**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформаційних систем та технологій**

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____2025 р.

# Дипломний проєкт

**на здобуття ступеня бакалавра**
**за освітньо-професійною програмою «Інформаційні управляючі системи та технології»**
**спеціальності 126 «Інформаційні системи та технології»**
**на тему: «Інформаційна система управління навчальним процесом у закладі освіти»**

Виконав:
студент IV курсу, групи ІС-12
Канупа Максим Олександрович _____

Керівник:
д.т.н., доцент кафедри ІСТ,
Поліщук Михайло Миколайович _____

Рецензент:
к.т.н., доцент кафедри ІПІ,
Полупан Юлія Вікторівна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2025 року

**Національний технічний університет України**
**«Київський політехнічний інститут імені Ігоря Сікорського»**
**Факультет інформатики та обчислювальної техніки**
**Кафедра інформаційних систем та технологій**

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2025 р.

# ЗАВДАННЯ

**на дипломний проєкт студенту**
**Канупі Максиму Олександровичу**

1. Тема проєкту «Інформаційна система управління навчальним процесом у закладі освіти», керівник проєкту Поліщук Михайло Миколайович, д.т.н., доцент кафедри ІСТ, затверджені наказом по університету від «23» травня 2025 р. № 1705-с

2. Термін подання студентом проєкту: «9» червня 2025 р.

3. Вихідні дані до проєкту: специфікація освітнього процесу, вимоги до обліку успішності, відвідуваності, формування розкладу та підтримки сертифікатів

4. Зміст пояснювальної записки:

1. Опис предметної області
2. Аналіз існуючих рішень
3. Вибір технологій та проектування системи
4. Розроблення системи
5. Математичне забезпечення

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): схема бази даних, діаграма послідовності

обробки сертифікату, діаграма діяльності процесу виставлення оцінки, діаграма прецедентів.

6. Дата видачі завдання: «17» лютого 2025 р.

Календарний план

| № з/п | Назва етапів виконання дипломного проєкту | Термін виконання етапів проєкту | Примітка |
|---|---|---|---|
| 1 | Аналіз предметної області | 10.05.2025 | |
| 2 | Постановка завдання | 14.05.2025 | |
| 3 | Аналіз існуючих рішень | 15.05.2025 | |
| 4 | Вибір технологій та проектування системи | 17.05.2025 | |
| 5 | Розроблення системи | 19.05.2025 | |
| 6 | Тестування системи | 30.05.2025 | |
| 7 | Виконання графічних документів | 01.06.2025 | |
| 8 | Оформлення пояснювальної записки | 03.06.2025 | |
| 9 | Подання ДП на захист | 09.06.2025 | |

Студент Максим КАНУПА

Керівник Михайло ПОЛІЩУК

# АНОТАЦІЯ

Інформаційна система управління навчальним процесом у закладі освіти.

Проєкт містить 68 с. тексту, 15 рисунків, 2 таблиці, посилання на 26 літературних джерел, додатки та 4 конструкторські документи.

ІНФОРМАЦІЙНА СИСТЕМА УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ, АВТОМАТИЗАЦІЯ, ВЕБ-СЕРВІС, РЕЄСТРАЦІЯ ТА КОНТРОЛЬ УСПІШНОСТІ.

Об’єктом розробки є інформаційна система управління навчальним процесом у закладі освіти.

Мета розробки — підвищення ефективності організації навчального процесу шляхом автоматизації ключових процесів, таких як формування розкладу занять, моніторинг успішності студентів, облік відвідуваності та підтримка результатів неформальної освіти.

У дипломному проєкті розроблено комплексну інформаційну систему, що охоплює управління освітнім процесом, включаючи серверну частину на основі .NET Core та клієнтську частину на Angular з використанням Bootstrap. Значну увагу приділено інтеграції з існуючими системами та забезпеченню захищеної авторизації користувачів за допомогою JWT-токенів. Проведено аналіз існуючих програмних рішень, таких як Moodle, Campus та інші подібні системи, що дозволило визначити унікальні переваги розробленої системи.

Отримані результати можуть бути корисними при реалізації подібних інформаційних систем у закладах освіти різного рівня.

## SUMMARY

Information System for Managing the Educational Process in an Educational Institution.

The project contains 68 pages. text, 15 figures, 2 tables, links to 26 literary sources, annexes and 4 design documents.

Keywords: information system for managing the educational process, automation, web service, registration and performance monitoring.

The purpose of the development is to increase the efficiency of organizing the educational process by automating key processes such as scheduling, monitoring students' academic performance, attendance tracking, and support for informal education achievements.

The graduation project has developed a comprehensive information system covering the management of the educational process, including a backend based on .NET Core and a frontend based on Angular with Bootstrap. Special attention is given to integration with existing systems and ensuring secure user authorization using JWT tokens. An analysis of existing solutions, such as Moodle, Campus, and other similar systems, has been conducted, highlighting the unique advantages of the developed system.

The results obtained can be useful for the implementation of similar information systems in educational institutions of various levels.

| Номер рядка | Формат | Позначення | Найменування | Кільк. аркушів | Номер екзем. | Примітка |
|---|---|---|---|---|---|---|
| 1 | | | Документація загальна | | | |
| 2 | | | | | | |
| 3 | | | Знову розроблена | | | |
| 4 | | | | | | |
| 5 | А4 | ІС12.150БАК.005 ПЗ | Пояснювальна записка | 68 | | |
| 6 | А3 | ІС12.150БАК.005 Д1 | Інформаційна система | 1 | | |
| 7 | | | управління навчальним | | | |
| 8 | | | процесом в закладі освіти. | | | |
| 9 | | | Схема бази даних | | | |
| 10 | А3 | ІС12.150БАК.005 Д2 | Інформаційна система | 1 | | |
| 11 | | | управління навчальним | | | |
| 12 | | | процесом в закладі освіти. | | | |
| 13 | | | Діаграма послідовності | | | |
| 14 | | | обробки сертифікату | | | |
| 15 | А3 | ІС12.150БАК.005 Д3 | Інформаційна система | 1 | | |
| 16 | | | управління навчальним | | | |
| 17 | | | процесом в закладі освіти. | | | |
| 18 | | | Діаграма діяльності процесу | | | |
| 19 | | | виставлення оцінки | | | |
| 20 | А3 | ІС12.150БАК.005 Д4 | Інформаційна система | 1 | | |
| 21 | | | управління навчальним | | | |
| 22 | | | процесом в закладі освіти. | | | |
| 23 | | | Діаграма прецедентів | | | |
| 24 | | | | | | |
| 25 | | | | | | |
| 26 | | | | | | |
| 27 | | | | | | |
| 28 | | | | | | |

| Зм. | Аркуш | № докум. | Підпис | Дата | ІС12.150БАК.005 ТП | | |
|---|---|---|---|---|---|---|---|
| Розроб. | | Канупа М.О. | | | Інформаційна система управління навчальним процесом у закладі освіти. Відомість проєкту | Літ. | Аркуш 1 / Аркушів 1 |
| Керівн. | | Поліщук М.М. | | | | | |
| Затв. | | | | | | КПІ ім. Ігоря Сікорського | |

# Пояснювальна записка
## до дипломного проєкту
## на тему: «Інформаційна система управління навчальним процесом у закладі освіти»

Київ – 2025 року

**ЗМІСТ**

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ....4
ВСТУП....5
1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ....7
1.1 Особливості навчального процесу у закладах освіти....8
1.2 Типові проблеми та обмеження в управлінні навчальним процесом....10
1.3 Постановка задачі....11
1.3.1 Призначення системи....11
1.3.2 Цілі та задачі розробки....12
Висновок до розділу 1....13
2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ....14
2.1 Moodle....14
2.2 Campus KPI....15
2.3 Google Classroom....16
2.4 Порівняльний аналіз систем....18
Висновок до розділу 2....19
3 ВИБІР ТЕХНОЛОГІЙ ТА ПРОЄКТУВАННЯ СИСТЕМИ....20
3.1 Вибір мови програмування....20
3.2 Вибір середовища розробки....21
3.3 Вибір фреймворку бекенду....23
3.4 Вибір бази даних....25
3.5 Архітектура системи....26
3.6 Вибір фронтенд-фреймворку....27
3.7 Проєктування системи....28
Висновок до розділу 3....29
4 РОЗРОБЛЕННЯ СИСТЕМИ....31
4.1 Структура системи....31
4.2 Функціональна модель системи....33

4.3 Модель бази даних ..... 35
4.4 Передавання та обробка даних ..... 36
4.5 Архітектура програмного забезпечення ..... 37
4.6 Діаграми класів ..... 42
4.7 Керівництво користувача ..... 43
4.8 Безпека системи ..... 49
4.9 Тестування системи ..... 51
4.9.1 Юніт-тести ..... 51
4.9.2 Ручне тестування ..... 53
4.9.3 Навантажувальне тестування ..... 54
Висновок до розділу 4 ..... 56
5 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ ..... 57
5.1 Змістовна постановка задачі ..... 57
5.2 Математична постановка задачі ..... 58
5.3 Обґрунтування методу розв’язання ..... 59
5.4 Опис методу розв’язання ..... 61
Висновок до розділу 5 ..... 62
ВИСНОВКИ ..... 64
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ..... 66
ДОДАТОК А ..... 69

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – База даних.

API – Application Programming Interface – прикладний програмний інтерфейс.

DDD – Domain-Driven Design – розробка, орієнтована на предметну область.

EF Core – Entity Framework Core – ORM для роботи з реляційними базами даних у C#.

ORM – Object-Relational Mapping – це технологія, яка дозволяє працювати з базою даних як з об'єктами в коді, автоматично перетворюючи дані між об'єктами програми та реляційними таблицями.

HTTP – HyperText Transfer Protocol – протокол передачі гіпертекстових документів.

IDE – Integrated Development Environment – інтегроване середовище розробки.

JWT – JSON Web Token – токен безпечного обміну інформацією між клієнтом і сервером.

MVC – Model-View-Controller – модель-вид-контролер, архітектурний патерн.

# ВСТУП

У сучасних умовах інтенсивного розвитку інформаційних технологій та впровадження цифрових рішень у всі сфери життя особливого значення набуває питання ефективної організації навчального процесу в закладах освіти. Зростання обсягів інформації, потреба у швидкому обміні даними, а також необхідність підтримки індивідуальних траєкторій навчання створюють об'єктивну потребу у впровадженні сучасних інформаційних систем. Саме тому тема дипломного проєкту є актуальною та обґрунтованою.

Аналіз сучасного стану проблеми свідчить про те, що в багатьох закладах освіти досі використовуються застарілі або розрізнені рішення для обліку навчальних даних, формування розкладу занять та моніторингу результатів студентів. Популярні системи, такі як Moodle чи CAMPUS KPI (із додатковими сервісами на кшталт розкладів занять та вибіркових дисциплін), забезпечують певний рівень автоматизації, проте мають обмеження щодо інтеграції з іншими освітніми процесами та можливості масштабування. Окрім того, ці рішення не завжди відповідають потребам конкретного навчального закладу, що створює передумови для розробки більш гнучких та адаптивних систем.

Мета дипломного проєкту полягає у створенні інформаційної системи, що забезпечить автоматизацію ключових процесів управління навчальним процесом, включаючи формування та адміністрування розкладу занять, облік відвідуваності студентів, моніторинг їхньої успішності, а також інтеграцію даних неформальної освіти (сертифікати з Coursera, Prometheus та інші). Основні завдання, які необхідно вирішити для досягнення поставленої мети, включають:

– розробку структури та функціональної моделі інформаційної системи;

– визначення вимог до технічного, програмного, інформаційного та математичного забезпечення;

– впровадження безпечної системи авторизації та аутентифікації користувачів;

– інтеграцію з існуючими освітніми платформами та сервісами;

– тестування працездатності та надійності системи.

Обґрунтування основних проєктних рішень базується на використанні сучасних технологій розробки вебзастосунків: серверна частина реалізована на основі .NET Core із застосуванням C# та Entity Framework Core, а клієнтська – за допомогою Angular із використанням Bootstrap для створення адаптивного та зручного інтерфейсу. Для забезпечення безпечної роботи системи впроваджено механізми авторизації за допомогою JWT-токенів, а також передбачено можливість масштабування сервісу для подальшого розвитку.

Результати проєкту мають широкий спектр можливих застосувань. Запропонована система може бути впроваджена як у вищих навчальних закладах, так і в коледжах, професійно-технічних училищах, ліцеях, гімназіях та школах. Система здатна підвищити ефективність управління навчальним процесом, зменшити навантаження на адміністративний персонал, забезпечити студентам і викладачам зручний доступ до актуальної інформації, а також сприяти підвищенню якості освіти загалом.

Дипломний проєкт складається з наступних розділів: вступ, основні розділи, висновки, список використаних джерел із 26 найменувань, додатки. Графічна частина включає 15 рисунків, 2 таблиці, 4 конструкторські документи. Загальний обсяг пояснювальної записки складає 68 сторінок.

# 1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Сучасний освітній процес у закладах освіти є складною багаторівневою системою, яка включає планування навчального процесу, організацію розкладу занять, облік відвідуваності студентів, контроль їхньої успішності, а також підтримку результатів неформальної освіти. У багатьох закладах ці процеси досі виконуються частково вручну або за допомогою розрізнених програмних рішень, що не дозволяє досягти належного рівня ефективності.

Типовий освітній процес починається з формування розкладу занять для кожної групи студентів та викладачів, з урахуванням навчальних програм і доступності ресурсів. Далі здійснюється моніторинг відвідуваності занять та успішності студентів шляхом фіксації результатів поточного та підсумкового контролю. Особлива увага приділяється забезпеченню можливості обліку сертифікатів і результатів курсів, отриманих на зовнішніх платформах (Coursera, Prometheus), які стають все більш популярними у підготовці фахівців [24].

Важливою частиною діяльності є управління доступом до системи: для кожного користувача створюється профіль із розподілом ролей (студент, викладач, адміністратор). Система має забезпечувати безпечну та зручну роботу користувачів, включаючи перегляд розкладу занять, оцінок, вибір дисциплін тощо.

Крім основних функцій, важливою складовою сучасного навчального процесу є ефективна комунікація між викладачами, студентами та адміністрацією навчального закладу. Це включає своєчасне інформування про зміни у розкладі занять, сповіщення про важливі події, новини, дедлайни та інші аспекти навчальної діяльності. Такі сповіщення можуть надсилатись як через систему, так і за допомогою інтеграції із зовнішніми месенджерами або електронною поштою.

Ще одним аспектом діяльності є інтеграція освітнього процесу з іншими системами управління (наприклад, фінансовими, кадровими), що дозволяє забезпечити комплексну автоматизацію освітньої діяльності. Це забезпечує безперервний обмін даними, спрощує адміністрування та підвищує загальну ефективність роботи закладу.

В умовах швидкого розвитку технологій також важливим є забезпечення гнучкості системи для можливості додавання нових функцій та адаптації до змін у законодавстві, навчальних програмах або потребах користувачів. Система повинна підтримувати масштабованість, інтеграцію з новими сервісами, а також бути сумісною з мобільними пристроями та хмарними середовищами.

Отже, сучасний навчальний процес у закладах освіти вимагає впровадження автоматизованих систем управління, що забезпечують гнучкість, ефективність та безпеку. Враховуючи зростаючу роль цифрових технологій та необхідність інтеграції із зовнішніми освітніми платформами, впровадження такої інформаційної системи є важливим кроком для підвищення якості освітнього процесу, зручності для користувачів та загальної продуктивності навчального закладу.

## 1.1 Особливості навчального процесу у закладах освіти

Навчальний процес у закладах освіти складається з комплексу взаємопов'язаних заходів, які забезпечують передачу знань, розвиток компетенцій і формування особистісних якостей студентів. Основними етапами навчального процесу є планування розкладу занять, проведення навчальних заходів (лекції, практичні та лабораторні заняття), оцінювання знань студентів, облік відвідуваності та формування підсумкових результатів. Для ефективної організації навчання заклади освіти повинні враховувати індивідуальні траєкторії студентів, наявність спеціалізованих курсів, вибіркових дисциплін, можливість інтеграції результатів неформальної освіти.

Особливу увагу в навчальному процесі приділяють створенню розкладу занять. Цей процес є складним через необхідність врахування великої кількості обмежень: наявність викладачів, доступність аудиторій, конфлікти між предметами, розподіл за групами, поєднання дисциплін із різними формами навчання (очна, дистанційна). У багатьох закладах освіти цей процес досі здійснюється вручну або із використанням окремих інструментів, що не

забезпечують повної автоматизації та часто призводять до помилок і конфліктів у розкладі [24].

