ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА

(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій

(повне найменування факультету)

Кафедра інформаційних технологій

(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «Магістр»

«Розробка системи підтримки прийняття рішень для планування навчального процесу на основі методів штучного інтелекту»

(тема кваліфікаційної роботи українською мовою)

«Development of a decision support system for planning the educational process based on artificial intelligence methods»

(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач денної форми навчання
спеціальності 122 Комп'ютерні науки
(код, назва спеціальності)
Освітня програма Комп'ютерні науки
(назва)
КАЗАКОВ Павло Олексійович
(прізвище, ім'я, по-батькові здобувача)

Керівник Фразе-Фразенко О.О. к.т.н., доцент _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент Домаскін О.М. к.т.н, Начальник ЦІТ ОНЕУ
(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
Інформаційних технологій
№ ____від _____2025 р.

Завідувачка кафедри

_____ КАЗАКОВА Надія
(підпис) (прізвище,ім'я)

Захищено на засіданні ЕК № _____
протокол № ___від _____2025 р.

Оцінка _____/ _____/ _____
(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК

_____ СОКОЛОВ Артем
(підпис) (прізвище,ім'я)

Одеса 2025

АНОТАЦІЯ

У кваліфікаційній роботі розробляється тема «Розробка системи прийняття рішень для планування навчального процесу на основі методів штучного інтелекту».

Мета роботи – створення та дослідження системи, що автоматизує процес формування навчального розкладу шляхом використання алгоритмів штучного інтелекту, зокрема генетичного алгоритму, а також моделювання обмежень, властивих освітньому процесу. У межах дослідження розглядаються методи оптимізації, принципи роботи еволюційних алгоритмів, структура даних для представлення елементів розкладу, а також підхід до виділення оптимізаційного ядра в окремий сервіс на основі Python.

У результаті реалізації створено веб-орієнтовану систему, яка дозволяє автоматично формувати навчальний розклад із урахуванням жорстких і м’яких обмежень. Оптимізаційне ядро реалізовано як окремий модуль штучного інтелекту, що забезпечує гнучкість конфігурації та можливість масштабування. Система підтримує візуалізацію сформованих розкладів, контроль конфліктів, редагування адміністраторами та інтеграцію з базою даних. Додатково проведено порівняльний аналіз ефективності алгоритмічного підходу та оцінку якості отриманих рішень.

ABSTRACT

The qualification work explores the topic “Development of a decision-making system for planning the educational process based on artificial intelligence methods”.

The aim of the work is to design and investigate a system that automates the construction of academic timetables using artificial intelligence techniques, particularly a genetic algorithm, together with a constraint-based model reflecting the specifics of the educational environment. The research examines optimization methods, the operational principles of evolutionary algorithms, data structures for timetable representation, and the approach of implementing a standalone Python-based optimization service.

As a result of the implementation, a web-based system was developed that automatically generates academic timetables while satisfying both hard and soft constraints. The optimization core is implemented as an independent AI module, enabling flexible configuration and scalability. The system supports timetable visualization, conflict detection, administrative editing, and integration with a database. Additionally, a comparative analysis of the algorithmic approach and an evaluation of the quality of the generated timetables were conducted.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ,УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

ЗВО – заклад вищої освіти

ГА – генетичний алгоритм

СКБД – систему керування базами даних

СППР – системи підтримки прийняття рішень

LMS – Learning Management Systems

UI – Інтерфейс користувача

UX – користувацький досвід

MVP – Minimal Viable Product

CRUD – Create, Read, Update, Delete

SSR – Server-Side Rendering

RBAC – Role-Based Access Control

MPA – Multi-Page Application

HTML – мова гіпертекстової розмітки

CSS – каскадні таблиці стилів

SSR – Server-Side Rendering

БД – база даних

CTE – Common Table Expressions

СУБД – система управління базами даних

PHP – препроцесор гіпертексту

SQL – мова структурованих запитів

ВСТУП

Сучасні освітні заклади працюють у динамічних умовах, що вимагає підвищення ефективності управлінських процесів шляхом автоматизації та оптимізації. Одним із найбільш ресурсоємних організаційних завдань є складання навчального розкладу, яке потребує врахування великої кількості обмежень: доступності викладачів, наповнюваності груп, місткості та типів аудиторій, навчальних планів, санітарно-гігієнічних вимог та педагогічних рекомендацій щодо розподілу навантаження. Складність цього процесу зростає пропорційно кількості груп, дисциплін та можливих комбінацій занять. Тому актуальною є розробка інформаційних систем, здатних забезпечити автоматизоване формування коректного та збалансованого розкладу.

Класичні підходи до складання розкладу часто ґрунтуються на жадібних алгоритмах або ручному коригуванні, що не дозволяє ефективно оптимізувати розклад за багатьма критеріями одночасно. Використання алгоритмів штучного інтелекту та еволюційних методів відкриває можливості для формування оптимізованих рішень у задачах, де повний перебір є обчислювально неприпустимим. Зокрема, генетичні алгоритми демонструють високу ефективність під час пошуку рішень у задачах із великою кількістю взаємопов’язаних обмежень і багатокритеріальною функцією якості.

У даній роботі розглядається підхід до автоматизації процесу складання навчального розкладу на основі генетичного алгоритму. Система дозволяє формувати розклад із дотриманням жорстких обмежень і можливістю мінімізувати порушення м’яких критеріїв, таких як кількість «вікон», рівномірність навантаження або пріоритетність певних часових інтервалів.

Метою роботи є побудова та дослідження моделі автоматизованого формування навчального розкладу, що ґрунтується на використанні генетичного алгоритму для оптимізації розміщення навчальних занять за умов наявності множини жорстких та м’яких обмежень, а також розробка

програмної реалізації цієї моделі у вигляді веб-системи. Для досягнення поставленої мети необхідно розв’язати такі завдання:

– проаналізувати предметну область та існуючі підходи до автоматизації складання розкладу;

– сформувати математичну постановку задачі із визначенням жорстких та м’яких обмежень;

– обґрунтувати вибір методів оптимізації та технологічного стеку;

– спроєктувати архітектуру системи, включно із моделлю даних та взаємодією між компонентами;

– реалізувати веб-додаток та модуль генетичного алгоритму;

– провести експериментальні дослідження й тестування роботи системи.

Об’єктом дослідження є процес автоматизованого формування навчального розкладу у закладах вищої освіти.

Предметом дослідження є методи оптимізації та технології, що застосовуються для побудови та реалізації системи складання розкладу, зокрема генетичний алгоритм та інструменти розробки веб-додатків.

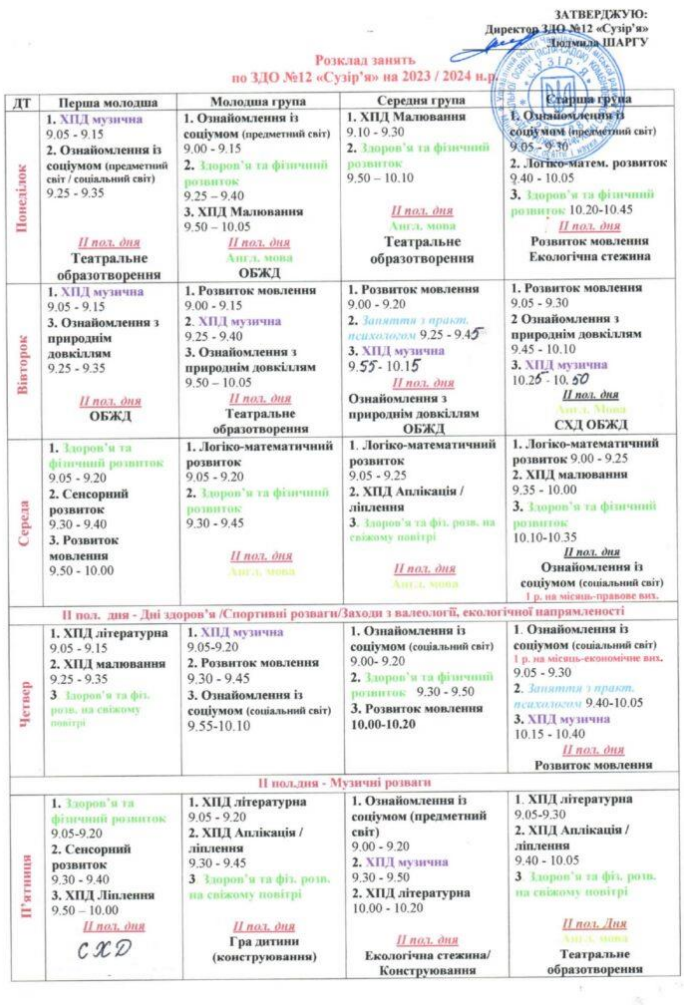
Практична цінність роботи полягає у створенні веб-системи, яка забезпечує формування навчального розкладу з урахуванням індивідуальних вимог освітнього закладу та дає змогу автоматизувати трудомісткий процес, зменшити кількість помилок та підвищити якість розкладу.

Робота містить 83 сторінки, 8 таблиць, 26 рисунків, 23 джерела посилань.

1 АНАЛІЗ МЕТОДІВ АВТОМАТИЗАЦІЇ СКЛАДАННЯ РОЗКЛАДУ

1.1 Предметна область задачі та огляд існуючих технологій автоматизації складання розкладів

Складання оптимального навчального розкладу в закладах вищої освіти є класичною складною комбінаторною задачею, яка входить до класу NP-складних проблем. Її вирішення вимагає одночасної оптимізації за десятками критеріїв якості та дотримання сотень обмежень різного типу, що робить неможливим повний перебір усіх варіантів для реальних масштабів навчального закладу (рис. 1.1). Для створення ефективної інформаційної системи підтримки прийняття рішень необхідно провести глибокий аналіз існуючих підходів, технологій та програмних рішень у цій галузі.



ЗАТВЕРДЖУЮ:
Директор ЗДО №12 «Сузір'я»
Людмила ШАРГУ

Розклад занять
по ЗДО №12 «Сузір'я» на 2023 / 2024 н.р.

ДТ	Перша молодша	Молодша група	Середня група	Старша група
Понеділок	1. ХПД музична 9.05 - 9.15 2. Ознайомлення із соціумом (предметний світ / соціальний світ) 9.25 - 9.35 II пол. дня Театральне образотворення	1. Ознайомлення із соціумом (предметний світ) 9.00 - 9.15 2. Здоров'я та фізичний розвиток 9.25 – 9.40 3. ХПД Малювання 9.50 – 10.05 II пол. дня Англ. мова ОБЖД	1. ХПД Малювання 9.10 - 9.30 2. Здоров'я та фізичний розвиток 9.50 – 10.10 II пол. дня Англ. мова Театральне образотворення	1. Ознайомлення із соціумом (предметний світ) 9.05 - 9.30 2. Логіко-матем. розвиток 9.40 - 10.05 3. Здоров'я та фізичний розвиток 10.20-10.45 II пол. дня Розвиток мовлення Екологічна стежина
Вівторок	1. ХПД музична 9.05 - 9.15 3. Ознайомлення з природнім довкіллям 9.25 - 9.35 II пол. дня ОБЖД	1. Розвиток мовлення 9.00 - 9.15 2. ХПД музична 9.25 - 9.40 3. Ознайомлення з природнім довкіллям 9.50 – 10.05 II пол. дня Театральне образотворення	1. Розвиток мовлення 9.00 - 9.20 2. Заняття з практ. психологом 9.25 - 9.45 3. ХПД музична 9.55 - 10.15 II пол. дня Ознайомлення з природнім довкіллям ОБЖД	1. Розвиток мовлення 9.05 - 9.30 2 Ознайомлення з природнім довкіллям 9.45 - 10.10 3. ХПД музична 10.25 - 10.50 II пол. дня Англ. мова СХД ОБЖД
Середа	1. Здоров'я та фізичний розвиток 9.05 - 9.20 2. Сенсорний розвиток 9.30 - 9.40 3. Розвиток мовлення 9.50 - 10.00	1. Логіко-математичний розвиток 9.05 - 9.20 2. Здоров'я та фізичний розвиток 9.30 - 9.45 II пол. дня Англ. мова	1. Логіко-математичний розвиток 9.05 - 9.25 2. ХПД Аплікація / ліплення 3. Здоров'я та фіз. розв. на свіжому повітрі II пол. дня Англ. мова	1. Логіко-математичний розвиток 9.00 - 9.25 2. ХПД малювання 9.35 - 10.00 3. Здоров'я та фізичний розвиток 10.10-10.35 II пол. дня Ознайомлення із соціумом (соціальний світ) 1 р. на місяць-правове вих.
II пол. дня - Дні здоров'я /Спортивні розваги/Заходи з валеології, екологічної напрямленості				
Четвер	1. ХПД літературна 9.05 - 9.15 2. ХПД малювання 9.25 - 9.35 3. Здоров'я та фіз. розв. на свіжому повітрі	1. ХПД музична 9.05-9.20 2. Розвиток мовлення 9.30 - 9.45 3. Ознайомлення із соціумом (соціальний світ) 9.55-10.10	1. Ознайомлення із соціумом (соціальний світ) 9.00- 9.20 2. Здоров'я та фізичний розвиток 9.30 - 9.50 3. Розвиток мовлення 10.00-10.20	1. Ознайомлення із соціумом (соціальний світ) 1 р. на місяць-економічне вих. 9.05 - 9.30 2. Заняття з практ. психологом 9.40-10.05 3. ХПД музична 10.15 - 10.40 II пол. дня Розвиток мовлення
II полдня - Музичні розваги				
П'ятниця	1. Здоров'я та фізичний розвиток 9.05-9.20 2. Сенсорний розвиток 9.30 - 9.40 3. ХПД Ліплення 9.50 – 10.00 II пол. дня СХД	1. ХПД літературна 9.05 - 9.20 2. ХПД Аплікація / ліплення 9.30 - 9.45 3 Здоров'я та фіз. розв. на свіжому повітрі II пол. дня Гра дитини (конструювання)	1. Ознайомлення із соціумом (предметний світ) 9.00 - 9.20 2. ХПД музична 9.30 - 9.50 2. ХПД літературна 10.00 - 10.20 II пол. дня Екологічна стежина/ Конструювання	1. ХПД літературна 9.05-9.30 2. ХПД Аплікація / ліплення 9.40 - 10.05 3 Здоров'я та фіз. розв. на свіжому повітрі II пол. Дня Англ. мова Театральне образотворення

Рисунок 1.1 – Приклад проблемного розкладу

Сучасні системи автоматизації складання розкладу можна класифікувати за кількома ознаками: архітектурою, функціональними можливостями та використанням передових методів оптимізації. Узагальнено їх можна поділити на три широкі категорії:

Глобальні освітні платформи та LMS. До цієї категорії належать такі популярні рішення, як Google Classroom, Microsoft Teams та Moodle. Вони надають широкий набір інструментів для організації навчального процесу, серед яких є базові функції планування занять. Їхня основна сила полягає в інтеграції з іншими сервісами, зручному користувацькому інтерфейсі та підтримці багатьох ролей[20]. Однак, вони орієнтовані на ручне управління розкладом або просте відображення готових даних. Вбудовані механізми для автоматизованої оптимізації, що враховують сотні обмежень, в них практично відсутні. Наприклад, у Moodle розклад створюється шляхом ручного внесення подій або імпорту, але система не може автоматично розподілити заняття по аудиторіях та викладачах, уникнувши конфліктів.

Локальні інформаційні системи навчальних закладів. Це спеціалізовані системи, розроблені для конкретних університетів або коледжів (наприклад, внутрішні портали українських ЗВО). Вони зазвичай включають модулі управління студентами, викладачами, аудиторіями та навчальними планами. Такі системи дозволяють зберігати всю необхідну інформацію в централізованій базі даних і часто мають модуль для напівавтоматичного складання розкладу, де розпорядник працює в режимі діалогу з системою. Головними їхніми недоліками є відсутність складних алгоритмів оптимізації[23], низька гнучкість та масштабованість, а також складність інтеграції з сучасними інструментами аналітики.

Спеціалізовані системи для автоматизованого складання розкладу. Цей клас систем є найбільш релевантним для даного дослідження. Вони спеціалізуються виключно на вирішенні задачі оптимізації розкладу. До них належать такі комерційні та академічні рішення, як Scientia University Timetabler, FET[1] та UniTime[2]. Подібні рішення дозволяють автоматично

генерувати кілька варіантів розкладу та обирати оптимальний на основі заданих критеріїв (табл. 1.1).

Таблиця 1.1 – Порівняльна характеристика існуючих систем

Система	Тип	Функції розкладу	Алгоритми оптимізації	Переваги	Недоліки
Moodle	LMS (глобальна/локальна)	Вручну / Імпорт	Відсутні	Модульність, широка популярність, підтримка багатьох ролей, відкритий код	Відсутність інтелектуальної оптимізації, орієнтація на відображення, а не на генерацію
Google Classroom	Онлайн-платформа	Вручну	Відсутні	Простота, безперервна інтеграція з екосистемою Google	Функціонал розкладу є додатковим, не підходить для складної оптимізації
Локальні портали ЗВО	Локальна IS	Напівавтоматичні	Елементарні евристики	Інтеграція з внутрішніми даними, облік специфіки закладу	Низька масштабованість, закритість архітектури, qвідсутність сучасних алгоритмів ШІ
FET	Спеціалізована	Автоматичні	Евристичний	Відкритий код, потужний	Складний для адаптації

Система	Тип	Функції розкладу	Алгоритми оптимізації	Переваги	Недоліки
			алгоритм	функціонал, підтримка складних обмежень	інтерфейс, орієнтований на технічних користувачів
UniTime	Спеціалізована	Автоматичні	Генетичний алгоритм, імітація відпалу	Багатокритеріальна оптимізація, веб-інтерфейс, підтримка розподілу іспитів	Складність розгортання та налаштування, вимоги до потужності сервера

Проведений аналіз показав, що хоча ринок і пропонує різноманітні рішення, більшість платформ або орієнтовані на ручне управління, або є надто складними для адаптації в умовах середнього українського ЗВО. Ключовим недоліком наявних систем є відсутність глибокої інтеграції методів штучного інтелекту для багатокритеріальної оптимізації, що не дозволяє автоматично знаходити дійсно якісні та збалансовані рішення в умовах обмеженого часу[22].

Виходячи з цього, розроблювана система має поєднати в собі найкращі практики:

Архітектура: використання трирівневої клієнт-серверної архітектури (рис. 1.2) для забезпечення модульності, масштабованості та легкості супроводу.

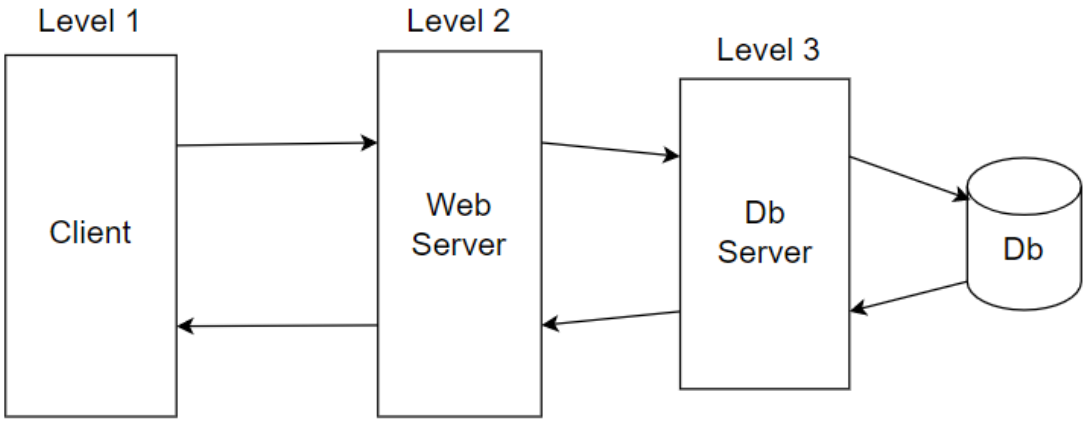


Рисунок 1.2 – Схема трирівневої клієнт-серверної архітектури

Оптимізаційний рушій: інтеграція потужних алгоритмів штучного інтелекту, зокрема генетичних алгоритмів або методів метаевристики, для глобального пошуку оптимальних рішень.

