Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

Кафедра комп'ютерних систем управління

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка автоматизованої системи управління навчальним процесом школи»

Виконав: студент 2 курсу, групи 2АКІТР-23м
спеціальності 174 – Автоматизація,
комп'ютерно-інтегровані технології та
робототехніка

Юлія КОЗАК
ім'я ПРІЗВИЩЕ

Керівник: к. т. н., доцент, доцент кафедри КСУ
ступінь, звання, посада

Олег КОВАЛЮК
ім'я ПРІЗВИЩЕ

«4» 12 2024 р.

Опонент: доцент кафедри АІІТ
степінь, звання, посада

Костянтин ОВЧИННИКОВ
ім'я ПРІЗВИЩЕ

«4» 12 2024 р.

Допущено до захисту

Зав. Кафедри КСУ

В'ячеслав КОВТУН

«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 – Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ
Зав. кафедри КСУ

В'ячеслав КОВТУН

"14" жовтня 2024 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Козак Юлії Миколаївні.
(прізвище, ім'я, по батькові)

1. Тема роботи. Розробка автоматизованої системи управління навчальним процесом школи
керівник роботи к.т.н., доцент Ковалюк Олег Олександрович
затверджені наказом ВНТУ від "17" вересня 2024 року №310
2. Термін подання студентом роботи "1" грудня 2024 року
3. Вихідні дані до роботи: організація управління навчальним процесом школи, загальні принципи оптимізації навчального процесу та полегшення адміністративних задач, потреби та вимоги до автоматизації навчального процесу школи. Перелік матеріалів до розробки:
   1. Структура та організація процесу навчання. [Електронний ресурс]. – URL: https://studentam.net.ua/content/view/2269/97/
   2. Карплюк С.О., Вакалюк Т.А. Огляд функціональних можливостей програмного забезпечення для управління освітнім процесом закладу вищої освіти. [Електронний ресурс] – URL: http://eprints.zu.edu.ua/27263/1/%D0%9A%D0%B0%D1%80%D0%BF%D0%BB%D1%8E%D0%BA_%D0%92%D0%B0%D0%BA%D0%B0%D0%BB%D1%8E%D0%BA_WOS.pdf

4. Зміст текстової частини: вступ, аналіз предметної області та існуючих систем автоматизації навчального процесу, огляд та вибір інструментів розробки системи, проєктування, розробка та тестування розробленої системи, економічна частина, висновки

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень). Архітектура системи, структура бази даних, UML Use-Case діаграми, вигляд екранів розробленого додатку.

1. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
4	Кавецький В.В. – доцент кафедри економіки підприємства і виробничого менеджменту		

2. Дата видачі завдання "14" жовтня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз об'єкта автоматизації	03.09.24	10.09.24	
2	Аналіз існуючих систем для автоматизації системи управління навчальним процесом школи	13.09.24	23.09.24	
3	Аналіз функцій систем управління школами та посктування архітектури автоматизованої системи	23.09.24	04.10.24	
4	Розробка та тестування автоматизованої системи	04.10.24	11.11.24	
5	Розрахунок економічної частини	11.11.24	18.11.24	
6	Оформлення матеріалів до захисту	18.11.24	01.12.24	
7	Захист МКР	09.12.24	09.12.24	

Студент _____ Юлія КОЗАК
(підпис) (*Ім'я ПРІЗВИЩЕ*)

Керівник роботи _____ Олег КОВАЛЮК
(підпис) (*Ім'я ПРІЗВИЩЕ*)

АНОТАЦІЯ

УДК 004.45

Козак Ю. М. Розробка автоматизованої системи управління навчальним процесом школи. Магістерська кваліфікаційна робота зі спеціальності 174 - Автоматизація, комп'ютерно-інтегровані технології, та робототехніка, освітня програма – Інтелектуальні комп'ютерні системи. Вінниця: ВНТУ, 2024. - 129 с.

Українською мовою, 4 розділи, 40 джерел, рис.:23, табл.: 12

Мета роботи – розробка автоматизованої системи управління навчальним процесом, яка спрямована на підвищення ефективності організації навчання, спрощення адміністрування освітнього процесу, забезпечення доступу до актуальної інформації для всіх учасників (учнів, вчителів, адміністрації) та інтеграцію сучасних технологій у шкільне середовище.

В оглядово-аналітичній частині проаналізовано ключові аспекти та значущість розробки такої системи, а також здійснено вибір сучасного стеку технологій із детальним обґрунтуванням кожного з його компонентів. У практичній частині виконано проєктування та розробку системи, що охоплює розробку структури бази даних, створення UML-діаграм, визначення ключових функціональних можливостей системи та вибір відповідних технологій та розроблене програмне забезпечення з використанням JavaScript та бібліотеки React та фреймворку .NET. У економічній частині проаналізований технічний рівень і розрахована собівартість та витрати на реалізацію проекту.

Ілюстративна частина складається з 16 рисунків, які включають як і UML діаграми, так і результати роботи системи.

Ключові слова: автоматизована система, управління, стек технологій, React, JavaScript, .NET, UML-діаграми.

ANNOTATION

UDC 004.45

Kozak Y. M. Development of an automated system for managing the educational process of a school. Master's qualification work in the specialty 174 - Automation, computer-integrated technologies, and robotics, educational program - Intelligent computer systems. Vinnytsia: VNTU, 2024. - 129 p.

In Ukrainian, 4 chapters, 40 sources, fig.:23, table: 12.

The purpose of the work is to develop an automated system for managing the educational process, which is aimed at increasing the efficiency of organizing learning, simplifying the administration of the educational process, ensuring access to relevant information for all participants (students, teachers, administration) and integrating modern technologies into the school environment.

The review and analytical part analyzes the key aspects and significance of developing such a system, and also selects a modern stack of technologies with a detailed justification of each of its components. In the practical part, the design and development of the system was performed, which includes the development of the database structure, the creation of UML diagrams, the determination of the key functional capabilities of the system and the selection of appropriate technologies, and the software was developed using JavaScript and the React library and the .NET framework. In the economic part, the technical level was analyzed and the cost and expenses for the project implementation were calculated.

The illustrative part consists of 16 figures, which include both UML diagrams and the results of the system.

Keywords: automated system, management, technology stack, React, JavaScript, .NET, UML diagrams.

ЗМІСТ

ВСТУП

Актуальність теми. В умовах стрімкого розвитку інформаційних технологій автоматизація освітнього процесу стає ключовим елементом для сучасних навчальних закладів. Впровадження новітніх технологій в управлінні освітою не лише підвищує ефективність, але й оптимізує рутинні завдання, звільняючи час для більш творчої та продуктивної діяльності.

Сьогодні, в умовах глобальних викликів, таких як війна чи сплески вірусних хвороб, які змушують переходити на дистанційне навчання, особливо важливим є забезпечення безперервності навчання. Перехід на дистанційні формати навчання виявив недостатності існуючих систем управління, які не завжди забезпечують інтеграцію всіх аспектів навчального процесу, таких як облік учнів, ведення документації, комунікація з батьками та моніторинг успішності. Ці фактори підкреслюють необхідність розробки інтегрованої автоматизованої системи, яка зможе адаптуватися до потреб сучасного освітнього середовища.

Автоматизовані системи дозволяють не лише підвищити ефективність ведення документації, але й забезпечити прозорість та доступність інформації для всіх учасників навчального процесу. Керівники навчальних закладів отримують можливість більш оперативно приймати управлінські рішення, а викладачі та учні — зосередитися на навчанні.

В умовах, коли навчальний процес переходить у нові формати, важливо також забезпечити якісну взаємодію між усіма учасниками: учнями, викладачами, адміністраторами та батьками. Інтеграція інформаційних технологій у навчальний процес дозволить створити більш гнучке, доступне та персоналізоване навчання, відповідаючи на сучасні виклики та вимоги.

Таким чином, метою даного дослідження є покращення ефективності та якості управління навчальним процесом шляхом впровадження автоматизованої системи, яка вдосконалить існуючі підходи та стане основою для подальшого розвитку освітнього процесу в Україні та за її межами.

Метою роботи є покращення управління різними аспектами навчального

процесу шляхом розробки інтегрованої системи, яка об'єднує в собі функції моніторингу, аналізу та оптимізації.

Для досягнення цієї мети передбачено вирішення наступних завдань:

• здійснити аналіз предметної області;

• провести огляд існуючих систем;

• визначити та застосувати сучасні методи та програми автоматизації процесу;

• розробити автоматизовану систему управління навчальним процесом

Об'єктом дослідження є навчальний процес, а саме його організаційні аспекти та взаємодія учасників**.**

Предмет дослідження: є розробка автоматизованої системи управління, спрямованої на оптимізацію організаційних процесів у навчальному закладі.

Практична цінність роботи полягає у створенні ефективної та інтегрованої автоматизованої системи управління навчальним процесом, яка дозволяє оптимізувати адміністративні завдання, зменшити рутинне навантаження на вчителів та керівників, а також забезпечити прозорість та оперативність в управлінні освітніми ресурсами. Впровадження цієї системи сприятиме покращенню організації навчання, швидшому доступу до інформації для всіх учасників процесу, підвищенню якості освітніх послуг та загальній ефективності роботи навчального закладу.

Інноваційність розробки полягає у покращенні навчального процесу, використовуючи сучасні технології для оптимізації рутинних завдань та поліпшення взаємодії всіх учасників освітнього процесу. Система включає зручні інтерфейси для комунікації між учасниками освітнього процесу, інструменти моніторингу успішності в реальному часі, а також сучасні рішення для захисту даних, забезпечуючи їх конфіденційність. Гнучкість і масштабованість системи дозволяють адаптувати її до різних типів навчальних закладів, сприяючи розвитку нових підходів у навчанні та підвищенню ефективності освітнього процесу.

Апробація результатів дослідження: Представлені в роботі результати апробовані в результаті участі в конференції «LIII Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2024)». За результатами виконаних досліджень підготовлено та подано тези

доповіді. РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ ШКОЛИ [24].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ СИСТЕМ АВТОМАТИЗАЦІЇ НАВЧАЛЬНОГО ПРОЦЕСУ

1.1 Аналіз предметної області

В даній роботі аналізом об’єкту автоматизації є сама школа, а саме її навчальний процес. [1]

Навчальний процес в школі – це комплексна система активностей, спрямованих на забезпечення навчання та розвитку учнів. Навчальний процес є ключовим елементом шкільної діяльності, спрямованим на формування знань, навичок, цінностей та розвиток особистості учня.

Системи автоматизації освітнього шкільного процесу - це програмне забезпечення головною функцією якого є надання можливості ведення документації та перегляд історії за тривалий час. За допомогою історії можна проводити аналіз та вести певну статистику.[10]

Системи даного роду будуються на основі клієнт-серверної архітектури. Система повинна бути розподіленою та масштабованою, що дає змогу проводити маніпуляції над великим потоком даних. Всі розподілені мікро-сервіси повинні бути зв’язані між собою та реплікувати дані, щоб система була відмово стійкою.

Вона складається з таких компонентів:

- сервери, що виступають в ролі сховища даних та їхнього життєвого циклу;
- клієнти, які запитують дані, та постачають їх в залежності від прав доступу;
- мережа, надає можливість взаємодії між клієнтами та серверами, а також якщо система широко масштабована надає взаємодію між мікро-сервісами.

Використання автоматизації освітнього процесу у школі варто розглянути з двох сторін: зі сторони розробника та зі сторони користувача системи.

Перевагою розробника є те що в минулому він здобув шкільну освіту і аналіз та дослідження предметної області проведено в шкільні роки. Також дане питання не обмежує в технологіях імплементації, а програмне рішення є простим через те, що

основна ціль – це забезпечення доставки/збору даних. Написання абстракцій спрощує реалізацію системи і впровадження нового функціоналу є швидким.

Також можливість інтегрування різних існуючих сервісів, таких як Telegram спрощує задачу написання взаємодії з клієнтом.

Перевагою користувача є зручний інтерфейс, можливість завантажувати дані в різних структурованих форматах. Також, якщо є інтеграції з сторонніми додатками, які вже використовуються ним, надають можливість «в один клік» почати використовувати дане програмне забезпечення.

Для покращення навчального процесу в сучасному світі використовують технології для забезпечення автоматизації даного процесу, що забезпечить якісніше навчання та можливість ведення обліку та аналізу освітнього процесу, це можуть бути:

1. Електронні Журнали та Портали:

Системи управління навчальним процесом (LMS): Вони дозволяють вчителям публікувати матеріали для навчання, виставляти оцінки, створювати завдання та взаємодіяти з учнями.

Електронні журнали: Замість паперових журналів де ведеться облік успішності учнів, використовуються електронні системи для автоматизації цього процесу.

2. Системи Дистанційного Навчання:

Платформи для віддалених уроків: Забезпечують можливість віддаленого навчання, включаючи відео-лекції, електронні тести та інтерактивні завдання.

Віртуальні класи: Створюють можливість для взаємодії учнів та вчителів в онлайн-середовищі, навіть якщо вони фізично знаходяться в різних місцях.

3. Інтерактивні Технології:

Електронні дошки: Застосування інтерактивних дошок для ефективного ведення уроків, демонстрації матеріалів та взаємодії з учнями.

Ігрові технології: Використання ігор для навчання, що робить процес більш захоплюючим та стимулює інтерес до предметів.

4. Аналітичні та Звітні Системи:

Системи аналізу успішності: Допомагають вчителям та адміністрації відстежувати прогрес учнів та ефективність навчального процесу.

Звітні системи для адміністрації: Надають можливість аналізу даних щодо фінансів, розкладу, ресурсів та інших аспектів шкільного управління.

5. Мобільні Додатки:

Додатки для батьків та учнів: Забезпечують доступ до інформації про успішність, розклад уроків, анонси подій тощо.

Мобільні додатки для вчителів: Спрощують процес ведення журналу, виставлення оцінок, спілкування з батьками.

6. Інтернет речей (IoT):

Системи безпеки та контролю доступу: Дозволяють відслідковувати рух осіб у шкільному приміщенні та забезпечують безпеку.

Системи "розумної" інфраструктури: Використання IoT для оптимізації освітнього середовища, такі як системи освітлення, клімат-контролю тощо.

Перевагами даних систем є підвищення ефективності навчання, системи дозволяють адаптувати навчальний матеріал до потреб кожного учня. Підвищуючи ефективність навчання, зручність та доступність, бо з’являється можливість навчатися віддалено. Миттєвий доступ до інформації, батьки, учні та вчителі можуть швидко отримати доступ до інформації про успішність, розклад, матеріали занять тощо. Зацікавлення учнів, використання технологій робить навчальний процес цікавішим та стимулює активну участь учнів. Спрощення адміністративних процесів, автоматизація адміністративних завдань полегшує роботу вчителів та адміністрації. Але також, звісно, є й недоліки використання таких систем: технічні проблеми, приватність та захист даних, навчання та педагогічний процес (використання великої кількості технологій може призвести до втрати особистого контакту між вчителем та учнем), витрати на впровадження та обслуговування.

1.2 Аналіз об’єкта автоматизації

Об'єкт автоматизації, у даному випадку навчальний процес в школі, може бути аналізований з різних точок зору для ефективної автоматизації.[1] Нижче наведено деякі аспекти аналізу об'єкта автоматизації:

1. Процеси:
    - Визначення ключових етапів навчального процесу в школі.
    - Аналіз ідентифікації, реєстрації та оцінки студентів.
    - Вивчення потоків даних і інформації між вчителями, учнями, адміністрацією та батьками.
2. Інформаційні потреби:
    - Визначення видів інформації, які потрібні для керівництва, вчителів, учнів і батьків.
    - Оцінка ефективності поточних систем збору та обробки інформації.
3. Технічні можливості:
    - Оцінка наявних технічних засобів та інфраструктури для підтримки автоматизації.
    - Аналіз можливостей впровадження нових технологій (наприклад, електронні підручники, платформи для дистанційного навчання).
4. Бізнес-процеси:
    - Вивчення організаційної структури школи та розподілу обов'язків між вчителями, учнями, адміністрацією і батьками.
    - Визначення можливостей для оптимізації бізнес-процесів шляхом автоматизації.
5. Безпека:
    - Аналіз важливих аспектів безпеки даних та конфіденційності інформації.
    - Визначення потенційних ризиків та розробка стратегій їх запобігання.
6. Супровід користувачів:
    - Визначення потреб у навчанні персоналу та учнів для коректного використання нових систем.
7. Метрики успішності:

- Розробка конкретних метрик для вимірювання ефективності автоматизованого навчального процесу.

Під час аналізу об'єкта автоматизації навчального процесу в школі можна використовувати різні підходи та методи для отримання комплексної інформації та ефективних рішень. Розробка послідовності дій та подій в рамках навчального процесу для визначення оптимальних точок автоматизації. Ось кілька з них:

1. SWOT-аналіз:
   - Strengths (Сильні сторони): Визначення переваг та сильних аспектів навчального процесу.
   - Weaknesses (Слабкі сторони): Аналіз недоліків та слабких моментів, які можна покращити через автоматизацію.
   - Opportunities (Можливості): Визначення можливостей для впровадження нових технологій та процесів.
   - Threats (Загрози): Виявлення можливих загроз і ризиків.
2. Створення діаграм потоків для візуалізації та розуміння руху інформації та процесів в навчальному процесі.
3. Проведення опитувань серед учнів, вчителів, адміністрації та батьків для отримання думок та потреб стосовно автоматизації.
4. Порівняння існуючих систем та технологій з можливими альтернативами для вибору найбільш оптимальних рішень.
5. Вивчення кейсів успішної автоматизації навчального процесу в інших школах або освітніх установах.
6. Визначення конкретних вимог від кінцевих користувачів (вчителів, учнів, адміністрації) щодо автоматизації.
7. Створення прототипів або концепцій автоматизованих систем для вивчення їх ефективності та прийняття рішень на ранніх етапах.

1.3 Аналіз існуючих систем для автоматизації системи управління навчальним процесом школи

З початком пандемії освітній процес поступово автоматизувався та перебіг у інтернет на існуючі платформи, одна з популярних це дистанційна система навчання «Google Classroom»[3]. Також до аналогів можна віднести систему «Moodle»[4].

1.3.1 Система для навчання «Google Classroom»

Дистанційна система навчання «Google Classroom»[3] – продукт одного з лідерів IT індустрії. Варто відмітити що Google раніше мав великий набір інструментів пов’язаних з освітою та навчанням. Їхнє об’єднання стало етапом створення Google Classroom. Організація та налаштування потребує великих зусиль та спеціальних навичок. Немає можливості гнучкого налаштування, що призводить до перебудови освітнього процесу. Також варто відмітити особливості системи Google Classroom:

- використання Google інструментів (Google диск, Google Doc …);
- створення власного класу або курсу
- поширення навчальних матеріалів
- поширення пропозицій
- організація спілкування між учасниками процесу

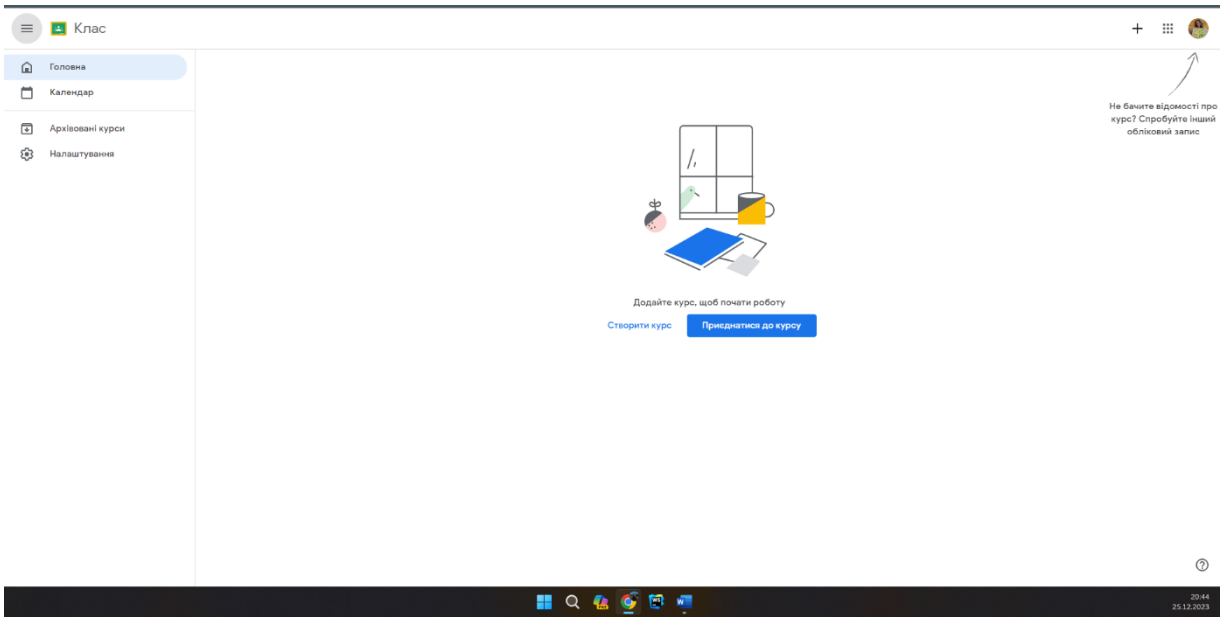


Рисунок 1.1 - Основна сторінка користувача Google Classroom

Переваги дистанційної системи навчання Google Classroom:

- підтримка української мови;
- безкоштовний;
- бренд;
- створювався для шкіл, в свою чергу підходить для вузів;
- реалізовані традиційні функції;
- можливість опублікувати учбовий матеріал, виставляти оцінки в журналі.

Недоліки:

- малий арсенал учбових інструментів;
- важко структуровані посилання;
- недостатня функціональність інтерфейсу.

1.3.2 Система для навчання «Moodle».

Moodle (Modular Object-Oriented Dynamic Learning Environment) [4] — це одна з найпопулярніших у світі платформ для управління навчанням, яка широко використовується для організації дистанційного навчання. Moodle дозволяє

ефективно організувати весь навчальний процес: забезпечити доступ до навчальних матеріалів, провести контроль знань, оцінювання і комунікацію між усіма учасниками навчання.

Особливості системи:

- відкритість і доступність - система є відкритим кодом, тобто її можна безкоштовно встановити, а також змінювати під конкретні потреби. Це дозволяє навчальним закладам налаштовувати систему під свої задачі, додаючи чи забираючи необхідні функції.
- широкий вибір інструментів для навчання та оцінювання - система пропонує багатий набір інструментів для тестів, опитувань, інтерактивних вправ, форумів, завдань на самоперевірку і навіть кросвордів та флеш-карток. Викладачі можуть індивідуально налаштовувати завдання для кожного студента, що допомагає забезпечити персоналізоване навчання.

Переваги системи Moodle:

- доступність;
- гнучке налаштування за допомогою модулів та плагінів;
- міжнародна підтримка;
- безпека;

Недоліки системи:

- складність для новачків;
- необхідність технічної підтримки;
- вимоги до ресурсів, для великої кількості користувачів потребує потужних серверів.

1.3.3 Постановка задачі дослідження

На підставі проведеного аналізу можна визначити, що впровадження автоматизації в систему управління навчальним процесом школи виявляється ключовою складовою її конкурентоспроможності. З метою збереження конкурентних

позицій на ринку необхідно постійно вдосконалювати процеси, впроваджувати передові технології та підвищувати ефективність існуючих систем.Аналіз конкурентів надає цінні висновки та рекомендації для подальшого розвитку власної освітньої системи. Ці знання можна використовувати для удосконалення різних процесів, впровадження нових технологій та пошуку конкурентних переваг у сфері освіти.

Для автоматизації програмного рішення необхідно використовувати підхід, що дозволить зробити систему універсальною і здатною працювати на різних операційних системах. Для цього слід обрати мову програмування, яка забезпечує кросплатформенну підтримку та однаковий набір функцій на кожній операційній системі. Для повноцінної автоматизації програмного продукту доцільно розділити його на кілька мікросервісів, кожен з яких відповідатиме за окрему функцію. Це допоможе уникнути конфліктів між функціями і зробить продукт стійкішим до можливих збоїв, а також спростить подальше розширення функціоналу.

Після автоматизації необхідно забезпечити безпеку даних та захист автентифікації користувачів. Це можна досягти завдяки ієрархічній архітектурі клієнт-серверного зв’язку. Важливо також розділити бази даних, які працюють з інформацією користувачів та з даними автентифікації, розмістивши їх на різних серверах. Такий підхід дозволить посилити захист даних клієнтів.

2 ПРОЄКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ ШКОЛИ

2.1 Аналіз функцій системи

Автоматизована система управління навчальним процесом школи, інтегрована з навчальними додатками, забезпечує ефективне управління школами, вчителями та учнями.[6] Основні функціональні можливості включають:

Для користувача – директор школи:

1. Реєстрація школи: Кожна школа може зареєструватися в системі, отримуючи окремий профіль, у якому можна керувати навчальним процесом.
2. Налаштування теми та логотипу школи: Директори шкіл мають можливість встановлювати тему оформлення та завантажувати логотип школи, щоб зробити інтерфейс системи персоналізованим.
3. Керування персоналом: Директори можуть додавати нових вчителів до системи або видаляти їх при необхідності.
4. Купівля та розподіл ліцензій: Система дозволяє школам придбати ліцензії для доступу до освітніх додатків і надавати їх учням.

Для користувача – вчитель:

1. Доступ через школу: Вчителі отримують доступ до системи через свою школу, використовуючи корпоративні email-акаунти.
2. Призначення завдань: Вчителі мають можливість створювати, призначати та керувати завданнями для учнів, переглядати їх виконання та ставити оцінки.
3. Перездача завдань: Система дозволяє вчителям надавати учням можливість повторно виконати завдання для покращення оцінки.
4. Управління ліцензіями: Вчителі можуть надавати або відкликати ліцензії учнів на доступ до додатків для навчання.
5. Керування учнями: Вчителі можуть додавати нових учнів, видаляти учнів або переносити їх в інші класи, забезпечуючи актуальність даних.

Для користувача – учень школи:

- Перегляд завдань: Учні можуть переглядати всі завдання, які їм призначив вчитель, а також терміни їх виконання.
- Перегляд оцінок: Учні мають доступ до своїх оцінок за кожне завдання та можуть бачити результати після перевірки.
- Доступ до додатків: За умови надання ліцензії, учні можуть використовувати інтегровані освітні додатки для виконання завдань та навчання.

Для наочного прикладу варіантів використання даної системи було розроблено UML діаграму (рис. 2.1), яка демонструє, як різні користувачі системи (актори) взаємодіють з основними функціональними можливостями системи автоматизованого управління навчальним процесом у приватному закладі освіти.

Діаграма відображає основні функції (варіанти використання), які система повинна виконувати, що полегшує визначення вимог до програмного забезпечення. Вона допомагає чітко визначити ролі користувачів (акторів) і їх взаємодію із системою. Це корисно для розуміння того, хто і як буде використовувати систему. Вона допомагає чітко визначити ролі користувачів (акторів) і їх взаємодію із системою. Це корисно для розуміння того, хто і як буде використовувати систему.

