Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра математичного моделювання та аналізу даних
До захисту допущено
Кафедрою ММ та АД протокол №___ від_____

завідувач кафедри _____ *Володимир СТРУКОВ*
(підпис) (ім'я, прізвище)

«___»_____2026 р.

Кваліфікаційна робота
здобувача першого (бакалаврського) рівня вищої освіти

Розробка інформаційної системи управління навчальним розкладом
(назва роботи)

Спеціальність (спеціалізація) F3 (122) Комп'ютерні науки

Освітня програма Комп'ютерні науки

Виконавець _____ Вікторія Пекар
(підпис) (ім'я, прізвище)

Науковий керівник _____ Олександр Споров
(підпис) (ім'я, прізвище)

Харків – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту Кафедра Математичного моделювання та аналізу даних
Рівень вищої освіти (освітньо-кваліфікаційний рівень) Бакалавр
Галузь знань: F – Інформаційні технології

Спеціальність: F3 – Комп'ютерні науки
Освітня програма: _Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри _ММ та
АД_ Струков Володимир
Михайлович
«_13__» _жовтня_2025_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЕКТ)

Пекар Вікторія Василівна
(прізвище, ім'я, по батькові студента)

1. Тема роботи Розробка інформаційної системи управління навчальним розкладом.

керівник роботи Споров Олександр Євгенович, кандидат фіз.-мат. наук, доцент зво кафедри математичного моделювання та аналізу даних
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "___"_____20__року
№_____

2. Строк подання студентом роботи _____

3. Перелік питань, які потрібно розробити
Аналіз предметної області управління навчальним розкладом у закладах освіти. Дослідження існуючих інформаційних систем для формування навчального розкладу. Формування функціональних та нефункціональних вимог до інформаційної системи. Розробка архітектури інформаційної системи з використанням хмарних технологій. Проєктування структури бази даних системи. Вибір програмних засобів та інструментів для реалізації. Розробка механізму завантаження та автоматичної обробки навчального плану. Реалізація інструменту створення, редагування та видалення елементів розкладу. Розробка

механізму перевірки конфліктів у розкладі (зайнятість викладачів та груп). Реалізація можливості викладачам зазначати доступний час для проведення занять. Розробка функціоналу перегляду розкладу для різних категорій користувачів. Реалізація можливості додавання тем занять, посилань на матеріали та посилань на онлайн-заняття. Проведення тестування системи та аналіз результатів її роботи. Написання пояснювальної записки до кваліфікаційної роботи, створення презентації.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Дослідження процесу формування навчального розкладу у закладах освіти, визначення основних проблем існуючих підходів та формування вимог до інформаційної системи.	01.12 – 21.12.2025
2	Аналіз сучасних програмних рішень і сервісів для планування навчального процесу, визначення їх переваг та недоліків.	23.12.2025 – 07.01.2026
3	Вибір технологій та інструментів розробки; розробка загальної структури системи та визначення взаємодії її компонентів із використанням хмарних технологій.	08.01 – 29.01.26
4	Розробка логічної та фізичної структури бази даних для зберігання інформації про викладачів, студентів, навчальні групи, предмети та розклад занять.	23.12.2025 – 27.01.2026
5	Розробка функціональних модулів системи: завантаження навчального плану, створення та редагування розкладу, перевірка конфліктів.	26.02 – 12.03.2026
6	Використання хмарних сервісів для автоматичної обробки навчального плану та підтримки роботи системи.	13.03 – 12.04.2026
7	Розробка користувацького інтерфейсу системи. Створення інтерфейсів для адміністратора, викладача та студента для перегляду та управління розкладом.	13.04 – 15.05.2026
8	Перевірка коректності функціонування системи, тестування модулів та оцінка ефективності роботи.	16.05 – 25.05.2026

9	Аналіз результатів роботи програмного рішення.	16.05 – 26.05.2026
10	Підготовка пояснювальної записки, презентації та доповіді.	20.05 –
		02.05.2026

5. Дата видачі завдання 13.10.2025

Студент Пекар В.В. _ _
ініціали, прізвище підпис

Керівник роботи Споров О.Є.
ініціали, прізвище підпис

АНОТАЦІЯ

Звіт про виконання: 90 с., 3 розділи, 14 рис., 7 таблиць, 1 додаток, 39 джерел.

У кваліфікаційній роботі досліджено та практично реалізовано процес автоматизованого формування навчального розкладу для закладів вищої освіти з використанням гібридних алгоритмічних підходів та сучасних безсерверних хмарних технологій.

Об’єкт дослідження: процес формування навчального розкладу та адміністрування освітнього навантаження у закладах вищої освіти.

Предмет дослідження: хмарні архітектурні рішення, безсерверні технології асинхронної обробки даних, моделі та алгоритми виявлення просторово-часових конфліктів, а також програмні засоби їх реалізації у вебінтерфейсі системи розкладу.

Мета роботи: обґрунтування та побудова хмароорієнтованої архітектурної моделі інформаційної системи формування розкладу ЗВО, розробка прикладної алгоритмічної схеми асинхронної обробки освітніх даних та інтерактивних вебмодулів взаємодії користувачів із контролем академічних конфліктів у реальному часі.

Розроблено хмароорієнтовану інформаційну систему та спеціалізований вебмодуль для автоматизованого формування розкладу. Запропоновано гібридний детерміновано-еволюційний алгоритм, що поєднує жадібну ініціалізацію для задоволення жорстких обмежень та генетичний пошук для максимального врахування м'яких критеріїв якості (мінімізації технологічних «вікон»). Реалізовано конвеєр асинхронної обробки навчальних планів на базі безсерверної архітектури (AWS Lambda, Amazon S3) та сучасні патерни взаємодії клієнта з сервером (Optimistic UI, Short Polling). Проведено наскрізне тестування, за результатами якого підтверджено високу швидкодію,

відмовостійкість системи та ефективність алгоритму при вирішенні просторово-часових колізій.

Ключові слова: НАВЧАЛЬНИЙ РОЗКЛАД, ГЕНЕТИЧНИЙ АЛГОРИТМ, ХМАРНІ ОБЧИСЛЕННЯ, БЕЗСЕРВЕРНА АРХІТЕКТУРА, AWS LAMBDA, АСИНХРОННА ОБРОБКА ДАНИХ, ІНФОРМАЦІЙНА СИСТЕМА, ЗАКЛАД ВИЩОЇ ОСВІТИ.

ANNOTATIO

Execution report: 90 pages, 3 chapters, 14 figures, 7 tables, 1 applications, 39 sources.

The qualification paper investigates and practically implements the process of automated academic scheduling for higher education institutions using hybrid algorithmic approaches and modern serverless cloud technologies.

Object of the research: the process of academic schedule generation and educational workload administration in higher education institutions.

Subject of the research: cloud architectural solutions, serverless technologies for asynchronous data processing, models and algorithms for detecting spatio-temporal conflicts, as well as software tools for their implementation in the scheduling system's web interface.

Purpose of the work: justification and construction of a cloud-oriented architectural model of the HEI (higher education institution) scheduling information system, development of an applied algorithmic scheme for asynchronous processing of educational data, and interactive web modules for user interaction with real-time academic conflict control.

A cloud-oriented information system and a specialized web module for automated schedule generation have been developed. A hybrid deterministic-evolutionary algorithm is proposed, which combines greedy initialization to satisfy hard constraints and genetic search to maximize the fulfillment of soft quality criteria (minimization of technological "windows"). An asynchronous curriculum processing pipeline based on a serverless architecture (AWS Lambda, Amazon S3) and modern client-server interaction patterns (Optimistic UI, Short Polling) has been implemented. End-to-end testing was conducted, the results of which confirmed the high performance and fault tolerance of the system, as well as the effectiveness of the algorithm in resolving spatio-temporal collisions.

Keywords: ACADEMIC SCHEDULE, GENETIC ALGORITHM, CLOUD COMPUTING, SERVERLESS ARCHITECTURE, AWS LAMBDA, ASYNCHRONOUS DATA PROCESSING, INFORMATION SYSTEM, HIGHER EDUCATION INSTITUTION.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ 11
ВСТУП 12
РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ 16
1.1 Актуальність проблеми автоматизованого формування навчального розкладу у закладах вищої освіти 16
1.2 Дослідження процесу формування навчального розкладу та аналіз існуючих підходів 18
1.3 Аналіз існуючих підходів та засобів автоматизованого формування навчального розкладу 20
1.4 Порівняльний аналіз сучасних програмних рішень і сервісів для планування навчального процесу 23
1.5 Функціональні та нефункціональні вимоги до інформаційної системи 31
РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ 37
2.1 Обґрунтування вибору технологій та інструментів розробки 37
2.2 Архітектура та взаємодія компонентів інформаційної системи 39
2.3 Хмароорієнтована архітектура та інтеграція сервісів AWS 43
2.3.1 Загальна топологія хмарної інфраструктури та сервіси AWS 44
2.3.2 Патерн безпечного безсерверного завантаження файлів (AWS Presigned URL) 46
2.3.3 Модуль автоматичної Serverless-обробки планів (AWS Lambda та Boto3) 47
2.4 Проєктування структури бази даних 47
2.5 Розробка гібридного алгоритму автоматичного формування розкладу 59
2.6 Розробка модулів системи та користувацьких інтерфейсів 66

ЗМІСТ

РОЗДІЛ 3 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ 70
3.1 Методика тестування 70
3.2 Оцінка ефективності роботи системи 73
3.3 Аналіз результатів роботи програмного рішення 75
3.4 Формулювання результатів досліджень та їх оцінка 76
ВИСНОВКИ 79
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 81
ДОДАТОК А. Приклад успішного виконання тестового сценарію 87

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

БД – база даних.

ГА – генетичний алгоритм.

ЗВО – заклад вищої освіти.

ПЗ – програмне

забезпечення.

СУБД – система управління базами даних.

ACID – Atomicity, Consistency, Isolation, Durability (атомарність, узгодженість, ізольованість, довговічність – базові вимоги до транзакційних баз даних).

API – Application Programming Interface (інтерфейс прикладного програмування).

AWS – Amazon Web Services (глобальна хмарна платформа).

EDA – Event-Driven Architecture (подійно-орієнтована архітектура). ER-модель – Entity-Relationship model (модель «сутність-зв'язок»).

FaaS – Function-as-a-Service (модель надання хмарних обчислень «функція як послуга»).

JSON – JavaScript Object Notation (текстовий формат обміну даними).

ORM – Object-Relational Mapping (технологія об'єктно-реляційного відображення баз даних).

RBAC – Role-Based Access Control (управління доступом до системи на основі призначених ролей).

SPA – Single Page Application (вебзастосунок, що працює на одній сторінці).

UX – User Experience (користувацький досвід).

UI - User Interface (користувацький інтерфейс).

ВСТУП

В умовах сучасного динамічного освітнього середовища та стрімкої цифровізації вищої школи управління організаційними процесами у закладах вищої освіти (ЗВО) вимагає впровадження високоефективних інформаційних інструментів. Однією з найбільш трудомістких, багатофакторних та критично важливих задач академічного менеджменту залишається побудова збалансованого навчального розкладу, що безпосередньо впливає на якість освітнього процесу та логістику розподілу людських і часових ресурсів [1-4]. Традиційні підходи, що спираються на ручне або фрагментарне напівавтоматизоване планування, демонструють неспроможність оперативно реагувати на часті зміни в навчальних планах, появу індивідуальних траєкторій студентів та динамічні обмеження професорсько-викладацького складу [4; 8].

Наявні програмні рішення у сфері планування навчального процесу здебільшого орієнтовані на статичний облік аудиторій, базовий імпорт даних або візуалізацію вже сформованих таблиць [9-11]. Разом із тим, було встановлено, що більшість комерційних і десктопних систем мають жорстку монолітну архітектуру, вимагають значних локальних обчислювальних потужностей освітнього закладу та не забезпечують гнучкого асинхронного опрацювання великих масивів даних чи інтерактивної взаємодії користувачів у реальному часі без ризику виникнення системних колізій.

З огляду на викладене, доцільність застосування методів розв'язання складних комбінаторних задач, зокрема еволюційних (генетичних) алгоритмів, мурашиних алгоритмів, імітації відпалу, розфарбовування графів та програмування в обмеженнях для вирішення задачі автоматизованого формування розкладу занять, не викликає сумнівів [5-7; 14-32]. Моделі на основі генетичних підходів, мурашиних алгоритмів та імітації відпалу уможливлюють ефективний пошук рішень у надвеликих просторах станів з урахуванням

множини м'яких обмежень [5; 6; 14-16; 25-32], тоді як алгоритми розфарбовування графів та програмування в обмеженнях дозволяють чітко відображати та виконувати жорсткі дискретні вимоги щодо недопустимості накладання занять в однакових локаціях чи часових слотах [17-22; 24]. Поєднання зазначених математичних та алгоритмічних інструментів із сучасними архітектурними концепціями розподілених систем дозволяє створити стійке до навантажень програмне рішення.

У межах цієї роботи розробка алгоритмів розглядається у прикладному програмно-інженерному сенсі: як побудова послідовності роботи програмного модуля від підготовки вхідних даних і безпечного завантаження файлів планів через підписані посилання (Presigned URLs) до сховища Amazon S3, асинхронного парсингу даних безсерверними функціями AWS Lambda, збереження у реляційній базі даних із використанням транзакцій лише для читання, і до безпосереднього розрахунку колізій та відображення розкладу у вебінтерфейсі на основі патернів Optimistic UI та Short Polling [12; 13; 33; 35-39]. Для побудови надійної архітектури вебмодуля та забезпечення безпеки й моніторингу використовуються інструменти Amazon CloudWatch Logs, принципи багатоорендності та рольова модель доступу (RBAC) [12; 13; 33; 38-39].

Метою бакалаврської роботи є обґрунтування та побудова хмароорієнтованої архітектурної моделі інформаційної системи формування розкладу ЗВО, розробка прикладної алгоритмічної схеми асинхронної обробки освітніх даних та інтерактивних вебмодулів взаємодії користувачів із контролем академічних конфліктів у реальному часі.

Об'єктом дослідження є процес формування навчального розкладу та адміністрування освітнього навантаження у закладах вищої освіти.

Предметом дослідження виступають хмарні архітектурні рішення, безсерверні технології асинхронної обробки даних, моделі та алгоритми

виявлення просторово-часових конфліктів, а також програмні засоби їх реалізації у вебінтерфейсі системи розкладу.

Задля досягнення визначеної мети у роботі розв'язано такі завдання:

1. проаналізовано предметну область управління навчальним розкладом у закладах освіти;
2. досліджено існуючі інформаційні системи для формування навчального розкладу;
3. сформовано функціональні та нефункціональні вимоги до інформаційної системи;
4. обрано програмні засоби й інструменти для реалізації та розроблено архітектуру інформаційної системи з використанням хмарних технологій;
5. спроєктовано логічну та фізичну структуру бази даних системи;
6. розроблено механізм завантаження та автоматичної обробки навчального плану;
7. реалізовано інструмент створення, редагування та видалення елементів розкладу;
8. розроблено механізм перевірки конфліктів у розкладі (накладання аудиторій, зайнятість викладачів та академічних груп);
9. реалізовано інтерактивну можливість викладачам зазначати доступний час для проведення занять;
10. розроблено функціонал перегляду розкладу для різних категорій користувачів із можливістю додавання тем занять, посилань на матеріали та онлайн-заняття;
11. проведено тестування системи та здійснено аналіз результатів її роботи.

У межах роботи використано такі дослідницькі методи: аналіз та систематизація науково-технічних джерел, що стосується проблематики

автоматизації розкладу [1-4; 7; 8]; еволюційні, евристичні підходи та методи програмування в обмеженнях для формалізації задачі розподілу ресурсів і задоволення обмежень комбінаторних задач [5; 6; 14; 17; 20; 22; 25; 28]; принципи структурного та об'єктного проєктування програмного забезпечення для побудови архітектури розподілених вебсистем [13]; методи хмарних обчислень та сервіс-орієнтованого програмування для побудови безсерверної архітектури, розв’язання комбінаторних задач і організації реляційних баз даних з урахуванням концепцій ORM та технологій захисту доступу [12; 33; 35-39]; а також експериментальний аналіз логів і метрик для визначення чутливості та ефективності обчислювальних хмарних процесів у реальному часі.

Новизна результатів полягає у розробці гібридного евристично-еволюційного алгоритму створення розкладу, що поєднує детерміновану ініціалізацію для задоволення жорстких обмежень та генетичний пошук для оптимізації м'яких критеріїв, а також у прикладній адаптації безсерверного (Serverless) обчислювального підходу для асинхронного імпорту та структурування вхідних академічних даних.

Практичне значення отриманих результатів полягає у створенні веборієнтованого модуля автоматизованого формування розкладу ЗВО. Розроблене програмне рішення дозволяє оперативно завантажувати й обробляти навчальне навантаження в хмарі без надмірного використання локальних серверних ресурсів, збирати часові побажання викладачів через інтерактивну матрицю, автоматично виявляти просторово-часові колізії, а також виконувати гнучке ручне коригування сітки занять у реальному часі.

РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Актуальність проблеми автоматизованого формування навчального розкладу у закладах вищої освіти

Стрімкий розвиток інформаційних технологій та цифровізація освітнього середовища сприяють активному впровадженню автоматизованих систем управління навчальним процесом у закладах вищої освіти. Одним із найбільш складних та ресурсомістких процесів організації навчальної діяльності є формування навчального розкладу, оскільки він безпосередньо впливає на узгодження роботи викладачів, академічних груп та організацію освітнього процесу загалом [1-3].

Процес складання розкладу включає значну кількість взаємопов’язаних параметрів: навчальні плани, розподіл дисциплін між семестрами, навантаження викладачів, поділ студентів на групи, часові обмеження та індивідуальні побажання учасників освітнього процесу. Ручне формування розкладу потребує значних часових витрат і супроводжується високою ймовірністю виникнення конфліктів між заняттями [2].

Особливу складність становить необхідність одночасного врахування жорстких та нежорстких умов формування розкладу. До жорстких умов належать неприпустимість одночасного проведення занять для однієї групи або викладача в один часовий проміжок. Нежорсткі умови стосуються рівномірності навантаження, відсутності «вікон» та врахування побажань викладачів. Жорсткі умови повинні виконуватись завжди, оскільки їх порушення призводить до некоректності сформованого розкладу [5].

У сучасних закладах освіти використовуються різні інформаційні системи для автоматизації роботи з навчальним розкладом. Такі системи забезпечують централізоване зберігання даних, електронний доступ до розкладу та підтримку змін у навчальному процесі [3]. Водночас значна частина існуючих рішень

орієнтована на універсальні моделі планування та має обмежені можливості щодо врахування специфіки організації навчального процесу у закладах вищої освіти.

У наукових працях зазначається, що сучасні системи формування розкладу повинні підтримувати роботу з різними типами занять, враховувати часові побажання викладачів, забезпечувати централізований доступ до інформації через вебтехнології та підтримувати автоматичне оновлення даних [3; 4]. Окремою проблемою залишається підтримка потокових лекцій і розподілених практичних занять, коли різні академічні групи можуть працювати з різними викладачами [3].

Додаткові труднощі виникають під час підтримки календарної структури навчального процесу, зокрема роботи з парними та непарними тижнями, автоматичного врахування змін навчального навантаження та синхронізації оновлень між різними категоріями користувачів. У більшості випадків коригування таких змін потребує ручного втручання адміністратора системи [2]. Аналіз наукових публікацій та існуючих програмних рішень показав, що значна частина систем не забезпечує достатньої гнучкості під час роботи з індивідуальними часовими побажаннями викладачів, розподілом практичних занять між кількома викладачами та автоматизованою обробкою навчальних планів [3; 4]. Крім того, багато систем мають складну структуру впровадження або потребують додаткових ресурсів для підтримки та супроводу [4].

Таким чином, актуальність теми зумовлена потребою у веб-модулі автоматизованого формування навчального розкладу у закладах вищої освіти, який забезпечує централізоване керування навчальним процесом, підтримує роботу з потоковими лекціями та окремими практичними заняттями, враховує часові побажання викладачів, підтримує календарну модель парних і непарних тижнів та забезпечує зручний доступ до розкладу через вебінтерфейс.

1.2 Дослідження процесу формування навчального розкладу та аналіз існуючих підходів

Формування навчального розкладу є одним із ключових процесів організації освітньої діяльності закладу вищої освіти. Від правильності побудови розкладу залежить ефективність використання навчальних аудиторій, узгодженість роботи викладачів, академічних груп та адміністративного персоналу, а також якість організації навчального процесу загалом [2; 3].