Адаптивність: забезпечення можливості легкої конфігурації критеріїв якості та системи обмежень без змін у вихідному коді, що дозволить системі адаптуватися до специфіки різних навчальних закладів.

1.2 Формальна постановка задачі оптимізації навчального розкладу

Основні компоненти моделі

Задача складання навчального розкладу формулюється як багатокритеріальна оптимізаційна задача з обмеженнями[17], яка потребує знаходження оптимального призначення навчальних занять до доступних ресурсів: часу, аудиторій та викладачів. Основні компоненти моделі включають такі сутності:

Множина навчальних груп: G={g1,g2,...,gm}$G=\{g1,g2,...,gm\}$

Множина викладачів: T={t1,t2,...,tn}$T=\{t1,t2,...,tn\}$

Множина навчальних дисциплін: S={s1,s2,...,sk}$S=\{s1,s2,...,sk\}$

Множина аудиторій: R={r1,r2,...,rl}$R=\{r1,r2,...,rl\}$

Множина часових слотів: P={p1,p2,...,pd}$P=\{p1,p2,...,pd\}$, де кожен слот

р*і*pi визначається парою (день тижня, номер пари)

Кожна сутність має специфічні характеристики: аудиторії мають місткість Cap(rj)*Cap*(*rj*) та тип Type(rj)*Type*(*rj*), викладачі мають графік доступності Avail(ti,pj)*Avail*(*ti*,*pj*), а навчальні дисципліни мають вимоги до типу аудиторії Req(si)*Req*(*si*) [3].

Система обмежень

Жорсткі обмеження є обов'язковими для виконання і визначають допустимість розкладу. Порушення будь-якого з цих обмежень робить розклад неприйнятним.

Таблиця 1.2 – Система жорстких обмежень

№	Тип обмеження	Математичний запис	Пояснення
1	Унікальність викладача	$\sum\sum x(c,r,p) \leq 1 \ \forall t \in T, \forall p \in P$	Викладач не може проводити більше одного заняття одночасно
2	Унікальність групи	$\sum\sum x(c,r,p) \leq 1 \ \forall g \in G, \forall p \in P$	Група не може мати більше одного заняття одночасно
3	Унікальність аудиторії	$\sum x(c,r,p) \leq 1 \ \forall r \in R, \forall p \in P$	В одній аудиторії не може бути більше одного заняття одночасно
4	Відповідність типу аудиторії	$x(c,r,p) = 1 \Rightarrow Type(r) = Req(c.s)$	Заняття повинно проводитися в аудиторії відповідного типу
5	Місткість аудиторії	$x(c,r,p) = 1 \Rightarrow Cap(r) \geq Size(c.g)$	Аудиторія повинна вміщати всіх студентів групи
6	Доступність викладача	$Avail(t_i,p_j) = 1$	Заняття призначається тільки у час, коли викладач доступний

М'які обмеження визначають якість розкладу і формалізуються у вигляді штрафних функцій, які необхідно мінімізувати. Відповідно до досліджень, цільові функції є недиференційованими і можуть бути поліекстремальними, що ускладнює пошук глобального оптимуму [3].

Критерії якості розкладу:

1) Мінімізація «вікон»:

Мінімізація порожніх часових проміжків між заняттями у викладачів

Мінімізація порожніх проміжків між заняттями у навчальних груп

2) Баланс навантаження:

Рівномірний розподіл заняття протягом тижня для кожної групи

Рівномірний розподіл навантаження для викладачів

3) Врахування піків працездатності згідно дослідженнь, біоритмічний оптимум розумової працездатності припадає на 10-12 години, коли відзначається найбільша ефективність засвоєння матеріалу. Тому більш складні предмети рекомендовано ставити на 2-4-ий уроки.

4) Ефективність використання ресурсів:

Мінімізація простою аудиторій

Оптимальне використання спеціалізованих аудиторій

Цільова функція

Цільова функція представляє собою зважену суму штрафів за порушення м'яких обмежень:

$F(X) = \min(w_1 \cdot F_1 + w_2 \cdot F_2 + w_3 \cdot F_3 + w_4 \cdot F_4 + w_5 \cdot F_5)$, де:

$F(X)$ – цільова функція, що мінімізується

$w_1 ... w_5$ – вагові коефіцієнти важливості критеріїв

$F_1 ... F_5$ – штрафні функції за порушення м'яких обмежень

Приклади штрафних функцій:

$F_1 = \sum\sum$ Windows(t,d) – штраф за «вікна» у викладачів

$F_3 = \sum$ [max Lessons(g,d) - min Lessons(g,d)] – штраф за нерівномірність навантаження

Практичні аспекти реалізації

Для ефективного розв'язання задачі рекомендовано:

Етапне розміщення занять (спочатку – сумісники, фізкультура тощо)

Врахування санітарно-гігієнічних вимог

Оптимальне використання спеціалізованих аудиторій

Врахування біоритмічних особливостей працездатності

Візуалізація: для наочності рекомендується додати схему, що відображає взаємозв'язок між основними компонентами моделі та обмеженнями[6].

Таким чином, задача формулюється як пошук оптимального призначення занять до аудиторій та часових слотів, яке задовольняє всім жорстким обмеженням та мінімізує сумарний штраф за порушення м'яких обмежень.

Практичні аспекти формулювання задачі

Під час формалізації задачі необхідно враховувати практичні рекомендації щодо складання розкладів:

Етапність процесу: рекомендується спочатку розмістити заняття вчителів-сумісників та адміністрації, потім уроки фізичної культури, трудового навчання, образотворчого мистецтва, музичного мистецтва, а вже потім - інших вчителів-предметників.

Врахування специфіки предметів: деякі предмети бажано розміщувати без «вікон» і по можливості в ті дні, коли в розкладі є уроки фізичної культури, що дозволяє раціональніше використовувати навчальні кабінети.

Санітарно-гігієнічні вимоги: необхідно враховувати вимоги державних санітарних правил і норм щодо організації навчально-виховного процесу[4].

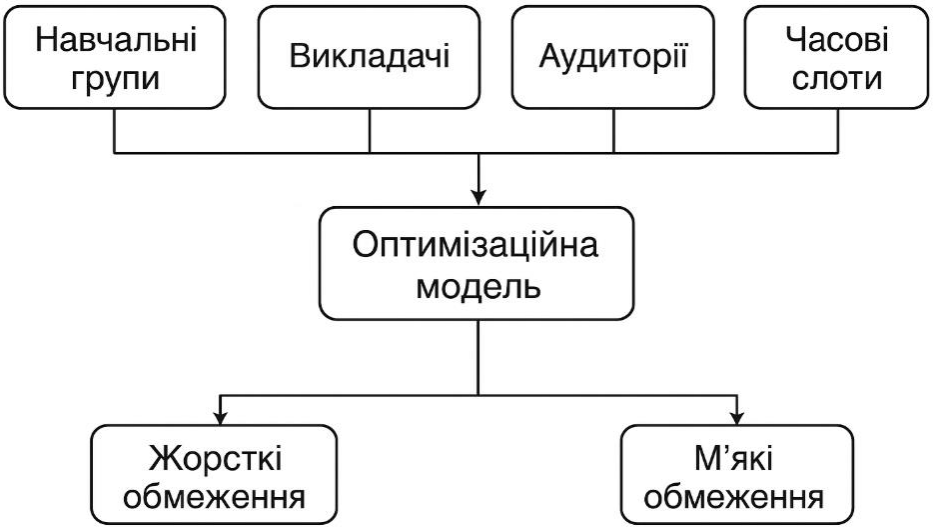


Рисунок 1.3 – Модель задачі оптимізації навчального розкладу

Таким чином, задача формулюється як пошук такого призначення занять до аудиторій та часових слотів, яке задовольняє всім жорстким обмеженням та мінімізує сумарний штраф за порушення м'яких обмежень (рис. 1.3). Ця формалізація є основою для подальшого вибору та реалізації алгоритмів оптимізації, зокрема еволюційних алгоритмів та метаевристичних підходів.

1.3 Оптимізаційне ядро як складова системи підтримки прийняття рішень у плануванні навчального процесу

Системи підтримки прийняття рішень (СППР) у сфері освітнього менеджменту покликані забезпечувати інформаційну, аналітичну та обчислювальну підтримку процесів планування. Задача формування навчального розкладу, що є однією з найбільш структурно складних задач в освітній діяльності, потребує обробки великої кількості взаємопов'язаних даних, одночасного врахування жорстких і м'яких обмежень та оцінки значної кількості можливих варіантів. У класичних СППР основні акценти робились на моделюванні предметної області, накопиченні історичних даних, забезпеченні інтерфейсів взаємодії й автоматизації бізнес-процесів. Проте сучасні вимоги до адаптивності, ефективності та точності формування

розкладу вимагають інтеграції методів штучного інтелекту, які можуть виконувати пошук рішень у великому просторові можливостей із урахуванням багатокритеріальних оцінок.

У контексті планування навчального процесу СППР виконує роль середовища, що забезпечує:

– формалізацію вхідних даних;

– побудову моделі обмежень;

– генерацію альтернативних сценаріїв;

– оцінювання варіантів за низкою метрик;

– інтерактивну підтримку адміністративних рішень.

У такому середовищі оптимізаційне ядро виступає центральним компонентом, здатним виконувати обчислювально інтенсивний пошук оптимального або наближеного розкладу. На відміну від традиційних алгоритмів, що орієнтуються на детерміновані процедури, оптимізаційне ядро у СППР використовує методи штучного інтелекту для моделювання процесу прийняття рішень, пошуку компромісних варіантів та гнучкої адаптації до змінних умов.

Функціональна роль оптимізаційного ядра полягає у трансформації вихідних даних (навчальних планів, списку аудиторій, обмежень викладачів) у множину допустимих та прирівняних до оптимальних розкладів. Це досягається за рахунок використання метаевристичних алгоритмів, які забезпечують ефективне дослідження простору можливих рішень. Одними з поширених методів, що використовуються у СППР для задач розкладів, є генетичні алгоритми, алгоритм імітації відпалу та табу-пошук. Кожен із цих методів має свої особливості та сильні сторони, що дозволяє адаптувати їх під конкретні характеристики освітнього середовища.

Генетичні алгоритми (ГА) забезпечують глобальний пошук оптимальних рішень завдяки популяційному підходу, що передбачає еволюцію множини кандидатних рішень під дією операторів селекції, кросоверу та мутації. Вони демонструють високу стійкість до локальних

мінімумів і добре працюють у задачах, де пріоритети та обмеження мають неоднозначний характер. Імітація відпалу, навпаки, орієнтована на стохастичний локальний пошук і дозволяє поступово зменшувати міру хаотичних змін для досягнення стабільних рішень. Табу-пошук застосовує механізм "пам’яті" для уникнення повторних переходів у вже досліджені стани, що підвищує різноманітність пошуку та пришвидшує збіжність.

Особливість використання оптимізаційного ядра у СППР полягає не лише в генерації розкладу, а й у можливості пояснення отриманих рішень. Система не просто пропонує конкретний варіант розкладу, а й здатна аргументувати, чому вибрані саме такі дні, пари та аудиторії, які обмеження вплинули на рішення, які критерії було оптимізовано та якою є якість результату. Це значно підвищує довіру користувачів до автоматизованої системи та дозволяє інтегрувати людський фактор у процес оптимізації.

На відміну від традиційних систем, де оптимізаційний компонент є вбудованою частиною серверної логіки, у сучасних СППР оптимізаційне ядро доцільно реалізовувати як автономний інтелектуальний модуль. Така модульність забезпечує:

– можливість використання різних алгоритмів без зміни основного коду;

– масштабованість обчислень, включно із розподіленою обробкою;

– незалежне оновлення моделі оптимізації;

– гнучке налаштування параметрів під конкретні сценарії;

– прозорість і контрольованість оптимізаційних процесів.

Таким чином, оптимізаційне ядро у СППР відіграє роль аналітичного та обчислювального центру, що перетворює систему зі статичної бази даних і формальних правил на інтелектуальний інструмент управління освітнім середовищем. Його інтеграція дозволяє автоматизувати процес прийняття рішень, зменшити вірогідність помилок, підвищити якість планування та забезпечити адаптивність системи до нових умов.

2 ОГЛЯД ТА ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ ТА ІНСТРУМЕНТІВ

Сучасний розвиток веб-технологій пропонує широкий вибір інструментів та підходів для розробки інформаційних систем. Основними критеріями вибору технологічного стеку для даного проекту стали: продуктивність, безпека, швидкість розробки, масштабованість та наявність активної спільноти.

2.1 Аналіз сучасного технологічного стеку для розробки веб-додатків

На стартовому етапі проекту було розглянуто два сучасні архітектурні підходи: моналітний та мікросервісний. Незважаючи на тренд на мікросервіси, для системи управління навчальним процесом, особливо на етапі MVP, було обрано класичну трирівневу моналітну архітектуру.

Переваги для даного проекту: чітке розділення на Presentation Tier, Application Tier та Data Tier спрощує розуміння кодової бази для команди, пришвидшує розробку завдяки узгодженості інструментів та зменшує операційну складність розгортання, моніторингу та тестування. Ця архітектура ідеально відповідає моделі MVC, яка є основою багатьох сучасних фреймворків.

Порівняння з мікросервісами: мікросервісна архітектура, при всіх перевагах гнучкості та незалежного масштабування, вносить значну складність: необхідність оркестрації контейнерів (наприклад, Kubernetes), налаштування міжсервісної комунікації (REST/gRPC), централізованого логування та відстеження запитів. Для проекту середнього масштабу з чітко визначеним доменом це призвело б до невиправданих витрат часу та ресурсів на початковій стадії.

Для системи планування навчального процесу було обрано трирівневу архітектуру, що включає:

1) Рівень представлення (Front-end)
2) Бізнес-логіку (Back-end)
3) Рівень даних (Database)

Такий підхід забезпечує чітке розділення відповідальності, спрощує супровід та модифікацію системи.

Серед сучасних серверних мов програмування розглядалися Python/Django, Node.js та PHP. Python пропонує потужні бібліотеки для алгоритмів штучного інтелекту, однак PHP з фреймворком Yii2 забезпечує кращу продуктивність для CRUD-операцій, має вбудовані механізми безпеки та дозволяє швидко реалізувати типовий функціонал.

Для інтерфейсу користувача розглядалися сучасні фреймворки React та Vue.js, проте для адміністративних систем, де важлива швидкість розробки стандартних інтерфейсів, традиційний підхід з використанням HTML, CSS та JavaScript у поєднанні з можливостями Yii2 є оптимальним рішенням.

Аналіз показав, що реляційні СКБД найкраще підходять для структурованих даних навчального процесу. PostgreSQL та MySQL пропонують надійність та підтримку транзакцій, що є критично важливим для цілісності даних розкладу.

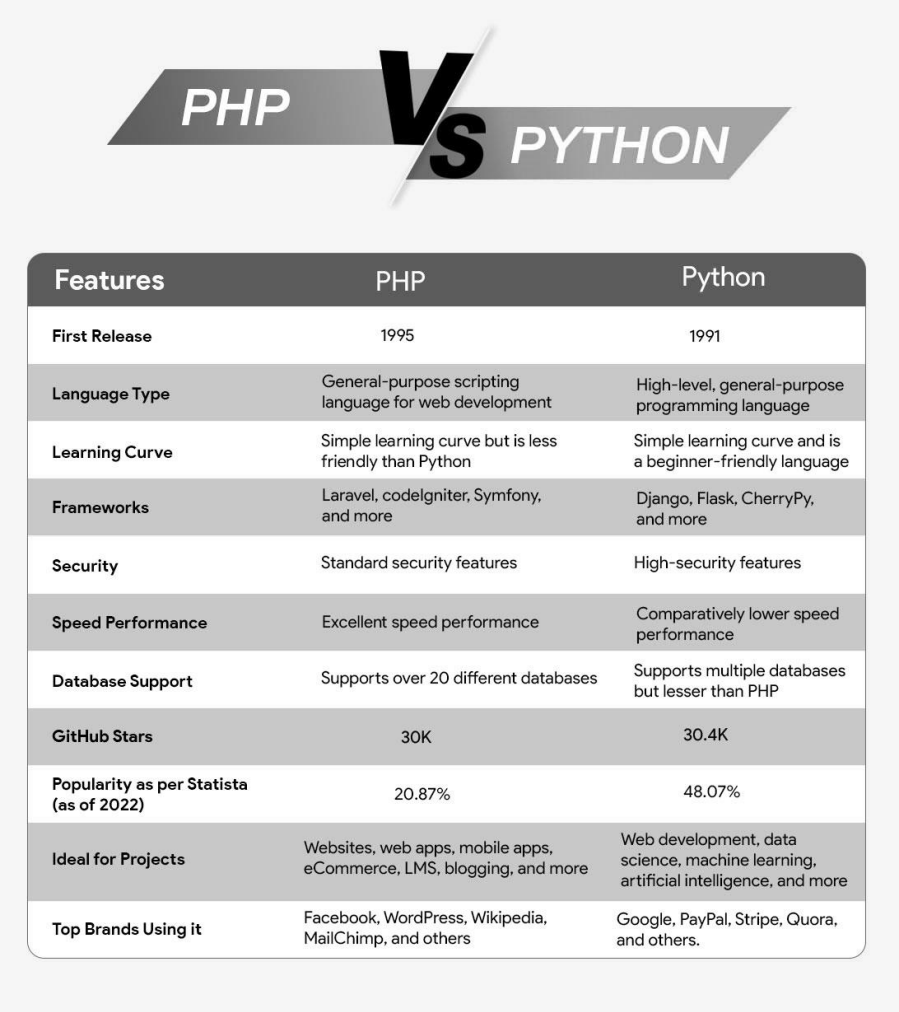
2.2 Вибір та обґрунтування технологій для реалізації системи

На основі проведеного аналізу та враховуючи специфіку проекту, було обрано наступний технологічний стек:

Серверна частина (Back-end):

1) Мова програмування: PHP 8.1+

Остаточний вибір на користь PHP був зроблений на основі порівняльного аналізу з двома основними альтернативами: Python та Node.js.



Features	PHP	Python
First Release	1995	1991
Language Type	General-purpose scripting language for web development	High-level, general-purpose programming language
Learning Curve	Simple learning curve but is less friendly than Python	Simple learning curve and is a beginner-friendly language
Frameworks	Laravel, codelgniter, Symfony, and more	Django, Flask, CherryPy, and more
Security	Standard security features	High-security features
Speed Performance	Excellent speed performance	Comparatively lower speed performance
Database Support	Supports over 20 different databases	Supports multiple databases but lesser than PHP
GitHub Stars	30K	30.4K
Popularity as per Statista (as of 2022)	20.87%	48.07%
Ideal for Projects	Websites, web apps, mobile apps, eCommerce, LMS, blogging, and more	Web development, data science, machine learning, artificial intelligence, and more
Top Brands Using it	Facebook, WordPress, Wikipedia, MailChimp, and others	Google, PayPal, Stripe, Quora, and others.