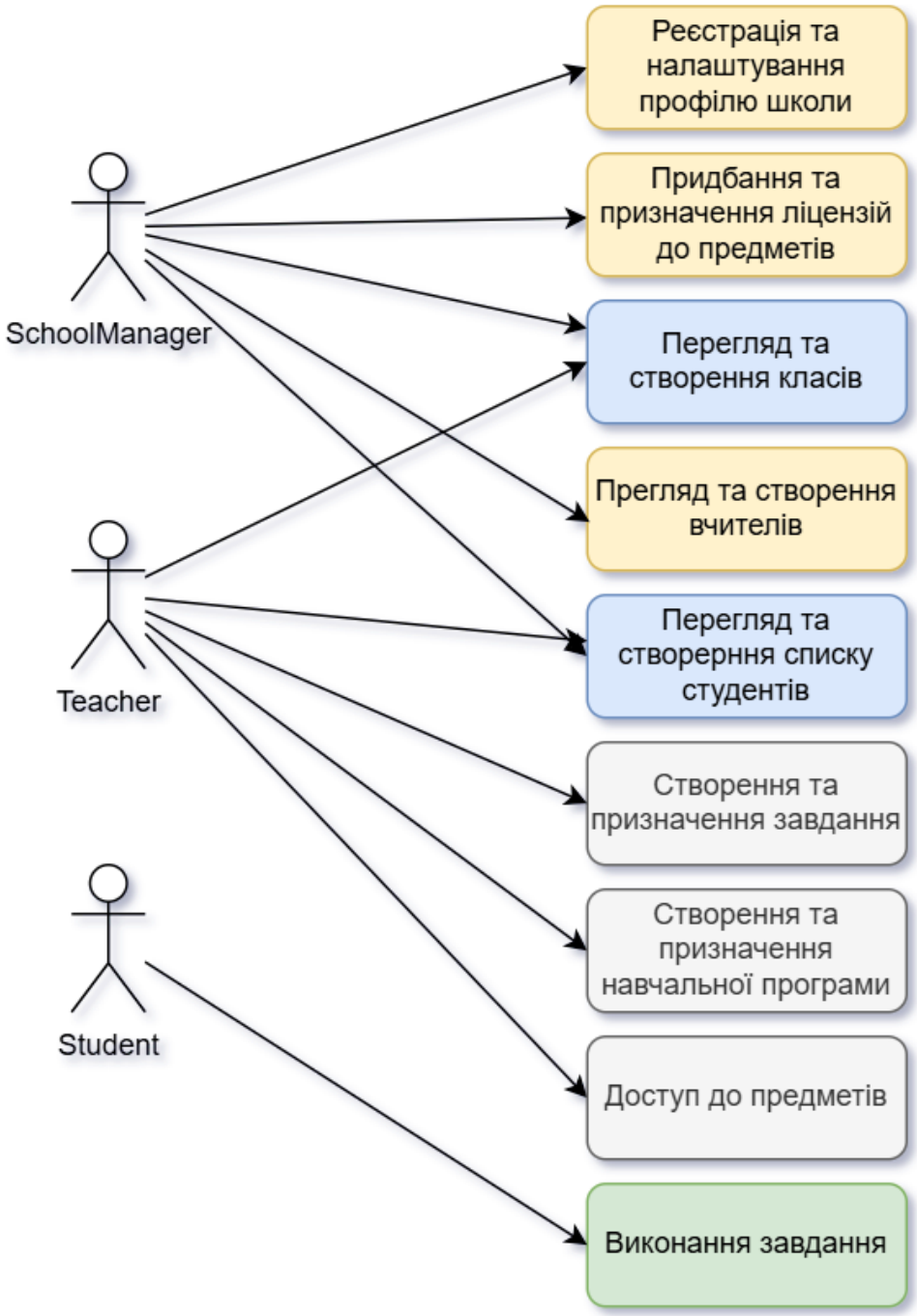


Рисунок 2.1 – UML діаграма варіантів використання

Ця діаграма демонструє логіку роботи системи та взаємодію користувачів із функціональними модулями системи. Вона показує:

- Розподіл ролей у системі.
- Логічний розподіл обов’язків між учасниками процесу.

- Структуру та функціональні можливості автоматизованої системи.

Зв'язки на діаграмі показують, як кожен актор (користувач) взаємодіє з відповідними функціями системи. Наприклад:

- School Manager має доступ до налаштувань школи, створення класів, управління викладачами та студентами.
- Teacher взаємодіє зі студентами через призначення завдань і створення навчальної програми.
- Student отримує доступ до матеріалів та виконує завдання.

Діаграма варіантів використання ілюструє основні функціональні можливості автоматизованої системи управління навчальним процесом. Вона показує взаємодію трьох основних типів користувачів (School Manager, Teacher, Student) із ключовими функціями системи, такими як управління даними, створення завдань і доступ до навчальних ресурсів. Діаграма допомагає зрозуміти, як система може адаптуватися до потреб кожного користувача, забезпечуючи ефективність і прозорість навчального процесу.

2.2 Проєктування архітектури автоматизованої веб системи

Розробка автоматизованої системи управління навчальним процесом школи є складним завданням, що потребує інтеграції різноманітних функціональних модулів та забезпечення безперебійної взаємодії між ними. Основною метою такого проекту є створення інтуїтивно зрозумілого, зручного та ефективного інструменту для адміністрації шкільного навчального процесу, викладачів, учнів та їхніх батьків.

Однією з найважливіших складових успішної реалізації цього проекту є вибір правильних інструментів для розробки веб-застосунку. У сучасному світі існує велика кількість фреймворків, бібліотек та інших інструментів, які можуть бути використані для створення веб-додатків. Вибір конкретних технологій визначає не лише технічні аспекти реалізації проекту, але й впливає на його масштабованість, продуктивність, зручність підтримки та подальший розвиток.

Архітектура інформаційно-обчислювальної системи складається з апаратного забезпечення (комп'ютерів), телекомунікаційних мереж та програмного забезпечення. Рівень розвитку кожного з цих компонентів визначає ефективність інформаційної системи та технології обробки даних, що привело до виникнення таких моделей обробки даних, як телеобробка, файловий сервер, клієнт-серверна архітектура, Інтернет-системи, сховища даних та системи оперативно-аналітичної обробки інформації тощо.

1.2.1 Архітектура файл-сервер

Модель організації системи обробки даних, в якій основні файли та ресурси зберігаються на центральному сервері, до якого мають доступ клієнти в мережі. Клієнтські комп'ютери надсилають запити до сервера на отримання файлів або частин даних, після чого сервер надсилає ці файли для подальшої обробки на клієнтській стороні. При цьому має: централізоване зберігання даних на сервері, передача даних по мережі, обробка даних на клієнтській стороні, підтримка мережевих файлових систем. Але потрібно враховувати, що передача великих файлів між клієнтами і сервером може спричинити значне навантаження на мережу, також якщо сервер виходить з ладу, доступ до даних для всіх клієнтів стає неможливим.[26]

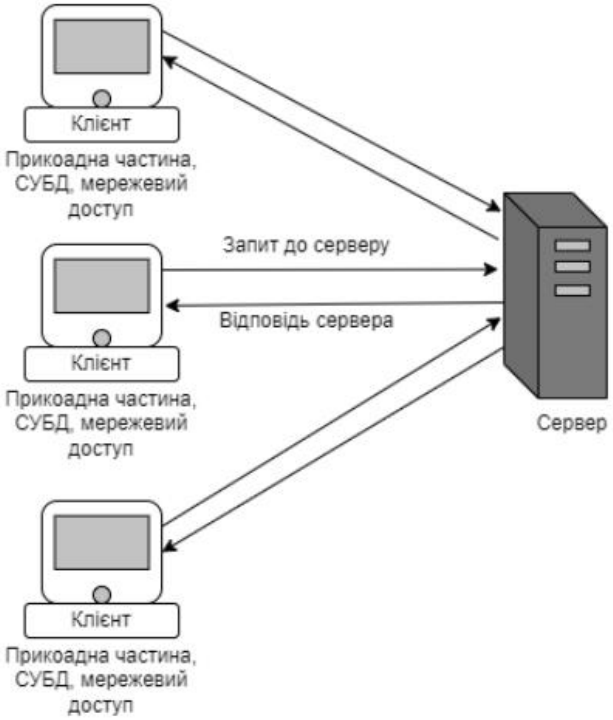


Рисунок 2.2 – Архітектура файл-сервер. [26]

Дана архітектура зазвичай використовується в локальних мережах (LAN) для спільного доступу до файлів в організаціях, де потрібне централізоване зберігання даних, але обробка файлів може виконуватися на клієнтських комп'ютерах.

2.2.2 Архітектура клієнт-сервер

Модель організації обробки даних, де обробка розподілена між двома частинами: клієнтом і сервером. У цій моделі сервер відповідає за зберігання, обробку та управління даними, а клієнт — за взаємодію з користувачем і передачу запитів до сервера. Сервер обробляє запити клієнтів і повертає їм результати. [25]

Якщо коротко підсумувати:

- Спочатку клієнт надсилає свій запит через пристрій з підтримкою мережі.
- Далі мережевий сервер приймає і обробляє запит користувача.
- Нарешті, сервер надсилає відповідь клієнту.

Серверне програмне забезпечення виконує функцію обслуговування клієнтів у системах клієнт-сервер. Для реалізації цієї архітектури зазвичай застосовуються багатокористувацькі системи управління базами даних (СУБД), такі як Oracle чи Microsoft SQL Server. Такі СУБД забезпечують механізми блокування даних і контроль багатокористувацького доступу, що захищає інформацію від ризиків, пов’язаних із одночасним доступом декількох користувачів. Додатково сервер баз даних гарантує безпеку даних від несанкціонованого доступу, оптимізує запити, забезпечує їх цілісність і контролює завершення транзакцій.

У архітектурі клієнт-сервер клієнтські пристрої часто мають обмежені ресурси ("тонкі клієнти"), тоді як сервер виконує більшість завдань, обробляючи запити від багатьох користувачів одночасно. Програмне забезпечення для клієнтів зазвичай включає засоби розробки додатків, генератори звітів, а також популярні офісні програми, такі як електронні таблиці або текстові редактори.

Користувачі через клієнтські пристрої встановлюють з’єднання із сервером, формують запити, які автоматично перетворюються в SQL-запити, і передають їх на сервер для обробки. Сервер, своєю чергою, виконує запит і надсилає результати клієнту.

Особливу роль у такій архітектурі відіграє проміжне програмне забезпечення, яке забезпечує взаємодію клієнтської частини з серверною. У цій моделі клієнт запитує у сервера дані, сервер їх обробляє та повертає тільки необхідну інформацію, зменшуючи обсяг мережевого трафіку.

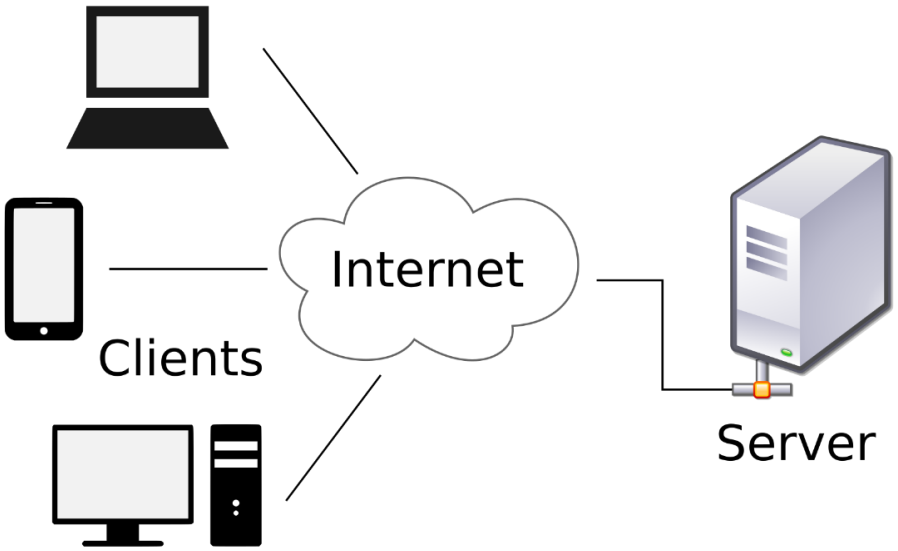


Рисунок 2.3 –Клієнт-серверна модель.[26]

У дворівневій моделі клієнт-сервер взаємодія відбувається між клієнтськими пристроями та сервером баз даних. Такий підхід підходить для невеликих організацій із числом користувачів до 100. Однак зі збільшенням кількості клієнтів навантаження на сервер значно зростає, що може призвести до перевантаження.

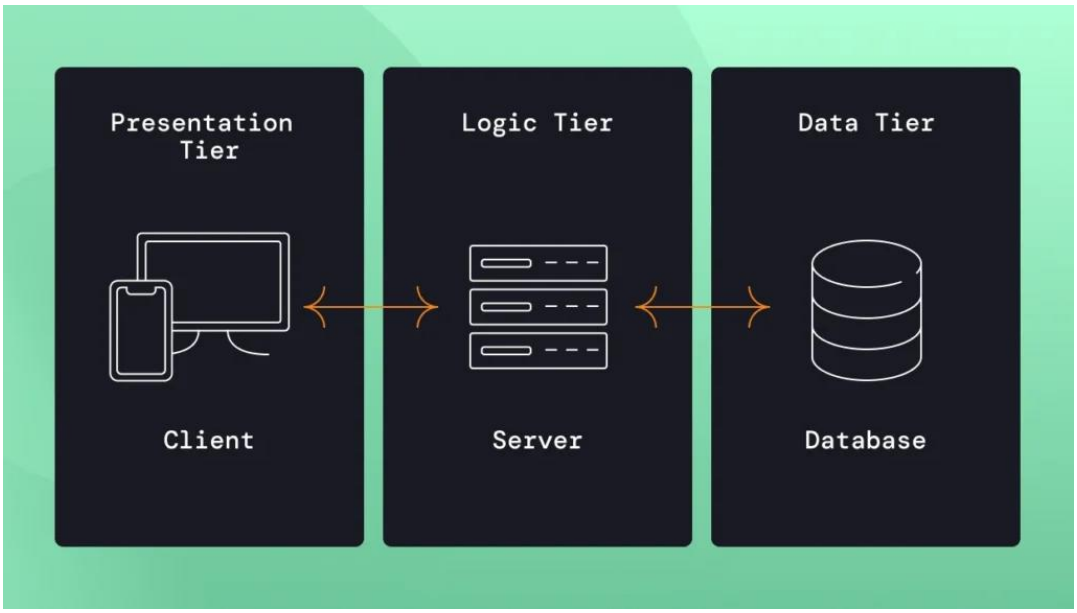


Рисунок 2.4 – Трирівнева архітектура клієнт-сервер.[27]

Трирівнева архітектура додає до цієї моделі сервер додатків, який виконує функції обробки запитів, управління транзакціями, оптимізації черги запитів і взаємодії з сервером баз даних. Така структура покращує масштабованість системи і є оптимальним рішенням для організацій із великою кількістю користувачів. [27]

Переваги архітектури клієнт-сервер:

- Ефективне використання мережевих ресурсів: зменшується обсяг переданих даних, оскільки сервер надсилає клієнту лише результати запитів.
- Висока продуктивність при великій кількості користувачів: система може підтримувати тисячі одночасних з'єднань.
- Централізоване зберігання бізнес-логіки: бізнес-правила зберігаються на сервері, що дозволяє уникати дублювання коду в різних клієнтських додатках.
- Потужні засоби безпеки: сучасні серверні СУБД забезпечують детальне управління привілеями користувачів, захист об'єктів бази даних, резервне копіювання та оптимізацію виконання запитів.

Таким чином, архітектура клієнт-сервер, особливо трирівнева модель, забезпечує високу продуктивність, надійність і гнучкість для організації роботи великих інформаційних систем.

2.2.3 Вибір архітектури системи

Для розробки автоматизованої системи управління навчальним процесом вирішено використати трирівневу архітектуру клієнт-сервер з використанням сучасних технологій для забезпечення масштабованості, безпеки та інтеграції з освітніми додатками.

Система організована за принципом багаторівневої архітектури складається з клієнтської частини (інтерфейсу користувача), серверної частини (бізнес-логіки), бази даних для зберігання інформації та хмарної інфраструктури для надійної роботи й масштабування.

Клієнтський рівень у трирівневій архітектурі зазвичай представлений графічним інтерфейсом, який виступає першим рівнем взаємодії для кінцевого

користувача. Цей рівень не повинен напряму взаємодіяти з базою даних (задля забезпечення безпеки), не має виконувати основні бізнес-операції (для забезпечення масштабованості) та не повинен зберігати стан додатка (для підвищення надійності). Прості елементи бізнес-логіки, як-от авторизація, шифрування, перевірка даних на коректність, сортування чи групування завантажених даних, можуть бути реалізовані на цьому рівні.

Другий рівень — сервер додатків — відповідає за виконання основної частини бізнес-логіки. На ньому обробляються запити клієнтів, реалізуються бізнес-процеси та організовується взаємодія з третім рівнем. Частина функціональності, як-от прості операції або збережені процедури, може бути передана на перший чи третій рівень відповідно.

Сервер баз даних, що розташований на третьому рівні, зазвичай працює на окремому обладнанні або кластері серверів. До нього через мережу підключаються один або кілька серверів додатків, які забезпечують доступ клієнтських пристроїв до даних. Така конфігурація сприяє підвищенню надійності, безпеки та масштабованості системи.

Порівняно з традиційними моделями, такими як клієнт-сервер чи файл-сервер, трирівнева архітектура має низку переваг:

1. Масштабованість: дозволяє ефективно розширювати систему під збільшення навантаження.
2. Гнучкість налаштувань: ізоляція рівнів дає змогу швидко адаптувати систему до змін або виконувати обслуговування без впливу на інші рівні.
3. Високий рівень безпеки: дані й бізнес-логіка розташовані на серверах, що зменшує ризики несанкціонованого доступу.
4. Надійність: ізольованість рівнів дозволяє зменшити вплив збоїв на загальну роботу системи.
5. Низькі вимоги до мережевої пропускної здатності між клієнтськими пристроями та сервером додатків.

6. Зниження вартості клієнтських пристроїв: клієнтський рівень не потребує значних обчислювальних ресурсів, тому терміналом може бути не тільки комп’ютер, а й, наприклад, мобільний пристрій.

Незважаючи на численні переваги, є й певні недоліки:

1. Складність розробки: створення трирівневих систем потребує більшого обсягу ресурсів і знань.
2. Складність розгортання та адміністрування: інтеграція всіх рівнів вимагає додаткових зусиль.
3. Високі вимоги до серверного обладнання: сервери додатків і баз даних потребують потужних апаратних ресурсів, що збільшує витрати.

Попри ці недоліки, трирівнева архітектура є доцільною для використання завдяки перевагам, які суттєво перевищують її недоліки. Це робить її оптимальним вибором для сучасних масштабованих і безпечних систем.

Компоненти системи:

1. Клієнтська частина (Frontend): Реалізована для побудови інтуїтивно зрозумілого та зручного користувацького інтерфейсу.
2. Серверна частина (Backend): Виконує бізнес-логіку, обробляє запити від клієнтської частини та взаємодіє з базою даних.
3. База даних (Database): Зберігання всіх даних відбувається в базі даних, яка використовується для управління даними про школи, учнів, вчителів, завдання та ліцензії.
4. Хмарна інфраструктура (Cloud Infrastructure): Система розміщена на хостингу, що забезпечує надійність, безперервність та безпеку роботи.

2.3 Вибір СУБД та проєктування моделі даних

При розробці автоматизованої системи управління навчальним процесом було прийнято рішення використовувати систему управління базами даних PostgreSQL. Це обґрунтовано її універсальністю, продуктивністю та відповідністю сучасним вимогам

до зберігання і обробки даних. Крім того, система побудована на мікросервісній архітектурі, що забезпечує гнучкість, масштабованість і високу надійність.

PostgreSQL — одна з найпопулярніших і найпотужніших реляційних систем управління базами даних із відкритим кодом.[22] Вибір саме цієї СУБД для реалізації проєкту зумовлений кількома важливими перевагами:

PostgreSQL забезпечує високу цілісність даних завдяки транзакційній моделі на основі ACID (атомарність, узгодженість, ізоляція, довговічність). Вона має вбудовані механізми шифрування, багаторівневий контроль доступу, можливість налаштування ролей і прав користувачів, що критично важливо для роботи з конфіденційною інформацією про учнів, викладачів і навчальні матеріали. Дозволяє створювати збережені процедури, функції, тригери та інші механізми, що дозволяють реалізувати складні алгоритми роботи з даними. Це зменшує навантаження на сервер додатків і забезпечує ефективну роботу бізнес-логіки на рівні бази даних. PostgreSQL ефективно працює з великими обсягами даних і може обслуговувати сотні тисяч користувачів одночасно. Завдяки підтримці паралельного виконання запитів і індексації, СУБД гарантує високу швидкість обробки запитів навіть при значному навантаженні. Підтримує велику кількість типів даних, можливість створення користувацьких типів, а також безліч розширень (наприклад, PostGIS для роботи з геоданими), що дозволяє адаптувати систему до специфічних потреб навчального процесу. PostgreSQL добре інтегрується з хмарними платформами, такими як AWS (Amazon Web Services), Google Cloud Platform і Microsoft Azure. Для даної системи управління навчальним процесом база даних розгорнута в хмарному середовищі AWS, що забезпечує доступність і масштабованість інфраструктури.

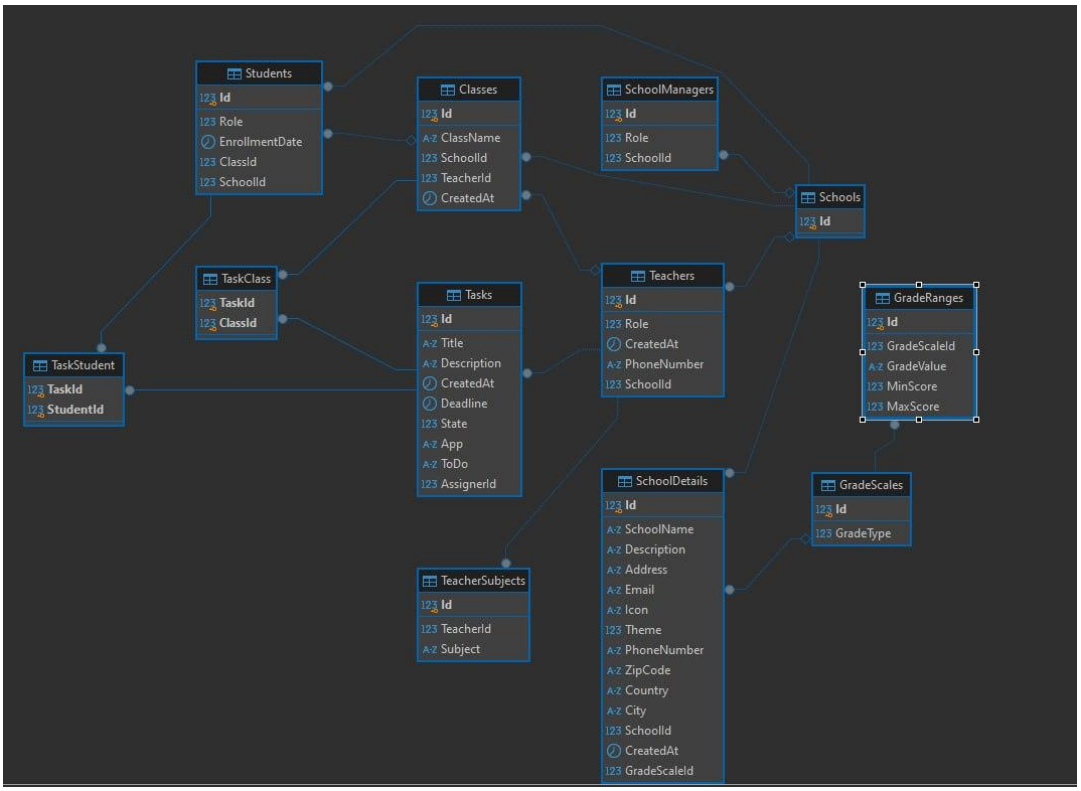


Рисунок 2.5 – Архітектура моделі школи в БД

Для реалізації автоматизованої системи управління навчальним процесом обрано мікросервісну архітектуру. Це сучасний підхід до побудови програмного забезпечення, який розділяє систему на незалежні компоненти (мікросервіси), кожен з яких виконує окрему задачу. Вище на рисунку 2.5 можна зображена структура, яка демонструє мікросервісний підхід до організації бази даних, що дозволяє легко масштабувати систему.

1. Переваги мікросервісної архітектури для проєкту:

- Гнучкість у розробці: кожен мікросервіс може розроблятися і вдосконалюватися незалежно від інших.
- Масштабованість: мікросервіси масштабуються окремо, що дозволяє ефективно використовувати ресурси для найнагальніших задач.
- Надійність: збій одного мікросервісу не впливає на роботу інших, що підвищує загальну стійкість системи.

- Легка інтеграція з іншими системами: за допомогою API мікросервіси взаємодіють як між собою, так і з іншими освітніми платформами.

2. Організація роботи з PostgreSQL у мікросервісній архітектурі. Кожен мікросервіс взаємодіє з PostgreSQL через API або ORM (наприклад, Entity Framework). Це забезпечує ізольований доступ до даних для кожного компонента системи, підвищуючи безпеку й контрольованість операцій.

3. Зберігання даних у PostgreSQL:

У базі даних зберігається інформація про:

- навчальні заклади (школи, класи, групи);
- учнів і викладачів;
- навчальні програми та завдання;
- ліцензії та права доступу.

4. Переваги хмарного розгортання PostgreSQL на AWS. Завдяки розгортанню в хмарній інфраструктурі AWS, система отримує:

- Автоматичне резервне копіювання: гарантія збереження даних у разі збою.
- Глобальна доступність: забезпечення безперервного доступу до системи з будь-якого місця.
- Можливість динамічного масштабування: адаптація до збільшення кількості користувачів.
- Безпека: використання вбудованих механізмів AWS для захисту даних і шифрування.

Обраний підхід із використанням PostgreSQL та мікросервісної архітектури повністю відповідає сучасним вимогам до автоматизованих систем управління навчальним процесом. Це рішення забезпечує надійність, безпеку, масштабованість та ефективне управління даними, що критично важливо для системи, яка працює з великою кількістю користувачів і складною бізнес-логікою.

2.5 Проектування UML-діаграми класів автоматизованої системи

UML-діаграма класів ілюструє структуру та взаємодію основних компонентів системи, що забезпечують управління навчальним процесом. (рис. 2.6). Діаграма класів використовується для опису взаємозв'язків між різними об'єктами, що дозволяє провести аналіз і створити дизайн програми, зображуючи її у статичному вигляді.

Як одна з найважливіших UML-діаграм, вона включає класи, їх атрибути та зв’язки, які є ключовими елементами. Діаграма класів слугує інструментом для отримання загального уявлення про структуру програми, що сприяє зменшенню часу, необхідного для ознайомлення з системою. Система розділена на класи, які виконують різні функції залежно від ролей і завдань. Основними класами є User, Teacher, Student, SchoolManager, School, Class, і Tasks. Діаграма показує зв’язки між ними, а також їх атрибути й операції.

Опис класів та їх функцій:

1. Клас User:

   Призначення: Базовий клас для всіх користувачів системи.

   Атрибути:

   - id — унікальний ідентифікатор користувача.
   - name — ім’я користувача.
   - email — електронна адреса.
   - role — роль користувача (вчитель, студент, менеджер школи).

   Операції:

   - Авторизація користувача.
   - Вихід із системи.
   - Перегляд і редагування профілю.

   Спадкування: Є базовим класом для підкласів Teacher, Student та SchoolManager.

2. Клас Teacher:

   Призначення: Відповідає за управління класами та завданнями.

   Атрибути:

   - id, name, createdDate — стандартні атрибути.

- classIds — список класів, якими керує вчитель.
- schoolId — ідентифікатор школи, де працює вчитель.

Операції:

- Перегляд списку студентів у класі.
- Призначення завдань студентам.
- Додавання, редагування або видалення студентів у класі.

3. Клас Student:

Призначення: Представляє студентів, які є учасниками навчального процесу.

Атрибути:

- id, name, createdDate, classId, schoolId — стандартні атрибути.

- Операції:

- Перегляд завдань, призначених учителем.
- Перегляд своїх оцінок.

4. Клас SchoolManager:

Призначення: Керує всіма процесами в школі.

Атрибути:

- schoolId — ідентифікатор школи.

Операції:

- Додавання й редагування вчителів.
- Призначення класів вчителям.
- Перегляд списків студентів.
- Додавання нових студентів у школу та редагування їх даних.
- Видалення студентів зі школи.

5. Клас School:

Призначення: Відображає дані про школу.

Атрибути:

- schoolId — унікальний ідентифікатор школи.
- schoolName — назва школи.
- location — розташування школи.

- theme, icon — параметри візуального оформлення.

6. Клас Class:

Призначення: Представляє класи в школі.

Атрибути:

- id, name — унікальний ідентифікатор та назва класу.
- teacherId — ідентифікатор вчителя, який керує класом.
- studentsIds — список студентів, які належать до класу.

7. Клас Tasks:

Призначення: Зберігає завдання, призначені студентам.

Атрибути:

- taskId — унікальний ідентифікатор завдання.
- description — опис завдання.
- deadline — строк виконання завдання.

Зв’язки між класами:

- User і підкласи: Клас User є базовим для Teacher, Student, і SchoolManager.
- SchoolManager — School: Менеджер школи має доступ до школи, яку він керує.
- Teacher — Class — Student: Вчитель пов’язаний із класами, а класи — зі студентами.
- Student — Tasks: Завдання пов’язані з конкретними студентами.