Процес формування розкладу базується на даних навчальних планів, які визначають перелік дисциплін, кількість годин на їх вивчення, види навчальних занять та їх розподіл між семестрами. Для кожної дисципліни необхідно визначити викладача, академічну групу, тип заняття та часовий проміжок проведення. У результаті формується система взаємопов’язаних елементів, що потребує узгодження великої кількості параметрів [2].

Особливістю організації навчального процесу у закладах вищої освіти є наявність різних типів навчальних занять: лекцій, практичних, лабораторних та семінарських робіт. При цьому лекційні заняття часто проводяться для кількох академічних груп одночасно у вигляді потоків, тоді як практичні та лабораторні заняття можуть проводитися окремо для кожної групи або підгрупи різними викладачами [3]. Така структура ускладнює процес планування та вимагає додаткових механізмів контролю конфліктів під час формування розкладу.

Одним із головних завдань під час складання розкладу є забезпечення виконання встановлених обмежень. У наукових дослідженнях виділяють жорсткі та нежорсткі обмеження [5]. До жорстких обмежень належать неможливість проведення двох занять одночасно для однієї академічної групи, викладача або навчальної аудиторії. Порушення таких умов робить розклад непридатним до використання. До нежорстких обмежень відносять рівномірний розподіл навчального навантаження протягом тижня, мінімізацію перерв між

заняттями, врахування побажань викладачів щодо часу проведення занять та інші організаційні фактори [2; 5].

Додатковою особливістю є необхідність врахування календарної структури навчального процесу. У багатьох закладах вищої освіти використовується система парних і непарних тижнів, за якої окремі заняття проводяться через тиждень. Це збільшує кількість варіантів розміщення дисциплін у розкладі та ускладнює контроль його коректності [4].

Складність формування розкладу також пов’язана з необхідністю врахування індивідуального навантаження викладачів. Один викладач може забезпечувати викладання декількох дисциплін для різних академічних груп, що створює додаткові часові обмеження під час планування занять. Крім того, у процесі роботи виникає потреба враховувати побажання викладачів щодо доступних днів або часових проміжків проведення занять [3; 4].

У сучасних умовах важливого значення набуває централізоване зберігання інформації про навчальний процес та забезпечення оперативного доступу до розкладу через вебтехнології. Використання електронних систем дозволяє спростити внесення змін до розкладу, автоматизувати пошук конфліктів та забезпечити швидке оновлення інформації для всіх учасників освітнього процесу [3].

Таким чином, процес формування навчального розкладу у закладах вищої освіти характеризується великою кількістю взаємозалежних параметрів, необхідністю дотримання жорстких і нежорстких обмежень, підтримкою різних типів навчальних занять, потокових лекцій та календарної структури навчального процесу. Сукупність зазначених особливостей визначає потребу в застосуванні автоматизованих інформаційних систем для ефективного формування та супроводу навчального розкладу.

1.3 Аналіз існуючих підходів та засобів автоматизованого формування навчального розкладу

Процес автоматизованого формування навчального розкладу належить до класу складних комбінаторних задач розподілу ресурсів, оскільки потребує одночасного врахування великої кількості взаємопов'язаних обмежень та критеріїв якості розкладу [3; 5; 6]. До таких обмежень належать доступність викладачів, аудиторій, академічних груп, навчальних дисциплін, а також часові та організаційні особливості освітнього процесу. У зв'язку з цим у науковій літературі запропоновано різні підходи до автоматизованого формування розкладів, що відрізняються принципами пошуку розв'язку, обчислювальною складністю та ефективністю застосування [3; 6].

Одним із поширених підходів є метод імітації відпалу (Simulated Annealing), який базується на моделюванні процесу охолодження металу та пошуку глобального мінімуму енергетичного стану системи. У задачі складання розкладу роль енергетичної функції виконує функція штрафів за порушення обмежень. Основною перевагою методу є можливість виходу з локальних мінімумів та отримання якісних розв'язків для складних задач. Водночас ефективність алгоритму значною мірою залежить від вибору параметрів охолодження, а час пошуку може бути досить значним [3; 6; 14-16].

Іншим підходом є метод розфарбування графів, у якому навчальні заняття подаються вершинами графа, а конфлікти між ними — ребрами. Розв'язок задачі полягає у призначенні кожній вершині кольору, що відповідає часовому інтервалу проведення заняття. Такий підхід дозволяє наочно моделювати конфлікти розкладу, проте задача пошуку допустимого розфарбування графа належить до NP-повних задач, що обмежує його використання для великих наборів даних [3; 6; 17-19].

Для моделювання процесу складання розкладу також застосовується імітаційне моделювання, яке відтворює дії диспетчера або адміністратора під

час розподілу занять. У межах цього підходу заняття послідовно додаються до розкладу з урахуванням доступних ресурсів та встановлених обмежень. Перевагою методу є можливість врахування специфічних особливостей конкретного закладу освіти, однак значна кількість ітерацій та відсутність гарантії отримання безконфліктного результату знижують його ефективність [3; 6; 20].

Значного поширення набув підхід програмування в обмеженнях (Constraint Programming). Його сутність полягає у формальному описі змінних і набору обмежень, після чого система автоматично виконує пошук допустимих розв’язків. Перевагою цього підходу є ефективне скорочення простору пошуку за рахунок автоматичного виключення варіантів, що не відповідають встановленим правилам. Завдяки можливості роботи з великою кількістю жорстких і м’яких обмежень програмування в обмеженнях широко використовується у сучасних системах автоматизованого складання розкладів [3; 6; 21-23].

До евристичних методів належить також алгоритм мурашиної колонії, принцип роботи якого базується на моделюванні поведінки мурах під час пошуку найкоротшого шляху. У процесі формування розкладу алгоритм поступово накопичує інформацію про найбільш успішні варіанти розміщення занять та використовує її для побудови нових розв’язків. Такий підхід характеризується високою адаптивністю та можливістю роботи з різними типами обмежень, проте потребує значних обчислювальних ресурсів та ретельного налаштування параметрів [3; 6; 24-27].

Одним із найбільш досліджених підходів є генетичні алгоритми, що ґрунтуються на принципах еволюційного розвитку та природного добору. Кожний можливий розклад розглядається як окрема особина популяції, а пошук раціонального рішення здійснюється шляхом виконання операцій селекції, схрещування та мутації. Генетичні алгоритми демонструють високу

ефективність при розв’язанні складних комбінаторних задач та дозволяють уникати застрягання алгоритму на недосконалих проміжних варіантах. Водночас їх реалізація потребує ретельного проєктування структури розкладу, функції пристосованості та параметрів еволюційного процесу [3; 5; 6; 28-32].

Різноманітність існуючих методів зумовлює необхідність їх систематизації за ключовими характеристиками: ефективністю пошуку розв’язку, здатністю враховувати складні обмеження та обчислювальною складністю. Для об’єктивної оцінки доцільності застосування кожного з підходів у межах розроблюваної системи було проведено порівняльний аналіз, результати якого наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння підходів до автоматизованого формування навчального розкладу

Підхід	**Переваги**	**Недоліки**
Імітація відпалу (Simulated Annealing)	Може виходити з локальних мінімумів, формує якісні рішення	Висока залежність від параметрів охолодження, тривалий час пошуку
Розфарбування графів	Просте представлення конфліктів, наочність моделі	Низька масштабованість для великих задач
Імітаційне моделювання	Легко адаптується до специфіки ЗВО	Не гарантує повністю безконфліктного результату
Constraint Programming	Ефективна робота з великою кількістю обмежень	Висока складність формалізації моделі
Ant Colony Optimization	Добре працює в умовах складних обмежень	Значні обчислювальні витрати
Genetic Algorithm	Висока гнучкість, можливість пошуку найбільш раціональних рішень	Потребує налаштування функції пристосованості та операторів еволюції

Як видно з таблиці 1.1, найбільш перспективними для задач автоматизованого формування навчального розкладу є методи програмування в обмеженнях [21; 22] та генетичні алгоритми [28-30]. Вони забезпечують достатню гнучкість під час врахування великої кількості жорстких і нежорстких обмежень, а також дозволяють працювати зі складними структурами навчального процесу.

Проведений аналіз показав, що існуючі підходи дозволяють ефективно розв’язувати задачу автоматизованого формування навчального розкладу, проте більшість із них або потребують значних обчислювальних ресурсів, або недостатньо враховують специфіку організації освітнього процесу у закладах вищої освіти [6; 24]. Особливо актуальними залишаються питання підтримки потокових лекцій, розподілу практичних занять між викладачами, врахування побажань викладачів та автоматичного формування розкладу для парних і непарних тижнів [4; 8].

З огляду на виявлені особливості предметної області, у подальшій роботі буде реалізовано веборієнтовану інформаційну систему автоматизованого формування розкладу навчального процесу, в основу якої покладено гібридний алгоритм, що поєднує жадібну евристику для швидкого побудування базового розкладу та генетичний підхід для його подальшого вдосконалення та покращення якості за визначеними критеріями [31; 32].

1.4 Порівняльний аналіз сучасних програмних рішень і сервісів для планування навчального процесу

Автоматизація планування навчального розкладу є одним із найскладніших завдань у межах цифровізації сучасної інфраструктури закладів вищої та фахової передвищої освіти [2]. Формування розкладу потребує врахування значної кількості взаємопов’язаних параметрів: навчальних планів,

академічних груп, викладачів, календарних обмежень, розподілу дисциплін між семестрами та часових побажань учасників освітнього процесу [1, 4].

Сучасні програмні рішення для формування навчального розкладу реалізуються у вигляді веб-орієнтованих сервісів або локальних програмних комплексів. Більшість таких систем підтримує базові функції адміністрування навчального процесу: створення та редагування розкладу, роботу з викладачами та групами, календарне планування, внесення змін, експорт даних та публікацію розкладу через вебінтерфейс [2].

Серед найбільш поширених систем, що використовуються для планування навчального процесу, можна виділити "Розклад Нова Школа", ASC Timetables та OpenEduCat Timetable Management System [9-11]. Дані рішення відрізняються за архітектурою, функціональністю, рівнем автоматизації та підходами до організації обробки даних, однак усі вони орієнтовані на централізоване керування навчальним розкладом.

На рисунках 1.1–1.3 наведено інтерфейси досліджуваних програмних рішень. Аналіз структури їх інтерфейсів дозволяє визначити поширені підходи до організації навігації, відображення календарної сітки, роботи з навчальними групами та адміністрування розкладу, що може бути використано під час проєктування власного вебінтерфейсу системи.

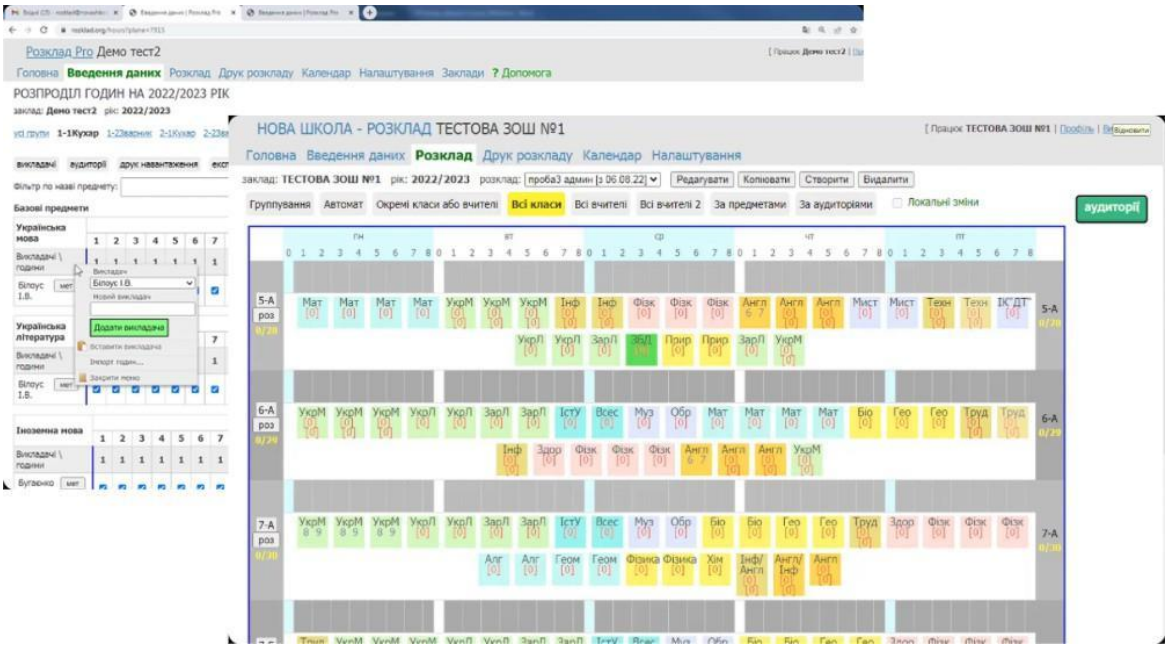


Рисунок 1.1 – Інтерфейс системи "Розклад Нова Школа" [9]

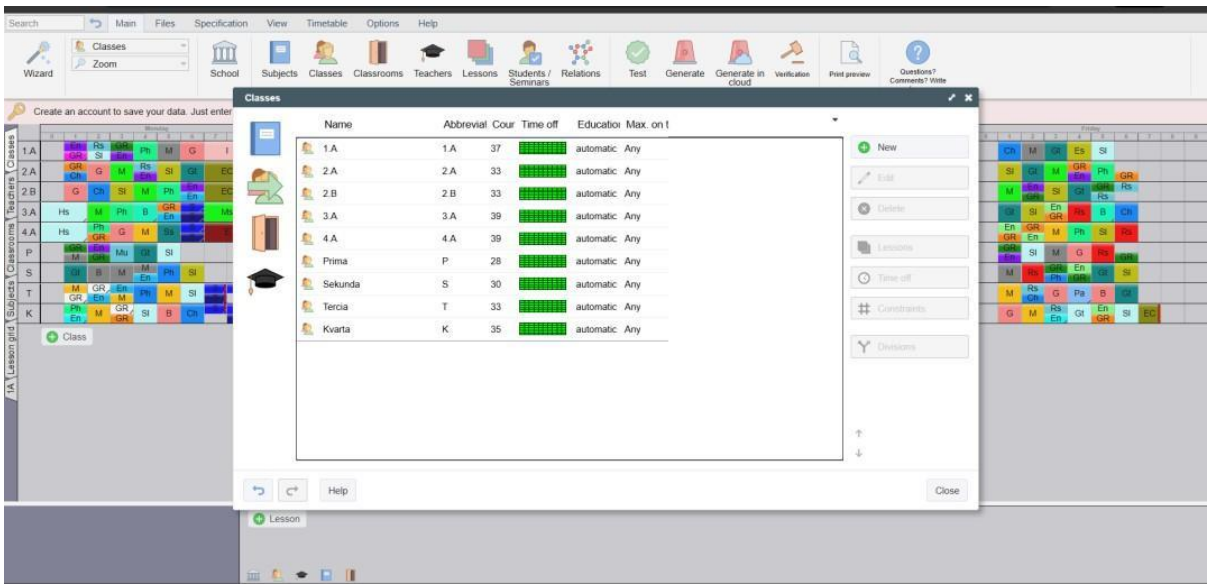


Рисунок 1.2 – Інтерфейс системи ASC Timetables [10]

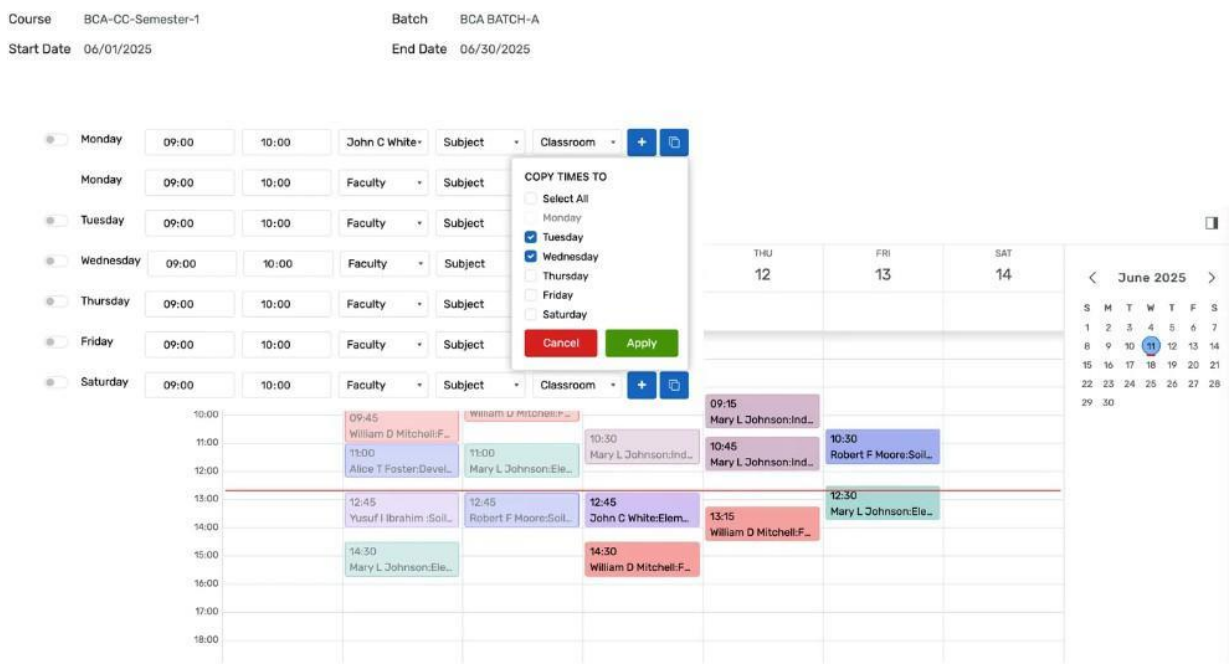


Рисунок 1.3 – Інтерфейс системи OpenEduCat Timetable Management System [11]

Критерії порівняльного аналізу інструментів. Для забезпечення об’єктивного та системного порівняння програмних рішень було визначено критерії, які безпосередньо впливають на процес формування та підтримки навчального розкладу у закладах вищої освіти. До основних критеріїв належать:

- підтримка роботи з викладачами та академічними групами;
- можливість автоматизованого формування розкладу;
- підтримка потокових лекцій та окремих практичних занять;
- підтримка календарної моделі парних і непарних тижнів;
- можливість автоматизованого завантаження навчальних планів;
- підтримка веб-інтерфейсу та хмарних технологій;
- гнучкість внесення змін до розкладу;
- особливості архітектури системи та організації обробки даних.

Аналіз системи "Розклад Нова Школа". "Розклад Нова Школа" є веборієнтованою системою для формування та перегляду навчального розкладу

[9]. Основною перевагою сервісу є централізований доступ до розкладу через браузер, простий інтерфейс користувача та підтримка базових механізмів адміністрування навчального процесу.

Система підтримує роботу з викладачами, навчальними групами, календарними періодами та забезпечує публікацію розкладу через вебінтерфейс. Також реалізовано підтримку парних і непарних тижнів та експорт сформованого розкладу.

Разом із цим система має низку обмежень. Формування навчального навантаження значною мірою виконується вручну, а автоматизована обробка навчальних планів відсутня. Система не підтримує гнучкий механізм розподілу практичних занять між різними викладачами та має обмежені можливості інтеграції з сучасними хмарними сервісами [9].

Аналіз системи ASC Timetables. ASC Timetables є одним із найбільш відомих програмних рішень для автоматизованого формування навчального розкладу [10]. Система реалізована у вигляді десктопного програмного забезпечення з підтримкою вебпублікації розкладу та онлайн-доступу через хмарні сервіси. ASC Timetables підтримує велику кількість параметрів та правил формування розкладу, включаючи часові обмеження, різні типи занять та календарні періоди.

До переваг системи належать:

- автоматизоване формування розкладу;
- підтримка складних правил планування;
- засоби адміністрування;
- підтримка вебпублікації;
- можливість внесення змін та перенесення занять.

Недоліком системи є складність налаштування та значна залежність від локального програмного середовища. Для повноцінної роботи необхідне встановлення окремого програмного забезпечення, а частина функціональності

доступна лише у платних версіях. Крім того, система орієнтована переважно на класичну модель серверної обробки даних та не підтримує сучасні підходи до асинхронної хмарної обробки навчальних планів [10].

Аналіз системи OpenEduCat Timetable Management System. OpenEduCat Timetable Management System є модулем ERP-платформи OpenEduCat для керування освітнім процесом через вебінтерфейс [11]. Система підтримує роботу з дисциплінами, викладачами, академічними групами та календарними подіями.