Рисунок 2.1 – Порівняльна характеристика Python та PHP для реалізації системи [5]

PHP vs. Python: Python, зокрема у зв'язці з фреймворком Django, є відмінним інструментом для швидкого прототипування, наукових обчислень та проєктів з елементами AI/ML. Однак, для типового веб-додатку з переважанням CRUD-операцій та складними формовими введеннями, PHP демонструє вищу продуктивність у синхронному веб-середовищі(рис. 2.1). PHP з його традиційною багатопотоковою моделлю (за допомогою FPM) ефективно обробляє кожен запит окремо, що ідеально підходить для навантажень, характерних для адміністративних систем. Крім того, екосистема PHP орієнтована саме на веб, тоді як екосистема Python більше універсальна. PHP 8.x приніс революційні зміни: JIT-компіляцію (Just-In-Time), що ще більше підняло продуктивність, суворішу типізацію, покращену

обробку помилок – що робить мову сучасною, безпечною та конкурентноздатною.

PHP та Node.js: Node.js з його подієво-орієнтованою, неблокуючою архітектурою є лідером у продуктивності для високонавантажених додатків реального часу. Проте, система планування навчального процесу є класичним додатком із запитом-відповіддю, де основне навантаження лягає на роботу з базою даних та генерацію HTML. У таких умовах перевага Node.js зникає, а складність асинхронної моделі програмування може суттєво збільшити час розробки та ускладнити дебаггінг, особливо для складних транзакційних операцій з БД. Синхронний код на PHP набагато простіший для читання, тестування та підтримки для типових бізнес-процесів.

Таким чином, PHP 8.1+ обрано через:

1) Оптимальну продуктивність для CRUD-навантажень.
2) Зрілу та веб-орієнтовану екосистему з тисячами стабільних бібліотек через Composer.
3) Низький поріг входу та велику спільноту, що важливо для подальшої підтримки та розширення системи на рівні навчального закладу.
4) Простоту розгортання та супроводу на стандартних веб-серверах.

Серед PHP-фреймворків основним конкурентом для Yii2 є Laravel. Обидва – сучасні, потужні інструменти, проте їхні філософія та акценти різняться, що визначило вибір на користь Yii2 для даного проєкту.

Yii2 та Laravel: Laravel славиться своєю елегантністю, «магією» (завдяки Facades та динамічним методам), широчайшою екосистемою (Forge, Vapor, Nova) та акцентом на розробнику (developer experience). Він чудово підходить для складних корпоративних додатків та стартапів, де потрібна гнучкість та експресівність. Yii2, натомість, сфокусований на ефективності та продуктивності «з коробки». Він менш «магічний», але більш прямий і передбачуваний, що спрощує глибоке розуміння потоку виконання.

Ключові переваги Yii2 для системи планування навчального процесу:

Надзвичайно висока швидкість розробки адмін-інтерфейсів:

Вбудований генератор коду Gii дозволяє за лічені хвилини створити CRUD-операції для будь-якої сутності БД, включаючи форму, модель, контролер та представлення з пошуком, сортуванням та пагінацією. Для проєкту з десятками сутностей (групи, викладачі, аудиторії, предмети, розклад) це дає економію сотень годин рутинного програмування.

Потужна робота з даними через Active Record: ORM Yii2 є одним з найзручніших і продуктивних. Він дозволяє працювати з об'єктами БД як зі звичайними PHP-об'єктами, надає зручний синтаксис для складних запитів, автоматично екранує дані і підтримує реляційні зв'язки. Це критично важливо для складної доменної моделі навчального процесу.

Всебічна безпека «з коробки»: Yii2 має не просто інструменти, а готові, продумані рішення:

RBAC: Потужна та гнучка система керування доступом на основі ролей, ідеально підходить для розмежування прав адміністраторів, викладачів та студентів.

Валідація даних: Глибока інтеграція валідації як на стороні клієнта, так і на сервері, на основі правил у моделі.

Захист від основних атак: Автоматична генерація та перевірка токенів CSRF, фільтрація даних для запобігання XSS, підготовлені SQL-запити через AR/Query Builder.

Висока продуктивність та оптимізація: Yii2 спроектований з акцентом на швидкодію. Він має потужну систему кешування, лектування ресурсів та мінімальні накладні витрати. Для системи, яка буде обробляти тисячі записів розкладу, це ключова перевага.

Чітке дотримання патерну MVC: Архітектура Yii2 є дуже чистою та передбачуваною, що сприяє підтримуваності коду та полегшує колаборацію розробників.

Поєднання PHP 8.1+ та фреймворку Yii2 формує ідеальну основу для розробки системи планування навчального процесу. Цей стек забезпечує:

1) Максимальну швидкість створення стабільного та безпечного

адміністративного функціоналу.

2) Високу продуктивність при роботі з великими обсягами структурованих даних.

3) Надійність та безпеку завдяки вбудованим механізмам та зрілій екосистемі.

4) Легкість майбутньої підтримки та розширення системи на рівні навчального закладу.

Цей вибір дозволяє сконцентрувати зусилля на реалізації специфічної бізнес-логіки планування, а не на вирішенні базових архітектурних та інфраструктурних завдань.

Клієнтська частина (Front-end):

Обраний підхід до фронтенду ґрунтується на концепції прогресивного покращення та MPA (Multi-Page Application). На відміну від сучасних SPA-фреймворків (React, Vue.js, Angular), які вимагають побудови окремого клієнтського додатку та API, наш підхід максимально використовує можливості серверного фреймворку Yii2 для генерації інтерфейсу, додаючи динаміку та сучасний вигляд за допомогою перевірених технологій. Це забезпечує оптимальне співвідношення швидкості розробки, продуктивності та SEO-дружності для системи з переважно адміністративним інтерфейсом.

HTML5 та CSS3 –це базові веб-технології обрані як стійкий і стандартизований фундамент.

HTML5: Використовується для семантичної розмітки структури інтерфейсу. Сучасні теги покращують доступність для користувачів з обмеженнями можливостей та сприяють кращому розумінню контенту пошуковими системами. Вбудована валідація форм дозволяє здійснювати базову перевірку даних без JavaScript.

CSS3: Забезпечує сучасне візуальне оформлення. Використання Flexbox та CSS Grid дозволяє створювати складні, гнучкі та адаптивні макети без зайвих хаков. Модулі Custom Properties спрощують підтримку кольорової схеми та тем. CSS3 робить інтерфейс сучасним, забезпечуючи анімації, тіні,

градієнти, що підвищує користувацький досвід (UX) без шкоди для продуктивності.

Сучасний стандарт JavaScript (ECMAScript 6 та вище) використовується для покращення UX шляхом додавання динамічної взаємодії без перезавантаження сторінки.

JS не несе основного навантаження з рендерингу інтерфейсу, а виступає як допоміжний інструмент для покращення.

Ключові завдання:

1) Асинхронні запити (AJAX): Валідація полів форм на льоту, динамічне завантаження та фільтрація даних у випадаючих списках (наприклад, підвантаження списку аудиторій при виборі корпусу), відправка даних форм без перезавантаження сторінки.

2) Клієнтська валідація: Поглиблена перевірка даних у доповнення до серверної (наприклад, перевірка конфлікту часу при створенні розкладу).

3) Маніпуляція DOM: Динамічне додавання/видалення полів у формах, відображення та приховування блоків, створення інтерактивних елементів для поліпшення організації інтерфейсу.

4) Робота з подіями: Створення зручної взаємодії, реалізація драг-енд-дроп функціоналу для візуального планування розкладу.

Перевага над SPA-фреймворками: Такий підхід дозволяє уникнути надмірної складності клієнтського стейт-менеджменту, не потрібні окремі інструменти для SSR. Код залишається простим, прив'язаним до конкретних сторінок, і легко інтегрується з серверними віджетами Yii2.

Bootstrap 5 обрано як інструмент швидкої розробки адаптивного, консистентного та професійного інтерфейсу.

1) Адаптивна сітка: Потужна система колонок забезпечує ідеальне відображення інтерфейсу на всіх пристроях — від моніторів адміністраторів до планшетів викладачів. Таблиці, форми та панелі управління коректно масштабуються.

2) Готові, доступні компоненти: Наявність великої бібліотеки

компонентів (модальні вікна, навігаційні панелі, картки, таблиці, кнопки, сповіщення) прискорює розробку в десятки разів. Це дозволяє не винаходити велосипед для кожного елементу UI, а також гарантує їхню крос-браузерну сумісність та базову доступність.

3) Утиліти: Широкий набір CSS-утиліт для швидкого стилізації (відступи, кольори, видимість, flexbox) дозволяє кастомізувати вигляд без написання власного CSS.

4) Відмова від jQuery: Bootstrap 5 переписаний на чистому JavaScript (ES6), що повністю відповідає нашому вибору сучасного JS та зменшує залежності.

Обґрунтування відмови від React/Vue.js для даного проєкту: Сучасні SPA-фреймворки є відмінним вибором для складних, інтерактивних клієнтських застосунків. Однак для системи планування навчального процесу їх використання було б надмірним та неефективним.

Потрібно було б створювати повноцінний REST API на бекенді та окремий фронтенд-додаток, що подвоює кодобазу. SPA, особливо на початковому етапі, може мати повільніший початковий рендеринг через необхідність завантаження великого бандлу JS. Також, більшість операцій у системі – це форми CRUD. Yii2 з генератором Gii та віджетами (GridView, DetailView) створює ці інтерфейси за хвилини. Їхня реалізація на React/Vue займала б години або дні.

Ефективна розробка сучасного веб-додатку залежить не лише від вибору мов програмування та фреймворків, але й від інструментів, що забезпечують контроль версій, стабільність середовища та продуктивність розробників. Для даного проєкту обрано наступний набір інструментів:

1) Docker

Docker обрано як фундаментальну технологію для створення, розгортання та управління додатком у легковагих, ізольованих контейнерах.

Консистентність середовищ: Docker дозволяє описати весь стек проєкту у файлі Dockerfile та docker-compose.yml. Це усуває класичну проблему «це

працювало на моїй машині». Середовище розробки, тестування та виробництва є ідентичними, що різко знижує кількість помилок, пов’язаних із налаштуваннями.

Швидке розгортання та масштабування: Запуск усієї інфраструктури проєкту зводиться до однієї команди docker-compose up. Це критично важливо для нових членів команди, які можуть почати роботу за лічені хвилини, а не години. У майбутньому контейнеризація спрощує розгортання на продакшн-серверах та горизонтальне масштабування окремих служб.

Ізоляція та безпека: Кожен сервіс (БД, PHP, веб-сервер) працює в окремому контейнері, що обмежує його доступ до ресурсів хоста та інших служб. Це підвищує безпеку та стабільність.

Для PHP/Yii2 є можливість легко керувати версією PHP, налаштуваннями PHP-FPM та необхідними розширеннями (gd, pdo_pgsql, opcache).

Для PostgreSQL є швидке розгортання бази даних з потрібною версією, попередньо налаштованими схемами та тестовими даними через міграції.

Для додатків є легка організація робочих просторів зі зв'язаними контейнерами.

2) Git – система контролю версій

Git є стандартом де-факто для сукупної розробки та обраний для проєкту через свою розподілену природу, гнучкість та потужність.

Контроль версій та історія змін: Git зберігає повну історію розробки, дозволяючи в будь-який момент відкотитися до стабільної версії, порівняти зміни, зрозуміти, хто і коли вніс конкретну правку. Це незамінно для відстеження виправлення помилок та аналізу розвитку проєкту.

Для проєкту використано моделі гілкування на Git Flow. Це дозволяє:

Вести розробку нових функцій в окремих гілках, не зачіпаючи стабільну основну гілку.

Виділяти гілки для виправлення критичних помилок на продакшні.

Організовано проводити код-рев'ю перед злиттям змін, що підвищує

якість коду.

Колаборація та резервування Git дозволяє декільком розробникам працювати паралельно, мінімізуючи конфлікти. Віддалений репозиторій слугує централізованим резервним сховищем та основою для організації CI/CD.

3) Visual Studio Code – інтегроване середовище розробки (IDE)

Visual Studio Code (VS Code) обрано як основний інструмент розробника завдяки його легковажності, швидкості, безкоштовності та надзвичайно розвинутій екосистемі розширень.

Підтримка всього технологічного стеку: завдяки плагінам VS Code забезпечує першокласну підтримку всіх обраних технологій:

PHP та Yii2: інтелектуальне автодоповнення (IntelliSense), навігація по коду, рефакторинг, дебаггер (XDebug) з інтеграцією в Docker.

HTML/CSS/JavaScript: вбудована Emmet-підтримка, лінтинг, форматування, дебаггер для JS.

Bootstrap: підсвічування та автодоповнення класів.

SQL (PostgreSQL): підсвічування синтаксису, виконання запитів.

Docker: підсвічування синтаксису Dockerfile, управління контейнерами з інтерфейсу.

Git: вбудований клієнт Git для перегляду змін, створення комітів, перемикання гілок.

Продуктивність розробника: вбудований термінал дозволяє запускати команди Composer, Docker, Artisan/Yii, Git без перемикання між вікнами. Система вкладок, розділення екрану, теми оформлення роблять роботу комфортною.

Інтеграція з іншими інструментами: VS Code легко інтегрується з системами контролю якості коду, збірки та розгортання.

Ця комбінація стеку мінімізує налаштувальні та адміністративні витрати, дозволяючи команді зосередитись на реалізації бізнес-логіки системи планування навчального процесу.

Таблиця 2.1 – Порівняльні характеристики обраних технологій

Технологія	Переваги	Вплив на проект
PHP 8.1+	Висока продуктивність, низькі апаратні вимоги	Зменшення часу виконання скриптів
Yii2	AR для роботи з БД, безпека, кешування	Скорочення часу розробки на 30-40%
PostgreSQL 14+	Розширені типи даних, висока продуктивність складних запитів	Оптимізація роботи з даними розкладу
Docker	Ізоляція, повторюваність середовищ	Спрощення розгортання

Для зберігання та обробки структурованих даних системи планування навчального процесу було проведено детальний аналіз сучасних реляційних СКБД. Основним альтернативним рішенням розглядався MySQL/MariaDB. Незважаючи на широке поширення останніх у веб-розробці, для даного специфічного проєкту було однозначно обрано PostgreSQL 14+ як технологічно переважну та більш відповідну платформу.

Формування оптимального розкладу – це задача з багатьма обмеженнями. Для її вирішення необхідні потужні аналітичні запити. PostgreSQL пропонує вбудовану підтримку складних конструкцій, які або відсутні, або реалізовані слабше в MySQL:

CTE та рекурсивні запити дозволяють структурувати складні багатокрокові запити, що можуть знадобитися для пошуку ланцюжків залежностей між заняттями або перебору варіантів розкладу.

Потужні віконні функції критично важливі для формування звітів та аналізу розкладу без групування даних. Наприклад, можна легко отримати навантаження кожного викладача по тижнях, розрахувати кумулятивну зайнятість аудиторії або проранжувати заняття за пріоритетом.

Дані навчального процесу мають бути консистентними та захищеними

від пошкодження. PostgreSQL слідує принципам ACID суворіше, ніж MySQL за замовчуванням. Ключові аспекти:

1) Складні обмеження цілісності: Окрім стандартних FOREIGN KEY, UNIQUE, CHECK, PostgreSQL підтримує EXCLUDE Constraints. Це унікальна можливість, яка дозволяє запобігти накладенню занять у розкладі за різними критеріями.

2) Повноцінна транзакційність: Усі зміни, пов'язані з оновленням розкладу, можуть бути об'єднані в одну транзакцію. Це гарантує, що система не залишиться в неузгодженому стані у разі помилки.

Тип даних JSONB є одним з найважливіших аргументів. Він дозволяє в межах строгої реляційної структури гнучко зберігати додаткові, неструктуровані або напівструктуровані параметри занять: специфічні налаштування онлайн-трансляції, додаткові матеріали, динамічні метадані. JSONB підтримує індексацію, що забезпечує швидкий пошук навіть усередині таких полів.

СУБД ефективно використовує можливості багатоядерних процесорів для обробки складних SELECT-запитів та індексних сканувань.

Окрім стандартних B-дерев, PostgreSQL пропонує індекси GIN, GiST, BRIN. Це дозволяє тонко налаштувати БД під конкретні шаблони пошуку в системі.

Просунутий оптимізатор і краща статистика роблять PostgreSQL часто швидшим за MySQL у вибірках, що потребують об'єднання багатьох таблиць або складних умов WHERE.

Yii2 має відмінну підтримку PostgreSQL. ActiveRecord та Query Builder повною мірою використовують її можливості. Система міграцій Yii2 коректно працює з типами даних PostgreSQL, включаючи JSONB.

Драйвер PDO_PGSQL стабільний і ефективний.

Офіційний образ PostgreSQL легко інтегрується в docker-compose-середовище проєкту.

2.3 Оптимізаційне ядро як окремий сервіс штучного інтелекту для планування навчального процесу

Сучасні інформаційні системи, що застосовуються в управлінні навчальним процесом, дедалі частіше використовують модульний підхід, у якому складні обчислювальні механізми виділяються в автономні компоненти. Такий підхід особливо актуальний у задачах, що потребують значної обчислювальної потужності, адаптивності та гнучкого налаштування алгоритмів. Однією з таких задач є оптимізація навчального розкладу, яка поєднує в собі елементи комбінаторної оптимізації, багатокритеріального аналізу та численних обмежень, що залежать від різних аспектів навчального процесу.

Розміщення дисциплін, узгодження розкладів викладачів і груп, раціональний розподіл аудиторного фонду, обмеження за типами приміщень, місткістю аудиторій і доступністю ресурсів — усе це формує складну задачу, що має експоненційну кількість можливих рішень. Традиційні підходи до оптимізації, що базуються на жорстких алгоритмах або фіксованих правилах, часто не здатні забезпечити прийнятні результати у масштабних сценаріях. Саме тому доцільним є використання методів штучного інтелекту, які забезпечують баланс між точністю, швидкістю обчислень і можливістю масштабування.

У класичних веб-системах алгоритмічний модуль зазвичай жорстко інтегрується до серверної частини застосунку. Однак у задачах високої складності цей варіант виявляється неефективним. Обчислювальні процедури, такі як генерація популяцій, оцінювання величезних просторів станів, застосування операторів мутації та кросоверу, можуть займати тривалий час і вимагати доступу до спеціалізованих бібліотек та оптимізованих середовищ виконання. Інтеграція таких процедур у ядро веб-застосунку може привести до деградації його продуктивності, блокування запитів користувачів або нестабільності при високому навантаженні.

Саме тому одним із найефективніших архітектурних рішень є відокремлення оптимізаційного ядра в автономний сервіс, який працює незалежно від основного застосунку та взаємодіє з ним через стандартизований інтерфейс обміну даними. Такий сервіс може бути реалізований у вигляді окремого процесу, контейнера або мікросервісу, що приймає на вхід структуровані дані, виконує обчислення та повертає результат у форматі, узгодженому з основною системою.

Вибір окремого сервісу також обумовлений тим, що методи оптимізації навчального розкладу найкраще реалізовувати у спеціалізованих середовищах. Наприклад, Python має велику кількість бібліотек для наукових обчислень, опрацювання масивів даних, реалізації еволюційних алгоритмів та вимірювання ефективності моделей. Це дозволяє зосередити складні алгоритмічні механізми у Python-модулі, тоді як веб-застосунок, створений на PHP/Yii2, зберігає свою структуру, стабільність і функціонал.

