МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЛЕСІ УКРАЇНКИ

Кафедра загальної математики та методики навчання інформатики

На правах рукопису

**КОВАЛЬЧУК ВЛАДИСЛАВ ВАЛЕРІЙОВИЧ**

# РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ДИНАМІЧНОГО ПЕРЕГЛЯДУ ТА АДМІНІСТРУВАННЯ РОЗКЛАДУ ЗАНЯТЬ НАВЧАЛЬНОГО ЗАКЛАДУ

Спеціальність: 014 Середня освіта (Інформатика)
Освітньо-професійна програма: «Середня освіта. Інформатика»
Кваліфікаційна робота на здобуття освітнього ступеня «магістр»

Науковий керівник:
**ФЕДОНЮК АНАТОЛІЙ АНАНІЙОВИЧ,**
кандидат фізико-математичних наук, доцент кафедри загальної математики та методики навчання інформатики

РЕКОМЕНДОВАНО ДО ЗАХИСТУ
Протокол №_____
засідання кафедри загальної математики та методики навчання інформатики
від « ___ » _____ 2025 р.
Завідувач кафедри

(_____) доц. Хомяк М. Я.

ЛУЦЬК − 2025

**ЗМІСТ**

# ВСТУП

**Актуальність теми.** В умовах цифрової трансформації освіти критично важливим є забезпечення оперативного доступу учасників навчального процесу до актуальної інформації. Одним із ключових елементів організації навчання є розклад занять. Від якості його планування, зручності відображення та швидкості оновлення залежить ефективність роботи всього навчального закладу.

Більшість українських закладів освіти досі використовують застарілі підходи до адміністрування розкладу: паперові носії, статичні файли Excel або PDF, що публікуються на сайтах. Такі методи мають суттєві недоліки: низьку оперативність внесення змін (заміни, перенесення пар), відсутність адаптивності для мобільних пристроїв та складність у підтримці актуальності даних для всіх користувачів одночасно. Існуючі автоматизовані системи (наприклад, Moodle або «Деканат») часто є надто громіздкими, дорогими у впровадженні або мають обмежений функціонал саме у частині візуалізації розкладу.

У зв’язку з цим виникає нагальна потреба у розробці спеціалізованих вебзастосунків, які поєднують простоту використання, низькі вимоги до апаратного забезпечення та можливість динамічного оновлення даних. Використання сучасних PHP-фреймворків, зокрема Laravel, дозволяє створювати надійні, масштабовані та безпечні системи, які легко розгортаються на доступних хостингових платформах.

Таким чином, розробка вебзастосунку для динамічного перегляду та адміністрування розкладу, який забезпечує миттєву синхронізацію даних між адміністратором та користувачами, є актуальним науково-прикладним завданням.

**Мета дослідження** полягає у підвищенні ефективності управління навчальним процесом шляхом проєктування, програмної реалізації та впровадження вебзастосунку для динамічного адміністрування та перегляду

розкладу занять на основі фреймворку Laravel.

Для досягнення поставленої мети необхідно вирішити такі **завдання**:

1. Проаналізувати існуючі підходи та програмні засоби для складання й адміністрування навчальних розкладів, виявити їхні переваги та недоліки.
2. Обґрунтувати вибір технологічного стека (PHP, Laravel, MySQL) для реалізації серверної частини та інтерфейсу користувача.
3. Розробити архітектуру бази даних та інформаційну модель системи, що забезпечує цілісність даних та уникнення колізій (накладання аудиторій чи викладачів).
4. Спроєктувати та програмно реалізувати вебзастосунок, який включає адміністративну панель для управління ресурсами та публічний інтерфейс для перегляду розкладу.
5. Реалізувати механізм динамічного відображення подій календаря без повного перезавантаження сторінки.
6. Виконати розгортання (deployment) розробленого вебзастосунку на реальному хостингу в мережі Інтернет та провести тестування його працездатності.

**Об’єкт дослідження** – процес управління розкладом навчальних занять у закладах вищої освіти з використанням вебтехнологій.

**Предмет дослідження** – методи та програмні засоби створення динамічних вебзастосунків для адміністрування освітнього процесу на основі MVC-архітектури.

**Методи дослідження.** У роботі використано комплекс методів:

– *теоретичні* – аналіз наукової літератури та документації для вибору інструментальних засобів розробки;

– *емпіричні* – моделювання бізнес-процесів предметної області, проєктування схеми бази даних (ER-діаграми);

– *експериментальні* – тестування програмного продукту, перевірка коректності роботи на сервері, оцінка швидкодії вебзастосунку.

**Наукова новизна одержаних результатів** полягає в удосконаленні

архітектурного підходу до побудови бюджетних систем управління розкладом, який, на відміну від існуючих аналогів, базується на легкому технологічному стеку (Laravel) і забезпечує можливість розгортання на загальнодоступних shared-хостингах без втрати функціональності та швидкодії.

**Практичне значення одержаних результатів.** Розроблено повнофункціональний вебзастосунок «School Timetable», який дозволяє:

- адміністраторам – керувати базою викладачів, аудиторій та занять, автоматично уникаючи конфліктів у розкладі;
- студентам та викладачам – переглядати актуальний розклад у зручному форматі календаря з будь-якого пристрою. Програмний продукт розгорнуто на хостингу, він є повністю готовим до експлуатації.

Результати роботи можуть бути використані для автоматизації навчального відділу будь-якого освітнього закладу.

**Апробація результатів дослідження** відбулася на базі Луцького ліцею № 9 Луцької міської ради Волинської області під час педагогічної та переддипломної практик.

Результати дослідження були представлені на:

- XIV Міжнародній науково-практичній конференції «Математика. Інформаційні технології. Освіта» (13–15 червня 2025 р., м. Луцьк – с. Світязь);
- V Міжнародній науковій конференції «Технології та суспільство: взаємодія, вплив, трансформація» (12 грудня 2025 р., м. Кропивницький).

1. Ковальчук В. В., Федонюк А. А. Розробка вебзастосунку для динамічного перегляду та адміністрування розкладу занять навчального закладу. *XIV Міжнародна науково-практична конференція «Математика. Інформаційні технології. Освіта»* : тези доповідей (Луцьк–Світязь, 13–15 черв. 2025 р.). Луцьк : ВНУ ім. Лесі Українки, 2025. С. 105.

2. Ковальчук В. В., Федонюк А. А. Розробка вебзастосунку для динамічного перегляду та адміністрування розкладу занять навчального закладу. *Технології та суспільство: взаємодія, вплив, трансформація* : зб. з матеріалами V Міжнар. наук. конф. (м. Кропивницький, 12 груд. 2025 р.), 2025. С. 266.