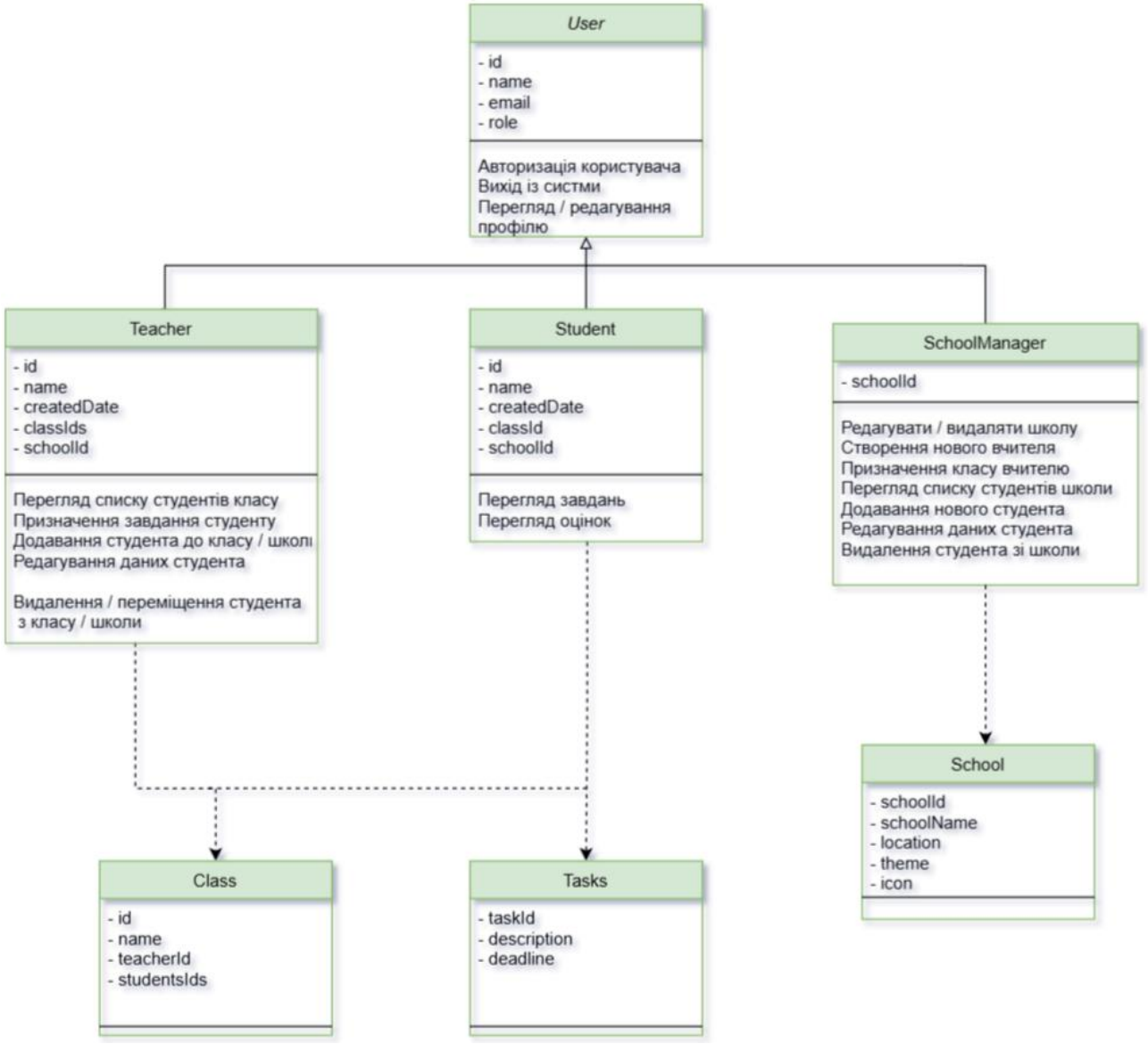


Рисунок 2.6 – UML Діаграма класів

Ця UML-діаграма відображає ключові компоненти автоматизованої системи управління навчальним процесом школи та їхню взаємодію, що дозволяє забезпечити ефективну реалізацію основних функцій.

2.5 Вибір мови програмування та допоміжних засобів

Для розробки клієнтської частини системи управління навчальним процесом було обрано JavaScript, одну з найбільш популярних мов програмування для

створення динамічних веб-додатків. Вибір цієї мови пояснюється її універсальністю, широким інструментарієм і активною підтримкою спільноти розробників.

Крім JavaScript, використовувалися сучасні бібліотеки та фреймворки, які дозволяють суттєво спростити розробку, забезпечити високу продуктивність і створити інтуїтивно зрозумілий користувацький інтерфейс.

Для front end частини:

• React – популярна бібліотека JavaScript для створення користувацьких інтерфейсів.

• Redux – бібліотека для управління станом додатків, яка часто використовується разом з React.

• Ant Design (AntD) – набір високоякісних компонентів для створення інтерфейсів користувача.

Для back end розробки:

• Мікросервісна архітектура на базі .NET – використання .NET для розробки бекенд-частини дозволяє створити масштабовану, незалежну та гнучку систему, де кожен мікросервіс відповідає за окрему функцію (наприклад, управління учнями, планування уроків, відстеження відвідуваності тощо). Це підхід забезпечує модульність і легку підтримку, а також дозволяє масштабувати окремі компоненти системи залежно від навантаження.

• PostgreSQL – потужна реляційна база даних з відкритим кодом, яка забезпечує надійне зберігання та управління даними. PostgreSQL є хорошим вибором для систем із складними запитами та великими обсягами даних, що дозволяє ефективно працювати з базою учнів, персоналу, розкладів та звітів.

2.5.1 HTML

HTML (Hyper Text Markup Language) — це код, який використовується для структурування веб-сторінки та її вмісту . Наприклад, вміст можна структурувати в межах набору абзаців, списку маркованих пунктів або використовувати зображення та таблиці даних. [28]

HTML є похідною мовою від SGML, успадковуючи концепцію структурної розмітки тексту та визначення типу документа. Хоча HTML не є мовою програмування, а скоріше штучною мовою розмітки, вона разом із каскадними таблицями стилів (CSS) та вбудованими скриптами є ключовою технологією для створення веб-сторінок.

Основними можливостями використання HTML є:

- створення структурованих документів шляхом позначення елементів тексту, таких як заголовки, абзаци, списки, таблиці, цитати та інші елементи;
- доступ до інформації через гіперпосилання у всесвітній мережі;
- розробка інтерактивних форм для збору та обробки даних користувача;
- інтеграція мультимедійних елементів, таких як зображення, аудіо, відео та інші об’єкти, у текст.

2.5.2 CSS

Cascading Style Sheets (CSS) — це мова стилів, яка використовується для опису вигляду і форматування документів, написаних мовами розмітки, такими як HTML або XML. CSS визначає, як елементи вебсторінки повинні виглядати на екрані, папері чи інших медіа. Основною метою CSS є відокремлення структури документа від його презентаційного вигляду, що сприяє кращій організації коду і полегшує підтримку вебсайту.[33]

Специфікації CSS розробляються Консорціумом World Wide Web (W3C). CSS складається з різних рівнів і профілів, де кожен новий рівень базується на попередньому, додаючи нові можливості або покращуючи існуючі. Ці рівні позначаються як CSS1, CSS2 та CSS3. Профілі CSS являють собою колекцію правил одного або кількох рівнів, адаптованих для конкретних пристроїв або середовищ. Наприклад, існують спеціалізовані профілі CSS для друкованих видань, мобільних пристроїв та інших типів інтерфейсів. CSS є основним інструментом для створення естетичного вигляду вебсайтів.

2.5.3 Методологія BEM

BEM — це підхід до розробки інтерфейсів, який базується на розділенні коду на окремі компоненти. Методологія дозволяє структурувати код так, щоб забезпечити легкість розробки складних інтерфейсів та можливість повторного використання компонентів, зменшуючи необхідність дублювання коду.[30]

Основна ідея BEM полягає в поділі інтерфейсу на незалежні блоки, які можуть існувати окремо або бути вкладені один в одного. Кожен блок є самостійним елементом сторінки і може бути багаторазово використаний. У HTML це реалізується за допомогою класів. Наприклад, основний блок може містити такі компоненти, як логотип, форму пошуку та блок авторизації (рис. 2.7).

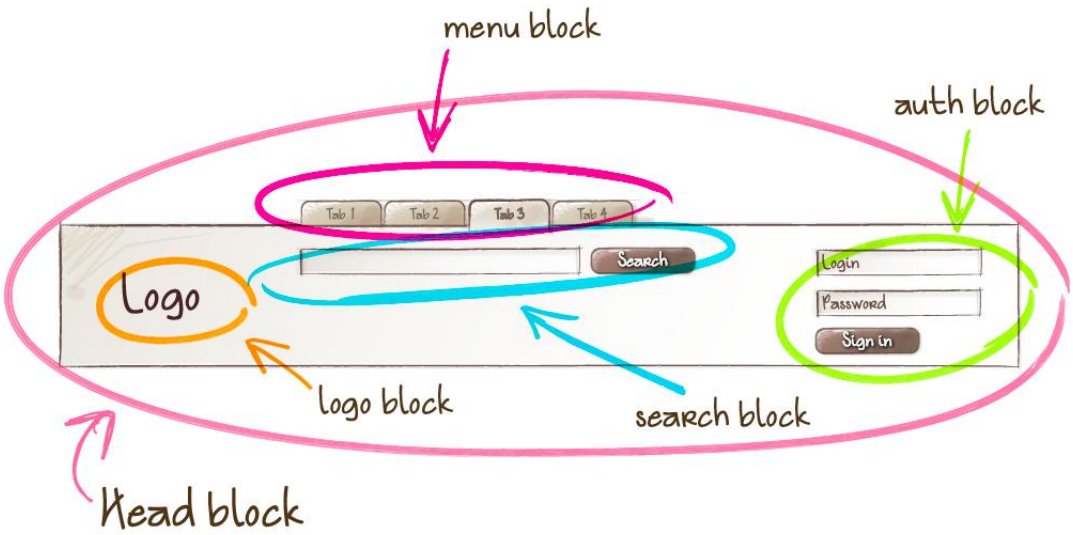


Рисунок 2.7 - Структура BEM.[30]

Блоки можна легко переміщувати як на одній сторінці, так і між різними сторінками чи проєктами. Завдяки незалежній реалізації блоків, вони зберігають свою функціональність та стиль незалежно від місця розташування на сторінці.

Рисунок 2.8 - Реструктуризація в методології BEM.[30]

Інтерфейс може містити декілька однотипних екземплярів одного й того ж блоку.

Це означає, що один і той самий блок можна використовувати на сторінці стільки разів, скільки потрібно, не змінюючи його внутрішню структуру чи стиль. Завдяки цьому забезпечується зручність розробки, зменшується дублювання коду та зберігається послідовний стиль інтерфейсу, незалежно від кількості повторних використань блоку.(рис 2.8)

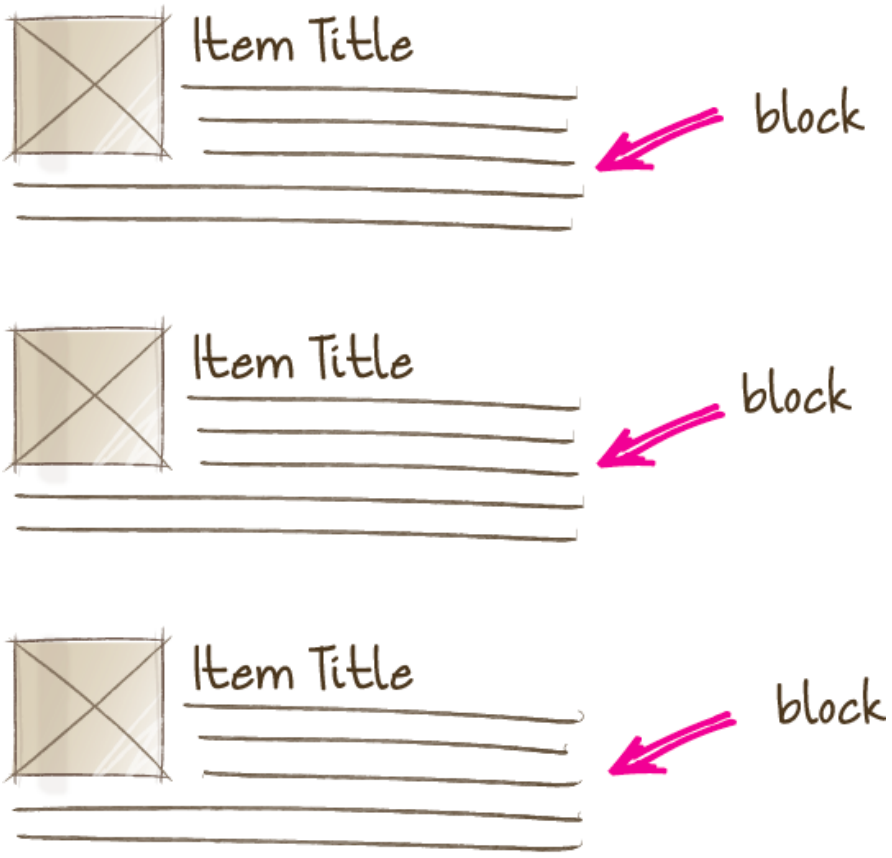


Рисунок 2.9 - Перевикористання екземплярів одного блоку в методології BEM.[30]

Модифікатор — це BEM-сутність, яка визначає зовнішній вигляд та поведінку блоку або елемента.

Використання модифікаторів не є обов'язковим, але вони допомагають змінювати вигляд та функціональність компонентів без зміни їх основної структури. Модифікатори за своєю суттю схожі на HTML-атрибути, оскільки вони надають додаткові властивості, які можуть змінювати зовнішній вигляд чи поведінку блоку.

2.5.4 Мова програмування TypeScript

TypeScript — це мова програмування, розроблена компанією Microsoft, яка є розширенням JavaScript і додає підтримку статичної типізації.[29] Основною метою TypeScript є покращення продуктивності та надійності коду, забезпечення кращої перевірки помилок на етапі розробки та полегшення масштабування великих

проєктів. TypeScript компілюється у звичайний JavaScript, що дозволяє використовувати його в будь-яких середовищах, де підтримується JavaScript. Основні особливості TypeScript:

1. TypeScript дозволяє визначати типи змінних, функцій та об'єктів, що допомагає виявляти помилки ще до виконання коду. Типи можна вказувати явно, або вони можуть бути визначені автоматично за допомогою інференції типів.
2. TypeScript дозволяє створювати інтерфейси та власні типи, що спрощує роботу з об'єктами та масивами.
3. TypeScript підтримує повноцінний синтаксис класів з такими поняттями, як спадкування, інкапсуляція та поліморфізм. Це спрощує написання об'єктно-орієнтованого коду та забезпечує кращу організацію проєкту.
4. TypeScript підтримує модулі, що дозволяють розділяти код на окремі файли та підключати їх за допомогою import/export.

Використання TypeScript в проєкті дозволить краще виявлення помилок, адже завдяки статичній типізації багато помилок можна виявити на етапі написання коду, що зменшує кількість багів під час виконання. Автодоповнення, рефакторинг та документація в середовищах розробки значно покращуються завдяки TypeScript.

2.5.5 React

React (також відомий як React.js) - це відкрита JavaScript-бібліотека для створення інтерфейсів користувача (UI). Її мета - спростити розробку динамічних UI, які швидко оновлюються без перезавантаження сторінки.[11]

React використовує декларативний підхід до програмування, де ви описуєте, як має виглядати інтерфейс користувача, а не як його змінювати. Це робиться за допомогою спеціальних компонентів - будівельних блоків UI, які можна повторно використовувати та складати для створення складних інтерфейсів.

Переваги використання React:

1. Простота: React має простий і зрозумілий синтаксис, що робить його легким для вивчення та використання.

2. Ефективність: React використовує віртуальне DOM-дерево для представлення UI, що робить його дуже ефективним у оновленні лише тих частин інтерфейсу, які змінилися.

3. Гнучкість: React можна використовувати для створення широкого спектру UI, від простих веб-сторінок до складних односторінкових застосунків (SPA).

4. Масштабованість: React добре масштабується, що робить його придатним для створення великих і складних застосунків.

5. Спільнота: React має велику та активну спільноту розробників, які створюють інструменти, бібліотеки та ресурси, які допоможуть вам у розробці.

Деякі з популярних випадків використання React:

Односторінкові веб-застосунки (SPA): React часто використовується для створення SPA, які завантажують лише одну HTML-сторінку та динамічно оновлюють її вміст за допомогою JavaScript.

Веб-інтерфейси: React можна використовувати для створення будь-якого типу веб-інтерфейсу, від простих кнопок і форм до складних карт і графіків.

Мобільні застосунки: React Native - це фреймворк, який дозволяє використовувати React для створення мобільних застосунків для iOS і Android.

React ґрунтується на концепції компонентів, які є повторно використовуваними будівельними блоками UI. Існує два основних типи компонентів: функціональні та класові. Функціональні компоненти простіші та описують, що виводити, як функцію. Класові компоненти мають стан та методи життєвого циклу, що робить їх більш підходящими для складнішої логіки. Компоненти можна складати один в один для створення ієрархії UI. Це сприяє модульності та повторному використанню коду, роблячи код чистим та організованим.

React використовує синтаксис JSX, який розширює JavaScript, дозволяючи писати HTML-подібний код безпосередньо в JavaScript. JSX робить код більш читабельним та описовим, особливо при описі UI. Він перетворюється в звичайний JavaScript, який може зрозуміти браузер.

React використовує віртуальне DOM-дерево для представлення UI. Це абстракція реального DOM в браузері. При внесенні змін до компонентів React

порівнює віртуальне DOM-дерево з реальним DOM і оновлює лише ті частини UI, які дійсно змінилися. Це робить React дуже ефективним та покращує продуктивність.

React має багату екосистему бібліотек, інструментів та фреймворків, які розширюють його функціональні можливості. Це робить React дуже універсальним та придатним для створення широкого спектру UI. Деякі популярні бібліотеки включають Redux, React Router, Material UI та Next.js.

Одностороння передача даних:

У React дані передаються від батьківських компонентів до дочірніх через властивості (props). Такий підхід називається односпрямованою передачею даних. Це забезпечує більш передбачувану логіку програми та полегшує відстеження змін у стані компонентів. Основні переваги односпрямованої передачі даних включають:

- Передбачуваність: Компоненти чітко знають, звідки надходять дані, що спрощує розуміння та налагодження коду.
- Контрольованість: Батьківські компоненти мають повний контроль над станом своїх дочірніх компонентів, що забезпечує більш контрольовану архітектуру застосунку.

Основою React є компонентний підхід до розробки інтерфейсів. Компоненти – це незалежні, повторно використовувані частини інтерфейсу, які можуть включати в себе інші компоненти або HTML-розмітку та логіку. Компоненти дозволяють розробникам створювати складні інтерфейси, розбиваючи їх на більш дрібні, керовані частини. Основні переваги компонентного підходу включають:

- Модульність: Компоненти можуть бути створені та протестовані окремо, що полегшує розробку і тестування.
- Повторне використання: Компоненти можуть бути використані в різних частинах застосунку, що сприяє зменшенню кількості дубльованого коду.
- Чітка структура: Компонентний підхід сприяє організованій та зрозумілій структурі проекту, що полегшує його підтримку та розширення.

2.5.6 Angular

Angular – це популярний фреймворк для розробки веб-застосунків, розроблений компанією Google. Він вперше був випущений у 2010 році як AngularJS, а у 2016 році був кардинально перероблений та перевипущений під назвою Angular (без "JS"). Angular пропонує потужні інструменти та функції для створення складних і масштабованих веб-застосунків. Нижче детально розглянемо основні особливості та переваги Angular, які роблять його потужним інструментом для розробки веб-застосунків.[18]

Архітектура на основі компонентів

Angular, як і React, використовує компонентний підхід для розробки інтерфейсів. Компоненти в Angular є незалежними, повторно використовуваними частинами інтерфейсу, що сприяє модульності та зручності у підтримці коду. Основні переваги компонентної архітектури в Angular включають:

1. Модульність: Компоненти можуть бути створені та протестовані окремо, що полегшує розробку і тестування.
2. Повторне використання: Компоненти можуть бути використані в різних частинах застосунку, що сприяє зменшенню кількості дубльованого коду.
3. Чітка структура: Компонентний підхід сприяє організованій та зрозумілій структурі проекту, що полегшує його підтримку та розширення.

Angular використовує декларативний підхід до створення шаблонів, що забезпечує більш читабельний та зрозумілий код. HTML-шаблони в Angular розширені спеціальними директивами, які дозволяють інтегрувати логіку безпосередньо в розмітку. Основні переваги декларативних шаблонів включають:

1. Читабельність: Декларативні шаблони полегшують розуміння структури та логіки інтерфейсу.

2. Спрощене управління станом: Використання директив та двостороннього зв'язування даних (two-way data binding) забезпечує простий та зрозумілий спосіб управління станом компонентів.

Однією з ключових особливостей Angular є двостороннє зв'язування даних. Це означає, що зміни в моделі даних автоматично відображаються в інтерфейсі

користувача, і навпаки, зміни в інтерфейсі автоматично оновлюють модель даних. Основні переваги двостороннього зв'язування даних включають:

- Простота: Двостороннє зв'язування даних спрощує синхронізацію між моделлю даних та інтерфейсом, зменшуючи обсяг коду, необхідного для відображення змін.
- Зручність: Цей підхід полегшує роботу з формами та іншими елементами інтерфейсу, які часто змінюються.

Angular має вбудовану систему dependency injection (DI), яка забезпечує інверсію управління залежностями та полегшує тестування та повторне використання коду. Основні переваги DI в Angular включають:

- Модульність: DI сприяє модульності коду, дозволяючи легко замінювати або оновлювати залежності без зміни коду, який їх використовує.
- Тестування: DI полегшує створення тестів, оскільки залежності можуть бути легко підмінені моками або стубами.

Angular інтегрується з бібліотекою RxJS (Reactive Extensions for JavaScript), яка надає потужні інструменти для роботи з асинхронними операціями та потоками даних. RxJS дозволяє використовувати реактивне програмування для створення більш ефективних та масштабованих додатків. Основні переваги RxJS в Angular включають:

- Асинхронність: RxJS забезпечує зручні методи для роботи з асинхронними операціями, такими як запити до серверу, таймери та обробка подій.
- Композиційність: Реактивне програмування дозволяє легко комбінувати та трансформувати потоки даних, що робить код більш гнучким та зрозумілим.

Angular має потужний інструмент командного рядка (CLI), який спрощує створення, розробку та деплоймент застосунків. Angular CLI надає команди для генерації компонентів, сервісів, модулів та інших артефактів, а також для запуску тестів та збирання проекту. Основні переваги Angular CLI включають:

Швидкий старт: Angular CLI дозволяє швидко розпочати новий проект з мінімальною конфігурацією.

Автоматизація: CLI автоматизує багато рутинних задач, що знижує ймовірність помилок та підвищує продуктивність розробників.

Тестування та деплоймент: CLI надає інструменти для запуску тестів та деплойменту застосунків, що спрощує процес підтримки та оновлення проекту.

Екосистема та підтримка

Angular має розвинену екосистему та велику спільноту розробників. Існує безліч додаткових бібліотек та інструментів, які інтегруються з Angular, що дозволяє розробникам легко розширювати функціональність своїх додатків. Крім того, велика кількість ресурсів, таких як документація, навчальні матеріали, форуми та блоги, роблять процес навчання та використання Angular більш доступним.

2.4.7 Vue.js

Vue.js – це прогресивний фреймворк для створення користувацьких інтерфейсів. Він був створений Еваном Ю (Evan You) і вперше випущений у 2014 році. Vue.js надає гнучкий та зручний у використанні підхід до розробки веб-застосунків, що робить його популярним серед розробників. У цьому розділі детально розглянемо основні особливості та переваги Vue.js, які роблять його потужним інструментом для розробки веб-застосунків.[17]

Vue.js позиціонується як прогресивний фреймворк, що означає, що його можна легко інтегрувати в існуючі проекти поступово. Це дозволяє використовувати Vue.js як для створення окремих компонентів, так і для побудови повноцінних односторінкових застосунків (SPA). Основні переваги прогресивного підходу включають:

Гнучкість: Можливість інтеграції Vue.js з будь-яким проектом, незалежно від його масштабу чи технологічного стеку.

Поступове прийняття: Можливість почати використовувати Vue.js для окремих частин проекту, а потім поступово розширювати його застосування.

Vue.js, як і React та Angular, використовує компонентний підхід для створення користувацьких інтерфейсів. Компоненти в Vue.js є незалежними та повторно

використовуваними частинами інтерфейсу, які можуть включати HTML, CSS та JavaScript. Основні переваги компонентного підходу включають:

Модульність: Компоненти можуть бути створені та протестовані окремо, що полегшує розробку і тестування.

Повторне використання: Компоненти можуть бути використані в різних частинах застосунку, що сприяє зменшенню кількості дубльованого коду.

Чітка структура: Компонентний підхід сприяє організованій та зрозумілій структурі проекту, що полегшує його підтримку та розширення.

Vue.js використовує декларативний синтаксис для опису шаблонів, що дозволяє розробникам легко створювати інтерфейси. Шаблони в Vue.js базуються на HTML та розширені спеціальними директивами, які додають динамічність та інтерактивність. Основні переваги розширюваних шаблонів включають:

- Читабельність: Декларативні шаблони полегшують розуміння структури та логіки інтерфейсу.
- Спрощене управління станом: Використання директив, таких як v-bind, v-model та v-if, забезпечує простий та зрозумілий спосіб управління станом компонентів.

Vue.js має вбудовану реактивну систему, яка автоматично оновлює інтерфейс користувача при зміні даних. Це досягається через двостороннє зв'язування даних (two-way data binding) та спостерігачі (watchers). Основні переваги реактивної системи включають:

- Простота: Реактивна система забезпечує автоматичне оновлення інтерфейсу при зміні даних, що зменшує обсяг коду, необхідного для відображення змін.
- Зручність: Цей підхід полегшує роботу з формами та іншими елементами інтерфейсу, які часто змінюються.

Vue Router і Vuex

Vue.js має офіційні бібліотеки для управління маршрутизацією та станом застосунку:

Vue Router: Бібліотека для управління маршрутизацією в односторінкових застосунках. Вона дозволяє створювати динамічні маршрути, які відповідають за завантаження різних компонентів в залежності від URL.

Vuex: Бібліотека для централізованого управління станом застосунку. Vuex забезпечує єдиний спосіб управління станом, що полегшує розробку та відлагодження складних застосунків.

Vue.js має потужний інструмент командного рядка (CLI), який спрощує створення, розробку та деплоймент застосунків. Vue CLI надає команди для генерації проектів, компонентів та інших артефактів, а також для запуску тестів та збирання проекту. Основні переваги Vue CLI включають:

1. Швидкий старт: Vue CLI дозволяє швидко розпочати новий проект з мінімальною конфігурацією.
2. Автоматизація: CLI автоматизує багато рутинних задач, що знижує ймовірність помилок та підвищує продуктивність розробників.
3. Тестування та деплоймент: CLI надає інструменти для запуску тестів та деплойменту застосунків, що спрощує процес підтримки та оновлення проекту.

Vue.js має розвинену екосистему та велику спільноту розробників. Існує безліч додаткових бібліотек та інструментів, які інтегруються з Vue.js, що дозволяє розробникам легко розширювати функціональність своїх додатків. Крім того, велика кількість ресурсів, таких як документація, навчальні матеріали, форуми та блоги, роблять процес навчання та використання Vue.js більш доступним.

Порівняю основні характеристики даних фреймворків, у вигляді таблиці 2.4, використовуючи дані розглянуті вище.

Таблиця 2.4 - Порівняння характеристик фреймворків.

Характеристика	**React**	**Angular**	**Vue.js**
Архітектура	Бібліотека	Фреймворк	Прогресивний фреймворк

Компонентний підхід	Так	Так	Так
JSX	Так	Ні	Ні
Virtual DOM	Так	Ні	Так
Двостороннє зв'язування даних	Обмежене	Так	Так
Масштабованість	Висока	Висока	Висока
Продуктивність	Висока	Висока	Висока
Екосистема та підтримка	Широка	Широка	Широка
Система управління станом	Redux	NgRx	Vuex

Переваги React:

1. Гнучкість та модульність:

React є бібліотекою, а не повноцінним фреймворком, що забезпечує гнучкість у виборі додаткових інструментів та бібліотек. Це дозволяє розробникам вибирати лише необхідні компоненти для проекту, уникаючи перевантаження зайвими функціями.

2. Висока продуктивність:

Використання Virtual DOM в React забезпечує високу продуктивність шляхом мінімізації прямих маніпуляцій з реальним DOM. Це особливо важливо для складних та динамічних інтерфейсів.

3. JSX:

React використовує JSX, який дозволяє розробникам писати HTML-подібний код безпосередньо у JavaScript. Це забезпечує кращу читабельність та зрозумілість коду, полегшуючи процес розробки.