Важливим аргументом на користь такого відокремлення є можливість масштабування обчислювальних процесів. Коли оптимізація виконується у межах одного ядра веб-застосунку, збільшення кількості обчислень неминуче погіршує швидкодію всієї системи. Натомість окремий Python-сервіс може бути розгорнутий у кількох екземплярах, розподілений по різних вузлах або контейнерах, підключений до кластеризованого середовища. Це дозволяє обробляти паралельні запити на оптимізацію, що є критично важливим у великих навчальних закладах.

Крім того, виділений сервіс штучного інтелекту дозволяє інтегрувати різні оптимізаційні методи без зміни архітектури веб-додатку. Система може використовувати генетичні алгоритми, алгоритм імітації відпалу, табу-пошук або інші метаевристики, перемикаючись між ними на рівні конфігурації. Основний застосунок у такому випадку лише передає параметри алгоритму та структури даних, а оптимізаційний сервіс виконує обчислення і повертає результат.

Ще однією важливою перевагою окремого сервісу є можливість

формування універсального API, що забезпечує взаємодію з будь-якими зовнішніми системами, які можуть потребувати механізмів оптимізації. Навіть якщо у майбутньому структура бази даних зміниться, або система буде адаптована під інші цілі, сервіс оптимізації може бути повторно використаний, оскільки він працює не з конкретними таблицями, а з абстрактними структурами вхідних даних.

Суттєвим аргументом також є підвищення стабільності веб-застосунку. Замість того щоб запускати важкі обчислення всередині серверного середовища PHP, система делегує їх модулю, який виконується незалежно і не впливає на користувацькі запити. Таким чином, відмови або затримки у роботі оптимізаційного алгоритму не впливають на доступність інтерфейсу, журналів, авторизації або інших критичних підсистем.

Окремий сервіс також дозволяє реалізувати розширений рівень пояснюваності результатів. Оптимізаційний модуль може повертати не тільки сам розклад, але й його детальну статистику: значення функції пристосованості, кількість порушень обмежень, динаміку зміни найкращих рішень, кількість мутацій, розмір популяції, використання ресурсів (рис. 2.2). Ці метрики можуть бути інтегровані в інтерфейс системи або використані для подальшого аналізу ефективності.

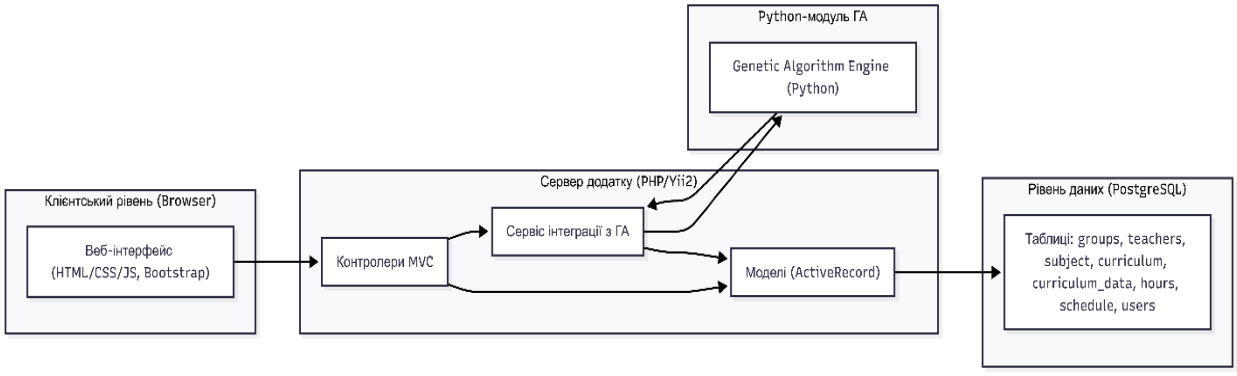


Рисунок 2.2 – Структурна схема моделі взаємодії між веб-застосунком та окремим оптимізаційним сервісом

Окремий сервіс працює з чітко структурованими даними. Формат вхідних даних може містити групи, аудиторії, доступність викладачів, часові

слоти, навчальний план та обмеження. Завдяки універсальності JSON-структур оптимізаційний модуль не прив’язаний до конкретного формату бази даних і може бути інтегрований у будь-яку систему, яка підтримує обмін структурованими повідомленнями.

Python-модуль може бути реалізований як окремий процес або сервіс.

У цьому підході веб-додаток відповідає за передавання даних у стандартизованому форматі, а Python-модуль — за побудову моделі оптимізації. Після завершення розрахунків результати можуть бути збережені у таблицях бази даних або представлені у вигляді набору структурованих JSON-об’єктів.

Узагальнюючи, виділення оптимізаційного ядра в окремий сервіс дозволяє:

- підвищити продуктивність системи,
- забезпечити ізоляцію складних алгоритмів,
- покращити масштабованість,
- підтримати різні методи оптимізації без зміни основного коду,
- забезпечити модульність та гнучкість архітектури,
- надати детальну аналітичну інформацію про процес оптимізації,
- робити систему стійкішою до змін і розширень.

Таким чином, окремий сервіс штучного інтелекту виступає ключовим елементом у створенні ефективної і сучасної системи планування навчального процесу.

3 ПРОЕКТУВАННЯ СИСТЕМИ АВТОМАТИЧНОГО ФОРМУВАННЯ РОЗКЛАДУ ЗАНЯТЬ

Спроектована система для автоматизованого складання навчального розкладу має забезпечувати не лише функціональну повноту, але й адаптивність до динамічних змін у навчальному процесі, які стали особливо актуальними в сучасних умовах [6]. Її ядром є гібридна архітектура, яка інтегрує чітку структурну організацію з потужними алгоритмами штучного інтелекту.

3.1 Проектування архітектури системи та моделей даних

Для забезпечення масштабованості, безпеки та легкості підтримки було обрано трирівневу клієнт-серверну архітектуру. Цей підхід розділяє відповідальність між рівнями, що дозволяє незалежно розвивати та оптимізувати кожен компонент системи. Особливо це критично для обробки великих даних та виконання ресурсомістких алгоритмів оптимізації, характерних для задачі планування розкладу.

Рівень представлення (Frontend): це веб-інтерфейс, через який взаємодіють адміністратори, викладачі та студенти. Він побудований з використанням сучасного JavaScript-фреймворка, що забезпечує динамічну та інтуїтивно зрозумілу роботу з розкладом, нагадуваннями та звітами. Усі складні обчислення виконуються на сервері, що робить клієнтську частину легкою та швидкою.

Рівень бізнес-логіки (Backend): серце системи. Тут реалізовано ядро оптимізації, алгоритми ШІ, систему логічних правил та API для обміну даними. Цей рівень розроблений на основі патерну MVC (рис. 3.1), що структурує код на [8]:

Моделі (Models): відповідають за структуру даних та бізнес-логіку їх взаємодії.

Контролери (Controllers): обробляють запити від клієнта, керують потоком виконання, викликають потрібні алгоритми та повертають результат.

Подання (Views): готують дані у форматі, прийнятному для передачі клієнту.

Рівень даних (Database): для надійного зберігання всієї структурованої інформації використовується реляційна база даних PostgreSQL. Вона зберігає не лише довідкові дані, але і всі обмеження, історію змін, результати роботи алгоритмів.

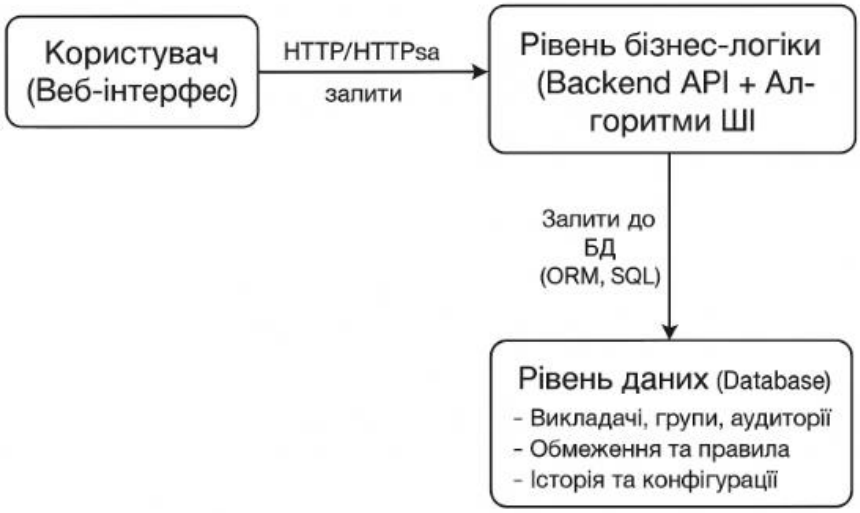


Рисунок 3.1 – Взаємодії рівнів архітектури

Задача автоматичного формування розкладу навчання є класичною NP-складною комбінаторною задачею оптимізації. Жоден єдиний алгоритм не може ефективно вирішити її з урахуванням усіх обмежень та цілей. Тому в системі реалізовано гібридний підхід, де різні методи працюють послідовно та взаємодоповнюють один одного[17].

Генетичний алгоритм для первинного пошуку рішень. ГА імітує процес природної еволюції:

Популяція: набір випадково згенерованих розкладів.

Функція пристосованості: ключовий компонент, який оцінює якість кожного розкладу. Вона враховує ступінь порушення обов'язкових обмежень та цільові критерії.

Відбір, схрещування та мутація: на кожній ітерації найкращі розклади мають більші шанси «успадкувати» свої частини наступному поколінню, з випадковими змінами. Це дозволяє ефективно досліджувати величезний простір можливих варіантів.

Розклад, отриманий ГА, часто можна покращити[16]. Для цього застосовуються:

Імітація відпалу: алгоритм дозволяє з певною ймовірністю приймати «гірші» рішення на початку, щоб вийти з локального мінімуму, поступово «охолоджуючись» і закріплюючись в якісному рішенні.

Пошук із заборонами: веде список недавніх змін, уникаючи циклічного повернення до вже переглянутих станів, що забезпечує більш різноманітний пошук.

Система логічних правил та обмежень. Це механізм, що гарантує дотримання жорстких обмежень. Він працює як фільтр на всіх етапах:

Обов'язкові правила: один викладач або група не може бути в двох місцях одночасно; аудиторія не перевищує місткість; заняття відповідає спеціалізації кафедри.

Інтеграція з алгоритмами: правила вбудовані в функцію пристосованості ГА та використовуються для перевірки допустимості будь-якої зміни в метаевристиках.

Методи багатокритеріальної оптимізації. Оскільки ідеальний розклад має відповідати багатьом, часто суперечливим вимогам, система використовує підходи для пошуку компромісів:

Метод Парето: знаходить множину розкладів, де покращення за одним критерієм призводить до погіршення за іншим. Кінцевий користувач може вибрати варіант з цієї множини.

Зважена сума цільових функцій: критерії об'єднуються в єдину функцію з налаштовуваними ваговими коефіцієнтами, що відображають пріоритети навчального закладу.

Для підвищення адаптивності система може включати ML-моделі, які

аналізують історичні дані для:

1. Прогнозування дефіциту аудиторій певного типу.
2. Оцінки популярності вибіркових дисциплін для оптимального розподілу потоків.
3. Моделювання навантаження викладачів з урахуванням індивідуальної продуктивності.

Обґрунтування вибору гібридної методології

Комбінація описаних підходів не є випадковою. Вона безпосередньо відповідає на виклики, описані в нормативних документах щодо організації освітнього процесу, які наголошують на необхідності гнучкості, адаптивності та оперативного реагування на зміни [6] [7].

Генетичний алгоритм забезпечує широкий пошук і здатний знайти принципово різні варіанти структури розкладу, що критично важливо при раптовій зміні навчального плану або доступності ресурсів.

Метаевристики дозволяють «полірувати» розклад, максимально адаптуючи його до конкретних поточних умов, наприклад, під час ущільнення навчального тижня або переходу на змішаний формат навчання [7, 18].

Централізована система правил гарантує, що будь-який згенерований варіант буде життєздатним і відповідатиме всім обов'язковим вимогам, зменшуючи навантаження на людину-планувальника.

Багатокритеріальна оптимізація дозволяє системі враховувати різні, часто конфліктуючі, інтереси учасників процесу, знаходячи збалансовані рішення.

Таким чином, проектування архітектури та логіки системи виходило з принципу створення не жорсткого інструменту, а адаптивної платформи. Ця платформа здатна не тільки автоматизувати рутинне складання розкладу, але й ефективно підтримувати прийняття управлінських рішень в умовах невизначеності , характерних для сучасної освітньої середовища [6].

3.2 Визначення вимог до системи

Для ефективної розробки системи необхідно чітко визначити категорії користувачів та їх функціональні можливості[21]. В інформаційній системі передбачено три основні типи користувачів: адміністратор, викладач та студент.

Адміністратор – має повний доступ до системи, відповідає за управління даними, формування навчального плану та розкладу (табл. 3.1).

Викладач – може переглядати розклад занять та своє навантаження.

Студент – має доступ лише до перегляда розкладу занять своєї групи

Таблиця 3.1 – Функціональні завдання користувачів системи

Завдання	Вхідні дані	Вихідні дані
1. Адміністратор		
1.1 Перегляд розкладу	Назва групи	Назва групи День тижня Номер уроку Назва предмету
1.2 Редагування розкладу конкретної групи	Назва групи	Назва групи День тижня Номер уроку Назва предмету
1.3 Додавання групи	Назва групи	
1.4 Видалення групи	Назва групи	
1.5 Створення навантаження для вчителя	Прізвище вчителя Ім'я вчителя Назва предмета Рік навантаження Назва групи Кількість тижнів Годин на тиждень	

Продовження таблиці 1.3.1

1.6 Розподіл навантаження викладача	Прізвище викладача Ім'я викладача По батькові викладача	Прізвище, ім'я, по батькові вчителя Назва предмета Рік навантаження Назва групи Кількість тижнів Годин на тиждень Всього годин на рік
1.7 Перегляд навантаження викладача	Прізвище викладача Ім'я вчителя По батькові вчителя	Прізвище вчителя Ім'я вчителя По батькові викладача Назва предмету Рік навантаження Назва групи Кількість тижнів Годин на тиждень Всього годин на рік
1.8 Додавання предмета	Назва предмета Кількість лекційних годин Кількість лабораторних годин Кількість практичних годин	
1.9 Видалення предмета	Назва предмета	Назва предмета Кількість лекційних годин Кількість лабораторних годин Кількість практичних годин

Продовження таблиці 1.3.1

1.10 Додавання викладача	Ім'я Прізвище По батькові Ім'я для входу Пароль	
1.11 Видалити вчителя	Логін Пароль	Ім'я Прізвище По батькові
Викладач		
2.2 Перегляд розкладу	Назва групи	Назва групи День тижня Номер уроку Назва предмету
2.3 Перегляд навантаження викладача	Прізвище викладача Ім'я викладача По батькові вчителя	Прізвище вчителя Ім'я вчителя По батькові викладача Назва предмету Рік навантаження Назва групи Кількість тижнів Годин на тиждень
Студент		
3.1 Перегляд розкладу	Назва групи	Назва групи День тижня Номер уроку Назва предмету

Система повинна відповідати наступним нефункціональним вимогам:

Продуктивність: час відгуку системи не повинен перевищувати 2-3 секунди для будь-якої операції.

Масштабованість: система повинна підтримувати до 1000 одночасних користувачів.

Безпека: реалізація системи розмежування прав доступу та аутентифікації користувачів.

Зручність використання: інтуїтивно зрозумілий інтерфейс для всіх категорій користувачів.

3.3 Проектування бази даних системи

Проектування сховища даних є критичним етапом розробки системи, оскільки від його ефективності залежить продуктивність всіх алгоритмів оптимізації та коректність бізнес-логіки. База даних формує основу рівня даних у прийнятій трирівневій архітектурі. Для забезпечення надійності, цілісності та узгодженості інформації було обрано реляційну модель даних. Вона оптимально підходить для структурованих даних з чіткими зв'язками, таких як навчальні плани, групи та викладачі (табл. 3.2)., а також забезпечує надійність транзакцій та легкість підтримки завдяки мові SQL [9].

Таблиця 3.2 – Опис сутностей та їх властивостей

Властивість	Опис	Обмеження
Об’єкт «speciality»		
number_of_speciality	Ідентифікатор	NOT NULL, UNIQUE
speciality	Назва спеціальності	NOT NULL
Об’єкт «groups»		
id_group	Ідентифікатор	NOT NULL, UNIQUE
name	Назва групи	NOT NULL

Продовження таблиці 3.2

number_of_speciality	Номер спеціальності	NOT NULL
Об’єкт «curriculum»		
id_of_curriculum	Ідентифікатор	NOT NULL, UNIQUE
year	Рік навчального плану	NOT NULL
number_of_speciality	Номер спеціальності	NOT NULL
number_of_curriculum	Номер навчального плану	NOT NULL
Об’єкт «subject»		
subject_id	Ідентифікатор	NOT NULL, UNIQUE
subject	Назва предмета	NOT NULL
Об’єкт «teacher»		
id_teacher	Ідентифікатор	NOT NULL, UNIQUE
surname	Прізвище викладача	NOT NULL
phone_number	Номер телефону викладача	NOT NULL
patronymic	По батькові викладача	NOT NULL
Name	Ім’я викладача	NOT NULL
Об’єкт «curriculum_data»		
id_of_curriculum	Ідентифікатор навчального плану	NOT NULL
subject_id	Ідентифікатор предмета	NOT NULL
id_of_curriculum_data	Ідентифікатор	NOT NULL, UNIQUE

Продовження таблиці 3.2

hours_per_week	Кількість годин на тиждень	
semester	Семестр	значення < 3
weeks	Загальна кількість тижнів у семестрі	
Об’єкт «hours»		
type_of_lesson	Тип заняття	
id_of_curriculum_data	Ідентифікатор навчального плану	
hours_of_specific_type	Кількість годин занять	NOT NULL
number_of_speciality	Номер спеціальності	NOT NULL
id	Ідентифікатор	NOT NULL, UNIQUE
Об’єкт «schedule»		
id	Ідентифікатор	NOT NULL, UNIQUE
lesson	Номер заняття	значення < 6
day_of_week	День тижня	
id_group	Ідентифікатор групи	NOT NULL
id_of_curriculum_data	Ідентифікатор навчального плану	NOT NULL
Об’єкт «users»		
username	Ім’я користувача	NOT NULL, UNIQUE
password	Хешований і посолений пароль	NOT NULL
id_teacher	Ідентифікатор викладача	NOT NULL

Концептуальна модель предметної області, представлена в таблиці 3.1, визначає основні сутності та їх атрибути. На її основі було розроблено логічну модель у вигляді схеми бази даних, яка візуалізує всі таблиці, їх поля та зв'язки між ними. Кінцевою метою проектування було приведення схеми до нормальної форми Бойса-Кодда, що усуває аномалії вставки, оновлення та видалення даних, забезпечуючи їхню цілісність та мінімізуючи надмірність [10].