Основними перевагами системи є:

- інтеграція з іншими модулями освітнього середовища;
- підтримка централізованого зберігання даних;
- можливість хмарного розгортання;
- веборієнтований інтерфейс.

Водночас система орієнтована на використання в межах великої ERP-інфраструктури, що ускладнює її використання як окремого модуля формування розкладу. Для роботи системи необхідне розгортання значної кількості супутніх компонентів, а імпорт навчальних планів потребує попередньої підготовки даних [11].

Отже, існуючі програмні рішення охоплюють основні потреби користувачів щодо перегляду, адміністрування та формування навчального розкладу. Водночас вони мають обмеження у підтримці автоматизованої обробки навчальних планів, гнучкому розподілі практичних занять між викладачами та використанні сучасних хмарних технологій.

Порівняльний аналіз функціональних можливостей та обмежень існуючих систем. Для узагальнення результатів аналізу було сформовано таблицю 1.2.

Таблиця 1.2 – Порівняльна характеристика функціональних можливостей та обмежень існуючих систем

Критерій	**"Розклад Нова Школа"**	**ASC Timetables**	**OpenEduCat**
Веб-інтерфейс для роботи з розкладом	Реалізовано	Реалізовано	Реалізовано
Робота з викладачами та групами	Реалізовано	Реалізовано	Реалізовано
Автоматичне формування розкладу	Відсутнє	Реалізовано	Частково реалізовано
Підтримка потокових лекцій	Частково реалізовано	Реалізовано	Частково реалізовано
Підтримка розподілених практичних занять	Відсутнє	Частково реалізовано	Частково реалізовано
Врахування часових побажань викладачів	Відсутнє	Реалізовано	Частково реалізовано
Підтримка парних та непарних тижнів	Реалізовано	Реалізовано	Частково реалізовано
Автоматичне завантаження навчального плану	Відсутнє	Відсутнє	Частково реалізовано
Хмарна підтримка	Частково реалізовано	Частково реалізовано	Реалізовано
Асинхронна обробка даних	Відсутнє	Відсутнє	Частково реалізовано

Наведене порівняння показує, що існуючі системи забезпечують базові механізми формування та відображення навчального розкладу, однак мають обмеження у реалізації автоматизованої обробки навчального навантаження, підтримці гнучкої взаємодії між викладачами та академічними групами, а також

у використанні сучасних хмарних підходів до обробки даних. Для усунення цих недоліків у новій системі буде реалізовано гібридний алгоритм генерації розкладу, здатний балансувати жорсткі академічні обмеження та м'які побажання викладачів. Крім того, система отримає зручні інструменти для ручного коригування занять у реальному часі. Технічною основою проєкту стане сучасна безсерверна (Serverless) архітектура, яка забезпечить швидкий та асинхронний імпорт навчальних планів без створення додаткового навантаження на сервери. Це дає можливість сформувати функціональні та нефункціональні вимоги до розроблюваного веб-модуля.

Окремої уваги потребує використання хмарних технологій під час роботи з навчальними планами та формування розкладу. У більшості існуючих систем основна логіка виконується на постійному серверному середовищі, що потребує підтримки серверної інфраструктури та ускладнює масштабування системи.

Перспективним підходом у сучасних вебсистемах є використання хмарних сервісів для асинхронної обробки даних. Зокрема, подієво-орієнтовані механізми обробки файлів дозволяють автоматизувати завантаження та обробку навчальних планів без необхідності постійного навантаження на центральний сервер [12]. Використання хмарних функцій та об’єктних сховищ забезпечує гнучкість, масштабованість та спрощення взаємодії між компонентами системи.

Такий підхід дозволяє зменшити навантаження на серверну інфраструктуру, забезпечити асинхронну обробку даних та спростити інтеграцію вебінтерфейсу із сервісами обробки навчальних планів.

Отже, існуючі рішення залишаються ефективними для базового адміністрування та перегляду навчального розкладу, проте мають обмеження щодо автоматизованої обробки навчальних планів, використання сучасних хмарних підходів та гнучкого розподілу навчального навантаження між викладачами й академічними групами. Наявний функціональний розрив у сфері безсерверної обробки даних та автоматизації імпорту планів свідчить про

необхідність створення спеціалізованого модуля, орієнтованого на високу масштабованість. Запропоноване у цій роботі програмне рішення дозволить усунути зазначені недоліки шляхом інтеграції хмарних обчислювальних функцій AWS та реалізації ефективного конвеєра асинхронної обробки інформації, що забезпечить підвищення автономності системи при роботі з великими масивами даних. Саме це є підставою для переходу до формування функціональної моделі системи та подальшої програмної реалізації веб-модуля автоматизованого формування навчального розкладу.

1.5 Функціональні та нефункціональні вимоги до інформаційної системи

На основі аналізу предметної області, дослідження існуючих систем автоматизації навчального розкладу та загальних підходів до проєктування програмного забезпечення сформовано функціональні та нефункціональні вимоги до інформаційної системи [2, 3, 13].

Функціональні вимоги визначають перелік можливостей системи та сценарії взаємодії користувачів із програмним забезпеченням. Вони описують основні процеси, пов’язані з керуванням користувачами, структурою навчального закладу, обробкою навчальних планів, формуванням розкладу та забезпеченням його перегляду різними категоріями користувачів [13]. Сформовані функціональні вимоги наведені у таблиці 1.3.

Таблиця 1.3 – Функціональні вимоги до інформаційної системи

Код вимоги	Назва вимоги	Опис функціональної вимоги
FR-001	Керування обліковими записами	Створення адміністраторів із можливістю гнучкого налаштування індивідуальних прав доступу.

Продовження таблиці 1.3

FR-002	Хешування автентифікаційних даних	Безпечне збереження паролів користувачів у системі за допомогою алгоритму BCrypt.
FR-003	Рольова модель доступу	Розмежування доступу до інтерфейсу та API відповідно до ролей користувачів (ADMIN, TEACHER, STUDENT, GUEST).
FR-004	Адміністрування структури ЗВО	Створення, редагування та видалення елементів структури: університетів, факультетів, кафедр, програм та груп.
FR-005	Профілювання учасників процесу	Закріплення студентів за академічними групами та викладачів за кафедрами для коректної побудови зв’язків.
FR-006	Завантаження навчальних планів	Безпечне завантаження файлів навчальних планів через генерацію Presigned URL до хмарного сховища AWS S3.
FR-007	Асинхронна обробка навчальних планів	Обробка завантажених навчальних планів та формування проміжного результату зіставлення даних викладачів і кафедр.
FR-008	Конструювання часової сітки	Створення базових шаблонів часових слотів (каркас розкладу) для подальшого планування занять.
FR-009	Керування відображенням сітки	Можливість глобального приховування або відображення базової часової сітки в інтерфейсі користувачів.
FR-010	Адресне очищення розкладу	Повне очищення або оновлення розкладу для конкретної академічної групи без впливу на інші дані.
FR-011	Автоматична генерація розкладу	Формування розкладу алгоритмом із урахуванням потокових лекцій, розподілу практик і обмежень викладачів.

Продовження таблиці 1.3

FR-012	Підтримка потокових лекцій	Планування спільних лекцій для кількох академічних груп одночасно.
FR-013	Розподіл практичних занять	Формування практичних занять окремо для кожної академічної групи з урахуванням викладачів.
FR-014	Моніторинг часових конфліктів	Автоматичне виявлення накладань занять для викладачів і академічних груп.
FR-015	Керування уподобанням и викладачів	Збереження та використання індивідуальних часових уподобань викладачів під час формування розкладу.
FR-016	Наповнення контенту занять	Додавання тем занять, посилань на онлайн-зустрічі та навчальних матеріалів до розкладу.

Наведені у таблиці 1.3 функціональні вимоги визначають ключові вектори взаємодії користувачів із програмним продуктом. Для візуалізації моделі доступу та чіткого розмежування прав між різними категоріями користувачів (ADMIN, TEACHER, STUDENT), було побудовано UML-діаграму прецедентів (Use Case Diagram), яку наведено на рисунку 1.4.

Дана діаграма наочно відображає ієрархію функціональних можливостей системи: від завдань адміністративного керування та автоматизації імпорту до специфічних функцій викладача та загального доступу до розкладу. Такий підхід до моделювання дозволяє не лише підтвердити повноту функціональних вимог, але й закласти основу для подальшого проектування модульної архітектури та структури бази даних системи [13].



Рисунок 1.4 – UML-діаграма прецедентів інформаційної системи

Нефункціональні вимоги характеризують якісні властивості системи, зокрема продуктивність, безпеку, надійність, масштабованість та особливості експлуатації програмного забезпечення. Вони визначають обмеження та критерії якості, яких повинна дотримуватися система під час роботи в реальних умовах [13]. Сформовані нефункціональні вимоги наведені у таблиці 1.4.

Таблиця 1.4 – Нефункціональні вимоги до інформаційної системи

Код вимоги	**Назва вимоги**	**Опис нефункціональної вимоги**
NFR-001	Швидкість формування розкладу	Час формування та балансування розкладу для факультету не повинен перевищувати 10 секунд.
NFR-002	Час обробки навчального плану	Час очікування відповіді асинхронного процесу обробки навчального плану через AWS Lambda обмежений 50 секундами.
NFR-003	Цілісність даних	Система повинна забезпечувати цілісність даних під час масових операцій оновлення або очищення розкладу.
NFR-004	Захист облікових записів	Паролі користувачів повинні зберігатися у зашифрованому вигляді з використанням BCrypt.
NFR-005	Контроль доступу	Операції видалення даних доступні лише користувачам із відповідним рівнем привілеїв (canDelete).
NFR-006	Безпечний доступ до файлів	Доступ до хмарного сховища AWS S3 здійснюється виключно через тимчасові підписані посилання (Presigned URL).
NFR-007	Кросбраузерна сумісність	Вебінтерфейс системи повинен коректно працювати у сучасних браузерах без втрати функціональності та стилізації.
NFR-008	Інтерактивність інтерфейсу	Переміщення по календарю (перехід між тижнями та повернення до поточної дати) повинно виконуватися без перезавантаження сторінки.

Таким чином, визначений комплекс функціональних та нефункціональних вимог слугує фундаментальною основою для подальшої розробки інформаційної системи. Якщо функціональні вимоги визначають, які саме завдання та процеси виконує система для користувача, то нефункціональні —

встановлюють технічні обмеження та критерії якості її функціонування. Наведені показники будуть базою для проектування архітектури, формування схеми бази даних, вибору технологічного стеку, а також для побудови тестових сценаріїв.

.

РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Обґрунтування вибору технологій та інструментів розробки

Розробка сучасної інформаційної системи автоматизованого формування розкладу вимагає ретельного підходу до вибору технологічного стека та архітектурних рішень, що відповідають сучасним стандартам інженерії програмного забезпечення [13]. Архітектура системи повинна забезпечувати високу продуктивність під час виконання ресурсоємних обчислень, гарантувати надійне зберігання складної ієрархічної структури даних, а також підтримувати можливість гнучкого масштабування.

З огляду на ці вимоги, технологічний стек було поділено на три основні сегменти: серверна частина (Backend), хмарна інфраструктура на базі подієво-орієнтованої архітектури (Event-Driven Architecture) [12] та система управління базами даних (СУБД).

Обґрунтування вибору технологій серверної частини (Backend). Основним інструментом для реалізації серверної логіки було обрано мову програмування Java (версія 17) та фреймворк Spring Boot 3. Вибір екосистеми Java обґрунтовується такими чинниками:

- Оскільки ядром системи є гібридний еволюційний алгоритм автоматичного складання розкладу [5], використання строго типізованої мови, що компілюється у байт-код (Java), забезпечує значну перевагу у швидкості виконання алгоритмічних розрахунків порівняно з інтерпретованими мовами.
- Екосистема Spring Boot дозволяє реалізувати надійну серверну архітектуру з мінімальними витратами часу на базову конфігурацію. Використання Spring Security (із застосуванням BCryptPasswordEncoder) забезпечує захист даних та реалізацію

рольової моделі керування доступом (RBAC) до системи управління навчальним процесом [33].

- Технологія Spring Data JPA у зв'язці з фреймворком Hibernate автоматизує процеси взаємодії з базою даних, трансляцію абстрактних типів даних застосунку в специфічні типи СУБД та обробку складних ієрархічних зв'язків між сутностями [37].

Вибір системи управління базами даних. Враховуючи необхідність зберігання складної реляційної структури даних предметної області (університети, факультети, кафедри, викладачі, студенти, розклад, часові слоти), для проєкту було обрано реляційну СУБД MySQL версії 8.0, що відповідає фундаментальним теоріям проєктування баз даних [35, 36].

До основних переваг MySQL у контексті даного проєкту належать:

- Відмінна підтримка транзакційності (ACID), що критично важливо під час масового пакетного генерування та збереження сотень записів розкладу в межах однієї операції.
- Висока швидкість обробки складних запитів із використанням JOIN-операцій при отриманні розкладу конкретного факультету, викладача чи студентської групи.
- Для забезпечення ізольованості середовища базу даних було розгорнуто за допомогою технології контейнеризації Docker. Це унеможливлює конфлікти версій та створює умови для горизонтального масштабування.

Обґрунтування використання хмарних сервісів (AWS). Однією з ключових вимог до системи є можливість автоматизованого завантаження та парсингу масивів даних навчальних планів. З метою уникнення перевантаження основного сервера та реалізації концепції обробки даних у реальному часі (Real-Time Data Processing) [12], цю задачу було винесено у хмарне середовище Amazon Web Services (AWS).

Для реалізації безсерверної (Serverless) архітектури використано такі сервіси:

- Amazon S3: Застосовується для збереження завантажених навчальних планів. Завантаження відбувається напряму від клієнта у хмару за допомогою механізму Presigned URLs, що повністю знімає навантаження з основного Spring Boot сервера.
- AWS Lambda: Безсерверна обчислювальна служба, що автоматично запускається у відповідь на подію (завантаження нового файлу в S3). Скрипт, написаний мовою Python, витягує файл із хмарного сховища, парсить його структуру, очищає дані та надсилає їх у вигляді JSON на основний REST API сервера.

Для забезпечення безпечного тунелювання запитів від AWS Lambda до локально розгорнутого сервера під час інтеграції використано інструмент ngrok.

Вибір технологій клієнтської частини. Клієнтська частина системи реалізована за допомогою базових вебтехнологій: HTML5, CSS3 та Vanilla JavaScript. Такий вибір зумовлений метою створення легкого, інтуїтивно зрозумілого та швидкого клієнт-розкладу для кінцевих користувачів закладів вищої освіти [4]. Використання чистого JavaScript у поєднанні з Fetch API дозволяє здійснювати асинхронні запити до серверної частини та динамічно оновлювати візуалізацію сітки розкладу без перезавантаження сторінки.

Таким чином, обраний технологічний стек (Java Spring Boot, MySQL, Docker, AWS S3 + Lambda, Vanilla JS) формує надійну, масштабовану та високопродуктивну екосистему, яка повністю задовольняє функціональні вимоги до розроблюваної інформаційної системи.

2.2 Архітектура та взаємодія компонентів інформаційної системи

Проектування надійної, масштабованої та зручної в супроводі інформаційної системи вимагає чіткого розділення бізнес-логіки, інтерфейсу та

контуру збереження даних. Для реалізації системи автоматизованого формування розкладу було обрано гібридний архітектурний підхід, який поєднує класичну трирівневу архітектуру (3-Tier Architecture), описану в фундаментальних працях з інженерії програмного забезпечення [13], із сучасними хмарними безсерверними обчисленнями (Serverless Cloud-Native).

Логічний та фізичний розподіл компонентів реалізовано за патерном еталонної архітектури, що детально відображено на діаграмі архітектури інформаційної системи (рис. 2.1).

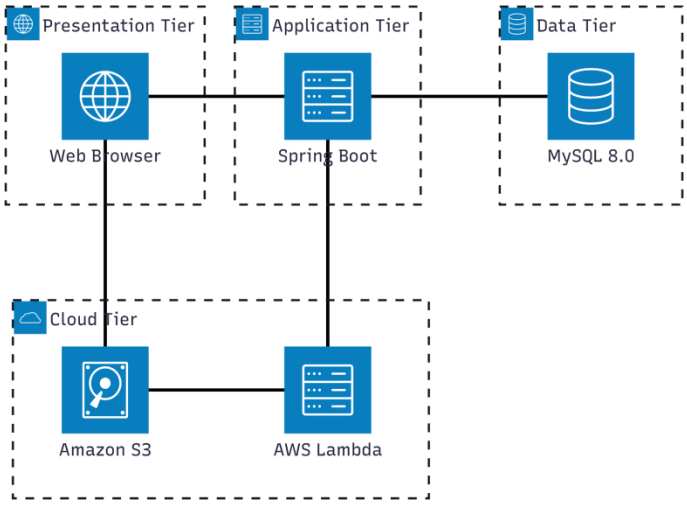


Рисунок 2.1 – Діаграма архітектури інформаційної системи

Система структурно поділена на такі ізольовані рівні:

- Рівень представлення даних (Presentation Tier): Клієнтський вебзастосунок, що виконується в середовищі браузера. Відповідає за рендеринг користувацького інтерфейсу та валідацію вводу.
- Рівень бізнес-логіки (Application Tier): Центральний бекенд-сервер на базі фреймворку Spring Boot. Реалізує основну алгоритмічну

логіку, захист програмних інтерфейсів (API) та управління транзакціями.

- Рівень зберігання даних (Data Access Tier): Реляційна база даних MySQL 8.0, розгорнута в ізольованому контейнері Docker, що забезпечує цілісне зберігання нормалізованої інформації відповідно до класичних принципів проєктування баз даних [35].
- Рівень хмарних обчислень (Cloud Computing Tier): Додатковий асинхронний шар (Amazon S3 та AWS Lambda). Цей контур забезпечує делегування ресурсомістких операцій на спеціалізовані потужності.

Для ілюстрації маршрутизації даних та покрокової послідовності обробки запитів між зазначеними рівнями розроблено схему взаємодії компонентів, яку наведено на рисунку 2.2.

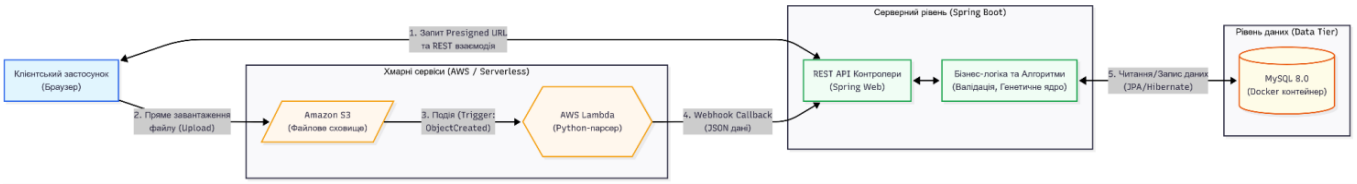


Рисунок 2.2 – Схема взаємодії компонентів під час обробки даних

Побудована модель відображає високорівневий алгоритм обробки вхідної інформації. Завдяки цій схемі чітко простежується перехід від синхронної взаємодії клієнта з сервером до прямого завантаження файлів у хмарне сховище, минаючи оперативну пам'ять локального сервера застосунків.

Для глибокого розуміння механізмів міжсервісної взаємодії під час виконання асинхронних операцій, побудованих за принципами подієво-орієнтованої архітектури (Event-Driven Architecture) [12], на рисунку 2.3 представлено деталізовану діаграму послідовності.