4. Активна спільнота та підтримка:

React має широку та активну спільноту розробників, що забезпечує велику кількість доступних ресурсів, навчальних матеріалів та бібліотек. Це значно спрощує процес навчання та вирішення проблем, що можуть виникнути під час розробки.

5. Масштабованість:

React добре підходить для створення як малих, так і великих додатків. Компонентний підхід та можливість використання Redux для управління станом забезпечують високий рівень масштабованості проекту.

6. Підтримка Facebook:

React був розроблений і підтримується компанією Facebook, що гарантує його постійний розвиток та підтримку на високому рівні.

React є потужним та гнучким інструментом для розробки сучасних веб-застосунків. Його висока продуктивність, зручність у використанні, гнучкість та підтримка широкої спільноти роблять його ідеальним вибором для розробки автоматизованої системи управління навчальним процесом школи. Вибір React забезпечить ефективність, зручність у розробці та підтримці, а також масштабованість проекту, що є ключовими факторами для успішного впровадження та використання системи.

- Завдяки використанню JSX та компонентного підходу, розробка інтерфейсу користувача стає простою та інтуїтивно зрозумілою, що дозволяє швидко створювати і підтримувати складні інтерфейси.
- Використання Virtual DOM забезпечує високу продуктивність та швидкий відгук інтерфейсу, що важливо для інтерактивних освітніх платформ з великою кількістю користувачів.

- Завдяки Redux та іншим інструментам управління станом, React дозволяє легко керувати складними станами додатка, що є критично важливим для автоматизованої системи управління навчальним процесом.
- Велика кількість доступних ресурсів, документації та підтримки з боку спільноти робить React ідеальним вибором для розробки складних проектів, забезпечуючи легкість у вирішенні можливих проблем та швидкий розвиток проекту.

2.4.8 Бібліотека Redux

Redux — це популярна бібліотека для управління станом у JavaScript-додатках, розроблена для створення передбачуваних програм із чітко організованою архітектурою. Вона часто використовується з React, але може працювати і з іншими бібліотеками або фреймворками. Основна мета Redux — забезпечити стабільне управління станом додатка, спростити тестування та покращити досвід розробки, зокрема завдяки можливості налагодження в реальному часі та відстеження змін у стані.[34]

Весь стан додатку зберігається в єдиному об’єкті, званому store. Це спрощує доступ до даних та зменшує кількість помилок, оскільки весь стан централізований та керований через єдине джерело. Завдяки цьому підходу можна легко серіалізувати стан для передачі між сервером і клієнтом або зберігати його для подальшого відтворення. Стан не змінюється безпосередньо. Єдиний спосіб оновити стан — це надсилати дії (actions), які є об'єктами з описом змін. Такий підхід унеможливлює непередбачувані зміни, оскільки дії централізовано обробляються у строгому порядку, зменшуючи ймовірність виникнення помилок через асинхронні операції.

Зміни стану визначаються чистими функціями, званими редукторами (reducers). Редуктори отримують попередній стан і дію, а повертають новий стан, не змінюючи оригінальних даних. Це допомагає забезпечити передбачуваність поведінки додатка та спрощує тестування, оскільки редуктори завжди дають однаковий результат при однакових вхідних даних.

Переваги використання Redux:

1. Завдяки єдиному джерелу стану та використанню чистих функцій, поведінка додатку стає передбачуваною, що спрощує процес розробки та тестування.
2. Всі дані додатку зберігаються в одному місці, що зменшує потребу в передачі стану через компоненти та спрощує роботу з даними.
3. Redux забезпечує чудовий досвід налагодження, зокрема можливість "подорожі в часі", коли можна відтворити або відкотити зміни стану, що значно полегшує пошук і виправлення помилок.

2.5.9 ANT design

Ant Design (AntD) — це популярна бібліотека компонентів для створення інтерфейсів користувача на основі React. Вона широко використовується для розробки веб-застосунків завдяки своїй елегантності, простоті у використанні та великій кількості готових рішень.[32]

Включає широкий набір готових до використання компонентів, таких як кнопки, форми, таблиці, модальні вікна, навігаційні панелі тощо. Забезпечує єдину стилістичну концепцію для всіх елементів, що дозволяє створювати гармонійні та професійні інтерфейси. Легко налаштовується під потреби проекту, включаючи зміну теми, кольорів і поведінки компонентів. Підтримка багатомовності для забезпечення зручного використання в різних країнах.

Завдяки готовим компонентам значно скорочується час на розробку інтерфейсів. Бібліотека відповідає сучасним принципам UX/UI-дизайну. Всі компоненти повністю сумісні з React, що робить її зручною для інтеграції з іншими бібліотеками або фреймворками.

2.5.10 Styled Components

Styled Components — це бібліотека для CSS-in-JS, яка дозволяє створювати стилі безпосередньо у файлах JavaScript, використовуючи сучасні можливості JavaScript та React. Вона дозволяє писати стилі компонентів у синтаксисі, схожому на звичайний CSS, але в рамках JavaScript-коду.[31]

Стилі прив'язані до конкретного компонента, що виключає конфлікти між CSS-класами. Можна змінювати стилі компонентів динамічно на основі пропсів або стану (state). Підтримує тему через ThemeProvider, що дозволяє легко керувати загальними стилями для всього застосунку. Усе стилювання можна зберігати безпосередньо в компонентах, що зменшує залежність від зовнішніх CSS-файлів.

Використовує шаблонні літерали JavaScript для стилізації. Стилі застосовуються тільки до конкретного компонента, що виключає випадкове перезаписування. Легко змінювати стилі за допомогою пропсів або станів. Зручне керування темами для створення універсального дизайну. Підходить як для маленьких, так і для великих проектів.

Styled Components часто використовуються разом із бібліотеками, як-от Ant Design, для кастомізації стилів готових компонентів, забезпечуючи гнучкість і сучасний підхід до розробки.

2.5.11 Vite

Vite – це сучасний інструмент для збірки і розробки JavaScript-додатків, який надає розробникам швидкий та ефективний робочий процес. Назва "Vite" походить від французького слова, що означає "швидко", і це відображає його основну мету – забезпечити високу швидкість розробки, особливо для проєктів на базі фреймворків, таких як React, Vue або Svelte. Завдяки використанню сучасних можливостей браузерів, Vite дозволяє майже миттєво запускати локальний сервер розробки. Це досягається за рахунок динамічного завантаження модулів через ES-модулі (ESM) замість традиційної попередньої компіляції всього проєкту. Функція HMR (Hot Module Replacement) в Vite забезпечує миттєве оновлення змін у браузері без необхідності повного перезавантаження сторінки, що покращує швидкість розробки та зручність роботи. Інструмент має простий у використанні базовий функціонал, але також підтримує налаштування за допомогою плагінів, якщо потрібна більш гнучка адаптація.[37]

Завдяки сучасним технологіям (esbuild, HMR, Rollup) Vite значно перевершує традиційні інструменти, такі як Webpack, у швидкості розробки та збірки.

Використовує останні можливості JavaScript і підтримує TypeScript, JSX та CSS Modules. Мінімальні налаштування для швидкого старту, що робить його ідеальним для невеликих і великих проєктів.

2.6 Вибір та налаштування бази даних у хмарному середовищі

Так, як база даних PostgreSQL, потужна, надійна та відкрита реляційна система управління базами даних (СУБД), яка підтримує SQL (Structured Query Language) і безліч сучасних функцій для роботи з даними. Вона широко використовується для створення веб-додатків, систем управління контентом, аналітики та багатьох інших програм. Найкращим варіантом на хмарному середовищі AWS для цієї бази даних є сервіс Amazon RDS.[35]

Керований сервіс баз даних від Amazon Web Services, що дозволяє легко створювати, налаштовувати та масштабувати реляційні бази даних у хмарі. Amazon RDS усуває необхідність ручного управління серверами баз даних, включаючи завдання створення резервних копій, оновлення програмного забезпечення, моніторингу продуктивності та масштабування.

Однією з переваг є підтримка популярних систем управління базами даних (СУБД), яка включає в себе PostgreSQL. Підтримка Multi-AZ Deployment, що дозволяє створювати резервні копії баз даних у кількох зонах доступності для забезпечення безперебійної роботи. Автоматичне перемикання на резервну копію у випадку збою. Дані можуть бути зашифровані як під час зберігання (AES-256), так і під час передачі (SSL), що робить данні безпечними. Можливість ізоляції бази даних у віртуальній приватній мережі. Основною з переваг є оптимізація продуктивності, використання автоматичного моніторингу запитів для виявлення вузьких місць у базі даних. Підтримка Provisioned IOPS (I/O Operations Per Second) для застосунків із високим навантаженням.

У системі автоматизації навчального процесу Amazon RDS з PostgreSQL стане оптимальним вибором для управління великою кількістю структурованих даних, таких як:

- інформація про студентів, викладачів та класи.
- дані про завдання, оцінки та їх шкалу.
- ліцензії на програмне забезпечення та деталі організацій.

3 РОЗРОБКА ТА ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ ШКОЛИ

У цьому розділі буде проведено дослідження наявних рішень, які застосовуються для реалізації автоматизованої системи управління навчальним процесом. Описано аргументи на користь вибору мови програмування TypeScript та її додаткових інструментів для створення та перевірки функціональності програмного продукту. Деталізовано алгоритм роботи основної програми, розробленої з використанням фреймворку React, а також надано характеристику функцій, створених для даної системи. Представлено огляд розробленої системи із зазначенням її ключових можливостей і функціоналу. Проведено аналіз отриманих результатів тестування та їх впливу на подальше вдосконалення програмного забезпечення.

3.1 Огляд сучасних підходів до реалізації функціоналу та розробки візуальної складової.

Для розробки програмного продукту були обрані мови програмування, які забезпечують його кросплатформеність, тобто можливість роботи на різних операційних системах, таких як Windows, Linux, Mac OS.

Для розробки візуальної частини обрано TypeScript, яка є розширенням JavaScript і на сьогоднішній день вважається однією з найбільш популярних та підтримуваних мов. Вона активно розвивається, пропонуючи розробникам удосконалений синтаксис, включаючи статичну типізацію, що значно спрощує розробку масштабованих і надійних застосунків. Завдяки відкритому вихідному коду TypeScript розробники мають можливість створювати нові інструменти та модифікації. Мова також підтримує використання численних бібліотек, серед яких ReactJS залишається однією з найбільш поширених і затребуваних на сьогодні.

ReactJS — це популярна бібліотека TypeScript/JavaScript, призначена для створення динамічних веб-застосунків із використанням компонентного підходу.

Вона забезпечує швидкодію завдяки віртуальному DOM, підтримує інтеграцію з інструментами для управління станом (наприклад, Redux), і пропонує гнучкість у розробці через широку екосистему додаткових бібліотек. Завдяки сумісності з TypeScript, ReactJS сприяє написанню надійного та масштабованого коду, що робить її одним із провідних інструментів для створення сучасних інтерфейсів.[10]

Для забезпечення зручності та швидкості стилізації веб-застосунку було обрано бібліотеку Ant Design (ANTD).[31] Ця бібліотека є одним із провідних рішень у сфері веб-розробки завдяки своїй гнучкості, широкому набору готових компонентів та підтримці адаптивного дизайну. Використання ANTD дозволяє створювати сучасний, привабливий і функціональний інтерфейс користувача, що відповідає стандартам UX/UI-дизайну.

Бібліотека ANTD надає великий вибір елементів інтерфейсу, таких як кнопки, форми, таблиці, модальні вікна, іконки, панелі та інші компоненти, які легко налаштовуються під потреби конкретного проєкту. Окрім цього, ANTD забезпечує повну підтримку адаптивного дизайну, що дозволяє веб-додатку коректно відображатися на пристроях із різними розмірами екранів, включаючи мобільні телефони, планшети та десктопи.

Важливо зазначити, що бібліотека інтегрується з TypeScript і React, що спрощує розробку, підвищує стабільність коду та зменшує кількість можливих помилок під час роботи з компонентами. Гнучка система налаштування стилів, доступна в Ant Design, дозволяє змінювати параметри, такі як кольорова схема, шрифти та інші візуальні характеристики, відповідно до вимог конкретного проєкту. Завдяки цьому ANTD сприяє створенню гармонійного та ефективного інтерфейсу для користувачів.

Для реалізації серверної частини програмної системи було прийнято рішення створити проєкт на основі .NET. Цей вибір обумовлений високою продуктивністю, масштабованістю та широкими можливостями платформи для побудови серверних рішень. Серверна частина слугуватиме центральним компонентом, що забезпечує бізнес-логіку програми, взаємодію з базою даних та обробку запитів від клієнтської сторони.

.NET — це потужна платформа для розробки серверних додатків, надає інструменти для створення веб-сервісів, інтеграції з базами даних та виконання бізнес-логіки, що робить її ідеальним вибором для розробки серверної частини програмних систем. .NET також підтримує різноманітні механізми безпеки та забезпечує гнучкість для подальшого розширення проекту.[20]

Для розміщення серверної частини, фронтової частини та бази даних було вибрано хмарну платформу AWS (Amazon Web Services). Для бази даних використовується Amazon RDS (Relational Database Service)[35], що забезпечує автоматичне керування базами даних, їх масштабування, резервне копіювання та забезпечення безпеки. Це дозволяє забезпечити високу доступність, надійність та ефективність роботи з даними. Система працює в хмарному середовищі AWS, що гарантує високу продуктивність і масштабованість інфраструктури, а також забезпечує зручне управління ресурсами та безпеку даних.

3.2 Розробка програмного забезпечення

3.2.1 Визначення ключових функцій програми

Перед початком розробки слід визначити ключові функції програми, а також додаткові можливості, які покращують взаємодію з додатком та підвищують його зручність для користувача. Система повинна включати такі функції:

- Авторизація користувача. Метою є забезпечити доступ до системи лише для зареєстрованих користувачів. Реалізація авторизації допоможе захистити конфіденційні дані, а також дозволить налаштувати різні рівні доступу залежно від ролі користувача (студент, вчитель, менеджер школи). Технології: реалізується за допомогою токенів (JWT) та безпечного збереження даних про сеанси.
- Меню навігації. Метою є надати інтуїтивний доступ до основних розділів системи (класів, завдань, списків користувачів тощо). Особливість: Меню

адаптується залежно від ролі користувача, показуючи лише ті функції, які доступні для його профілю.

- Налаштування сайту школи (завантаження логотипу, вибір теми). Для забезпечення адміністраторам (менеджерам школи) можливість персоналізувати сайт, зокрема завантаження логотипу школи, вибір візуальної теми та кольорової схеми для підвищення впізнаваності школи.
- Додавання\Редагування вчителів та школярів. Адміністрування списку користувачів школи. Функція дозволяє додавати нових користувачів, редагувати їхні дані або видаляти їх із системи. Дані синхронізуються з базою даних, забезпечуючи актуальність списків учасників навчального процесу.
- Створення\Редагування класів. Забезпечити зручне управління класами, закріплюючи вчителів і учнів за конкретними навчальними групами.
- Надсилання сповіщень на пошту користувачів. Інформувати користувачів про важливі події, наприклад, про нові завдання, зміни в розкладі або оновлення системи.
- Купівля та призначення ліцензій. Забезпечити школам можливість оплачувати ліцензії для доступу до системи та розподіляти їх між користувачами.
- Створення завдань для учнів та їх призначення. Забезпечити вчителям інструменти для створення навчальних завдань із гнучкими налаштуваннями дедлайнів і критеріїв оцінювання

На етапі створення MVP (minimum viable product) необхідно реалізувати базові функції:

1. Авторизація користувачів.
2. Меню навігації.
3. Управління користувачами (додавання/редагування вчителів і учнів).
4. Створення та редагування класів.

Додаткові можливості, такі як персоналізація сайту та інтеграція із платіжними системами, можуть бути реалізовані в наступних етапах розвитку системи.

Реалізація вищезазначених функцій дозволить:

1. Зробити взаємодію з системою інтуїтивною для кожної категорії користувачів.
2. Спрощувати адміністративні процеси для менеджерів шкіл.
3. Покращити організацію навчального процесу для вчителів.
4. Підвищити мотивацію учнів завдяки легкому доступу до завдань і оцінок.

Таким чином, чітке визначення функцій системи сприяє формуванню її загального дизайну, а також дозволяє оптимізувати процес розробки та майбутньої підтримки системи.

3.2.3 Розробка клієнтської та серверної частин програми з використанням сучасних технологій

З самого початку було створено проекти та налаштовано проекти. Для розробки Frontend частини програми було виконано наступні кроки:

- Створення проєкту на основі React і Vite. Для швидкої ініціалізації проєкту обрано інструмент Vite, який забезпечує високу швидкість створення та оптимізації веб-додатків. React було обрано через його популярність, гнучкість і підтримку компонентного підходу, що ідеально підходить для побудови складних інтерфейсів.
- Налаштування Prettier та ESlint. Для забезпечення єдиного стилю написання коду в команді було встановлено та налаштовано інструменти:
  - Prettier: Автоматичне форматування коду відповідно до заданих правил, що підвищує зручність читання та підтримки коду.
  - ESLint: Використовується для аналізу якості коду та виявлення потенційних помилок або порушення стилістики. Це значно полегшує процес розробки та знижує ймовірність помилок.

- Встановлення бібліотеки компонентів ANTD. Для створення сучасного та естетичного дизайну інтерфейсу було використано бібліотеку Ant Design (ANTD). Вона надає великий набір готових UI-компонентів, таких як кнопки, форми, таблиці та модальні вікна, що дозволило суттєво скоротити час розробки. Крім того, була налаштована власна палетка кольорів, щоб адаптувати дизайн до вимог системи, зробити інтерфейс зручним для користувачів та витриманим у єдиному стилі.
- Реалізація підходу BEM. У розробці компонентів використовувалася методика BEM (Block-Element-Modifier), яка забезпечує структурованість та зрозумілість CSS-класів. (рис 3.1) Це дозволило розробити повторно використовувані shared компоненти, такі як:
  - Input: Уніфікований компонент для введення даних.
  - Button: Кастомізовані кнопки з різними стилями та поведінкою.
  - Modal: Модальні вікна для відображення спливаючої інформації або форм.
  - Інші компоненти, які сприяють підвищенню ефективності та зменшенню дублювання коду.

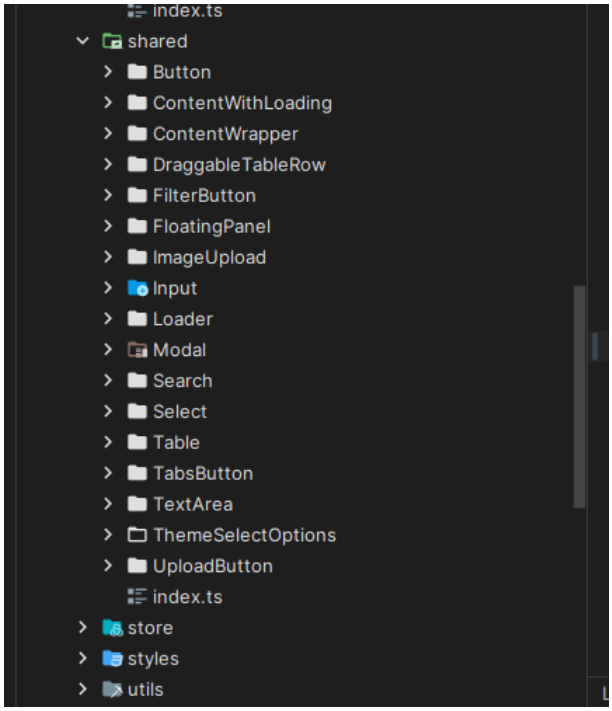


Рисунок 3.1 – Shared компоненти у системі

- Налаштування маршрутизації (Routes). (рис 3.2) Було створено систему маршрутів за допомогою React Router, що дозволяє користувачам переміщуватися між різними сторінками додатка. Кожен маршрут відповідає за відображення конкретного функціонального модуля, наприклад:
  - Dashboard — головна сторінка адміністративної панелі.
  - Students — модуль керування студентами.
  - Teachers — модуль керування викладачами.
  - Settings — сторінка з налаштуваннями програми.
- Адаптивність. Додаток розробляється з урахуванням адаптивного дизайну. Компоненти ANTD у поєднанні з кастомними стилями забезпечують коректне відображення інтерфейсу на різних пристроях — від смартфонів до великих екранів.

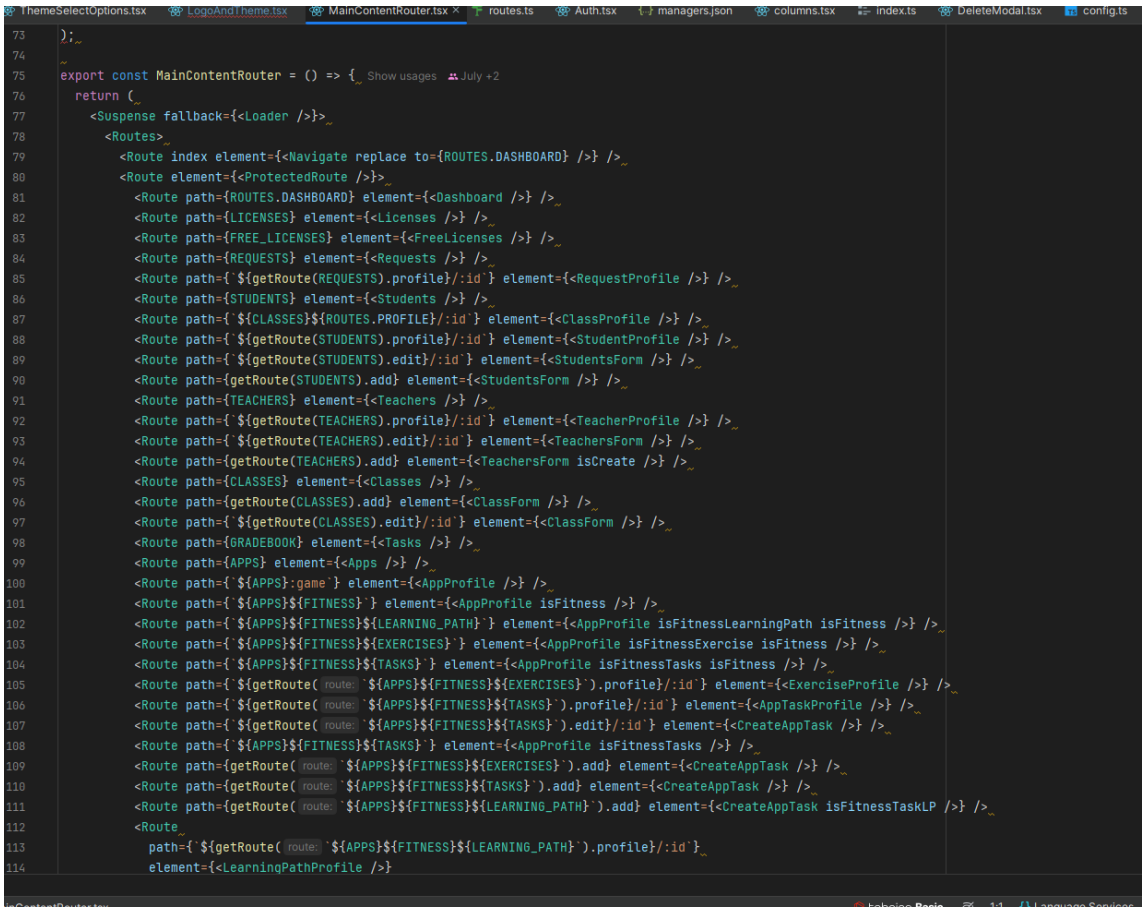


Рисунок 3.2 – Головний файл з налаштуванням шляхів по системі

Для розробки backend частини:

1. Створення проєкту на .NET Core. Був створений серверний проєкт з використанням платформи .NET Core, яка забезпечує високу продуктивність, масштабованість і кросплатформеність. Це рішення дозволяє інтегрувати серверну частину з базою даних і реалізувати бізнес-логіку програми.
2. Налаштування роботи з базою даних. Для взаємодії з базою даних використано Entity Framework Core, який дозволяє здійснювати ORM (Object-Relational Mapping), що суттєво спрощує роботу з даними. Розроблено моделі даних для зберігання інформації про школи, класи, студентів, викладачів та інші сутності системи.
3. Реалізація API. Впроваджено RESTful API, що дозволяє фронтенд-частині програми взаємодіяти з бекендом. Реалізовано основні ендпоінти для CRUD-операцій (створення, читання, оновлення та видалення даних).
4. Авторизація та аутентифікація. Для захисту доступу до системи впроваджено механізм аутентифікації з використанням JWT (JSON Web Token). Це забезпечує безпечне управління доступом користувачів, таких як директор школи, викладачі та студенти.
5. Обробка запитів. Реалізовано сервісний рівень, який відповідає за обробку запитів із фронтенду та виконання бізнес-логіки, такої як керування школами, класами, розкладом і даними студентів.
6. Запуск сервера. Сервер налаштований для роботи на хмарному середовищі AWS, що дозволяє забезпечити стабільність, швидкість доступу та легкість масштабування.

Для налаштування роботи з базою даних за допомогою Entity Framework Core (EF Core) необхідно декілька основних етапів, серед яких створення моделей даних, налаштування контексту бази даних та виконання міграцій для застосування змін до

бази даних. Нижче наведено приклад реалізації, який демонструє процес налаштування контексту бази даних.

```
public class ApplicationDbContext : DbContext
{
   public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options) { }
   // DbSet для кожної моделі
   public DbSet<School> Schools { get; set; }
   public DbSet<Class> Classes { get; set; }
   public DbSet<Student> Students { get; set; }

   protected override void OnModelCreating(ModelBuilder modelBuilder)
   {
      base.OnModelCreating(modelBuilder);
      // Налаштування зв'язків
      modelBuilder.Entity<School>()
         .HasMany(s => s.Classes)
         .WithOne(c => c.School)
         .HasForeignKey(c => c.SchoolId);
      modelBuilder.Entity<Class>()
         .HasMany(c => c.Students)
         .WithOne(s => s.Class)
         .HasForeignKey(s => s.ClassId);
   }
}
```

Контекст бази даних у Entity Framework Core є центральним компонентом для взаємодії програми з базою даних. Він виступає посередником між моделями додатка та таблицями бази даних, дозволяючи виконувати запити та зберігати дані. У контексті бази даних налаштовуються об’єкти DbSet, які представляють таблиці, та конфігурації зв’язків між моделями, такі як ключі, обмеження та навігаційні властивості. Для налаштування контексту бази даних його потрібно успадкувати від DbContext і перевизначити метод OnModelCreating, де задаються правила взаємодії між сутностями, після чого контекст реєструється у контейнері залежностей додатка для подальшого використання.

Щодо frontend частини, основна навігація в додатку реалізована у вигляді бокового меню, яке забезпечує зручний доступ до ключових розділів системи.(рис. 3.3) У верхній частині меню розташований логотип, який надає додатку ідентичність.

Меню містить посилання на основні функціональні розділи, включаючи: Dashboard для загального огляду системи, Licenses для управління ліцензіями, Requests для перегляду запитів, Students для роботи зі студентами, Teachers для керування вчителями, Classes для управління класами, Gradebook для доступу до журналу оцінок, Apps для інтеграції додаткових програм, Settings для налаштувань системи та Support для звернення за допомогою.

Також в нижній частині меню передбачено блок з можливістю швидкого придбання додаткових ліцензій, що робить систему більш адаптивною до потреб користувача. Дизайн бокового меню створений таким чином, щоб забезпечити інтуїтивний інтерфейс і швидкий доступ до всіх основних функцій програми.