Модель даних системи складається з дев'яти взаємопов'язаних сутностей (рис. 3.2). Нижче наведено детальний опис кожної з них та її ролі в системі.

speciality (Спеціальність). Базова довідкова сутність, що зберігає перелік спеціальностей навчального закладу. Є точкою входу для формування навчальних планів та груп.

groups (Група). Відображає академічні групи студентів. Кожна група однозначно прив'язана до однієї спеціальності через зовнішній ключ number_of_speciality, що реалізує зв'язок «один-до-багатьох».

curriculum (Навчальний план). Зберігає затверджені навчальні плани для кожної спеціальності у конкретному році. Зв'язок зі speciality також є «один-до-багатьох».

subject (Дисципліна). Довідник усіх навчальних дисциплін. Нормалізація вимагає винесення назв предметів в окрему таблицю для уникнення дублювання.

teacher (Викладач). Містить персональні та контактні дані викладачів. Поле user_id є зовнішнім ключем для зв'язку «один-до-одного» з таблицею облікових записів.

users (Користувач). Відповідає за аутентифікацію та авторизацію. Зберігає унікальне ім'я користувача та хеш пароля. Зв'язок «один-до-одного» з teacher гарантує, що кожен обліковий запис відповідає одному викладачу.

curriculum_data (Деталі навчального плану). Ключова сутність, що агрегує інформацію для генерації розкладу. Вона об'єднує конкретний навчальний план, дисципліну, призначеного викладача та нормативні

параметри. Саме ця сутність є основним об'єктом для алгоритмів оптимізації.

hours (Розподіл годин). Деталізує curriculum_data, вказуючи, скільки годин з загального обсягу відводиться на конкретні типи занять. Це дозволяє системі точніше планувати аудиторний фонд.

schedule (Розклад). Фінальна сутність, що містить результат роботи системи – сформований розклад занять. Кожен запис визначає, в який день та під яким порядковим номером група має заняття з конкретної дисципліни. Ця таблиця безпосередньо відображається в інтерфейсі користувача.

Рисунок 3.2 – Діаграма зв'язків сутностей системи

Для гарантії коректності інформації в базі даних реалізована система обмежень. Первинні ключі забезпечують унікальні ідентифікатори для кожного запису. Зовнішні ключі реалізують всі описані зв'язки, забезпечуючи цілісність посилань. Унікальні обмеження забороняють дублювання критичних даних, такі як назва групи або ім'я користувача [11]. Обмеження на значення NULL гарантують заповненість обов'язкових полів. Додаткова бізнес-логіка, така як перевірка семестру або номера заняття, реалізована на рівні коду додатку.

Спроектована база даних є оптимальною основою для алгоритмів оптимізації. Зв'язок між деталями навчального плану та фінальним розкладом є центральним: алгоритм працює з множиною навчальних завдань як з набором елементів, які потрібно розмістити в часовій сітці. Система обмежень бази даних забезпечує, що будь-яке сгенероване рішення буде узгодженим зі структурою навчального процесу. Таким чином, база даних виступає не лише сховищем, а й активним інструментом, що підтримує коректність бізнес-логіки та забезпечує високу продуктивність пошуку та фільтрації даних під час роботи алгоритмів.

3.4 Інтеграція Python-модуля оптимізації в бізнес-логіку системи підтримки прийняття рішень

Інтеграція оптимізаційного ядра, реалізованого у вигляді окремого Python-модуля, є ключовим аспектом побудови сучасної системи підтримки прийняття рішень, орієнтованої на автоматизацію планування навчального процесу. Python-модуль оптимізації виступає центральним елементом обчислювального контуру, який взаємодіє з веб-додатком (реалізованим на PHP/Yii2) через стандартизований інтерфейс обміну даними.

Особливість архітектури полягає у тому, що між бізнес-логікою системи та оптимізаційним ядром існує чіткий розподіл відповідальності. Веб-додаток виконує роль інтерфейсу доступу до даних, контролює взаємодію з

користувачами, здійснює авторизацію та автентифікацію, відповідає за збереження і структурування інформації у базі даних, а також забезпечує відображення результатів оптимізації. Python-модуль, у свою чергу, концентрує всі обчислювальні процеси: формування моделей для оптимізації, застосування операторів генетичного алгоритму, побудову популяцій, оцінювання функції пристосованості та генерування оптимального або наближеного до оптимального розкладу.

У типовому сценарії роботи процес інтеграції Python-модуля розпочинається з формування веб-додатком вхідної структури даних, що описує навчальний план, групи, аудиторії, викладачів, обмеження та параметри оптимізації. На стороні PHP реалізується спеціальний сервісний клас, який відповідає за трансляцію внутрішніх структур БД у стандартизований JSON-формат. Цей файл або об’єкт передається Python-модулю через один із двох механізмів: запуск зовнішнього процесу (subprocess) або HTTP-виклик у разі, якщо оптимізаційне ядро розгорнуте як мікросервіс.

Особливістю такої архітектури є асинхронність взаємодії між веб-додатком та ядром оптимізації. Оскільки процес оптимізації може потребувати значного часу, бізнес-логіка не повинна блокувати запити користувачів. Натомість вона ініціює завдання оптимізації й отримує відповідь лише після завершення обчислювального циклу. Це дозволяє системі залишатися доступною й забезпечувати високу відмовостійкість навіть за умов значного навантаження.

З боку Python-модуля важливим аспектом є перетворення вхідних даних у внутрішнє представлення хромосомите популяцій, що використовуються генетичним алгоритмом. Дані, отримані від веб-додатку, містять параметри навчального плану, конфігурації занять, перелік аудиторій та доступність викладачів. Модуль повинен виконувати попередню валідацію даних, забезпечувати коректність їх структури та переводити їх у формат, придатний для обчислення. Це включає побудову таблиць відповідностей, підготовку

масивів допустимих значень та формування початкової популяції на основі дозволених комбінацій.

У рамках інтеграції Python-модуля важливою є чітка реалізація API між модулями. Веб-додаток має передавати оптимізаційному ядру такі дані:

1. Довідники (групи, викладачі, предмети, аудиторії).
2. Навчальний план (>= одна структура curriculum_data).
3. Обмеження (доступність викладачів, типи аудиторій, місткість).
4. Параметри алгоритму (розмір популяції, кількість поколінь, коефіцієнти мутації та кросоверу).

У відповідь Python-модуль формує:

1. найкращий знайдений варіант розкладу;
2. статистичні показники оптимізації;
3. перелік порушених обмежень (за наявності);
4. значення функції пристосованості;
5. службову інформацію про процес оптимізації.

Отриманий результат веб-додаток розміщує у таблиці schedule, де кожен запис відповідає компоненту навчального плану, призначеному до певної аудиторії у певний час. Це дозволяє користувачеві переглядати розклад у різних представленнях: за групою, за викладачем, за аудиторією або за днем тижня.

Лістинг 3.1 показує базовий механізм трансформації вхідних JSON-даних у структуру, що буде використана ядром ГА. Модульна структура дозволяє розширювати функції без зміни загального інтерфейсу.

```
import json

def load_input(json_data):
    data = json.loads(json_data)

    groups = {g["id"]: g for g in data["groups"]}
    teachers = {t["id"]: t for t in data["teachers"]}
    rooms = {r["id"]: r for r in data["rooms"]}
    timeslots = data["timeslots"]
    components = data["components"]
```

```
    return {
        "groups": groups,
        "teachers": teachers,
        "rooms": rooms,
        "timeslots": timeslots,
        "components": components,
        "params": data.get("ga_params", {})
    }
```

Лістинг 3.1 – Обробка вхідних даних

Спрощений приклад на лістингу 3.2 демонструє структуру «хромосоми», де кожен ген — це конкретне заняття, призначене у часовий слот та аудиторію.

```
import random

def generate_initial_population(components, rooms, timeslots,
size=100):
    population = []
    for _ in range(size):
        individual = []
        for comp in components:
            room = random.choice(list(rooms.keys()))
            slot = random.choice(timeslots)
            individual.append((comp["id"], room, slot["id"]))
        population.append(individual)
    return population
```

Лістинг 3.2 – Формування початкової популяції у Python-модулі

У бізнес-логіці PHP/Yii2 реалізується сервіс, що викликає Python-модуль. Це може відбуватися через класичний механізм shell_exec(), якщо модуль викликається як процес, або через HTTP-клієнт, якщо модуль розгорнуто як окремий мікросервіс. Обидва варіанти забезпечують ізоляцію алгоритмічної частини, що дозволяє підтримувати її незалежно від веб-застосунку. Схема (рис. 3.3) демонструє роль кожного компоненту та порядок виконання операцій.

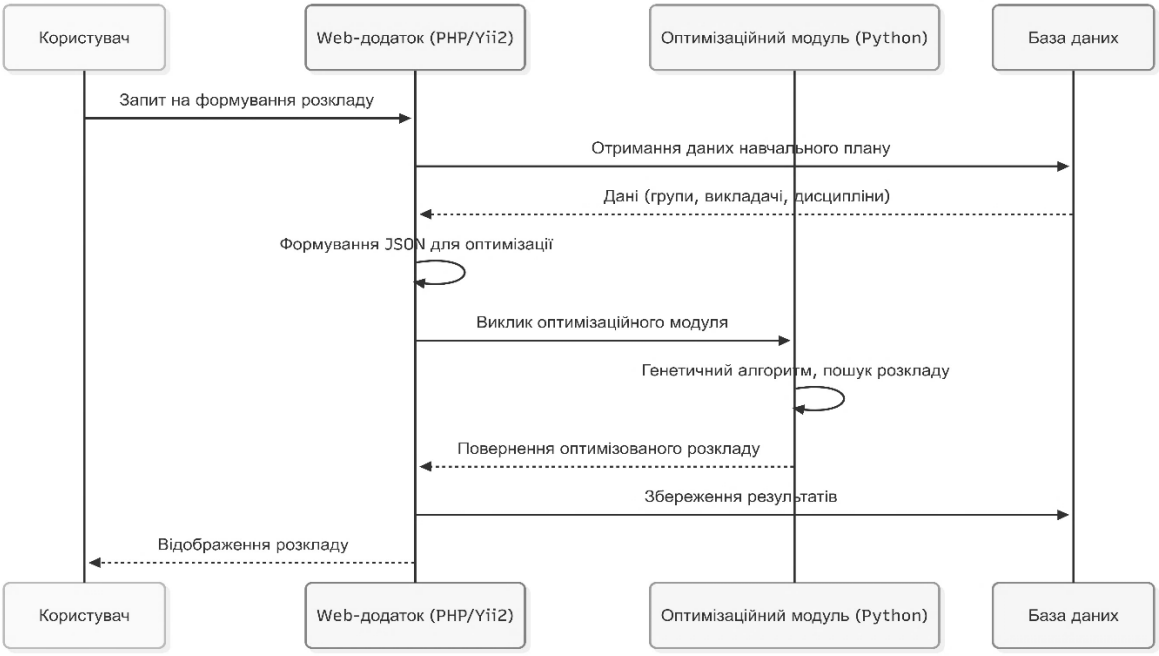


Рисунок 3.3 – Послідовність взаємодії СППР і Python-ядра

Інтеграція у бізнес-логіку СППР

У моделі СППР оптимізаційне ядро розглядається як компонент рівня аналітики, що має власний життєвий цикл та окремий програмний інтерфейс. PHP-частина системи залишається відповідальною за:

– керування користувачами;

– моделювання навчальних планів;

– забезпечення інтерактивного інтерфейсу;

– формування параметрів оптимізації;

– збереження та візуалізацію результатів.

Python-модуль не прив’язаний до конкретної реалізації БД чи бізнес-логіки. Він працює виключно з переданими структурами, що забезпечує універсальність і можливість його використання у різних системах.

Таке розділення дозволяє реалізувати гнучку конфігурацію СППР, де оптимізаційне ядро можна оновлювати, тестувати, масштабувати або замінювати без необхідності втручання у фронтенд чи бекенді бібліотеку. Наприклад, можна підключити альтернативний алгоритм або новий модуль ШІ, не змінюючи архітектуру.

Крім того, взаємодія між модулями може бути розширена механізмами черг (RabbitMQ, Redis Queue), що дозволяє обробляти запити на оптимізацію у відкладеному режимі. Це особливо актуально для великих навчальних закладів, де одночасно може виконуватися кілька задач оптимізації.

Узагальнюючи, інтеграція Python-модуля оптимізації в бізнес-логіку СППР не лише забезпечує коректну та ефективну роботу алгоритмів ШІ, але й формує гнучку архітектуру, яка може адаптуватися до майбутніх змін у навчальному процесі, вимогах користувачів чи технічних інструментах. Це дозволяє створити комплексну систему, здатну підтримувати прийняття рішень у складних умовах багатокритеріальної оптимізації.

3.5 Проектування інтерфейсу користувача (UI/UX).

Інтерфейс користувача (UI) та користувацький досвід (UX) є критичними компонентами системи, оскільки саме через них користувачі взаємодіють із складним функціоналом та даними. Для системи автоматизації розкладу, яка обслуговує різні категорії користувачів (адміністратор, викладач, студент), ключовим завданням було створення інтуїтивного, ефективного та безпечного інтерфейсу, що ґрунтується на чіткій інформаційній архітектурі.

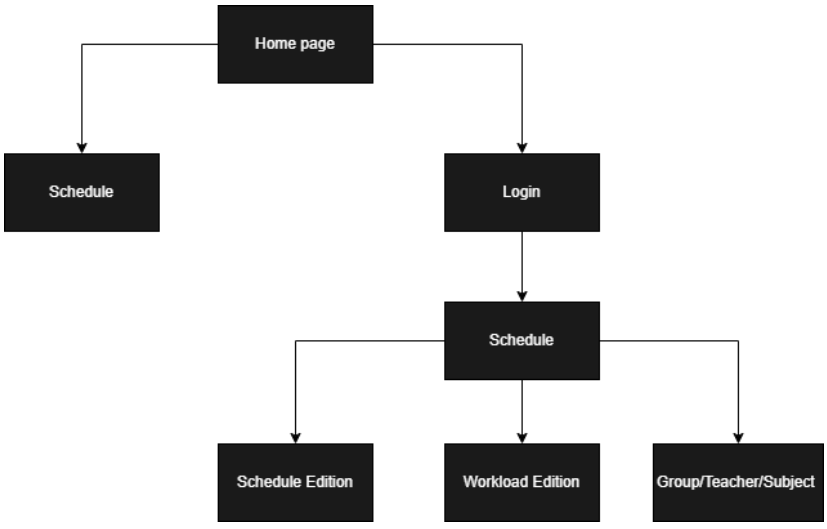


Рисунок 3.4 – Ієрархія сторінок додатку для користувача admin у розкладі занять для ІС

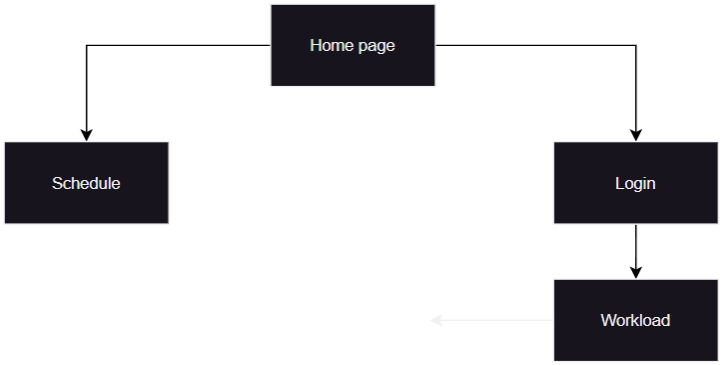


Рисунок 3.5 – Ієрархія сторінок додатку для користувача teacher у системі розкладу занять

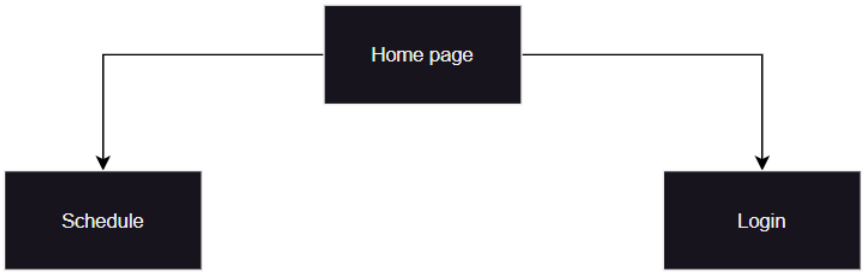


Рисунок 3.6 – Ієрархія сторінок додатку для користувача student у системі розкладу занять коледжу

Як зазначено в попередніх розділах, ядром UX-проектування стала ієрархічна модель сторінок(рис. 3.4-3.6). Цей підхід є класичним для організації складних систем, оскільки дозволяє логічно структурувати інформацію та функціонал, забезпечуючи зручну навігацію. Вибір цієї моделі прямим чином пов'язаний із концепцією «проектування на основі сценаріїв використання», коли інтерфейс будується навколо реальних завдань конкретних користувачів.

Для кожної ролі був визначений унікальний шлях користувача, що відображає послідовність дій для виконання ключових задач:

Студент: шлях мінімальний та лінійний: головна сторінка → вибір групи → перегляд розкладу. Авторизація не потрібна, що забезпечує максимальну доступність.

Викладач: шлях включає авторизацію для доступу до персональних даних: головна сторінка → вхід → портал з особистим розкладом та навантаженням.

Адміністратор: найскладніший шлях з розгалуженою структурою після авторизації: головна сторінка → вхід → центральний портал (дашборд) → доступ до модулів керування (групи, дисципліни, плани, генерація розкладу).

При розробці візуальної складової інтерфейсу керувалися набором фундаментальних принципів, визначених у сучасній практиці UI/UX-дизайну(табл. 3.3).

Таблиця 3.3 – Застосування принципів UI/UX у системі розкладу

Принцип	**Опис**	**Реалізація в системі**
Консистентність	Єдність візуальних елементів та поведінки в усіх частинах інтерфейсу.	Уніфікована бібліотека компонентів (кнопки, форми, таблиці), однакова структура меню для всіх адміністративних модулів.
Інтуїтивність та передбачуваність	Елементи поводяться так, як очікує користувач, без необхідності вивчати інструкції.	Кнопка «Зберегти» завжди зберігає зміни, «Назад» – повертає. Перетягування занять у розкладі має природню логіку.
Простота (Simplicity)	Мінімізація складності та навантаження на користувача.	Чистий інтерфейс для студента. Використання майстернів (wizards) для складних дій адміністратора
Ясність та зрозумілість	Чіткі мітки, легкий для читання текст, хороша контрастність.	Поле «Назва групи» замість «Назва_гр». Інформативні заголовки сторінок

Продовження таблиці 3.3

Обробка помилок	Система запобігає помилкам та надає зрозумілі повідомлення.	Підтвердження при видаленні запису. Підсвічування конфліктів у розкладі з поясненням типу («Перетин викладача»).
Гнучкість та свобода дій	Користувач має контроль, може скасовувати дії та відновлювати дані.	Кнопка «Скасувати» у формах. Автозбереження введеного тексту при перезавантаженні сторінки.

Проектування UI тісно інтегроване з логічною моделлю даних. Інтерфейс є «вікном» у базу даних, тому його структура безпосередньо відображає сутності та зв'язки схеми. Наприклад, форми редагування для таблиць teacher, subject, groups створюються на основі їхніх полів та обмежень (NOT NULL, UNIQUE).

Інтерфейс створення curriculum_data візуалізує зв'язок «багато-до-багатьох» між навчальним планом, дисципліною та викладачем через випадаючі списки.

Головна сторінка розкладу (schedule) агрегує дані з кількох таблиць (groups, curriculum_data, subject, teacher), трансформуючи їх у зручну для сприйняття табличну або календарну форму.

Цей підхід узгоджується з практикою паралельної розробки інтерфейсу та моделі даних, коли кожен компонент інформує та вдосконалює інший, забезпечуючи єдиність системи.

Процес проектування не був лінійним. Він включав ітеративні цикли прототипування та тестування.