# РОЗДІЛ 1
# ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ЗАСАДИ СТВОРЕННЯ СИСТЕМ УПРАВЛІННЯ НАВЧАЛЬНИМ РОЗКЛАДОМ

## 1.1. Аналіз проблеми складання та адміністрування розкладу у закладах освіти

Організація навчального процесу в сучасних закладах освіти є багатогранною системою, стабільність якої безпосередньо залежить від якості планування часових та просторових ресурсів. Розклад занять виступає не просто графіком відвідування, а центральним регуляторним механізмом, що синхронізує роботу викладацького складу, навчальну діяльність студентів та використання аудиторного фонду. В умовах стрімкої цифровізації суспільства та переходу до змішаних форм навчання вимоги до цього механізму суттєво зросли: він повинен забезпечувати не лише статичне відображення інформації, але й динамічну адаптацію до змін у реальному часі. Проте аналіз поточної ситуації у більшості українських закладів освіти свідчить про наявність системної кризи в методах адміністрування розкладу, яка має як глибоке математичне підґрунтя, так і виражені організаційно-технічні вади. [1]

Фундаментальна складність побудови ідеального розкладу полягає в його математичній природі. У теорії розкладів та дослідженні операцій задача складання університетського розкладу (University Course Timetabling Problem — UCTP) класифікується як NP-повна задача (NP-complete). Термін «NP-повна» означає, що обчислювальна складність задачі зростає експоненціально залежно від кількості вхідних змінних. Якщо для малого навчального закладу з десятком груп і викладачів оптимальний варіант можна знайти шляхом простого перебору комбінацій, то для середніх та великих закладів кількість можливих варіантів розміщення занять у сітці годин перевищує кількість атомів у відомому Всесвіті. [2]

Математична складність посилюється необхідністю одночасного

задоволення суперечливих вимог, які в науковій літературі поділяють на «жорсткі» (hard constraints) та «м'які» (soft constraints). Жорсткі обмеження є критичними: їх порушення робить розклад фізично неможливим для виконання. До таких обмежень належать фізична неможливість присутності викладача у двох аудиторіях одночасно, накладання занять для однієї академічної групи, а також відповідність місткості аудиторії кількості студентів. Порушення навіть одного з цих правил призводить до зриву навчального процесу. М'які обмеження, у свою чергу, стосуються якості освітнього середовища: мінімізація «вікон» між парами, рівномірність навантаження протягом тижня, врахування методичних днів викладачів. Дослідження показують, що при ручному складанні розкладу диспетчери здатні контролювати виконання жорстких обмежень, однак оптимізація м'яких обмежень часто залишається поза межами можливостей людського мозку через надмірний обсяг даних. [3]

Попри існування теоретичних алгоритмів автоматизації, на практиці значна частина закладів освіти продовжує використовувати архаїчні методи адміністрування. Найбільш розповсюдженим залишається паперовий метод, який, незважаючи на свою простоту, має критичні недоліки в епоху інформаційних технологій. Паперовий розклад, зазвичай розміщений на центральному стенді закладу (Рис. 1.1), створює проблему локальності доступу.

Затверджую
Директор МГ № 20
_____ О.В. Бакаляр

РОЗКЛАД УРОКІВ НА 2024 - 2025 НАВЧАЛЬНИЙ Р

| | | Понеділок | Вівторок | Середа | Четвер | П'ятниця |
|---|---|---|---|---|---|---|
| 9-А | 1 | труд | Зарубіжна літер | Англійська мова | зарубіжна літер | Англійська мова |
| | 2 | географія | Фізика | Математика | Історія | Історія |
| | 3 | Математика | Фізика | Математика | фізика | хімія |
| | 4 | Математика | Інформ / Укр. Мова | Українська літер | Інформ / укр мова | Математика |
| | 5 | фізична культура | Інформ / Укр. Мова | фізична культура | Інформ / укр мова | фізична культура |
| | 6 | Хімія | історія | біологія | ОЗ | Право<br>Географія |
| | 7 | Англійська мова | Мистецтво | право | Українська літер | Біологія |

| | | | | | | |
|---|---|---|---|---|---|---|
| 9-Б | 1 | Математика | історія | фізика | Історія | Історія<br>географія |
| | 2 | Математика | Мистецтво | Фізика | укр мова / англ | хімія |
| | 3 | географія | Українська літер | Укр мова / англ . | укр мова / англ | Математика |
| | 4 | Фізика | фізична культура | Математика | зарубіжна літер | фізична культура |
| | 5 | труд | Зарубіжна літер | Математика | ОЗ | Українська літер |
| | 6 | право | Інформ / німец | Укр мова / англ . | Інформ / німец | Біологія |
| | 7 | Хімія | інформ / німец | біологія | Інформ / німец | Англійська мова |

Рисунок 1.1 – Традиційний спосіб відображення розкладу на паперових носіях

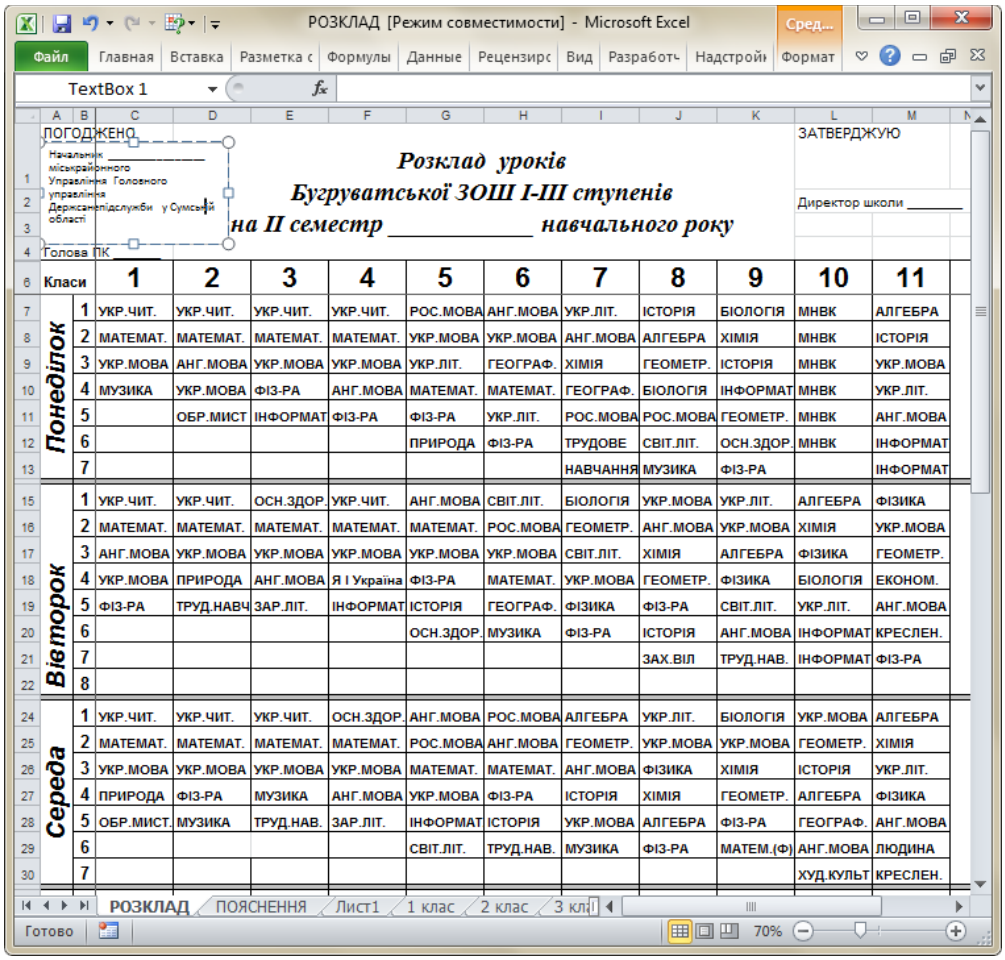
Інформація стає доступною лише тим учасникам освітнього процесу, які фізично знаходяться біля стенду, що унеможливлює оперативне планування часу для студентів та викладачів, які перебувають за межами закладу. Окрім того, паперовий носій є статичним: внесення будь-яких змін вимагає механічного втручання — закреслення, написання поверх старого тексту або повного передруку аркуша, що не лише виглядає неестетично, але й часто призводить до помилок при прочитанні.