Ще однією важливою складовою навчального процесу є моніторинг успішності студентів. Це включає оцінювання знань через поточні та підсумкові контрольні заходи, формування заліково-екзаменаційної відомості, облік отриманих сертифікатів із додаткових освітніх платформ. На практиці облік успішності часто ведеться у вигляді паперових журналів або локальних електронних таблиць, що не дозволяє своєчасно отримувати зведені звіти, аналізувати динаміку навчання та визначати проблемні зони.

Важливою складовою навчального процесу є також облік відвідуваності студентів. Відсутність автоматизованих інструментів обліку призводить до значних витрат часу викладачів та адміністраторів, підвищує ризик помилок і неточностей. Впровадження системи, що дозволяє вести облік відвідуваності в режимі реального часу, автоматично формувати звіти та аналізувати дані, є важливим кроком до підвищення прозорості та об’єктивності оцінювання дисциплінованості студентів.

Ще одна характерна риса сучасного навчального процесу – потреба у врахуванні результатів неформальної освіти. Багато студентів додатково навчаються на онлайн-платформах, таких як Coursera, Prometheus, отримують сертифікати та проходять курси підвищення кваліфікації. Ці результати часто залишаються поза межами офіційного навчального процесу, хоча можуть мати суттєве значення для оцінювання загальної підготовки студента. Інформаційна система має забезпечити можливість інтеграції таких даних у загальний облік успішності [22].

Таким чином, навчальний процес у закладах освіти є багатогранним та потребує ретельної організації, яка має враховувати велику кількість змінних та обмежень. Використання сучасної інформаційної системи дозволить автоматизувати рутинні завдання, зменшити навантаження на викладачів і адміністрацію, забезпечити точність даних, оперативність їх оновлення та підвищення прозорості управління навчальною діяльністю.

## 1.2 Типові проблеми та обмеження в управлінні навчальним процесом

Однією з головних проблем є відсутність інтегрованих інформаційних систем, які б дозволяли ефективно об’єднати всі дані про навчальний процес в єдину базу. У багатьох навчальних закладах інформація про розклади, успішність, відвідуваність студентів та інші аспекти ведеться окремо, у різних форматах – від паперових журналів до електронних таблиць або локальних баз даних. Це призводить до дублювання даних, помилок при введенні інформації, складнощів у доступі до актуальної інформації, що уповільнює роботу адміністрації та викладачів.

Ще одна поширена проблема – складність формування розкладу занять. Врахування великої кількості факторів, таких як доступність аудиторій, графіки викладачів, навчальні плани груп, призводить до високої ймовірності виникнення конфліктів у розкладі. При відсутності автоматизованих інструментів цей процес часто займає багато часу, потребує значних людських ресурсів і не гарантує повної відсутності помилок. Як наслідок, студенти та викладачі можуть стикатися з накладенням занять або відсутністю вільних аудиторій [24].

Окремим викликом є моніторинг успішності студентів. Багато навчальних закладів все ще використовують паперові журнали для обліку оцінок або несистематизовані електронні засоби, які не забезпечують зручного доступу до статистики та звітів. Це ускладнює відстеження динаміки навчання, швидке виявлення проблем у підготовці студентів, створення зведених звітів для адміністрації та акредитаційних органів. Відсутність єдиної системи унеможливлює оперативний аналіз навчального процесу та прийняття ефективних управлінських рішень.

Ще одним значущим обмеженням є неефективний облік відвідуваності занять. Традиційно цей процес здійснюється вручну, шляхом ведення паперових журналів або списків, що займає час і схильне до помилок. Відсутність інтеграції даних про відвідуваність із загальною системою управління навчанням ускладнює

отримання повної картини про залученість студентів та своєчасне реагування на проблеми [22].

Не менш важливою проблемою є недостатня інтеграція неформальної освіти в загальний навчальний процес. Студенти часто додатково навчаються на онлайн-курсах, здобувають сертифікати, проходять практичні стажування, однак ці досягнення залишаються поза увагою офіційних систем навчального обліку. Це призводить до неповного урахування навчальної активності студентів і не дозволяє адміністрації отримати повну картину про підготовку здобувачів освіти.

Також слід зазначити недостатній рівень цифрової компетентності частини персоналу закладів освіти. Впровадження нових систем потребує не лише технічної готовності, але й навчання користувачів, формування навичок роботи з електронними інструментами. Без належної підготовки впровадження нових рішень може зустріти опір або бути неефективним.

Таким чином, типові проблеми та обмеження управління навчальним процесом охоплюють організаційні, технічні та кадрові аспекти. Їх вирішення потребує впровадження комплексних інформаційних систем, що забезпечують інтеграцію даних, автоматизацію процесів, підвищення точності та доступності інформації, а також навчання користувачів для ефективного використання нових технологій.

## 1.3 Постановка задачі

### 1.3.1 Призначення системи

Розроблювана інформаційна система призначена для автоматизації управління навчальним процесом у закладах освіти. Основними об'єктами автоматизації є:

– формування та управління розкладом занять;

– облік відвідуваності та оцінок студентів;

– підтримка результатів неформальної освіти (сертифікати Coursera, Prometheus);

– забезпечення адміністрування користувачів та їхніх ролей.

Система має на меті полегшити роботу адміністративного та викладацького персоналу, забезпечити студентів зручним інтерфейсом для взаємодії з навчальним процесом та підвищити загальну ефективність освітньої діяльності.

Інформаційна система орієнтована на комплексне управління навчальним процесом, що включає моніторинг успішності студентів, формування розкладу, контроль відвідуваності та управління навчальними матеріалами. Це дозволяє значно знизити адміністративне навантаження та мінімізувати ризик помилок у документації, пов'язаних з ручною обробкою даних.

Однак ключовою особливістю системи є можливість інтеграції з популярними платформами неформальної освіти, такими як Coursera та Prometheus, що дозволяє враховувати результати додаткового навчання студентів. Таким чином, система сприяє створенню єдиного інформаційного середовища, де об'єднані аспекти формальної і неформальної навчальної діяльності.

### 1.3.2 Цілі та задачі розробки

Метою розробки є підвищення ефективності організації навчального процесу в закладі освіти за допомогою комплексної інформаційної системи.

Для досягнення цієї мети необхідно розробити архітектуру системи, що забезпечує масштабованість і надійність. Слід реалізувати серверну частину на базі .NET Core, мовою C#, з використанням ASP.NET Core як основного фреймворку та Entity Framework Core для роботи з базою даних. Потрібно створити клієнтську частину на Angular з використанням Bootstrap для формування адаптивного інтерфейсу. Система повинна забезпечувати безпечну аутентифікацію користувачів через JWT. Також слід реалізувати модулі управління розкладом, обліку відвідуваності та оцінок, підтримки вибору дисциплін, обробки сертифікатів. Важливо інтегрувати систему з існуючими освітніми платформами для обміну даними. Нарешті, необхідно провести тестування та валідацію функціоналу для забезпечення стабільної роботи системи [12, 11, 13].

Крім того, розробка інформаційної системи передбачає створення дружнього інтерфейсу користувача для забезпечення зручності роботи студентів, викладачів та адміністративного персоналу. Особлива увага приділяється забезпеченню інтуїтивної навігації, доступності основних функцій, а також підтримці мобільних пристроїв для полегшення доступу до інформації в будь-який час.

Також важливо врахувати можливість подальшого розширення функціоналу системи. Вона повинна бути розроблена таким чином, щоб легко інтегрувати додаткові сервіси, включаючи нові модулі для покращення обліку результатів неформальної освіти, аналізу успішності студентів та автоматизації процесів звітності. Реалізація цих можливостей дозволить закладу освіти ефективно адаптувати систему під потреби сучасного навчального середовища та забезпечить конкурентні переваги серед інших освітніх установ.

Висновок до розділу 1

У цьому розділі було описано предметну область дослідження, визначено основні процеси діяльності в освітньому закладі та сформульовано постановку задачі дипломного проєкту. Розробка інформаційної системи управління навчальним процесом дозволить автоматизувати ключові освітні процеси, підвищити ефективність управління, забезпечити прозорість та зручність доступу до навчальної інформації для всіх учасників освітнього процесу. Система також надасть можливість розширення, що робить її гнучкою та придатною для широкого застосування в закладах освіти різного рівня.

# 2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

У сучасному освітньому середовищі ефективне управління навчальним процесом є критично важливим для забезпечення якості освіти. Існує низка систем управління навчанням (Learning Management Systems, LMS), які широко використовуються в закладах освіти для автоматизації та оптимізації навчального процесу. Серед них особливо виділяються такі платформи, як Moodle, Campus KPI та Google Classroom [26].

## 2.1 Moodle

Moodle — це відкрита платформа для управління навчанням, яка дозволяє створювати персоналізовані онлайн-курси та забезпечує гнучкість у навчальному процесі. Основні функції Moodle включають:

– створення та управління курсами: Moodle дозволяє викладачам створювати курси з використанням різноманітних ресурсів, таких як відео, тести, завдання та форуми;

– оцінювання та зворотний зв'язок: система забезпечує інструменти для оцінювання студентів, включаючи журнали оцінок, рубрики та можливість надання зворотного зв'язку;

– мобільний доступ: Moodle має мобільний додаток, що дозволяє студентам отримувати доступ до курсів та завдань з мобільних пристроїв.

Завдяки відкритому коду та широким можливостям налаштування, Moodle є популярним вибором для багатьох освітніх установ по всьому світу [20].

Скріншот з веб-інтерфейсу Moodle зображений на рисунку 2.1.

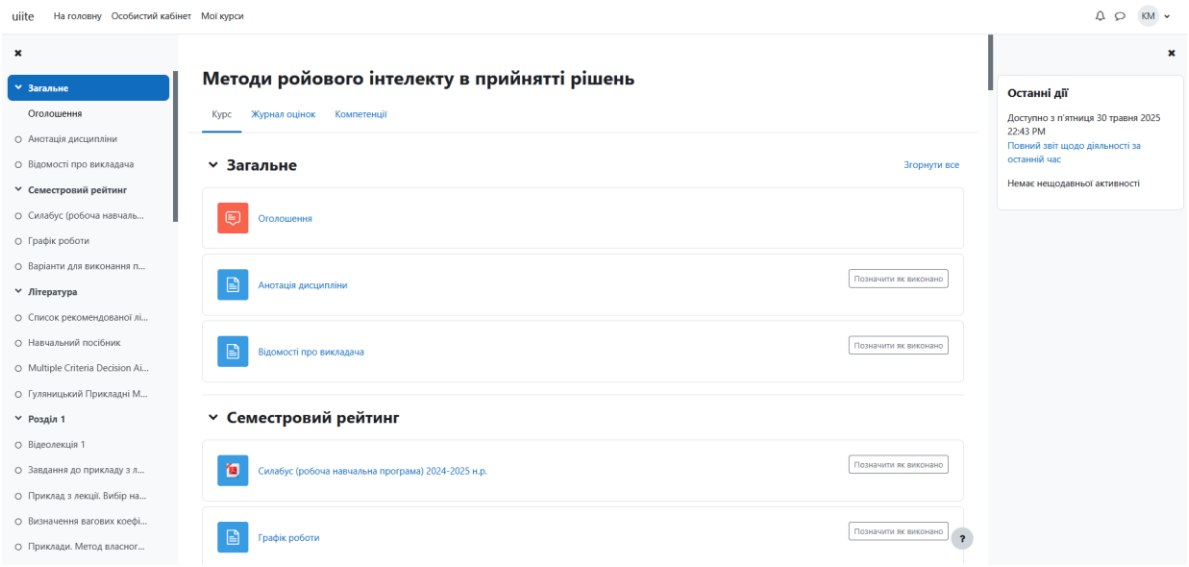


Рисунок 2.1 – Скріншот Moodle

Проте, незважаючи на численні переваги Moodle, існують і певні недоліки. Одним із основних є обмежена підтримка автоматизації процесів, таких як автоматичне завантаження та збереження сертифікатів з зовнішніх платформ, наприклад, Coursera або Prometheus [25, 17]. Для реалізації подібного функціоналу потрібне додаткове налаштування або інтеграція з іншими сервісами, що може ускладнити впровадження.

## 2.2 Campus KPI

Campus KPI — це система управління навчальним процесом, розроблена спеціально для Київської Політехніки. Основні функції Campus KPI включають:

– підтримку різних систем оцінювань: система дозволяє відстежувати показники успішності студентів (календарні контролі, поточні рейтинги успішності, сесії);

– матеріали курсів: Campus забезпечує інструменти для надання навчальних матеріалів (РСО, методичні вказівки) та потрібних для навчання ресурсів.

Campus орієнтований на забезпечення управління навчальним процесом та в основному використовується лише для контролю успішності. Скріншот електронного кампусу КПІ з новим дизайном зображено на рисунку 2.2.

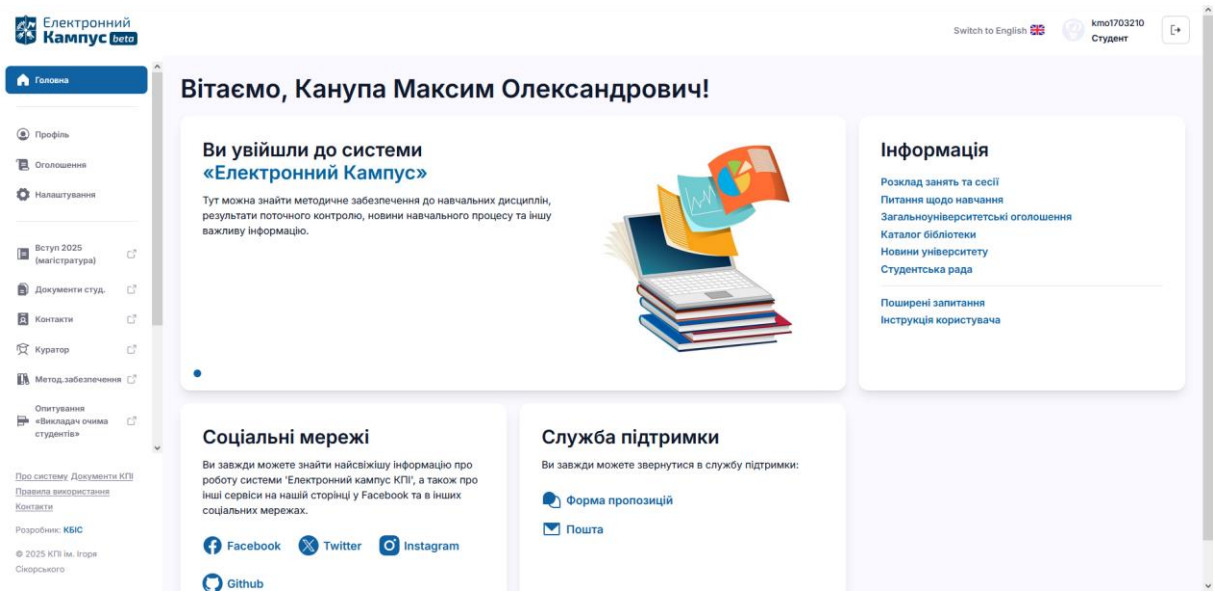


Рисунок 2.2 – Скріншот Campus

Однак, Campus KPI також має певні обмеження та недоліки. Одним із суттєвих мінусів є відсутність автоматизованої підтримки результатів неформальної освіти, зокрема, сертифікатів з платформ Coursera чи Prometheus. У разі потреби додавання таких даних це зазвичай потребує ручного введення та підтвердження, що суттєво ускладнює процес і збільшує навантаження на адміністративний персонал.

Крім того, система має обмежений інтерфейс і функціонал, які вже не повною мірою відповідають сучасним вимогам користувачів. Застарілий дизайн і відсутність адаптивного веб-інтерфейсу ускладнюють доступ до системи зі смартфонів та планшетів, що негативно впливає на зручність використання студентами та викладачами. Такі недоліки роблять Campus менш зручним рішенням у порівнянні з сучасними платформами управління навчальним процесом.

## 2.3 Google Classroom

Google Classroom — це безкоштовна платформа для управління навчанням, розроблена компанією Google. Основні функції Google Classroom включають:

– створення та розповсюдження завдань: викладачі можуть створювати завдання, надавати інструкції та матеріали, а також збирати виконані роботи студентів;

– оцінювання та зворотний зв'язок: система дозволяє оцінювати роботи студентів, надавати коментарі та відстежувати прогрес;

– інтеграція з Google Workspace: Google Classroom інтегрується з такими інструментами, як Google Docs, Google Drive та Google Calendar, що забезпечує зручність у використанні та спільній роботі;

– мобільний доступ: платформа доступна через веб-інтерфейс та мобільні додатки для Android та iOS, що дозволяє студентам та викладачам працювати з будь-якого пристрою.