Каркасні прототипи: були створені схематичні макети ключових сторінок для визначення розміщення основних блоків без детального дизайну.

Інтерактивні прототипи: на основі каркасів розроблені клікабельні прототипи високої точності, що імітували основні сценарії.

Юзабіліті-тестування: прототипи були представлені представникам цільових аудиторій. Тестування зосереджувалося на виконанні конкретних завдань («знайдіть розклад групи XX», «змініть викладача для дисципліни YY»). Зібраний фідбек дозволив виявити проблеми навігації та зрозумілості інтерфейсу на ранньому етапі.

Уточнення дизайну: на основі результатів тестування в інтерфейс були внесені правки: оптимізовано структуру меню, змінено мітку деяких кнопок, додано підказки.

Таким чином, проектування інтерфейсу системи автоматизації розкладу було комплексним процесом, що поєднав орієнтованість на користувача, дотримання фундаментальних принципів UI/UX, тісну інтеграцію з логічною моделлю даних та ітеративне вдосконалення на основі тестування. Це забезпечило створення не лише функціонального, але й зручного, ефективного та безпечного інструменту для всіх учасників навчального процесу.

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

4.1 Створення та наповнення бази даних

Першим практичним кроком у реалізації системи стало створення фізичної бази даних. Це фундаментальний етап, оскільки вся бізнес-логіка та алгоритми оптимізації побудовані навколо структурованого сховища інформації[12,13]. Виконана роботу з перетворення логічної моделі(рис. 4.1) у реальні таблиці PostgreSQL, використовуючи фреймворк Yii2 як основний інструмент оркестрації цього процесу.

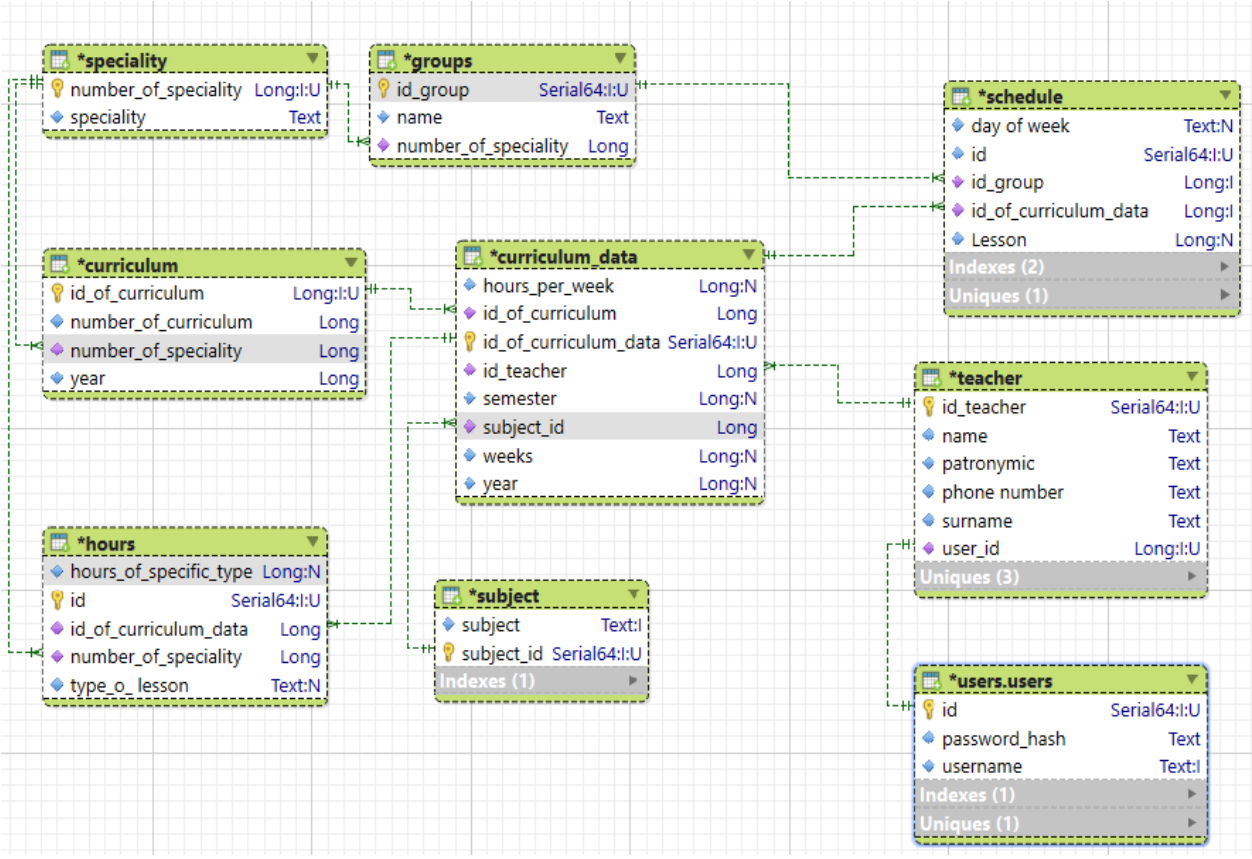


Рисунок 4.1 – Схема БД

Замість ручного виконання SQL-скриптів ми використали потужний механізм міграцій Yii2. Це дозволило нам вести історію змін схеми бази даних, легко відкатувати помилкові кроки та гарантувати однаковий стан БД на всіх серверах розробки, тестування та продакшену. Кожна сутність була

реалізована окремою міграцією. Наприклад, для створення ключової таблиці curriculum_data був написаний клас міграції (ліст. 4.1).

```
public function safeUp()
{
    $this->createTable('curriculum_data', [
        'id_of_curriculum_data' => $this->primaryKey()-
>unsigned(),
        'id_of_curriculum' => $this->integer()->notNull()-
>unsigned(),
        'subject_id' => $this->integer()->notNull()->unsigned(),
        'id_teacher' => $this->integer()->notNull()->unsigned(),
        'hours_per_week' => $this->integer(),
        'semester' => $this->integer()->check('semester < 3'),
        'weeks' => $this->integer(),
    ]);

    $this->addForeignKey(
        'fk-curriculum_data-curriculum',
        'curriculum_data',
        'id_of_curriculum',
        'curriculum',
        'id_of_curriculum',
        'CASCADE'
    );
    $this->addForeignKey(
        'fk-curriculum_data-subject',
        'curriculum_data',
        'subject_id',
        'subject',
        'subject_id',
        'CASCADE'
    );
    $this->addForeignKey(
        'fk-curriculum_data-teacher',
        'curriculum_data',
        'id_teacher',
        'teacher',
        'id_teacher',
        'CASCADE'
    );

    $this->createIndex(
        'idx-curriculum_data-teacher_semester',
        'curriculum_data',
        ['id_teacher', 'semester']
    );
}
```

Лістинг 4.1 – Клас міграції для створення ключової таблиці

Після того як структура була готова, базу потрібно було наповнити. поділено цей процес на етапи, використовуючи концепцію seeding(насіння). У спеціальних класах-сідерах для кожного факультету програмно було створено:

Довідники (speciality, subject): «Комп'ютерні науки», «Програмна інженерія», «Бази даних», «Математичний аналіз».

Актори (teacher, groups): викладачі з їхніми контактами та академічні групи на кшталт КН-401, ПІ-302.

Складні зв'язки (curriculum, curriculum_data): осередок всієї системи.

Тут наприклад, викладач id_teacher=5 читає предмет subject_id=12 («Бази даних») групі id_group=7 (КН-401) у першому семестрі протягом 17 тижнів, проводячи 4 години на тиждень (2 лекції + 2 практики).

Ці дані не були випадковими – вони базувалися на реальних навчальних планах коледжу, що дозволило відразу тестувати систему в умовах, близьких до бойових.

Сила реалізації – у повній інтеграції з філософією Yii2. Для кожної таблиці було створено модель, яка успадковується від yii\db\ActiveRecord. Це перетворило роботу з даними з написання сирих SQL на роботу з об'єктами та методами (ліст. 4.2).

```
namespace app\models;

class CurriculumData extends \yii\db\ActiveRecord
{
    public function getTeacher()
    {
        return $this->hasOne(Teacher::class, ['id_teacher' =>
'id_teacher']);
    }

    public function getSubject()
    {
        return $this->hasOne(Subject::class, ['subject_id' =>
'subject_id']);
    }

    public function getCurriculum()
```

```
    {
        return $this->hasOne(Curriculum::class,
['id_of_curriculum' => 'id_of_curriculum']);
    }

    public function getSchedules()
    {
        return $this->hasMany(Schedule::class,
['id_of_curriculum_data' => 'id_of_curriculum_data']);
    }
}
```

Лістинг 4.2 – Реалізація моделі CurriculumData із визначенням реляційних зв’язків між даними

Найважливіші бізнес-процеси, такі як формування звіту про навантаження викладача, реалізовуються елегантно та ефективно. Замість монолітних SQL-рядків запити побудовані динамічно (ліст. 4.3).

```
// Отримання повного навантаження для викладача з ID=42
$teacherId = 42;
$report = CurriculumData::find()
    ->select([
        'subject_name' => 'subject.subject',
        'group_name' => 'groups.name',
        'curriculum_data.semester',
        'curriculum_data.hours_per_week',
        'curriculum_data.weeks',
        'total_hours' => 'curriculum_data.hours_per_week *
curriculum_data.weeks'
    ])
    ->joinWith(['subject', 'curriculum.groups'])
    ->where(['curriculum_data.id_teacher' => $teacherId])
    ->orderBy(['semester' => SORT_ASC, 'subject_name' =>
SORT_ASC])
    ->asArray()
    ->all();
```

Лістинг 4.3 – Отримання структурованого звіту про навантаження викладача

Для ще складнішої логіки, наприклад, перевірки конфліктів у розкладі, ми створювали спеціальні методи у моделях. інкапсулювано логіку в потрібному місці (ліст. 4.4).

```
public static function findConflicts($groupId, $dayOfWeek,
$lessonNumber, $excludeScheduleId = null)
{
    $query = self::find()
        ->where([
            'id_group' => $groupId,
            'day_of_week' => $dayOfWeek,
            'lesson' => $lessonNumber
        ]);

    if ($excludeScheduleId) {
        $query->andWhere(['!=', 'id', $excludeScheduleId]);
    }

    return $query->all();
}
```

Лістинг 4.4 – Метод визначення конфліктних записів розкладу з можливістю виключення поточного елемента

Таким чином, створена база даних – це не просто статичне сховище, а повноцінна, жива основа системи. Її структура, наповнена реальними даними, та тісна інтеграція з ORM Yii2 забезпечили надійну та продуктивну платформу для впровадження всієї складної бізнес-логіки планування, що набуде форми в наступних підрозділах.

4.2 Розробка серверної та клієнтської частин системи

Реалізація системи розкладу перетворила теоретичні моделі у повноцінний веб-додаток. Цей процес об'єднав розробку надійного серверного ядра (бекенду) на фреймворку Yii2[11,14,15](рис. 4.2) та створення інтуїтивного інтерфейсу користувача (фронтенду).

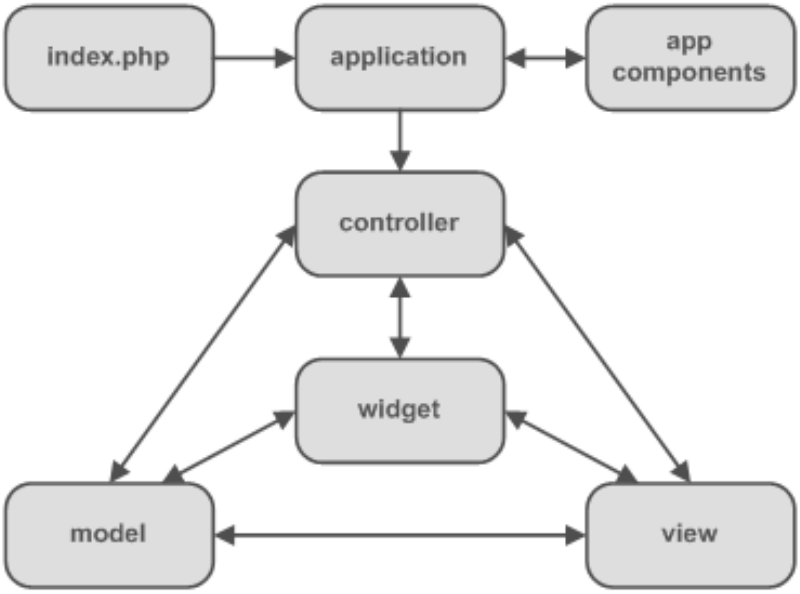


Рисунок 4.2 – Діаграма взаємодії компонентів бекенду (MVC) у Yii2

```
class User extends ActiveRecord implements IdentityInterface
{
    public static function tableName(){
        return 'users';
    }

    public function getTeacher()
    {
        return $this->hasOne(Teacher::class, ['id_teacher' =>
'id_teacher']);
    }

    public static function findByUsername($username)
    {
        return static::find()->joinWith('teacher')-
>where(['username' => $username])->one();
    }

    public function validatePassword($password)
    {
        return Yii::$app->getSecurity()-
>validatePassword($password, $this->password_hash);
    }
}
```

Лістинг 4.5 – Ключові моделі системи

Клас User забезпечує:

1) Авторизацію користувачів через інтерфейс IdentityInterface
2) Зв'язок облікового запису з даними викладача
3) Безпечну перевірку паролів через хешування

```
class LoginForm extends Model
{
    public $username;
    public $password;

    public function rules()
    {
        return [
            [['username', 'password'], 'required'],
            ['password', 'validatePassword'],
        ];
    }

    public function validatePassword($attribute, $params)
    {
        if (!$this->hasErrors())
        {
            $user = $this->getUser();
            if (!$user || !$user->validatePassword($this-
>password))
            {
                $this->addError($attribute, Incorrect username
or password');
            }
        }
    }

    public function login(){
        if ($this->validate())
        {
            return Yii::$app->user->login($this->getUser());
        }
        return false;
    }
}
```

Лістинг 4.6 – Система аутентифікації

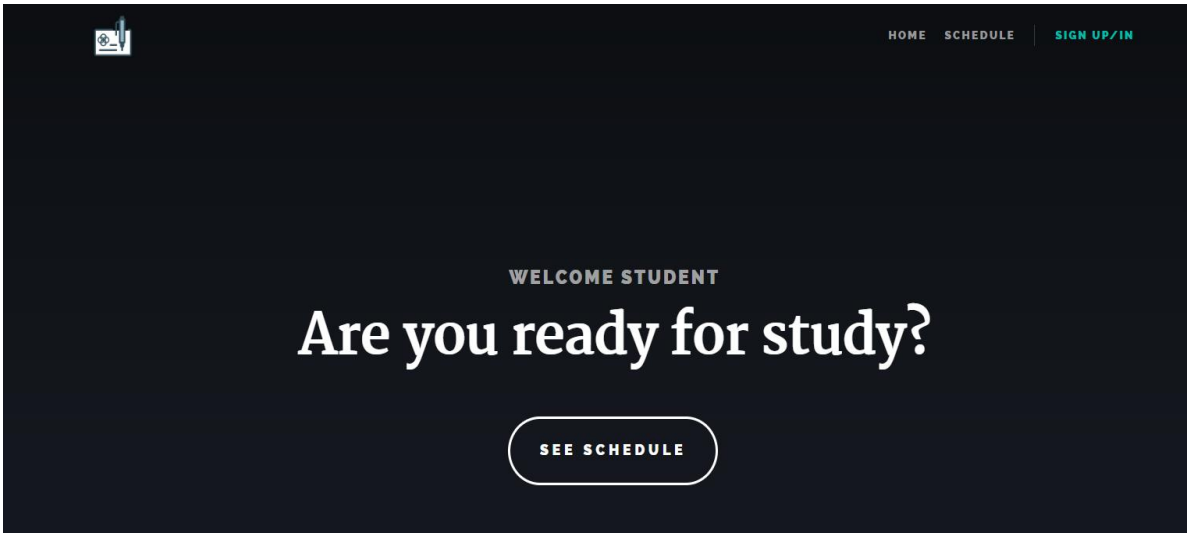


Рисунок 4.3 – Головна сторінка системи

Головна сторінка(рис. 4.3) є вхідною точкою до системи та містить:

1) Навігаційне меню з основним функціоналом
2) Доступ до розкладу без авторизації
3) Кнопки входу для викладачів та адміністраторів

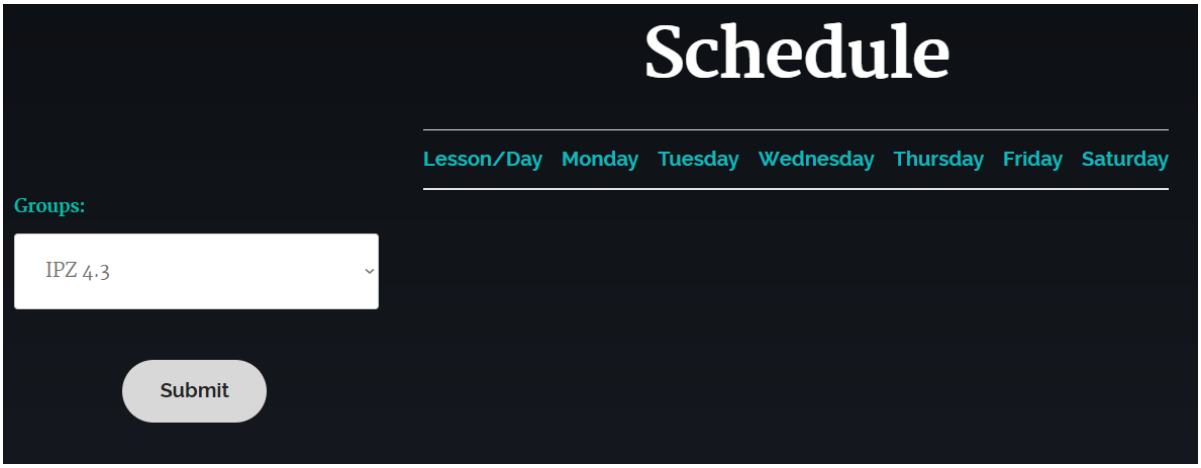


Рисунок 4.4 – Сторінка перегляду розкладу з вибором групи

Інтерфейс перегляду розкладу(рис. 4.4) включає:

1) Випадаючий список для вибору академічної групи
2) Кнопку підтвердження вибору
3) Область для відображення результату

```
class ScheduleController extends Controller
{
    public function actionIndex()
    {
        $model = new ScheduleForm();

        if ($model->load(Yii::$app->request->post()) && $model-
>validate()) {
            $schedule = Schedule::find()
                ->where(['id_group' => $model->group_id])
                ->orderBy(['day_of_week' => SORT_ASC, 'lesson'
=> SORT_ASC])
                ->all();

            return $this->render('view', [
                'schedule' => $schedule,
                'group' => Groups::findOne($model->group_id)
            ]);
        }

        return $this->render('index', ['model' => $model]);
    }
}
```

Лістинг 4.7 – Реалізація механізму вибірки та візуалізації розкладу у контролері ScheduleController

Сформований розклад(рис. 4.5) відображається у вигляді таблиці з:

1) Днями тижня по горизонталі
2) Парами по вертикалі
3) Детальною інформацією про кожне заняття