Намагаючись модернізувати процес, багато закладів перейшли до використання табличних процесорів, таких як Microsoft Excel. Цей етап цифровізації, безумовно, спростив процес тиражування та друку розкладу, проте не вирішив головної проблеми адміністрування — відсутності цілісності даних. Excel, будучи потужним інструментом для роботи з цифрами, не є спеціалізованою базою даних для розкладів. Він не має вбудованих механізмів логічної валідації, які б автоматично забороняли диспетчеру призначити одну аудиторію двом різним викладачам. У результаті виникають так звані «колізії», які виявляються вже постфактум, коли два викладачі з групами приходять під кабінет одночасно. Це спричиняє зрив занять, пошук вільних приміщень у стресовому режимі та зниження авторитету адміністрації закладу. [4]

Ще однією критичною проблемою використання локальних файлів (Excel, Word) є конфлікт версій. Процес складання розкладу часто є ітеративним, у ньому беруть участь кілька співробітників. Обмін файлами через електронну пошту або месенджери призводить до появи множинних копій документа з назвами на кшталт «Розклад_фінал», «Розклад_виправлений», «Розклад_новий_v2». Відсутність єдиного джерела істини (Single Source of Truth) призводить до ситуацій, коли студенти користуються однією версією графіку, а викладачі — іншою. Такий підхід робить систему вразливою до людського фактору та унеможливлює синхронізацію даних у реальному часі. [5]

Візуальна складова табличних розкладів також є слабким місцем (Рис. 1.2). Величезні масиви даних, скомпоновані у щільні таблиці, важко сприймаються візуально, особливо з екранів мобільних телефонів, які є основним засобом

отримання інформації для сучасного студентства. Необхідність масштабувати зображення, шукати свій рядок серед сотень інших створює зайве когнітивне навантаження та незручності для користувачів.



ПОГОДЖЕНО

ЗАТВЕРДЖУЮ

Розклад уроків
Бугруватської ЗОШ І-ІІІ ступенів
на ІІ семестр _____ навчального року

| | | Класи | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Понеділок | 1 | | УКР.ЧИТ. | УКР.ЧИТ. | УКР.ЧИТ. | УКР.ЧИТ. | РОС.МОВА | АНГ.МОВА | УКР.ЛІТ. | ІСТОРІЯ | БІОЛОГІЯ | МНВК | АЛГЕБРА |
| | 2 | | МАТЕМАТ. | МАТЕМАТ. | МАТЕМАТ. | МАТЕМАТ. | УКР.МОВА | УКР.МОВА | АНГ.МОВА | АЛГЕБРА | ХІМІЯ | МНВК | ІСТОРІЯ |
| | 3 | | УКР.МОВА | АНГ.МОВА | УКР.МОВА | УКР.МОВА | УКР.ЛІТ. | ГЕОГРАФ. | ХІМІЯ | ГЕОМЕТР. | ІСТОРІЯ | МНВК | УКР.МОВА |
| | 4 | | МУЗИКА | УКР.МОВА | ФІЗ-РА | АНГ.МОВА | МАТЕМАТ. | МАТЕМАТ. | ГЕОГРАФ. | БІОЛОГІЯ | ІНФОРМАТ | МНВК | УКР.ЛІТ. |
| | 5 | | | ОБР.МИСТ | ІНФОРМАТ | ФІЗ-РА | ФІЗ-РА | УКР.ЛІТ. | РОС.МОВА | РОС.МОВА | ГЕОМЕТР. | МНВК | АНГ.МОВА |
| | 6 | | | | | | ПРИРОДА | ФІЗ-РА | ТРУДОВЕ | СВІТ.ЛІТ. | ОСН.ЗДОР. | МНВК | ІНФОРМАТ |
| | 7 | | | | | | | | НАВЧАННЯ | МУЗИКА | ФІЗ-РА | | ІНФОРМАТ |
| Вівторок | 1 | | УКР.ЧИТ. | УКР.ЧИТ. | ОСН.ЗДОР. | УКР.ЧИТ. | АНГ.МОВА | СВІТ.ЛІТ. | БІОЛОГІЯ | УКР.МОВА | УКР.ЛІТ. | АЛГЕБРА | ФІЗИКА |
| | 2 | | МАТЕМАТ. | МАТЕМАТ. | МАТЕМАТ. | МАТЕМАТ. | МАТЕМАТ. | РОС.МОВА | ГЕОМЕТР. | АНГ.МОВА | УКР.МОВА | ХІМІЯ | УКР.МОВА |
| | 3 | | АНГ.МОВА | УКР.МОВА | УКР.МОВА | УКР.МОВА | УКР.МОВА | УКР.МОВА | СВІТ.ЛІТ. | ХІМІЯ | АЛГЕБРА | ФІЗИКА | ГЕОМЕТР. |
| | 4 | | УКР.МОВА | ПРИРОДА | АНГ.МОВА | Я І Україна | ФІЗ-РА | МАТЕМАТ. | УКР.МОВА | ГЕОМЕТР. | ФІЗИКА | БІОЛОГІЯ | ЕКОНОМ. |
| | 5 | | ФІЗ-РА | ТРУД.НАВЧ | ЗАР.ЛІТ. | ІНФОРМАТ | ІСТОРІЯ | ГЕОГРАФ. | ФІЗИКА | ФІЗ-РА | СВІТ.ЛІТ. | УКР.ЛІТ. | АНГ.МОВА |
| | 6 | | | | | | ОСН.ЗДОР. | МУЗИКА | ФІЗ-РА | ІСТОРІЯ | АНГ.МОВА | ІНФОРМАТ | КРЕСЛЕН. |
| | 7 | | | | | | | | | ЗАХ.ВІЛ | ТРУД.НАВ. | ІНФОРМАТ | ФІЗ-РА |
| | 8 | | | | | | | | | | | | |
| Середа | 1 | | УКР.ЧИТ. | УКР.ЧИТ. | УКР.ЧИТ. | ОСН.ЗДОР. | АНГ.МОВА | РОС.МОВА | АЛГЕБРА | УКР.ЛІТ. | БІОЛОГІЯ | УКР.МОВА | АЛГЕБРА |
| | 2 | | МАТЕМАТ. | МАТЕМАТ. | МАТЕМАТ. | МАТЕМАТ. | РОС.МОВА | АНГ.МОВА | ГЕОМЕТР. | УКР.МОВА | УКР.МОВА | ГЕОМЕТР. | ХІМІЯ |
| | 3 | | УКР.МОВА | УКР.МОВА | УКР.МОВА | УКР.МОВА | МАТЕМАТ. | МАТЕМАТ. | АНГ.МОВА | ФІЗИКА | ХІМІЯ | ІСТОРІЯ | УКР.ЛІТ. |
| | 4 | | ПРИРОДА | ФІЗ-РА | МУЗИКА | АНГ.МОВА | УКР.МОВА | ФІЗ-РА | ІСТОРІЯ | ХІМІЯ | ГЕОМЕТР. | АЛГЕБРА | ФІЗИКА |
| | 5 | | ОБР.МИСТ. | МУЗИКА | ТРУД.НАВ. | ЗАР.ЛІТ. | ІНФОРМАТ | ІСТОРІЯ | УКР.МОВА | АЛГЕБРА | ФІЗ-РА | ГЕОГРАФ. | АНГ.МОВА |
| | 6 | | | | | | СВІТ.ЛІТ. | ТРУД.НАВ. | МУЗИКА | ФІЗ-РА | МАТЕМ.(Ф) | АНГ.МОВА | ЛЮДИНА |
| | 7 | | | | | | | | | | | ХУД.КУЛЬТ | КРЕСЛЕН. |