Google Classroom є популярним вибором для шкіл та університетів завдяки простоті використання та інтеграції з іншими сервісами Google [8]. Приклад курсу на платформі Classroom зображено на рисунку 2.3.

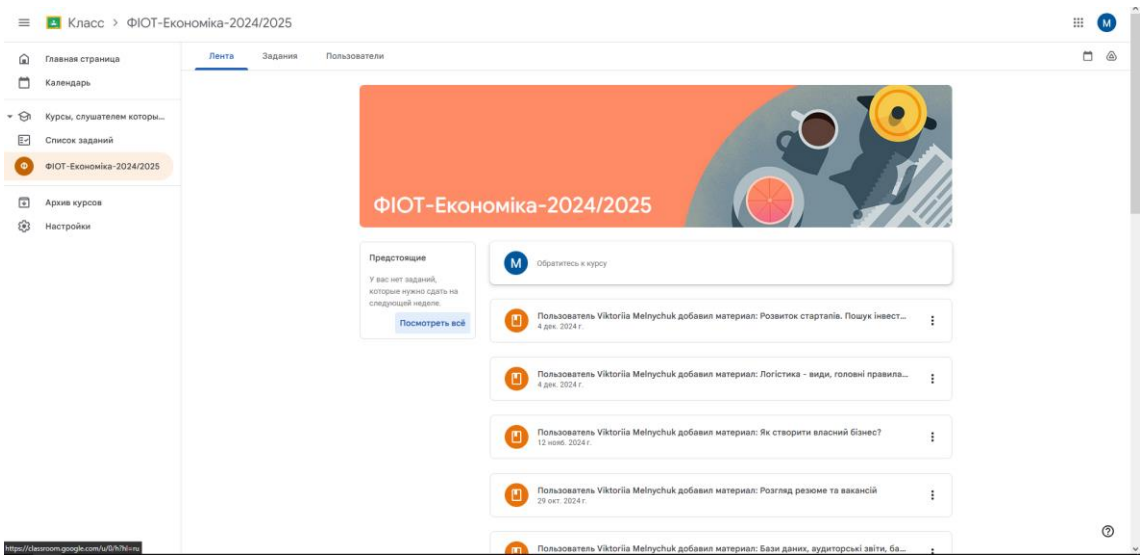


Рисунок 2.3 – Скріншот Google Classroom

Хоча Google Classroom має багато переваг, існують і певні обмеження, які знижують його ефективність для повного управління навчальним процесом. Одним із недоліків є відсутність підтримки автоматизованого обліку результатів неформальної освіти, таких як сертифікати Coursera, Prometheus та інші. Для

додавання таких сертифікатів користувачам доводиться самостійно їх прикріплювати та підтверджувати, що створює додаткове навантаження та знижує зручність використання системи.

Крім того, інтерфейс Google Classroom, хоч і зручний для створення та розповсюдження завдань, має певні недоліки в аспектах перегляду успішності. Оцінки студентів не представлені у вигляді зведених таблиць, що ускладнює викладачам та адміністрації навчального закладу швидке отримання повної картини успішності групи чи курсу [8]. Це обмежує можливості системи у використанні для комплексного управління навчальним процесом, порівняно з більш спеціалізованими освітніми платформами.

## 2.4 Порівняльний аналіз систем

Під час аналізу сучасного стану проблеми управління навчальним процесом було розглянуто найбільш популярні програмні рішення, які використовуються у закладах освіти. Порівняльний аналіз систем наведено в таблиці 2.1.

Таблиця 2.1 – Порівняльний аналіз систем

| Критерій | Moodle | Campus KPI | Google Classroom |
|---|---|---|---|
| Тип ліцензії | Відкрита (open-source) | Пропрієтарна | Безкоштовна |
| Гнучкість налаштувань | Висока | Середня | Обмежена |
| Інтеграція з іншими системами | Середня | Низька | Висока |
| Мобільний доступ | Так | Частково | Так |
| Аналітика та звітність | Розширена | Розширена | Базова |
| Простота використання | Середня | Середня | Середня |

Згідно з наведеним аналізом, кожна з розглянутих систем має свої переваги та обмеження. Вибір конкретної платформи залежить від потреб та ресурсів освітнього закладу. Враховуючи результати порівняльного аналізу, можна зробити висновок, що Moodle забезпечує найвищу гнучкість налаштувань та потужний інструментарій для аналітики й звітності, однак потребує додаткових ресурсів для налаштування та підтримки. Campus KPI, розроблений спеціально для потреб КПІ, добре інтегрований у внутрішні процеси навчального закладу, але має обмежені можливості інтеграції з зовнішніми системами та застарілий інтерфейс. Google Classroom, у свою чергу, пропонує простоту використання та високу інтеграцію з сервісами Google, проте обмежений у можливостях аналізу та налаштувань, що робить його менш ефективним для комплексного управління навчальним процесом.

Отже, розробка власної інформаційної системи дозволяє поєднати сильні сторони кожного рішення та усунути їх недоліки, створивши інноваційну платформу для закладів освіти.

## Висновок до розділу 2

Аналіз існуючих систем управління навчальним процесом показує, що хоча такі платформи, як Moodle, Campus KPI та Google Classroom, забезпечують широкий спектр функцій для підтримки навчального процесу, жодна з них не є універсальним рішенням для всіх типів освітніх установ. Тому розробка нової інформаційної системи управління навчальним процесом, яка б поєднувала гнучкість, розширені можливості підтримки неформальної освіти та зручність використання, є актуальним завданням.

# 3 ВИБІР ТЕХНОЛОГІЙ ТА ПРОЄКТУВАННЯ СИСТЕМИ

Розробка інформаційної системи управління навчальним процесом у закладі освіти потребує ретельного вибору технологічного стеку, середовищ розробки, архітектурних рішень та підходів до проєктування. В цьому розділі описано вибір мови програмування, інструментів розробки, бази даних, фреймворків для фронтенду та бекенду, а також архітектурних принципів.

## 3.1 Вибір мови програмування

Серверна частина розробляється на мові C#, що є однією з найбільш потужних, сучасних та популярних мов програмування для створення корпоративних і високонавантажених застосунків. C# поєднує у собі високу продуктивність та стабільність із простотою синтаксису, що робить її зручною для розробників різного рівня підготовки [12]. Одна з ключових переваг цієї мови полягає у потужній підтримці об'єктно-орієнтованого програмування. ООП дозволяє будувати чітко структуровані застосунки, де кожна частина системи представлена у вигляді класів та об'єктів, з чітко визначеними ролями та обов'язками. Це полегшує розробку, супровід, масштабування та тестування коду.

Мова C# підтримує сучасні парадигми розробки, включаючи асинхронне програмування, делегати, події та вирази лямбда, що дозволяє створювати ефективні, зрозумілі та масштабовані рішення. Застосування LINQ (Language Integrated Query) у C# дає можливість працювати з даними безпосередньо у коді, забезпечуючи високу читабельність та зменшуючи кількість помилок. LINQ дозволяє здійснювати запити до колекцій об'єктів, баз даних, XML-документів і навіть потоків даних у зрозумілій та компактній формі, що значно прискорює розробку і спрощує підтримку проєкту [12].

Ще однією значущою перевагою C# є використання потужного фреймворку .NET Core, який дозволяє створювати високопродуктивні серверні застосунки, що здатні працювати на різних платформах. Це забезпечує можливість запуску

застосунку як на Windows, так і на Linux чи macOS. ORM Entity Framework Core, що використовується у цьому проєкті, інтегрується з C# та надає можливість працювати з базами даних на високому рівні абстракції. Це дозволяє розробникам взаємодіяти з базою даних через об'єкти і властивості без потреби писати складні SQL-запити. Entity Framework Core також підтримує міграції бази даних, що спрощує управління змінами у структурі бази даних під час розвитку системи [13].

Крім того, C# має потужну підтримку спільноти, що дозволяє розробникам отримувати допомогу, знаходити приклади коду та використовувати велику кількість готових рішень. Мова активно розвивається компанією Microsoft і підтримує нові стандарти та інструменти розробки, що робить її перспективною для довготривалих проєктів. Усе це забезпечує високу надійність, масштабованість і гнучкість для створення серверної частини системи, здатної забезпечувати обробку великої кількості запитів, інтеграцію з іншими сервісами та ефективне управління даними [11].

## 3.2 Вибір середовища розробки

При розробці інформаційної системи управління навчальним процесом вибір середовища розробки є ключовим фактором для забезпечення ефективності роботи, зручності для команди розробників і якості кінцевого продукту. Від цього вибору залежить швидкість створення проєкту, можливості інтеграції, тестування та підтримки. У проєкті для серверної та клієнтської частин було обрано оптимальні середовища розробки після ретельного аналізу доступних варіантів.

При виборі середовища розробки для серверної частини було розглянуто декілька популярних варіантів, таких як Visual Studio Code [14], Visual Studio [19] та JetBrains Rider [23]. Visual Studio Code є легким та універсальним редактором, який підтримує багато мов програмування, включаючи C#, завдяки численним розширенням. Однак цей редактор більше підходить для менш масштабних проєктів або для виконання окремих завдань, наприклад, редагування конфігураційних файлів, а не для повноцінної розробки складних .NET-

застосунків. Visual Studio, у свою чергу, є потужним та глибоко інтегрованим середовищем розробки для роботи з .NET, що надає широкий набір інструментів, включаючи дизайнер форм, відладчик, інструменти для роботи з базами даних та засоби створення діаграм класів. Саме Visual Studio було використано у проєкті для побудови діаграм класів, оскільки цей інструмент забезпечує просте та ефективне створення візуальних моделей системи [19].

Порівняння цих двох середовищ наведено в таблиці 3.1.

Таблиця 3.1 – Порівняння Visual Studio та JetBrains Rider

| Критерій | Visual Studio | JetBrains Rider |
|---|---|---|
| Потужність інтеграції | Висока | Висока |
| Гнучкість налаштувань | Обмежена | Висока |
| Підтримка рефакторингу | Базова | Розширена |
| Швидкість роботи | Відносно повільна | Вища |
| Підтримка платформ | Тільки Windows | Windows, Linux, macOS |

Для основної розробки серверної частини було обрано JetBrains Rider. Цей вибір пояснюється тим, що Rider поєднує у собі кращі можливості інших середовищ: має гнучке налаштування, підтримує інтеграцію з системами контролю версій, такими як Git, забезпечує глибоку інтеграцію з .NET Core, Entity Framework Core та іншими інструментами, що використовуються у проєкті. Rider також надає зручний та швидкий інтерфейс, сучасні засоби рефакторингу, ефективну навігацію по коду, що робить його ідеальним для розробки великого корпоративного застосунку. Підтримка середовища Windows, Linux та macOS дозволяє забезпечити кросплатформенність при розробці, а розширений функціонал JetBrains Rider підходить для складних сценаріїв програмування, від роботи з кодом до тестування [23].

Для клієнтської частини також були розглянуті різні варіанти середовищ, серед яких Visual Studio Code та JetBrains WebStorm. WebStorm є потужним інструментом для веброзробки з підтримкою сучасних JavaScript-фреймворків,

включаючи Angular. Однак WebStorm є комерційним продуктом і має обмежену гнучкість налаштування порівняно з Visual Studio Code [14].

Visual Studio Code був обраний для клієнтської частини завдяки своїй легкості, швидкодії, великій кількості безкоштовних розширень і модулів, які дозволяють налаштувати середовище відповідно до потреб Angular-розробки [2]. Visual Studio Code забезпечує підтримку сучасних стандартів веброзробки, включаючи TypeScript, HTML, CSS, а також має інтеграцію з системами контролю версій, що полегшує спільну роботу над проєктом. Крім того, завдяки великій спільноті користувачів цей редактор має багатий набір плагінів і регулярні оновлення, що робить його гнучким та ефективним рішенням для розробки фронтенду.

### 3.3 Вибір фреймворку бекенду

Для створення серверної частини інформаційної системи було обрано ASP.NET Core як основний фреймворк для реалізації бекенду.

ASP.NET Core є кросплатформеним, високопродуктивним і модульним фреймворком, який дозволяє будувати масштабовані та надійні вебзастосунки. Завдяки своїй архітектурі він підтримує використання сучасних підходів, таких як dependency injection для зручного впровадження залежностей, middleware-компоненти для обробки запитів і відповідей, а також інтеграцію з логуванням, моніторингом та обробкою помилок. Middleware у ASP.NET Core дозволяють створювати гнучкий конвеєр обробки HTTP-запитів, де кожен компонент відповідає за окремий аспект обробки – від аутентифікації користувачів до форматування відповіді [11].

ASP.NET Core також підтримує архітектурний патерн Model-View-Controller (MVC), який дозволяє чітко розділити логіку обробки даних, інтерфейс користувача та управління запитами. У цьому патерні контролери (Controller) відповідають за прийом і обробку вхідних запитів, викликають відповідні сервіси або бізнес-логіку, а також керують потоками даних між моделями і

представленнями. Моделі (Model) містять дані і бізнес-логіку, що описує стан об'єктів та правила їх взаємодії, забезпечуючи цілісність і правильність обробки інформації. Представлення (View) відповідають за відображення результатів обробки даних користувачу у вигляді інтерфейсу.

Завдяки використанню MVC в ASP.NET Core забезпечується структурованість коду, легкість супроводу та можливість швидкого розвитку функціональності. У проєкті даний підхід дозволяє реалізувати REST API у вигляді контролерів, які обробляють HTTP-запити, повертають дані у форматі JSON та взаємодіють із клієнтською частиною через чітко визначені ендпоінти.

Однією з важливих переваг ASP.NET Core є вбудована підтримка автентифікації та авторизації, зокрема з використанням JWT (JSON Web Token) [7]. Завдяки цьому забезпечується безпечний обмін даними між клієнтами та сервером, а також можливість реалізації різних рівнів доступу для користувачів із різними ролями. Система автентифікації легко конфігурується за допомогою middleware, що дозволяє інтегрувати зовнішні сервіси авторизації або реалізовувати власні механізми безпеки. ASP.NET Core також підтримує політики авторизації, що дозволяють детально налаштувати доступ до окремих ресурсів системи залежно від ролей або прав користувачів.

Для спрощення розробки, тестування та документування API у проєкті використовується інтеграція зі Swagger (OpenAPI). Завдяки цьому автоматично генерується інтерактивна документація для REST API, яка дозволяє розробникам та тестувальникам перевіряти методи API, параметри запитів, приклади відповідей і статуси. Swagger забезпечує зручний інтерфейс для тестування, а також дозволяє швидко виявляти та виправляти помилки у запитах чи обробці даних.

ASP.NET Core також має гнучкі можливості для інтеграції з різними базами даних, такими як SQL Server, PostgreSQL, MySQL та іншими, через ORM Entity Framework Core. Це дозволяє розробникам працювати з даними на високому рівні абстракції, використовуючи об'єкти C# для доступу та маніпуляцій даними. Крім того, ASP.NET Core легко інтегрується з зовнішніми сервісами через REST або

gRPC, забезпечуючи гнучкість архітектури та можливість масштабування проєкту [13].

Усе це робить ASP.NET Core ідеальним вибором для розробки серверної частини інформаційної системи управління навчальним процесом. Цей фреймворк забезпечує високу продуктивність, безпеку, підтримку сучасних протоколів і технологій, а також гнучкість для подальшого розвитку та інтеграції з іншими системами.

## 3.4 Вибір бази даних

При виборі бази даних для зберігання даних інформаційної системи управління навчальним процесом було розглянуто як реляційні, так і нереляційні СУБД, оскільки кожна категорія має свої переваги та обмеження. Реляційні бази даних (SQL) відзначаються високою надійністю, забезпечують підтримку складних запитів, транзакцій і дотримання принципів цілісності даних, що критично важливо для навчальної системи, де потрібно зберігати розклади, оцінки, відвідуваність та іншу структуровану інформацію.

З іншого боку, NoSQL бази даних, наприклад MongoDB або Redis, відомі своєю гнучкістю у зберіганні неструктурованих або слабо структурованих даних, високою продуктивністю при роботі з великими обсягами інформації та можливістю масштабування по горизонталі. Однак у даному проєкті NoSQL бази даних були відхилені через потребу у складній реляційній логіці, яка найкраще реалізується в реляційних СУБД.