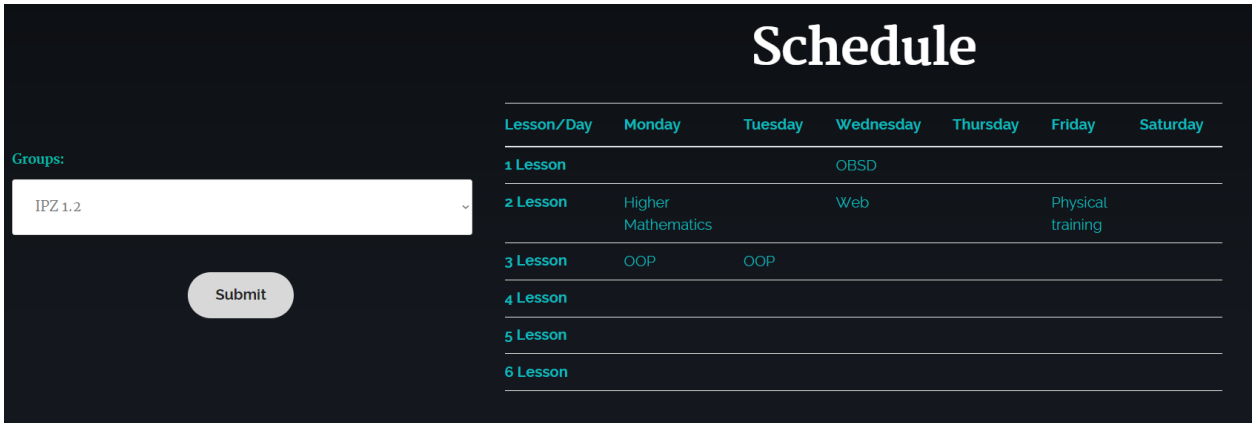
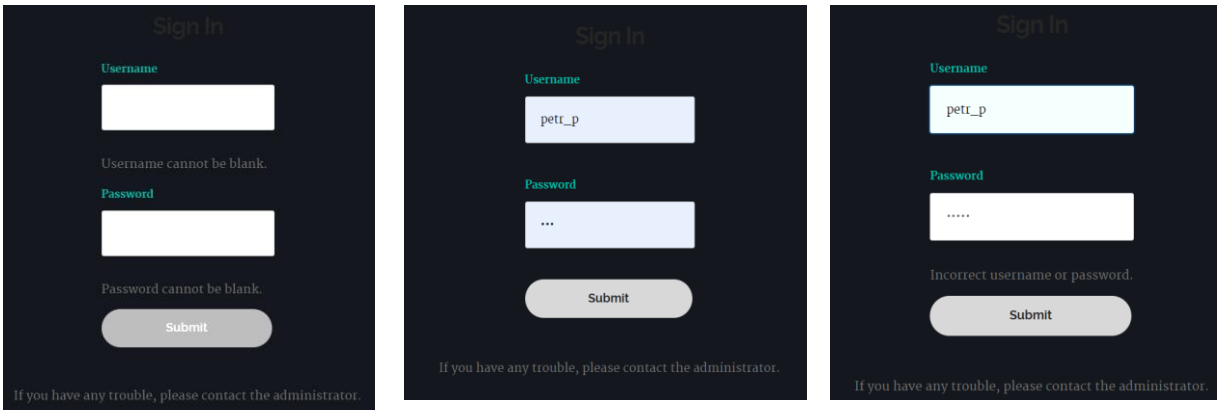


Рисунок 4.5 – Відображення розкладу обраної групи

Система авторизації(рис. 4.6) забезпечує:

1) Перевірку заповнення обов'язкових полів
2) Валідацію правильності даних
3) Виведення зрозумілих повідомлень про помилки

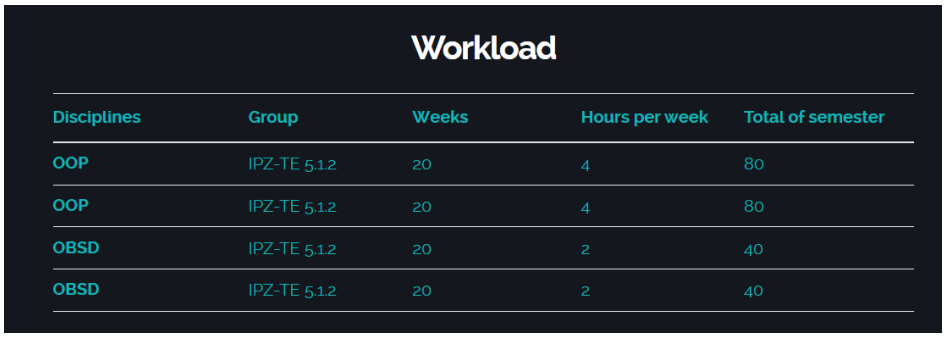


А) Б) В)

Рисунок 4.6– Форма авторизації з валідацією

Після успішного входу викладач отримує доступ(рис. 4.7) до:

1) Особистого розкладу занять
2) Детального навантаження по дисциплінах
3) Інструментів перегляду та фільтрації



Workload

Disciplines	Group	Weeks	Hours per week	Total of semester
OOP	IPZ-TE 5.1.2	20	4	80
OOP	IPZ-TE 5.1.2	20	4	80
OBSD	IPZ-TE 5.1.2	20	2	40
OBSD	IPZ-TE 5.1.2	20	2	40

Рисунок 4.7 – Особистий кабінет викладача з навантаженням

Навігаційна система (рис. 4.8) динамічно змінюється залежно від ролі:

1) Для викладача: кнопки «Навантаження», «Мій розклад»

2) Для адміністратора: повний доступ до редагування

3. Для студента: лише перегляд розкладу

4.2.3. Адміністративний інтерфейс

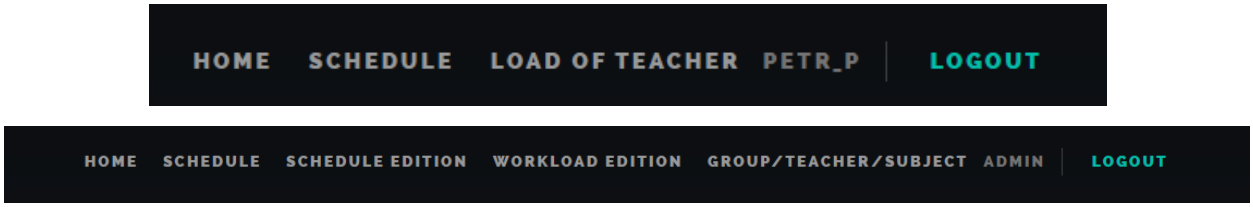


Рисунок 4.8 – Адаптивне меню для різних ролей користувачів

Адміністративний інтерфейс(рис. 4.9) включає:

1) Вибір групи для редагування
2) Табличне представлення розкладу
3) Інструменти зміни часу, аудиторії та викладача
4) Миттєве збереження змін

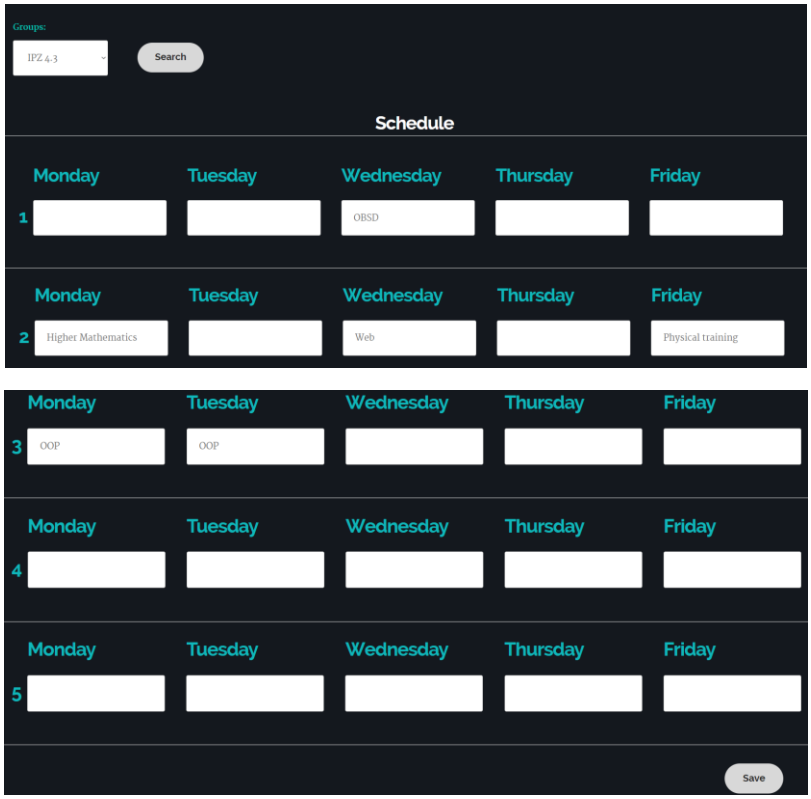


Рисунок 4.9 – Сторінка редагування розкладу

```
class AdminScheduleController extends Controller
{
    public function actionEdit($group_id = null)
    {
        $this->checkAdminAccess();
                $schedule = Schedule::find()
            ->where(['id_group' => $group_id])
            ->indexBy('id')
            ->all();

        if (Yii::$app->request->isPost) {
            foreach (Yii::$app->request->post('Schedule', []) as
$id => $attributes) {
                if (isset($schedule[$id])) {
                    $schedule[$id]->load($attributes, '');
                    $schedule[$id]->save();
                }
            }
            Yii::$app->session->setFlash('success', 'Schedule
updated');
        }
        return $this->render('edit', [
            'schedule' => $schedule,
            'group' => Groups::findOne($group_id)
        ]);
    }
}
```

Лістинг 4.8 – Контролер адміністрування розкладу з підтримкою масового оновлення записів через ActiveRecord-моделі

Модуль редагування навантаження(рис. 4.10) дозволяє:

1) Вибирати викладача зі списку
2) Переглядати поточне навантаження
3) Додавати нові дисципліни
4) Коригувати кількість годин

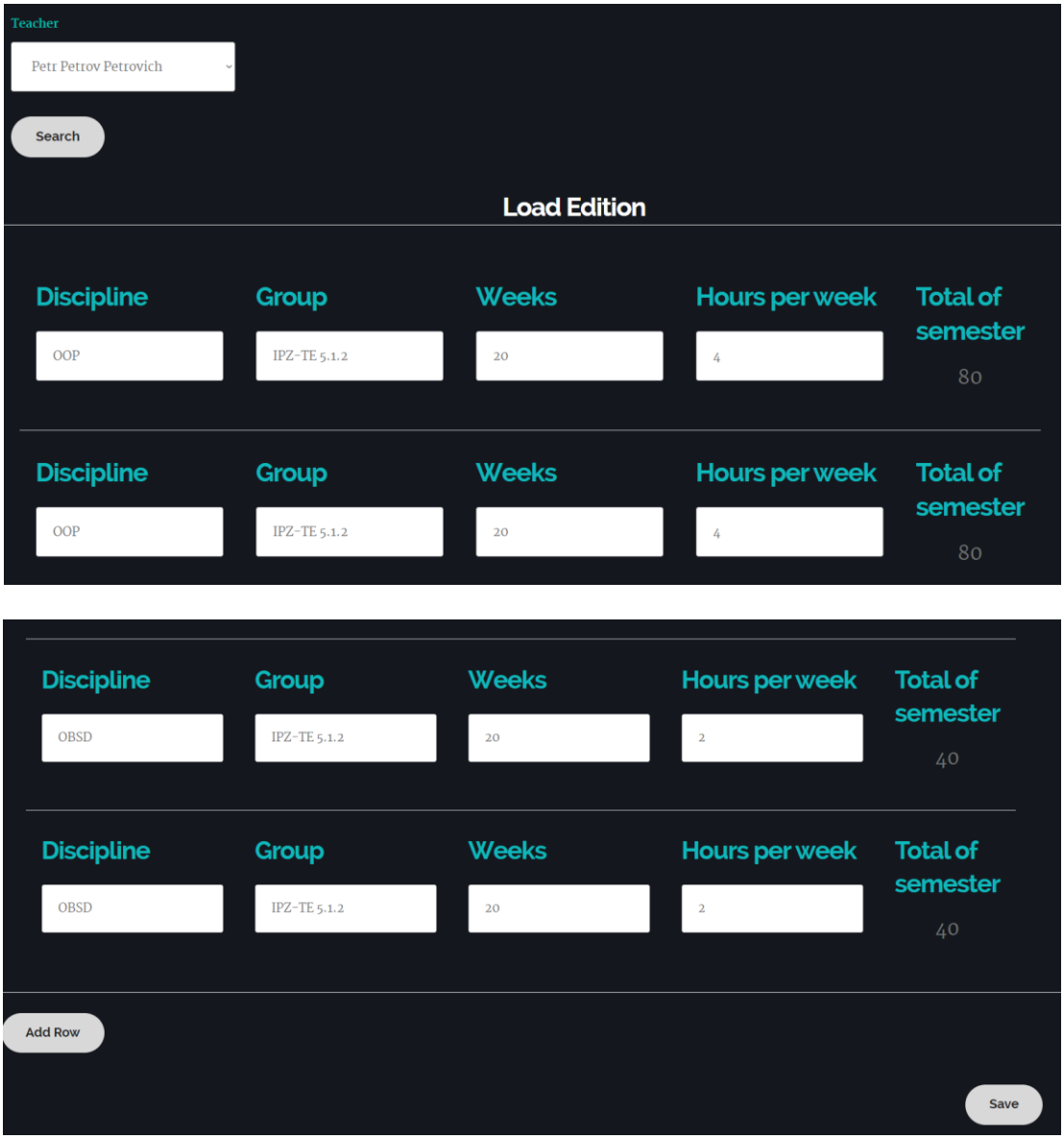


Рисунок 4.10 – Управління навантаженням викладачів

Централізоване управління даними(рис. 4.11) включає три основні модулі:

1) Дисципліни – додавання, редагування, видалення предметів
2) Викладачі – управління персоналом викладачів
3) Групи – керування академічними групами

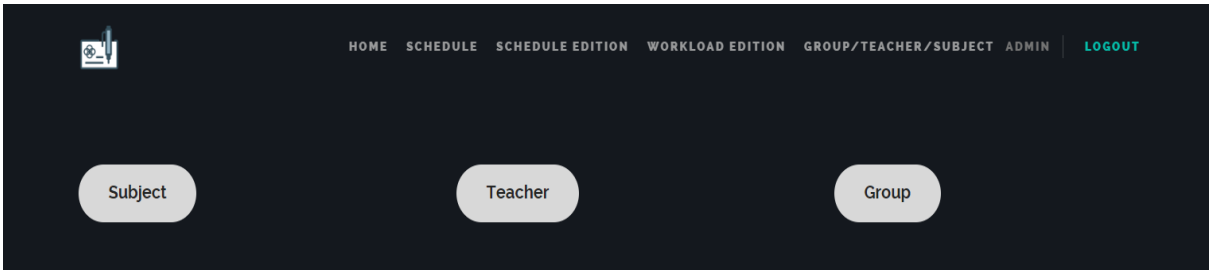


Рисунок 4.11 – Центр управління довідниковими даними

Інтерфейс додавання дисциплін(рис. 4.12) містить:

1) Поле для введення назви предмету
2) Кнопки збереження та перегляду всіх записів
3) Валідацію унікальності назви

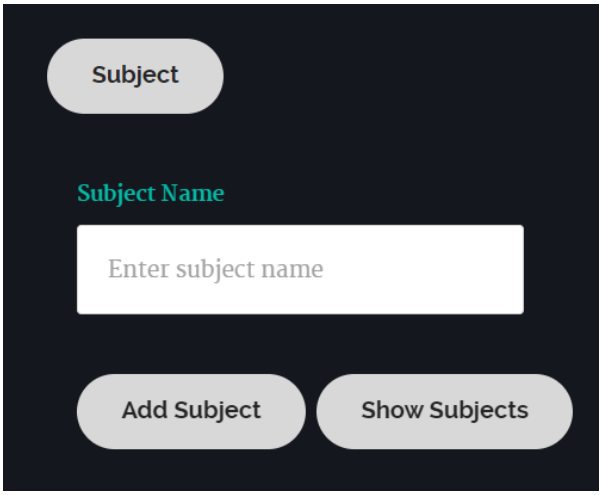


Рисунок 4.12 – Форма додавання нової дисципліни

Табличне представлення(рис. 4.13) дає можливість:

1) Швидкого перегляду всіх записів
2) Сортування за назвою
3) Видалення непотрібних записів

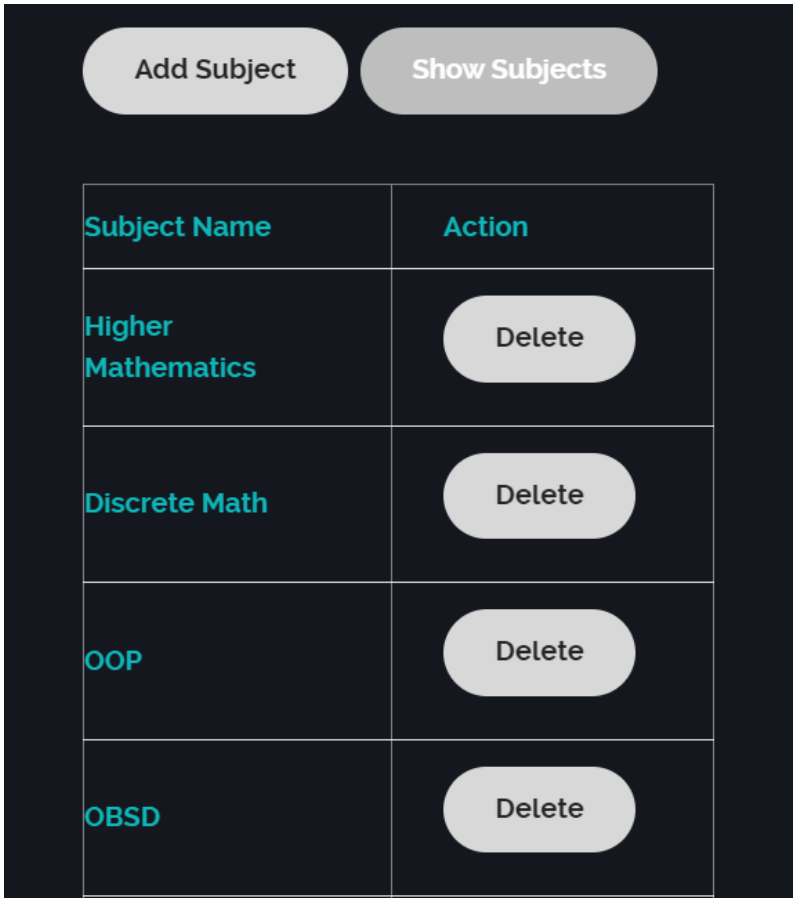


Рисунок 4.13 – Таблиця з переліком дисциплін

Модуль управління викладачами(рис. 4.14) забезпечує

1) Додавання нових викладачів з повною інформацією
2) Редагування існуючих даних
3) Формування облікових записів для доступу

Система управління групами(рис. 4.15) включає:

1) Форму додавання нових груп
2) Прив'язку до спеціальностей
3) Табличний перегляд з можливістю видалення

Взаємодія компонентів відбувається за схемою:

1) Користувач взаємодіє з інтерфейсом
2) Браузер відправляє запит на сервер
3) Yii2 маршрутизує запит до відповідного контролера
4) Контролер виконує бізнес-логіку через моделі
5) Результат повертається у вигляді HTML або JSON