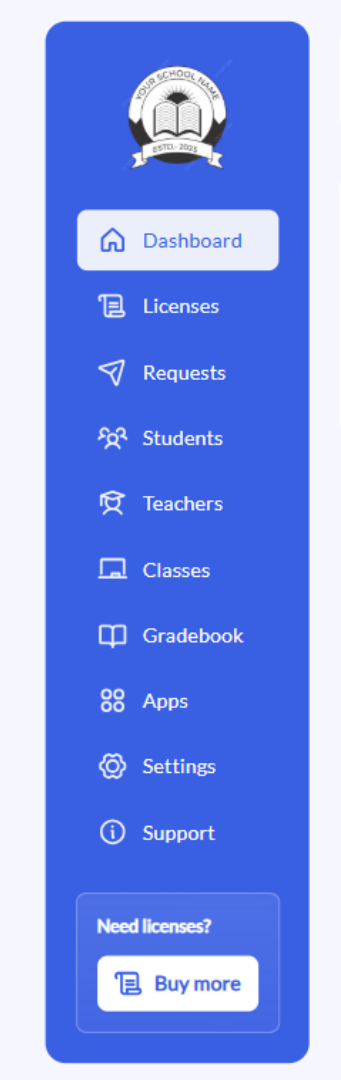


Рисунок 3.3 – Вигляд навігаційного меню

Для авторизації було створено форму реєстрації та входу у систему. Зареєструватись може лише директор школи (School Manager).(рис 3.4) В даній формі вказується електронна адреса школи, назва школи та пароль. Також, щоб було додано reCAPTCHA – це інструмент, розроблений Google, який використовується на веб-сайтах для відрізнення людей від ботів. Головна мета reCAPTCHA – запобігти спаму, злому акаунтів та іншим видам зловмисних дій, які можуть здійснюватися автоматизованими програмами.[36]

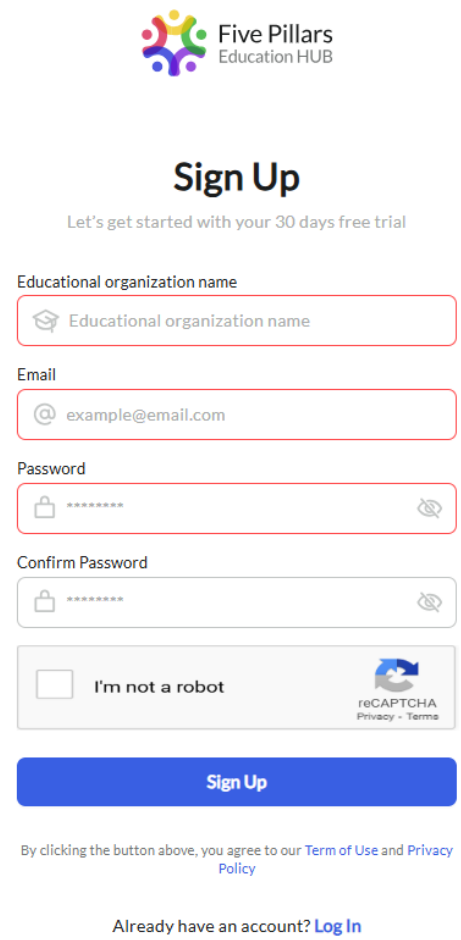


Рисунок 3.4 – Форма реєстрації.

Наступним етапом для реєстрації школи необхідно завантажити логотип школи та обрати тему для сайту школи.(рис 3.5) Після чого одразу користувач авторизується та переходить на початкову сторінку Dashboard.

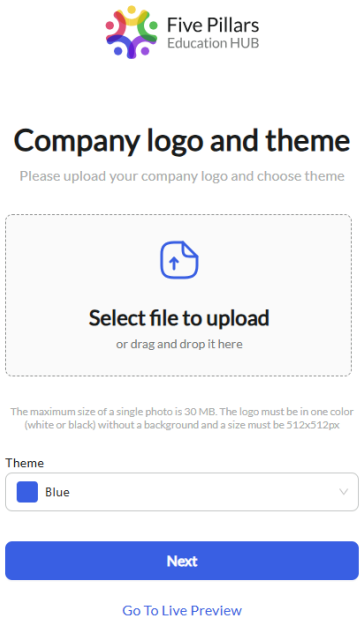


Рисунок 3.5 – Форма завантаження логотипу та теми школи

Для всі користувачів також розроблено форму логіну,(рис 3.6) при введені невірних даних нам повертається відповідна помилка, яка нам про це скаже, якщо користувач не зареєстрований йому також про це повідомлять.

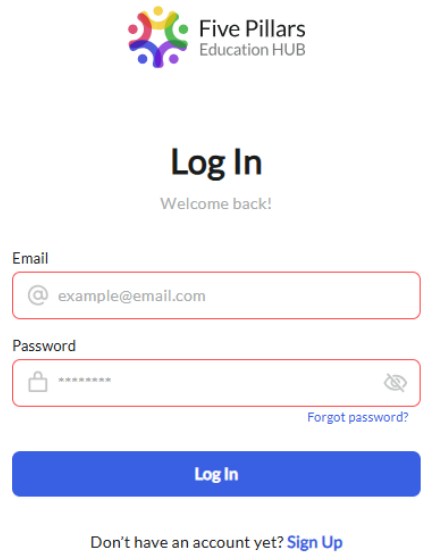


Рисунок 3.6 – Форма логіну

Для коректної реалізації авторизації користувачів і забезпечення їхньої безперебійної взаємодії з веб-додатком, на серверній стороні була розроблена єдина універсальна сутність – "User".(рис 3.7) Такий підхід обрано з метою спрощення структури даних та забезпечення масштабованості системи.

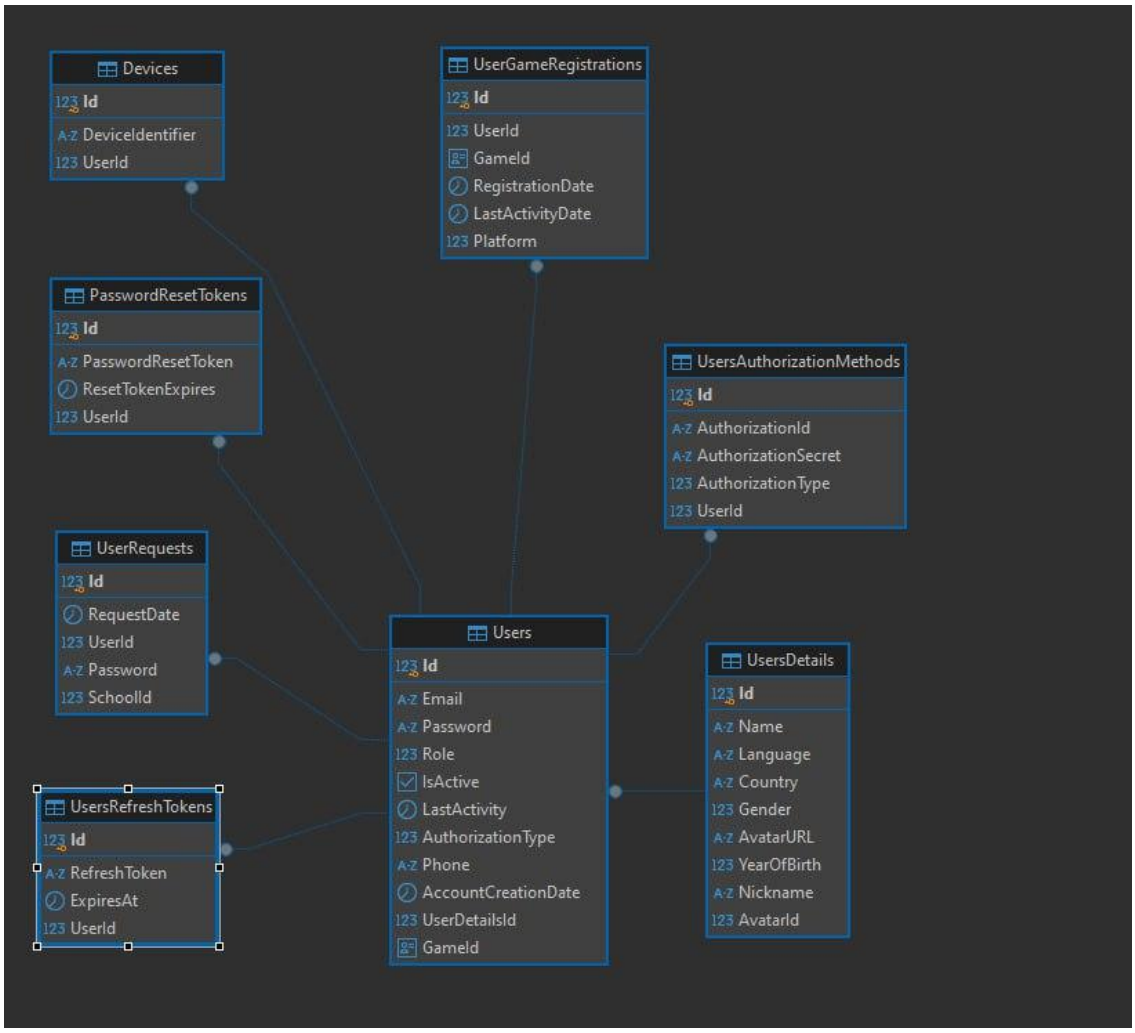


Рисунок 3.7 – Сутність «User» в базі даних

При авторизації користувача його введені облікові дані (електронна пошта та пароль) перевіряються на сервері. У разі успішної автентифікації сервер генерує два токени:

Access Token — короткостроковий токен, який дозволяє користувачу отримувати доступ до захищених ресурсів системи. Термін його дії зазвичай становить кілька хвилин, що забезпечує безпеку у разі його компрометації.

Refresh Token — довгостроковий токен, який використовується для отримання нового Access Token без повторного введення облікових даних. Термін його дії може складати кілька днів або тижнів.

Access Token передається клієнтом (Frontend) у заголовках кожного запиту до захищених API. Коли термін дії Access Token закінчується, клієнт автоматично надсилає запит на сервер із Refresh Token для отримання нового Access Token. Цей механізм забезпечує безперервність сесії користувача та знижує навантаження на сервер, оскільки облікові дані користувача не перевіряються повторно при кожному запиті. Refresh Token зберігається у захищених HTTP-only cookies, які недоступні для JavaScript, що знижує ризики компрометації токенів. Всі токени підписуються за допомогою алгоритму HMAC із секретним ключем, який зберігається на сервері. Це гарантує, що токени не можуть бути підроблені.

Для реалізації роботи з токенами на frontend стороні було написано сервіс на JavaScript для зручної роботи з cookies. Нижче наведено код сервісу:

```
class CookieService {
  getAccessToken(): string {
    const LOCAL_TOKEN = Cookies.get(SETTINGS.ACCESS_TOKEN);
    return LOCAL_TOKEN || '';
  }
  getRefreshToken(): string {
    const REFRESH_TOKEN = Cookies.get(SETTINGS.REFRESH_TOKEN);
    return REFRESH_TOKEN || '';
  }
  setData(key: string, value: any): void {
    Cookies.set(key, JSON.stringify(value));
  }
  setAccessToken(token: string) {
    Cookies.set(SETTINGS.ACCESS_TOKEN, token);
  }
  setRefreshToken(token: string, expires: string) {
    Cookies.set(SETTINGS.REFRESH_TOKEN, token, { expires: new
Date(expires).getDate() });
  }
  clear(): void {
    Object.keys(Cookies.get()).forEach(key => {
      Cookies.remove(key);
    });
  }
}
export const cookieService = new CookieService();
```

Для того щоб після закінчення часу життя access token відправлявся запит на refresh токен, в Redux є дуже крутий метод Interceptor, за допомогою якого не потрібно буде в кожному запиті прописувати логіку на це. Вся логіка роботи з токенами розробляється в цьому методі. Метод initInterceptor розроблений для автоматизації обробки запитів і відповідей у додатку, який використовує Axios для HTTP-запитів. Цей підхід дозволяє реалізувати централізовану логіку для роботи з access token та refresh token, уникаючи дублювання коду в кожному запиті. Логіка побудована на основі використання інтерцепторів Axios для перехоплення запитів і обробки помилок.

Якщо сервер повертає помилку авторизації (код 401 або 403), інтерцептор автоматично ініціює запит на оновлення токена (refresh token).

У разі успіху оновлений access token додається до повторного запиту, який був раніше відхилений. У випадку невдачі користувач перенаправляється на сторінку авторизації, а сесія завершується. У разі помилки сервера інтерцептор також враховує контекст. Наприклад, якщо помилка стосується запиту на оновлення токенів, виконується вихід із системи і перенаправлення користувача на сторінку авторизації. Кодування до методу initInterceptor можна переглянути у Додатку Б.

У разі виходу користувача з системи або спроби компрометації токенів, Refresh Token додається до списку відкликаних токенів (Blacklist). Усі запити з використанням цих токенів автоматично відхиляються.

Для наочного відображення процесу авторизації та його структуризації було створено UML-діаграму діяльності, яка допомагає зрозуміти основні етапи взаємодії користувача із системою. (рис 3.8)

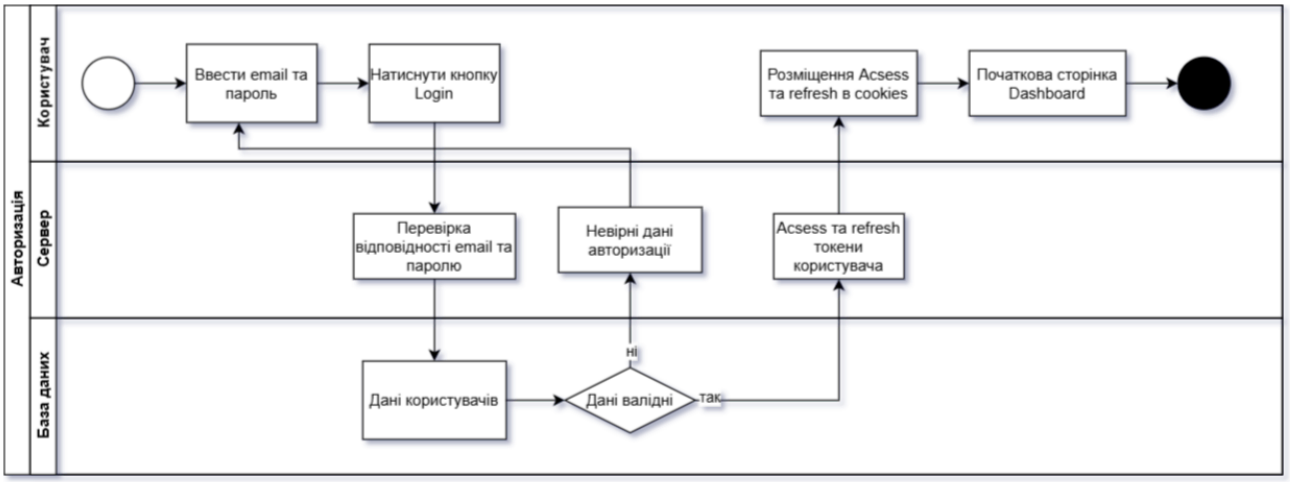


Рисунок 3.8 – UML діаграма діяльності

На представленій діаграмі діяльності описано процес авторизації користувача в інформаційній системі. Діаграма складається з трьох рівнів: користувач, сервер та база даних, що взаємодіють між собою для виконання авторизації. Основні етапи процесу:

1. Дії користувача:
   - Користувач вводить email і пароль у форму авторизації.
   - Натискає кнопку "Login" для відправлення даних.
2. Дії на сервері:
   - Сервер приймає дані, перевіряє відповідність email і пароля.
   - У разі невідповідності надаються повідомлення про невірні дані авторизації.
   - Якщо дані валідні, сервер запитує інформацію про користувача з бази даних.
3. Дії бази даних:
   - Виконується перевірка наявності відповідного запису користувача з вказаним email і паролем.
   - Повертається результат валідності даних.

4. Робота з токенами:
    - У разі успішної авторизації сервер генерує access та refresh токени для користувача.
    - Ці токени зберігаються в cookies для подальшого використання.
5. Переадресація користувача:
    - Після успішної авторизації користувача автоматично перенаправляють на початкову сторінку Dashboard.

У разі невдалої авторизації користувач отримує відповідне повідомлення, а у випадку успішної авторизації йому надається доступ до системи.

3.3 Тестування програмного забезпечення

Однією із головних функцій системи є створення учнівських завдань для їхнього навчання. Щоб створити завдання необхідно перейти на вкладку Apps, після чого обрати предмет зі списку(Read – читання, Fitness – фізкультура). В кожному із предметів є навігаційна панель де:

- Students - можна переглянути, учнів, які вивчають цей предмет
- Learning Path – список навчальних програм, які містять в собі групи завдань, є можливість створити, видалити і редагувати їх та призначити учню для навчання
- Tasks – список доступних завдань, є можливість створювати, редагувати та видаляти їх та призначати студенту до виконання
- Exercises – вправи, які містить в собі програма, вони є створенні в базі даних. (рис 3.9)

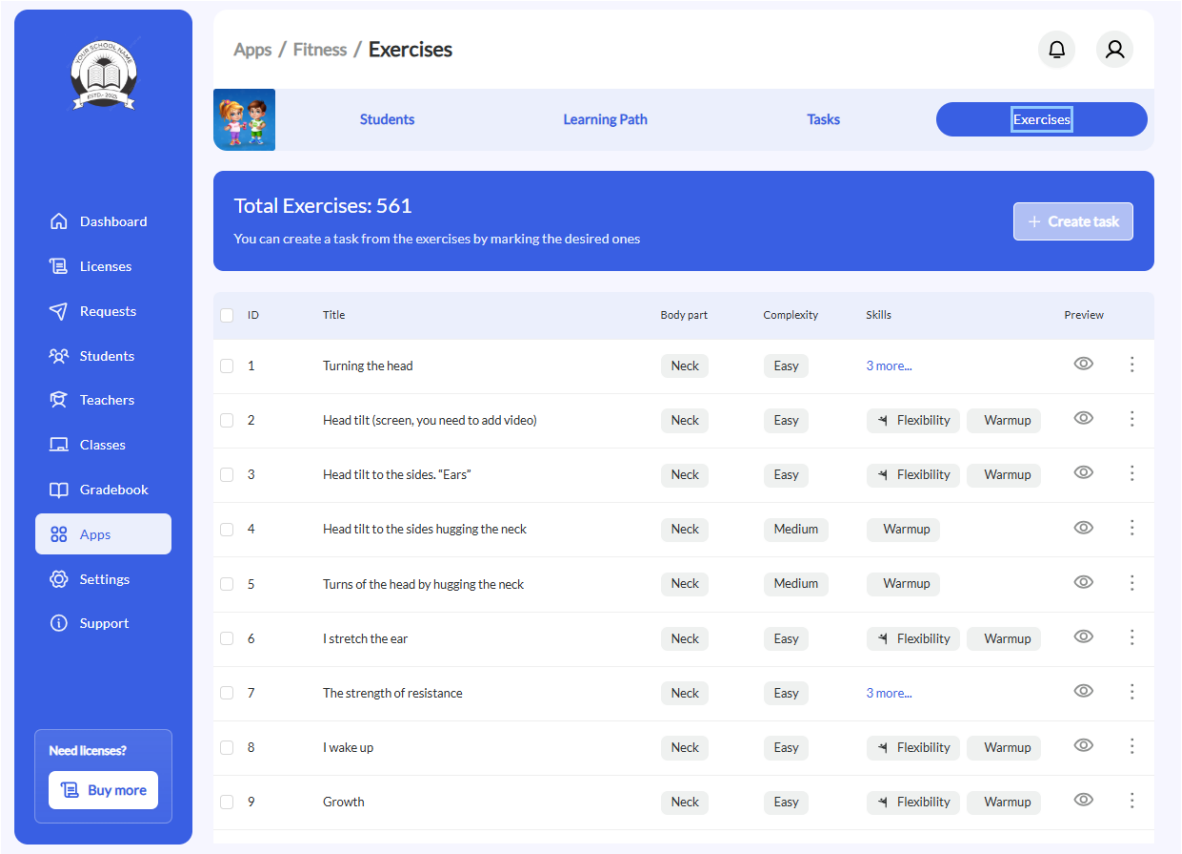


Рисунок 3.9 – Вкладка Exercises

Створити завдання можна як на вкладці Tasks так і в вкладці Exercises.

З сторінки Exercises: необхідно обрати вправи, які будуть додані до цього завдання, після чого кнопка ‘Create task’ стає активною та перекидає на форму створення завдання.(рис 3.10)

З сторінки Tasks: просто натискаємо на кнопку ‘Create task’, але вибір вправи в завданнях є обов’язковим, тому потрібно клікнути на кнопку ‘Add exercise’ і обрати спосіб, через який буде додано вправи до форми. (рис 3.11)

Функціональні компоненти вкладки Apps тісно взаємодіють між собою. Наприклад:

- Створені вправи в секції "Exercises" можуть бути включені до завдань у секції "Tasks".
- Завдання з секції "Tasks" формують основу для навчальних програм у секції "Learning Path".

- Інформація про виконання завдань у секціях "Tasks" і "Learning Path" дозволяє оновлювати дані в секції "Students", що допомагає відслідковувати індивідуальний прогрес учнів.

Такий підхід забезпечує гнучкість і адаптивність системи, дозволяючи викладачам і учням ефективно організовувати навчальний процес.

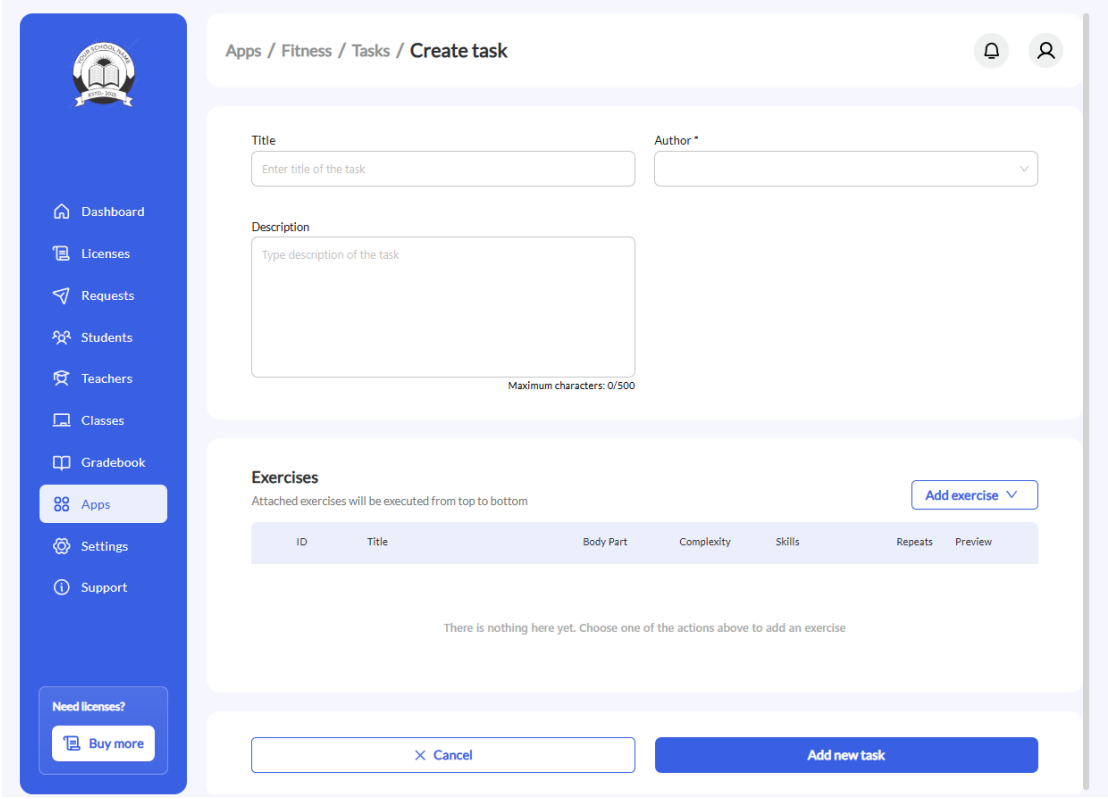


Рисунок 3.10 – Форма створення завдання

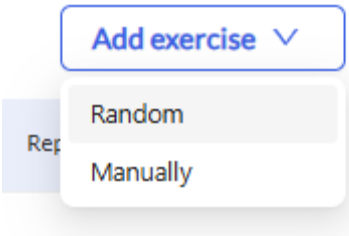


Рисунок 3.11 – Способи додавання вправ при створенні завдання

Однією з головних функцій у цій секції є призначення вчителями завдань учням. Для цього необхідно перейти на вкладку Tasks в обраній із програм та обравши із списку завдання натискаємо три крапки вкінці завдання та обираємо ‘Assign student’.(рис 3.12) Отримуємо модальне вікно із списком студентів для яких ми можемо призначити виконання цього завдання.(рис 3.13)

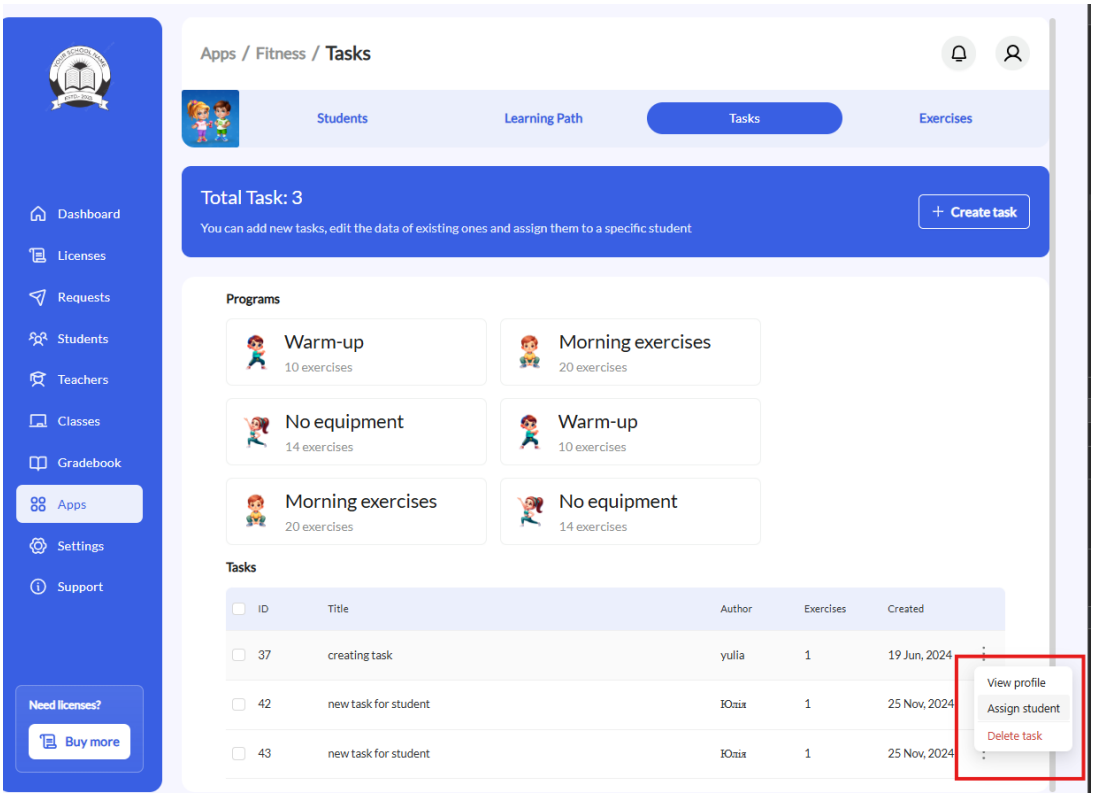


Рисунок 3.12 – Випадаюче меню з додатковими дії з завданнями

У модальному вікні, якщо призначає завдання user з роллю Teacher то перше поле (селект Teacher) одразу має дані юзера і є неактивним. Але якщо user – SchoolManager, то необхідно обрати вчителя, який буде відповідальним за призначене завдання учню. Для призначення завдання обов’язково має бути обрана дата дедлайну. Також є можливість призначити завдання одразу декільком учням.