Для продуктивного середовища було обрано PostgreSQL. Це потужна, масштабована реляційна система управління базами даних, яка підтримує складні SQL-запити, транзакції, обмеження цілісності, тригери та індекси. PostgreSQL забезпечує високий рівень надійності, стійкості до відмов, можливість реплікації та роботи у хмарних середовищах, таких як AWS, Azure або GCP. Це дозволяє масштабувати систему під збільшення кількості користувачів і обсягу даних без ризику втрати продуктивності. Крім того, PostgreSQL має розширені можливості

роботи з JSON-форматом, що може бути корисним у перспективі для додаткових функцій.

Для локальної розробки та тестування у проєкті використовується SQLite. Ця легка, вбудована реляційна СУБД не потребує розгортання окремого сервера, що значно спрощує налаштування середовища розробки. SQLite дозволяє швидко створювати, тестувати та налагоджувати застосунок без необхідності підключення до віддалених серверів. Крім того, у проєкті SQLite використовується для забезпечення роботи Hangfire – інструмента для планування та виконання фонових завдань, наприклад, для обробки черг повідомлень, створення звітів або відправки електронної пошти. Завдяки легкості SQLite цей підхід дозволяє знизити навантаження на основну СУБД і забезпечити стабільну роботу фонового процесу без додаткової конфігурації.

Таким чином, вибір бази даних базується на поєднанні переваг обох підходів: для основного зберігання структурованих даних у продуктивному середовищі використовується потужна і гнучка PostgreSQL, тоді як для локальної розробки, тестування і підтримки фонових завдань Hangfire застосовується легка SQLite. Такий підхід забезпечує гнучкість, масштабованість і надійність інформаційної системи.

## 3.5 Архітектура системи

Система реалізована за принципами DDD (Domain-Driven Design), що передбачає чіткий поділ коду на логічні шари:

– domain layer (доменний рівень) містить сутності, агрегати, інтерфейси репозиторіїв, бізнес-правила;

– application layer (рівень застосування) реалізує сервіси, обробники команд і запитів, DTO, інтерфейси для інтеграцій із зовнішніми системами;

– web layer (презентаційний рівень) реалізує API-контролери, обробку запитів від клієнтів, авторизацію і автентифікацію користувачів [6].

Такий архітектурний підхід до розробки системи допомагає дотримуватися підходів SOLID, в особливості SRP (single responsibility principle) – принципу єдиної відповідальності, як на рівні окремих класів, які описують лише той рівень абстракції, що потрібен на даному рівні, так і на рівні усіх логічних шарів, які мають свою окрему зону відповідальності. Це дуже допомагає при тестуванні системи, як при загальному тестуванні, так і при тестуванні окремих її компонентів і їх взаємодій.

Особливо даний архітектурний підхід відрізняється від класичного підходу, коли в нас база даних є найнижчим рівнем, на якому будуються, або від якого залежать всі інші. В даному випадку всі рівні використовують лише абстракцію до джерела даних, а сама реалізація бази даних підключається на верхньому рівні як елемент інфраструктури. Це робить систему стійкою до змін у бізнес-вимогах і дозволяє ефективно впроваджувати новий функціонал, що відповідає сучасним вимогам розробки корпоративних інформаційних систем [6].

### 3.6 Вибір фронтенд-фреймворку

Для клієнтської частини інформаційної системи було обрано Angular, потужний фреймворк для створення продуктивних і масштабованих односторінкових застосунків (SPA), які забезпечують швидкий та динамічний інтерфейс користувача. Angular підтримує двосторонню прив’язку даних, що дозволяє автоматично синхронізувати стан моделі та відображення на сторінці, забезпечуючи зручність і мінімізацію помилок. Крім того, Angular відзначається своєю модульністю, що дозволяє розбивати код на окремі компоненти та сервіси, полегшуючи супровід, тестування та масштабування проєкту. Інтеграція з API здійснюється за допомогою HttpClient, що дозволяє ефективно виконувати HTTP-запити та обробляти відповіді від сервера. Використання реактивних форм Angular дозволяє реалізувати складну валідацію даних користувачів без потреби створення додаткового коду, що значно підвищує надійність і безпеку клієнтської частини.

Додатково у проєкті для забезпечення сучасного вигляду інтерфейсу та підтримки мобільних пристроїв інтегровано Bootstrap – популярну бібліотеку стилів і компонентів. Bootstrap дозволяє швидко створювати адаптивні вебінтерфейси, які коректно відображаються на різних пристроях і розмірах екранів, забезпечуючи комфортний доступ користувачів до функціоналу системи. Це рішення дозволяє поєднати можливості Angular для складної логіки з візуально привабливим і інтуїтивно зрозумілим інтерфейсом користувача [3].

Таким чином, інтеграція Bootstrap у клієнтську частину системи дозволила створити сучасний і привабливий інтерфейс, що не лише відповідає вимогам адаптивності для різних пристроїв, але й значно спрощує реалізацію функціональних елементів, підвищує зручність роботи користувачів та скорочує час розробки інтерфейсу. Bootstrap забезпечує єдність стилів та структури сторінок, дозволяє легко масштабувати інтерфейс при розширенні системи та підтримує високу якість візуальної презентації системи.

## 3.7 Проєктування системи

Система проєктується як сучасний вебзастосунок із багаторівневою архітектурою, що забезпечує масштабованість, продуктивність, гнучкість та легкість супроводу. Основна ідея полягає у чіткому розділенні відповідальностей між різними компонентами системи та використанні перевірених підходів до побудови корпоративних застосунків. У основі архітектури лежить модель Domain-Driven Design (DDD), яка дозволяє відокремити бізнес-логіку від технічних деталей реалізації. Завдяки цьому структура коду стає зрозумілою, легко тестованою і готовою до подальшого розвитку.

API-інтерфейс реалізується у форматі REST, що забезпечує сумісність із різними клієнтами та стандартами взаємодії між компонентами. REST API підтримує стандартні методи CRUD (створення, читання, оновлення, видалення), що дозволяє легко реалізувати основні операції з даними. Для забезпечення безпеки система впроваджує автентифікацію та авторизацію за допомогою JWT

(JSON Web Token), що дозволяє захистити ресурси від несанкціонованого доступу, а також реалізувати різні рівні прав для користувачів.

Вхідні дані від користувача надходять з клієнтської частини, побудованої на Angular. Дані спочатку проходять валідацію на клієнті для попередньої перевірки коректності та запобігання помилок користувача. Після надходження на сервер дані додатково перевіряються за допомогою FluentValidation, що дозволяє реалізувати гнучкі правила валідації без дублювання коду. Валідовані дані мапляться у доменні об'єкти за допомогою Mapster – інструмента для автоматичного перетворення DTO у сутності бізнес-логіки. Ці доменні об'єкти потрапляють у шар бізнес-логіки, де виконуються обчислення, перевірки та інші операції згідно з правилами предметної області.

Результати обробки даних зберігаються у реляційній базі даних PostgreSQL у продуктивному середовищі, або у SQLite для локальної розробки та роботи Hangfire. Ця гнучкість дозволяє легко перемикатися між середовищами та забезпечує ефективне управління даними під час тестування і розгортання.

Клієнтська частина Angular взаємодіє з API за допомогою HttpClient, отримуючи оброблені та перевірені дані для відображення. Інтерфейс створено на основі Bootstrap для забезпечення адаптивності та підтримки мобільних пристроїв.

Завдяки використанню DDD система має чітко визначені області відповідальності: шар Core містить доменні моделі та логіку, Application – сервіси, DTO, інтеграції, WebApi – контролери та DI, що дозволяє підтримувати чисту архітектуру та легко масштабувати систему.

## Висновок до розділу 3

У цьому розділі було розглянуто і обґрунтовано вибір технологічного стеку, мов програмування, фреймворків, середовищ розробки, архітектурних підходів і засобів проєктування для розробки інформаційної системи управління навчальним процесом у закладі освіти. Основою серверної частини обрано .NET Core з використанням C#, що забезпечує високу продуктивність, безпеку та

масштабованість. Для клієнтської частини обрано Angular у поєднанні з Bootstrap, що дозволяє створити сучасний адаптивний вебінтерфейс із широкими можливостями налаштування та підтримкою мобільних пристроїв.

Архітектурно система реалізована за принципами DDD (Domain-Driven Design), що забезпечує чітке розмежування обов'язків між рівнями, спрощує розробку, тестування та подальшу підтримку проєкту. Використання PostgreSQL для продуктивного середовища та SQLite для тестового дозволяє гнучко масштабувати систему та спрощує локальну розробку. Інтеграція із зовнішніми сервісами, такими як Coursera та Prometheus, реалізується через API, забезпечуючи гнучкість та розширюваність функціоналу.

Вибір технологій та архітектури системи забезпечує надійність, високу якість коду, простоту розширення та підтримки, а також дозволяє ефективно реалізувати ключові завдання автоматизації навчального процесу. Такий підхід забезпечить ефективність, безпеку та масштабованість інформаційної системи в умовах сучасної цифровізації освіти.

# 4 РОЗРОБЛЕННЯ СИСТЕМИ

Цей розділ присвячений опису основних етапів проєктування та реалізації інформаційної системи управління навчальним процесом у закладі освіти. Система призначена для автоматизації ключових аспектів навчального процесу, включаючи управління розкладом занять, облік відвідуваності та успішності студентів, підтримку результатів неформальної освіти, адміністрування користувачів і навчальних матеріалів.

Розробка інформаційної системи здійснювалась з урахуванням сучасних вимог до програмних продуктів, включаючи масштабованість, безпеку, багатофункціональність та зручність користування. У розділі буде детально описано структуру системи, функціональну модель, архітектуру програмного забезпечення, модель бази даних, особливості передавання та обробки даних. Окремо буде висвітлено розробку діаграм класів і керівництво користувача, що дозволяє наочно представити побудову системи та принципи її використання.

Таким чином, розділ демонструє системний підхід до реалізації інформаційної системи управління навчальним процесом, який включає планування, розробку, тестування та підготовку до експлуатації програмного забезпечення.

## 4.1 Структура системи

Інформаційна система має багаторівневу структуру, що включає такі основні компоненти:

– клієнтська частина (Angular SPA);

– серверна частина (ASP.NET Core Web API);

– реляційна база даних (SQLite та PostgreSQL);

– фонові служби (Hangfire).

Кожен компонент взаємодіє через інтерфейси, що спрощує підтримку та розширення системи.

Клієнтська частина реалізована як односторінковий додаток (SPA) на основі Angular версії 19. Цей вибір обумовлений високою продуктивністю, підтримкою сучасних стандартів веб-розробки, зокрема TypeScript, а також гнучкою системою компонентів і широкими можливостями для інтеграції з іншими технологіями. Angular 19 забезпечує оптимізацію продуктивності, сучасну систему реактивного програмування через RxJS, підтримку серверного рендерингу (SSR) та вдосконалену систему управління станом. Для побудови інтерфейсу використовується Bootstrap через бібліотеку ng-bootstrap, яка дозволяє інтегрувати компоненти Bootstrap без додаткової залежності від jQuery, забезпечуючи чисту та гнучку адаптацію інтерфейсу до різних пристроїв та роздільних здатностей екранів. Це рішення гарантує швидкість завантаження сторінок, зручну верстку та сучасний дизайн інтерфейсу.

Серверна частина системи реалізована на платформі ASP.NET Core, що забезпечує високу продуктивність, кросплатформенність, гнучку систему безпеки та підтримку сучасних стандартів розробки. Архітектура серверної частини побудована за принципами Clean Architecture із поділом на шари: Domain, Application та Web API. У шарі Domain описані бізнес-моделі та логіка, в Application розташовано сервіси, DTO та інтеграції, тоді як Web API містить контролери, DI-контейнери та маршрутизацію. Для забезпечення гнучкості та повторного використання коду створено власну бібліотеку загальних сервісів та утиліт, яка містить логіку для роботи з JWT, логування через Serilog, обробки помилок та автоматичного мапінгу об'єктів через Mapster. Таке розділення дозволяє легко тестувати кожен компонент, спрощує впровадження нових функцій і підтримує високу якість коду.

Реляційна база даних системи реалізована на основі PostgreSQL, яка забезпечує надійність, масштабованість та підтримку сучасних реляційних механізмів, включаючи ACID, складні транзакції та JSONB для збереження напівструктурованих даних. Водночас для фонового менеджера Hangfire

використовується SQLite через легкість його налаштування та мінімальні вимоги до ресурсів, що дозволяє ізолювати фонові завдання від основної бази даних без шкоди для продуктивності. Завдяки використанню Entity Framework Core система легко підтримує зміну провайдера бази даних: наприклад, у разі необхідності можна використовувати SQLite як основну базу для всієї системи або перейти на інший провайдер без істотних змін у коді.

Фонові служби реалізовані з використанням Hangfire — потужного інструменту для управління чергами завдань, який дозволяє запускати тривалі чи відкладені задачі у фоновому режимі. Це особливо важливо для реалізації таких функцій, як обробка великої кількості даних, відправка електронних листів, обробка сертифікатів або формування звітів без затримок у роботі основного інтерфейсу користувача. Hangfire інтегрується з ASP.NET Core і підтримує збереження стану черг у базі даних, що забезпечує надійність виконання завдань навіть у разі перезавантаження сервера чи відмови системи.

Загалом така структура дозволяє забезпечити стабільну роботу інформаційної системи управління навчальним процесом із можливістю масштабування, гнучкого розширення функціоналу та легкої підтримки протягом усього життєвого циклу програмного забезпечення.

## 4.2 Функціональна модель системи

Функціональна модель інформаційної системи управління навчальним процесом у закладі освіти описує основні дії та сценарії використання, що дозволяють користувачам взаємодіяти з системою для досягнення своїх цілей. Ця модель визначає ключових учасників процесу (акторів) та їхні ролі, забезпечуючи чітке розуміння функціоналу системи для кожної категорії користувачів. Завдяки цьому підходу забезпечується прозорість архітектури та спрощується подальша розробка та тестування програмного забезпечення.

Актори:

– адміністратор — створює викладачів, групи, предмети, налаштовує сесію та атестації;

– викладач — керує уроками, виставляє оцінки, атестації, додає студентів;

– студент — переглядає розклад, оцінки, обирає предмети, завантажує сертифікати.

У системі виділено три основні категорії користувачів — адміністратор, викладач і студент, кожен з яких має свій набір функцій. Адміністратор відповідає за налаштування навчального процесу, зокрема створення груп, додавання викладачів та предметів, а також керування сесіями та атестаціями. Викладачі мають можливість керувати уроками, виставляти оцінки та атестації, а також додавати студентів до груп. Студенти можуть переглядати свої розклади, оцінки, обирати дисципліни для навчання та завантажувати сертифікати, отримані на зовнішніх освітніх платформах.

Модель варіантів використання системи включає такі основні сценарії взаємодії: авторизація та реєстрація користувачів, керування навчальними групами адміністратором, додавання та редагування уроків викладачами, виставлення оцінок та атестацій, вибір предметів студентами, завантаження сертифікатів та перегляд особистого кабінету й розкладу занять як для студентів, так і для викладачів. Кожен сценарій супроводжується перевіркою прав доступу та валідацією даних для забезпечення безпеки та коректності роботи системи.

Модель варіантів використання (Use Case):

– авторизація та реєстрація (усі актори);

– керування навчальними групами (адміністратор);

– перегляд/додавання/редагування уроків (викладач);

– виставлення атестацій, оцінок (викладач);

– обрання вибіркових предметів (студент);

– завантаження сертифікатів (студент);

– перегляд особистого кабінету та розкладу (викладач та студент).

Таким чином, функціональна модель системи дозволяє чітко визначити ролі користувачів та їхні дії, що забезпечує ефективну організацію навчального процесу та зручність користування системою. Реалізація цих сценаріїв у програмному забезпеченні сприятиме підвищенню ефективності роботи закладу освіти, забезпечить прозорість і контроль за навчальним процесом.

## 4.3 Модель бази даних

Для забезпечення ефективного та структурованого збереження даних інформаційна система управління навчальним процесом використовує реляційну базу даних, яка реалізована на основі сучасних рішень SQL. Основною системою управління базами даних обрано PostgreSQL, яка відзначається високою продуктивністю, масштабованістю та підтримкою складних запитів, що дозволяє працювати з великими обсягами даних освітнього закладу [16]. Також у системі використовується SQLite для зберігання даних фонових задач, таких як планування та моніторинг через Hangfire, завдяки легкості налаштування та використання.

Схема бази даних зображена кресленнику ІС12.150БАК.005 Д1.