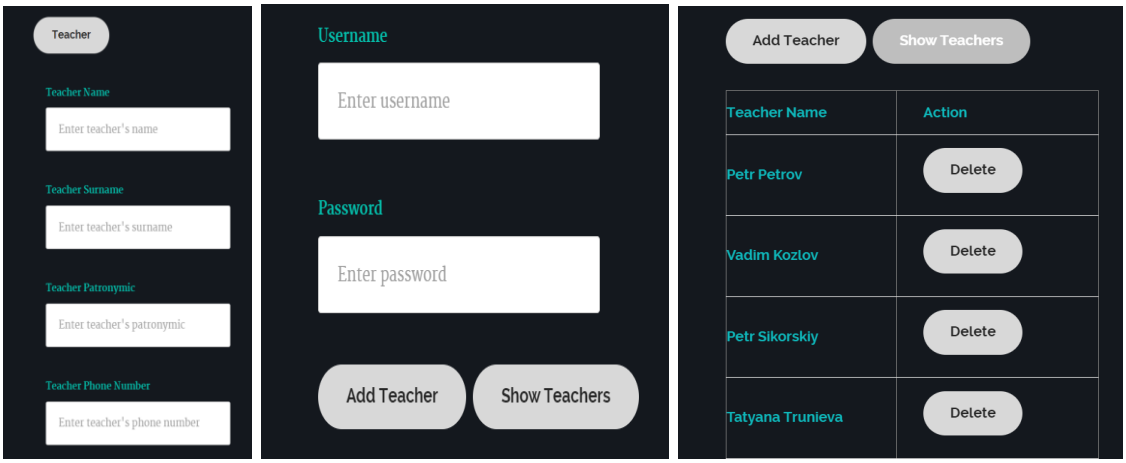


Рисунок 4.14 – Управління даними викладачів

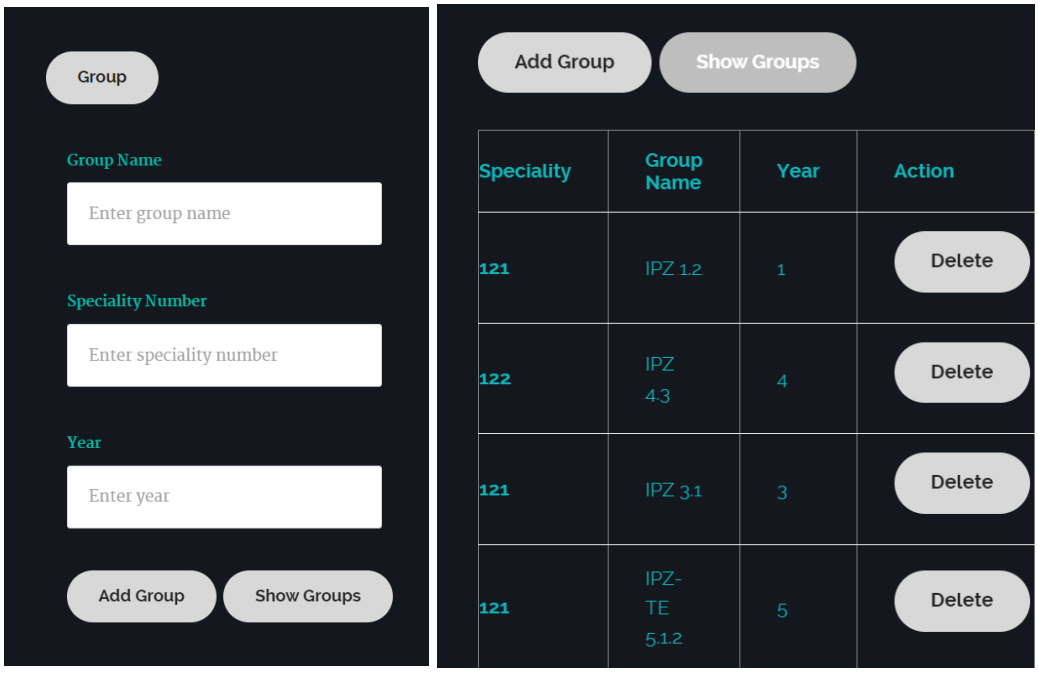


Рисунок 4.15 – Керування академічними групами

Архітектура системи забезпечує:

Масштабованість: модульна структура дозволяє легко додавати новий функціонал

Безпеку: багаторівнева система аутентифікації та авторизації

Продуктивність: оптимізовані запити до бази даних

Зручність: адаптивний інтерфейс для всіх типів пристроїв

```
class SiteController extends Controller
{
    public function actionSchedule()
    {
        $request = Yii::$app->request;
        $group_id = $request->post('group_id');

        if (!Yii::$app->user->isGuest) {
            $user = Yii::$app->user->identity;
        }

        $schedule = Schedule::getGroupSchedule($group_id);
        return $this->render('schedule', [
            'schedule' => $schedule,
            'group' => Groups::findOne($group_id)
        ]);
    }
}
```

Лістинг 4.9 – Метод контролера сайту, що забезпечує отримання та відображення розкладу вибраної групи

Розробка серверної та клієнтської частин системи демонструє ефективне поєднання потужного бекенду на Yii2 з сучасним інтерфейсом користувача. Система забезпечує повний цикл роботи з розкладом: від публічного перегляду студентами до повного адміністративного контролю. Архітектурні рішення, такі як розділення за ролями, модульна структура та оптимізована робота з даними, забезпечили створення надійного та зручного інструменту для автоматизації навчального процесу в коледжі.

4.3 Тестування функціональності та ефективності системи

Тестування було завершальним та критично важливим етапом розробки, метою якого була перевірка відповідності системи всім вимогам, встановленим на етапі проектування. Комплексний підхід до тестування забезпечив впевненість у стабільності, безпеці та ефективності системи перед її впровадженням в реальний навчальний процес.

Для забезпечення якісного покриття була прийнята комбінована

стратегія, що включала різні типи тестування на кожному етапі життєвого циклу розробки. Базовою методологією стало тестування на основі сценаріїв використання, оскільки воно дозволяє перевіряти систему з точки зору кінцевого користувача.

Основним інструментом автоматизації модульних та функціональних тестів для бекенду був обраний Codeception у поєднанні з фреймворком PHPUnit, що є стандартом для екосистеми Yii2. Це дозволило створювати, виконувати та автоматизувати регресивні тести(табл. 4.1). Для тестування продуктивності та навантаження використовувались інструменти Apache JMeter та вбудований Yii2 Debug Toolbar.

На цьому етапі перевірялася коректність роботи ядра системи: моделей даних, бізнес-логіки та ключових алгоритмів.

Тестування моделей (ActiveRecord): для кожної моделі (User, Teacher, Schedule, CurriculumData) були написані модульні тести, що перевіряють:

Коректність валідації даних (наприклад, обов'язковість полів, формат семестру).

Працездатність зв'язків між моделями (наприклад, чи правильно метод getTeacher() повертає зв'язаний об'єкт).

Роботу основних CRUD-операцій (створення, читання, оновлення, видалення).

Окремо тестувалися складні запити та функції бази даних,. Наприклад, функція teacher_load(teacher_id) перевірялася на коректність агрегації даних про години та предмети для конкретного викладача. Тест перевіряв, чи кількість рядків у результаті відповідає очікуваній, і чи коректно розраховується загальне навантаження.

Таблиця 4.1 – Результати тестування

Тип тестування	Об’єкт тестування	Мета	Основні інструменти/Методи
Модульне (Unit)	Окремі класи та методи (моделі, сервіси).	Перевірити коректність найменших логічних одиниць.	PHPUnit, Codeception Unit.
Функціональне (Functional)	Окремі сценарії використання (користувацькі історії).	Перевірити, чи функція системи працює відповідно до специфікації.	Codeception Functional, сценарії в браузері.
Приймальне (Acceptance)	Система в цілому з точки зору користувача.	Підтвердити, що система готова до використання.	Ручне тестування за чек-листами, Codeception Acceptance.
Навантажувальне (Load)	Продуктивність системи під навантаженням.	Визначити межі продуктивності та час відгуку.	Apache JMeter, симуляція паралельних запитів.
Безпеки (Security)	Механізми автентифікації, авторизації, валідації.	Виявити вразливості та перевірити захист даних.	Аналіз коду, тестування на SQL-ін'єкції, XSS.

```
class CurriculumDataTest extends \Codeception\Test\Unit
{
    public function testValidation()
    {
        $model = new CurriculumData();

        $model->id_of_curriculum = 1;
        $model->subject_id = 5;
        $model->semester = 5;
```

```
        $this->assertFalse($model->validate());
        $this->assertArrayHasKey('semester', $model->errors);
    }

    public function testTeacherRelation()
    {

        $curriculumData = CurriculumData::findOne(1);
        $teacher = $curriculumData->teacher;
        $this->assertInstanceOf(Teacher::class, $teacher);
        $this->assertEquals($curriculumData->id_teacher,
$teacher->id_teacher);
    }
}
```

Лістинг 4.11 – Приклад модульного тесту для моделі CurriculumData

Для кожної з трьох ролей був сформований набір критичних шляхів (happy path):

Студент:

1) Вибір групи зі списку на головній сторінці.
2) Відображення розкладу у вигляді таблиці
3) Тест негативного сценарію: спроба вибору неіснуючої групи.

Викладач:

1) Успішна авторизація через форму LoginForm.
2) Перехід на сторінку навантаження (/teacher/load).
3) Коректність відображення персональних даних (прізвище, ім’я) та таблиці навантаження.

Вихід з системи (logout).

Адміністратор:

Повний цикл CRUD для довідників (додавання, перегляд, видалення дисципліни).

Редагування розкладу групи: зміна викладача для конкретного заняття та збереження змін.

Генерація нового розкладу з використанням алгоритму оптимізації та перевірка відсутності конфліктів.

Тестування інтерфейсу (UI) та юзабіліті: проводилося ручне тестування для перевірки адаптивності верстки на різних розмірах екрану, зручності навігації, зрозумілості повідомлень про помилки та підтверджень дій. Особлива увага приділялася інтуїтивності адміністративних інтерфейсів редагування.

Тестування продуктивності та безпеки

Навантажувальне тестування: було імітовано роботу системи в умовах, близьких до реальних(табл. 4.2). За допомогою JMeter створювалися сценарії, де десятки віртуальних користувачів одночасно:

1) Запитують розклад для популярних груп.
2) Авторизуються в системі (навантаження на механізм аутентифікації).
3) Виконують пошук у великих довідниках (викладачі, дисципліни).

Таблиця 4.2 – Результати тестування продуктивності ключових операцій

Операція	Середній час відгуку (мс)	Макс. час відгуку при навантаженні (100 корист.)	Примітки
Завантаження головної сторінки	121	450	Задовільно
Відображення розкладу групи	220	809	Залежить від кількості занять
Авторизація (логін)	301	1192	Обґрунтовано використанням хешування
Генерація розкладу (алгоритм)	15000	N/A	Виконується в фоні, результат кешується

Загалом, процес тестування показав високу стабільність та відповідність системи вимогам. Усі критичні функціональні сценарії виконані успішно.

Алгоритм генерації розкладу продемонстрував адекватну продуктивність для потреб коледжу, генеруючи розклад без конфліктів за 15-20 секунд.

Було виявлено та виправлено кілька незначних помилок, переважно пов’язаних з валідацією крайніх значень у формах та оптимізацією окремих SQL-запитів для великих обсягів даних. Система показала хорошу стійкість під навантаженням, а час відгуку ключових операцій знаходиться в прийнятних межах.

Проведений комплекс тестів підтвердив готовність системи автоматизації складання розкладу до впровадження в експлуатацію. Система є функціонально повною, безпечною, зрозумілою для користувачів різних категорій та здатною ефективно працювати в умовах реального навчального закладу. Результати тестування стали підставою для формування позитивного висновку щодо якості розробки в цілому.

ВИСНОВКИ

Проведене дослідження та розробка були спрямовані на створення сучасної, ефективної системи автоматизації формування навчального розкладу для закладів вищої освіти. В результаті виконаної роботи було досягнуто поставленої мети та отримано низку важливих теоретичних і практичних результатів.

У розділі 1 було проведено глибокий аналіз предметної області, що дозволило чітко формалізувати задачу автоматизації складання розкладу. Було визначено основні компоненти моделі (групи, викладачі, аудиторії, дисципліни, часові слоти), сформовано систему жорстких та м'яких обмежень та визначено цільову функцію, що враховує мінімізацію конфліктів, «вікон» та рівномірність навантаження. Детальний аналіз існуючих рішень підтвердив актуальність розробки спеціалізованої, гнучкої системи, адаптованої під специфіку вітчизняних навчальних закладів. Крім того, були чітко визначені функціональні та нефункціональні вимоги до системи, а також ролі трьох основних категорій користувачів: адміністратора, викладача та студента.

У розділі 2 було обґрунтовано вибір сучасного технологічного стеку для реалізації. В якості основи бекенду обрано фреймворк Yii2, що забезпечує високу продуктивність, безпеку та швидку розробку за рахунок архітектури MVC та потужного ActiveRecord. Для роботи з даними обрано СУБД PostgreSQL, що гарантує надійність, підтримку складних зв'язків та цілісність даних. Для інтерфейсу користувача використано Bootstrap у поєднанні з можливостями Yii2, що дозволило створити адаптивний, інтуїтивний фронтенд. Обраний стек технологій повністю відповів усім сформованим вимогам.

У розділі 3 було розроблено архітектурні та дизайн-рішення майбутньої системи. Була запропонована трирівнева клієнт-серверна архітектура, що

забезпечує масштабованість, безпеку та розділення відповідальності між рівнями даних, бізнес-логіки та представлення. Ключовим інноваційним елементом архітектури стало проектування гібридної логіки оптимізації на основі методів штучного інтелекту. Для вирішення задачі була запропонована комбінація генетичного алгоритму для первинної генерації розкладу, метаевристичних методів для пост-оптимізації, а також системи логічних правил для гарантії дотримання жорстких обмежень. Це забезпечило не лише пошук життєздатних рішень, але й здатність системи знаходити якісні, збалансовані розклади. На цій основі було спроектовано логічну модель даних, нормалізовану до BCNF, яка включає всі необхідні сутності (speciality, groups, curriculum, subject, teacher, curriculum_data, schedule тощо) та зв'язки між ними. Крім того, було детально розроблено концепцію інтерфейсу користувача (UI/UX) з окремими ієрархіями сторінок для кожної ролі, що забезпечило зручну та ефективну навігацію.

У розділі 4 була повністю реалізована та всебічно протестована робоча система. Практична реалізація включила не лише створення інфраструктури, але й успішну імплементацію та інтеграцію запропонованих алгоритмів ШІ в ядро системи. Конкретні кроки включали:

1) Створення та наповнення бази даних шляхом написання міграцій Yii2 та seed-даних, що сформувало надійну основу для роботи алгоритмів оптимізації.

2) Розробку серверної частини на Yii2, де, окрім моделей ActiveRecord та контролерів, було реалізовано спеціалізовані сервісні класи для генетичного алгоритму та інших методів оптимізації, які ефективно взаємодіють з моделями даних.

3) Розробку клієнтської частини – інтуїтивного веб-інтерфейсу, який надає адміністратору зручний доступ до запуску процесу оптимізації та перегляду його результатів.

4) Комплексне тестування, яке підтвердило не лише функціональну повноту системи, але й ефективність реалізованих методів ШІ. Тестування

продуктивності показало, що система здатна генерувати оптимальний розклад для типового навчального закладу за прийнятний час (15-20 секунд), гарантуючи дотримання всіх обмежень та високий рівень якості розкладу за обраними критеріями.

Узагальнюючи, в рамках даної роботи успішно розроблено та впроваджено повнофункціональну систему автоматизованого формування навчального розкладу, яка використовує гібридний підхід на основі методів штучного інтелекту для вирішення ключової оптимізаційної задачі. Система інтегрує ефективні алгоритмічні рішення, сучасну трирівневу архітектуру, нормалізовану базу даних та зручний рольовий інтерфейс. Гнучка архітектура, потужний технологічний стек та успішно реалізоване ядро оптимізації створюють міцну основу для подальшого масштабування та вдосконалення системи. Перспективи включають впровадження додаткових евристик, інтеграцію методів машинного навчання для прогнозування навантаження, покращення інтерфейсу користувача або розширення мобільним додатком. Результати роботи демонструють готовність системи до експлуатації в реальному навчальному середовищі з метою значного підвищення ефективності управління навчальним процесом.

.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. FET (Free Timetabling Software) [Електронний ресурс]. – Режим доступу: https://lalescu.ro/liviu/fet/
2. UniTime: A Comprehensive Educational Scheduling System // Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling. – 2004.
3. Технологія еволюційного формування розкладів у закладах вищої освіти [Електронний ресурс]. – Режим доступу: https://www.academia.edu/98898889/Технологія_еволюційного_формування_розкладів_у_закладах_вищої_освіти
4. Рекомендації щодо складання розкладу уроків [Електронний ресурс]. – Режим доступу: http://profosvitamk.org/index.php?option=com_content&view=article&id=623:2015-09-24-13-20-57&catid=7:2011-06-22-11-10-48&Itemid=7
5. PHP vs Python – порівняння. Nethues [Електронний ресурс]. – Режим доступу: https://www.nethues.com/blog/php-vs-python
6. Про організацію 2025/2026 навчального року в закладах загальної середньої освіти (Лист № 1/17526-25) [Електронний ресурс] / Міністерство освіти і науки України. – 2025. – Режим доступу: https://osvita.ua/legislation/Ser_osv/95314/
7. МОН: організація навчання в школах потребує все більшої гнучкості [Електронний ресурс] / Міністерство освіти і науки України. – Освіта.ua, 2025. – Режим доступу: https://osvita.ua/school/95936/
8. Контролер представлення моделі [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Glossary/MVC
9. Конноллі Т., Бегг К. Бази даних: проєктування, реалізація та керування. Теорія та практика. 7-ме вид. – К.: Вільямс, 2020.
10. Garcia-Molina H., Ullman J. D., Widom J. Database Systems: The Complete Book. – 2nd ed. – Pearson, 2020.

11. The Definite Guide to Yii 2.0 [Електронний ресурс]. – Режим доступу: https://www.yiiframework.com/doc/guide/2.0/en/
12. PostgreSQL 14.8 Documentation [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/14/index.html
13. Малахов Є. В. Проєктування баз даних. Лекційні матеріали [Електронний ресурс]. – Режим доступу: https://e-learning.suitt.edu.ua/course/view.php?id=515
14. PHP Documentation [Електронний ресурс]. – Режим доступу: https://www.php.net/docs.php
15. JavaScript Documentation [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript
16. Pillay N. A survey of school timetabling research // Annals of Operations Research. – 2014.
17. Lewis R. A Survey of Metaheuristic-Based Techniques for University Timetabling Problems // Expert Systems with Applications. – 2021.
18. Sörensen K., Glover F. Metaheuristics // Encyclopedia of Operations Research and Management Science. – 2013.
19. De Werra D. An introduction to timetabling // European Journal of Operational Research. – 1985.
20. Burke E. K., Petrovic S. Recent research directions in automated timetabling // European Journal of Operational Research. – 2002.
21. Abdullah S., Turabieh H. Generating university course timetable using genetic algorithms and local search // 3rd International Conference on Computing and Informatics. – 2012.
22. Fahl S., Prinz K., Kullenkötter B. Systematic review on machine learning (ML) methods for manufacturing processes – Identifying artificial intelligence (AI) methods for field application // Procedia CIRP. – 2020. – № 93. – P. 413–418.
23. Mata J., de Miguel I., Durán R. H. Artificial intelligence (AI) methods in optical networks: A comprehensive survey // Optical Switching and Networking. – 2018. – № 28. – С. 43–57.