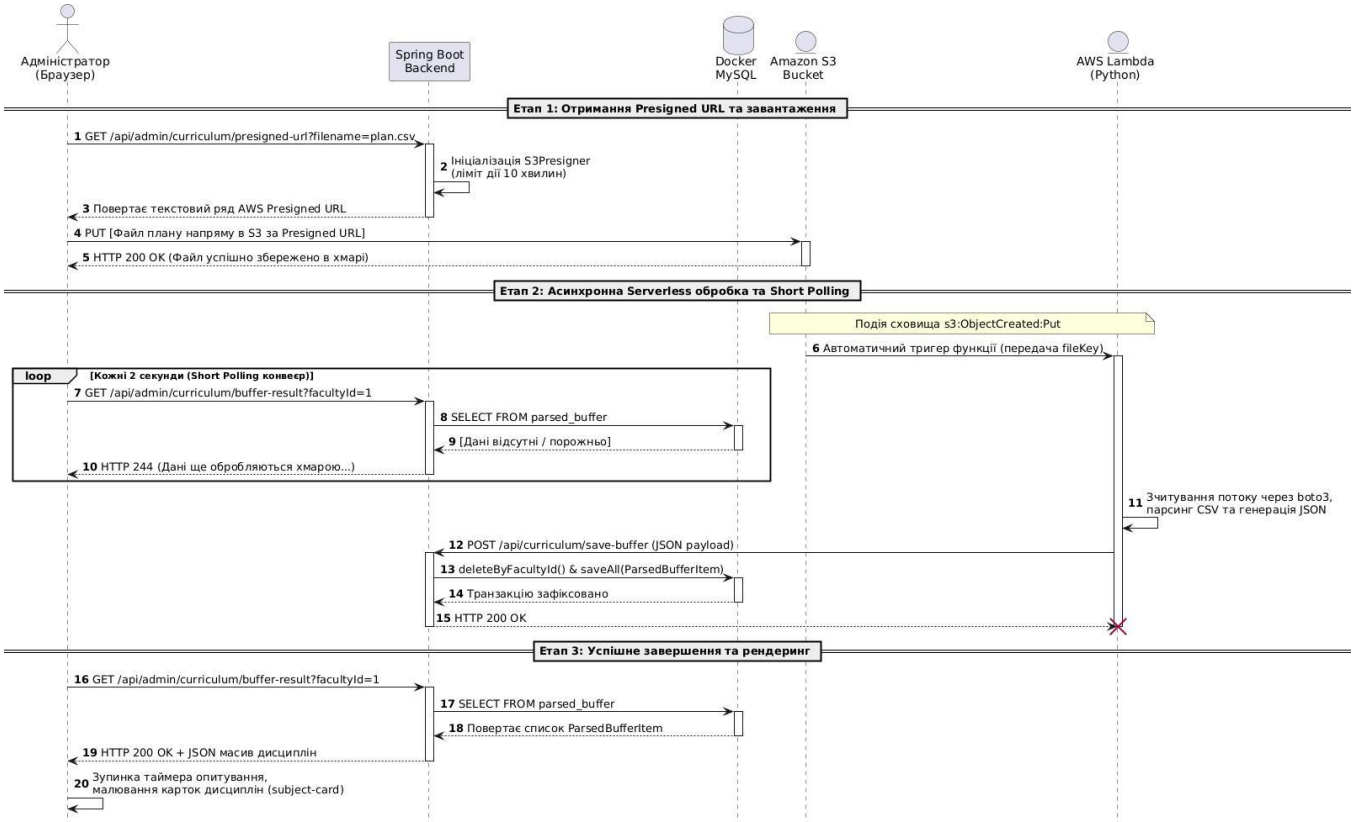


Рисунок 2.3 – Діаграма послідовності ітерації під час імпорту та обробки даних

Представлений алгоритм вирішує проблему блокування основного потоку сервера. Автоматичний тригер об'єктного сховища ініціалізує безсерверну функцію (AWS Lambda), яка після завершення нормалізації даних самостійно повертає результат до бази даних. До моменту завершення транзакції клієнтський інтерфейс використовує механізм періодичного опитування (Short Polling) для підтримки актуального стану системи без розірвання з'єднання.

Такий архітектурний дизайн гарантує високу відмовостійкість системи та унеможливлює перевантаження локального сервера під час масового імпорту навчальних планів.

2.3 Хмароорієнтована архітектура та інтеграція сервісів AWS

У розробленій системі автоматизованого формування навчального розкладу одним із важливих завдань є обробка електронних навчальних планів. Аналіз і перетворення таких даних потребують виконання ресурсоємних операцій, що можуть негативно впливати на швидкодію основного вебзастосунку. Тому під час проєктування сучасних систем автоматизації освітнього процесу доцільним є використання архітектурних підходів, які забезпечують масштабованість, незалежність компонентів та можливість асинхронної обробки даних [3; 13; 34].

Одним із таких підходів є подійно-орієнтована архітектура (Event-Driven Architecture, EDA), у межах якої взаємодія між компонентами системи здійснюється через події. Подія виникає внаслідок виконання певної дії в системі та виступає сигналом для запуску подальших процесів обробки. Використання EDA дозволяє розділити процес надходження даних і процес їх обробки, зменшити взаємозалежність між окремими компонентами системи та спростити масштабування програмного забезпечення [12; 13; 34].

Однією з основних переваг подійно-орієнтованого підходу є підтримка асинхронної обробки даних. На відміну від традиційної синхронної взаємодії, де користувач очікує завершення всіх операцій перед отриманням результату, подійна архітектура дозволяє виконувати складні обчислення окремо від основного процесу обслуговування запитів. Це особливо важливо для систем управління освітнім процесом, де обробка навчальних планів включає перевірку структури документа, аналіз навчального навантаження та підготовку даних для подальшого формування розкладу [12].

У сучасних хмарних системах подійно-орієнтований підхід часто поєднується з концепцією безсерверних обчислень (Serverless Computing). Безсерверна архітектура передбачає передачу завдань керування інфраструктурою, масштабуванням та розподілом ресурсів хмарному

провайдеру, що дозволяє розробнику зосередитися на реалізації прикладної логіки системи. Однією з найпоширеніших моделей такого підходу є Function-as-a-Service (FaaS), відповідно до якої програмний код виконується у вигляді окремих функцій у відповідь на виникнення подій. Поєднання принципів EDA та serverless забезпечує ефективне використання обчислювальних ресурсів, підвищує гнучкість системи та спрощує обробку великих обсягів даних [12; 13].

2.3.1 Загальна топологія хмарної інфраструктури та сервіси AWS

Для реалізації подійно-орієнтованої архітектури у розробленій системі використано хмарну платформу Amazon Web Services (AWS), яка надає набір сервісів для побудови масштабованих та безсерверних інформаційних систем. AWS підтримує механізми асинхронної взаємодії між компонентами та дозволяє реалізовувати обробку подій у режимі реального часу [12; 34].

У межах розробленої системи центральним компонентом хмарної інфраструктури виступає сервіс Amazon Simple Storage Service (Amazon S3), який використовується для зберігання навчальних планів та інших документів, що підлягають подальшій автоматизованій обробці.

Для виконання обчислювальних операцій використовується сервіс AWS Lambda, який реалізує модель Function-as-a-Service. Взаємодія між сховищем даних та модулем обробки здійснюється за допомогою механізму подій. Після завантаження файлу до сховища автоматично формується подія, яка ініціює виконання відповідної безсерверної функції.

Застосування такого підходу дозволяє розділити процеси зберігання та обробки даних, зменшити навантаження на основний сервер застосунку та забезпечити ефективне масштабування системи відповідно до поточних потреб.

Для деталізації внутрішньої логіки роботи описаної хмарної інфраструктури та покрокового процесу прийняття рішень під час завантаження

файлів розроблено відповідний алгоритм. Блок-схему алгоритму імпорту та обробки навчального плану представлено на рисунку 2.4.

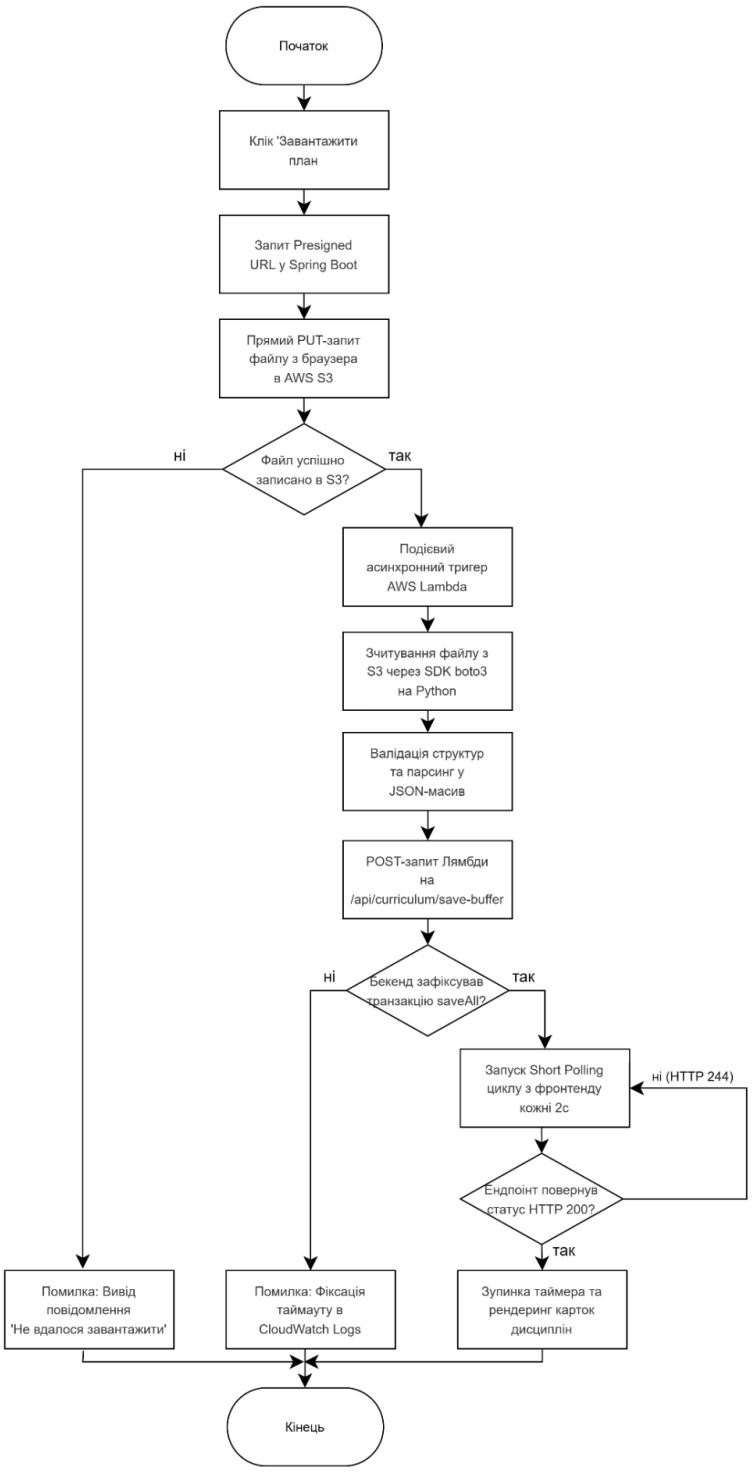


Рисунок 2.4 - Блок-схема алгоритму імпорту та обробки навчального плану

На блок-схемі відображено процес завантаження навчального плану до сховища Amazon S3, автоматичне ініціювання безсерверної функції AWS Lambda, обробку та валідацію даних, а також передачу результатів до основної інформаційної системи для подальшого формування навчального розкладу.

2.3.2 Патерн безпечного безсерверного завантаження файлів (AWS Presigned URL)

Важливою складовою хмарної інфраструктури є забезпечення безпечного доступу до ресурсів сховища. Для реалізації механізму завантаження файлів використовується підхід на основі тимчасових підписаних URL (Presigned URL).

Сутність даного механізму полягає у формуванні спеціального посилання, що містить криптографічний підпис та має обмежений термін дії. Завдяки цьому користувач отримує можливість виконати завантаження файлу без безпосереднього доступу до облікових даних хмарної платформи.

Процес завантаження передбачає формування сервером тимчасового посилання, яке передається клієнтському застосунку. Після отримання такого посилання файл завантажується безпосередньо до хмарного сховища, минаючи сервер прикладної логіки. Під час генерації підписаного посилання задається обмежений термін його дії, що забезпечує контрольований доступ до об'єкта протягом визначеного часу.

Використання даного підходу дозволяє підвищити рівень безпеки системи, зменшити навантаження на сервер та раціоналізувати використання обчислювальних ресурсів. Крім того, пряме завантаження файлів до хмарного сховища мінімізує кількість проміжних операцій обробки та сприяє підвищенню продуктивності системи під час роботи з великими документами.

2.3.3 Модуль автоматичної Serverless-обробки планів (AWS Lambda та Boto3)

Для автоматизації обробки навчальних планів у системі використано безсерверні функції AWS Lambda. Такий підхід дозволяє виконувати ресурсоємні операції окремо від основного серверного застосунку та забезпечує незалежне масштабування обчислювальних процесів.

Після завантаження файлу до сховища автоматично генерується подія, яка ініціює виконання безсерверної функції. Функція виконує аналіз структури документа, перевірку коректності даних, їх валідацію та перетворення у стандартизований формат для подальшого використання в системі.

Для взаємодії із сервісами AWS використовується програмний інтерфейс Boto3, який забезпечує доступ до ресурсів хмарної платформи та дозволяє реалізувати інтеграцію між сховищем даних і модулем обробки.

Результатом виконання функції є формування структурованого набору даних, який передається до інформаційної системи для подальшого збереження та використання під час формування навчального розкладу. Такий підхід забезпечує відокремлення процесів обробки навчальних планів від основної бізнес-логіки вебзастосунку та сприяє підвищенню загальної продуктивності системи.

2.4 Проєктування структури бази даних

Для реалізації структури бази даних інформаційної системи автоматизованого формування розкладу навчального процесу було виконано послідовне концептуальне моделювання предметної області. На етапі проєктування використано підхід Entity–Relationship (ER), який є одним із найбільш поширених методів побудови логічної структури баз даних [35]. Основою ER-моделі є виділення сутностей предметної області, визначення їх атрибутів та встановлення логічних зв’язків між ними. Такий підхід дозволяє

формалізувати інформаційні об’єкти системи, забезпечити цілісність даних та підготувати структуру до подальшої реалізації у реляційній базі даних.

Детальне опрацювання структури бази даних передбачає проходження кількох послідовних етапів. Першим етапом є визначення сутностей предметної області та формування їх опису. Сутність у контексті ER-моделі розглядається як об’єкт реального світу, інформацію про який необхідно зберігати в системі. На цьому етапі було виділено основні інформаційні об’єкти системи розкладу:

- Університет (University);
- Факультет (Faculty);
- Кафедра (Department);
- Освітня програма (Program);
- Академічна група (UserGroup);
- Користувач (User);
- Студент (Student);
- Викладач (Teacher);
- Адміністратор (Administrator);
- Дисципліна (Subject);
- Навчальне навантаження (CurriculumAssignment);
- Навантаження_Групи (CurriculumGroup);
- Навантаження_Лектори (CurriculumLecturer);
- Навантаження_Практики (CurriculumPractice);
- Часовий слот (TimeSlot);
- Розклад (ScheduleSlot);
- Розклад_Груп (ScheduleGroup);
- Побажання викладача (TeacherPreference).

Визначення сутностей дозволило сформувати концептуальний каркас майбутньої бази даних та відобразити структуру предметної області без прив’язки до конкретної системи керування базами даних.

Сутність «Університет» є кореневим елементом організаційної структури системи та використовується для зберігання загальної інформації про заклад вищої освіти. З нею безпосередньо пов’язані сутності «Факультет», «Кафедра», «Освітня програма» та «Академічна група», які формують ієрархічну структуру навчального процесу. Такий підхід забезпечує логічне групування даних та дозволяє розмежовувати інформаційні потоки між окремими структурними підрозділами університету.

Сутність «Користувач» є центральним елементом ER-моделі системи та використовується для автентифікації й авторизації всіх учасників освітнього процесу. На її основі реалізовано спеціалізовані профілі «Студент», «Викладач» та «Адміністратор», що дозволяє розділити загальні облікові дані користувачів та їх функціональні ролі в системі [33]. Сутності «Студент» і «Викладач» забезпечують зв’язок користувачів з академічними групами та кафедрами відповідно, тоді як сутність «Адміністратор» відповідає за управління системою та налаштування процесів формування розкладу.

Сутність «Дисципліна» описує навчальні предмети, які викладаються в межах освітніх програм. Для деталізації навчального процесу використовується сутність «Навчальне навантаження», яка містить інформацію про семестр, курс, кількість лекційних та практичних занять. Додатково застосовуються проміжні сутності «Навантаження_Групи», «Навантаження_Лектори» та «Навантаження_Практики», що реалізують складні зв’язки між дисциплінами, викладачами та академічними групами. Використання таких сутностей дозволяє підтримувати як потокові лекції для кількох груп одночасно, так і індивідуальний розподіл практичних занять між викладачами.

Для представлення часової структури навчального процесу використано сутності «Часовий слот» та «Розклад». Сутність «Часовий слот» визначає абстрактну часову сітку занять, що містить інформацію про день тижня, час проведення та тип навчального тижня. Сутність «Розклад» представляє

конкретні заняття, сформовані алгоритмом генерації розкладу, та пов’язує між собою дисципліни, викладачів і часові інтервали. Для підтримки потокових лекцій та занять, у яких бере участь декілька академічних груп, використовується проміжна сутність «Розклад_Груп».

Окрему роль у моделі відіграє сутність «Побажання викладача», яка реалізує механізм урахування нежорстких обмежень під час автоматичного формування розкладу. Вона дозволяє фіксувати рівень доступності викладача для кожного часового слота та використовується алгоритмом генерації розкладу для пошуку вільного часу для проведення занять.

Другим етапом стало визначення атрибутів сутностей. Атрибути використовуються для опису характеристик конкретної сутності та забезпечують збереження необхідної інформації про об’єкти предметної області. В ER-моделі атрибути виступають властивостями сутностей і формують структуру майбутніх таблиць бази даних. На цьому етапі для кожної сутності було визначено набір атрибутів, необхідних для реалізації функціоналу системи. Така деталізація дозволяє забезпечити повноцінне збереження даних і підтримку бізнес-логіки системи. На рисунку 2.5 наведено сутності разом із визначеними для них атрибутами.

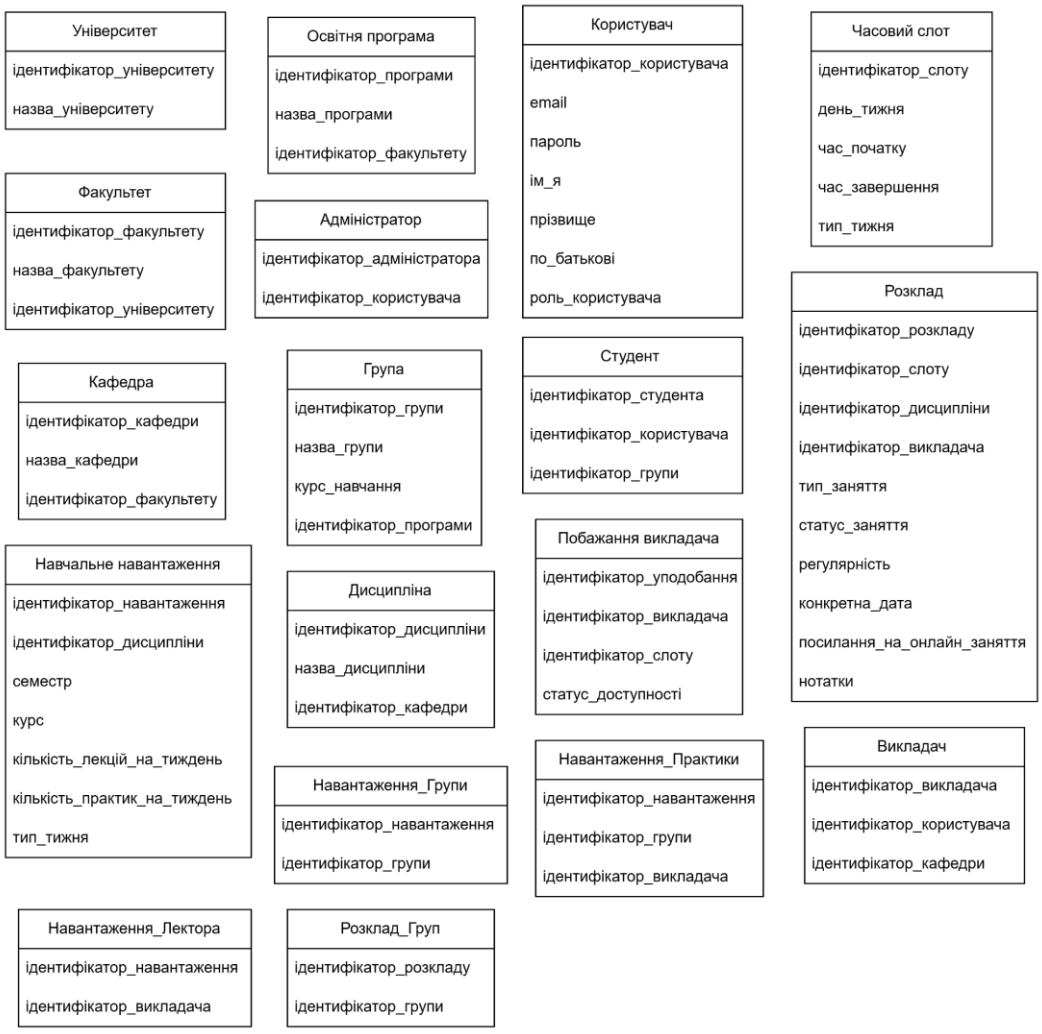


Рисунок 2.5 - Визначені атрибути для сутностей

Третім етапом стало визначення зв’язків між сутностями. Зв’язки є ключовим елементом ER-моделі, оскільки відображають взаємодію між об’єктами предметної області та забезпечують логічну цілісність інформації. У процесі проєктування були використані зв’язки типів «один-до-багатьох» (1:M), «один-до-одного» (1:1) та «багато-до-багатьох» (M:N).