Взаємодія з базою даних здійснюється за допомогою Entity Framework Core (EF Core) — сучасного ORM (Object-Relational Mapping) для платформи .NET, який спрощує розробку та забезпечує безпечну роботу з даними. EF Core дозволяє абстрагуватися від конкретної реалізації бази даних та легко змінювати її при необхідності, наприклад, перевести систему повністю на SQLite для локального використання або іншого середовища. Нормалізація бази даних забезпечує відсутність надмірностей і дублювання даних, що дозволяє підвищити ефективність та цілісність інформації [13].

У результаті побудована модель бази даних є гнучкою, масштабованою та зручною для підтримки ключових бізнес-процесів системи, забезпечуючи швидкий доступ до необхідної інформації як для користувачів, так і для розробників.

База даних включає основні таблиці:

– teachers (викладачі);

– students (студенти);

– lessons (уроки);

– groups (групи);

– subjects (предмети);

– subjectgrades (оцінки);

– attestations (атестації);

– sessions (сесії);

– certificates (сертифікати);

– selectedSubjectGroups (вибрані групи предметів).

Усі зв'язки реалізовано через зовнішні ключі з підтримкою каскадного видалення та оновлення. Нормалізація — 3НФ.

## 4.4 Передавання та обробка даних

Ефективна передача та обробка даних є ключовим аспектом функціонування інформаційної системи управління навчальним процесом. З огляду на складну багаторівневу архітектуру системи, дані повинні передаватися між клієнтською та серверною частинами швидко, безпечно та з мінімальними затримками. Основою для цього є використання сучасних технологій передачі даних, таких як HTTPS для забезпечення конфіденційності та захисту інформації, а також формат JSON для легкості обробки та інтеграції з фронтендом.

Вхідні дані системи надходять із кількох джерел. Основним джерелом є форми, заповнені користувачами на клієнтській частині Angular. Ці форми передаються у форматі JSON, що дозволяє швидко та ефективно обробляти інформацію на сервері. Іншим важливим джерелом є сертифікати, які імпортуються із зовнішніх освітніх платформ, таких як Coursera та Prometheus. Вони також передаються у структурованому форматі та зберігаються для подальшої обробки. Схема обробки сертифікату зображена на кресленнику ІС12.150БАК.005 Д2.

Третє джерело — результати авторизації, які включають дані про користувача, отримані при вході до системи за допомогою JWT-токенів [7].

На виході система генерує різноманітні дані, необхідні для роботи користувачів і адміністраторів. Це можуть бути таблиці з фільтрами для перегляду успішності, графіки та діаграми, які відображають навчальні досягнення студентів. Крім того, система формує API-відповіді для інтеграції з іншими сервісами та клієнтськими додатками.

Передача даних між клієнтом та сервером здійснюється через захищене з'єднання HTTPS, що гарантує безпеку інформації та захист від атак типу "людина посередині". Для автентифікації користувачів використовується механізм JWT (JSON Web Token), який дозволяє підтвердити особу користувача та його права доступу до ресурсів системи. Це забезпечує безпечний та контрольований доступ до даних.

На сервері всі вхідні дані проходять ретельну перевірку з використанням FluentValidation, що дозволяє виявляти помилки на етапі обробки запиту, забезпечуючи високу якість і цілісність даних. Після перевірки дані перетворюються у доменні моделі за допомогою бібліотеки Mapster [10]. Це дозволяє ефективно працювати з даними в рамках бізнес-логіки системи, знижуючи ризики помилок та підвищуючи продуктивність роботи.

У результаті реалізована схема передавання та обробки даних забезпечує надійність, безпеку та швидкодію системи. Вона дозволяє інтегрувати різні джерела інформації, забезпечує коректне функціонування внутрішніх процесів та створює

гнучке середовище для розвитку і масштабування інформаційної системи управління навчальним процесом.

## 4.5 Архітектура програмного забезпечення

Серверна частина системи реалізована за шаблоном Domain Driven Development (DDD) з поділом на шари:

– core (домени, моделі);

– application (логіка, DTO, сервіси та інтеграції);

– web (контролери, DI, модулі).

Структура проекту, поділ на шари та директорії зображено на рисунку 4.1.

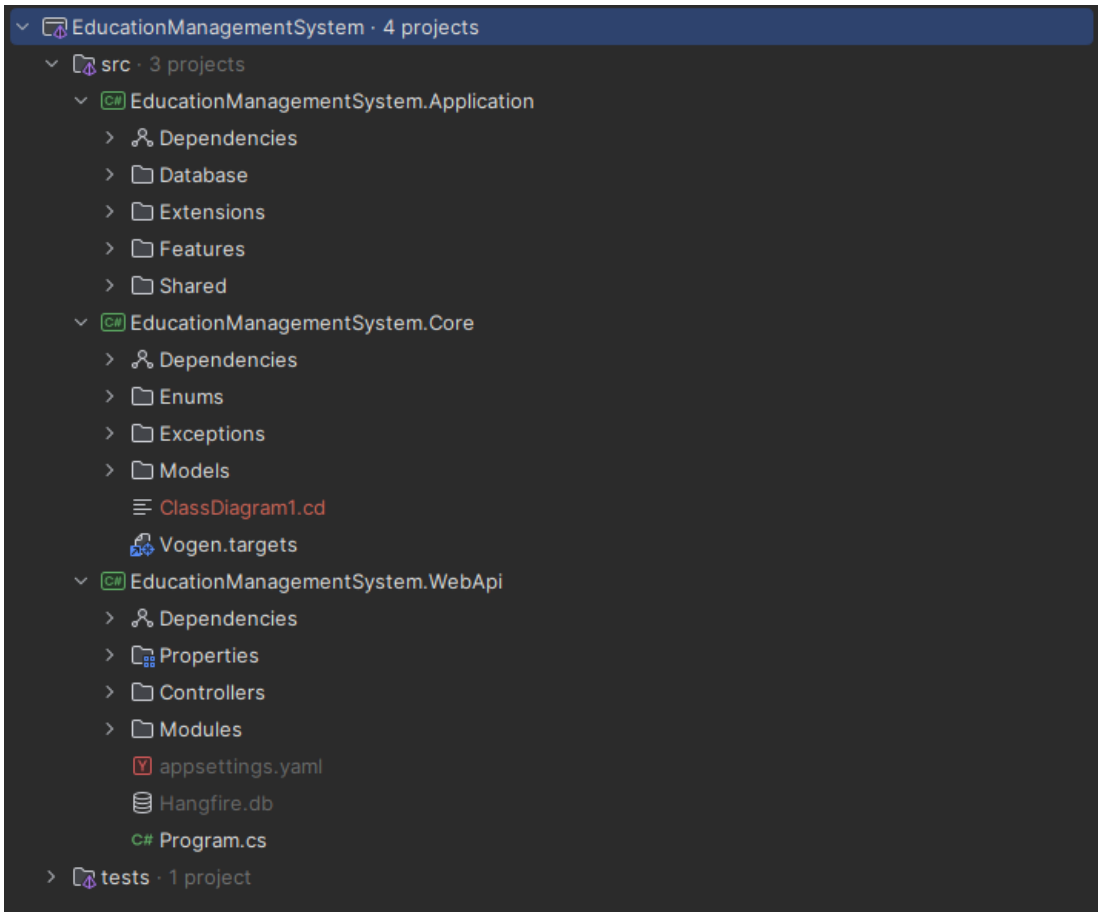


Рисунок 4.1 – Структура проекту серверної частини

Серверна частина системи побудована на основі архітектурного підходу Domain Driven Design (DDD), який дозволяє створити добре структуровану, масштабовану та легко підтримувану програмну систему. Основна ідея DDD полягає у відокремленні доменної логіки від інфраструктури та інших аспектів реалізації, що дає змогу зосередитися на бізнес-процесах і правилах предметної області.

Шар Core відповідає за визначення основних доменних моделей, ентіті та агрегацій, які відображають логіку предметної області. Саме тут визначаються бізнес-правила та інваріанти, які мають бути дотримані в системі. У реалізації на .NET Core цей шар є окремим проєктом у рішенні, що містить класи для основних сутностей, інтерфейси репозиторіїв та сервіси домену.

Шар Application реалізує бізнес-логіку застосунку, де описуються DTO (Data Transfer Objects), сервіси, що реалізують прикладну логіку, та інтеграції із зовнішніми системами. Він взаємодіє з Core через інтерфейси та забезпечує виконання бізнес-сценаріїв, наприклад, створення нових записів, обробку запитів користувачів чи формування звітів. У структурі рішень .NET цей шар оформлено як окремий проєкт із залежністю від Core.

Шар WebApi включає контролери ASP.NET Core, які приймають HTTP-запити, викликають сервіси з Application та формують відповіді клієнтам. Тут реалізовано Dependency Injection (DI) для управління життєвим циклом сервісів, налаштування middleware та політик безпеки. Цей шар не містить бізнес-логіки, лише координує роботу системи та маршрутизацію запитів.

Серед основних переваг використання DDD у цьому проєкті варто відзначити:

– чіткий розподіл обов'язків між шарами, що забезпечує легкість супроводу та розширення функціоналу;

– зосередження на бізнес-логіці в Core, що дозволяє уникнути надмірної залежності від інфраструктури;

– можливість повторного використання компонентів у різних частинах системи;

– гнучкість у виборі технологій інфраструктури завдяки інверсії залежностей та контрактам між шарами;

– зручність для тестування, оскільки бізнес-логіка та API можуть перевірятися окремо.

У рішенні на .NET Core кожен шар оформлено як окремий проєкт у .NET рішенні, що забезпечує чіткість структури та легкість навігації. Інтерфейси репозиторіїв з Core реалізуються в Application, а конкретні сервіси та інтеграції — у WebApi. Такий підхід дозволяє легко впроваджувати нові функції та масштабувати систему відповідно до потреб замовника.

Система складається з наступних підсистем:

– attestations – підсистема для управління атестаціями студентів, що дозволяє викладачам фіксувати підсумкові результати та атестаційні оцінки за певний період навчання. Забезпечує облік результатів та формування звітності;

– educationEvents – підсистема, яка відповідає за планування, проведення та відображення освітніх подій, таких як семінари, конференції, майстер-класи тощо. Дає можливість створювати календарі подій, додавати опис та залучати учасників;

– groups – підсистема управління навчальними групами, що забезпечує створення, редагування та облік студентських груп, їх складу, а також формування розкладу занять для кожної групи;

– sessions – підсистема для організації сесій, яка дозволяє планувати екзамени, контролювати хід їх проведення та облік підсумкових результатів;

– subjects – підсистема управління навчальними дисциплінами, яка дозволяє додавати нові предмети, редагувати інформацію про них, призначати їх викладачам та формувати навчальні програми;

– teachers – підсистема обліку та управління інформацією про викладачів, що містить дані про кваліфікацію, контактну інформацію та навантаження, а також дозволяє призначати викладачів на конкретні предмети;

– certificates – підсистема для роботи з результатами неформальної освіти, яка дозволяє студентам завантажувати сертифікати, отримані на платформах Coursera, Prometheus та інших сервісах, та інтегрувати їх у загальний профіль;

– grades – підсистема управління оцінками студентів, яка дозволяє викладачам фіксувати поточні та підсумкові оцінки, переглядати історію успішності студентів, формувати аналітичні звіти;

– lessons – підсистема для планування та проведення навчальних занять, що включає розклад уроків, контроль їх проведення, відмітки про відвідування та зміни у розкладі;

– students – підсистема управління інформацією про студентів, яка містить дані про особисті профілі, результати навчання, відвідуваність та участь у додаткових освітніх активностях;

– subjectsSelection – підсистема для вибору студентами навчальних дисциплін, що дозволяє обирати предмети для індивідуальних навчальних планів, реєструватися на курси та відслідковувати заповненість груп.

Кожна підсистема інтегрується з іншими через інтерфейси, забезпечуючи цілісність та зручність користування системою. Взаємодія між підсистемами реалізована відповідно до DDD-архітектури, що дозволяє легко масштабувати та розширювати функціональність у майбутньому.

На рисунку 4.2 зображено структуру клієнтської частини системи, яка використовує стандартні підходи до структурування файлів для Angular.


Рисунок 4.2 – Структура проекту клієнтської частини

## 4.6 Діаграми класів

У цьому підрозділі наведено діаграми класів, які відображають структуру основних сутностей програмного забезпечення інформаційної системи управління навчальним процесом. Діаграми класів є важливою частиною об'єктно-орієнтованого аналізу та проєктування, оскільки дозволяють візуалізувати класи, їхні атрибути, методи та взаємозв'язки між ними. Такий підхід сприяє чіткому розумінню архітектури системи, правильному визначенню відповідальностей компонентів та забезпеченню гнучкості при подальшій розробці, тестуванні й масштабуванні застосунку. Кожен клас у діаграмі відповідає певній сутності доменної області. Діаграма класів зображена на рисунку 4.3.

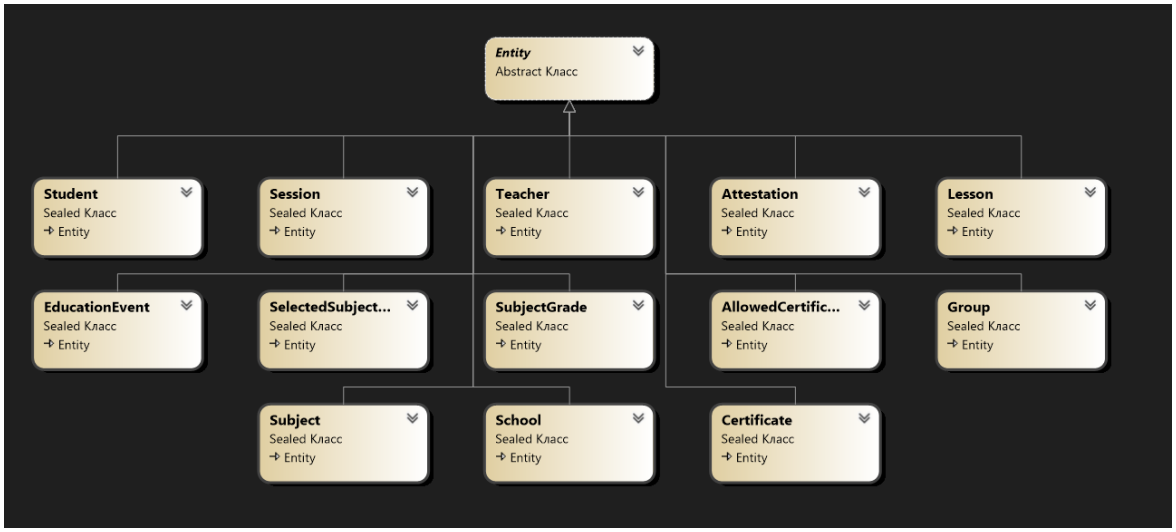


Рисунок 4.3 – Діаграма класів

Деталізована діаграма класу зображена на рисунку 4.4 на прикладі класу сертифікату.

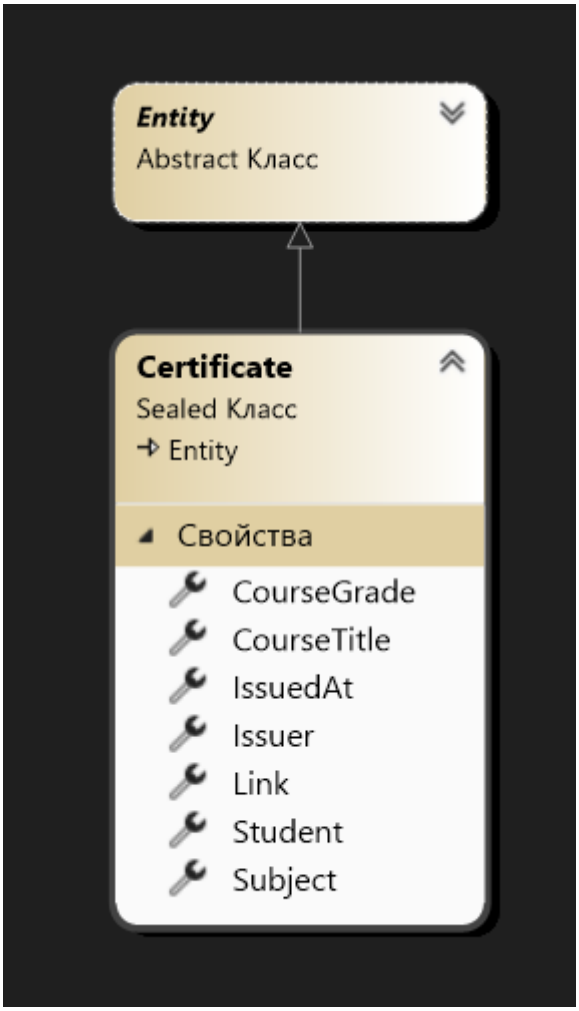


Рисунок 4.4 – Діаграма класу сертифікат

## 4.7 Керівництво користувача

У цьому підрозділі подано короткий опис функціональних можливостей інформаційної системи для різних категорій користувачів.

