCP/IP, що забезпечує безпечне та ефективне виконання SQL-запитів до бази даних. Дані зберігаються та обробляються на сервері, що підтримує SQL Server, який забезпечує доступність і безпеку даних. Лог-файли, що містять інформацію про помилки та інші важливі події системи, зберігаються на сервері для подальшого аналізу. Всі ці компоненти об'єднуються для забезпечення ефективного і безпечного функціонування системи, зберігання даних та обробки запитів користувачів. На рис. 3.2 зображено діаграму розгортання:

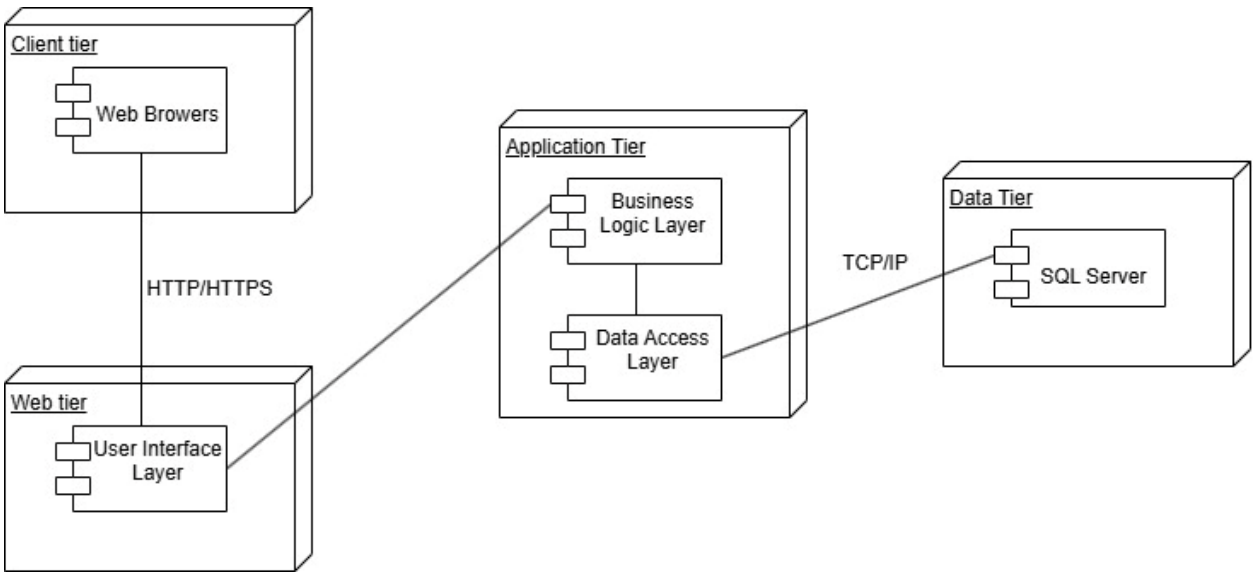


Рисунок 3.2 — Діаграма розгортання системи

Візуалізуємо взаємодію користувіч із системою за допомогою діаграми варіантів використання для автоматизованої системи формування команд учнів демонструє. Ця взаємодія дозволя виконанувати ключових завдання системи, пов’язаних із реєстрацією, формуванням команд, управлінням завданнями та оцінкою прогресу. У рамках цієї системи різні ролі користувачів (адміністратор, студент, викладач) мають чітко визначені функції та доступ до певних модулів і можливостей системи, що забезпечує її ефективну роботу.

Основною метою цієї системи є автоматизація процесу створення збалансованих команд учасників для виконання спільних проєктів, надання інструментів для управління командними завданнями та забезпечення прозорого відстеження прогресу на основі зібраних даних. У цій діаграмі кожна взаємодія відображає конкретний сценарій використання, починаючи з реєстрації користувачів та закінчуючи генерацією звітів про успішність команд. Це дозволяє детально проаналізувати функціонал системи, виявити її ключові можливості та визначити взаємозв’язки між її складовими частинами. У додатку Б зображено діагрму варіантів викорситання.

3.2 Розробка клієнтської частини та бізнес логіки системи

Розпочнемо розробку автоматизованої системи формування команд із впровадження процесу реєстрації до системи, що дозволить враховувати аспекти, описані у постановці задач та відображені в попередньо наведених діаграмах. Процес реєстрація забезпечує збирання необхідних даних для формування команд. Саме цей процес визначає якість вихідних даних, що впливають на точність роботи алгоритму автоматизації.

Система розроблена таким чином, щоб надавати адміністраторам можливість ініціювати реєстрацію через запрошення користувачів за допомогою унікальних посилань по вказані мейл адерси. Відкриваючий персональні посилання із електронних поштових скриньок користувачі заповнюють форму, де вони вводять

персональну інформацію, навички, досвід і переваги, зацівкавленість в конктерних дисциплінах та освітніх напрямках. Ці дані зберігаються на сервері та стають основою для подальшого формування команд. На рис. 3.3 зображено інтрефейс, який здійснюється надсилання переснальних запрошень:

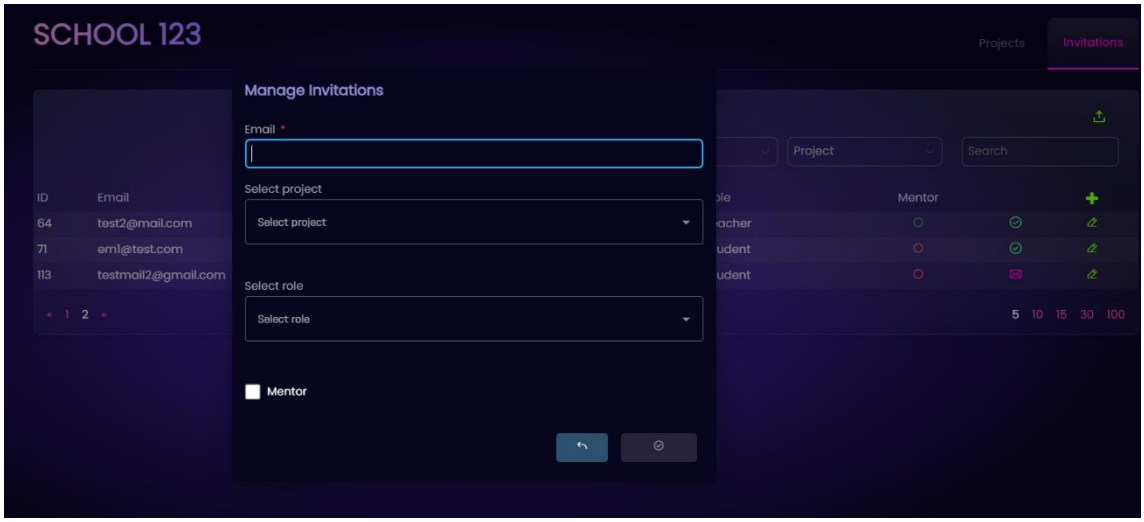


Рисунок 3.3 — створення реєстраційного запрошення

Реєстраційна форма складається із декількох етапів, у кожному із яких користувачів вводить дані. На кожному зних присутні поля введння, які мають прописану валідацію відповідно до потреби. На рис. 3.4 зображено екран введення паролю:

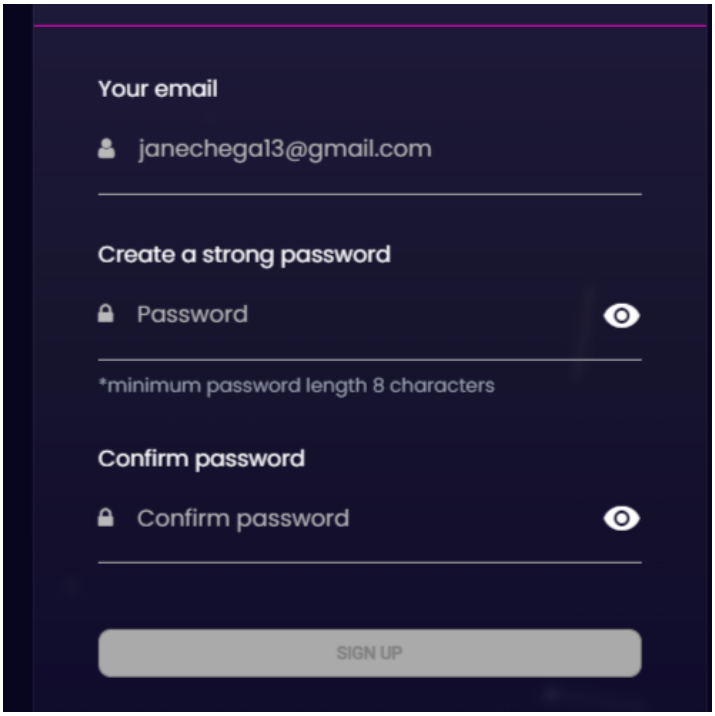


Рисунок 3.4 — процес введення паролю

Одним із етапів реєстраціної форми є оцінування зацікавленості користувачем у конктених дисциплінах, технічні та гуманітарні науки, шкала від 1 до 5. На рис. 3.5 зображено інтрефейс оцінювання зацікавленості в дисципліні:

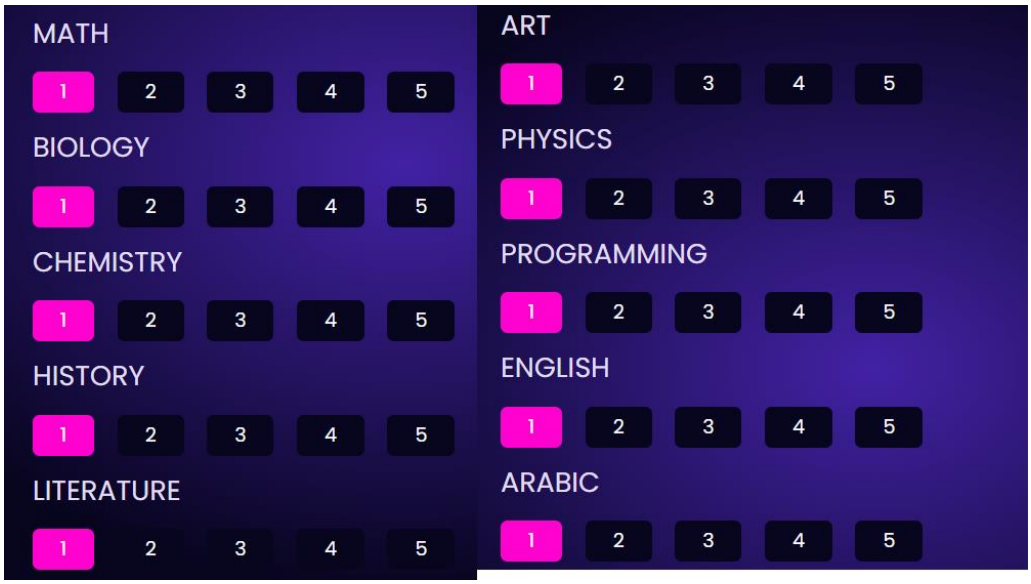


Рисунок 3.5 — оцінювання зацікавленості в дисциплінах

Важливо також оцінити загальні інтереси користувача, його хобі, ступінь зацікавленості в розробці комп'ютерних іграх та бажана позиція в цьому процесі. На рис. 3.6 зображено оцінку інтересів:

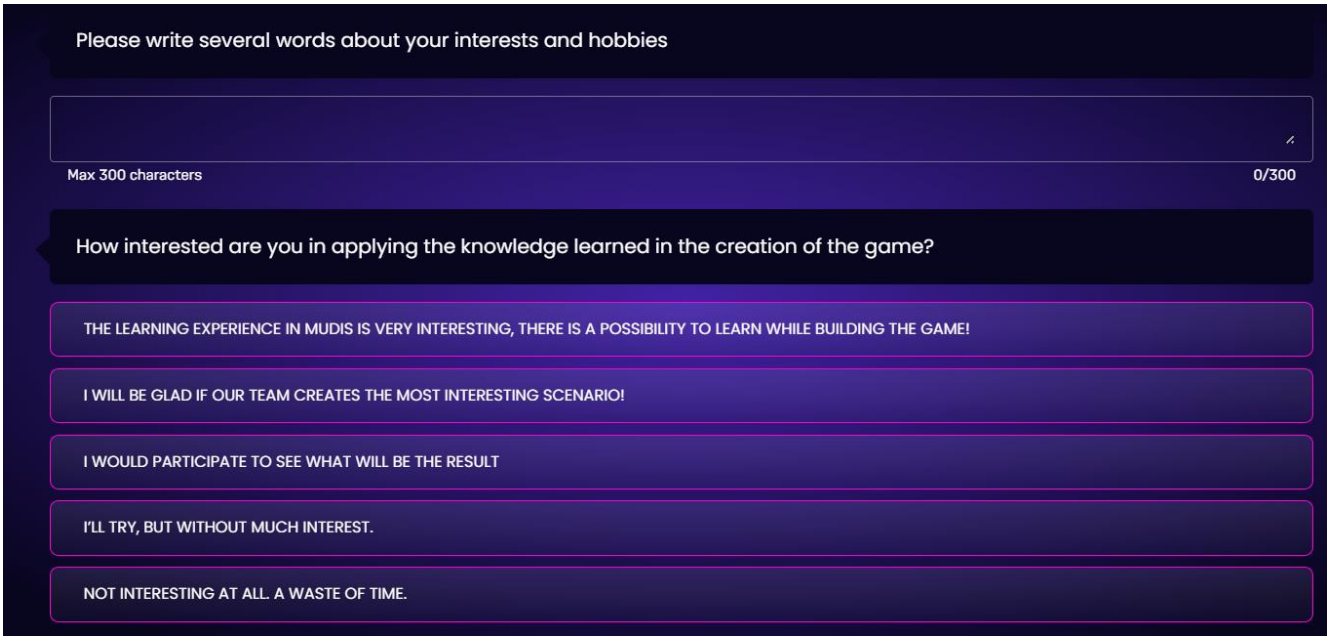


Рисунок 3.6 — оцінка інтересів

Процес автоматичного формування команд учнів передбачає наявність розпочатого проекту до якого й потрібні будуть команди. Команди учнів можуть буду сформані лише тоді, коли набралася необіхідна кількість учнів для виконання даного проекту. На рис. 3.7 зображено процес генерування команд для конкретного проекту:

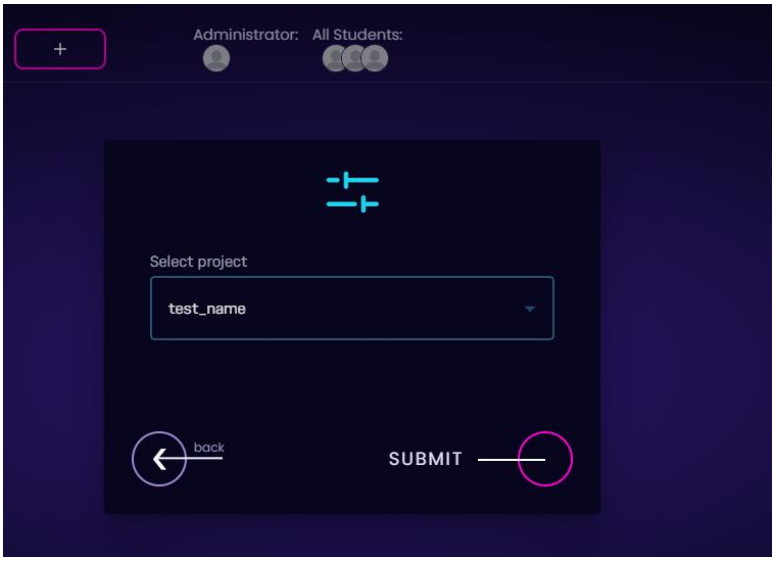


Рисунок 3.7 —генерування команд учнів

Важливою складовою командної робити призначення конкретних тасків конкретним виконавцям. На рис. 3.8 зображено інтерфейс призначення таску до конкретного виконавця:

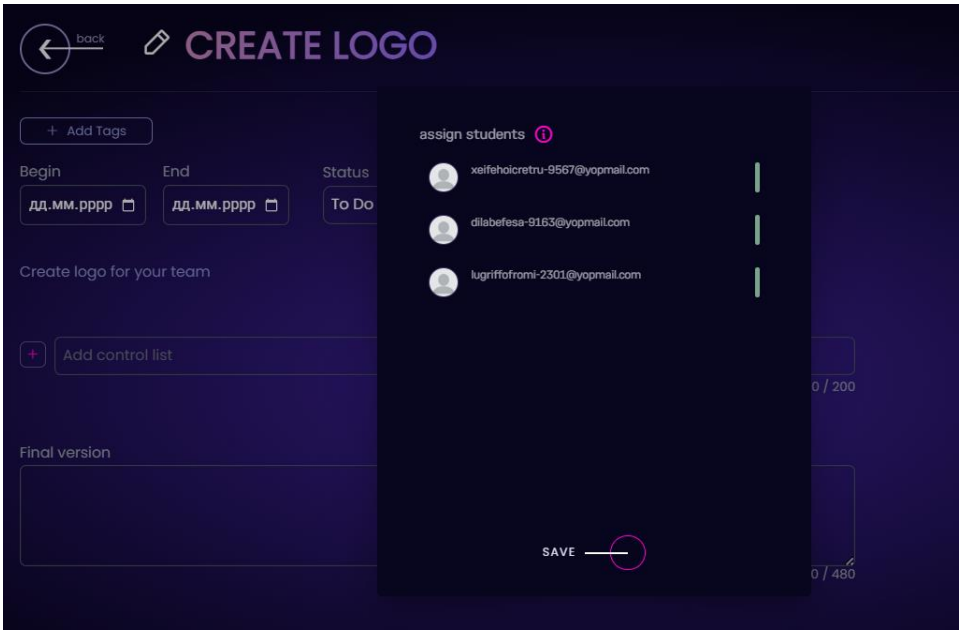


Рисунок 3.8 — інтерфейс призначення таску

Також реалізовано інтерфейс, який дозволяє спостерігати за прогресом виконання завдань, ступенем завершеності проекту команди. На рис. 3.9 зображено статиска виконання завдань:

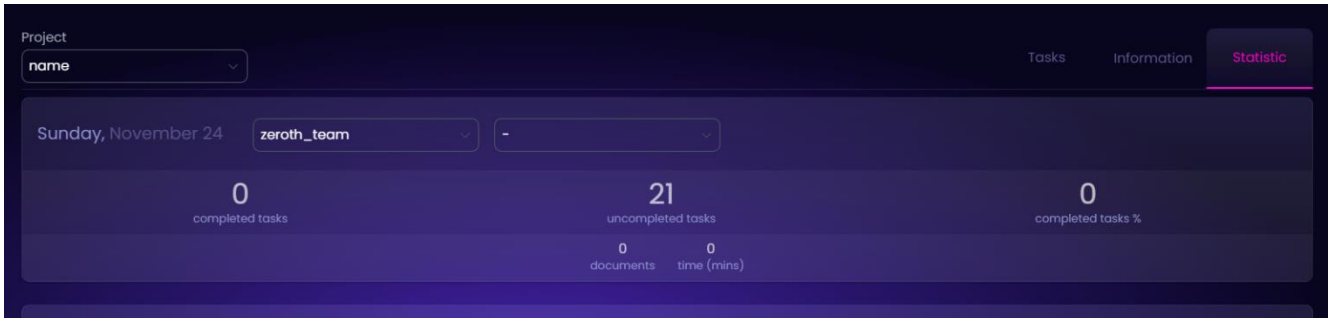


Рисунок 3.9 — статистика виконання завдань

Розроблено модулі які дозволяються переглядати загальний список учнів, якщо персональні дані та закріплені за ними проекти. На рис. 3.10 зображенно можливість редагування даних користувача:

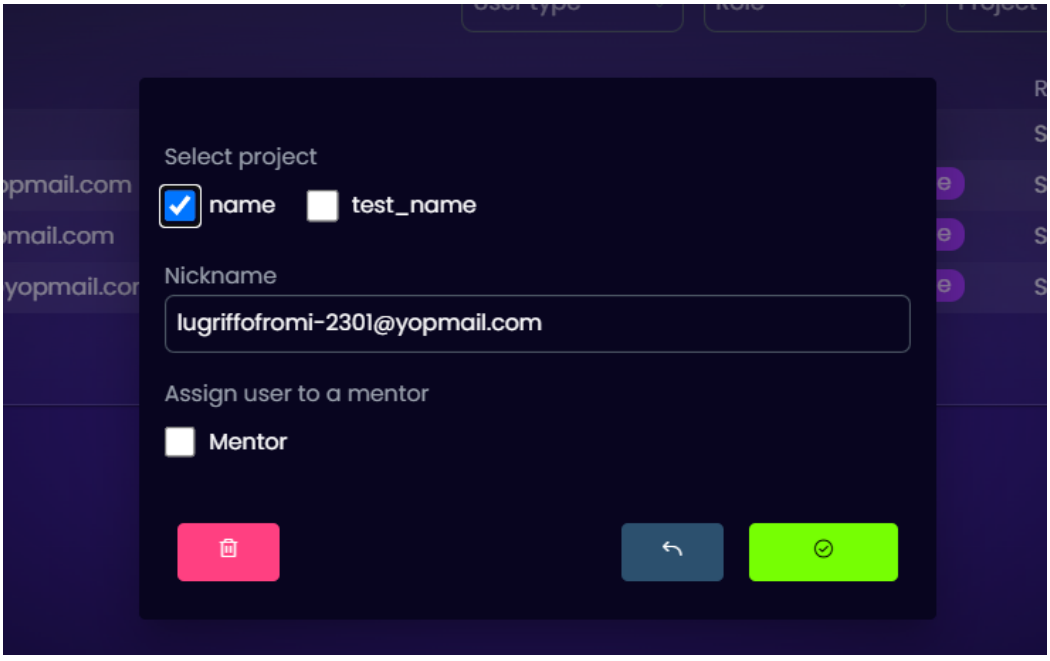


Рисунок 3.10 — редагування даних користувача

Розроблено системі модулі які дозволяють вносити зміни в персональний акаунт, такі як аватар, пароль нікнейм. Також розробленою системою передбачено, що користувач може повторно заповнити дані пов’язані із оцінюванням

зацікавленості в дисциплінах та напрямках. На рис. 3.11 зображено сторінку персонаольної сторінки користувача:

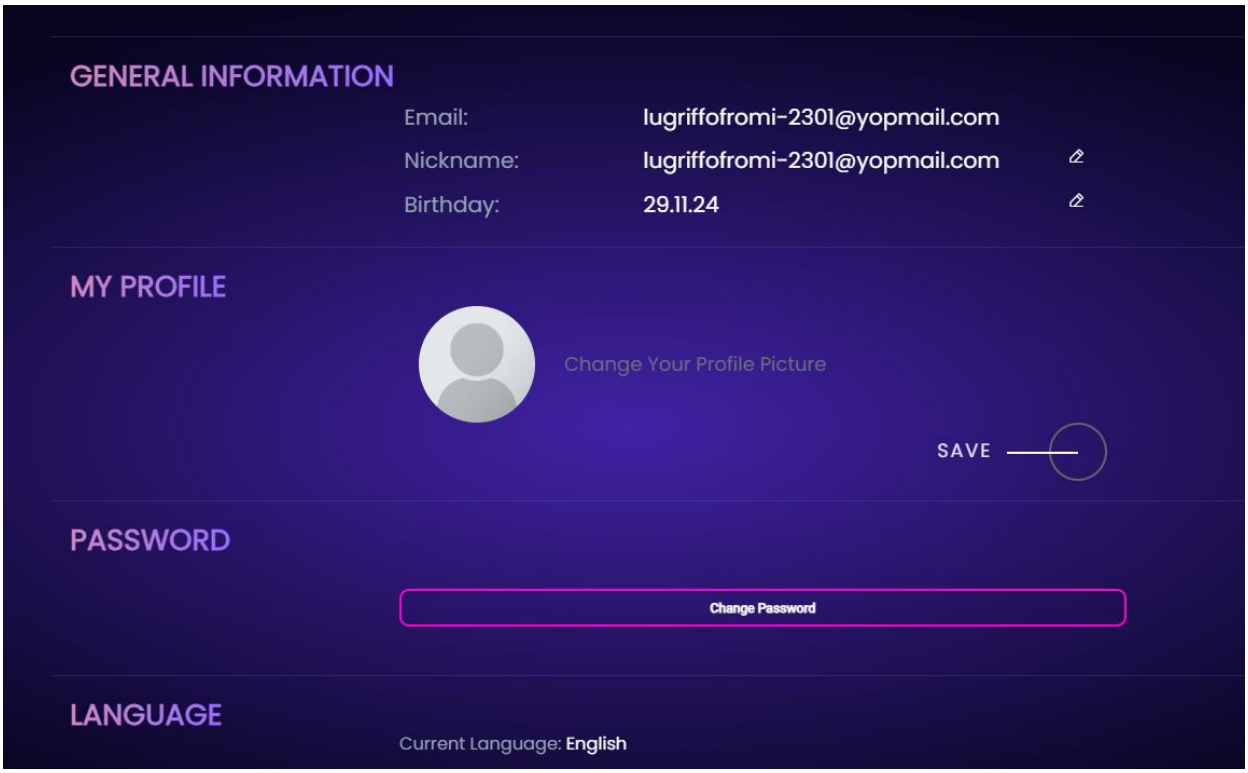


Рисунок 3.11 — персональна сторінка користувача

Також було для користувача відкрито доступ до таблиці завдання команди, із якою він може взаємодіяти вході виконання конкретного проекту. На рис. 3.12 зображено таблицю завдань команди:

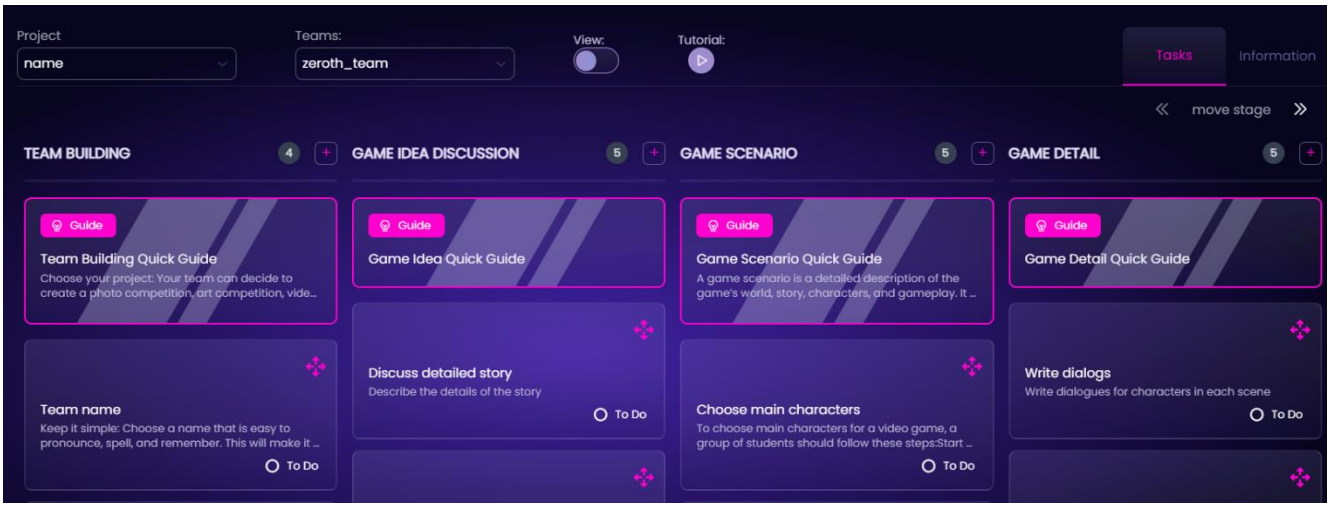


Рисунок 3.12 — таблиця завдань

3.3 Тестування розробленого алгоретмічного та програмного забезпечення

Для забезпечення високої якості автоматизованої системи формування команд тестування було інтегровано на всіх етапах розробки, з особливим акцентом на проєктний підхід. Це дозволило врахувати особливості архітектури та функціональних компонентів системи, оптимізувати процес перевірки та гарантувати її відповідність поставленим вимогам. Основними напрямками для тестування стали інтеграція методики TDD (Test-Driven Development), функціональне та UX-тестування.