Окрему увагу було приділено моделюванню користувачів системи. Базова сутність «User» використовується для реалізації механізмів автентифікації та авторизації, тоді як сутності «Student», «Teacher» та «Administrator» виступають спеціалізованими профілями користувачів і пов’язані із сутністю «User» через

зв’язки типу 1:1. Такий підхід дозволяє уникнути дублювання даних та забезпечує розширюваність структури системи. Аналогічно було реалізовано модель навчального навантаження та розкладу, де складні взаємозв’язки між дисциплінами, викладачами, академічними групами та часовими слотами відображають реальні процеси організації навчального процесу. В таблиці 2.1 наведено встановлені зв’язки між сутностями.

Таблиця 2.1 – Характеристика зв’язків між сутностями ER-моделі

Пов’язані сутності	Тип зв’язку	Призначення зв’язку
Університет – Факультет	1:M	Один університет об’єднує декілька факультетів, кожен факультет належить лише одному університету.
Факультет – Кафедра	1:M	Факультет включає декілька кафедр, які забезпечують викладання дисциплін та роботу викладачів.
Факультет – Освітня програма	1:M	Один факультет реалізує кілька освітніх програм підготовки студентів.
Освітня програма – Академічна група	1:M	За однією освітньою програмою може бути сформовано декілька академічних груп.
Користувач – Студент	1:1	Кожен студент має один обліковий запис користувача системи.
Користувач – Викладач	1:1	Кожен викладач пов’язаний з одним обліковим записом користувача.
Користувач – Адміністратор	1:1	Адміністратор є спеціалізованим профілем користувача з розширеними правами доступу.
Академічна група – Студент	1:M	До однієї академічної групи входить багато студентів, студент належить лише одній групі.
Кафедра – Викладач	1:M	На кафедрі працює декілька викладачів, кожен викладач закріплений за однією кафедрою.

Продовження таблиці 2.1

Кафедра – Дисципліна	1:M	Кафедра відповідає за викладання та супровід декількох дисциплін.
Дисципліна – Навчальне навантаження	1:M	Для однієї дисципліни можуть існувати різні варіанти навчального навантаження залежно від курсу або семестру.
Навчальне навантаження – Академічна група	M:N	Одне навантаження може бути призначене кільком групам, а група має багато навчальних навантажень (реалізується через «Навантаження_Групи»).
Навчальне навантаження – Викладач (лекції)	M:N	Дозволяє призначати одного або кількох викладачів для проведення лекційних занять (реалізується через «Навантаження_Лектори»).
Навчальне навантаження – Група – Викладач (практики)	M:N	Закріплення викладача за практичними заняттями окремих груп (реалізується через «Навантаження_Практики»).
Часовий слот – Розклад	1:M	Один часовий слот може використовуватися багатьма елементами розкладу.
Дисципліна – Розклад	1:M	Одна дисципліна може мати багато занять протягом семестру.
Викладач – Розклад	1:M	Один викладач проводить багато занять у розкладі.
Розклад – Академічна група	M:N	Одне заняття може проводитися для кількох груп (реалізується через «Розклад_Груп»).
Викладач – Часовий слот	M:N	Визначає доступність викладача у певні часові інтервали (реалізується через «Побажання викладача»).

Після завершення етапів визначення сутностей, формування атрибутів та встановлення зв’язків між ними всі елементи були об’єднані в єдину ER-діаграму, що відображає логічну структуру бази даних системи

автоматизованого формування розкладу. Повну структуру розробленої моделі подано на рисунку 2.6.

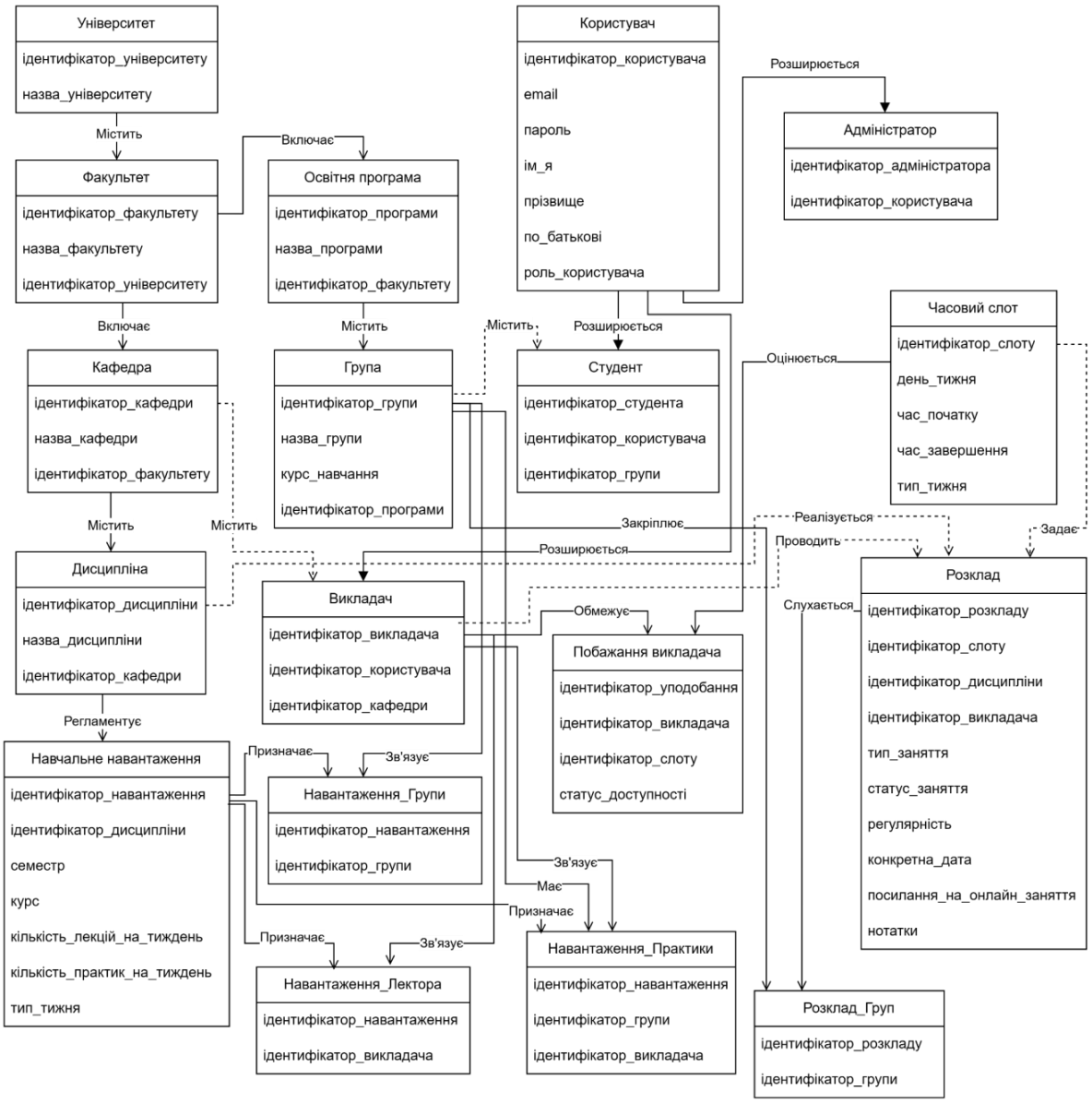


Рисунок 2.6 - ER діаграма системи

Логічна модель даних. Після завершення етапу концептуального проєктування та побудови ER-діаграми було виконано розроблення логічної моделі бази даних. Логічне проєктування є наступним кроком після визначення

сутностей предметної області, їх атрибутів та взаємозв'язків і полягає у трансформації концептуальної моделі в реляційну структуру, придатну для подальшої реалізації в системі керування базами даних.

На даному етапі кожна виділена сутність предметної області перетворюється на окреме відношення (таблицю), а встановлені між сутностями зв'язки реалізуються за допомогою первинних та зовнішніх ключів. Такий підхід забезпечує структуроване зберігання інформації, підтримання цілісності даних та мінімізацію дублювання відомостей у базі даних.

Розроблена логічна модель охоплює всі ключові компоненти системи управління навчальним процесом, зокрема організаційну структуру університету, користувачів різних ролей, освітні програми, академічні групи, дисципліни, навчальне навантаження, часові слоти, розклад занять та механізм урахування побажань викладачів. Окрему увагу приділено реалізації зв'язків типу «багато-до-багатьох», для яких передбачено спеціалізовані проміжні таблиці, що дозволяють коректно відображати потокові лекції, розподіл практичних занять та структуру сформованого розкладу.

У процесі логічного проєктування виконано нормалізацію структури бази даних до третьої нормальної форми (3НФ) [36]. Застосування нормалізації дозволило усунути надлишковість даних, мінімізувати ризик виникнення аномалій вставки, оновлення та видалення інформації, а також забезпечити логічну цілісність даних. Перша нормальна форма забезпечує атомарність атрибутів та відсутність повторюваних груп даних, друга нормальна форма усуває часткові функціональні залежності, а третя нормальна форма виключає транзитивні залежності між неключовими атрибутами. У результаті всі атрибути розроблених таблиць функціонально залежать лише від первинних ключів відповідних сутностей.

У розробленій структурі дані про університети, факультети, кафедри, освітні програми, академічні групи, користувачів, дисципліни, навчальне

навантаження, розклад занять та побажання викладачів зберігаються в окремих взаємопов'язаних таблицях, що дозволило уникнути дублювання інформації та забезпечити підтримку цілісності даних за допомогою механізму зовнішніх ключів.

У результаті було сформовано повну логічну модель бази даних, яка відображає структуру предметної області та визначає спосіб організації даних у майбутній інформаційній системі. Логічна модель бази даних побудована відповідно до функціональних вимог системи та визначених бізнес-процесів, пов'язаних із веденням організаційної структури університету, управлінням користувачами, формуванням навчального навантаження, обліком побажань викладачів та автоматизованим генеруванням розкладу занять. Логічну модель розробленої бази даних наведено на рисунку 2.7.

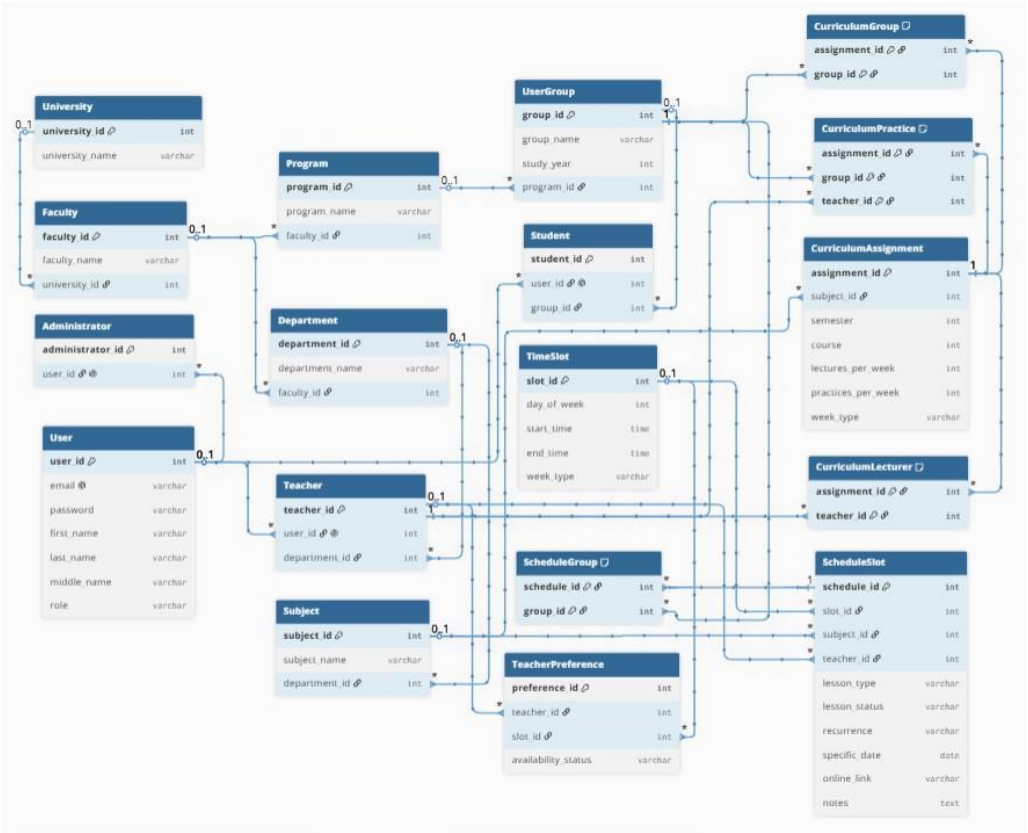


Рисунок 2.7 - Логічна модель розробленої бази даних

Фізична модель даних. Розроблення фізичної моделі є наступним та завершальним кроком після побудови концептуальної та логічної моделей предметної області. На відміну від логічної моделі, яка відображає структуру даних незалежно від конкретної системи керування базами даних (СУБД), фізична модель деталізує специфіку їх зберігання, організації доступу та обробки в обраному програмному середовищі.

Цільовою платформою для розробленої системи обрано реляційну СУБД MySQL версії 8.0. З метою забезпечення ізольованості, зручності масштабування та спрощення процесів розгортання, фізичний сервер бази даних функціонує у контейнеризованому середовищі за допомогою технології Docker.

Трансформація логічної моделі у фізичну базу даних реалізується автоматизовано за допомогою технології об'єктно-реляційного відображення (ORM) на базі фреймворку Hibernate [37]. Під час генерації фізичної схеми абстрактні типи даних застосунку транслюються у специфічні типи MySQL: рядкові дані перетворюються на VARCHAR(255), цілочисельні ідентифікатори — на BIGINT із властивістю AUTO_INCREMENT, а логічні значення — на BIT(1). Для забезпечення коректного збереження кириличних символів фізичні таблиці конфігуруються з використанням кодування utf8mb4.

Важливим аспектом фізичної реалізації є обробка складних ієрархічних зв’язків між сутностями. При використанні ORM-технологій та автоматичної JSON-серіалізації двонаправлені зв’язки часто спричиняють критичні проблеми циклічних посилань та рекурсивного зациклення під час формування REST-відповідей [33]. Для запобігання подібним проблемам на фізичному рівні архітектури використано механізми контролю серіалізації (анотації @JsonManagedReference та @JsonBackReference бібліотеки Jackson), які дозволяють обмежувати вкладені зв’язки та коректно формувати структуру відповідей сервера [33].

Особливу увагу під час фізичного проєктування приділено коректній побудові структури та забезпеченню цілісності даних. На рівні СУБД автоматично генеруються обмеження первинних (Primary Keys) та зовнішніх (Foreign Keys) ключів, створюються індекси для прискорення запитів і визначаються правила каскадних операцій для залежних сутностей. Це створює надійну та продуктивну основу для функціонування ресурсоємних алгоритмів автоматичного формування розкладу навчального процесу.

У результаті сформовано фізичну модель, яка відображає реальну структуру бази даних системи та деталізує спосіб зберігання інформації про ключові об'єкти: організаційну структуру університету, користувачів, дисципліни, навчальне навантаження та розклад занять. Остаточну фізичну схему розробленої бази даних наведено на рисунку 2.8.

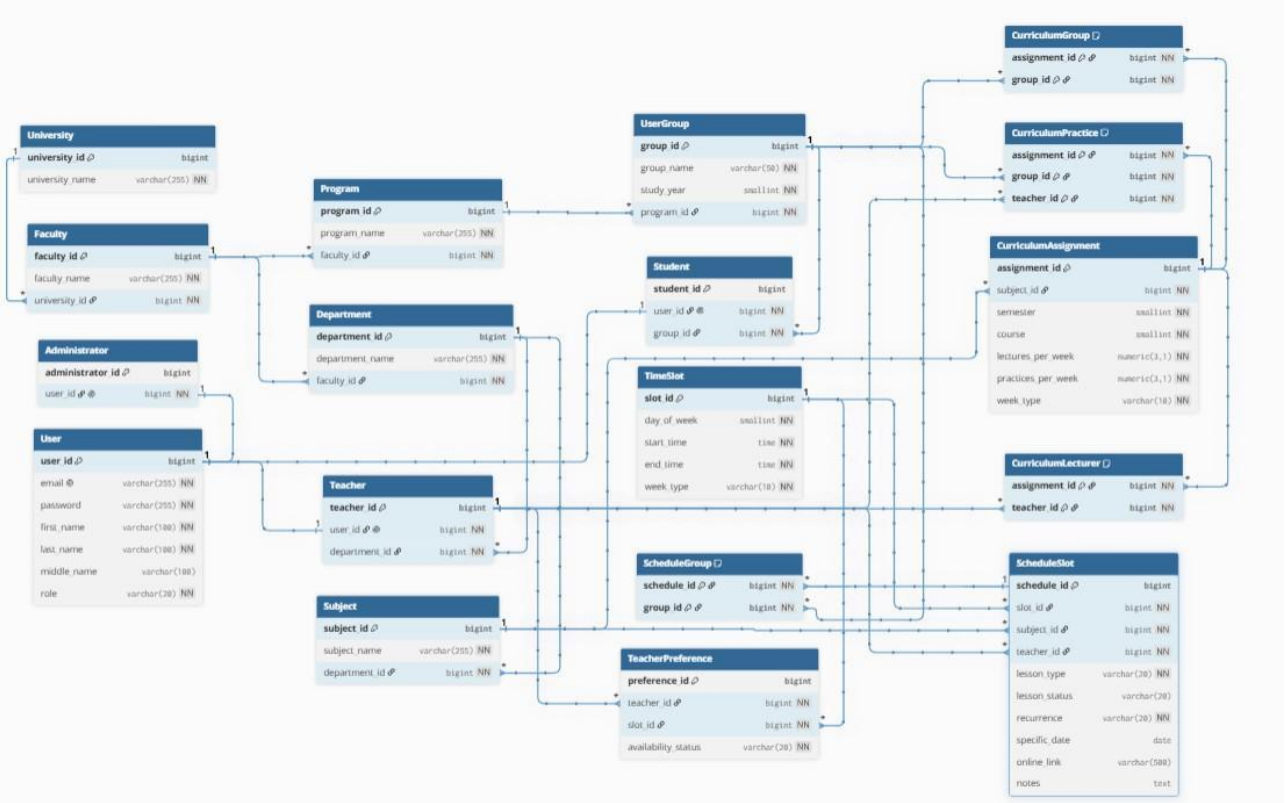


Рисунок 2.8 - Фізична модель розробленої бази даних

Таким чином, розроблена структура бази даних забезпечує цілісне та узгоджене зберігання інформації про організаційну структуру закладу освіти, користувачів системи, навчальні дисципліни, академічні групи, навчальне навантаження та розклад занять. Використання первинних і зовнішніх ключів, обмежень цілісності та проміжних таблиць для реалізації зв'язків типу «багато-до-багатьох» дозволяє підтримувати коректність даних і виключати їх дублювання.

Особливістю розробленої моделі є підтримка мультиорендної архітектури (multi-tenant architecture) [38], що забезпечує можливість роботи системи одночасно з кількома закладами вищої освіти. Ізоляція даних реалізується через сутність університету та пов'язані з нею структурні елементи, що дозволяє масштабувати систему без зміни її логічної структури. При цьому використовується підхід Shared Database [39], за якого всі університети працюють у межах єдиної бази даних із логічним розмежуванням інформації. За необхідності архітектура може бути розширена до моделі Schema per University, що забезпечує фізичне розділення даних між окремими закладами освіти.

Запропонована структура бази даних підтримує горизонтальне масштабування, спрощує супровід системи та створює основу для подальшого розвитку функціональності. Побудована фізична модель повністю відповідає вимогам автоматизованого управління навчальним процесом і забезпечує ефективне функціонування алгоритмів автоматичного формування розкладу занять з урахуванням навчального навантаження, потокових лекцій, практичних занять та побажань викладачів.