Рисунок 1.2 – Приклад організації даних у табличному процесорі

Окремо варто виділити проблему комунікаційної затримки (latency). Навчальний процес є динамічним середовищем: хвороби викладачів, технічні несправності в аудиторіях, проведення незапланованих заходів вимагають оперативних змін у розкладі. У традиційній схемі адміністрування ланцюжок передачі інформації від моменту прийняття рішення до моменту оповіщення студента є надто довгим. Диспетчер вносить правки у локальний файл, генерує нову версію для друку або публікації на сайті, передає її відповідальним особам, і лише потім інформація стає публічною. Цей процес може займати від кількох годин до доби. Як наслідок, студенти часто дізнаються про скасування першої

пари, вже приїхавши до навчального закладу, що призводить до непродуктивної витрати часу та зниження мотивації. [6]

Відсутність автоматизованих інструментів аналітики також є суттєвим недоліком існуючих підходів. Адміністрація закладу, користуючись паперовими або Excel-версіями, не має змоги оперативно оцінити завантаженість аудиторного фонду, підрахувати фактичне навантаження на викладачів або спрогнозувати потреби у ресурсах на наступний семестр. Будь-який аналітичний звіт вимагає ручного підрахунку даних, що є трудомістким процесом із високою ймовірністю помилок. [7]

Підсумовуючи вищевикладене, можна стверджувати, що традиційні методи адміністрування розкладу (паперовий та файловий) вичерпали свій ресурс ефективності. Вони не здатні впоратися зі зростаючою складністю освітніх процесів, не забезпечують необхідної гнучкості та швидкості реакції на зміни. Вирішення описаних проблем лежить у площині переходу від статичних документів до динамічних веб-орієнтованих систем, які базуються на реляційних базах даних та забезпечують миттєву синхронізацію інформації для всіх учасників освітнього процесу.

## 1.2. Огляд існуючих програмних рішень та аналогів

Сучасний ринок програмного забезпечення для освітньої сфери (EdTech) пропонує широкий спектр інструментів, що можуть бути адаптовані або безпосередньо використані для адміністрування та візуалізації навчального розкладу. Вибір конкретного рішення для навчального закладу є складним управлінським завданням, яке вимагає врахування багатьох факторів: від бюджетних обмежень та технічної кваліфікації персоналу до специфічних вимог щодо захисту персональних даних. Для обґрунтування доцільності розробки власного вебзастосунку необхідно провести детальний аналіз існуючих класів систем, виокремивши їхні архітектурні переваги, функціональні обмеження та економічну ефективність. Умовно всі існуючі рішення можна поділити на

чотири групи: універсальні офісні інструменти, хмарні календарні сервіси, модулі у складі комплексних LMS-систем та спеціалізоване комерційне програмне забезпечення. [8]

### 1.2.1. Універсальні табличні процесори (Microsoft Excel, Google Sheets)

Історично склалося так, що першим кроком у цифровізації розкладу для більшості українських закладів освіти став перехід від паперових носіїв до електронних таблиць. Програмні продукти, такі як Microsoft Excel та Google Sheets, фактично стали галузевим стандартом де-факто для зберігання структурованих даних. Основною причиною такої популярності є «нульовий поріг входження»: інтерфейс цих програм знайомий кожному співробітнику деканату чи навчальної частини, а ліцензії на офісні пакети зазвичай вже закуплені закладом.

Однак, з точки зору проєктування інформаційних систем, використання електронних таблиць для управління реляційними даними (якими є розклад) вважається архітектурною помилкою, відомою як «антипатерн золотого молотка». Електронна таблиця є двовимірною структурою, тоді як розклад занять має багатовимірну природу (час, група, викладач, аудиторія, тип заняття, парність тижня). Спроба спроектувати цю багатовимірність у плоску таблицю призводить до створення громіздких файлів із складною системою вкладок та перехресних посилань, які надзвичайно важко підтримувати. [9]

Критичним недоліком є відсутність механізмів забезпечення цілісності даних (Referential Integrity). У базі даних неможливо видалити запис про викладача, якщо у нього є призначені пари, тоді як в Excel це робиться одним натисканням клавіші Delete, що руйнує структуру розкладу. Крім того, Google Sheets, попри можливості спільного доступу, погано справляється з розмежуванням прав (Role-Based Access Control). Неможливо надати старості групи право редагувати лише розклад своєї групи, не відкриваючи доступу до

всього документа. Це створює серйозні ризики несанкціонованої зміни даних.

Візуальна складова табличних процесорів також є слабким місцем (Рис. 1.3). Статична сітка комірок погано адаптується під екрани мобільних пристроїв. Студенту доводиться масштабувати зображення, скролити сторінку в пошуках потрібного дня тижня, що суперечить сучасним стандартам UX (User Experience).

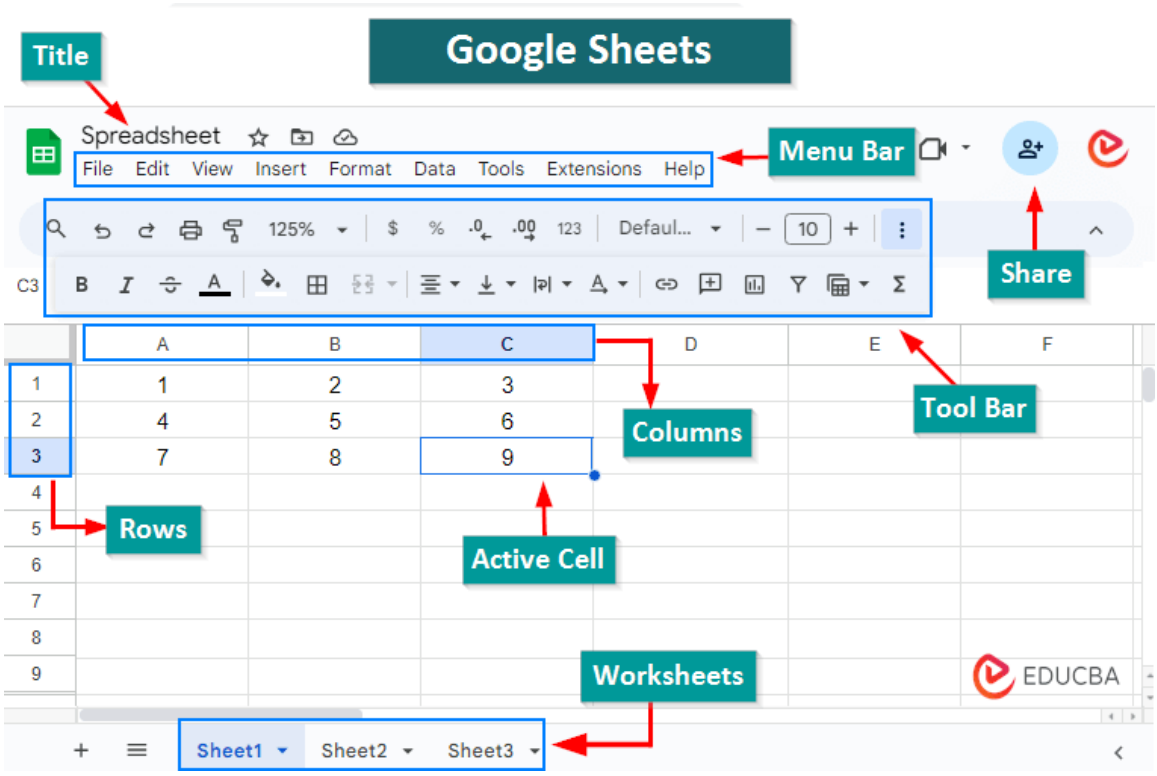



Рисунок 1.3 – Інтерфейс Google Sheets

### 1.2.2. Хмарні календарні сервіси (Google Calendar, Outlook Calendar)

Наступним рівнем еволюції є використання спеціалізованих календарних сервісів. Google Calendar (Рис. 1.4) є потужним інструментом для персонального тайм-менеджменту, який пропонує зручний інтерфейс, синхронізацію між пристроями та систему нагадувань. Деякі університети намагаються імплементувати розклад шляхом створення окремих публічних календарів для кожної академічної групи та надання студентам посилань для підписки (iCal).