Оскільки система передбачає рольовий доступ, набір доступних дій залежить від ролі, яку має авторизований користувач. Інтерфейс системи побудовано таким чином, щоб забезпечити простоту використання, швидкий доступ до основних функцій, а також підтримку мобільних пристроїв.

Для студента:

– авторизуватись через логін і пароль;

– переглянути свій розклад і оцінки;

– обрати предмети у період вибору;

– завантажити сертифікати через форму з посиланням.

Для викладача:

– додати або редагувати урок;

– виставити оцінку чи атестацію;

– переглянути успішність групи.

Для адміністратора:

– додати предмети, групи, викладачів;

– налаштувати періоди освітніх подій;

– контролювати сесію, атестації та звітність.

Діаграма прецедентів зображена на кресленнику ІС12.150БАК.005 Д4.

Інтерфейс інтуїтивно зрозумілий, адаптований під мобільні пристрої, всі поля мають валідацію, дії підтверджуються через спливаючі повідомлення. Далі зображено скріншоти основних сторінок додатку.

Сторінка профілю (рисунок 4.5) користувача містить детальну інформацію про поточного користувача, включаючи ім’я, логін, роль у системі (наприклад, студент, викладач або адміністратор) та дату реєстрації.

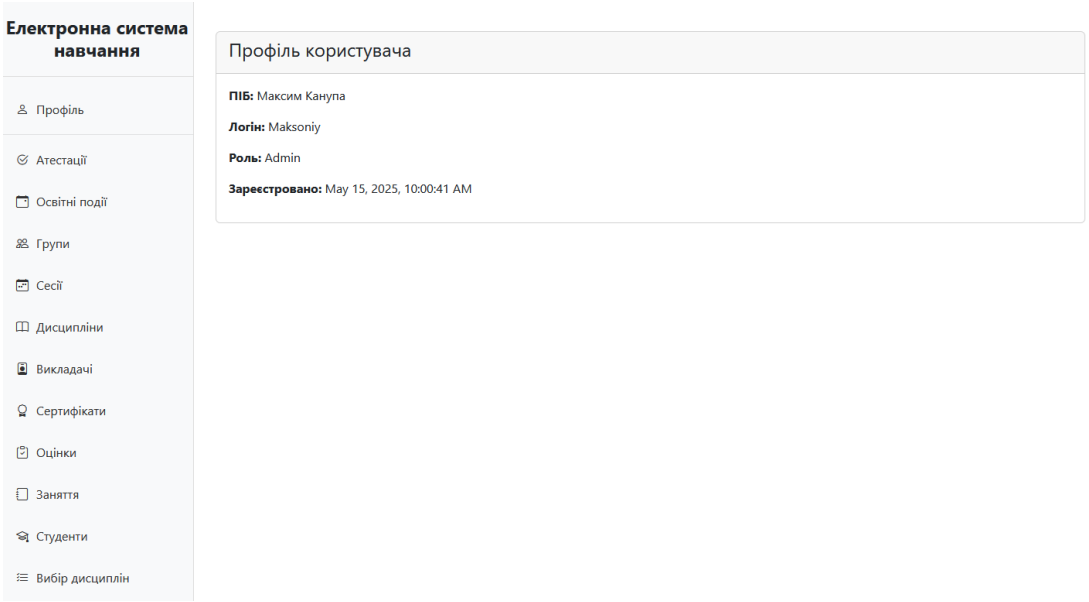


Рисунок 4.5 – Сторінка профілю користувача

Сторінка груп (рисунок 4.6) відображає перелік усіх навчальних груп у системі. Користувач може переглядати список груп, здійснювати пошук та фільтрацію за назвою, кількістю учасників, відповідальним викладачем. Для адміністратора доступна можливість створення нової групи або редагування наявних, що включає зміну складу студентів, додавання або видалення учасників. Дії з управління групами супроводжуються спливаючими повідомленнями для підтвердження змін або вказівки на помилки.

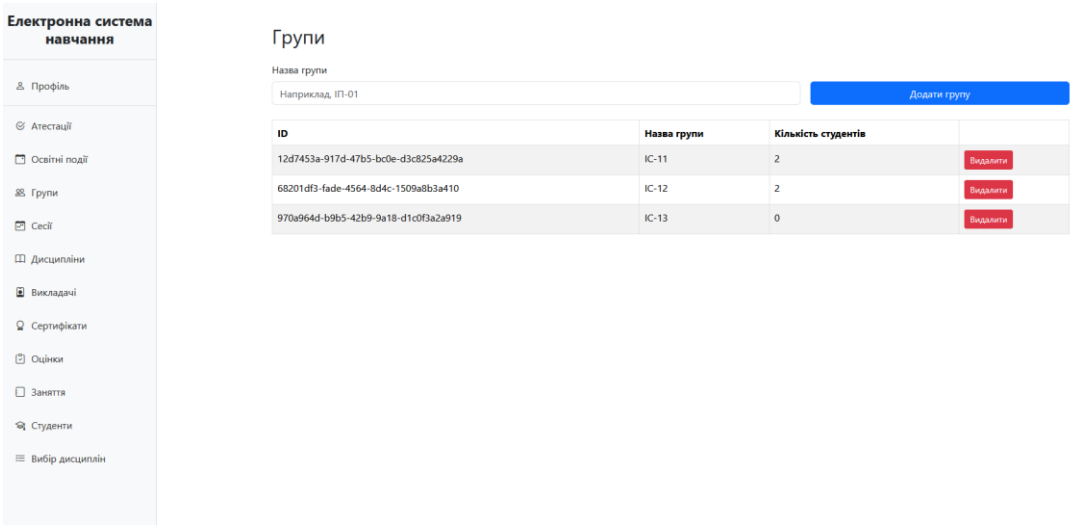


Рисунок 4.6 – Сторінка груп

Сторінка «Предмети» (рисунок 4.7) містить перелік дисциплін, які викладаються у закладі освіти. Інтерфейс дозволяє переглядати детальну інформацію про кожну дисципліну: назву, опис, відповідального викладача, кількість годин. Для адміністратора та викладача передбачено можливість додавання нових дисциплін, редагування опису та складу навчальних матеріалів.

Також адміністратор може видаляти дисципліни. Функціонал сторінки передбачає пошук дисциплін за назвою, що полегшує навігацію.

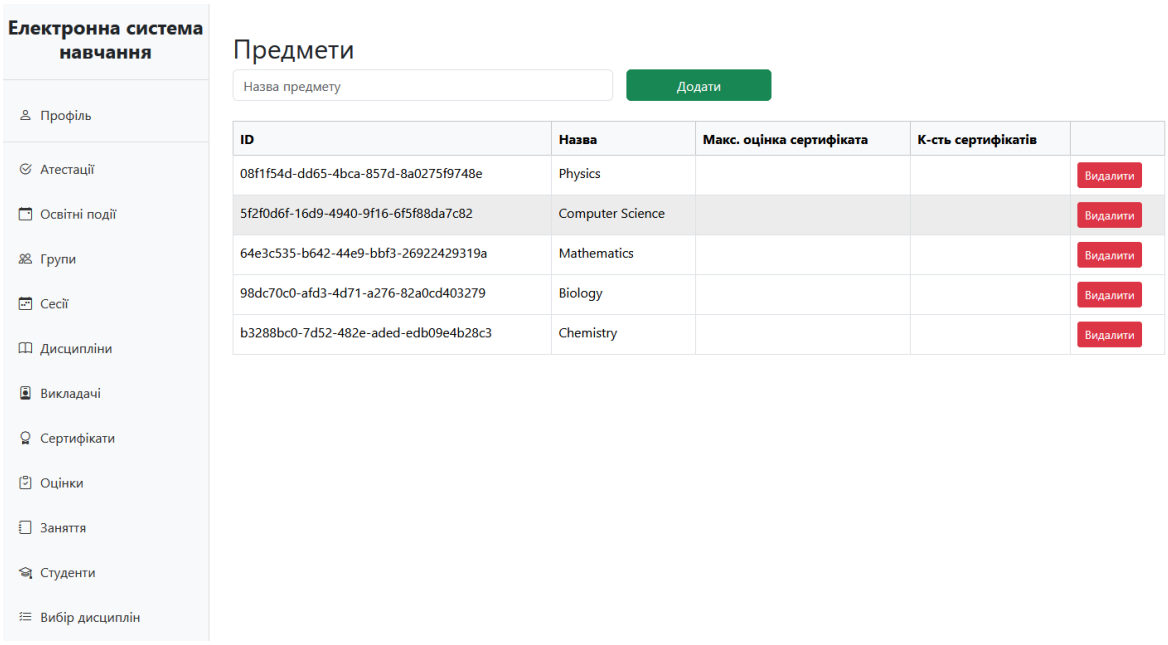


Рисунок 4.7 – Сторінка дисциплін

Сторінка розкладу (рисунок 4.8) є однією з ключових сторінок системи, оскільки вона дозволяє користувачам переглядати актуальний розклад занять. Розклад відображається у табличній формі з зазначенням днів тижня, часу занять, назв дисциплін, аудиторій та викладачів.

Для зручності реалізовано можливість фільтрації розкладу за групами або викладачами. У разі внесення змін до розкладу користувачі бачать спливаючі повідомлення, що інформують їх про оновлення. Адаптивний дизайн дозволяє коректно переглядати розклад навіть на мобільних пристроях, забезпечуючи доступність інформації у будь-який час.

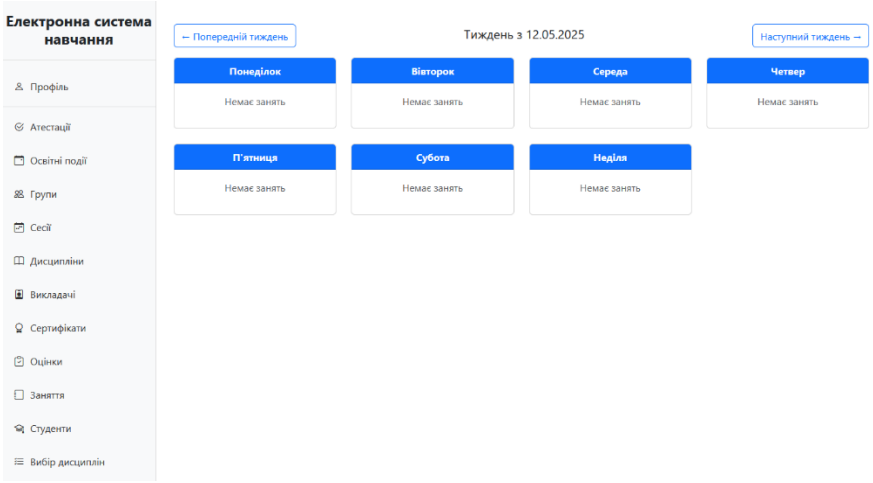


Рисунок 4.8 – Сторінка розкладу

Сторінка освітніх подій (рисунок 4.9) надає користувачам доступ до списку основних освітніх подій закладу освіти. Інтерфейс сторінки реалізовано у вигляді таблиці, де відображаються дати проведення подій, їх назви та короткий опис. Для зручності передбачена можливість швидкого перегляду переліку подій, що дозволяє користувачам завжди бути в курсі важливих заходів. Адміністратор системи має можливість редагувати дані про події, додаючи нові, змінюючи існуючі або видаляючи неактуальні.

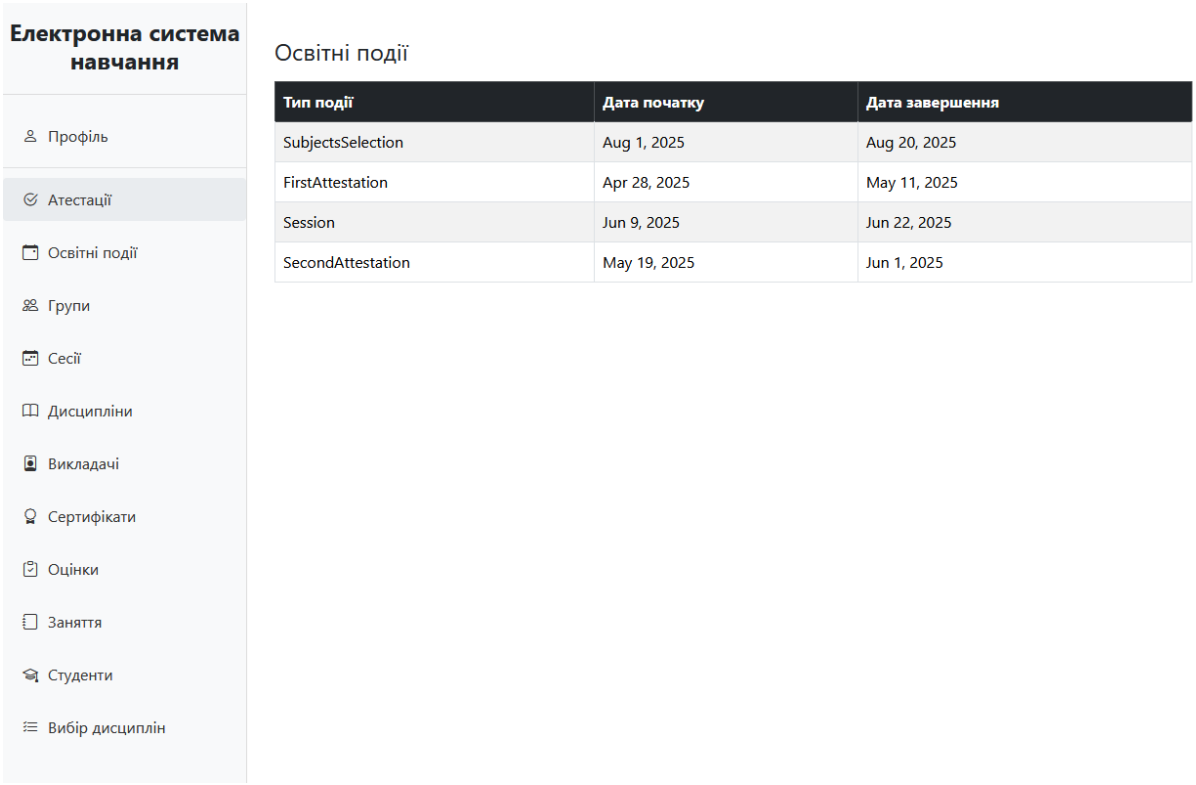


Рисунок 4.9 – Сторінка освітніх подій

Сторінка сертифікатів (рисунок 4.10) забезпечує користувачам доступ до переліку отриманих сертифікатів з неформальної освіти, таких як Coursera, Prometheus та інші платформи. Інтерфейс сторінки реалізовано у вигляді таблиці, де відображаються назви курсів, дати отримання сертифікатів та оцінка. Це дозволяє студентам відслідковувати свої додаткові освітні досягнення і додавати нові пройдені курси, а адміністраторам – інтегрувати ці дані у загальну систему навчального обліку.

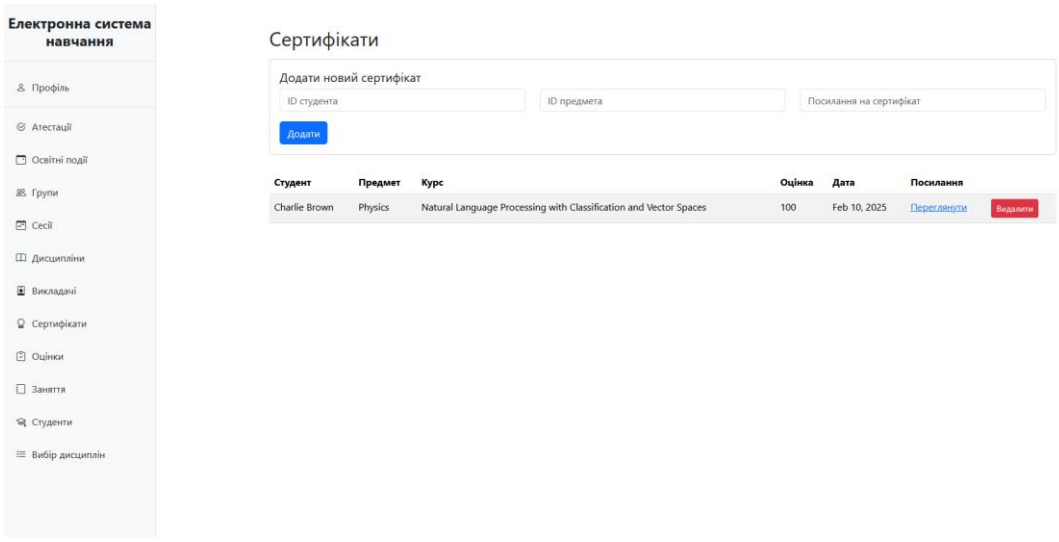


Рисунок 4.10 – Сторінка сертифікатів

Сторінка оцінок (рисунок 4.11) дозволяє користувачам переглядати результати навчання у зручному табличному форматі. На сторінці відображаються оцінки за дисциплінами, включаючи інформацію про назву курсу, дату здачі, отриманий бал та коментарі викладача. Це забезпечує студентам можливість легко відстежувати свій прогрес та аналізувати результати. Для адміністратора передбачена можливість додавання та редагування оцінок, що дозволяє оперативно вносити зміни та підтримувати актуальність інформації. Сторінка має адаптивний дизайн, що забезпечує коректне відображення даних на різних пристроях та дозволяє зручно користуватися системою у будь-якому середовищі. Діаграма діяльності процесу виставлення оцінки зображена на кресленнику ІС12.150БАК.005 Д3.