2.5 Розробка гібридного алгоритму автоматичного формування розкладу

Математична постановка задачі та обґрунтування гібридного підходу. Проблематика синтезу розкладу навчальних занять (Timetabling Problem)

класифікується у світовій науковій практиці як NP-повна комбінаторна задача комбінаторного пошуку [7]. Специфіка її розв'язання полягає у необхідності просторово-часового розподілу множини навчальних подій за умови дотримання комплексу взаємозалежних обмежень. Збільшення розмірності вхідних даних провокує експоненційне зростання простору пошуку рішень, що робить застосування методів повного перебору обчислювально неможливим та вимагає впровадження автоматизованих систем для підвищення ефективності функціонування закладу освіти [3].

Сучасні дослідження доводять, що для розв'язання задач такого класу найвищу результативність демонструють метаевристичні та еволюційні методи обчислень [8]. Зокрема, технологія еволюційного формування розкладів базується на поданні розкладу у вигляді абстрактної інформаційної макроструктури (хромосоми), яка піддається ітеративним операціям селекції та модифікації з метою поступового наближення до найбільш раціонального значення цільової функції [6].

Проте, із застосування генетичних алгоритмів (ГА) для освітніх закладів, класична парадигма ГА має суттєвий концептуальний недолік. Стохастичне формування початкової популяції неминуче призводить до генерації критичної маси нежиттєздатних рішень, що порушують жорсткі обмеження і спричиняють передчасну збіжність алгоритму до локальних екстремумів [5].

З метою нівелювання цього недоліку в межах даної роботи спроєктовано та реалізовано гібридний детерміновано-еволюційний алгоритм. Його наукова новизна полягає у синергії жадібної евристики для забезпечення початкової валідності простору рішень та генетичного пошуку для виконання м'яких критеріїв якості. Процес функціонування запропонованого алгоритму структурно декомпозується на три стадії: жадібна ініціалізація, стохастичне вдосконалення та математичне календарне розгортання.

Етап детермінованої жадібної ініціалізації. Формування стартової популяції генетичного ядра здійснюється не випадковим генератором, а за допомогою спеціалізованої жадібної евристики. Вхідним вектором даних виступають параметризовані об'єкти дисциплін, матриця доступності аудиторій та персональні обмеження викладацького складу [8].

Алгоритм здійснює первинний розподіл за принципом пріоритезації структурної складності:

- Потокові лекційні заняття: Розміщуються першочергово, оскільки вони формують жорсткі вузли графа розкладу, зв'язуючи одного лектора з множиною академічних груп. Алгоритм виконує пошук першого доступного часового слота, що задовольняє умову відсутності колізій для всіх учасників потоку.
- Практичні заняття: Розподіляються ітеративно після закріплення лекцій. При цьому застосовується строге правило хронологічної залежності: індекс дня тижня для практичного заняття повинен бути математично більшим або дорівнювати індексу дня відповідної лекції [6].

Наслідком виконання цього етапу є генерація популяції з 30 початкових хромосом. Кожна з них гарантовано є валідною і задовольняє всі жорсткі обмеження, проте характеризується низьким рівнем ергономічності через наявність незручних часових розривів.

Механізм еволюційного вдосконалення та оцінки пристосованості. Наступним кроком є запуск еволюційного процесу, тривалість якого обмежена 50 поколіннями. Ядром процесу покращення розкладу є математична модель цільової функції (Fitness Function), яка формалізує вимоги до якості розкладу [5].

Головним критерієм якості визначено мінімізацію так званих технологічних «вікон» — часових проміжків між заняттями. Ліквідація таких

розривів є фундаментальною умовою покращення соціально-економічної та логістичної складової функціонування закладу освіти, оскільки вона забезпечує раціональне використання часу здобувачів освіти та науково-педагогічних працівників [4].

Цільова функція реалізує алгоритмічну штрафну стратегію, яку математично можна формалізувати у вигляді рівняння (2.1)

$$F = F_{\max} - \sum_{i=1}^{N} (W_{\text{i}} \cdot P_{\text{w}}) \quad (2.1)$$

де F — підсумкове значення цільової функції (пристосованості) хромосоми, при цьому мета алгоритму полягає у максимізації цього показника ($F \rightarrow max$);

$F_{\max}$ — базовий ідеальний показник пристосованості (у розробленій системі ініціалізується значенням 10 000 балів);

N — загальна кількість академічних груп у системі;

W_{i} — кількість зафіксованих технологічних «вікон» (часових розривів між заняттями тривалістю понад 20 хвилин) у розкладі і-ї групи;

P_{w} — фіксована штрафна константа за одне знайдене «вікно» (P_{w} = 150).

Під час оцінювання алгоритм ітерує розклад кожної академічної групи та підраховує кількість виявлених технологічних розривів. Отже, чим менше «вікон» у розкладі, тим менший сумарний штраф віднімається від ідеалу, і тим вищу підсумкову оцінку пристосованості отримує згенерований варіант розкладу.

Після ранжування популяції за показником цільової функції, вмикається оператор селекції. У розробленій системі використано метод жорсткого

елітарного відбору: топ-10 найкращих хромосом безумовно мігрують у наступну генерацію, тоді як інші 20 особин видаляються.

Для забезпечення генетичного різноманіття та виходу алгоритму з локальних екстремумів) застосовується оператор мутації [5]. Із множини елітних хромосом випадково обираються донори, в яких точково модифікуються часові координати виключно практичних занять. Лекційні потоки при цьому залишаються незмінними, що гарантує збереження структурної цілісності розкладу без ризику виникнення критичних колізій.

Еволюційний цикл триває рівно 50 генерацій. Дана кількість ітерацій є гіперпараметром моделі, який було визначено емпіричним шляхом під час попереднього тестування алгоритму. Аналіз динаміки збіжності показав, що при меншій кількості ітерацій (наприклад, 20–30) алгоритм не встигає ефективно мінімізувати всі часові розриви. Водночас після 50-ї ітерації значення цільової функції стабілізується (виходить на плато), і подальше виконання еволюційного циклу лише збільшує час обчислень без суттєвого покращення фінального результату. Таким чином, обране значення забезпечує раціональний баланс між обчислювальною швидкістю системи та якістю згенерованої сітки занять. Після проходження 50 ітерацій еволюційного циклу алгоритм екстрагує єдину хромосому з максимальним показником пристосованості, яка затверджується як фінальний тижневий шаблон.

Математичне календарне розгортання семестру. Фінальним етапом роботи гібридного алгоритму є проєкція знайденого абстрактного тижневого шаблону на фізичний календар навчального семестру [3]. Для коректної генерації занять розроблено алгоритм динамічного обчислення парності тижня відносно адміністративної еталонної дати.

Тип поточного тижня визначається алгоритмічно на основі обчислення часової дельти між адміністративною еталонною датою-орієнтиром та поточною датою ітерації семестру. Алгоритм визначає загальну кількість календарних днів

між цими датами, після чого розраховує кількість повних тижнів, що минули шляхом цілочисельного ділення кількості днів на 7. Далі система перевіряє парність отриманого індексу тижня за допомогою операції знаходження остачі від ділення на 2. Якщо остача відсутня, поточний тиждень класифікується як непарний, а в іншому випадку — ідентифікується як парний.

Під час циклічного проходу по всіх днях семестру система зіставляє тип тижня та день тижня з даними хромосоми-шаблону. У разі позитивного збігу генеруються фізичні сутності занять, які пакетно фіксуються в реляційній базі даних, завершуючи процес автоматизованого формування розкладу. На рисунку 2.9 наведено блок-схему роботи розробленого гібридного алгоритму автоматичного формування розкладу.

Для перевірки працездатності розробленого гібридного алгоритму та генерації фінальної сітки занять було використано реальний тестовий сценарій. Через підсистему імпорту завантажувався файл навчального плану, який містив 6 навчальних дисциплін для першого семестру. Згідно з налаштуваннями, ці дисципліни розподілялися між 3 академічними групами. Для забезпечення навчального процесу алгоритм оперував кадровим резервом факультету, що складався з 15 науково-педагогічних працівників (по 3 викладачі на кожну з 5 кафедр). Під час розподілу навантаження система також автоматично зчитувала та враховувала персональні обмеження (зафіксовані часові побажання) викладачів з бази даних. Така розмірність даних дозволила повноцінно протестувати як швидкодію еволюційного ядра алгоритму, так і його здатність ефективно вирішувати просторово-часові колізії.



Рисунок 2.9 – Блок-схема гібридного алгоритму автоматичного формування розкладу

2.6 Розробка модулів системи та користувацьких інтерфейсів

Процес розробки користувацьких інтерфейсів інформаційної системи базувався на принципі створення високопродуктивного та інтерактивного робочого середовища. В основу архітектури покладено модель тонкого клієнта, де основна бізнес-логіка та збереження даних ізольовані на сервері, а браузер клієнта відповідає виключно за відображення контенту, валідацію вводу та асинхронну взаємодію з програмним інтерфейсом шляхом обміну даними у форматі JSON.

Архітектурні рішення та компонентний підхід. Для забезпечення гнучкості та легкості подальшого масштабування системи розробка інтерфейсу велася з використанням компонентного підходу. Весь користувацький інтерфейс було декомпозировано на незалежні модулі, кожен з яких відповідає за окрему бізнес-функцію. Серед ключових рішень реалізовано модуль адміністрування для керування структурою закладу вищої освіти та управління правами користувачів, модуль імпорту для завантаження навчальних планів, модуль

викладача для встановлення часових уподобань, а також клієнтський модуль розкладу для відображення фінальної сітки занять. Для підтримки візуальної цілісності було розроблено власну бібліотеку базових елементів, яка включає стандартизовані форми, таблиці з динамічним сортуванням, модальні вікна та індикатори завантаження. Це дозволило уникнути дублювання коду та забезпечити єдиний корпоративний стиль системи.

Розробка модуля візуалізації часової сітки. Основним і найбільш технологічно складним елементом системи є модуль відображення розкладу. Замість використання класичних статичних таблиць, які погано адаптуються під складні перехресні заняття, було розроблено механізм динамічного позиціонування карток дисциплін. Алгоритм рендерингу перетворює абстрактні часові дані сервера у фізичні координати на екрані пристрою. Кожне навчальне заняття обробляється як окремий об'єкт. Його позиція по вертикалі вираховується на основі часу початку пари, а висота картки є пропорційною до тривалості заняття у хвилинах. Позиція по горизонталі визначається приналежністю до конкретної аудиторії чи академічної групи. Застосування абсолютного позиціонування дозволяє системі коректно відображати потокові лекції, розтягуючи їх на кілька колонок, наочно демонструвати вільні проміжки між заняттями та уникати візуального накладання у разі виникнення конфліктів, підсвічуючи такі ситуації попереджувальними кольорами.

Модуль збору часових уподобань викладачів. Для реалізації вимоги щодо врахування побажань викладачів розроблено інтерактивну матрицю доступності. Інтерфейс представляє собою таблицю, де стовпці відповідають дням тижня, а рядки – стандартним часовим слотам. З метою забезпечення максимальної ергономіки у цьому модулі реалізовано патерн оптимістичного оновлення інтерфейсу. При натисканні на елемент вибору його стан змінюється миттєво, надаючи викладачу візуальний зворотний зв'язок. Паралельно у фоновому режимі відправляється асинхронний запит до сервера для збереження

змін у базі даних. Якщо сервер повертає помилку, інтерфейс автоматично скасовує візуальну зміну і показує повідомлення про збій. Це дозволяє користувачу безперервно та швидко заповнювати матрицю, не чекаючи на відповідь сервера після кожної дії.

Інтерфейс адміністрування та моніторингу асинхронних процесів. Особливу увагу було приділено розробці інтерфейсу для імпорту навчальних планів, оскільки ця операція виконується асинхронно через хмарні сервіси. Процес у фронтенді реалізовано шляхом отримання тимчасового підписаного посилання від сервера. Файл завантажується безпосередньо у хмарне сховище, оминаючи основний сервер, що значно розвантажує мережевий трафік системи. Одразу після завантаження активується алгоритм короткого опитування статусу. Клієнтська частина періодично надсилає запити до програмного інтерфейсу для перевірки стану обробки файлу хмарною функцією. У цей час інтерфейс блокує подальші дії адміністратора та відображає динамічний індикатор прогресу. Після отримання статусу успішного завершення сторінка автоматично оновлює списки дисциплін без повного перезавантаження сторінки браузера.

Інтеграція механізмів безпеки у користувацький інтерфейс. Попри те, що безпека є прерогативою серверної частини, інтерфейс відіграє ключову роль у підтримці рольової моделі доступу. Фронтенд-модулі повністю ізольовані від конфіденційних даних користувачів. Після успішної автентифікації клієнтський додаток отримує сесійний токен, який зберігається у захищеному сховищі браузера та додається до заголовків усіх наступних запитів. На основі цього токена фронтенд динамічно приховує або відображає певні розділи меню відповідно до прав користувача. Наприклад, студент не має доступу до панелі завантаження планів, а механізми маршрутизації примусово перенаправляють користувачів при спробі несанкціонованого доступу до адміністративних сторінок.

Забезпечення якості, валідація та адаптивність. Для підвищення відмовостійкості системи у фронтенді впроваджено строгу попередню валідацію даних. Перед відправкою будь-якої форми система перевіряє коректність вводу за допомогою регулярних виразів, що створює перший рубіж захисту та доповнюється серверною верифікацією. Крім того, інтерфейс розроблено з урахуванням принципів адаптивного вебдизайну. Використання гнучких сіток дозволяє системі коректно відображатися як на широкоформатних моніторах в аудиторіях, так і на мобільних пристроях студентів, зберігаючи повну функціональність перегляду розкладу.

Візуальні результати виконання описаних сценаріїв, включно зі знімками екрана, що підтверджують коректність роботи користувацьких інтерфейсів системи, наведено у Додатку А.

РОЗДІЛ 3 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Методика тестування

Методика тестування розробленої системи базується на підході «піраміди тестування», що охоплює модульні (unit), інтеграційні та прийомочні тести. Основним середовищем для розробки та виконання автоматизованих тестів було обрано IntelliJ IDEA, яка надає вбудовану підтримку фреймворку JUnit 5.

Для перевірки бізнес-логіки використовувалися юніт-тести з анотацією @Test, що дозволяє ізолювати окремі компоненти (сервіси обробки планів, валідатори, алгоритми генерації) та перевіряти їх коректність без підключення до бази даних чи зовнішніх сервісів. Інтеграційні тести виконувалися з використанням контейнерів (Testcontainers) для емуляції бази даних MySQL та перевірки коректності взаємодії з AWS S3 через мокові інтерфейси.

Тестування функціональних модулів. Для підтвердження відповідності системи функціональним вимогам (розділ 1.4) було проведено комплексне наскрізне тестування всіх підсистем. Тестові сценарії (Test Cases) охоплюють перевірку рольової моделі, модуля адміністрування структури, процесів імпорту, роботи алгоритму генерації та користувацьких інтерфейсів викладача і студента. Результати тестування наведено у таблиці 3.1.

Таблиця 3.1 – Комплексні тест-кейси перевірки інформаційної системи

ID	Назва тесту	Передумови	Очікуваний результат	Статус
TC-01	Успішна автентифікація	Неавторизований користувач	Успішний вхід, отримання JWT-токена, перенаправлення відповідно до ролі	Пройдено
TC-02	Помилка автентифікації	Неавторизований користувач	Відображення повідомлення «Невірний логін або пароль»	Пройдено

Продовження таблиці 3.1

ТС-03	Рольова модель (RBAC)	Авторизація: Студент	Блокування доступу до адмін-панелі	Пройдено
ТС-04	Створення об'єкта структури	Авторизація: Адмін	Новий факультет успішно створено та збережено	Пройдено
ТС-05	Валідація дублікатів	Авторизація: Адмін	Повідомлення про існування групи	Пройдено
ТС-06	Завантаження валідного плану	Авторизація: Адмін	Успішне завантаження в AWS S3 та обробка Lambda	Пройдено
ТС-07	Блокування невалідного формату	Авторизація: Адмін	Заборона завантаження файлів .pdf та .docx	Пройдено
ТС-08	Обробка неповного плану	Авторизація: Адмін	Відображення списку помилок валідації	Пройдено
ТС-09	Таймаут хмарної обробки	Авторизація: Адмін	Коректна обробка втрати мережі та відновлення роботи	Пройдено
ТС-10	Успішна генерація розкладу	Авторизація: Адмін	Розклад згенеровано без конфліктів менш ніж за 10 секунд	Пройдено
ТС-11	Генерація з нестачею ресурсів	Авторизація: Адмін	Нерозподілені заняття підсвічуються як конфлікти	Пройдено
ТС-12	Ручне вирішення конфлікту	Авторизація: Адмін	Конфлікт усунуто, зміни збережено	Пройдено
ТС-13	Збереження уподобань	Авторизація: Викладач	Дані миттєво відображаються та успішно зберігаються	Пройдено
ТС-14	Обробка гонки запитів	Авторизація: Викладач	Фінальний стан UI відповідає БД	Пройдено

Продовження таблиці 3.1

ТС-15	Каскадні фільтри студента	Авторизація: Гість/Студент	Динамічне оновлення списку груп	Пройдено

Буферна синхронізація та аналіз продуктивності за логами CloudWatch. Оскільки обробка навчальних планів виконується асинхронно, для взаємодії між серверною частиною системи та клієнтським інтерфейсом використовується механізм короткого опитування (Short Polling). Суть підходу полягає у періодичному надсиланні клієнтським застосунком запитів до сервера з метою перевірки готовності результатів обробки. Після завершення виконання безсерверної функції клієнтський інтерфейс отримує відповідні дані та відображає результати обробки користувачу.

Для моніторингу роботи безсерверної інфраструктури та аналізу її продуктивності використано сервіс Amazon CloudWatch Logs. Журнали виконання дозволяють контролювати використання пам’яті, тривалість виконання функцій та виявляти потенційні проблеми під час роботи системи (рис. 3.1).


Recent invocations

#	Timestamp	RequestId	LogStream	DurationInMS	BilledDurationInMS	MemorySetInMB	MemoryUsed
1	2026-06-02T13:22:55.481Z	1f022bf6-ba1a-4cd4-a2dd-5859c1a6cc37	2026/06/02/[$LATEST]f978372faf27455dbb4fa2785cb878d9	189.84	190.0	512.0	96.0
2	2026-06-02T13:21:53.116Z	1f022bf6-ba1a-4cd4-a2dd-5859c1a6cc37	2026/06/02/[$LATEST]f978372faf27455dbb4fa2785cb878d9	223.54	1077.0	512.0	95.0
3	2026-06-01T14:09:15.251Z	33e51adc-fe26-417a-b3d3-4d664ee2f80e	2026/06/01/[$LATEST]79e22698de44439382685 6c06a27786d	596.8	1263.0	512.0	95.0
4	2026-06-01T13:55:49.021Z	5ceabee4-1176-44ea-a9be-659a12c49b84	2026/06/01/[$LATEST]1f25a8eeca0b43b2baef8e07e80fd40c	527.81	528.0	512.0	95.0
5	2026-06-01T13:50:50.608Z	2d53468b-9572-4fe7-ab63-5ed4fa6dfb33	2026/06/01/[$LATEST]1f25a8eeca0b43b2baef8e07e80fd40c	592.74	1117.0	512.0	95.0
6	2026-06-01T13:41:35.384Z	8fc03345-7a0e-48c7-a602-1e802cef4c62	2026/06/01/[$LATEST]ae08188e787144b79ec21acd58c827e7	720.19	1545.0	512.0	95.0
7	2026-06-01T13:15:06.594Z	3d36f76c-c329-4f13-9516-a968e8c50c9b	2026/06/01/[$LATEST]296cc8ff793f42fe89454672f129e67e	714.38	1521.0	512.0	95.0
8	2026-06-01T13:02:23.165Z	44c41c1d-9cb7-430b-abb8-ed9b106c155b	2026/06/01/[$LATEST]4ae3a74223c44a55a752c388bcb5eef1	688.54	1316.0	512.0	95.0
9	2026-06-01T12:56:48.563Z	efebcb93-e911-4d15-ab01-900d7ca34c51	2026/06/01/[$LATEST]f5744c8f6c884b5aad4875d011a22ffa	620.18	621.0	512.0	97.0

Рисунок 3.1 - Журнал виконання хмарної функції у консолі Amazon CloudWatch

Для об’єктивної оцінки наведених метрик важливо враховувати обсяг вхідної інформації, що опрацьовувалася хмарною інфраструктурою. Тестування функції AWS Lambda виконувалося на реальному файлі навчального плану

(фрагмент якого наведено на рисунку 4.8). Файл містив матрицю з 10 дисциплін, розподілених між різними кафедрами та двома семестрами. Безсерверна функція мала розпізнати структуру документа, зчитати текстові дані та відфільтрувати 6 активних дисциплін для обраного першого семестру, щоб підготувати їх до розподілу між 3 академічними групами.