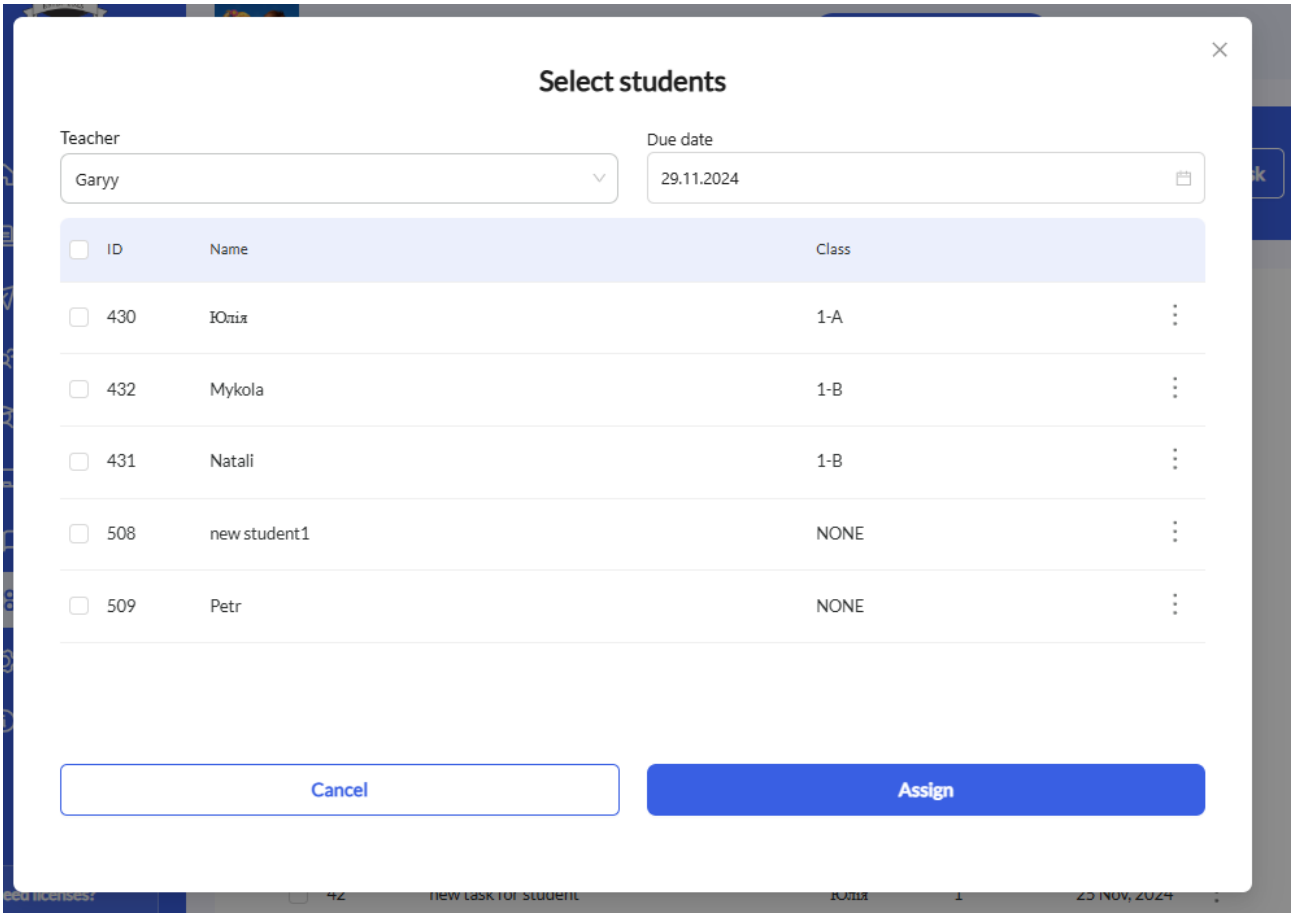


Рисунок 3.13 – Модальне вікно для призначення завдання учням

Також можна як і видалити завдання так і перейти на сторінку з деталями про це завдання та відредагувати його, додавши ще якихось вправ чи змінити опис. Також на сторінці деталей завдання є три вкладки: General, Exercises, Students.

На вкладці General вся детальна інформація про завдання, можливість видалити та відредагувати його. На вкладці Exercises: всі вправи які входять в це завдання, є можливість видалити вправу та додати нову. На вкладці Student відображаються всі студенти, яким призначили це завдання, також побачити оцінку яку отримав учень за це завдання, якщо воно виконане.(рис 3.14) Для майбутньої реалізації ще є ідея, щоб якщо учень отримав погану оцінку міг відправити завдання на перездачу. Також в правому кутку вкладки можемо побачити, що у нас є можливість додавати також і тут студентів, яким призначити виконання цього завдання можна.

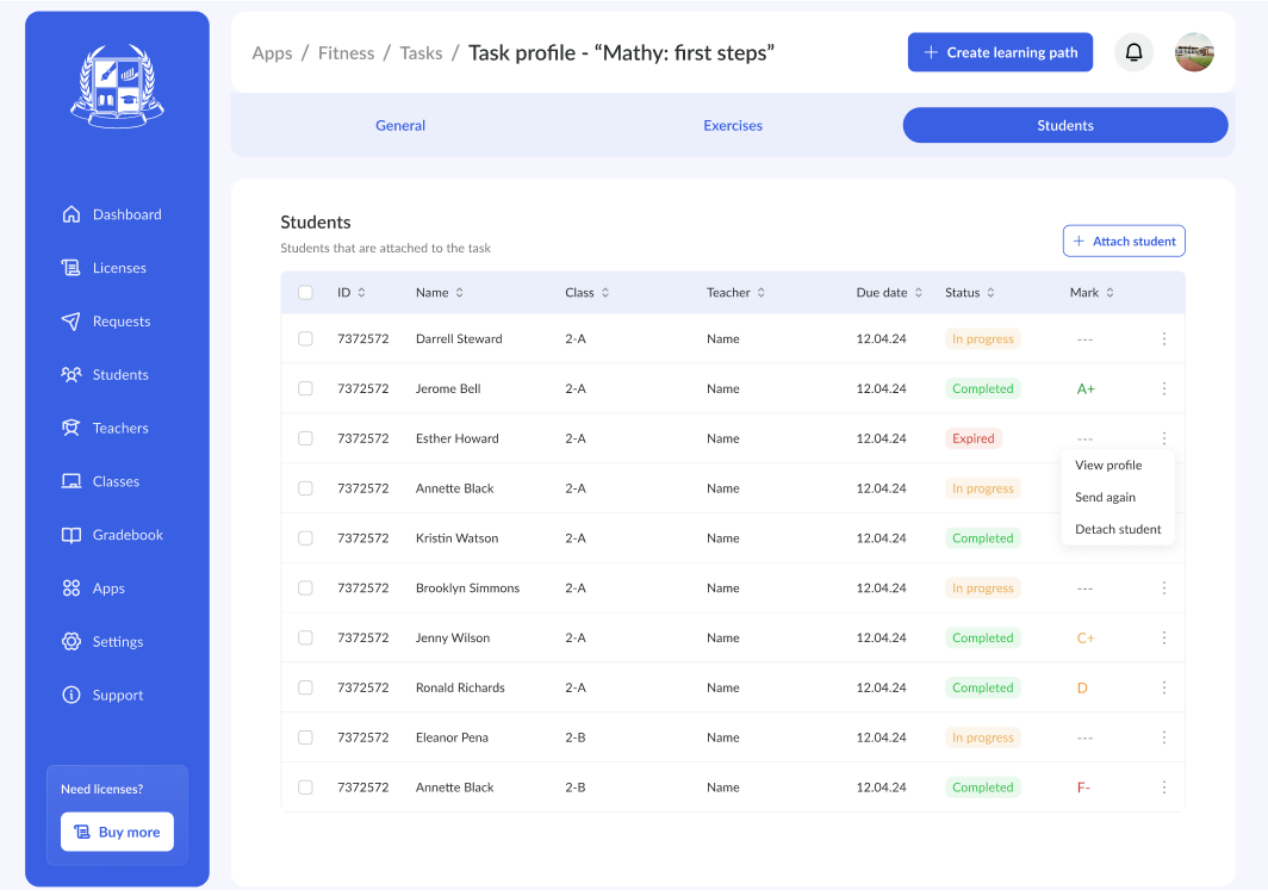


Рисунок 3.14 – Вкладка студентів, які виконують дане завдання

Проведене тестування розробленої системи управління навчальним процесом школи підтвердило її коректну роботу та відповідність поставленим вимогам. Усі функціональні модулі, включаючи авторизацію користувачів, управління класами, студентами, викладачами, а також створення та виконання завдань, працюють належним чином. Тестування показало стабільність роботи системи за різних сценаріїв використання та її здатність ефективно обробляти великий обсяг даних. Отримані результати свідчать про готовність системи до впровадження в навчальний процес та використання у реальних умовах.

4 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота «Розробка автоматизованої системи управління навчальним процесом школи» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Розробка автоматизованої системи управління навчальним процесом школи» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. [39].

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 4.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	5	5	5
2. Ринкові переваги (наявність аналогів)	3	3	2
3. Ринкові переваги (ціна продукту)	3	3	3
4. Ринкові переваги (технічні властивості)	3	3	3
5. Ринкові переваги (експлуатаційні витрати)	2	2	3
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	2	2	2
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	2	3	2
10. Практична здійсненність (необхідність нових матеріалів)	3	4	4
11. Практична здійсненність (термін реалізації)	3	4	4
12. Практична здійсненність (розробка документів)	4	4	3
Сума балів	37	40	38
Середньоарифметична сума балів $СБ_c$	38,3		

За результатами розрахунків, наведених в таблиці 4.1, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.2 [39].

Таблиця 4.2 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ ,розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий

Таблиця 4.2 (Продовження)

31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка автоматизованої системи управління навчальним процесом школи» становить 38,3 бала, що, відповідно до таблиці 4.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

4.2 Розрахунок узагальненого коефіцієнта якості розробки

Окрім комерційного аудиту розробки доцільно також розглянути технічний рівень якості розробки, розглянувши її основні технічні показники. Ці показники по-різному впливають на загальну якість проектної розробки.

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення розрахуємо за формулою [40]:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i \,, \tag{4.1}$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

α_i – коефіцієнт, який враховує питому вагу *i-го* технічного показника в загальній якості розробки. Коефіцієнт α_i визначається експертним шляхом і при цьому має виконуватись умова $\sum_{i=1}^{k} \alpha_i = 1$;

β_i – відносне значення *i-го* технічного показника якості нової розробки.

Відносні значення β_i для різних випадків розраховуємо за такими формулами:

для показників, зростання яких вказує на підвищення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ні}}{I_{аі}}, \tag{4.2}$$

де $I_{ні}$ та $I_{на}$ – чисельні значення конкретного i-го технічного показника якості відповідно для нової розробки та аналога;

для показників, зростання яких вказує на погіршення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{аі}}{I_{ні}}; \tag{4.3}$$

Використовуючи наведені залежності можемо проаналізувати та порівняти техніко-економічні характеристики аналогу та розробки на основі отриманих наявних та проектних показників, а результати порівняння зведемо до таблиці 4.3.

Таблиця 4.3 – Порівняння основних параметрів розробки та аналога.

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектований пристрій	Відношення параметрів нової розробки до аналога	Питома вага показника
Кількість підтримуваних ОС	шт.	1	2	2	0,1
Кількість підтримуваних адміністративно-управлінських функцій	шт.	18	32	1,78	0,3

Таблиця 4.3 (продовження)

Можливість взаємодії з	бал	5	8	1,6	0,15

наявними базами даних					
Інтегрування з функціональними підсистемами	бал	4	8	2	0,25
Рівень захищеності баз даних	бал	7	9	1,26	0,2

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення складе:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i = 2 \cdot 0,1 + 1,78 \cdot 0,3 + 1,6 \cdot 0,15 + 2 \cdot 0,25 + 1,26 \cdot 0,2 = 1,73.$$

Отже за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,73 рази.

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Розробка автоматизованої системи управління навчальним процесом школи», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

4.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [Козловський, Лесько, Кавецький]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.4)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=22 дні.

$$З_о = 16000{,}00 \cdot 15 / 22 = 10909{,}09 \; грн.$$

Проведені розрахунки зведемо до таблиці.

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	16000,00	727,27	15	10909,09
Інженер-програміст	15800,00	718,18	12	8618,18
Консультант з забезпечення управління навчальним процесом (завуч школи)	15950,00	725,00	5	3625,00
Інженер-проектувальник автоматизованих систем управління	15000,00	681,82	12	8181,82
Консультант (адміністратор)	14200,00	645,45	2	1290,91
Всього				32625,00

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Розробка автоматизованої системи управління навчальним процесом школи» розраховуємо за формулою:

$$З_p = \sum_{i=1}^{n} C_i \cdot t_i , \quad (4.5)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} , \quad (4.6)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо *Мм*=8000,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [39].

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 22 дн;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 8000{,}00 \cdot 1{,}10 \cdot 1{,}15 / (22 \cdot 8) = 57{,}50$ *грн.*

$З_{p1} = 57{,}50 \cdot 6{,}30 = 362{,}25$ *грн.*

Таблиця 4.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Установка офісного обладнання	6,30	2	1,10	57,50	362,25

Підготовка робочого місця розробника системи	7,20	3	1,35	70,57	508,09
Інсталяція програмного забезпечення розробки програмних модулів	5,00	5	1,70	88,86	444,32
Формування інформаційної бази даних управління навчальним процесом	11,00	4	1,50	78,41	862,50
Тренування системи взаємодії програмних функціональних блоків управління	8,00	4	1,50	78,41	627,27
Тестування підсистем	12,00	3	1,35	70,57	846,82
Монтаж інтерфейсних блоків функціонального управління	6,00	6	2,00	104,55	627,27
Всього					4278,52

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.7)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (32625,00 + 4278,52) · 11 / 100% = 4059,39 грн.

4.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.8)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (32625,00 + 4278,52 + 4059,39) · 22 / 100% = 9011,84 грн.

4.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Розробка автоматизованої системи управління навчальним процесом школи».

Витрати на матеріали ($М$), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}\ , \quad (4.9)$$

де $Н_j$ – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ *= 1,1 ... 1,15*);

$В_j$ – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$М_1$ = 2,0 · 190,00 · 1,02 - 0 · 0 = 387,60 грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 4.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір канцелярський	190,00	2,0	0	0	387,60

офісний AMOCOOL-500 (A4)					
Папір для заміток AMOCOOL-B (A5)/70	88,00	3,0	0	0	269,28
Начиння канцелярське AMOCOOL	152,00	2,0	0	0	310,08
Органайзер офісний AMOCOOL light	102,00	4,0	0	0	416,16
Картридж для принтера HP-7021	1085,00	2,00	0	0	2213,40
Диск оптичний COOL-CD/RW	25,00	4,0	0	0	102,00
FLASH-пам'ять KingsBAFF 32GB	119,00	2,0	0	0	242,76
Тека для паперів	78,00	4,000	0	0	318,24
Всього					4658,87

4.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Розробка автоматизованої системи управління навчальним процесом школи», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (4.10)$$

де $Н_j$ – кількість комплектуючих *j*-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих *j*-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ *= 1,1 ... 1,15*).

$К_в = 5 \cdot 420,00 \cdot 1,02 = 2142,00$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.7 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Інтерфейсні блоки взаємозв'язків функціональних систем управління	5	420,00	2142,00
Всього			2142,00

4.3.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (4.11)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань устаткування.

$B_{спец} = 15700,00 \cdot 1 \cdot 1,02 = 16014,00$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.8 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Серверний блок системи управління	1	15700,00	16014,00

Автоматизована система відеоспостереження	1	12200,00	12444,00
Інтерфейсний блок обміну інформацією з системою планування навчального процесу	1	9400,00	9588,00
Всього			38046,00

4.3.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.12)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 6720,00 \cdot 1 \cdot 1,02 = 6854,40$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.9 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн

Прикладний пакет MATLab VisProject	1	6720,00	6854,40
Project Management 12 PRO	1	7800,00	7956,00
Абонентна плата доступу до мережі Internet (1 місяць)	1	325,00	331,50
Всього			15141,90

4.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (4.13)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (29200{,}00 \cdot 1) / (2 \cdot 12) = 1216{,}67$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.10 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн

Програмно-аналітичний комплекс	29200,00	2	1	1216,67
Графічно-обчислювальний комплекс обробки даних	27200,00	2	1	1133,33
Програмне забезпечення розробки автоматизованих систем	10620,00	2	1	442,50
Обладнання виводу інформації	8600,00	4	1	179,17
Оргтехніка	8730,00	4	1	181,88
Приміщення лабораторії	245000,00	25	1	816,67
Прикладний Офісний пакет Microsoft Office 2016	6540,00	2	1	272,50
OS Windows 10	6800,00	2	1	283,33
Всього				4526,04

4.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \qquad (4.14)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо Це = 10,98 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,35 \cdot 160,0 \cdot 10,98 \cdot 0,95 / 0,97 = 614,88$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.11 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Програмно-аналітичний комплекс	0,35	160,0	614,88
Графічно-обчислювальний комплекс обробки даних	0,25	160,0	439,20
Обладнання виводу інформації	0,12	4,0	5,27
Оргтехніка	0,10	2,5	2,75
Серверний блок системи управління	0,12	160,0	210,82
Автоматизована система відеоспостереження	0,10	160,0	175,68
Всього			1448,59

4.3.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Розробка автоматизованої системи управління навчальним процесом школи» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{H_{св}}{100\%}, \qquad (4.15)$$

де $H_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$В_{св}$ = (32625,00 + 4278,52) · 20 / 100% = 7380,70 грн.

4.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» відсутні.

4.3.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.16)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$І_в$ = (32625,00 + 4278,52) · 50 / 100% = 18451,76 грн.

4.3.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.17)$$

де $H_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $H_{нзв}$ = 105%.

$B_{нзв} = (32625,00 + 4278,52) \cdot 105 / 100\% = 38748,70$ *грн.*

Витрати на проведення науково-дослідної роботи на тему «Розробка автоматизованої системи управління навчальним процесом школи» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + I_в + B_{нзв}. \quad (4.18)$$

$B_{заг}$= *32625,00 +4278,52 +4059,39 +9011,84 +4658,87 +2142,00 +38046,00 + 15141,90 + 4526,04 +1448,59 +7380,70 +0,00 +18451,76 +38748,70 = 180519,32 грн.*

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (4.19)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

$$ЗВ = 180519,32 / 0,9 = 200577,02 \text{ грн.}$$

4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

Результати дослідження проведені за темою «Розробка автоматизованої системи управління навчальним процесом школи» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

Показник	1-й рік	2-й рік	3-й рік	4-й рік

Збільшення кількості споживачів, осіб	25	50	50	30

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 300 осіб;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 68500,00 грн;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 4731,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [39]:

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.20)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2024 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =35%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2024 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ =(4731,00·300,00+73231,00·25)·0,83·0,35·(1-0,18/100%)=774200,37 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ =(4731,00·300,00+73231,00·75)·0,83·0,35·(1-0,18/100%)=1646418,19 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (4731{,}00 \cdot 300{,}00 + 73231{,}00 \cdot 125) \cdot 0{,}83 \cdot 0{,}35 \cdot (1\text{-}0{,}18/100\%) = 2518636{,}02$ грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4 = (4731{,}00 \cdot 300{,}00 + 73231{,}00 \cdot 155) \cdot 0{,}83 \cdot 0{,}35 \cdot (1\text{-}0{,}18/100\%) = 3041966{,}71$ грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.21)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,25;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП = 774200{,}37/(1+0{,}25)^1 + 1646418{,}19/(1+0{,}25)^2 + 2518636{,}02/(1+0{,}25)^3 +$

$+ 3041966{,}71/(1+0{,}25)^4 = 619360{,}29 + 1053707{,}64 + 1289541{,}64 + 1245989{,}57 =$

$= 4208599{,}14$ грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \qquad (4.22)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 200577,02 грн.

$PV = k_{інв} \cdot ЗВ = 2 \cdot 200577{,}02 = 401154{,}04$ грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{4.23}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 4208599,14 грн;

PV – теперішня вартість початкових інвестицій, 401154,04 грн.

$E_{абс} = ПП - PV = 4208599{,}14 - 401154{,}04 = 3807445{,}10$ грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.24}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 3807445,10 грн;

PV – теперішня вартість початкових інвестицій, 401154,04 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+3807445{,}10/401154{,}04)^{1/4} = 0{,}80.$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \tag{4.25}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d =0,11;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,15.

$\tau_{мін}$ = 0,11+0,15 = 0,26 < 0,80 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Розробка автоматизованої системи управління навчальним процесом школи» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{4.26}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,80 = 1,25 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Висновки до розділу

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка автоматизованої системи управління навчальним процесом школи» становить 38,3 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,73 рази.

Також термін окупності становить 1,25 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Розробка автоматизованої системи управління навчальним процесом школи».

ВИСНОВКИ

У ході виконання роботи було досягнуто поставленої мети – покращення ефективності та якості управління навчальним процесом шляхом розробки інтегрованої автоматизованої системи. Розроблена система об'єднує функції моніторингу, аналізу та оптимізації, спрямовані на автоматизацію організаційних процесів у навчальному закладі.

Було проведено детальне дослідження освітнього процесу, визначено його ключові аспекти та проблеми, пов’язані з рутинними адміністративними завданнями, комунікацією між учасниками освітнього процесу, а також з моніторингом і аналізом успішності учнів.

Проаналізовано сучасні системи управління освітнім процесом, їх переваги та недоліки. Визначено, що більшість наявних рішень не забезпечують повної інтеграції всіх аспектів управління навчальним закладом і не адаптовані до Розроблено автоматизовану систему управління навчальним процесом із використанням сучасних інструментів та технологій. Система забезпечує:

1. інтеграцію баз даних для зберігання даних учнів та співробітників;
2. функції для моніторингу успішності, управління документацією та організації комунікацій;
3. гнучкий інтерфейс, який забезпечує зручність у використанні для різних ролей (адміністратори, вчителі, учні, батьки);
4. захист даних відповідно до сучасних стандартів безпеки, сучасних викликів, таких як дистанційне навчання.

Практична цінність розробленої системи полягає у значному скороченні рутинного навантаження на керівників і вчителів, оптимізації адміністративних процесів, підвищенні прозорості та доступності інформації для всіх учасників навчального процесу. Це сприяє підвищенню загальної ефективності роботи навчального закладу та якості освітніх послуг. Інноваційність розробки полягає у створенні інтегрованої автоматизованої системи, що дозволяє ефективно поєднувати

різні аспекти управління освітнім процесом, забезпечувати персоналізовану взаємодію між учасниками освітнього середовища та оперативно адаптуватися до змінних умов навчання, зокрема в умовах дистанційного навчання.

Результати дослідження апробовані в межах науково-технічної конференції та підтвердили практичну значущість і ефективність запропонованого рішення. У майбутньому передбачається подальше вдосконалення системи для розширення її функціоналу та впровадження у більшу кількість навчальних закладів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Топ системи для дистанційного навчання. [Електронний ресурс]. // Ельвіра Жуковська - 2020 URL: https://unicheck.com/ua/blog/top-3-sistemi-dlya-distanczijnogo-navchannya

2. Сутність і структура процесу навчання у вищій школі. [Електронний ресурс] // Потапюк Л.М - 2022 URL: https://studentam.net.ua/content/view/2269/97/

3. Google Classroom: Everything you need to know. [Електронний ресурс] // Ganesh Mukundan, Oct 24, 2024 – URL: https://hiverhq.com/blog/google-classroom-basics

4. Moodle. [Електронний ресурс]. //Content is available under GNU General Public License, official documentation 25 April 2024 – URL: https://docs.moodle.org/405/en/About_Moodle

5. Automated schools: 7 school processes you can automate. [Електронний ресурс] // Jotform Editorial Team, December 5, 2023 – URL: https://www.jotform.com/blog/automated-school/

6. Process Management In Education: Understand Its Importance. [Електронний ресурс] // Leonardo Carvalho, Updated at 21/02/2024 – URL: https://www.sydle.com/blog/process-management-in-education-626c2a0361423f655c608217

7. Кравченко О.В. Роль інформаційних технологій в сучасному освітньому просторі. [Електронний ресурс] // Кравченко О.В., 2019 – URL: https://naurok.com.ua/rol-informaciynih-tehnologi-v-suchasnomu-osvitnomu-prostori-135431.html

8. Проблеми інформатизації навчального процесу в школі і в вузі. [Електронний ресурс] // Жалдак М.І. , Київський державний педагогічний інститут ім.М.П.Драгоманова), 01.01.2011 – URL: https://core.ac.uk/download/pdf/11083869.pdf

9. Карплюк С.О., Вакалюк Т.А. Огляд функціональних можливостей програмного забезпечення для управління освітнім процесом закладу вищої освіти. [Електронний ресурс]// Карплюк Світлана Олександрівна, Вакалюк Тетяна Анатоліївна, Інформаційні технології і засоби навчання 2018 – URL:

http://eprints.zu.edu.ua/27263/1/%D0%9A%D0%B0%D1%80%D0%BF%D0%BB%D1%8E%D0%BA_%D0%92%D0%B0%D0%BA%D0%B0%D0%BB%D1%8E%D0%BA_WOS.pdf

10. Довганюк В., Рогатинська Л. Автоматизована система управління навчальним процесом у загальноосвітньому навчальному закладі. [Електронний ресурс] // В. Довганюк, Л. Рогатинська, (Тернопільський національний технічний університет імені Івана Пулюя)– URL: https://elartu.tntu.edu.ua/bitstream/123456789/7892/2/Conf_2014_Dovhaniuk_V-Avtomatyzovana_systema_upravlinnia_21.pdf

11. React. [Електронний ресурс] // March 16, 2023 by Dan Abramov and Rachel Nabors – URL https://react.dev/blog/2023/03/16/introducing-react-dev

12. . Introduction to React. Apress. [Електронний ресурс] // Gackenheimer C, 2015 URL: https://pepa.holla.cz/wp-content/uploads/2016/12/Introduction-to-React.pdf

13. Turner A. Angular analysis. In: Proceedings of the 3rd international symposium on space syntax. Atlanta, GA: Georgia Institute of Technology, 2001. p. 30-11.

14. Hanchett E., Listwon B. Vue.js in Action. Simon and Schuster, 2018.

15. Conallen J. Modeling web application architectures with UML. Communications of the ACM, 1999, 42(10): 63-70.

16. Del Pilar Salas-Zárate M., et al. Analyzing best practices on Web development frameworks: The lift approach. Science of Computer Programming, 2015, 102: 1-19.

17. Vue.js. [Електронний ресурс]. // Released under the MIT License. Copyright © 2014-2024 Evan You – URL: https://vuejs.org/

18. Angular. [Електронний ресурс] // official documentation. Super-powered by Google ©2010-2024. URL: https://angular.io/

19. Web frameworks. [Електронний ресурс]. – geeksforgeeks.org, 15 Oct, 2024 . URL: https://www.geeksforgeeks.org/top-10-frameworks-for-web-applications/

20. Curie D.H., et al. Analysis on web frameworks. Journal of Physics: Conference Series. IOP Publishing, 2019. p. 012114.

21. .NET Framework documentation. [Електронний ресурс] // Bill Wagner and other 12 contributions, .03.30.2023 – URL: https://learn.microsoft.com/en-us/dotnet/framework/

22. What is PostgreSQL. [Електронний ресурс] // Neon, January 18, 2024 – URL: https://www.postgresql.org/docs/current/

23. Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch, and others. [Електронний ресурс] // AltexSoft, 16 May, 2023 – URL: https://www.altexsoft.com/blog/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/

24. Козак Ю. Розробка автоматизованої системи управління навчальним процесом школи. [Електронний ресурс] LIII Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації – URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/view/20929

25. Клієнт-серверна архітектура. . [Електронний ресурс] // Пульс дня, 30.10.2023 – URL: https://www.volyn.com.ua/news/256294-khostynh-saitu-vzaiemodiia-kliient-server

26. What is File server. [Електронний ресурс], geeksforgeeks.org, 31 Aug, 2022 – URL: https://www.geeksforgeeks.org/what-is-file-server/

27. 3-tier application architecture. Michael Chiaramonte, May 13, 2024 [Електронний ресурс] Michael Chiaramonte, May 13, 2024 – URL: https://vfunction.com/blog/3-tier-application/

28. HTML. [Електронний ресурс] // Nov 21, 2024 by MDN contributors – URL: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics

29. TypeScript. [Електронний ресурс] // Contributor: anshubajaj911, 11 Mar, 2024 – URL: https://www.geeksforgeeks.org/introduction-to-typescript/

30. BEM. [Електронний ресурс] // Methodology, 2020 – URL: https://en.bem.info/methodology/key-concepts/

31. Styled Components. [Електронний ресурс] Made by @glenmaddern, @mxstbr, @_philpl, @probablyup, @imbhargav5 and contributors. – URL: https://styled-components.com/

32. ANTD. [Електронний ресурс] Ant Group and Ant Design Community, till now – URL: https://ant.design/

33. CSS. [Електронний ресурс] // last modified on Jul 25, 2024 by MDN contributors – URL: https://developer.mozilla.org/en-US/docs/Web/CSS

34. Для чого і коли використовується Redux. [Електронний ресурс] // foxminded 08.11.2023 – URL: https://foxminded.ua/shcho-take-redux/

35. Amazon Relational Database Service. [Електронний ресурс] Amazon Web Service s, Inc. Or its affiliates, 2024 – URL: https://aws.amazon.com/rds/?did=ap_card&trk=ap_card

36. reCAPTCH. [Електронний ресурс] Official Google documentation, 2024 – URL: https://cloud.google.com/security/products/recaptcha

37. Vite. [Електронний ресурс] Directed by vite team, 2024 – URL: https://vite.dev/guide/

38. 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] / уклад. В. М. Дубовой. – Вінниця :ВНТУ, 2023 – (PDF, 45 с.). URL: https://iq.vntu.edu.ua/repository/getfile.php/6077.pdf

39. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

40. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.