Методика TDD (Test-Driven Development) була впроваджена як системний підхід для забезпечення надійності та високої якості програмного коду системи. Основна ідея полягає в розробці в тому заздалегідь створених тестів, які визначали очікувану поведінку системи. Тому було прийнято рішення по створення тест дизайну, який буду вміщувати всі можливі варіації тестових сценаріїв та безпосередньо самі тест кейси. На рис. 3.13 зображено тест дизайн для розділу Account, який узагальнює тестові сценарії пов’язані із відповідними діями із акаунтом:

A	B	C	D
Case ID	**Use case**	**Status**	**Comment**
	Account		
A1	Logout from profile	Pass	
A2	Language toggle	Pass	
A3	Account settings	Pass	
A4	Change nickname	Pass	
A5	Validation "Nickname" field	Pass	
A6	Change birthday date	Pass	
A7	Validation "Birthday" field	Pass	
A8	Change avatar	Pass	
A9	Change password	Pass	
A10	Validation "Password" fields	Pass	
A11	Current Language	Pass	
A12	Profile menu	Pass	
A13	Edit questionnaire form	Pass	
A14	Zoom map with ratings	Pass	

Рисунок 3.13 — тест дизайн для розділу Account

Важливо, щоб тестові сценарії чітко й лаконічно описували конкрентний функціонал системи й описували порядок дій по його виконанню. На рис. 3.14 зображено тест кейси розділу Account:

Use case	Description	Expected Result	Pass/Fail
		You need to log in before performing a test cases	
A1 - Logout from profile	1. Click on username dropdown button	After pressing on button, should be a dropdown list with next buttons: - logout; - language toggle; - edit; - profile.	Pass
	2. Click on "logout" button	After clicking on this button, the "Sign in" page should open. The page should have next elements: - "E-mail" field; - "Password" field; - "Sign In" button; - "Remember box" check box; - "Forgot password" button; - support inscription; - support e-mail.	Pass
A2 - Language toggle	1. Click on username dropdown button	After pressing on button, should be a dropdown list with next buttons: - logout; - language toggle; - edit; - profile.	Pass
	2. Click on language toggle button	After pressing the button: - the language should change (from English to Hebrew); - the toggle should turn green; - alignment should be occur depending on the language.	Pass
A3 - Account settings	1. Click on username dropdown button	After pressing on button, should be a dropdown list with next buttons: - logout; - language toggle; - edit; - profile.	Pass
	2. Click on "Edit" button	After pressing on button, should be next sections: 1. General Information: - email field; - nickname field; - birthday field. 2. My profile: - upload avatar button; - save button. 3. Password: - old password field; - new password field; - change password button. 4. Language: - current language status.	Pass

Рисунок 3.14 — тест кейси розділу Account

Також були розроблені тестові сценарії, які описують негативні сценарії тестування, за яких описується ситуації, де користувач вводить невідповідні дані, та невідповідному форматі. На рис. 3.15 зображено негативний сценарій тестування:

		The "Updated successfully" toaster should also appear.		
A5 - Validation "Nickname" field	1. Click on username dropdown button	After pressing on button, should be a dropdown list with next buttons: - logout; - language toggle; - edit; - profile.	Pass	
	2. Click on "Edit" button	After pressing on button, should be next sections: 1. General Information: - email field; - nickname field; - birthday field. 2. My profile: - upload avatar button; - save button. 3. Password: - old password field; - new password field; - change password button. 4. Language: - current language status.	Pass	
	3. Click "Nickname" field	After pressing on button, should be open modal window with next elements: - nickname field; - cancel button; - submit button.	Pass	
	4. Fill the "Nickname" field	Leave field empty.	Pass	
	5. Click on "Submit" button	After pressing on button, changes must be applied. The "Updated successfully" toaster should also appear.	Fail	The "Submit" button should not be clickable.
	6. Fill the "Nickname" field	Fill the field with 32+ characters.	Pass	
	7. Click on "Submit" button	After pressing on button, changes must be applied. The "Updated successfully" toaster should also appear.	Fail	Should be an inscription with the maximum number of characters in the nickname.

Рисунок 3.15 — негативний сценарій тестування

Важливо описати тестові сценраії для головного функціоналу системи, автоматичного формування команд. На рис. 3.16 зображенові тестові сценарії генерування команд:

C58 - Automatic team generation	1. Click on "Communications menu" button	After clicking on this button, the system should display the "Communications" page. This page should contain elements: - "Name of project" inscription; - scrole line; - "Teams" dropdown list; - "View" radio button; - "Tasks/Informarion page" button; - "Tutorial" button; - "Uni chat" button.	Pass
	2. Click on the "Tasks/Informarion page" button	After clicking on this button, the system should display the "Informayion" page. This page should contain elements: - "Generate teams" button;	Pass
	3. Click on the "Generate teams" button	After clicking on the "Generate teams" button, the command generation window should appear. It should contain the following elements: - "Select prodject" drop down list; - "Submit" button; The created command should be displayed correctly. The drag and drop page should contain the tasks for the group.	Pass
	4. Automatic team generation	A team should be generated for every 3 students, and each group should have a leader.	Pass

Рисунок 3.16 — тестовий сценерій генерування команд

Після генерування команди, для користувачів системи також повинні відобразитися базові таскі, які передбачені для будь якої команди. На рис. 3.17 зображено тестовий сценарій сторінки тасків:

C1 - Task menu	1. Click on "Communications menu" button	After clicking on this button, the system should display the "Communications" page. This page should contain elements: - "Name of project" inscription; - scrole line; - "Teams" dropdown list; - "View" radio button; - "Tasks/Informarion page" button; - "Move stage" button; - "Tutorial" button; - "Team building" tasks list; - "Game IDEA Discussion" tasks list; - "Game Scenario" tasks list; - "Game Detail" tasks list; - "Game Presentation" tasks list; - "Add a Team building" button; - "Add a Game IDEA Discussion"button; - "Add a Game Scenario" button; - "Add aGame Detail" button; - "Add a Game Presentation" button; - "Task" kebab menu; - "Uni chat" button.	Pass
	2. Interaction with "Tasks menu"	By default, the user is placed on the "Tasks" page. If necessary, you can change the display by clicking on the "Tasks/Informarion page" button.	Pass

Рисунок 3.17 — тестовий сценарій сторінки тасків

Функціональне тестування автоматизованої системи формування команд зосереджено на перевірці роботи її основних функцій у реальних умовах використання. Метою було гарантувати, що кожна частина системи виконує заявлені завдання відповідно до технічних вимог та очікувань користувачів.

Було проведено функціональну перевірку процесу реєстрації користувача, починаючий із моменту надсилання запрошення, закінчуючи процесом завершення реєстрації користувачем. Під час цієї перевірки було переглянуто коректність роботи валідації під надсилання реєстраційного запрошення. На рис. 3.18 зображено валідація користувацього запрошення:

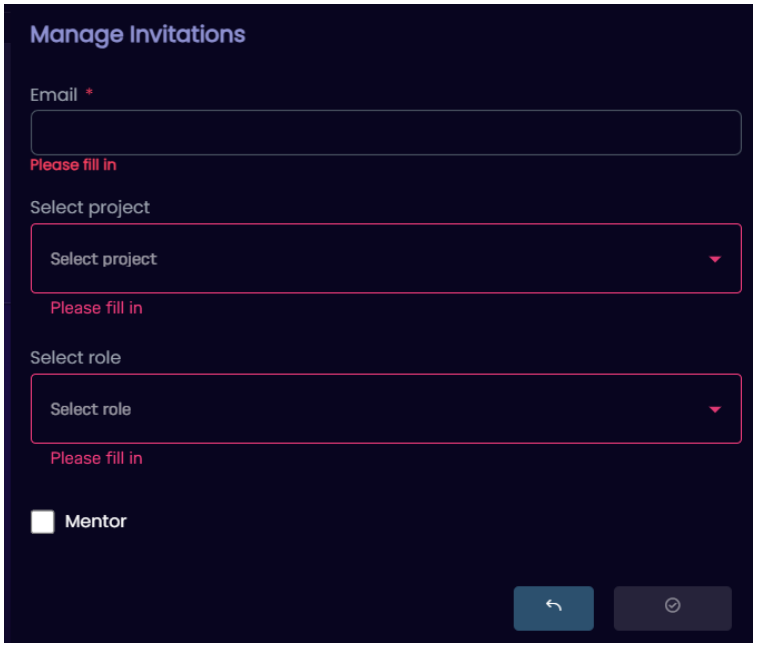


Рисунок 3.18 — валідація користувацього запрошення

Заповнивши вище зазначені поля, система повідомить нас про успішне надсилання запрошення. На рис. 3.19 зображено успішне запрошення:

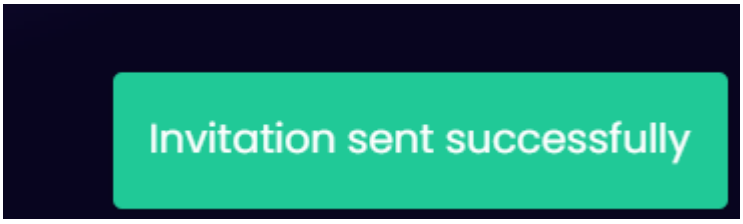


Рисунок 3.19 — успішне запрошення

Також важливо переконатися що запрошення було надіслано на вказану електронну пошту користувача. На рис. 3.20 зображено поштову скриньку із запрошенням:

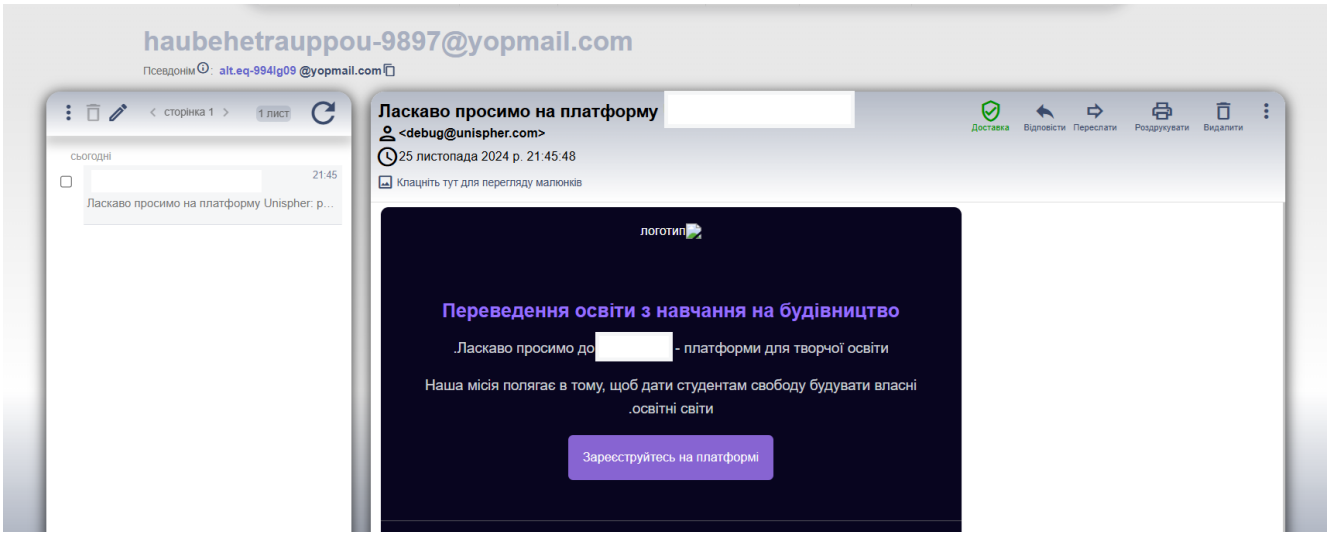
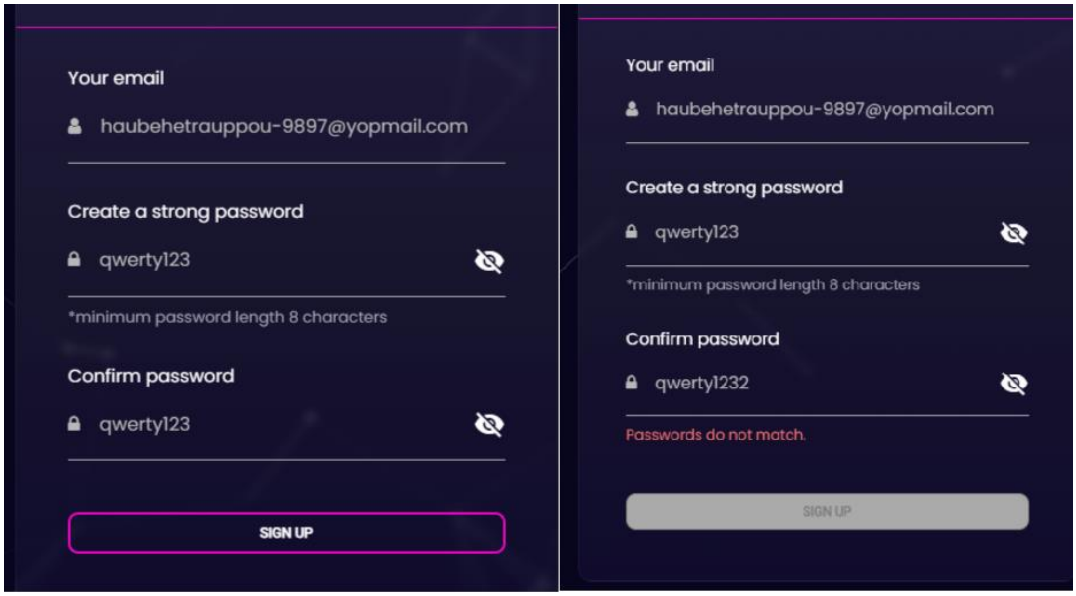


Рисунок 3.20 — поштова скринька із запрошенням

При переході за посиланням користувач створює пароль до свого персонального акаунту. Важливо щоб пароль відповідам вимогам безпеки та був повторно правильно прописаний в полі «Confirm password». На рис. 3.21 зображено створення персонального паролю:



Ринок 3.21 — створення персонального паролю

Важливо також, щоб відновлення працювало коректно, система повинна відхиляти неіснуючі паролі в базі даних, й надсилати листи лише на існуючі електрону пошту. На рис. 3.22 зображено відновлення пароля:

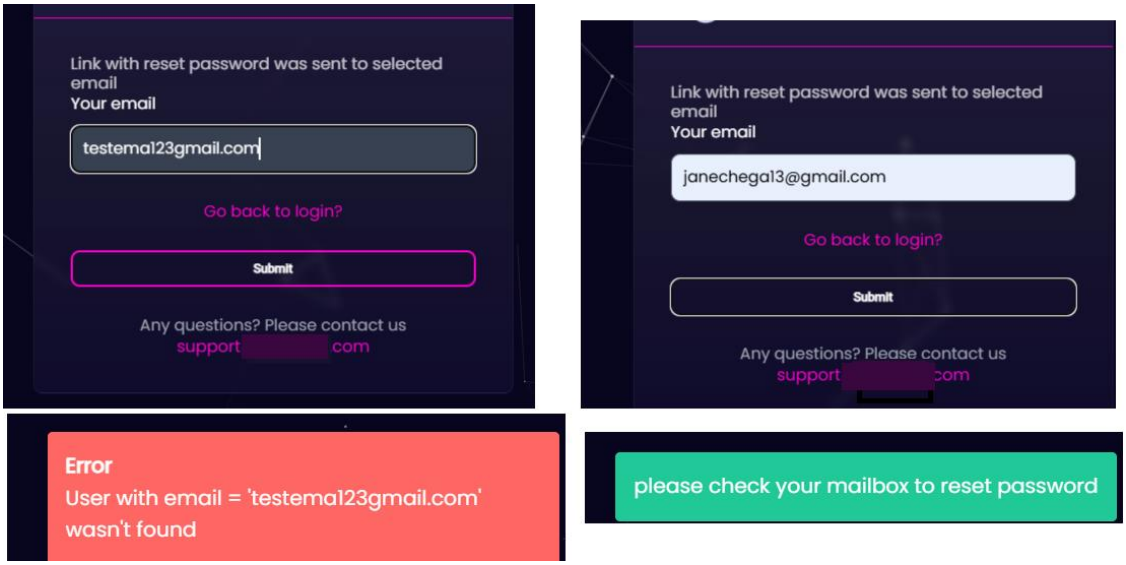


Рисунок 3.22 — відновлення пароля

При редагуванні доданої інформації такої як нікнейм важливо бути впевненому в тому, що поля буду заповненні, введенна інформація буде коректно відображатися на не накладатися на інші поля. На рис. 3.23 зображено зміну нікнейму користувача:

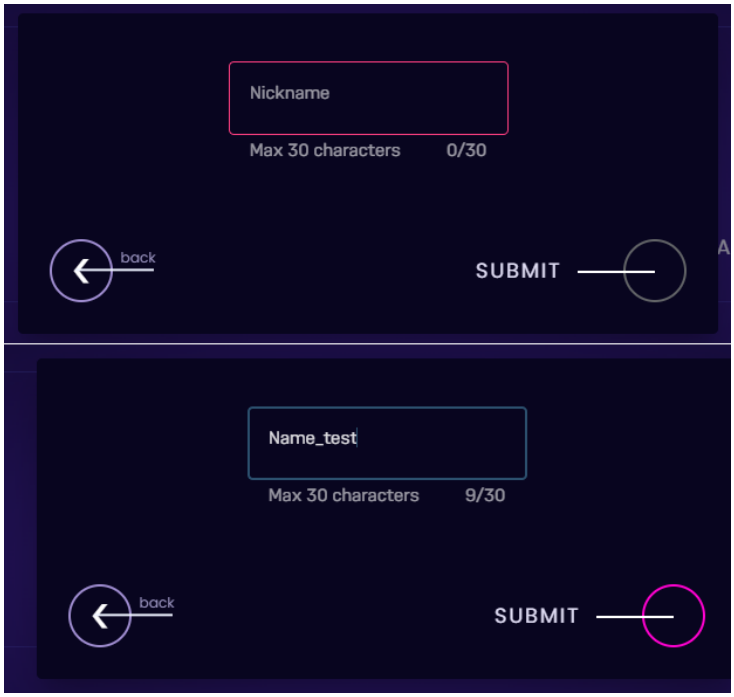


Рисунок 3.23 — зміна нікнейму

Також було проведено перевірку автоматичного розподіл користувачі по командам для заданого проекту. В ході цієї перівірки було визначено що автоматичний розподіл працює коректно. На рис. 3.24 зображено користувачів які було розподілено до конкретного проекту:

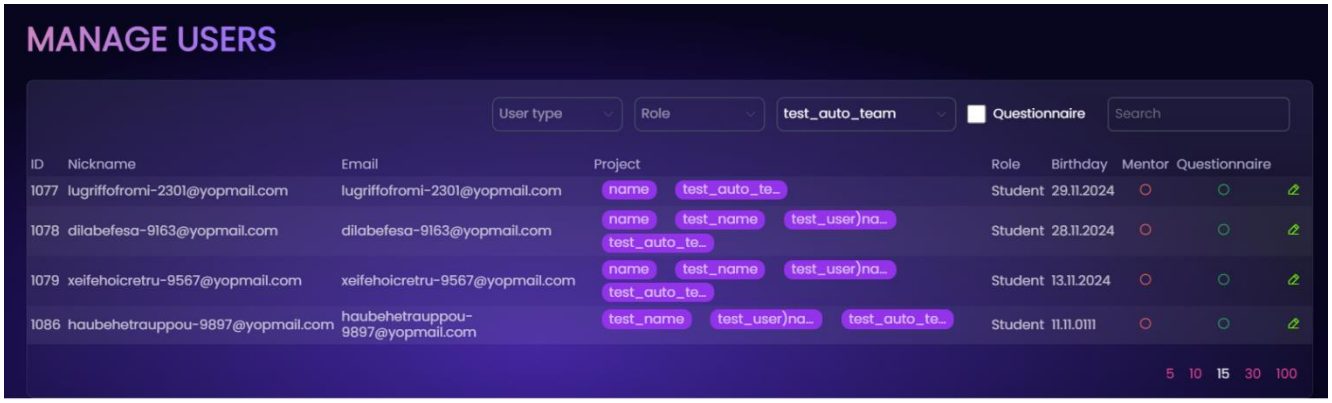


Рисунок 3.24 — розподілені користувачі

Перевірено коретність додавання нового проекту до системи, перевіріно роботу полів введення на наявність введного тексту та формату данів. На рис. 3.25 зображено валідація полів додавання проєкту:

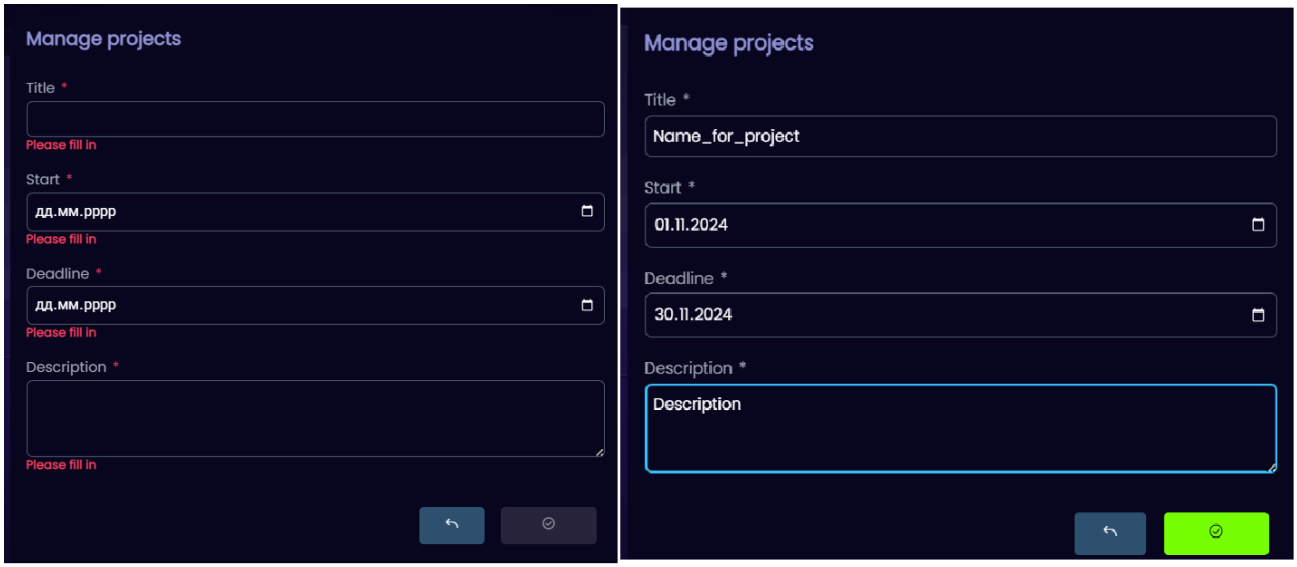


Рисунок 3.25 — валідація полів додавання проєкту

Додавання завдань до проекту є частиною робити команди та її членів, даний функціонал було перевірено на коректність заповнення його заголовку та прикріплених до нього тегів. На рис. 3.26 зображено процес додавання таску:

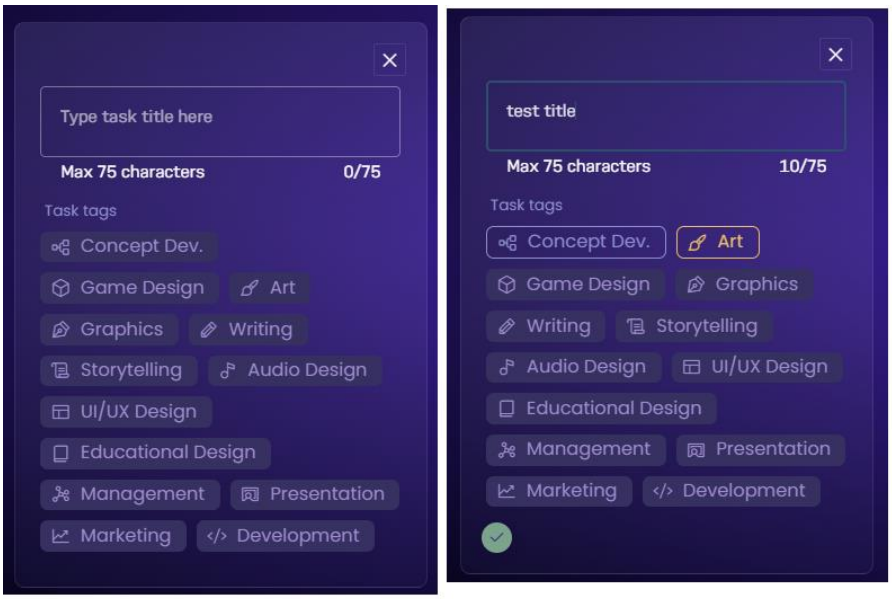


Рисунок 3.26 — процес додавання таску

Також важливо редагувати вже інсуючі завдання, змінювати час виконання статуси, додавати додатковий опис. Даний функціонал було перевірено на доступність внесення та збереження змін. На рис. 3.27 зображено редагування таску:

Рисунок 3.27 — редагування таску

Сторінка із статискою відображує загальну ситуацію по проекту, тому бо перевірено коректність робити відображення статистики. На рис. 3.28 зображено статистику по проєкту:

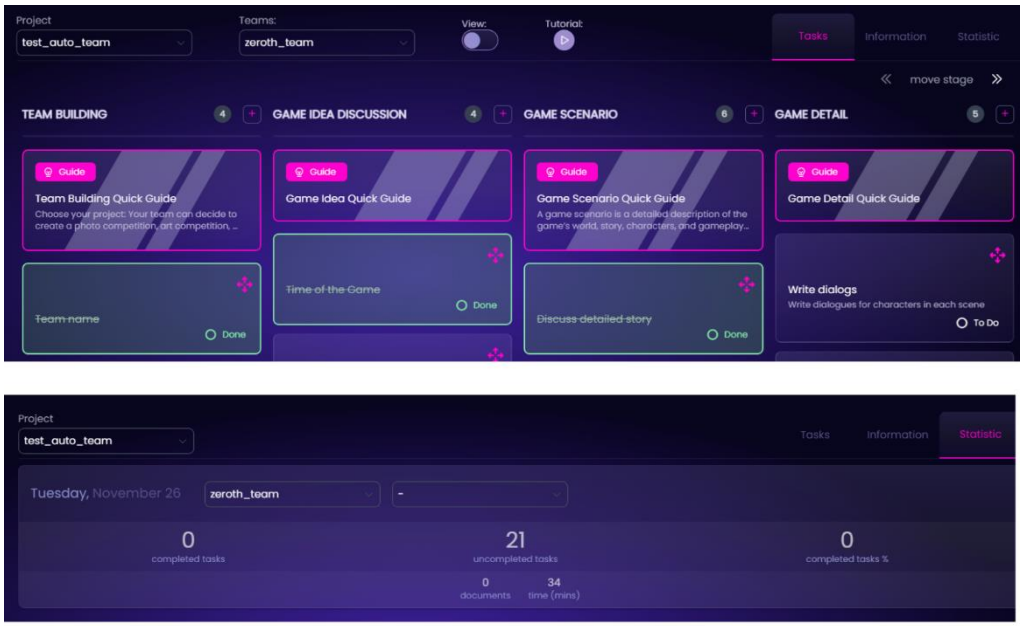


Рисунок 3.28 — статистику по проєкту

Було проведено перевірку на зручність інтрефейсу на різних дивайсах, адже важливо щоб системо була зручною для взаємодії на різних типах пристороїв. На рис. 3.29 зображено мобільний інтрерфейс логін сторінки:

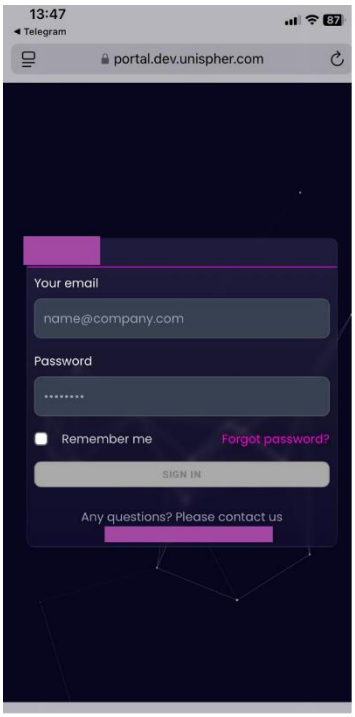


Рисунок 3.29 —мобільний інтрерфейс логін сторінки

Важлива також перевірити роботи системи в різних браузерах, щоб надати можливість користувач працювати зі системою прицьому не змінючи при цьому звичний браузер. На рис. 3.30 зображено роботу системи в різних браузерах:

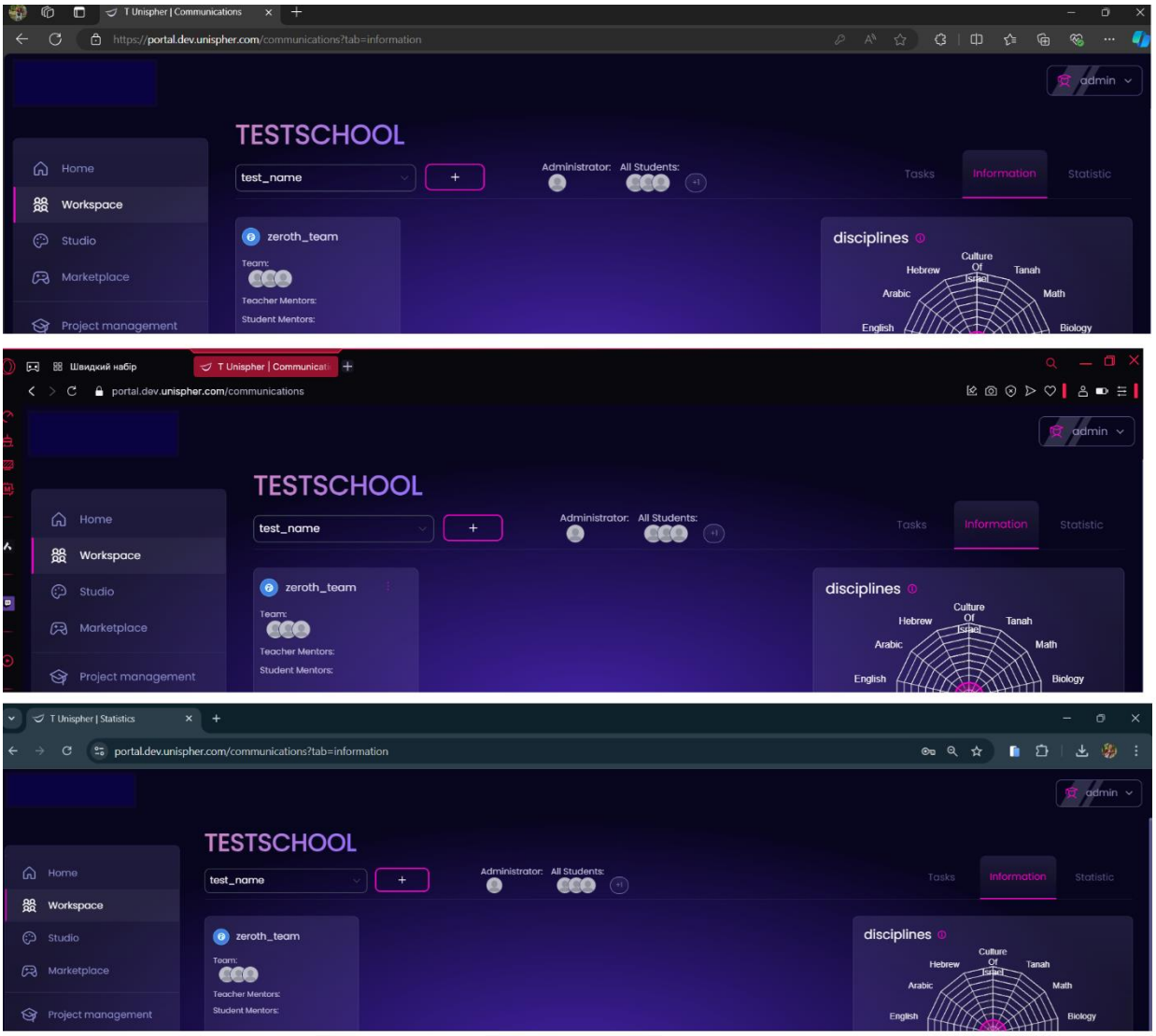


Рисунок 3.30 — робота системи в різних браузерах

Продемонстровані вище підходи до тестування автоматизованої системи формування команд учнів дозволяє забезпечити комплексну перевірку її функціональності, зручності використання та надійності. Інтеграція методології TDD на ранніх етапах розробки дозволила уникнути значної кількості помилок, завдяки чіткій формалізації очікуваної поведінки системи.

Функціональне тестування охопило аспекти роботи системи, зокрема реєстрацію, управління акаунтами, формування команд і взаємодії з проєктами.

Перевірка коректності обробки даних, таких як валідація полів, додавання та редагування завдань, продемонструвала високу відповідність системи технічним вимогам. Також було розглянуто негативні сценарії тестування, які виявили потенційні уразливості, пов'язані з некоректними діями користувачів.

Окремо варто відзначити тестування інтерфейсу, яке охопило перевірку адаптивності дизайну на різних пристроях та коректності відображення елементів у різних браузерах. Це є критично важливим для забезпечення доступності системи широкому колу користувачів. Результати тестування підтвердили, що система функціонує стабільно на мобільних пристроях і різних веб-браузерах, що є вагомою перевагою.

Тестування автоматичного розподілу користувачів по командах виявило відповідність функціоналу заявленим вимогам, що свідчить про готовність системи до інтеграції та експлуатації. Крім того, перевірка сторінок статистики підтвердила їхню інформативність та коректність роботи, що сприяє підвищенню ефективності управління проєктами. Загалом, проведене тестування дозволило виявити та усунути недоліки системи ще до її впровадження, забезпечивши високий рівень якості продукту. Комплексний підхід до перевірки функціоналу, стабільності та зручності використання гарантує, що система відповідає потребам кінцевих користувачів та є готовою до використання в реальних умовах.

3.4 Висновки до розділу

У даному розділі було описано процес розробки клієнтської частини автоматизованої системи формування команд учнів, яка повністю відповідає поставленим задачам та реалізує функціональність, спрямовану на підвищення ефективності командної роботи та організації освітніх проектів.

На етапах розробки клієнтської частини та бізнес-логіки було створено інтуїтивно зрозумілий інтерфейс, функціональні модулі та алгоритми, що забезпечують високий рівень автоматизації й точності в управлінні командами.

Реалізовано механізм реєстрації, який базується на запрошеннях через унікальні посилання, що надсилаються електронною поштою, та багатоступеневу форму реєстрації з валідацією даних на кожному етапі. Це гарантує, що користувачі вводять коректну інформацію, необхідну для формування команд. Особливу увагу приділено збору інформації про навички, досвід, зацікавленість у дисциплінах, хобі та бажаних ролях, що значно підвищує ефективність подальшого автоматичного формування команд.

Впровадження методології TDD (Test-Driven Development) дозволило створити надійний програмний код, перевірений на всіх етапах розробки, а функціональне тестування підтвердило коректність роботи ключових модулів, зокрема процесів реєстрації, формування команд, додавання проектів і завдань. UX-тестування забезпечило зручність і логічність взаємодії користувачів із системою, що дозволило створити максимально комфортне середовище для роботи. Інтерфейс редагування даних учнів і проектів забезпечує швидке внесення змін адміністраторами, а учасники отримують актуальну інформацію про свою команду та проект.

Розроблена система демонструє високий рівень функціональності, інтуїтивність у використанні та гнучкість в управлінні навчальними проектами. Її модульна структура дозволяє легко масштабувати систему під більші обсяги користувачів або інтегрувати нові функції. Розроблена автоматизована система формування команд учнів є ефективним рішенням для освітніх платформ, що прагнуть підвищити якість командної роботи, оптимізувати процеси управління проектами та створити комфортне середовище для користувачів.

ВИСНОВКИ

У ході виконання магістреської кваліфікаційної роботи було проведено дослідницьку роботу за темою автоматизації процесу формування команд учнів для онлайн навчання. Під час дослідницької роботи були розглянуті особливості дистанційної освіти, її викликів та можливості для покращення. Була визначено, що інтеграція автоматизованого формуваннях команд до системи дозволяє підвищити ефективність організацію навчального процесу, оскільки сприяє не лише розподілу учнів, але й підвищенню їхньої мотивації, залученості та розвитку соціальних навичок.

Також під час дослідницької роботи було проведено аналіз популярних освітніх платформ, таких як Moodle, Google Classroom, Blackboard і Canvas. Виявлено, що кожна з них має свої переваги та недоліки, але більшість не враховують комплексного підходу до автоматизації формування команд з урахуванням індивідуальних характеристик учнів. Дослідження також охоплювало сучасні підходи до формування команд, які варіюються від випадкового розподілу до використання алгоритмів на основі інтересів, навичок, академічних показників і психологічної сумісності.

На основі отриманих результатів дослідження було сформовано концепцію автоматизованої системи формування команд, визначено її функціональні вимоги та обґрунтовано вибір клієнт-серверної архітектури. Було встановлено, що така архітектура забезпечує масштабованість, зручність внесення змін до бізнес-логіки, централізовану безпеку даних і ефективний розподіл ресурсів між клієнтом та сервером. Технологічний стек розробки включає HTML, CSS, Vue.js для клієнтської частини та REST API для взаємодії з серверною частиною.