Проведений аналіз показав, що під час парсингу та структурування цього документа фактичне використання пам’яті становило близько 95–97 МБ при виділеному обсязі 512 МБ. Це свідчить про високу якість написання коду та раціональне використання доступних обчислювальних ресурсів хмари. Водночас тривалість виконання ресурсоємних операцій з обробки файлу перебувала в межах від 189 до 720 мс. Для заданої розмірності вхідного масиву даних та необхідності текстової фільтрації такий показник є надзвичайно успішним, оскільки він гарантує швидке реагування системи на завантаження нових документів та забезпечує своєчасне оновлення інформації у клієнтському інтерфейсі.

У процесі тестування також було виявлено проблему конкурентного доступу до даних, яка виникала під час одночасного виконання операцій читання та запису. Для усунення зазначеної проблеми було виконано рефакторинг механізму доступу до даних та впроваджено режим транзакцій лише для читання (Read-Only Transactions) під час виконання операцій перегляду інформації. Це дозволило забезпечити стабільну роботу системи при одночасному виконанні кількох паралельних процесів та підвищити надійність взаємодії між компонентами хмарної інфраструктури.

3.2 Оцінка ефективності роботи системи

Оцінка ефективності розробленого програмного комплексу здійснювалася шляхом зіставлення ключових метрик продуктивності нової системи з

традиційними (ручними) методами формування розкладу та існуючими настільними рішеннями.

Головним показником ефективності є час, витрачений на обробку масивів вхідних даних та безпосередню генерацію кінцевої часової сітки. Завдяки делегуванню ресурсомістких задач до безсерверного середовища хмарної інфраструктури (AWS Lambda), час первинної обробки та парсингу навчальних планів вдалося скоротити до діапазону 189–720 мс. При цьому повний цикл автоматичної генерації розкладу з урахуванням обмежень аудиторного фонду виконується менш ніж за 10 секунд.

Для наочності оцінки результатів сформовано порівняльну таблицю ключових показників (Таблиця 3.2).

Таблиця 3.2 – Порівняльна характеристика ефективності методів планування

Критерій оцінки	**Традиційний метод (ручний / Excel)**	**Локальні десктопні програми**	**Розроблена хмарна система**
Час формування розкладу	кілька робочих днів	1–3 години	До 10 секунд
Використання ресурсів ПК	Високе (ручна обробка та постійна участь користувача)	Максимальне навантаження на процесор та пам’ять	Мінімальне, оскільки основні обчислення виконуються у хмарі
Виявлення конфліктів	Візуальне, високий ризик людської помилки	Часткове автоматичне виявлення	Повністю автоматизоване підсвічування конфліктів
Врахування уподобань викладачів	Складне, потребує збору заяв або повідомлень	Обмежене або відсутнє	Автоматизоване через матрицю уподобань в інтерфейсі

Продовження таблиці 3.2

Масштабованість	Низька	Середня	Висока завдяки хмарній архітектурі
Доступність системи	Лише локально	На конкретному ПК	Доступ через веббраузер з будь-якого пристрою
Спільна робота користувачів	Практично відсутня	Обмежена	Одночасна робота адміністрації, викладачів та студентів
Резервне збереження даних	Ручне копіювання файлів	Локальні резервні копії	Автоматичне збереження та резервування у хмарі
Оновлення системи	Виконується вручну	Потребує перевстановлення	Автоматичне централізоване оновлення
Відмовостійкість	Низька	Залежить від стану ПК	Висока завдяки хмарній інфраструктурі

Додатковим фактором ефективності є економія апаратних ресурсів навчального закладу. Оскільки система побудована за мікросервісними принципами з використанням AWS S3 та Lambda, основний сервер (Spring Boot) не перевантажується під час масового завантаження планів на початку семестру, що гарантує безперебійний доступ до системи іншим користувачам.

3.3 Аналіз результатів роботи програмного рішення

Аналіз результатів експлуатаційного та наскрізного тестування підтверджує, що розроблене програмне рішення успішно виконує своє головне

призначення та повністю задовольняє функціональні й нефункціональні вимоги, сформульовані на етапі проєктування.

Аналізуючи роботу системи, можна виділити наступні ключові результати:

- Вирішення проблеми академічних конфліктів: Впроваджений алгоритм генерації довів свою здатність ефективно балансувати між жорсткими обмеженнями (неможливість проведення двох занять в одній аудиторії одночасно) та м'якими обмеженнями (часові уподобання викладачів). Усі нерозв'язані колізії, що виникають через об'єктивну нестачу аудиторного фонду, прозоро візуалізуються в інтерфейсі адміністратора для подальшого ручного коригування.
- Вдосконалення користувацького досвіду (UX): Застосування підходу «оптимістичного оновлення» (Optimistic UI) у модулі викладача та механізму короткого опитування (Short Polling) у модулі адміністратора дозволило створити ефект безперервної та миттєвої взаємодії із системою. Користувачі не стикаються із «зависанням» сторінок під час виконання важких фонових операцій.
- Надійність та цілісність даних: Як показав аналіз логів тестування, система стійка до високих навантажень. Виявлена на етапі тестування проблема конкурентного доступу до даних була успішно вирішена шляхом впровадження транзакцій лише для читання (Read-Only Transactions). Це гарантує, що під час одночасного перегляду розкладу сотнями студентів база даних не блокується і продовжує стабільно обробляти інші запити.

3.4 Формулювання результатів досліджень та їх оцінка

За результатами проведеного тестування та експлуатаційного аналізу можна констатувати, що поставлена мета роботи була повністю досягнута. У ході виконання роботи було успішно розроблено стабільну, масштабовану та

зручну інформаційну систему для автоматизації процесу формування навчального розкладу закладів вищої освіти.

Для досягнення цієї мети було успішно вирішено комплекс науково-технічних завдань. Спочатку було проведено системний аналіз предметної області та формалізовано вимоги до програмного продукту, після чого спроєктовано реляційну базу даних та налаштовано механізми доступу до неї шляхом впровадження транзакцій лише для читання. Окрім цього, розроблено серверну та клієнтську частини системи з використанням підходу Single Page Application та патерну Optimistic UI. Завершальним етапом стало інтегрування хмарних безсерверних обчислень AWS Lambda та Amazon S3 для перенесення ресурсомістких операцій з парсингу даних у хмару, що забезпечило високу швидкість обробки файлів у межах від 189 до 720 мс, а також створення й налаштування алгоритму виявлення академічних конфліктів у реальному часі.

Одержані результати повністю відповідають сучасним світовим тенденціям у галузі розробки програмного забезпечення, зокрема переходу до архітектури Cloud-Native та використання безсерверних обчислень. Це дозволило системі бути гнучкою і не залежати від потужності локального обладнання навчального закладу. Поряд із позитивними досягненнями, в ході дослідження та наскрізного тестування було зафіксовано і негативні результати та обмеження розробленого рішення. Зокрема, при критичній нестачі аудиторного фонду, коли кількість груп значно перевищує місткість закладу, базовий алгоритм не здатен самостійно вирішити всі колізії, залишаючи частину занять зі статусом нерозподілених, що вимагає обов'язкового ручного втручання адміністратора. Також було виявлено проблему гонки запитів у клієнтському інтерфейсі при надшвидкій взаємодії користувача з матрицею уподобань, що вказує на необхідність додаткового впровадження механізмів затримки відправки даних у майбутніх версіях програмного продукту.

Передбачувані області використання результатів роботи охоплюють безпосереднє впровадження розробленого рішення у навчальних відділах та деканатах університетів, академій та інститутів. Завдяки гнучкій структурі бази даних та уніфікованим інтерфейсам взаємодії, система може бути легко адаптована для використання в закладах фахової передвищої освіти, таких як коледжі, а також у спеціалізованих школах чи ліцеях із модульною системою організації навчального процесу.

Економічна значущість роботи полягає в тому, що впровадження системи дозволяє скоротити час, який витрачається на складання розкладу, з кількох днів до лічених хвилин, що забезпечує більш раціональне використання робочого часу персоналу. При цьому використання безсерверної архітектури AWS Lambda знижує витрати закладу на утримання власних високопродуктивних серверів, оскільки оплата хмарних ресурсів здійснюється виключно за мілісекунди фактичного часу виконання обчислень під час імпорту чи генерації даних.

Наукова значущість роботи полягає у розробці та програмній інтеграції гібридного евристично-еволюційного алгоритму із безсерверною інфраструктуро, а також у дослідженні ефективності архітектурного патерну асинхронної взаємодії (комбінації Short Polling та Serverless-функцій) для мінімізації обчислювального навантаження на центральний вебсервер при обробці великих масивів структурованих даних.

Соціальна значущість проєкту полягає в тому, що система автоматично враховує побажання викладачів, усуваючи непорозуміння з адміністрацією під час складання розкладу. Для студентів впровадження системи означає значне підвищення прозорості освітнього процесу, швидкий доступ до актуальної інформації про заняття з будь-яких пристроїв та мінімізацію вільних проміжків між парами впродовж навчального дня.

ВИСНОВКИ

У кваліфікаційній роботі було успішно вирішено актуальне науково-практичне завдання з автоматизації процесу формування навчального розкладу для закладів вищої освіти шляхом створення цілісного хмароорієнтованого програмного комплексу. Інформаційна система, що була спроєктована та реалізована, повністю задовольняє всі поставлені функціональні та нефункціональні вимоги, демонструє високу швидкість обробки даних, надійність рольової моделі доступу та відмовостійкість під час паралельної роботи користувачів. Головним результатом роботи є розробка та програмна інтеграція гібридного евристично-еволюційного алгоритму із безсерверною інфраструктурою, що дозволило перетворити процес складання розкладу зі складної рутинної роботи на гнучкий автоматизований процес із контролем колізій у реальному часі.

Важливим результатом розробки є практичне впровадження архітектури Cloud-Native на базі сервісів AWS Lambda та Amazon S3 для асинхронного парсингу великих масивів навчальних планів. Це інженерне рішення дозволило повністю ізолювати ресурсомісткі обчислювальні процеси від основного сервера Spring Boot, забезпечивши його стабільність та доступність навіть під час пікових навантажень на початку навчального семестру. Крім того, важливе практичне значення має реалізація інтерактивної матриці часових уподобань викладачів з використанням патерну Optimistic UI. Виявлена на етапі експлуатаційного аналізу проблема конкурентного обміну даними та узгодження станів під час синхронізації (Short Polling) була успішно ліквідована завдяки впровадженню механізмів оптимістичного блокування даних на рівні ORM-моделей бекенду та оптимізації транзакцій читання.

Як продовження та вдосконалення вирішення даної задачі у майбутньому, можна виділити кілька перспективних напрямків розвитку системи. Перш за все,

логічний алгоритм генерації розкладу можна модернізувати шляхом впровадження методів машинного навчання, що дозволить системі аналізувати історичні дані минулих семестрів та інтелектуально прогнозувати найбільш ефективний розподіл занять із мінімальною кількістю «вікон» для студентів. Також важливим кроком для покращення користувацького досвіду є створення нативного мобільного застосунку для платформ iOS та Android із системою push-сповіщень, яка б миттєво інформувала студентів про перенесення занять чи зміну аудиторій. Нарешті, архітектурний потенціал системи дозволяє реалізувати відкриті інтерфейси API для безшовної синхронізації даних з існуючими університетськими системами управління навчанням, такими як Moodle або Google Classroom, що перетворить розроблене програмне рішення на універсальний та центральний елемент цифрової екосистеми сучасного закладу вищої освіти.

Результати дослідження були опубліковані в матеріалах 7-ї Міжнародної науково-практичної конференції «Innovative Development in the Global Science» (May 26–28, 2026, Boston, USA) та прийняті до публікації у збірнику матеріалів 11-ї Міжнародної науково-практичної конференції «Scientific Community: Interdisciplinary Research» (June 6–8, 2026, Hamburg, Germany).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Островська Г. В., Мічківський С. М. Формування розкладу заліково-екзаменаційної сесії у вищих навчальних закладах // Матеріали наукової конференції професорсько-викладацького складу, наукових працівників і здобувачів наукового ступеня за підсумками науково-дослідної роботи за період 2017–2018 рр. : у 2 т. – Вінниця : Донецький національний університет імені Василя Стуса, 2019. – Т. 2. – С. 110–111. URL: https://jpvs.donnu.edu.ua/article/view/7243 (дата звернення: 14.05.2026).

2. Римар П. В., Войтко Б. С. Автоматизація процесу формування розкладу навчального процесу // Вісник Хмельницького національного університету. – 2021. – № 5. – С. 36–40. URL: https://journals.khnu.km.ua/vestnik/wp-content/uploads/2021/12/301-text_2021_5_t-3 6-40.pdf (дата звернення: 14.05.2026).

3. Шевченко С., Жданова Ю., Білоус М. Автоматизація розкладу занять як необхідна умова оптимізації функціонування закладу освіти // TECHNICAL, AGRICULTURAL AND MATHEMATICAL SCIENCES: SCIENTIFIC TRENDS, PROBLEMS AND WAYS OF THEIR DEVELOPMENT : collective monograph. – 2025. – С. 296–308. DOI: 10.46299/ISG.2025.MONO.TECH.2.5.6. URL: https://www.researchgate.net/publication/392702655 (дата звернення: 14.05.2026).

4. Гафіяк А. М., Бородіна О. О., Гусак В. Проблеми розробки клієнт-розкладу закладів вищої освіти // Економіка та суспільство. – 2018. – № 18. DOI: 10.32782/2524-0072/2018-18-148. URL: https://economyandsociety.in.ua/journals/18_ukr/148.pdf (дата звернення: 14.05.2026).

5. Мулява І. Я. Вирішення задачі автоматизованого формування розкладу навчального закладу за допомогою генетичних алгоритмів // Інтернаука. – 2018. – № 9. URL:

http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?I21DBN=LINK&P21DBN=UJRN&Z21ID=&S21REF=10&S21CNR=20&S21STN=1&S21FMT=ASP_meta&C21COM=S&2_S21P03=FILA=&2_S21STR=mnj_2018_9%281%29 20 (дата звернення: 14.05.2026).

6. Снитюк В. Є., Сіпко О. М. Технологія еволюційного формування розкладів у закладах вищої освіти : монографія. – Київ : ФОП Піча Ю. В., 2022. – 136 с. – ISBN 978-966-969-152-1. URL: https://www.researchgate.net/publication/366759465 (дата звернення: 14.05.2026).

7. Burke E. K., Petrovic S. Recent research directions in automated timetabling // European Journal of Operational Research. – 2002. – Vol. 140, No. 2. – P. 266–280. DOI: 10.1016/S0377-2217(02)00069-3. URL: https://www.sciencedirect.com/science/article/abs/pii/S0377221702000693 (дата звернення: 14.05.2026).

8. Кисіль В. В., Драч І. В., Кисіль Т. М. Модель задачі складання та оптимізації розкладу занять за умови задоволення об'єктивних та суб'єктивних вимог навчального закладу // Вчені записки ТНУ імені В. І. Вернадського. Серія: технічні науки. – 2019. – С. 65–70. DOI: 10.32838/2663-5941/2019.6-1/12. URL: https://tech.vernadskyjournals.in.ua/journals/2019/6_2019/part_1/14.pdf (дата звернення: 14.05.2026).

9. Rozklad.org: вебсайт. URL: https://rozklad.org/ (дата звернення: 14.05.2026).

10. ASC Timetables: вебсайт. URL: https://www.asctimetables.com/ (дата звернення: 14.05.2026).

11. OpenEduCat Timetable Management System: вебсайт. URL: https://openeducat.org/feature-timetable-management-system/ (дата звернення: 14.05.2026).

12. Kumar R. Event-Driven Architectures for Real-Time Data Processing: A Deep Dive into System Design and Optimization. – 2023. DOI:

10.5281/zenodo.15026990. URL: https://www.researchgate.net/publication/391633680 (дата звернення: 14.05.2026).

13. Sommerville I. Software Engineering. – 10th ed. – Boston : Pearson Education, 2016. – 816 p. URL: https://www.studyhalo.com/media/resources/resources/INF3705/Textbook/Software_Engineering_-_Ian_Sommerville.pdf (дата звернення: 14.05.2026).

14. Development of a Modified Simulated Annealing to School Timetabling Problem / O. Olufemi Ayodeji, E. O. Omidiora, S. O. Olabiyisi, A. Aluko // International Journal of Applied Information Systems. – 2015. – Vol. 8, No. 2. – P. 16–24. DOI: 10.5120/ijais14-451277.

15. Chunlei T., Liu Y., Zheng L. Hybrid Element Heuristic Algorithm Optimizing Neural Network-Based Educational Courses // Wireless Communications and Mobile Computing. – 2021. – Article ID 9581793. DOI: 10.1155/2021/9581793.

16. List-Based Simulated Annealing Algorithm for Traveling Salesman Problem / S.-H. Zhan, S. Lin, Z.-J. Zhang, Y. Zhong // Computational Intelligence and Neuroscience. – 2016. DOI: 10.1155/2016/1712630.

17. Dandashi A., Al-Mouhamed M. Graph Coloring for Class Scheduling // 2010 International Conference on Advances in Computational Tools for Engineering Applications. – 2010. – P. 1–4. DOI: 10.1109/AICCSA.2010.5586963.

18. Malviya A., Agrawal B., Kumar S. Graph Coloring Algorithm for Course Time Table Scheduling Problem // ShodhKosh: Journal of Visual and Performing Arts. – 2024. – Vol. 5, No. 3. – P. 452–460. DOI: 10.29121/shodhkosh.v5.i3.2024.2182.

19. Vyas M. Exam Scheduling Using Graph Coloring // Journal of Information Systems Engineering and Management. – 2025. – Vol. 10. – P. 10–15. DOI: 10.52783/jisem.v10i24s.3868.

20. Muller T. Some Novel Approaches to Lecture Timetabling // Proceedings of the 4th Workshop of Constraint Programming for Decision and Control (CPDC 2002). – Gliwice, 2002. – P. 352–369.

21. Kiyohara M., Ishihata M. A Constraint Programming Approach to Fair High School Course Scheduling. – 2024. DOI: 10.48550/arXiv.2408.12032. URL: https://arxiv.org/abs/2408.12032 (дата звернення: 14.05.2026).

22. Rudová H., Müller T., Murray K. Complex University Course Timetabling // Journal of Scheduling. – 2011. – Vol. 14. – P. 187–207. DOI: 10.1007/s10951-010-0171-3.

23. An Artificial Intelligence Approach to Course Timetabling / L. F. Lai, C. C. Wu, N. L. Hsueh, L. T. Huang, S. F. Hwang // International Journal on Artificial Intelligence Tools. – 2008. – Vol. 17, No. 1. – P. 223–240. DOI: 10.1142/S0218213008003868.

24. Гайтан О. М. Автоматизація розкладу навчальних курсів університету // Вчені записки Таврійського національного університету імені В. І. Вернадського. Серія: Технічні науки. – 2020. – Т. 31 (70), № 2. – С. 58–66. DOI: 10.32838/2663-5941/2020.2-1/09.

25. University Course Timetabling Model Using Ant Colony Optimization Algorithm Approach / M. Mazlan, M. Makhtar, A. Khairi, M. A. Mohamed // Indonesian Journal of Electrical Engineering and Computer Science. – 2019. – Vol. 13, No. 1. – P. 72–76. DOI: 10.11591/ijeecs.v13.i1.pp72-76.

26. Mahmud A. Highly Constrained University Class Scheduling Using Ant Colony Optimization // International Journal of Computer Science & Information Technology. – 2021. – Vol. 13, No. 1. URL: https://ssrn.com/abstract=3801441 (дата звернення: 14.05.2026).

27. Herianto T. Implementation of the Ant Colony System Algorithm in the Lecture Scheduling Process // Instal: Jurnal Komputer. – 2021. – Vol. 12, No. 2. – P. 50–56. DOI: 10.54209/jurnalkomputer.v12i02.23.

28. Budhi G., Gunadi K., Wibowo D. Genetic Algorithm for Scheduling Courses // Information and Communication Technology. – 2015. – P. 51–63. DOI: 10.1007/978-3-662-46742-8_5.

29. Subang K., Agoylo Jr J. Optimizing Course Scheduling with Genetic Algorithms: A Dynamic Approach // SAR Journal – Science and Research. – 2024. – Vol. 7, No. 4. – P. 296–302. DOI: 10.18421/SAR74-02.