ДОДАТКИ

Додаток А

(обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: _« Розробка автоматизованої системи управління навчальним процесом школи »__

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_____Ковалюк О.О., доцент кафедри КСУ_____

(прізвище, ініціали, посада)

Показники звіту подібності

Turnitin	
Оригінальність	91%
Загальна схожість	9%

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 _____ _____Юлія КОЗАК_____

(підпис) (прізвище, ініціали)

Опис прийнятого рішення

_____

_____

_____

Особа, відповідальна за перевірку_ _____ __ _Володимир ДУБОВОЙ _

(підпис) (прізвище, ініціали)

Додаток Б

(обов'язковий)

ВНТУ

ЗАТВЕРДЖЕНО
Зав. кафедри КСУ ВНТУ,
д.т.н., професор
_____ В'ячеслав КОВТУН

"14" 10 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ ШКОЛИ

08-33.МКР.04.00.000 ТЗ

Студент групи 2АКІТР-23м

_____ Юлія КОЗАК
Підпис *Ім'я ПРІЗВИЩЕ*

Керівник _____

_____ Олег КОВАЛЮК
Підпис *Ім'я ПРІЗВИЩЕ*

Вінниця 2024

1. Назва та галузь застосування

1.1. Назва – Розробка автоматизованої системи управління навчальним процесом школи.

1.2. Галузь застосування – освіта.

2. Підстава для проведення розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ від Протокол №310 від «17» вересня 2024року.

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є покращення управління різними аспектами навчального процесу шляхом розробки інтегрованої системи, яка об'єднує в собі функції моніторингу, аналізу та оптимізації.

4. Джерела розробки.

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

1. Сутність і структура процесу навчання у вищій школі. [Електронний ресурс] // Потапюк Л.М - 2022 URL: https://studentam.net.ua/content/view/2269/97/

2. Проблеми інформатизації навчального процесу в школі і в вузі. [Електронний ресурс] // Жалдак М.І. , Київський державний педагогічний інститут ім.М.П.Драгоманова), 01.01.2011 – URL: https://core.ac.uk/download/pdf/11083869.pdf

3. Карплюк С.О., Вакалюк Т.А. Огляд функціональних можливостей програмного забезпечення для управління освітнім процесом закладу вищої освіти. [Електронний ресурс]// Карплюк Світлана Олександрівна, Вакалюк Тетяна Анатоліївна, Інформаційні технології і засоби навчання 2018 – URL: http://eprints.zu.edu.ua/27263/1/%D0%9A%D0%B0%D1%80%D0%BF%D0%BB%D1%8E%D0%BA_%D0%92%D0%B0%D0%BA%D0%B0%D0%BB%D1%8E%D0%BA_WOS.pdf

5. Вимоги до розробки.

5.1. Перелік головних функцій:

- автоматизація та оптимізація процесу навчання;
- керування базою даних;
- модуль підтримки дистанційного навчання;

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:

- PostgreSQL,
- React,
- .Net,
- AWS,
- Windows 11,

5.2.2. Умови експлуатації системи:

- безперебійне цілодобове функціонування системи;
- регулярне оновлення та підтримка актуальності даних
- надійний захист даних

6. Стадії та етапи розробки.

6.1 Пояснювальна записка:

1. Аналіз методів, принципів, підходів і засобів реалізації задачі автоматизації процесами в об’єкті управління відповідно до теми кваліфікаційної роботи. Постановка задач дослідження 03.09.2024 р.
2. Удосконалення технології прийняття рішень при автоматизації об’єкту управління 23.09.2024 р.
3. Визначення технічних характеристик системи 04.10.2024 р.
4. Розробка програмного забезпечення системи 10.11.2024 р.

6.2 Графічні матеріали:

1. Розробка UML-діаграм системи 15.12.2024 р.
2. Розробка моделі бази даних системи 04.10.2024 р.
3. Тестування програмного забезпечення 11.11.2024 р.

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до «20» 11 2024 р.

7.2. Атестація МКР здійснюється на попередньому захисті. Попередній захист магістерської кваліфікаційної роботи провести до «1» 12 2024 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК. Захист магістерської кваліфікаційної роботи провести до «10» 12 2024 р.

Додаток В
(вибірковий)
Лістинг програми

```
//Interceptors

import axios, { AxiosInstance } from 'axios';
import{GameBaseUrl, LearningBaseUrl, LicenseBaseUrl, SchoolManagerBaseUrl, UserBaseUrl } from 'config/index';
import { cookieService as CookieService, ROUTES, SCHOOL_MANAGER_API } from 'core/index';
import store from 'store/root';
import { refreshAccessToken } from 'store/school-service/action';
import { logout } from 'store/school-service/reducers';

const axiosSchoolManagerConfig = {
  baseURL: SchoolManagerBaseUrl,
};
const axiosGameManagerConfig = {
  baseURL: GameBaseUrl,
};
const axiosLicenseManagerConfig = {
  baseURL: LicenseBaseUrl,
};
const axiosUserManagerConfig = {
  baseURL: UserBaseUrl,
};
const axiosLearningManagerConfig = {
  baseURL: LearningBaseUrl,
};
export const SchoolManagerInstance = axios.create(axiosSchoolManagerConfig);
export const GameInstance = axios.create(axiosGameManagerConfig);
export const LicenseInstance = axios.create(axiosLicenseManagerConfig);
export const UserInstance = axios.create(axiosUserManagerConfig);
export const LearningInstance = axios.create(axiosLearningManagerConfig);
```

```
export const initInterceptor = (instance: AxiosInstance) => {
  instance.interceptors.request.use(
    async req => {
      const token: string = CookieService.getAccessToken();

      if (token && req?.url && !req.url.includes('refresh-token') && !req.headers?.Authorization) {
        Object.assign(req.headers, {
          Authorization: `Bearer ${token}`,
        });
      }
      return req;
    },
    error => {
      return Promise.reject(error.response);
    }
  );
  instance.interceptors.response.use(
    response => response,
    async error => {
      const { dispatch } = store;

      if (
        (error?.response.status === 403 || error?.response.status === 401) &&
        !window.location.href.includes(ROUTES.LOG_IN)
      ) {
        error.config._retry = true;
        try {
          const accessToken = await dispatch(refreshAccessToken()).unwrap();
          const failedRequest = {
            ...error.config,
            headers: {
              ...error.config.headers,
              Authorization: `Bearer ${accessToken}`,
            },
```

```
        };
        return instance(failedRequest);
      } catch (e) {
        window.location.href = ROUTES.AUTH_PAGE.replace('*', '');
        throw error.response;
      }
    } else if (error?.response?.status === 500) {
      error.config._retry = true;

      if (error.config.url.includes(SCHOOL_MANAGER_API.refreshAccessToken)) {
        window.location.href = ROUTES.AUTH_PAGE.replace('*', '');
        dispatch(logout());
        throw error.response;
      }
      throw error.response;
    }
    throw error.response;
  }
 );
 return instance;
};
export const SchoolManagerAxios = initInterceptor(SchoolManagerInstance);
export const GameManagerAxios = initInterceptor(GameInstance);
export const LicenseManagerAxios = initInterceptor(LicenseInstance);
export const UserManagerAxios = initInterceptor(UserInstance);
export const LearningManagerAxios = initInterceptor(LearningInstance);

//Routes

export const MainContentRouter = () => {
 return (
  <Suspense fallback={<Loader />}>
   <Routes>
    <Route index element={<Navigate replace to={ROUTES.DASHBOARD} />} />
    <Route element={<ProtectedRoute />}>
```

```
<Route path={ROUTES.DASHBOARD} element={<Dashboard />} />
<Route path={LICENSES} element={<Licenses />} />
<Route path={FREE_LICENSES} element={<FreeLicenses />} />
<Route path={REQUESTS} element={<Requests />} />
<Route path={`${getRoute(REQUESTS).profile}/:id`} element={<RequestProfile />} />
<Route path={STUDENTS} element={<Students />} />
<Route path={`${CLASSES}${ROUTES.PROFILE}/:id`} element={<ClassProfile />} />
<Route path={`${getRoute(STUDENTS).profile}/:id`} element={<StudentProfile />} />
<Route path={`${getRoute(STUDENTS).edit}/:id`} element={<StudentsForm />} />
<Route path={getRoute(STUDENTS).add} element={<StudentsForm />} />
<Route path={TEACHERS} element={<Teachers />} />
<Route path={`${getRoute(TEACHERS).profile}/:id`} element={<TeacherProfile />} />
<Route path={`${getRoute(TEACHERS).edit}/:id`} element={<TeachersForm />} />
<Route path={getRoute(TEACHERS).add} element={<TeachersForm isCreate />} />
<Route path={CLASSES} element={<Classes />} />
<Route path={getRoute(CLASSES).add} element={<ClassForm />} />
<Route path={`${getRoute(CLASSES).edit}/:id`} element={<ClassForm />} />
<Route path={GRADEBOOK} element={<Tasks />} />
<Route path={APPS} element={<Apps />} />
<Route path={`${APPS}:game`} element={<AppProfile />} />
<Route path={`${APPS}${FITNESS}`} element={<AppProfile isFitness />} />
<Route path={`${APPS}${FITNESS}${LEARNING_PATH}`} element={<AppProfile
isFitnessLearningPath isFitness />} />
<Route path={`${APPS}${FITNESS}${EXERCISES}`} element={<AppProfile isFitnessExercise
isFitness />} />
<Route path={`${APPS}${FITNESS}${TASKS}`} element={<AppProfile isFitnessTasks
isFitness />} />
<Route path={`${getRoute(`${APPS}${FITNESS}${EXERCISES}`).profile}/:id`}
element={<ExerciseProfile />} />
<Route path={`${getRoute(`${APPS}${FITNESS}${TASKS}`).profile}/:id`}
element={<AppTaskProfile />} />
<Route path={`${getRoute(`${APPS}${FITNESS}${TASKS}`).edit}/:id`}
element={<CreateAppTask />} />
<Route path={`${APPS}${FITNESS}${TASKS}`} element={<AppProfile isFitnessTasks />} />
```

```
        <Route                          path={getRoute(`${APPS}${FITNESS}${EXERCISES}`).add}
element={<CreateAppTask />} />
        <Route path={getRoute(`${APPS}${FITNESS}${TASKS}`).add} element={<CreateAppTask />}
/>
        <Route                      path={getRoute(`${APPS}${FITNESS}${LEARNING_PATH}`).add}
element={<CreateAppTask isFitnessTaskLP />} />
        <Route
          path={`${getRoute(`${APPS}${FITNESS}${LEARNING_PATH}`).profile}/:id`}
          element={<LearningPathProfile />}
        />
        <Route
path={`${getRoute(`${APPS}${FITNESS}${LEARNING_PATH}`).edit}/:id`}
          element={<CreateAppTask isFitnessTaskLP />}
        />
        <Route path={ROUTES.PROFILE} element={<Profile />} />
        <Route path={ROUTES.SETTINGS} element={<div>SETTINGS</div>} />
        <Route path={ROUTES.SUPPORT} element={<div>SUPPORT</div>} />
      </Route>
    </Routes>
  </Suspense>
 );
};

//Головний файл системи
const App = () => {
 const isAuthenticated = LocalService.getData(SETTINGS.IS_AUTH);
 return (
  <>
    <ToastContainer position='bottom-right' autoClose={5000} />
    <Routes>
      <Route
        index
        element={<Navigate replace to={isAuthenticated ? ROUTES.DASHBOARD :
ROUTES.AUTH_PAGE.replace('*', '')} />}
      />
```

```
        <Route path={ROUTES.AUTH_PAGE} element={<Auth />} />
        <Route  path={`${ROUTES.PROFILE}/${ROUTES.PREVIEW}`}
          element={
            <Styled.PreviewWrapper>
              <PreviewTheme />
            </Styled.PreviewWrapper>
          }
        />
        <Route path={BUY_LICENSES} element={<BuySubscribe />} />
        <Route
          element={
            <Styled.App>
              <SideBar />
              <Styled.Content>
                <HeaderRouter />
                <main>
                  <MainContentRouter />
                </main>
              </Styled.Content>
            </Styled.App>
          }
          path='/*'
        />
      </Routes>
    </>
  );
};

export default App;
```

Додаток Г

(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ ШКОЛИ

Перелік ілюстративних матеріалів:

1. Слайд 1 – Початковий слайд
2. Слайд 2 – Мета та задачі роботи
3. Слайд 3 – Аналіз існуючих систем
4. Слайд 4 – UML діаграма варіантів використання
5. Слайд 5 – Проєктування архітектури автоматизованої системи
6. Слайд 6 – Технології розробки front-end частини
7. Слайд 7 – Технології розробки та розгортання backend частини
8. Слайд 8 – Сутність «User» в базі даних
9. Слайд 9 – UML діаграма класів
10. Слайд 10 – Розробка програмного забезпечення
11. Слайд 11 – Екранні форми програми
12. Слайд 12 – Розробка програмного забезпечення
13. Слайд 13 – Тестування програмного забезпечення
14. Слайд 14 – Тестування програмного забезпечення
15. Слайд 15 - Економічна частина
16. Слайд 16 - Висновки

Студент групи 2АКІТР-23м

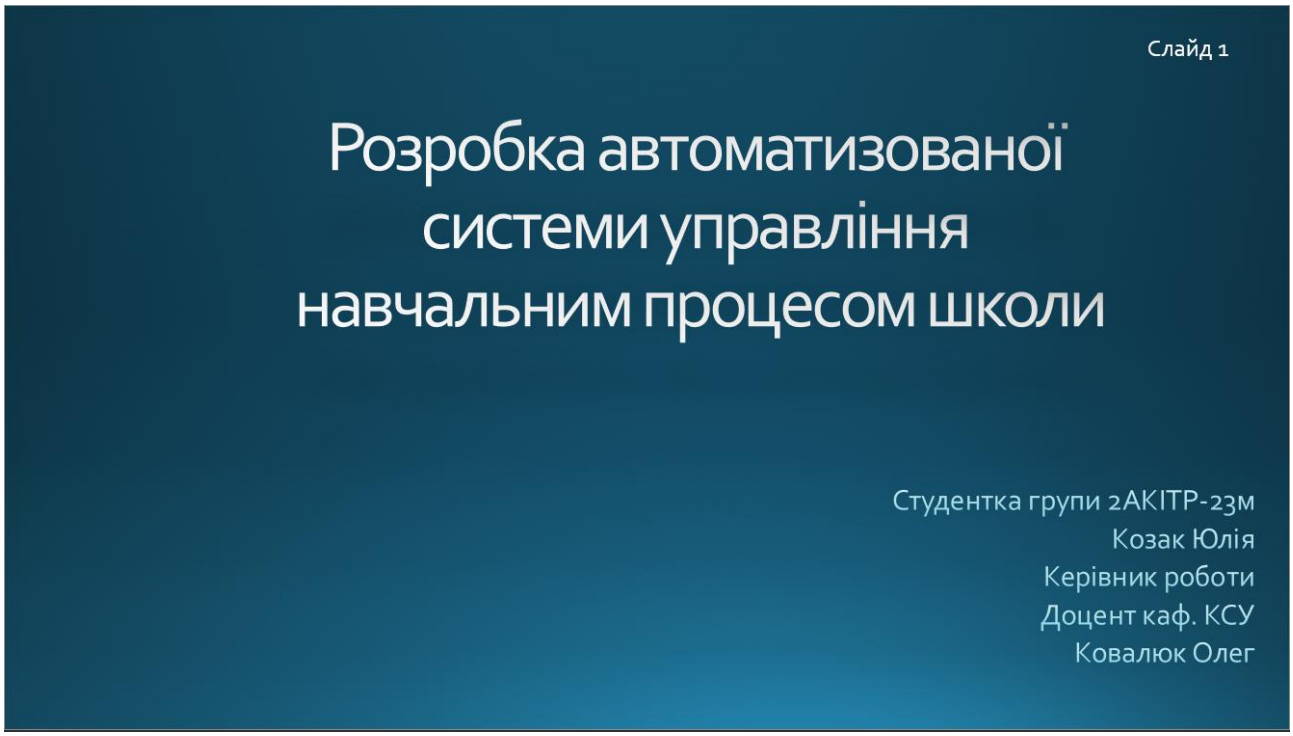
_____ Юлія КОЗАК

Підпис *Ім'я ПРІЗВИЩЕ*

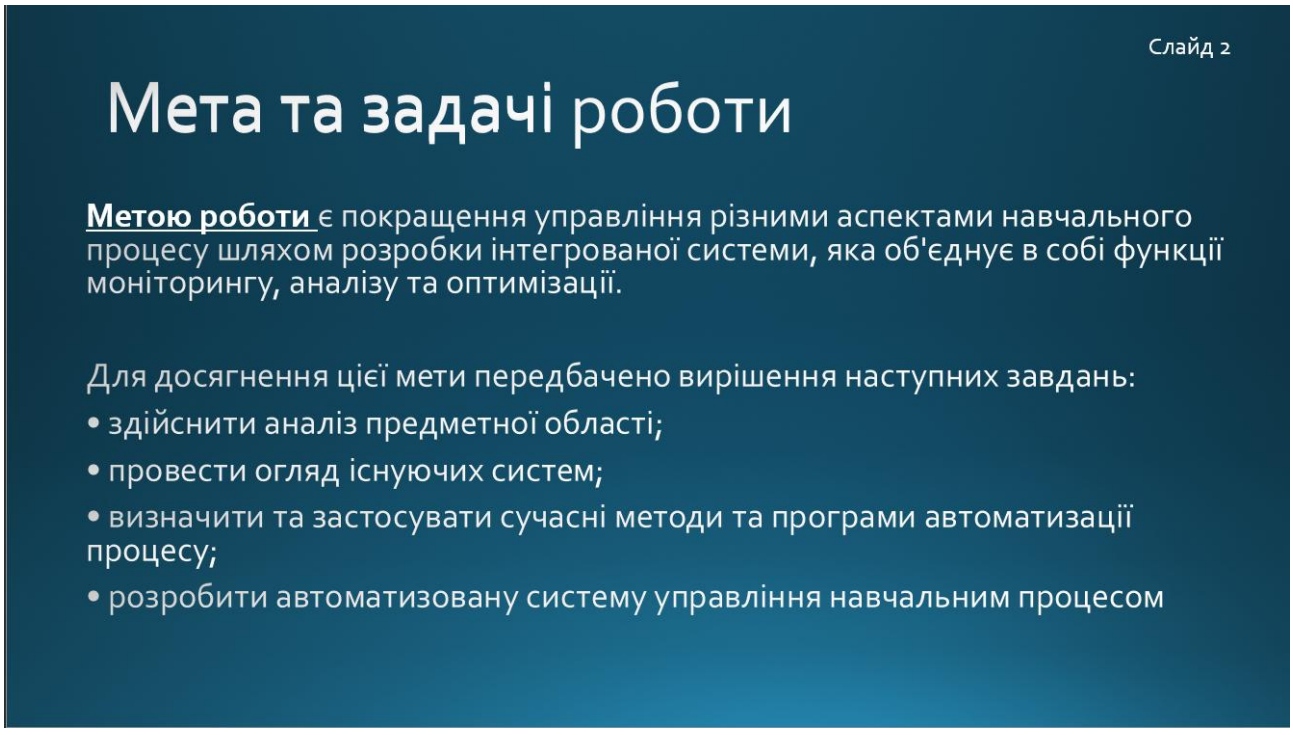
Керівник _____

_____ Олег КОВАЛЮК

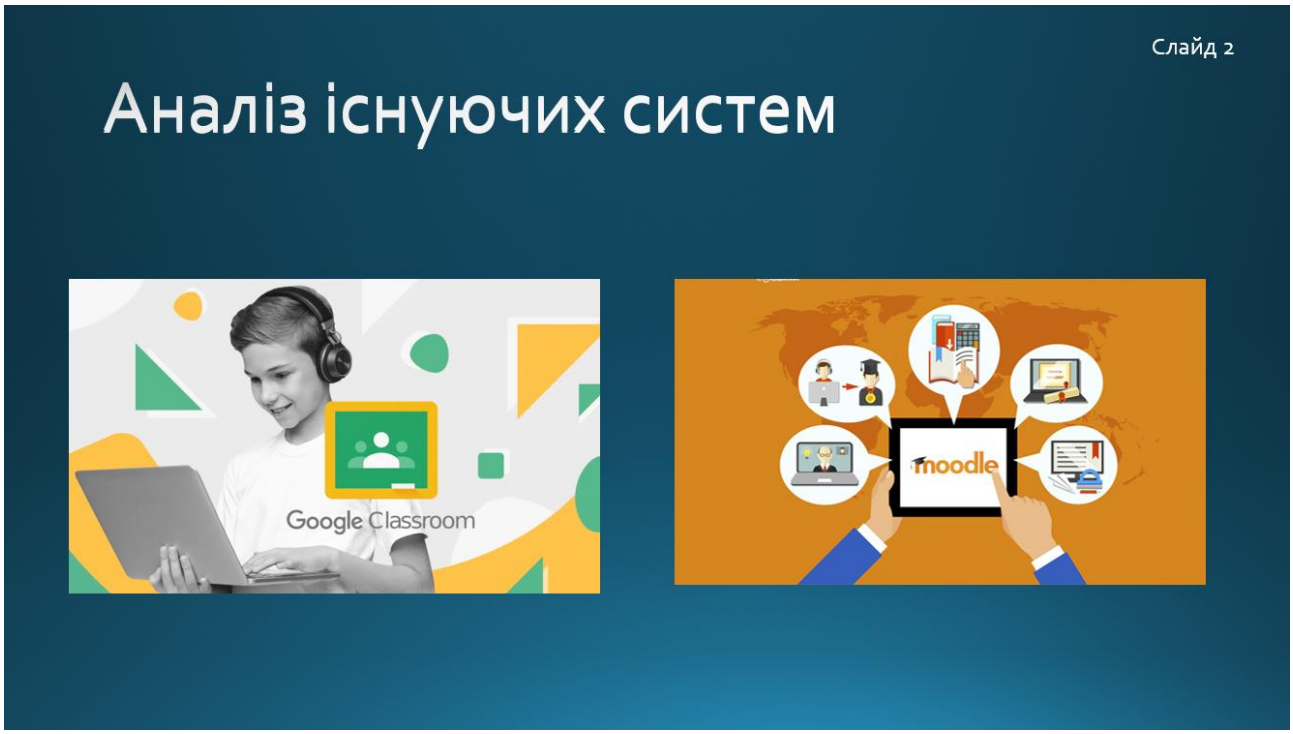
Підпис *Ім'я ПРІЗВИЩЕ*



Слайд 1 – Початковий слайд

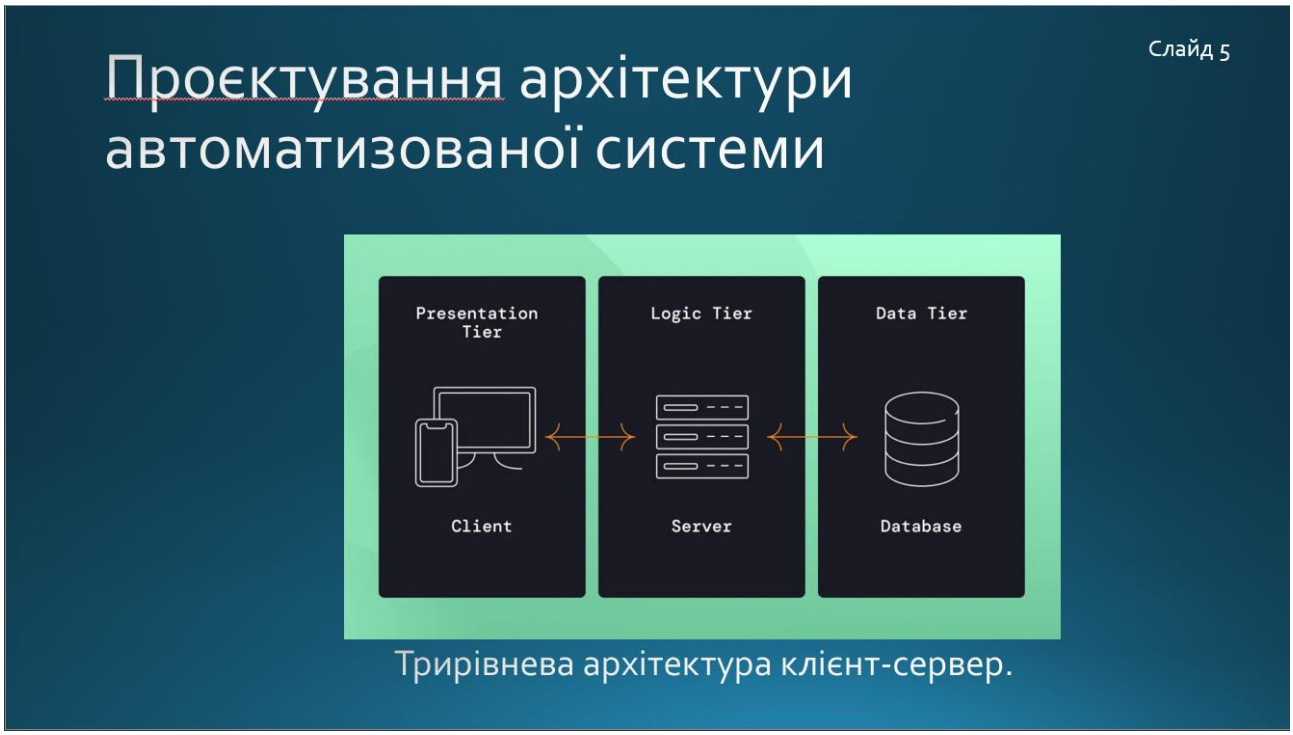


Слайд 2 – Мета та задачі роботи

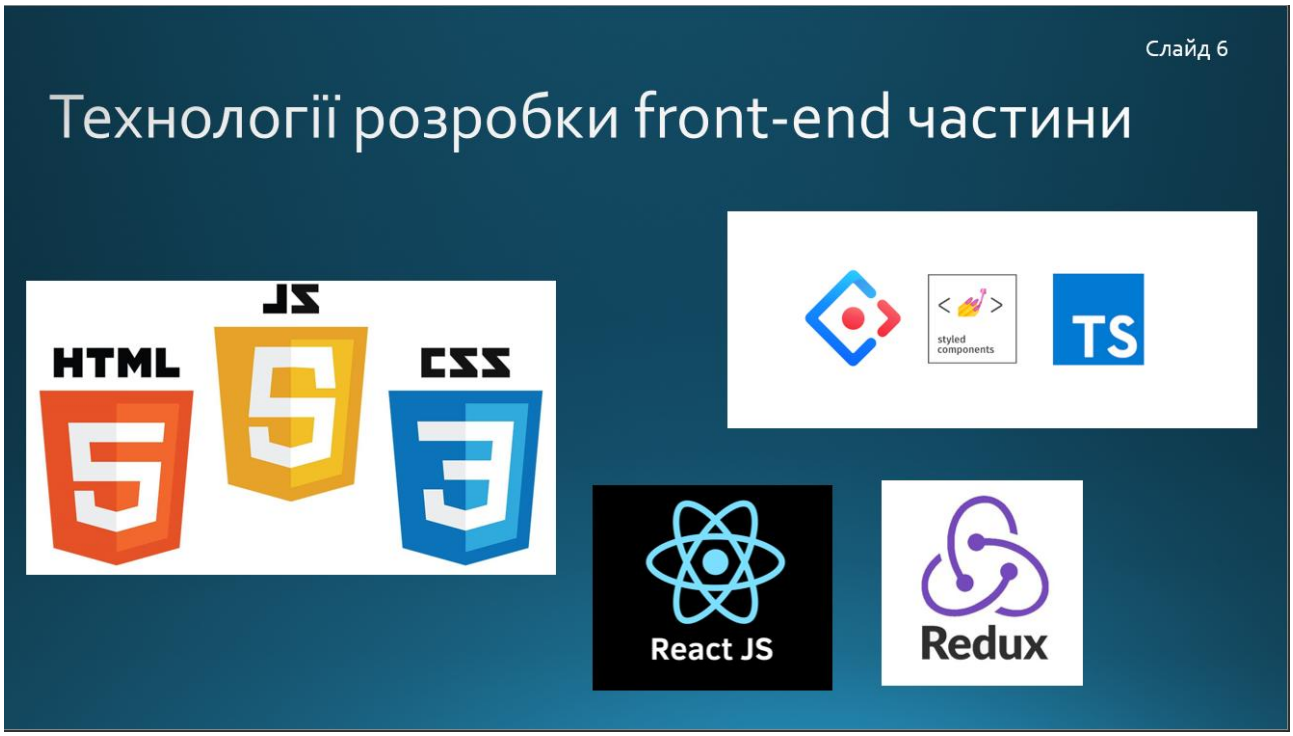


Слайд 3 – Аналіз існуючих систем

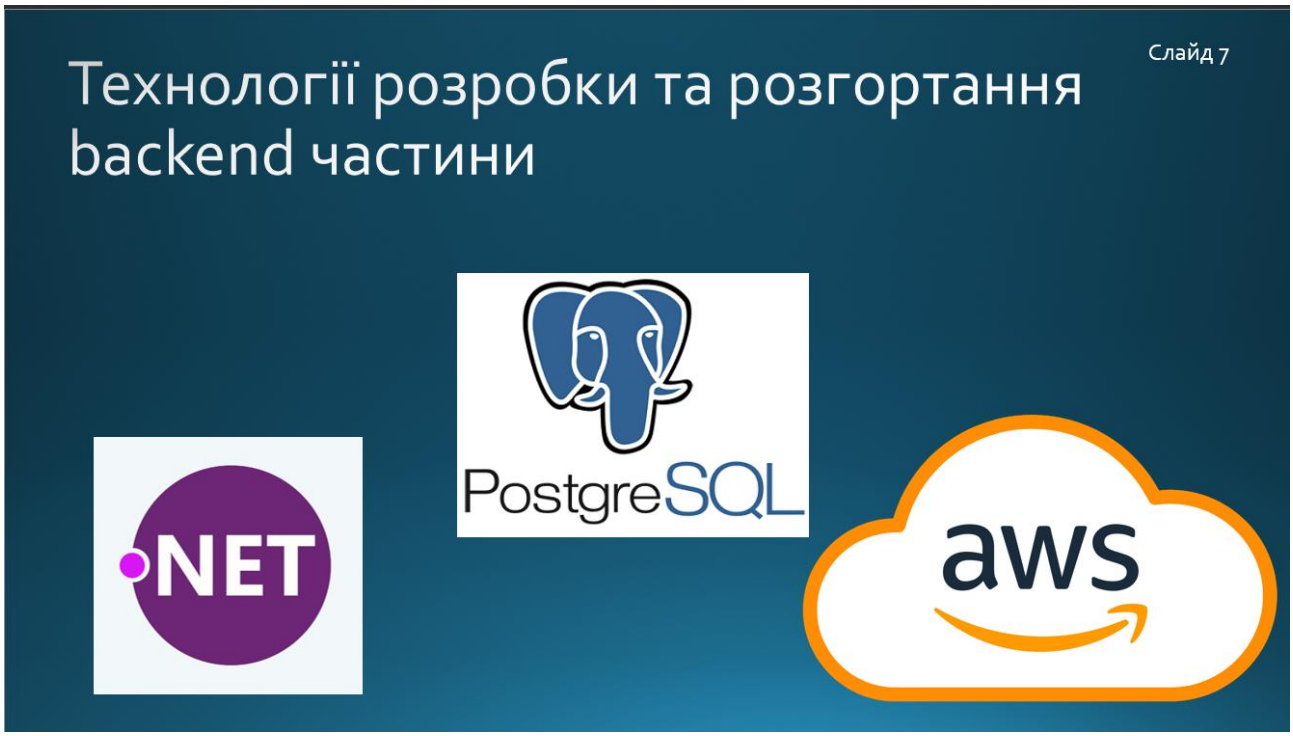
Слайд 4 – UML діаграма варіантів використання