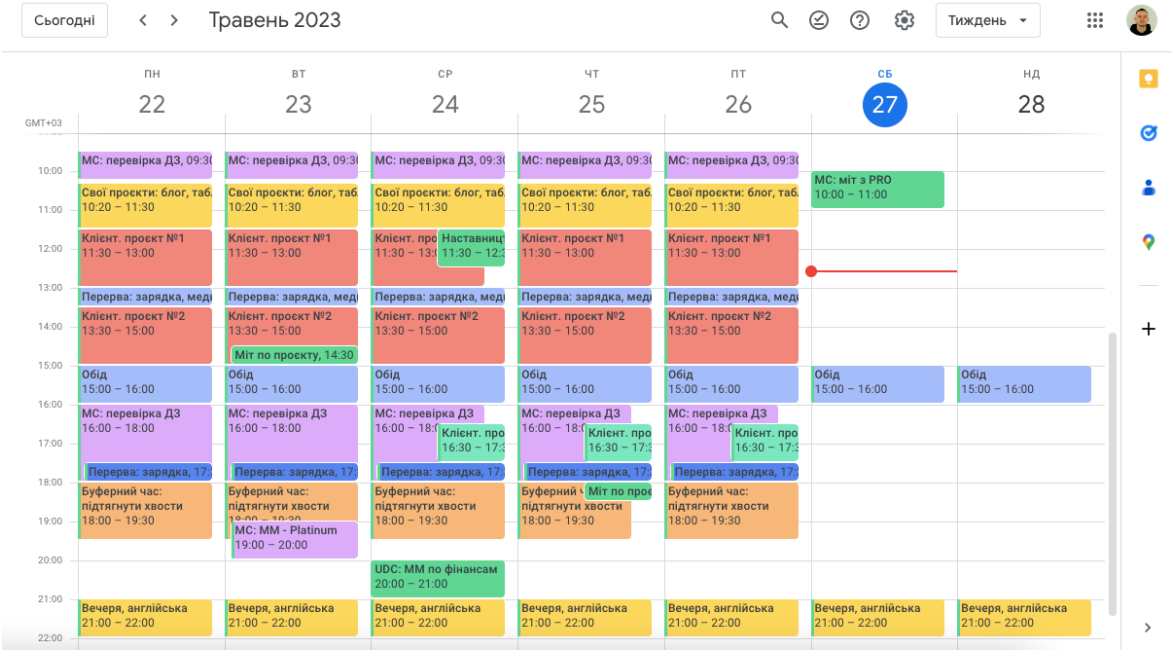



Рисунок 1.4 – Приклад розкладу Google Calendar

Головною проблемою такого підходу є концептуальна невідповідність моделі даних. Google Calendar оперує поняттям «подія» (Event) , яка має час, назву та опис. Для навчального закладу «подія» є значно складнішою сутністю, яка включає зв’язки з базою викладачів та аудиторій. У стандартному календарі неможливо реалізувати фільтрацію за параметрами, наприклад: «показати всі лекції професора Іваненка в аудиторії 101». Крім того, адміністрування такої системи в масштабах університету перетворюється на рутину: якщо в закладі 100 груп, адміністратору потрібно створити та підтримувати 100 окремих календарів. При зміні розкладу (наприклад, перенесення робочих днів) необхідно масово редагувати тисячі подій, що без використання складних скриптів (Google Apps Script) є неможливим. [10]

Інтерфейс Google Calendar, хоч і є зручним, орієнтований на тижневе або місячне планування однієї особи. При накладанні розкладу кількох підгруп або потоків візуальна сітка стає нечитабельною через нагромадження блоків.

### 1.2.3. Модулі у складі систем управління навчанням (LMS Moodle)

Системи управління навчанням (Learning Management Systems), зокрема Moodle, є стандартом для організації дистанційної освіти в Україні. Moodle — це модульна об’єктно-орієнтована динамічна система, яка дозволяє розширювати функціонал за допомогою плагінів. Існують розширення (наприклад, "Scheduler" або "Attendance"), які дозволяють вносити розклад занять безпосередньо в курси.

Перевагою використання Moodle є інтеграція в єдине освітнє середовище: студент авторизується в системі і бачить як навчальні матеріали, так і графік занять. Це забезпечує високий рівень безпеки даних, оскільки доступ до розкладу мають лише авторизовані користувачі. Проте Moodle (Рис. 1.5) характеризується як «важка» система (resource-heavy system). Архітектура платформи спроектована таким чином, що завантаження будь-якої сторінки ініціює сотні запитів до бази даних та виконання складних PHP-скриптів для перевірки прав доступу, ролей та налаштувань курсу. [11]

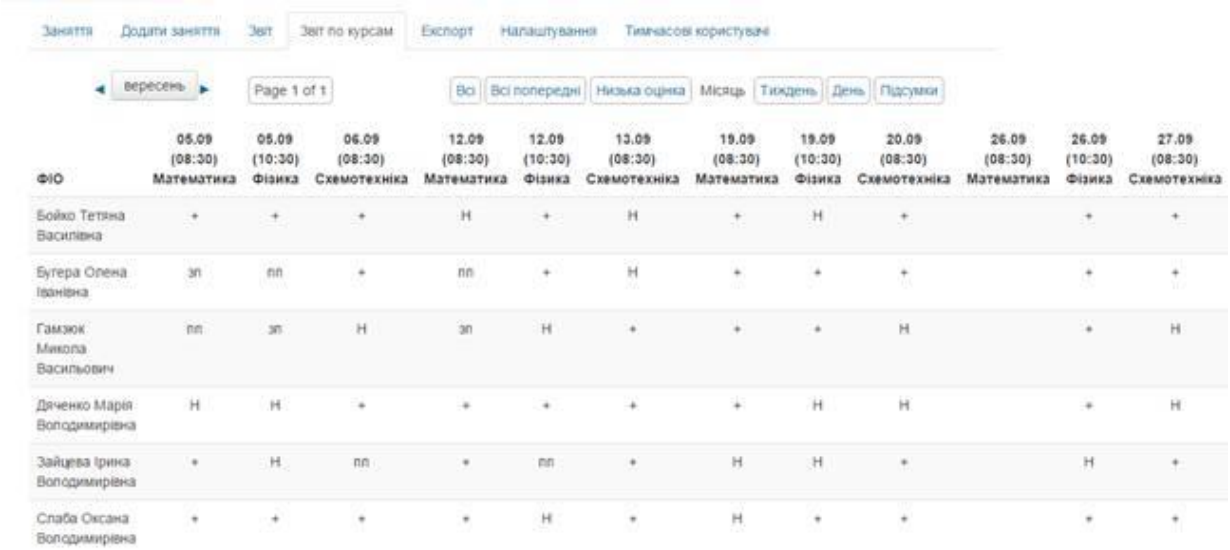



Рисунок 1.5 – Модуль розкладу в системі Moodle

У пікові моменти навантаження (наприклад, ранок понеділка, коли тисячі студентів одночасно намагаються перевірити першу пару) сервери Moodle часто

не витримують навантаження, що призводить до відмови в обслуговуванні (Error 503). Для задачі швидкого перегляду розкладу («дістати телефон — глянути аудиторію — сховати телефон») використання важкого інтерфейсу LMS є надлишковим. Крім того, UX-дизайн Moodle часто критикують за складність навігації: щоб дістатися до розкладу, користувачеві потрібно зробити 3-5 кліків, що знижує ефективність використання системи «на ходу».