Було проведено розробку клієнтьскої частини автоматизованої системи формування команд, функціональних модулів для управління проектами та засобів для збору й аналізу даних. Зокрема, реалізація включатиме багатоступеневу систему реєстрації, яка враховує навички, досвід і ролі учнів, а також модуль моніторингу прогресу команд у реальному часі.

Під час розробки використовуалася методологія TDD (Test-Driven Development), що дозволило забезпечити створення надійного програмного коду, перевіреного на всіх етапах реалізації. Функціональне тестування підтвердило коректну роботу ключових модулів, зокрема алгоритмів формування команд, модулів реєстрації та управління проектами. UX-тестування дозволило оцінити зручність інтерфейсу для різних категорій користувачів, забезпечивши комфортне середовище для взаємодії з системою.

Таким чином, практична реалізація програмного забезпечення буде спрямована на створення інноваційного освітнього інструмента, який підвищить якість онлайн-навчання, автоматизуючи складні процеси формування та управління командами. Розроблена система стане ефективним рішенням для освітніх платформ, орієнтованих на інтеграцію командної роботи, полегшення організації навчальних проектів та підтримку сучасних методів взаємодії учнів і викладачів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Дубовой В. М., Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спеціальності 174 «Автоматизація, комп’ютерно- інтегровані технології та робототехніка» (кафедра комп'ютерних систем управління) [Електронний ресурс] / Вінниця : ВНТУ, 2023 – (PDF, 45 с.);
2. Коцюбинський В.Ю. Управління ІТ-проектами та проектуванням інформаційних систем: навчальний посібник / [Електронний ресурс]. Режим доступу до ресурсу: http://surl.li/tyfhj
3. Алхімова С. М. Об’єктно-орієнований підхід до розробки програмного забезпечення/ Київський політехнічний інститут імені Ігоря Сікорського, 2019. 194с.
4. Олена Амеліна, Олександр Цуркан. Дистанційне та змішане навчання. Досвід, поради, інструменти / Київ «Основа», 2022 – 128с.
5. STEM-освіти в умовах інтеграції формальної і неформальної освіти обдарованих учнів: методичні рекомендації / Поліхун Н. І. , К. Г. Постова, Сліпухіна І. А. , Онопченко Г. В. , Онопченко О. В – Київ : Інститут обдарованої дитини НАПН України, 2019. – 80 с.
6. Програма автоматизації навчального центру [Електронний ресурс] – Режим доступу: https://evergreens.com.ua/ua/articles/education-system.html
7. How technology is shaping learning in higher education [Електронний ресурс] – Режим доступу: https://www.mckinsey.com/industries/education/our-insights/how-technology-is-shaping-learning-in-higher-education
8. Шквір В. Д. Інформаційні системи і технології в обліку: Практикум : навчальний посібник для студентів вищих навчальних закладів. Київ : Знання, 2006. 429 с.
9. Saddleback Educational Saddleback Educational Publishing. Science and Technology Words. Saddleback Educational Publishing, Incorporated, 2010. 125 с.

10. Learning management system [Електронний ресурс] – Режим доступу: https://www.techtarget.com/searchcio/definition/learning-management-system
11. Використання аналітики в LMS для покращення навчання та оцінки результатів [Електронний ресурс] – Режим доступу: https://skillzrun.com/blog/vikoristannya-analitiki-v-lms-dlya-pokrashhennya-navchannya-ta-oczinki-rezultativ/
12. Машинне навчання: основні поняття та алгоритми [Електронний ресурс] – Режим доступу: https://optima.study/blog/mashinne-navchannya-osnovni-ponyattya-ta-algoritmi-v-data-science
13. Pais M., Skelton M. Team Topologies: Organizing Business and Technology Teams for Fast Flow. IT Revolution Press, 2019.
14. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, Incorporated, 2022.
15. Abbott D. Applied Predictive Analytics: Principles and Techniques for the Professional Data Analyst. Wiley & Sons, Incorporated, John, 2022. 450 с.
16. Moodle – офіційний веб-сайт [Електронний ресурс] – Режим доступу: https://moodle.org/?lang=uk
17. Google Classroom is the main address of Education. [Електронний ресурс] – Режим доступу: https://sites.google.com/view/classroom-workspace/
18. Blackboard - The most modern and innovative LMS available [Електронний ресурс] – Режим доступу: https://www.anthology.com/products/teaching-and-learning/learning-effectiveness/blackboard
19. Canvas LMS [Електронний ресурс] – Режим доступу: https://www.webhostingzone.org/solutions/canvas-lms-hosting.html
20. Маліков, І. С. Клієнтська частина системи забезпечення доступу до сховища документів кафедри : дипломна робота ... бакалавра : 122 Комп’ютерні науки / Маліков Іван Сергійович. – Київ, 2022. – 77 с.
21. Типи архітектур інформаційних систем [Електронний ресурс] – Режим доступу: http://surl.li/nzmeqc

22. Клієнт-Серверної Архітектура [Електронний ресурс] – Режим доступу: https://dou.ua/forums/topic/44636/
23. Методи масштабування реляційних баз даних: переваги, недоліки та кейси використання [Електронний ресурс] – Режим доступу: https://dou.ua/forums/topic/45890/
24. Integrated DEFinition Methods (IDEF) [Електронний ресурс] – Режим доступу: https://cio-wiki.org/wiki/Integrated_DEFinition_Methods_(IDEF)
25. UML для бізнес-моделювання [Електронний ресурс] – Режим доступу: https://evergreens.com.ua/ua/articles/uml-diagrams.html
26. Duckett J. HTML & CSS: Design and build websites. Wiley, 2014. 490 p.
27. Samuel A. M. Hyper Text Markup Language (HTML). Independently Published, 2018.
28. Основи CSS [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/ua/docs/Learn/Getting_started_with_the_web/CSS_basics
29. Vue.js як JavaScript-фреймворк [Електронний ресурс] – Режим доступу: https://foxminded.ua/vue-js/
30. What is API: Definition, Types, Specifications, Documentatio [Електронний ресурс] – Режим доступу: https://www.altexsoft.com/blog/what-is-api-definition-types-specifications-documentation/
31. REST API URL - Best Practices and Examples [Електронний ресурс] – Режим доступу: https://apidog.com/blog/rest-api-url-best-practices-examples
32. Principles of Information Systems. Course Technology, 2016. 752 p.
33. Швець Т. Е. ВАРІАТИВНЕ ВІДКРИТЕ ОСВІТНЄ СЕРЕДОВИЩЕ ЯК УМОВА РЕАЛІЗАЦІЇ ТЬЮТОРИНГУ В ЗАКЛАДІ ЗАГАЛЬНОЇ СЕРЕДНЬОЇ ОСВІТИ. *Pedagogical Sciences Theory and Practice*. 2023. № 4. С. 130–137. URL: https://doi.org/10.26661/2786-5622-2023-4-19
34. Technological Pedagogical Content Knowledge / ed. by C. Angeli, N. Valanides. Boston, MA : Springer US, 2015.

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 2. Клієнтська 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	100%
Загальна схожість	0%

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 _____ Євгеній ЧЕГА

(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 магістреської кваліфікаційної роботи

«Розробка клієнт-серверної системи для автоматизації процесу формування команд учнів для навчання онлайн. Частина 2. Клієнтська частина.»

08-33.МКР.016.03.000 ТЗ

Студент групи ЗАКІТР-23м
_____ Чега Євгеній

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

Вінниця 2024

1. 1. Назва та галузь застосування

   1.1. Назва – Розробка клієнт-серверної системи для автоматизації процесу формування команд учнів для навчання онлайн. Частина 2. Клієнтська 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 магістреської кваліфікаційної роботи є підвищення ефективності онлайн-навчання через автоматизацію процесу формування команд учнів за допомогою клієнт-серверної системи.

4. Джерела розробки.

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

   - Дубовой В. М., Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спеціальності 174 «Автоматизація, комп’ютерно- інтегровані технології та робототехніка» (кафедра комп'ютерних систем управління) [Електронний ресурс] / Вінниця : ВНТУ, 2023 – (PDF, 45 с.);
   - Олена Амеліна, Олександр Цуркан. Дистанційне та змішане навчання. Досвід, поради, інструменти / Київ «Основа», 2022 – 128с.
   - Об’єктно-орієнований підхід до розробки програмного забезпечення/ С. М. Алхімова. – Київський політехнічний інститут імені Ігоря Сікорського, 2019. 194с.

5. Вимоги до розробки.

   5.1. Перелік головних функцій:

      1. - реєстрація аккаунту;
      2. - проходження анкети;
      3. - 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 усіх типах девайсів;
- можливість цілодобового функціонування системи;
- дані оновлюються і є актуальними.

6. Стадії та етапи розробки.

6.1. Поянювальна записка:

1. Аналіз методів, підходів і засобів реалізації задачі автоматизації процесами в об’єкті управління відповідно до теми кваліфікаційної роботи. Постановка задач дослідження «17 » 09 2024 р.
2. Удосконалення технології прийняття рішень при автоматизації об’єкту « 25 » 09 2024 р.
3. Визначення техніних характеристик системи « 19 » 10 2024 р.

7. Порядок контроллю й приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до « 29 » 11 2024 р.

7.2. Атестація проекту здійснюється на попередньому захисті. Попередній захист магістреської дипломної роботи провести до «1 » 12 2024 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК. Захист магістреської дипломної роботи провести до « 12 » 12 2024 р.

ДОДАТОК В

(обов'язковий)

Ілюстративна частина

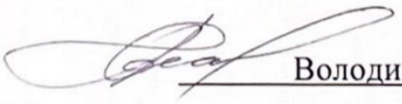
ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА КЛІЄНТ-СЕРВЕРНОЇ СИСТЕМИ ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСУ ФОРМУВАННЯ КОМАНД УЧНІВ ДЛЯ НАВЧАННЯ ОНЛАЙН. ЧАСТИНА 2. КЛІЄНТСЬКА ЧАСТИНА.