30. Design and Application of an Improved Genetic Algorithm to a Class Scheduling System / X. Chen, X.-G. Yue, R. Y. M. Li, A. Zhumadillayeva, R. Liu // International Journal of Emerging Technologies in Learning. – 2021. – Vol. 16, No. 1. – P. 44–59. DOI: 10.3991/ijet.v16i01.18225.

31. Camungao R. A New Class Scheduling Approach Using Genetic Algorithm Methods for Isabela State University // Journal of Information Systems Engineering and Management. – 2025. – Vol. 10. – P. 593–599. DOI: 10.52783/jisem.v10i49s.9909.

32. Isa Martinus A., Hartono M. B., Wicaksono F. Implementation of the Constraint Satisfaction Problems Method in Genetic Algorithms for Course Scheduling Systems // Jurnal Improsci. – 2023. – Vol. 1, No. 3. – P. 140–147. DOI: 10.62885/improsci.v1i3.149.

33. Пекар В. В. Реалізація моделі RBAC та Permissions у веб-орієнтованій системі управління навчальним процесом // Proceedings of the International Scientific and Practical Conference InterConf. – Boston, 2026. DOI: 10.51582/interconf.2026.296. URL: https://archive.interconf.center/index.php/conference-proceeding/issue/view/26-28.05.2026/307 (дата звернення: 14.05.2026).

34. Пекар В. В. Організація безсерверної обробки даних на базі архітектурного шаблону Event-Driven у хмарних сервісах AWS // Матеріали міжнародної науково-практичної конференції Scientific Community: Interdisciplinary Research. – Hamburg, Germany, 2026. – Прийнята до друку.

35. Коннолліі Т., Бегг К. Бази даних: проєктування, реалізація та супровід. Теорія і практика. – Київ : Вільямс, 2017.

36. Date C. J. An Introduction to Database Systems. – 8th ed. – Pearson, 2003. – 1008 p. URL: https://www.mbit.edu.in/wp-content/uploads/2020/05/An-Introduction-to-Database-S ystems-8e-By-C-J-Date-CodeBlah.com_.pdf (дата звернення: 14.05.2026).

37. Bauer C., King G. Java Persistence with Hibernate. – Manning Publications, 2006. – 880 p. URL: https://theswissbay.ch/pdf/Gentoomen%20Library/Programming/Java/Java%20Persistence%20with%20Hibernate%20-%20ISBN%201932394885.pdf (дата звернення: 14.05.2026).

38. ESBMT: Enabling Multi-Tenancy in Enterprise Service Buses / S. Strauch, V. Andrikopoulos, F. Leymann, D. Muhler // Proceedings of CloudCom 2012. – IEEE Computer Society Press, 2012. – P. 456–463.

39. Chong F., Carraro G. Architecture Strategies for Catching the Long Tail. – Redmond : Microsoft Corporation, 2006. – 44 p.

ДОДАТОК А. Приклад успішного виконання тестового сценарію

У цьому додатку наведено екранні форми розробленого програмного модуля, які ілюструють повний цикл автоматизованого формування розкладу: від початкового налаштування та імпорту навчального плану до генерації кінцевої сітки занять (рис. 4.1 - 4.8)

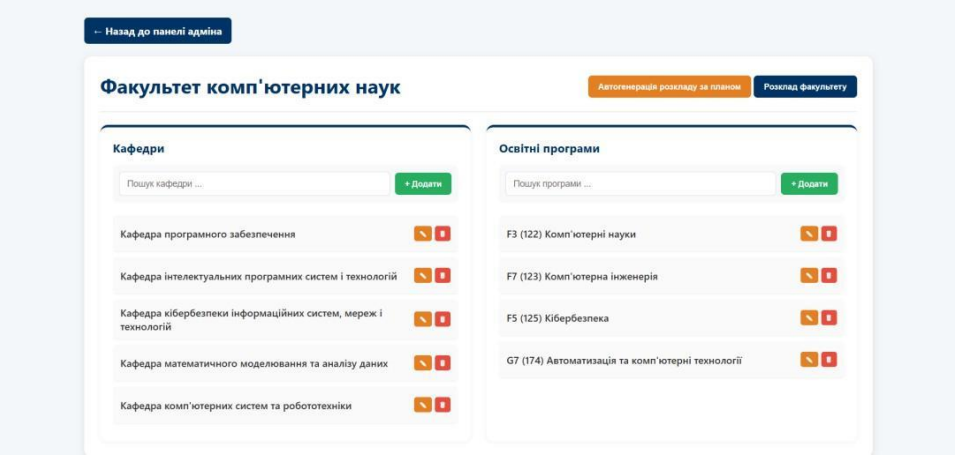


Рисунок 4.1 - Інтерфейс адміністрування структури факультету та ініціалізації процесу генерації розкладу

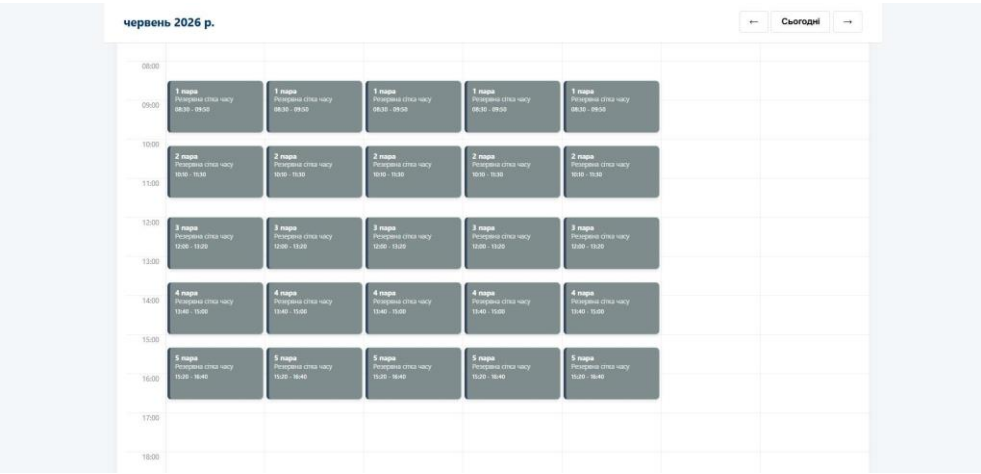


Рисунок 4.2 - Базова сітка часових слотів факультету, підготовлена адміністратором для роботи алгоритму

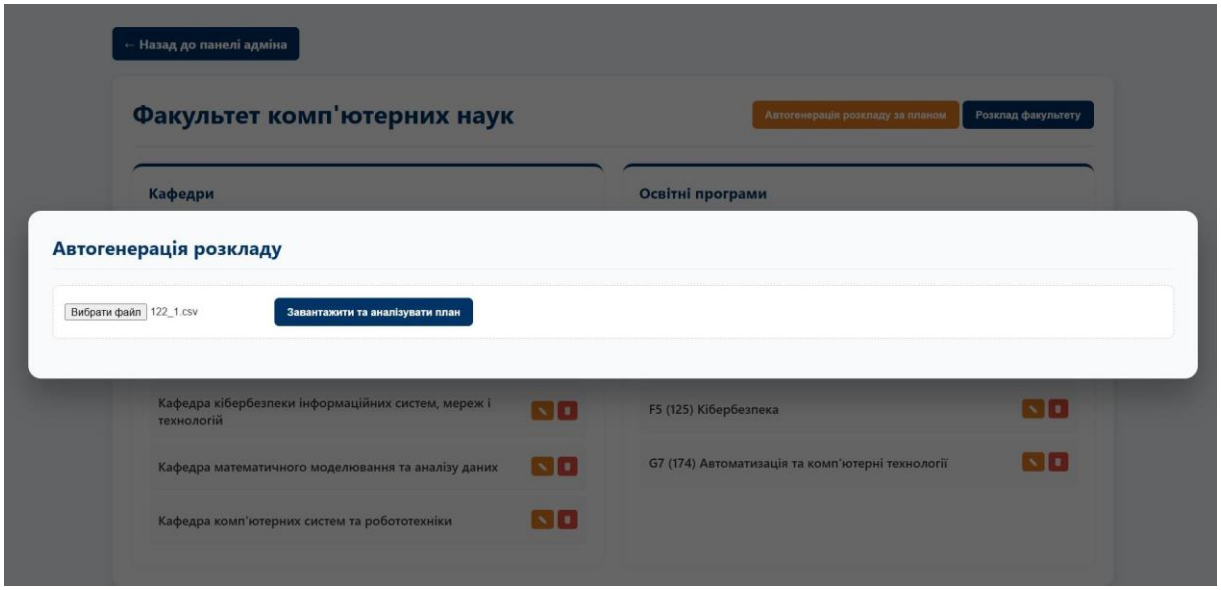


Рисунок 4.3 - Інтерфейс модуля імпорту перед завантаженням файлу навчального плану

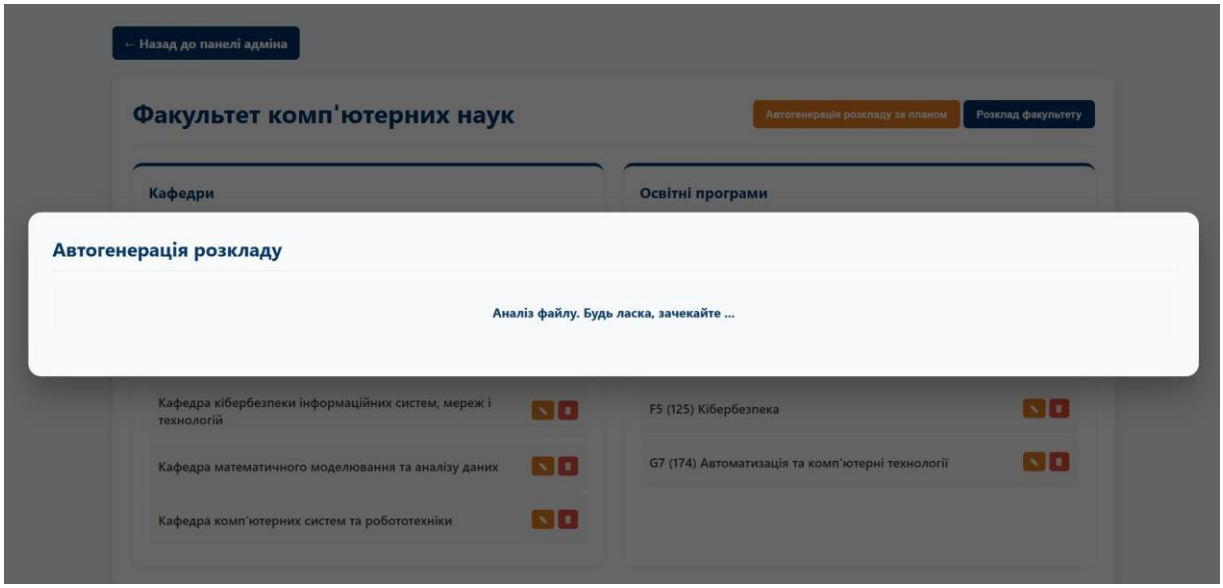


Рисунок 4.4 - Процес асинхронного аналізу даних у хмарному середовищі

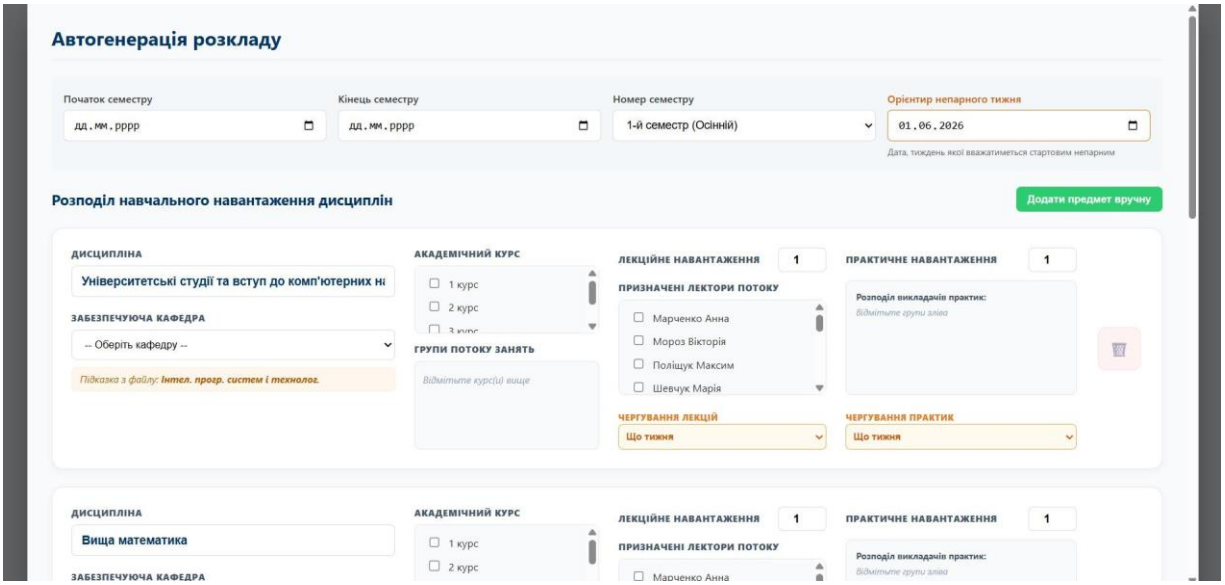


Рисунок 4.5 - Результат автоматичного розпізнавання та імпорту переліку дисциплін

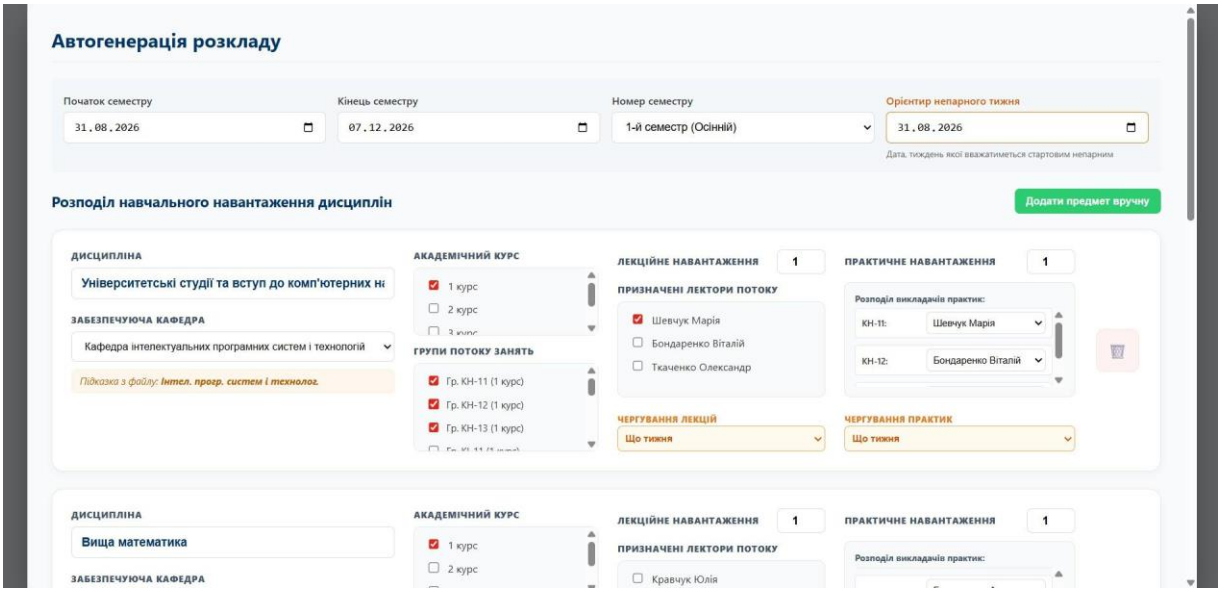


Рисунок 4.6 - Приклад налаштування параметрів навчальних дисциплін перед запуском алгоритму генерації

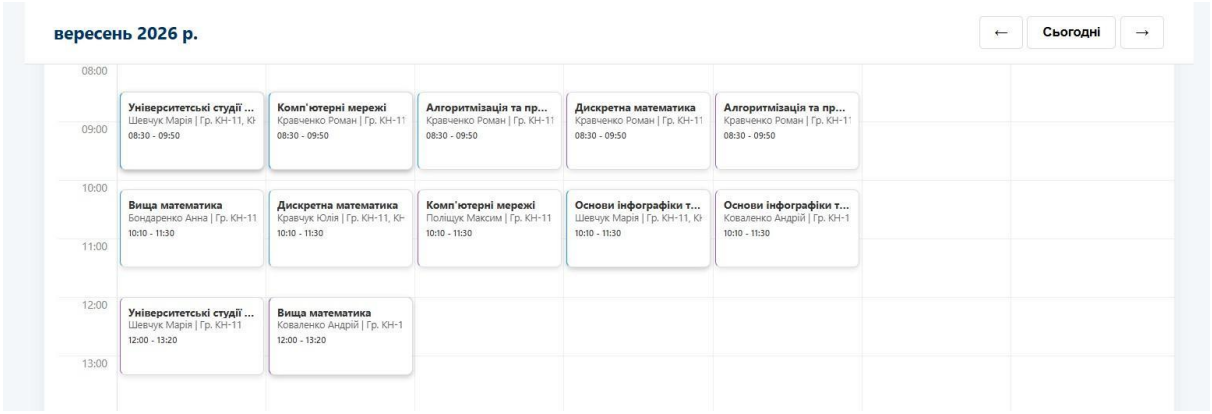
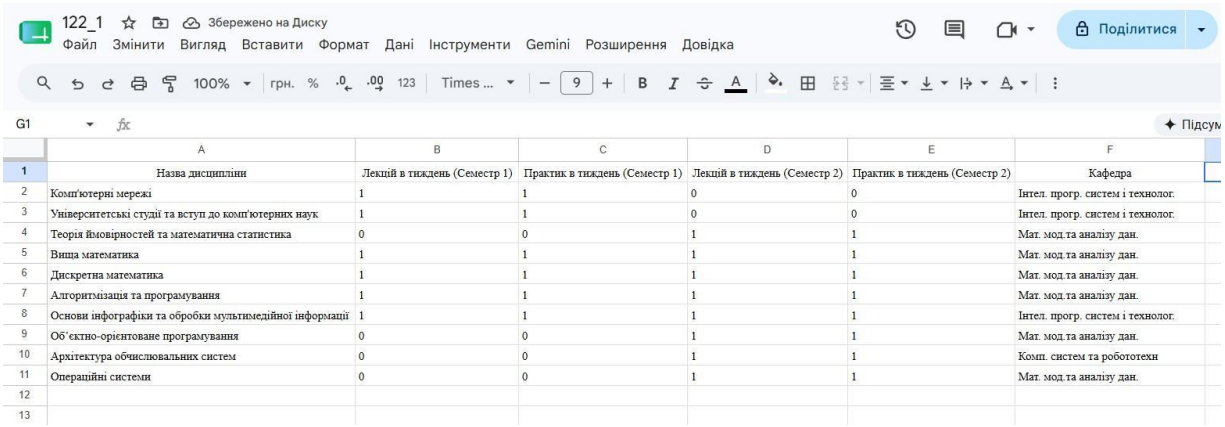


Рисунок 4.7 - Результат автоматичного формування розкладу для обраної академічної групи



	A	B	C	D	E	F
1	Назва дисципліни	Лекцій в тиждень (Семестр 1)	Практик в тиждень (Семестр 1)	Лекцій в тиждень (Семестр 2)	Практик в тиждень (Семестр 2)	Кафедра
2	Комп'ютерні мережі	1	1	0	0	Інтел. прогр. систем і технолог.
3	Університетські студії та вступ до комп'ютерних наук	1	1	0	0	Інтел. прогр. систем і технолог.
4	Теорія ймовірностей та математична статистика	0	0	1	1	Мат. мод.та аналізу дан.
5	Вища математика	1	1	1	1	Мат. мод.та аналізу дан.
6	Дискретна математика	1	1	1	1	Мат. мод.та аналізу дан.
7	Алгоритмізація та програмування	1	1	1	1	Мат. мод.та аналізу дан.
8	Основи інфографіки та обробки мультимедійної інформації	1	1	1	1	Інтел. прогр. систем і технолог.
9	Об'єктно-орієнтоване програмування	0	0	1	1	Мат. мод.та аналізу дан.
10	Архітектура обчислювальних систем	0	0	1	1	Комп. систем та робототехн
11	Операційні системи	0	0	1	1	Мат. мод.та аналізу дан.
12						
13						

Рисунок 4.8 - Фрагмент структури вхідного файлу навчального плану