### 1.2.4. Спеціалізовані комерційні SaaS-рішення (Rozklad.org, Dekanat)

На ринку існують спеціалізовані програмні комплекси («АСУ ВНЗ», «Деканат», сервіс Rozklad.org), розроблені професійними IT-компаніями спеціально для потреб української освіти. Ці системи пропонують максимальну функціональність: автоматичну генерацію розкладу з урахуванням навантаження викладачів, контроль аудиторного фонду, генерацію звітності для міністерства тощо. Вони часто мають мобільні додатки та сучасні веб-інтерфейси.

Однак головним бар'єром для їх впровадження є фінансовий та організаційний аспекти. По-перше, вартість ліцензії та технічної підтримки таких систем може сягати сотень тисяч гривень на рік, що є критичним для бюджетних установ. По-друге, це пропрієтарні (закриті) системи. Навчальний заклад потрапляє у залежність від розробника (vendor lock-in): якщо адміністрації потрібно додати специфічну функцію (наприклад, особливий тип тижня або інтеграцію з турнікетами на вході), вони змушені чекати, поки розробник погодиться це реалізувати за додаткову плату. [12]

Також виникає питання суверенітету даних. Використання хмарних SaaS-рішень (Software as a Service) передбачає, що база даних студентів та викладачів фізично знаходиться на серверах приватної компанії, що може створювати ризики в контексті законодавства про захист персональних даних.

Підсумовуючи огляд аналогів, можна зробити висновок, що на ринку існує

певний вакуум рішень для середнього сегменту. Універсальні інструменти (Excel, Google) є безкоштовними, але функціонально неспроможними забезпечити якісне адміністрування. Комплексні системи (LMS, SaaS) є потужними, але надто дорогими або ресурсомісткими. Це обґрунтовує актуальність теми магістерської роботи: створення легковагого (lightweight), безкоштовного у розгортанні вебзастосунку на базі PHP-фреймворку Laravel. Таке рішення дозволить поєднати швидкість роботи та зручний інтерфейс (як у Google Calendar), надійну структуру даних та захист від помилок (як у комерційних системах) та повний контроль над кодом і даними (на відміну від SaaS-рішень). Використання архітектури MVC дозволить створити масштабовану систему, яку можна розгорнути навіть на безкоштовному shared-хостингу, що робить її ідеальним вибором для українських навчальних закладів в умовах обмеженого фінансування.

### 1.3. Порівняльний аналіз сучасних вебтехнологій для розробки динамічних застосунків

Вибір базової технології для реалізації серверної частини (Back-end) є стратегічним рішенням, яке визначає не лише процес розробки, але й подальшу вартість володіння програмним продуктом (TCO — Total Cost of Ownership). Для освітніх закладів, які часто обмежені у фінансуванні та не мають штату висококваліфікованих системних адміністраторів, критичними критеріями вибору стають не стільки «модність» технології, скільки її стабільність, легкість розгортання на стандартному обладнанні та невимогливість до ресурсів. На сьогоднішній день у сфері веб-розробки домінують три основні екосистеми: Python, Node.js та PHP. Кожна з них має свої архітектурні особливості, які необхідно проаналізувати крізь призму поставленої задачі — створення доступної системи розкладу. [13]

### 1.3.1. Аналіз середовища Node.js

Node.js — це середовище виконання коду JavaScript, побудоване на рушії V8 від Google. Його ключовою особливістю є асинхронна, подієво-орієнтована модель введення-виведення (non-blocking I/O). Це означає, що сервер Node.js не створює окремий потік для кожного клієнта, а обробляє всі запити в одному потоці, використовуючи механізм Event Loop. Така архітектура робить Node.js ідеальним рішенням для застосунків реального часу, де потрібно тримати постійне з’єднання з користувачем (веб-сокети), наприклад, для чатів, онлайн-ігор або стрімінгових платформ.

Однак, у контексті розробки системи розкладу занять, переваги Node.js нівелюються специфікою задачі. Перегляд розкладу — це класична CRUD-операція (створення, читання, оновлення, видалення), яка не вимагає постійного двостороннього з’єднання. Крім того, розгортання Node.js застосунків має суттєвий недолік для бюджетних установ: воно вимагає використання VPS (Virtual Private Server) або спеціалізованих хмарних платформ. Node.js-процес повинен постійно працювати у пам'яті сервера, що унеможливлює його використання на дешевих тарифах віртуального хостингу (Shared Hosting), які є стандартом для університетських сайтів. Необхідність налаштування серверного оточення, менеджерів процесів (PM2) та зворотного проксі-сервера (Nginx) значно підвищує поріг входження для обслуговуючого персоналу навчального закладу. [14]

### 1.3.2. Аналіз мови програмування Python

Python є однією з найпопулярніших мов програмування у світі, особливо в академічному середовищі, завдяки своєму лаконічному синтаксису та домінуванню в сферах штучного інтелекту та аналізу даних. Веб-розробка на Python зазвичай ведеться з використанням WSGI-серверів (Web Server Gateway Interface), таких як Gunicorn.

Головною проблемою Python у контексті даної роботи є його продуктивність у веб-середовищі порівняно з сучасними альтернативами. Попри високу швидкість розробки, інтерпретатор Python (CPython) має певні обмеження швидкодії через глобальне блокування інтерпретатора (GIL). У синтетичних тестах обробки HTTP-запитів Python часто поступається конкурентам, споживаючи при цьому більше оперативної пам'яті на один процес. . Для системи розкладу, яка може зазнавати пікових навантажень під час сесій або початку семестру (так званий «ефект 1 вересня», коли тисячі студентів одночасно заходять на сайт), це може стати вузьким місцем. Крім того, аналогічно до Node.js, хостинг Python-застосунків є складнішим і дорожчим, ніж класичний веб-хостинг, оскільки вимагає доступу до терміналу сервера для керування віртуальними оточеннями (venv). [15]

### 1.3.3. Аналіз мови програмування PHP

Мова PHP (Hypertext Preprocessor) є унікальною тим, що вона була створена спеціально для веб-розробки. Її архітектурна модель, відома як «Shared Nothing», кардинально відрізняється від Node.js та Python. У цій моделі кожен HTTP-запит запускає окремий ізольований процес, який виконується і повністю очищає пам'ять після завершення. Це робить PHP-застосунки надзвичайно стійкими: помилка в обробці запиту одного студента ніяк не вплине на роботу системи для інших, і не призведе до «падіння» всього сервера.

Сучасні версії PHP (7.4 та 8.x), які розглядаються у даній роботі, здійснили квантовий стрибок у продуктивності. Впровадження нового рушія Zend Engine та механізму попереднього завантаження (Preloading) дозволило PHP випереджати Python у тестах веб-продуктивності у 2-3 рази. [16] Однак вирішальним фактором вибору є інфраструктурна доступність. PHP є «рідною» технологією для стеку LAMP (Linux, Apache, MySQL, PHP), який підтримується абсолютно всіма хостинг-провайдерами світу, включаючи безкоштовні платформи. Для розгортання проєкту на PHP не потрібно налаштовувати демони

чи відкривати порти — достатньо просто завантажити файли на сервер.

Проведений аналіз дозволяє зробити висновок, що для реалізації вебзастосунку динамічного розкладу оптимальним вибором є мова програмування PHP версії 7.4 або вище. Цей вибір ґрунтується на трьох факторах:

1. Архітектурна відповідність: Модель обробки запитів PHP ідеально підходить для веб-сторінок з розкладом, забезпечуючи високу стабільність та ізоляцію помилок.