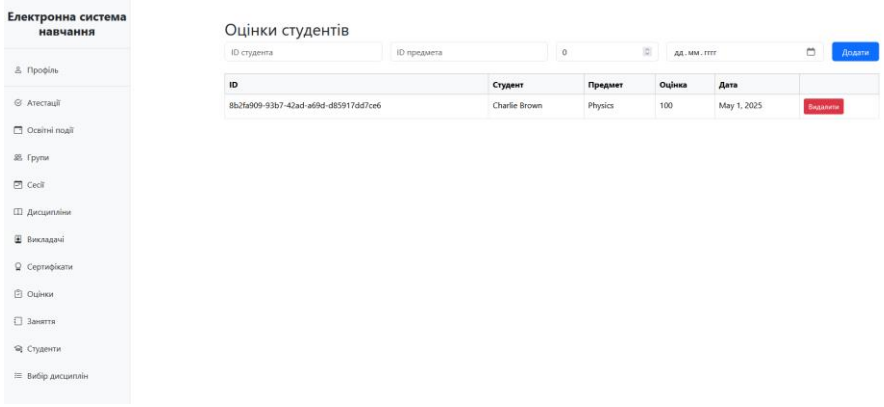


Рисунок 4.11 – Сторінка оцінок

Таким чином, інформаційна система управління навчальним процесом розроблена з урахуванням сучасних вимог до зручності, продуктивності та гнучкості використання. Інтерфейс системи інтуїтивно зрозумілий, адаптований до мобільних пристроїв, підтримує валідацію даних і надає користувачам зрозумілі повідомлення про виконані дії. Ретельно продумана архітектура дозволяє розділити обов'язки між серверною та клієнтською частинами, забезпечуючи ефективну обробку даних і безпечний обмін інформацією. Кожна зі сторінок системи має чітке призначення і логічну структуру, що дозволяє користувачам швидко знаходити потрібну інформацію і здійснювати необхідні дії. Такий підхід до розробки створює передумови для масштабування, розвитку функціональності та інтеграції із зовнішніми сервісами, що робить систему гнучким інструментом для підвищення ефективності управління навчальним процесом у закладах освіти.

## 4.8 Безпека системи

Забезпечення безпеки є одним із ключових аспектів розробки та впровадження інформаційної системи управління навчальним процесом. Система реалізована з урахуванням сучасних стандартів кібербезпеки, що гарантує надійний захист інформації, обмеження несанкціонованого доступу та захист даних користувачів від можливих атак.

Одним із основних елементів безпеки системи є механізм авторизації та автентифікації. Для цього використовується технологія JWT (JSON Web Tokens), яка дозволяє передавати унікальний токен з інформацією про роль користувача, його ідентифікатор та строк дії. Токен створюється при вході користувача до системи та передається у заголовках HTTP-запитів, де перевіряється сервером на валідність. Цей механізм дозволяє забезпечити рольовий доступ, який обмежує функціональність користувачів відповідно до їх прав [7].

Основні частини JWT токену зображено на рисунку 4.12. Типовий JWT токен складається з трьох основних частин, розділених крапками: Header, Payload та Signature. Header містить інформацію про алгоритм підпису та тип токену, Payload

– дані про користувача та інші claims, що описують контекст, наприклад, ідентифікатор користувача, його роль у системі та термін дії токену. Signature створюється шляхом підписування закодованих Header та Payload за допомогою секретного ключа або пари ключів (у випадку використання асиметричного шифрування). Така структура дозволяє забезпечити цілісність та достовірність даних, що передаються у токені, та захищає систему від підробки даних користувача.


Рисунок 4.12 – Структура JWT токену [5]

Авторизація реалізована за допомогою ASP.NET Core Authentication, із використанням протоколу HTTPS, що забезпечує захищений канал передачі даних. Паролі користувачів перед збереженням у базі даних шифруються з використанням bcrypt та засобів ASP.NET Core, що виключає можливість їх розшифрування у разі отримання доступу до бази даних.

Передача даних між клієнтською та серверною частинами здійснюється виключно через захищений протокол HTTPS, що дозволяє шифрувати інформацію і захищати її від атак типу «людина посередині» (man-in-the-middle).

Система забезпечує надійний захист від типових веб-атак. Зокрема, застосовується захист від SQL-ін'єкцій завдяки використанню Entity Framework Core, який автоматично параметризує запити до бази даних. Для запобігання XSS-атак (міжсайтове виконання скриптів) проводиться перевірка та екранування всіх

вхідних даних. Також використовується політика CORS (Cross-Origin Resource Sharing) для обмеження доступу лише з дозволених доменів, що знижує ризик атак з інших веб-ресурсів.

Валідація даних є обов'язковою як на клієнтській, так і на серверній частині. На стороні клієнта (Angular) застосовуються реактивні форми з перевіркою полів, що забезпечує правильність введених користувачами даних. Серверна частина додатково перевіряє отримані запити за допомогою бібліотеки FluentValidation, яка дозволяє централізовано реалізувати правила перевірки даних.

Для забезпечення прозорості роботи системи та оперативного реагування на можливі інциденти впроваджено механізми журналювання та моніторингу. Використовується система логування Serilog, яка дозволяє відстежувати ключові події, помилки та підозрілі дії користувачів [18].

Таким чином, реалізований комплекс заходів із безпеки дозволяє гарантувати захист даних, забезпечує безпечний доступ до системи для користувачів різних ролей та забезпечує високу надійність роботи програмного забезпечення.

## 4.9 Тестування системи

Забезпечення надійності та стабільності роботи інформаційної системи управління навчальним процесом потребує всебічного тестування, яке охоплює всі рівні розробки програмного забезпечення – від окремих функцій до взаємодії між компонентами та навантаження системи. Тестування виконувалось на всіх ключових етапах створення системи для виявлення та усунення помилок, а також для перевірки відповідності функціоналу вимогам.

### 4.9.1 Юніт-тести

На рівні окремих сервісів серверної частини (ASP.NET Core) реалізовано юніт-тести, що забезпечують перевірку логіки роботи сервісів, правильності перетворення даних у DTO, валідації вхідних даних та обробки помилок.

Для написання юніт-тестів застосовуються бібліотеки xUnit та NSubstitute, що дозволяють створювати незалежні тести для кожної функції. Завдяки юніт-тестам забезпечено високий рівень покриття коду тестами, що дозволяє швидко виявляти помилки під час рефакторингу або розширення системи.

xUnit – це популярна бібліотека для створення юніт-тестів у .NET, яка забезпечує гнучкість та легкість написання тестових сценаріїв. Вона підтримує параметризовані тести, асинхронне тестування, організацію тестових колекцій та інші можливості, що дозволяють ефективно перевіряти логіку сервісів, включаючи різні сценарії роботи, вхідні параметри та очікувані результати. Завдяки гнучкому механізму атрибутів і можливості запуску тестів паралельно, xUnit забезпечує високу швидкість виконання тестового набору [21].

NSubstitute використовується для створення підставних об'єктів (моків), що дозволяє ізолювати тестовані компоненти від їхніх залежностей. Це особливо важливо для перевірки логіки сервісів, які взаємодіють із базою даних, зовнішніми сервісами або іншими частинами системи. За допомогою NSubstitute можна створювати імітації інтерфейсів, задавати поведінку методів і перевіряти, як саме ці методи викликаються під час тестування. Це дозволяє перевірити не тільки результат функції, а й правильність взаємодії з іншими компонентами [15].

Для генерації тестових даних використовується бібліотека Bogus, яка дозволяє створювати реалістичні випадкові значення для об'єктів, що тестуються. Це забезпечує варіативність тестів, дозволяючи виявляти помилки, які можуть виникати при обробці різноманітних комбінацій вхідних даних. Використання Bogus особливо корисне для перевірки функцій валідації, перетворення даних та обробки сценаріїв з граничними або некоректними значеннями [4].

Завдяки впровадженню юніт-тестів забезпечено високий рівень покриття коду тестами. Тести дозволили перевірити коректність роботи основних бізнес-правил, зменшити ризик помилок під час внесення змін до коду, підвищити впевненість у стабільності та надійності системи. В процесі тестування були виявлені та виправлені помилки у логіці перетворення даних, обробці нестандартних вхідних значень та взаємодії сервісів.

У підсумку, впровадження xUnit, NSubstitute та Bogus дозволило створити повноцінну систему автоматичного тестування серверної частини, яка забезпечує якість коду, полегшує супровід системи та зменшує час на перевірку працездатності після внесення змін. Це робить систему більш надійною, продуктивною та готовою до масштабування у майбутньому.

### 4.9.2 Ручне тестування

Для клієнтської частини (Angular) та всієї системи в цілому реалізовано комплексне ручне тестування, яке дозволяє перевірити коректність роботи усіх компонентів у реальному середовищі. Цей підхід дозволяє виявити помилки або невідповідності, які можуть залишатися непоміченими під час автоматичного тестування. Ручне тестування охоплює повний цикл взаємодії користувача із застосунком – від входу до системи до виконання основних бізнес-функцій, забезпечуючи надійність роботи та високу якість інтерфейсу [2].

Тестування розпочинається з перевірки правильності навігації по сторінках додатку. Використовуючи Firefox Developer Tools, перевіряється завантаження всіх компонентів інтерфейсу, правильність переходів між сторінками, коректність відображення меню та кнопок. Особливу увагу приділено адаптивності інтерфейсу – перевірено, як система відображається на різних розмірах екранів, включаючи мобільні пристрої.

Далі тестуються процеси авторизації та автентифікації. Перевіряється правильність введення облікових даних, обробка помилок (наприклад, введення неправильного пароля), використання JWT для захисту сесій користувачів, коректність збереження токенів у LocalStorage або SessionStorage. Ці перевірки здійснюються як через сам Angular застосунок, так і за допомогою Postman, де вручну надсилаються HTTP-запити на ендпоінти авторизації для перевірки коректності відповіді сервера та обробки різних сценаріїв, включаючи помилки.

Наступним етапом є перевірка валідації форм, що здійснюється безпосередньо у браузері та за допомогою Firefox Developer Tools. Тестуються всі

поля введення: перевіряється наявність підказок, повідомлень про помилки, блокування кнопок у разі некоректних даних. Ретельно перевірено обробку сценаріїв із порожніми полями, помилками у форматі введення (наприклад, некоректні email-адреси), а також правильність валідації на серверній стороні.

Далі перевіряються основні бізнес-функції системи, зокрема створення та редагування уроків, вибір дисциплін, перегляд розкладу та оцінок. Тестувальник вручну заповнює форми, перевіряє правильність збереження даних, коректне відображення введеної інформації після перезавантаження сторінки або повторного входу. Особливу увагу приділено перевірці правильності взаємодії з API: у Postman надсилаються запити до відповідних ендпоінтів, перевіряється структура відповідей, статус-коди та коректність обробки даних.

На завершення проводиться перевірка повідомлень про помилки та підтвердження дій. Усі спливаючі повідомлення перевіряються на предмет інформативності, правильності тексту, а також наявності їх автоматичного зникнення або можливості закриття користувачем. Перевіряється, як система реагує на спроби виконати некоректні або заборонені дії.

Ручне тестування забезпечує високу якість кінцевого продукту, дозволяє виявити логічні помилки, недопрацювання у валідації, помилки відображення даних та проблеми з навігацією. Завдяки застосуванню інструментів, таких як Firefox Developer Tools та Postman, тестувальники отримують повний контроль над усіма аспектами роботи системи, що дозволяє ефективно перевірити як клієнтську, так і серверну частину застосунку. У підсумку це забезпечує стабільну та зручну взаємодію користувачів із системою в реальному середовищі.

### 4.9.3 Навантажувальне тестування

Навантажувальне тестування системи здійснюється за допомогою сучасного інструмента k6, який дозволяє ефективно оцінити продуктивність та стійкість вебзастосунку під умовами високого навантаження. k6 – це високопродуктивний інструмент для створення та запуску скриптів навантажувального тестування,

розроблений з орієнтацією на розробників і тестувальників. Він використовує JavaScript для написання скриптів, що забезпечує гнучкість і можливість легко моделювати різні сценарії поведінки користувачів [9].

У рамках тестування системи за допомогою k6 було змодельовано одночасну роботу великої кількості користувачів, які надсилали запити до API сервісу. Ці запити включали автентифікацію, отримання даних, створення та оновлення записів, а також інші ключові бізнес-операції. Тестові скрипти були налаштовані для генерації пікового навантаження, що дозволило перевірити, як система поводиться при різкому зростанні кількості одночасних підключень.

Основними метриками, які вимірювалися під час тестування, були час відгуку сервера (response time), кількість запитів за секунду (requests per second), частота помилок (error rate), а також стійкість системи до тривалого високого навантаження (stability under load). Результати показали, що система здатна обробляти значну кількість одночасних запитів без суттєвих затримок. Середній час відгуку залишався в межах допустимих значень навіть при пікових навантаженнях, тоді як кількість запитів за секунду демонструвала високу стабільність.

Під час тестування були також виявлені окремі вузькі місця, пов’язані з обробкою складних запитів до бази даних та підготовкою деяких відповідей API. Це дозволило своєчасно внести оптимізації до коду, що покращило загальну продуктивність та забезпечило підготовленість системи до роботи в умовах реального середовища із великою кількістю користувачів.

Таким чином, завдяки використанню k6 було отримано цінні дані про продуктивність і стійкість системи, виявлено потенційні вузькі місця, а також підтверджено готовність застосунку до експлуатації в умовах високого навантаження, що забезпечує стабільну та надійну роботу у реальних умовах.

Висновок до розділу 4

У цьому розділі було детально описано архітектуру, функціональну модель та реалізацію інформаційної системи управління навчальним процесом у закладі освіти. Було визначено багаторівневу структуру системи, яка складається з клієнтської частини (Angular), серверної частини (ASP.NET Core), реляційної бази даних (PostgreSQL), фонового обробника задач (Hangfire) та інтеграції із зовнішніми сервісами (Coursera, Prometheus).

Функціональна модель системи, представлена через акторів та сценарії використання, відображає основні можливості системи для адміністратора, викладача та студента. Описана модель бази даних з розподілом на основні сутності та міжсистемні зв'язки забезпечує збереження цілісності та консистентності даних.

Архітектура програмного забезпечення, побудована за принципами Clean Architecture, гарантує модульність, розширюваність та легкість підтримки коду. Додаткові аспекти розробки, такі як передавання даних у форматі JSON із використанням JWT для автентифікації, забезпечують захищену та надійну комунікацію між компонентами.

Діаграми класів і структура проєкту демонструють логічну організацію системи, що сприяє чіткому розумінню її будови та ефективному управлінню процесами розробки.

Завдяки продуманій структурі та реалізації система відповідає сучасним вимогам до веб-застосунків в освітній сфері, забезпечує зручність використання для всіх категорій користувачів і створює передумови для подальшого розвитку та інтеграції з іншими освітніми сервісами.

# 5 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

## 5.1 Змістовна постановка задачі

Формування розкладу занять для студентів навчального закладу є однією з ключових задач управління навчальним процесом. Завдання полягає у визначенні оптимального розміщення занять у доступних часових інтервалах та аудиторіях із урахуванням обмежень: обмеженої кількості ресурсів (аудиторій, викладачів), відсутності конфліктів між заняттями однієї групи або з участю одного викладача, а також вимог до впорядкування занять за пріоритетністю чи директивними строками.