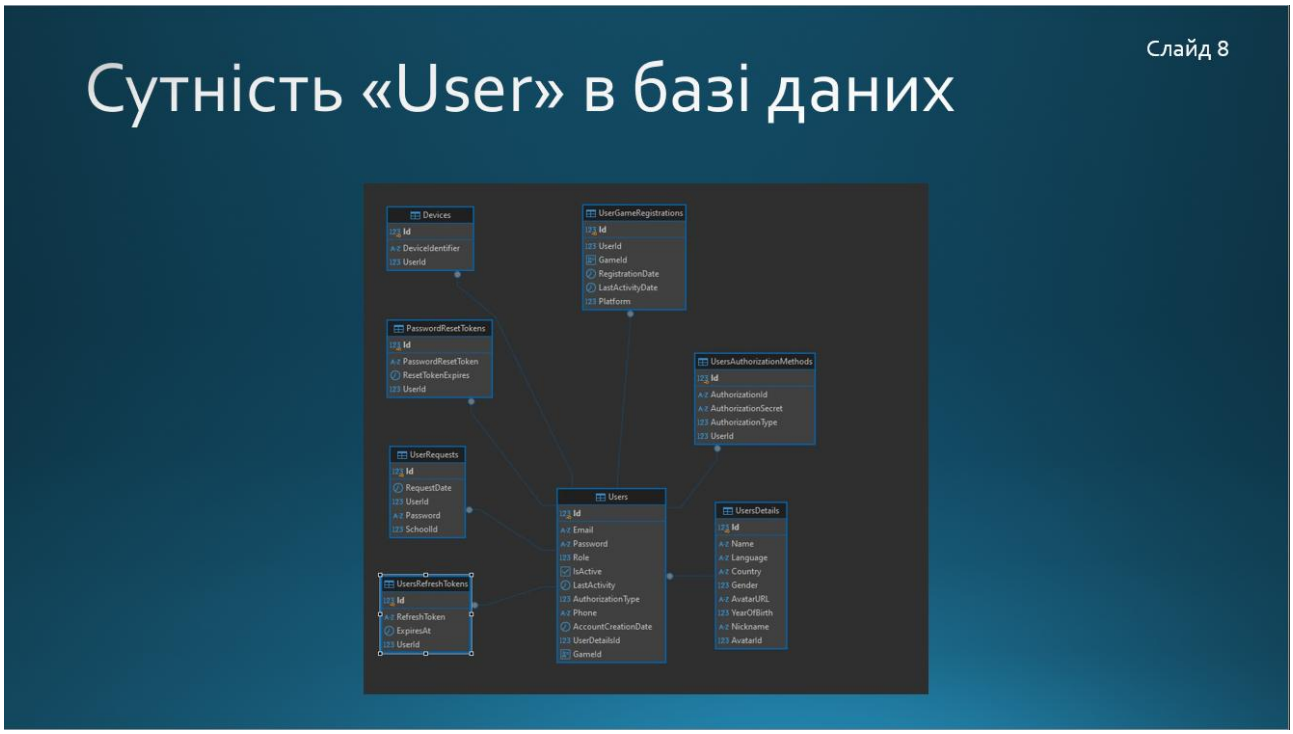
Слайд 5 – Проєктування архітектури автоматизованої системи



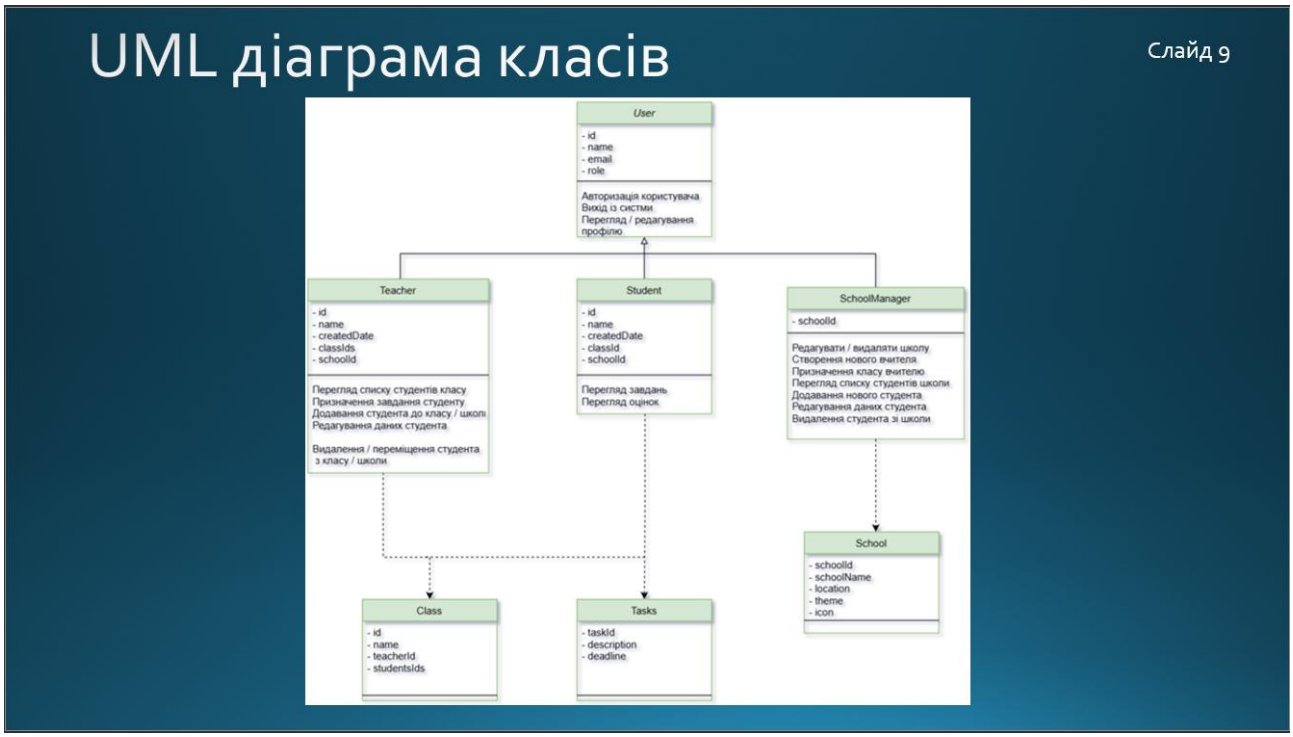
Слайд 6 – Технології розробки front-end частини



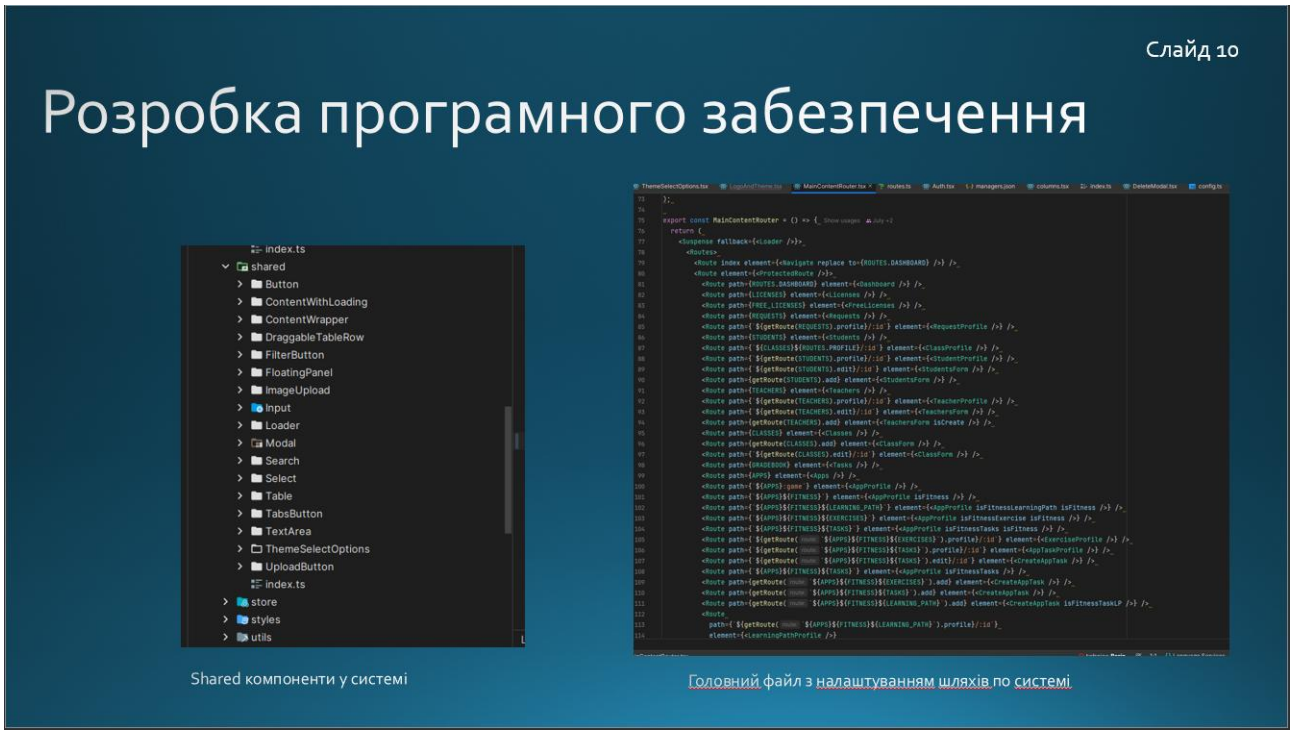
Слайд 7 – Технології розробки та розгортання backend частини



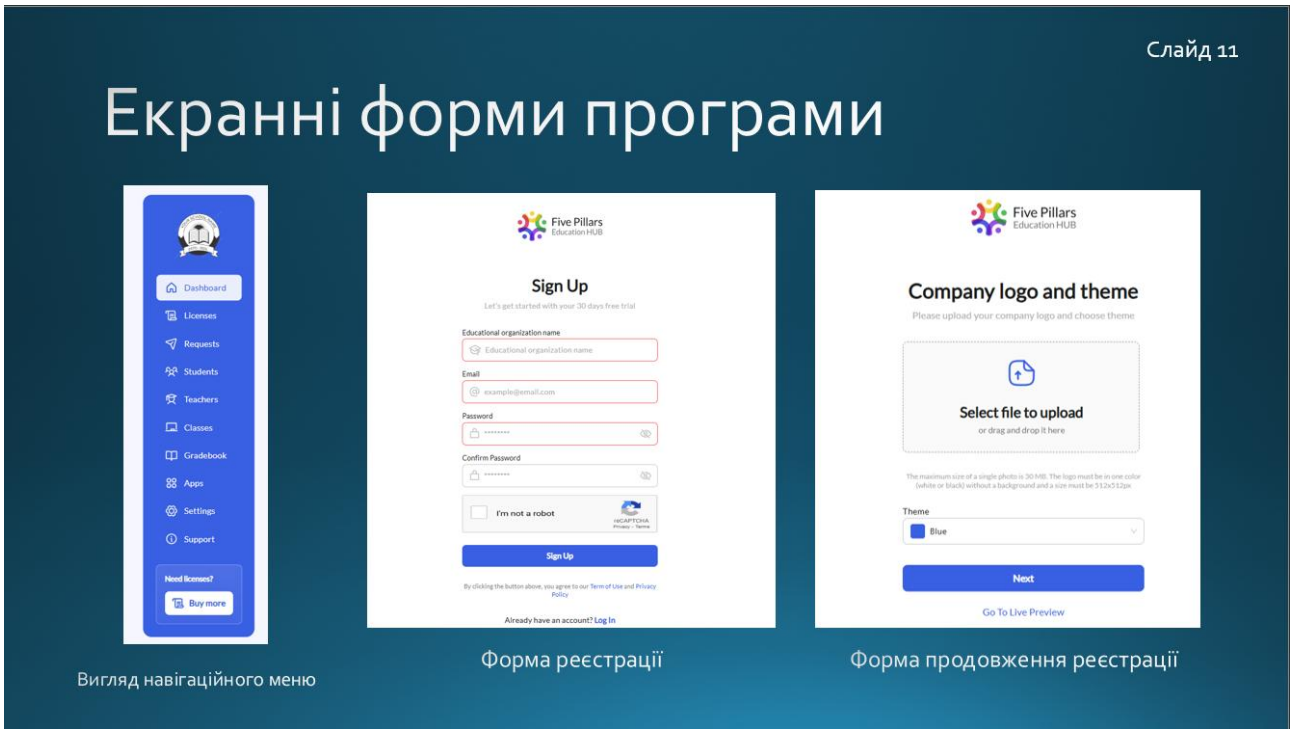
Слайд 8 – Сутність «User» в базі даних



Слайд 9 – UML діаграма класів

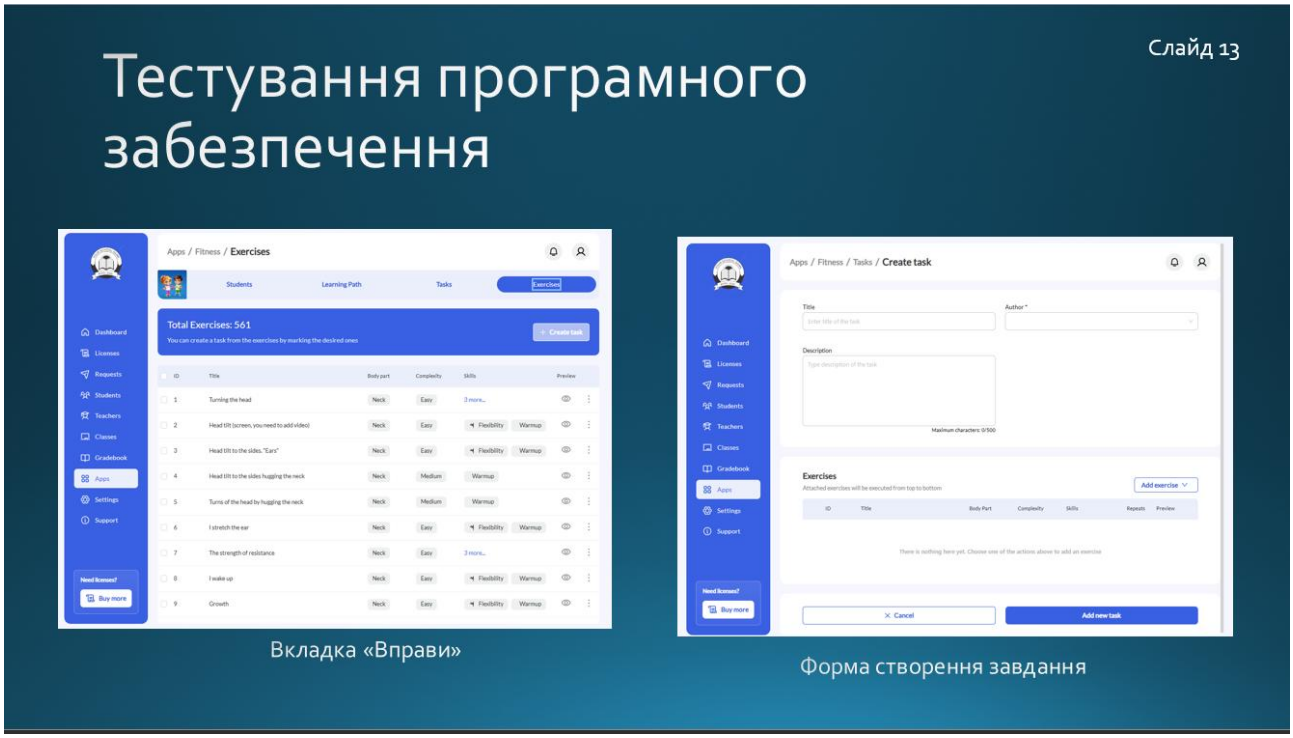


Слайд 10 – Розробка програмного забезпечення

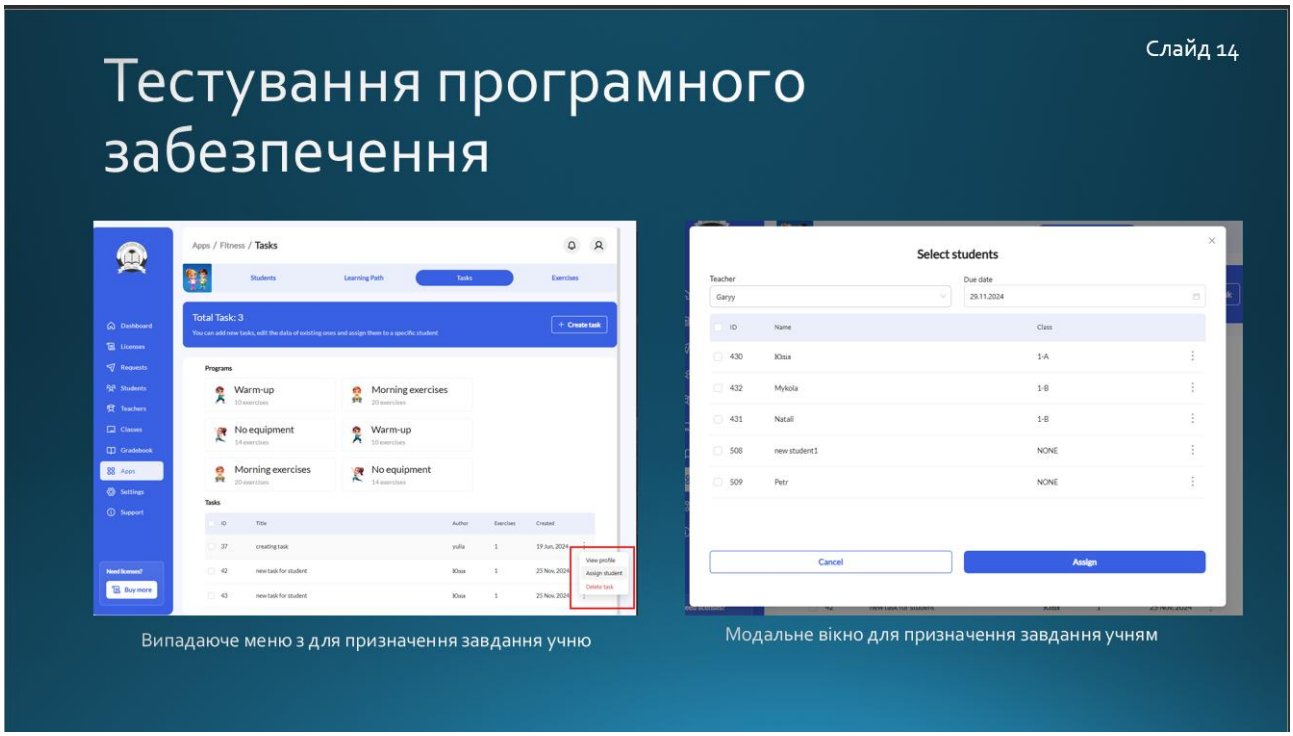


Слайд 11 – Екранні форми програми

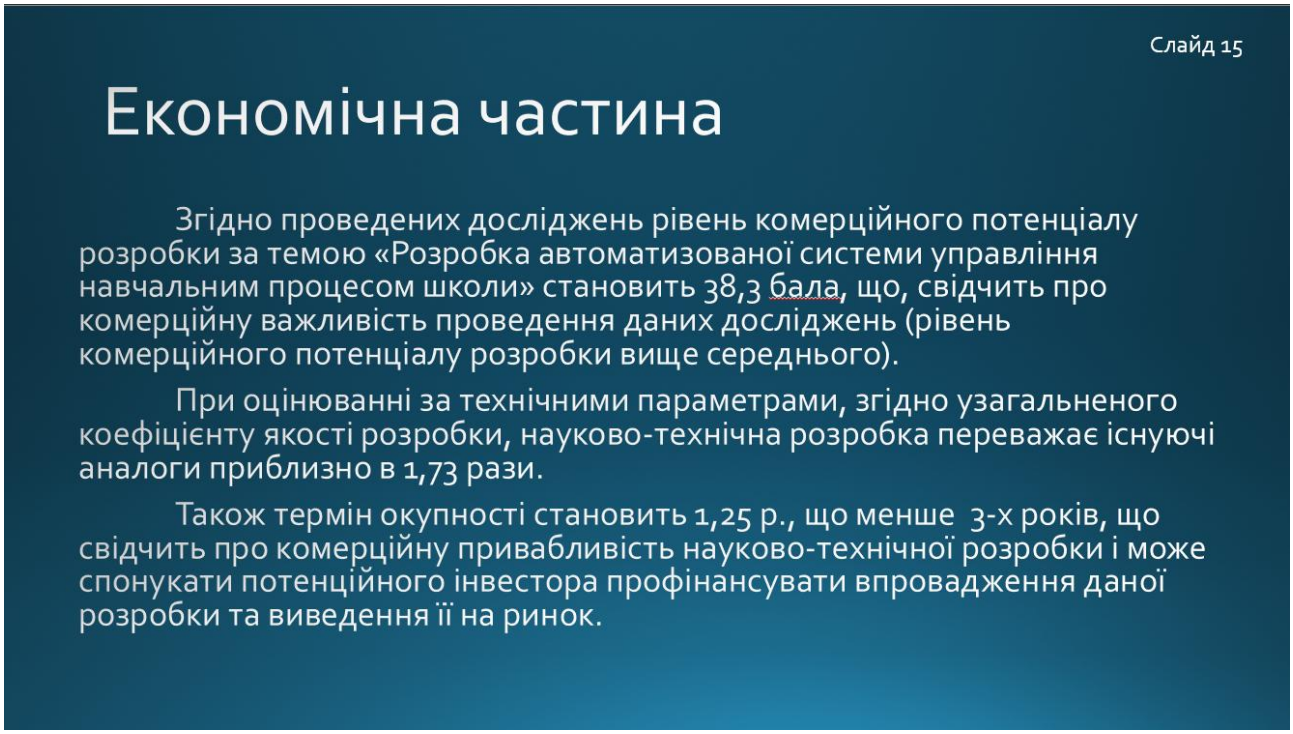
Слайд 12 – Розробка програмного забезпечення



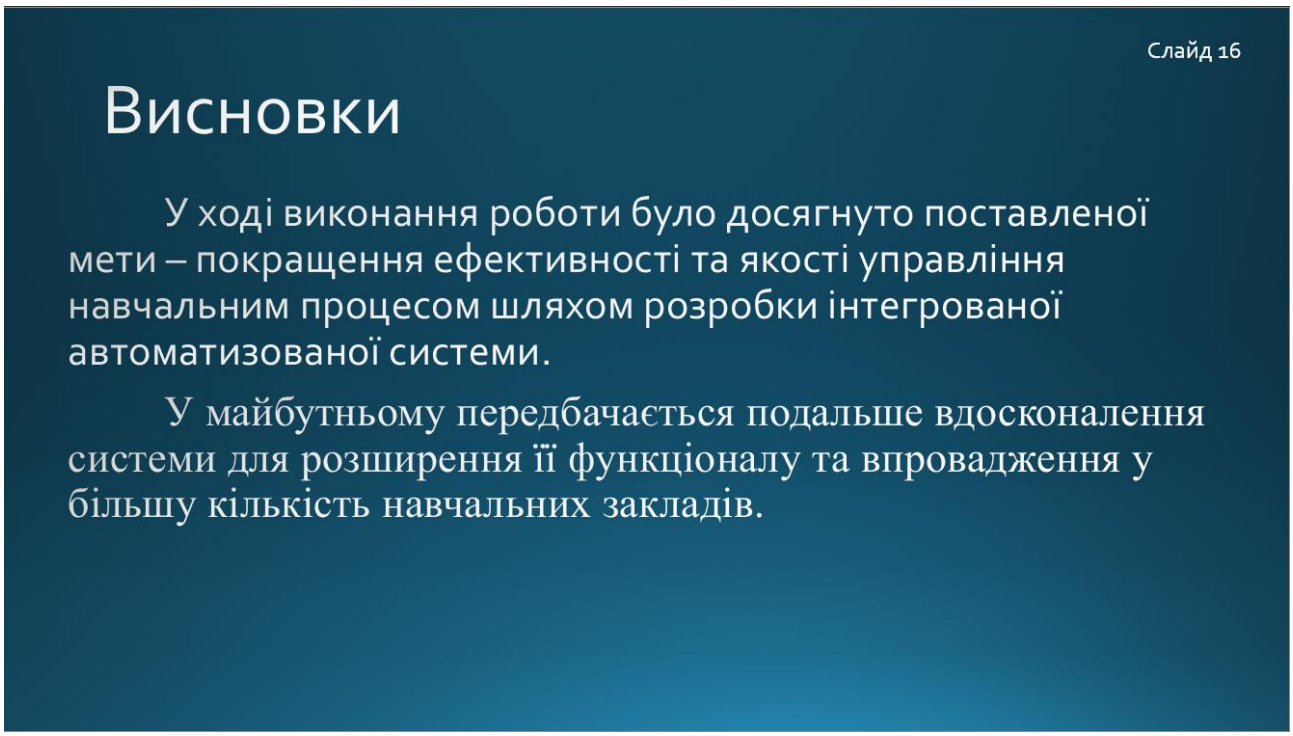
Слайд 13 – Тестування програмного забезпечення



Слайд 14 – Тестування програмного забезпечення



Слайд 15 - Економічна частина



Слайд 16 - Висновки