2. Продуктивність: Сучасні версії PHP забезпечують достатню швидкодію для обробки запитів тисяч студентів без необхідності закупівлі дорогого обладнання.

3. Легкість розгортання: Це єдина технологія, яка дозволяє запустити повноцінний динамічний сайт на безкоштовному хостингу без використання командного рядка сервера, що критично важливо для подальшої підтримки системи співробітниками закладу. [17]

Саме тому PHP обрано як базову платформу для подальшої розробки. Вибір конкретних інструментів на базі PHP (фреймворків) та архітектурних патернів буде обґрунтовано у наступному підрозділі.

## 1.4. Обґрунтування вибору технологічного стека (Laravel, MySQL) та архітектурного патерну MVC

Вибір мови програмування PHP, обґрунтований у попередньому підрозділі, є лише першим кроком у проєктуванні інформаційної системи. Створення сучасного вебзастосунку «з нуля» на чистому PHP (native PHP) сьогодні вважається неефективним підходом, оскільки розробник змушений витрачати значний час на реалізацію базових механізмів: маршрутизації, аутентифікації користувачів, валідації даних та захисту від кібератак. Для вирішення цих завдань в індустрії використовуються програмні каркаси — фреймворки. Серед широкого спектру PHP-фреймворків (Symfony, Yii,

CodeIgniter) для реалізації системи адміністрування розкладу було обрано **Laravel**, який базується на архітектурному патерні **MVC** та використовує систему управління базами даних **MySQL**. Такий вибір потребує детального технічного та методологічного обґрунтування. [18]

### 1.4.1. Обґрунтування архітектурного патерну MVC

Ключовою проблемою розробки складних інформаційних систем є зростання ентропії (хаосу) у коді зі збільшенням функціоналу. У системі розкладу, де необхідно поєднувати логіку перевірки аудиторій, інтерфейс календаря та запити до бази даних, змішування HTML-верстки та PHP-коду (так званий «spaghetti code») призводить до неможливості подальшої підтримки продукту. Для вирішення цієї проблеми було застосовано архітектурний патерн Model-View-Controller (Модель-Вигляд-Контролер) (Рис. 1.6).

Сутність патерну MVC полягає у чіткому розділенні обов'язків між трьома компонентами системи:

1. Модель (Model): Відповідає за роботу з даними та бізнес-логіку. У контексті розкладу моделі (сутності «Lesson», «Teacher», «Classroom») містять правила валідації та зв'язки між об'єктами. Модель не «знає», як ці дані будуть відображені користувачеві.
2. Вигляд (View): Відповідає виключно за візуалізацію даних (користувацький інтерфейс). Це HTML-шаблони, які отримують дані від контролера і формують сторінку календаря або таблицю.
3. Контролер (Controller): Виступає посередником. Він приймає запит від користувача (наприклад, «показати розклад на вівторок»), звертається до Моделі, щоб отримати відповідні дані з бази, і передає їх у Вигляд для відображення.

Використання MVC у магістерській роботі дозволяє досягти модульності системи: зміна дизайну календаря (Front-end) не вимагає переписування логіки перевірки конфліктів (Back-end), що суттєво спрощує процес розробки та

тестування. [19]

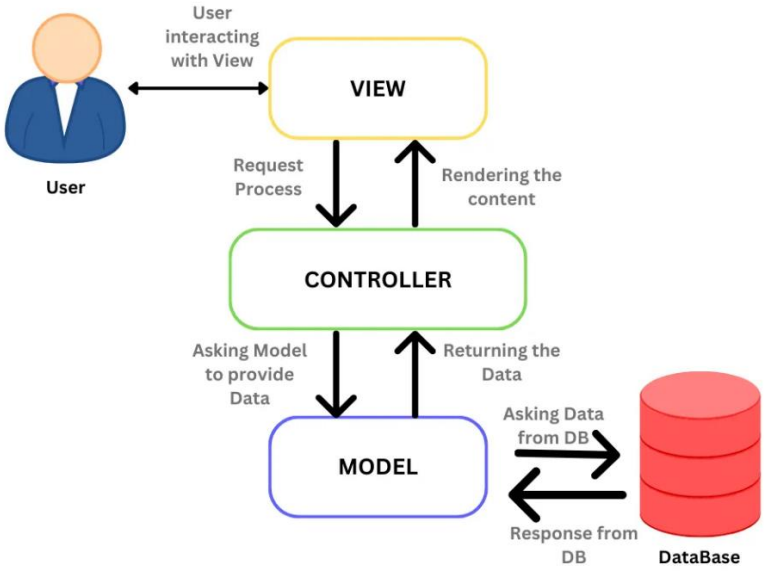



Рисунок 1.6 – Схема взаємодії компонентів в архітектурі MVC

### 1.4.2. Вибір фреймворку Laravel

Laravel — це безкоштовний веб-фреймворк з відкритим кодом, створений Тейлором Отвелом у 2011 році. На сьогодні він є найпопулярнішим PHP-фреймворком у світі завдяки своїй екосистемі та виразному синтаксису. Вибір саме Laravel для реалізації системи розкладу зумовлений кількома технологічними перевагами:

По-перше, це наявність потужної ORM (Object-Relational Mapping) під назвою **Eloquent**. Розклад занять — це система з глибокими реляційними зв’язками: один викладач веде багато уроків; одна група відвідує багато предметів. Написання «сирих» SQL-запитів для вибірки таких даних є складним і вразливим до помилок. Eloquent дозволяє працювати з таблицями бази даних як з об’єктами. Наприклад, щоб отримати всі уроки конкретного викладача, достатньо написати $teacher->lessons, замість складного SQL-запиту з кількома JOIN. Це значно прискорює розробку та робить код читабельним. [20]

По-друге, вбудована система безпеки. Laravel автоматично захищає

вебзастосунок від найпоширеніших вразливостей (OWASP Top 10), таких як SQL-ін'єкції, міжсайтовий скриптинг (XSS) та підробка міжсайтових запитів (CSRF). Для навчального закладу, де в системі зберігаються персональні дані викладачів, цей аспект є критично важливим.

По-третє, наявність шаблонізатора **Blade**, який дозволяє створювати складні динамічні інтерфейси (календарні сітки) з мінімальними витратами ресурсів сервера, що корелює з обраною стратегією використання бюджетного хостингу.

### 1.4.3. Вибір системи управління базами даних (MySQL)

Останнім компонентом технологічного стека є система управління базами даних (СУБД). Для зберігання інформації про розклад було обрано реляційну СУБД **MySQL**. Альтернативою могли б виступати NoSQL рішення (наприклад, MongoDB), які є гнучкішими, але менш придатними для строго структурованих даних.

Навчальний розклад вимагає суворого дотримання цілісності даних (ACID — Atomicity, Consistency, Isolation, Durability). Неприпустимою є ситуація, коли урок існує, але посилання на викладача або аудиторію втрачено. MySQL як реляційна база даних забезпечує механізми зовнішніх ключів (Foreign Keys), які на рівні ядра забороняють видалення даних, що використовуються в розкладі. Це гарантує надійність системи. [21]

Крім того, вибір MySQL обумовлений його повною сумісністю з PHP та доступністю на будь-якому хостингу. У поєднанні з PHP (Server Side) та Laravel (Framework), база даних MySQL формує класичний, перевірений часом стек технологій, який забезпечує найкраще співвідношення продуктивності та вартості впровадження.