У дипломному проєкті з розробки інформаційної системи управління навчальним процесом доцільно використати задачу складання розкладу з урахуванням обмежень на ресурси (RCPSP – Resource-Constrained Project Scheduling Problem). Ця задача дозволяє моделювати процес формування розкладу занять для студентів з урахуванням обмежень, таких як доступність аудиторій, викладачів та часових слотів. Для вирішення цієї задачі можна застосувати метод Джексона (Jackson's Rule), який є ефективним для задач з директивними строками [1].

Задача полягає у формуванні розкладу занять для студентів, що забезпечує оптимальне використання ресурсів (аудиторій, викладачів) та дотримання всіх обмежень, таких як:

– кожна дисципліна повинна бути проведена у визначеній кількості годин;

– викладачі можуть проводити заняття лише у визначені години;

– заняття не повинні перекриватися для однієї групи студентів.

Метою є мінімізація загального часу проведення занять (makespan) та забезпечення рівномірного розподілу навантаження.

5.2 Математична постановка задачі

Нехай маємо множину занять $J = \{1, 2, \dots, n\}$, кожне з яких характеризується:

– тривалістю $p_j$;

– набором необхідних ресурсів $R_j$;

– множиною попередників $P_j$ (занять, які повинні бути завершені до початку $j$).

Маємо також множину ресурсів $R = \{1,2,\dots, m\}$, кожен з яких має обмежену доступність $k_r$.

Змінні:

– $S_j$ — час початку заняття $j$.

Мета: мінімізувати час завершення останнього заняття (формула 5.1):

$$\min\max_{j \in J}(S_j + p_j), \tag{5.1}$$

Обмеження:

1. Обмеження передування (формула 5.2):

$$\forall j \in J, \forall i \in P_j : S_j \geq S_i + p_i, \tag{5.2}$$

2. Обмеження ресурсів:

У будь-який момент часу сумарне використання кожного ресурсу не перевищує його доступність (формула 5.3):

$$\forall t, \forall r \in R: \sum_{j \in J : S_j \leq t < S_j + p_j} r_{j,r} \leq k_r, \tag{5.3}$$

де $r_{j,r}$ — кількість ресурсу $r$, необхідна для заняття $j$.

3. Обмеження на розклад:

Заняття не перекриваються для однієї групи студентів.

Для кожної групи $g$ і для кожної пари занять $j, j'$, які призначені для цієї групи (формула 5.4):

$$\forall g \in G, \forall j, j' \in J_g, j \neq j' : S_j + p_j \leq S_{j'} \quad \text{або} \quad S_{j'} + p_{j'} \leq S_j, \tag{5.4}$$

де:

- $J_g$ — множина занять, призначених для групи $g$;
- $S_j$ — час початку заняття $j$;
- $p_j$ — тривалість заняття $j$.

## 5.3 Обґрунтування методу розв'язання

Задача RCPSP (Resource-Constrained Project Scheduling Problem), яка формалізує процес складання розкладу з обмеженнями на ресурси, є однією з найскладніших задач дискретної оптимізації. Вона належить до класу NP-складних задач, що означає, що для великих обсягів даних (значної кількості занять, обмежених ресурсів та складних умов) не існує відомого поліноміального алгоритму, який би гарантував оптимальне рішення за прийнятний час.

На сьогодні у науковій та прикладній практиці застосовуються різні підходи до розв'язання цієї задачі, які можна умовно розділити на дві великі групи: точні методи та наближені методи.

Точні методи включають:

- метод повного перебору: вичерпний пошук усіх можливих варіантів розкладу, застосовується лише для дуже малих задач, оскільки обчислювальні витрати зростають експоненціально;
- метод гілок і меж: дозволяє зменшити простір пошуку за рахунок відсікання гілок, які не можуть призвести до оптимального рішення, ефективний для невеликих задач, але потребує значних обчислювальних ресурсів;

– цілочислове програмування: дозволяє сформулювати задачу у вигляді системи лінійних обмежень і цільової функції, однак рішення таких задач для великих обсягів даних вимагає використання потужних обчислювальних засобів.

Наближені методи включають:

– евристики: наприклад, жадібні алгоритми, де заняття призначаються на найближчі можливі часові слоти або ресурси, такі методи не гарантують оптимального рішення, але дозволяють швидко отримати практично прийнятний результат;
– правила диспетчеризації: SPT (shortest processing time), LPT (longest processing time), EDD (earliest due date) дозволяють впорядковувати роботи за певними критеріями, зменшуючи час простою або запізнень;
– метаевристики: алгоритми, що поєднують пошук у просторі рішень з використанням спеціальних стратегій, наприклад, алгоритм мурашиних колоній (ant colony optimization), генетичні алгоритми (genetic algorithms), симульоване відпалу (simulated annealing), вони ефективні для складних задач великої розмірності, але потребують тонкого налаштування параметрів.

У нашому випадку формування розкладу для студентів в інформаційній системі управління навчальним процесом є практичною задачею, де важливо швидко отримати рішення, яке забезпечує баланс між обмеженнями на ресурси та якістю розкладу. Враховуючи обмеження на доступні ресурси (кількість аудиторій, доступність викладачів) і необхідність дотримання директивних строків (наприклад, проведення важливих лекцій до певної дати), було обрано евристичний підхід, заснований на методі Джексона (Jackson's Rule).

Метод Джексона дозволяє ефективно впорядкувати заняття за зростанням директивних строків (термінів, до яких має бути завершено заняття) і забезпечити їх розміщення у перші доступні часові слоти з урахуванням обмежень на ресурси. Це дозволяє мінімізувати загальне запізнення занять та уникнути конфліктів за ресурси. Метод є простим для реалізації, має низьку обчислювальну складність

порівняно з точними методами і забезпечує результат, який на практиці є достатньо якісним для реального використання у навчальному процесі.

Таким чином, було обґрунтовано вибір саме методу Джексона для розв'язання задачі RCPSP у контексті побудови розкладу для студентів у дипломному проєкті. Цей метод дозволяє отримати оптимізований розклад, що враховує обмеження ресурсів, взаємозалежності занять та їх пріоритети, забезпечуючи ефективність і практичну застосовність системи.

## 5.4 Опис методу розв'язання

Враховуючи складність задачі RCPSP, яка належить до класу NP-складних задач, для розробки інформаційної системи управління навчальним процесом було обрано евристичний підхід, що поєднує простоту реалізації та практичну ефективність. Зокрема, використовується метод Джексона, який дозволяє впорядкувати заняття за пріоритетом директивних строків і призначити їх у перші доступні часові слоти.

Алгоритм Джексона:

– впорядкувати всі заняття за зростанням директивних строків $d_j$;

– послідовно призначати заняття у перший доступний часовий слот, враховуючи обмеження на ресурси та передування;

– у разі конфлікту вибирати заняття з найменшим $d_j$.

Розглянемо приклад. Нехай маємо 3 заняття з наступними характеристиками (таблиця 5.1).

Таблиця 5.1 – Характеристики занять для прикладу

| Заняття | Тривалість $p_j$ | Директивний строк $d_j$ | Попередники $P_j$ |
|---|---|---|---|
| 1 | 2 | 5 | - |
| 2 | 3 | 7 | 1 |
| 3 | 1 | 6 | 1 |

Впорядкування за $d_j$ (заняття 1, 3, 2):

- заняття 1: $S_1 = 0$;
- заняття 3: $S_3 = S_1 + p_1 = 2$;
- заняття 2: $S_2 = S_1 + p_1 = 2$.

Таким чином, розклад задовольняє всі обмеження. Застосування методу Джексона дозволяє створити розклад занять, який забезпечує мінімізацію запізнень та ефективне використання ресурсів.

Цей метод відзначається простотою реалізації та високою ефективністю для практичних задач, що робить його оптимальним вибором для розробленої інформаційної системи управління навчальним процесом.

Висновок до розділу 5

У цьому розділі було детально розглянуто математичне забезпечення інформаційної системи управління навчальним процесом, зокрема задачу формування розкладу занять із урахуванням обмежень на ресурси (RCPSP). Було описано як змістовну, так і математичну постановку задачі, що включає всі необхідні змінні, обмеження та критерії оптимальності. Також проведено порівняльний аналіз існуючих методів розв'язання задачі, що дало змогу обґрунтувати вибір найбільш ефективного підходу для даного проєкту. Зокрема, серед широкого спектру методів було обрано евристичний алгоритм Джексона, який забезпечує простоту реалізації, високу швидкість отримання результатів та практичну ефективність навіть у складних умовах.

Метод Джексона дозволяє ефективно впорядковувати заняття за директивними строками та забезпечує їх розміщення у перші доступні часові слоти, що мінімізує конфлікти за ресурси та запізнення занять. Його застосування до задачі розкладу дозволяє зменшити час простою ресурсів, забезпечити більш рівномірне навантаження викладачів та аудиторій, а також знизити ймовірність виникнення логічних конфліктів між заняттями. Наведений приклад із трьома заняттями демонструє практичну застосовність цього алгоритму, що є

підтвердженням його ефективності в умовах обмежених ресурсів та взаємозалежностей між навчальними подіями.

Крім основного алгоритму, були також розглянуті можливі варіанти розширення математичної моделі, наприклад, за рахунок врахування змінної доступності ресурсів у часі, пріоритетів дисциплін, додаткових вимог до сумісності занять та обліку зовнішніх факторів (змішане навчання, онлайн-формати тощо). Це дозволяє гнучко адаптувати систему до специфіки конкретного навчального закладу та потреб сучасного освітнього середовища.

Тож, математичне забезпечення є важливою складовою розробки інформаційної системи управління навчальним процесом, що забезпечує її надійність, гнучкість та можливість масштабування для різних умов роботи навчального закладу. Воно є основою для формалізації задач розкладу, оптимального розподілу ресурсів, аналізу даних та прогнозування освітніх результатів. Завдяки обраному підходу система здатна адаптуватися до змін у розкладі, швидко обробляти нові дані та забезпечувати високу якість управлінських рішень у сфері освітньої діяльності. Це сприяє підвищенню ефективності функціонування закладу освіти, зниженню людського фактору в управлінських процедурах та підтримці безперервного вдосконалення освітнього процесу.

# ВИСНОВКИ

Система управління навчальним процесом є популярним та необхідним рішенням у сучасних закладах освіти, адже вона дозволяє автоматизувати ключові процеси навчальної діяльності, забезпечуючи зручний доступ до розкладів, оцінок та відвідуваності. У дипломному проєкті було створено інформаційну систему, яка об'єднує можливості ASP.NET для реалізації стабільного та надійного серверного середовища та Angular для створення гнучкого та зручного клієнтського інтерфейсу. Система забезпечує можливість створення розкладів занять, облік відвідуваності та успішності студентів та надає користувачам зручний доступ до необхідної інформації. Одним із основних досягнень системи є підтримка результатів неформальної освіти у вигляді сертифікатів, які студенти можуть завантажувати самостійно, після чого система автоматично обробляє дані сертифікату та враховує їх як частину загальної успішності з відповідної дисципліни.

Розроблений продукт пройшов комплексне тестування, включаючи юніт-тести на серверній частині, які перевіряють логіку бізнес-процесів, валідацію даних та обробку помилок. Крім того, проведено ручне тестування клієнтської частини із застосуванням інструментів перевірки правильності навігації, обробки запитів та відповідей API, а також валідації форм. Оцінки продуктивності системи під навантаженням дозволила змоделювати роботу великої кількості користувачів та виявити вузькі місця в архітектурі. У результаті тестування система показала стабільність роботи при значному навантаженні, прийнятний час відгуку та високу стійкість.

Слід відзначити, що під час розробки було виявлено необхідність додаткової оптимізації взаємодії з базою даних при високому навантаженні. Проте ці питання були вирішені завдяки впровадженню оптимізованих алгоритмів обробки даних і використанню перевірених технологій. В результаті отримано систему, яка задовольняє вимогам дипломного проєкту, відповідає поставленій меті та дозволяє

ефективно вирішувати завдання управління навчальним процесом у навчальному закладі.

Отримані результати демонструють можливість реального впровадження системи у практичну діяльність закладів освіти різного рівня. Система здатна підвищити ефективність управління навчальним процесом, зменшити навантаження на адміністративний персонал, забезпечити студентам і викладачам зручний доступ до актуальної інформації, підвищити прозорість і якість навчального процесу. У перспективі передбачено можливість масштабування функціоналу, інтеграції з додатковими сервісами та розвитку інтерфейсу для ще більшої зручності користувачів. Таким чином, дипломний проєкт повністю відповідає вимогам, поставленим у завданні на проєктування, та підтверджує доцільність і ефективність розробленого рішення.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. An optimization model to solve the resource constrained project scheduling problem RCPSP in new product development projects. *Sistema de Información Científica Redalyc, Red de Revistas Científicas*. URL: https://www.redalyc.org/journal/496/49663642022/html/
2. Angular Official Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/docs
3. Bootstrap Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/docs/
4. Chavez B. Bogus: A simple fake data generator for .NET [Електронний ресурс] // GitHub. – Режим доступу: https://github.com/bchavez/Bogus
5. Dev D. F. S. JWT, OAuth, OpenID Connect, and Azure AD: Authentication Levels. *DEV Community*. URL: https://dev.to/dotnetfullstackdev/jwt-oauth-openid-connect-and-azure-ad-authentication-levels-4a4m
6. Evans E. Best Practice: An Introduction to Domain-Driven Design [Електронний ресурс] // MSDN Magazine. – Режим доступу: https://learn.microsoft.com/en-us/archive/msdn-magazine/2009/february/best-practice-an-introduction-to-domain-driven-design
7. Getachew S. JWT Authentication in .NET 8: A Complete Guide for Secure and Scalable Applications. *Medium*. URL: https://medium.com/@solomongetachew112/jwt-authentication-in-net-8-a-complete-guide-for-secure-and-scalable-applications-6281e5e8667c
8. Google Classroom: All you need to know | Hiver™. *Hiver: AI-powered customer service platform*. URL: https://hiverhq.com/blog/google-classroom-basics
9. Grafana k6 | Grafana k6 documentation. *Grafana Labs*. URL: https://grafana.com/docs/k6/latest/.
10. Mapster Documentation – Object Mapping for .NET [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/MapsterMapper/Mapster/wiki

11. Microsoft Docs. ASP.NET Core Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/aspnet/core/
12. Microsoft Docs. C# Programming Guide [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/dotnet/csharp/
13. Microsoft Docs. Entity Framework Core Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/ef/core/
14. Microsoft. Documentation for Visual Studio Code. *Visual Studio Code - Code Editing. Redefined*. URL: https://code.visualstudio.com/docs
15. NSubstitute. Getting started [Електронний ресурс] // NSubstitute Documentation. – Режим доступу: https://nsubstitute.github.io/help/getting-started/
16. PostgreSQL 17.5 Documentation. *PostgreSQL Documentation*. URL: https://www.postgresql.org/docs/current/index.html
17. Prometheus – Платформа онлайн-курсів [Електронний ресурс] – Режим доступу до ресурсу: https://prometheus.org.ua/
18. Serilog Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://serilog.net/
19. Visual Studio documentation. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/visualstudio/windows/?view=vs-2022
20. What is Moodle? The ultimate guide to Moodle LMS. *Hubken Group: Totara & Moodle E-learning Platforms By The LMS Experts*. URL: https://www.hubkengroup.com/resources/what-is-moodle-lms-guide
21. xUnit.net. Unit testing C# in .NET using dotnet test and xUnit [Електронний ресурс] // Microsoft Learn. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-csharp-with-xunit
22. Бугай А. В., Костікова М. В. Інформаційні технології в системі сучасної освіти // Харківський національний автомобільно-дорожній університет. – 2021. – Режим доступу: https://dspace.khadi.kharkov.ua/bitstreams/6df1ed46-3c06-490e-8313-fa734a588c91/download.

23. Документація по Rider [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/help/rider/Introduction.html
24. Круглик В. С. Сучасні підходи до використання інформаційно-комунікаційних технологій у навчанні // Херсон: Вид. ХДУ, 2008. – С. 114–119.
25. Офіційний сайт Coursera. Сертифікати [Електронний ресурс] – Режим доступу до ресурсу: https://www.coursera.support/s/article/learner-000001187?language=en_US
26. Рамський Ю. С. Використання відкритих онлайн-курсів в умовах змішаного навчання майбутніх фахівців з інформаційних технологій // Інформаційні технології і засоби навчання. – 2021. – №1(81). – С. 1–18. – Режим доступу: https://elibrary.kubg.edu.ua/39420/1/A_Ramskyi_ITLT_FITU.pdf