Студент групи ЗАКІТР-23м

Євгеній ЧЕГА

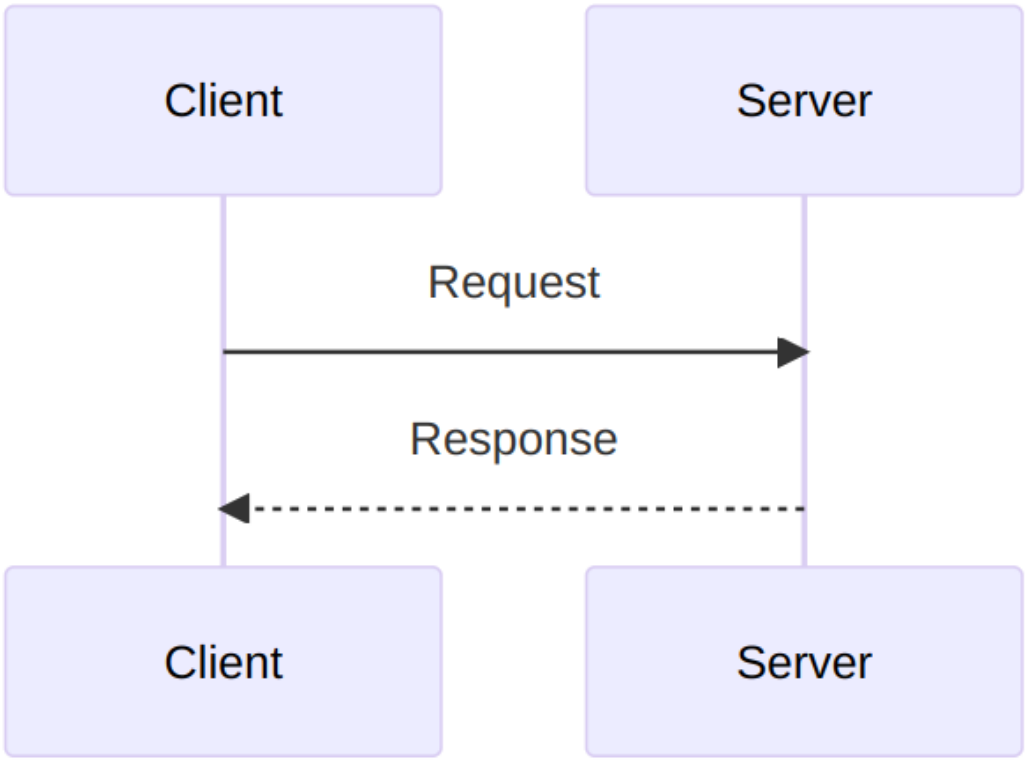
Керівник: к.т.н., доцент каф. АІІТ



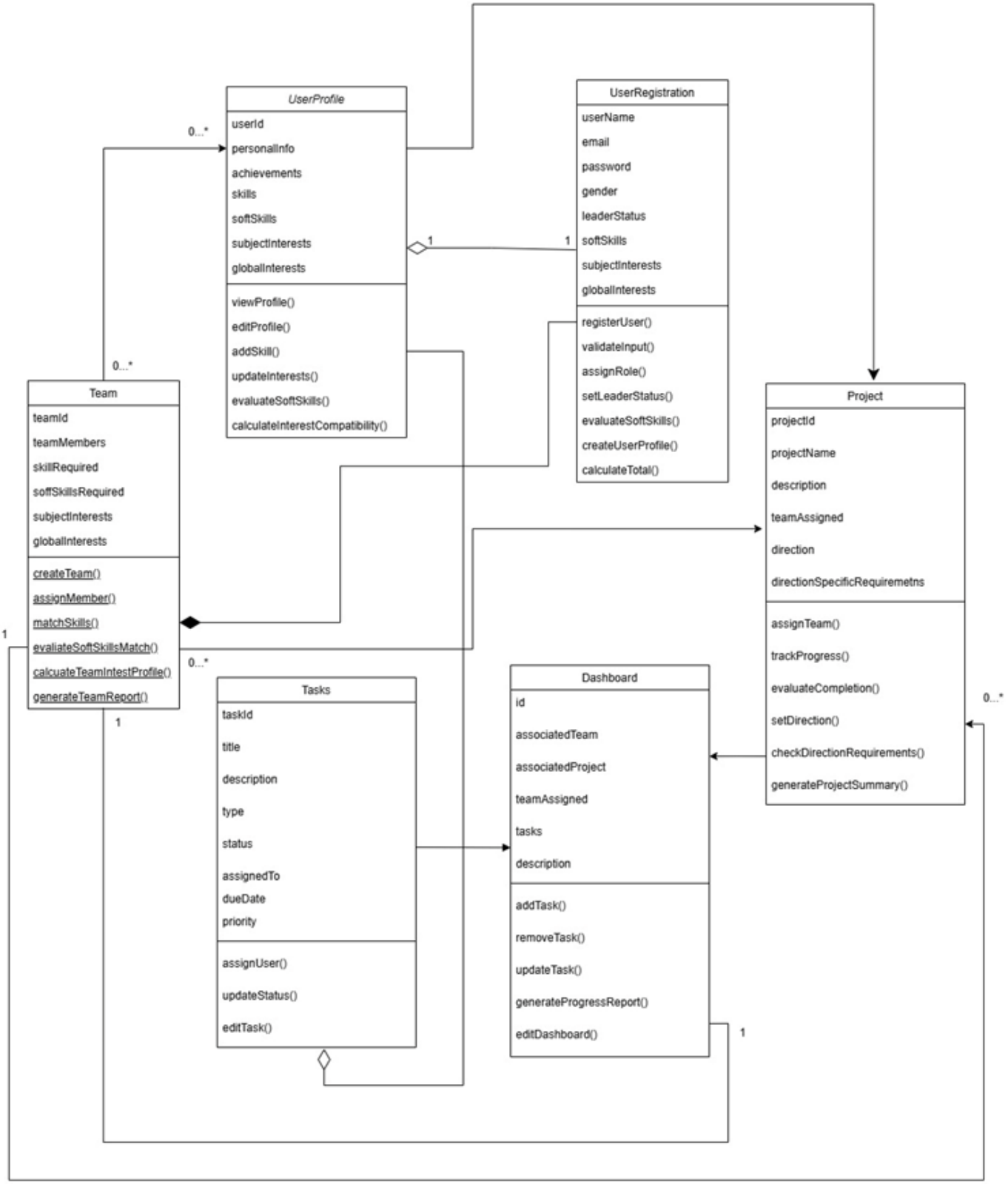
Володимир КОЦЮБИНСЬКИЙ

Вінниця 2024

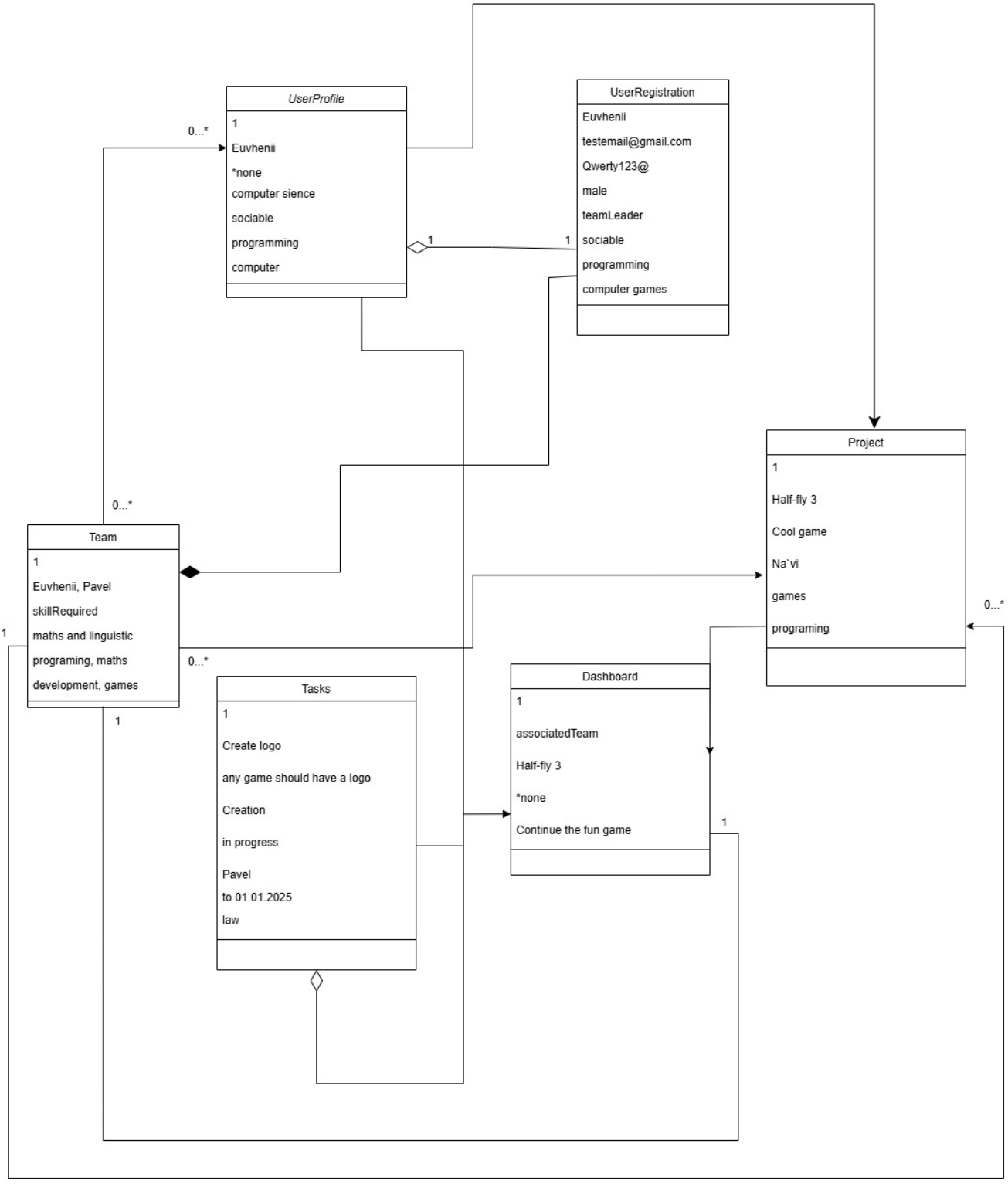
Блок схема клієнт-сервеної архітектури



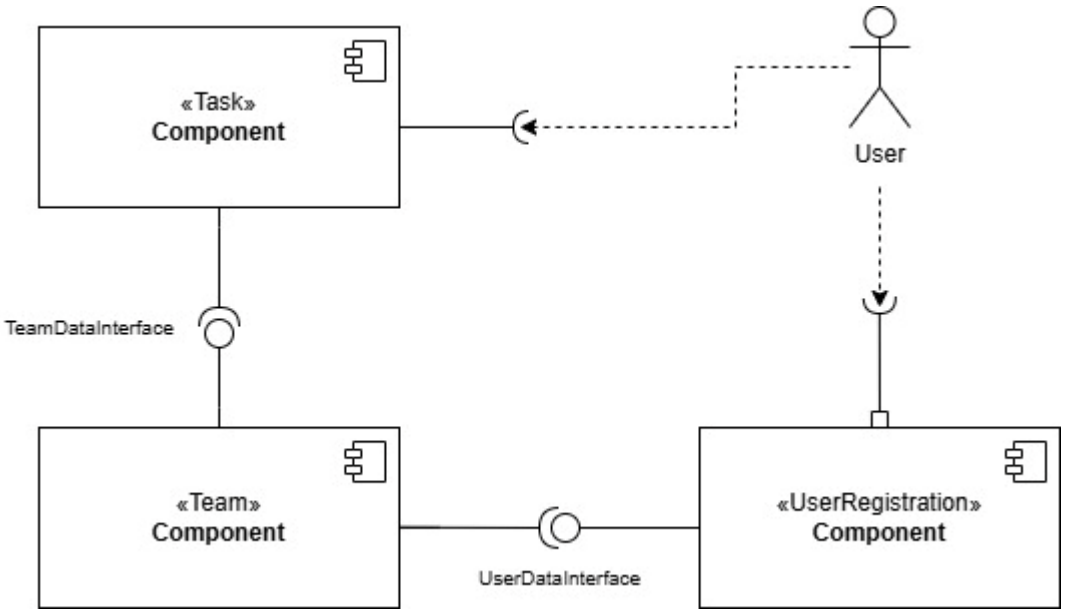
Діаграма класів



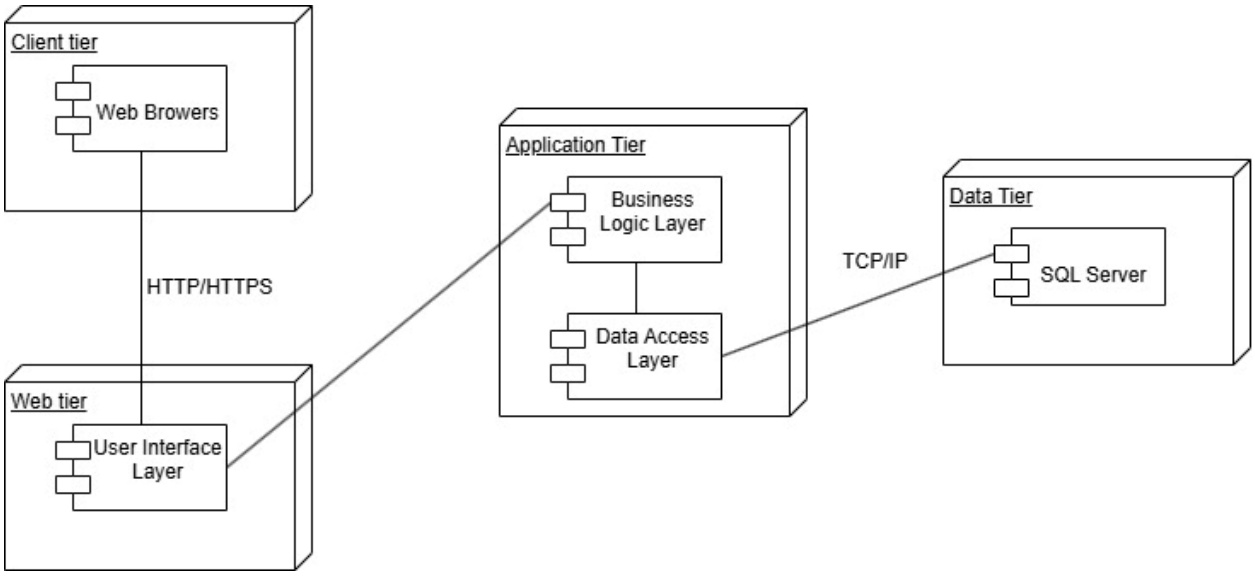
Діаграма об’єктів



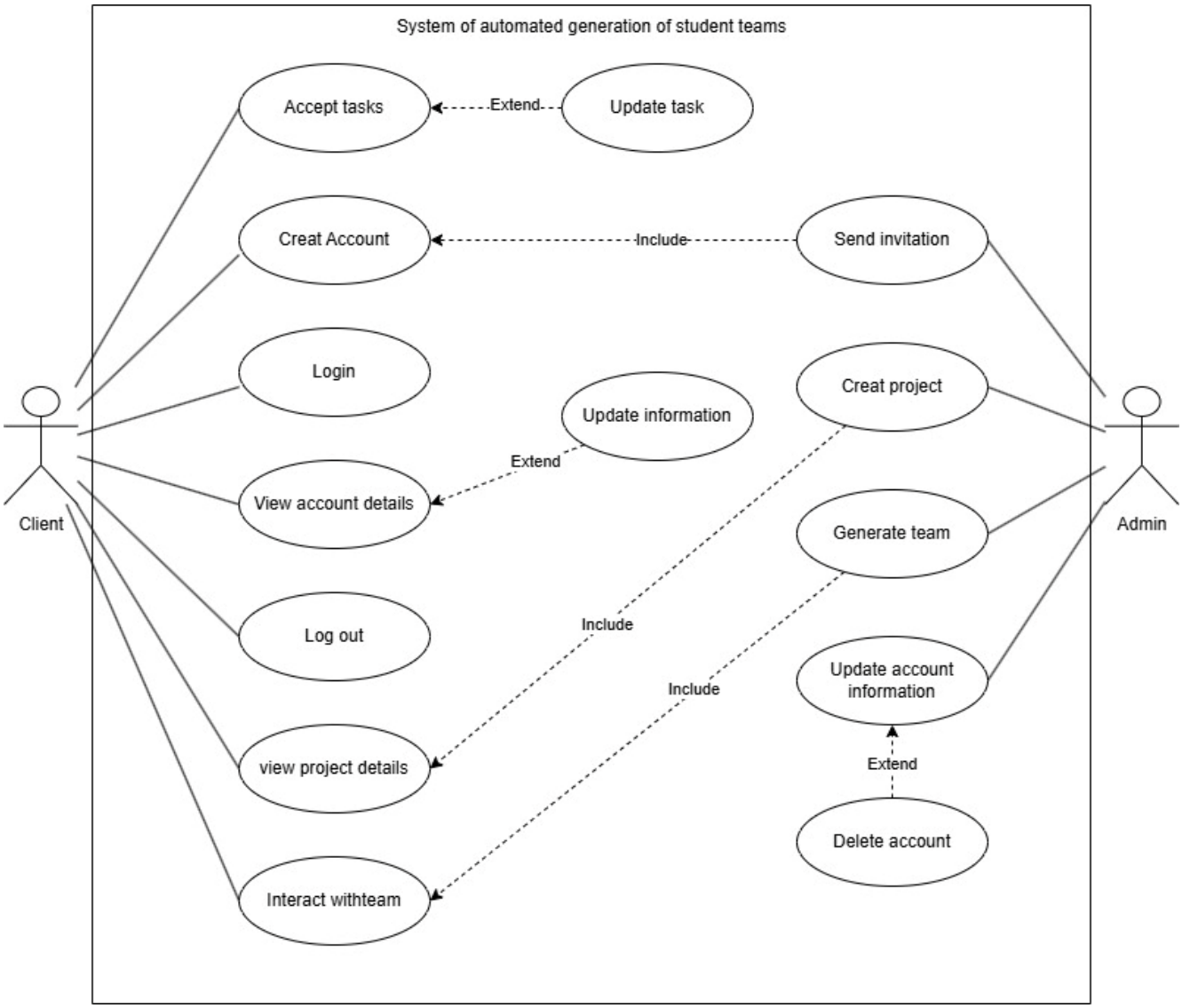
Діаграма компонентів



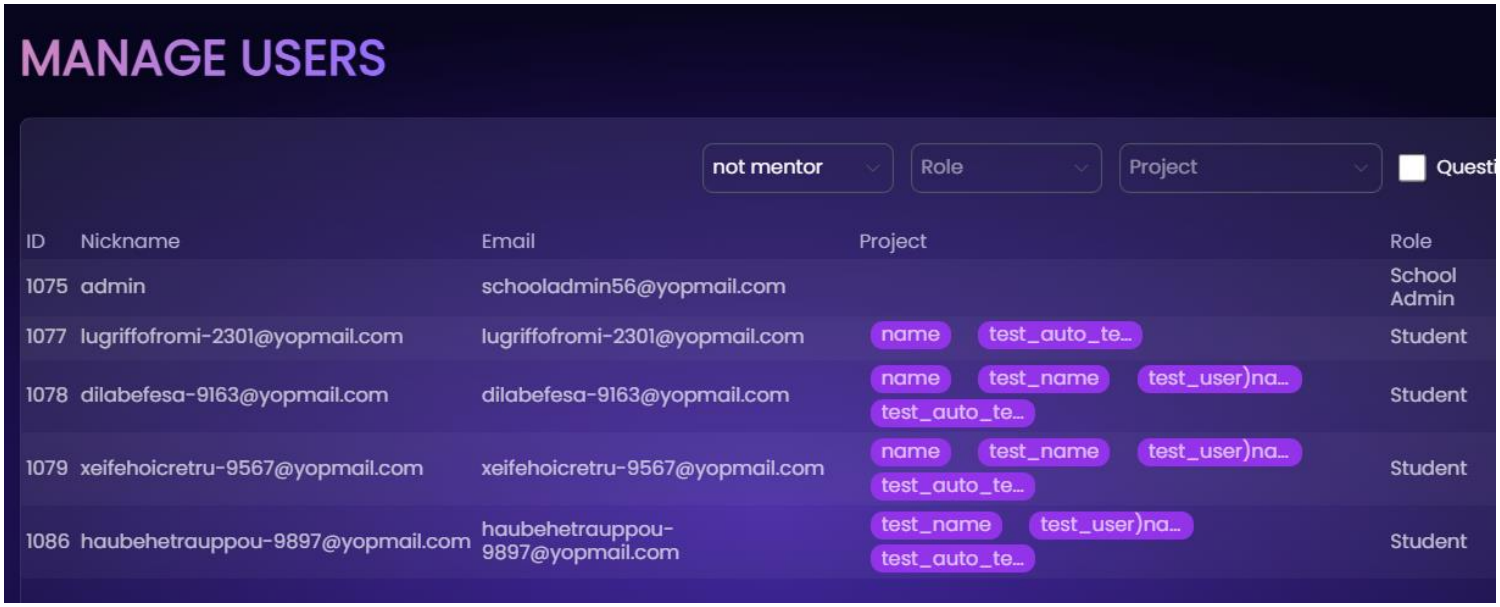
Діаграма розгротання



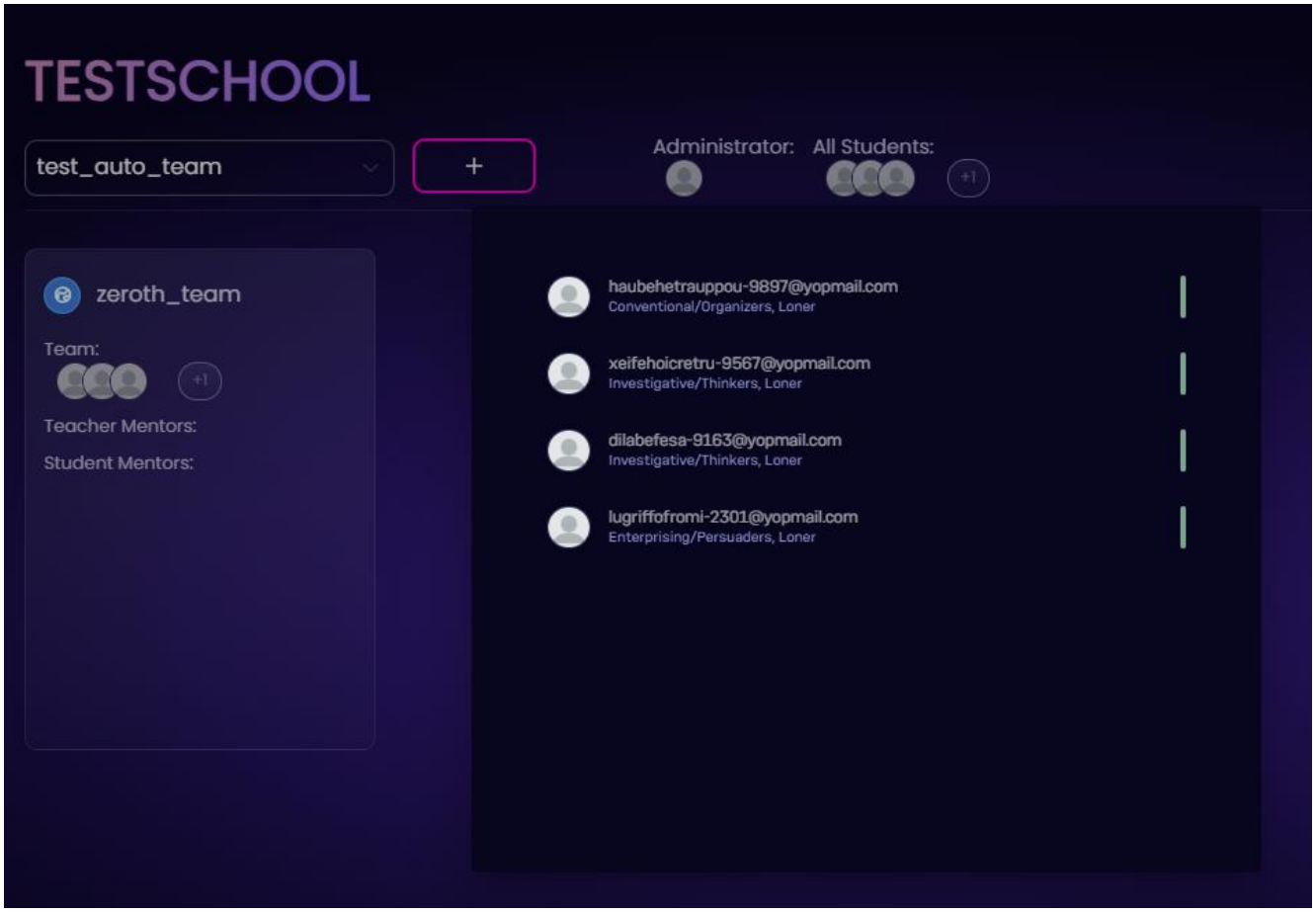
Діаграма варіантів розгортання



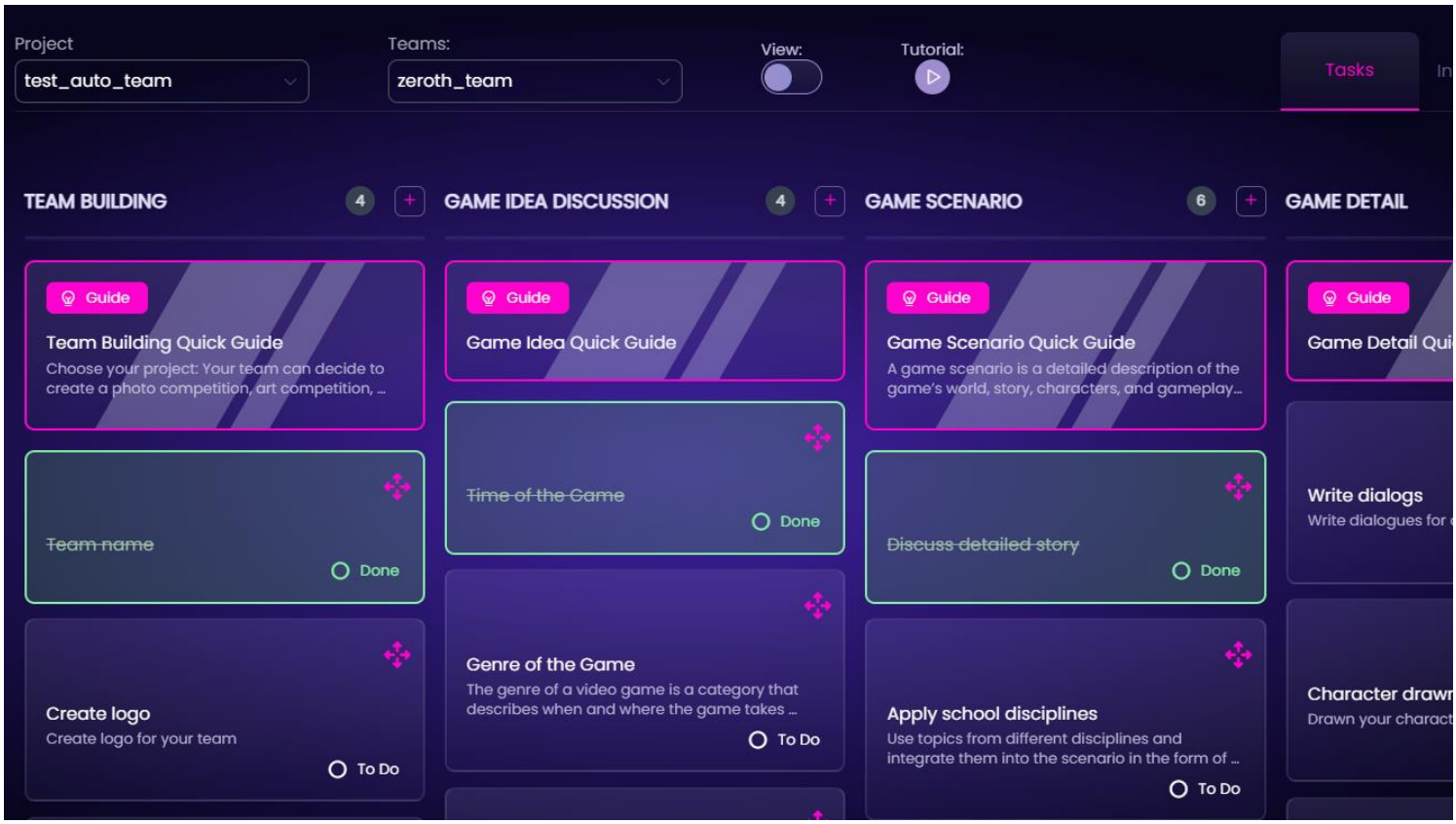
Інтерфейс списку учнів



Інтерфейс списку команди



Інтерфейс таблиці завдань



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 2021р.

АКТ

впровадження результатів магістерської роботи

Чеги Євгенія Іванович «Розробка клієнт-серверної системи для автоматизації процесу формування команд учнів для навчання онлайн. Частина 2. Клієнтська 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 Чегою Є. І. 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 Є. І. Чеги 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 Чегою Є. І. передано програмне забезпечення, яке дозволяє підвищити ефективність організації онлайн навчання шляхом розробки клієнтської частини для автоматизації процесу формування команд учнів.

Члени комісії:

Головний спеціаліст. Сергій ЖИТАНСЬКИЙ

Керівник відділу Олександр КИРИЛЕНКО