Таким чином, технологічний стек **«PHP 7.4 + Laravel + MySQL»**, побудований на принципах **MVC**, є оптимальним фундаментом для створення надійної, безпечної та зручної системи адміністрування навчального розкладу,

здатної працювати в умовах обмежених ресурсів.

## 1.5. Методи забезпечення актуальності даних та вирішення конфліктів у розкладі

Створення програмного забезпечення для управління навчальним процесом вимагає вирішення двох фундаментальних проблем: забезпечення цілісності даних (Data Integrity) при одночасному доступі багатьох користувачів та реалізацію алгоритмів валідації, які унеможливлюють логічні колізії в розкладі. Якщо вибір технологічного стека визначає інструментарій розробки, то методи забезпечення актуальності визначають саму бізнес-логіку системи. У рамках даного дослідження розглядається підхід, що базується на реляційній теорії баз даних та інтервальній часовій логіці.

### 1.5.1. Концепція «Єдиного джерела істини» (Single Source of Truth)

Першим кроком до забезпечення актуальності даних є відмова від децентралізованого зберігання інформації. У розроблюваній системі реалізовано архітектурний принцип SSOT (Single Source of Truth). Це означає, що будь-який елемент розкладу — урок, подія чи бронювання — існує лише в одному екземплярі в центральній базі даних MySQL. Відображення цієї інформації в інтерфейсах адміністратора, викладача чи студента є лише різними проекціями (Views) одних і тих самих даних.

Такий підхід вимагає суворої нормалізації бази даних (зазвичай до третьої нормальної форми 3NF). Наприклад, інформація про викладача (ПІБ, кафедра) зберігається в окремій таблиці teachers, а в таблиці уроків lessons використовується лише посилання на його ідентифікатор (ID). Це гарантує актуальність: якщо адміністратор виправить помилку в прізвищі викладача, ця зміна миттєво і автоматично відобразиться у всіх заняттях цього викладача за

весь семестр, без необхідності редагування кожного уроку окремо. [22]

### 1.5.2. Алгоритмічне вирішення часових колізій

Найскладнішим завданням при автоматизації розкладу є запобігання конфліктам, коли один ресурс (аудиторія або викладач) використовується двічі в один і той же час. Для вирішення цієї задачі використовується алгебра часових інтервалів, зокрема логіка Аллена (Allen's interval algebra).

З математичної точки зору, заняття — це часовий інтервал $T = [Start, End]$. Конфлікт виникає тоді, коли інтервал нового заняття $T\_new$ перетинається з інтервалом будь-якого вже існуючого заняття $T\_existing$ для того ж ресурсу. У розроблюваній системі реалізовано алгоритм перевірки накладання, який спрацьовує ще до моменту запису даних у базу. Умова перетину двох подій $A$ і $B$ визначається булевим виразом: ($Start\_A < End\_B$) І ($End\_A > Start\_B$).

Якщо цей вираз повертає *true* (істину), система ідентифікує колізію:

– Перевірка аудиторії: Чи зайнята аудиторія $X$ у проміжок часу $T$?
– Перевірка викладача: Чи має викладач $Y$ інше заняття у проміжок часу $T$?
– Перевірка групи: Чи не призначено групі $Z$ іншу пару у цей час?

Тільки якщо всі три перевірки повертають негативний результат (перетинів немає), система дозволяє збереження уроку [23].

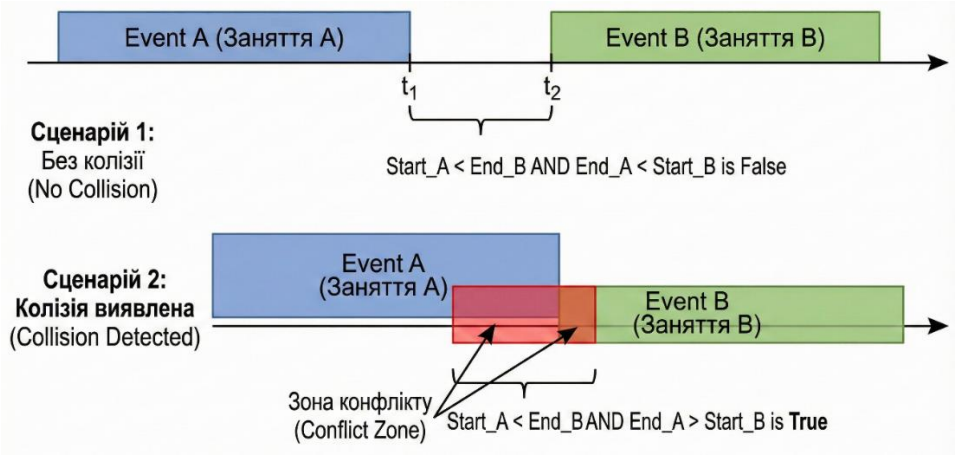



Рисунок 1.10 – Графічне представлення логіки виявлення часових колізій між подіями

### 1.5.3. Вирішення проблеми конкурентного доступу (Race Conditions)

У багатокористувацькому середовищі можлива ситуація «гонки даних» (Race Condition), коли два адміністратори намагаються забронювати одну аудиторію на один і той же час із різницею в мілісекунди. Програмна валідація може не спрацювати, оскільки обидва процеси «побачать», що аудиторія вільна, до того як перший процес встигне записати дані.

Для вирішення цієї проблеми у системі застосовується механізм транзакцій бази даних (Database Transactions) та блокувань. Використання транзакцій (ACID) гарантує, що операція збереження розкладу є атомарною: вона або виконується повністю, або не виконується взагалі. Додатково, на рівні СУБД MySQL можуть застосовуватися унікальні композитні індекси (Unique Composite Indexes), які фізично забороняють створення двох записів з однаковими параметрами часу та ресурсу, генеруючи помилку на рівні бази даних, яку перехоплює додаток і повідомляє користувачеві. [24]

### 1.5.4. Забезпечення динамічності інтерфейсу (AJAX та реактивність)

Актуальність даних для кінцевого користувача забезпечується технологією асинхронного обміну даними AJAX (Asynchronous JavaScript and XML).

У розроблюваній системі клієнтська частина (Front-end) взаємодіє із сервером через REST API. Коли користувач змінює тиждень у календарі, браузер відправляє фоновий запит на сервер, отримує масив подій у форматі JSON і миттєво перемальовує сітку розкладу за допомогою JavaScript-бібліотек, не перезавантажуючи сторінку. Це створює відчуття роботи з настільним додатком (Desktop-like experience) і суттєво знижує навантаження на сервер та споживання трафіку, що є критичним для мобільних користувачів. [25]

# РОЗДІЛ 2
# ПРОЄКТУВАННЯ, РОЗРОБКА ТА РОЗГОРТАННЯ ВЕБЗАСТОСУНКУ ДЛЯ АДМІНІСТРУВАННЯ РОЗКЛАДУ

## 2.1. Постановка задачі, призначення та вимоги до розробки

Розробка сучасного програмного забезпечення для освітньої сфери вимагає чіткого визначення проблематики, цілей та технічних обмежень ще на етапі ініціації проєкту. У рамках даної магістерської роботи вирішується науково-прикладне завдання автоматизації процесу складання та розповсюдження розкладу навчальних занять. Постановка задачі базується на необхідності переходу від застарілих форм адміністрування, таких як паперові носії та статичні електронні документи, до динамічного веб-середовища, що забезпечує актуальність даних у режимі реального часу. Головною метою проєкту є створення вебзастосунку, який виступає єдиною точкою доступу до інформації про навчальний процес для всіх категорій користувачів, забезпечуючи при цьому цілісність даних та унеможливлюючи виникнення організаційних колізій.

### 2.1.1. Призначення та сфера застосування

Розроблюваний вебзастосунок «School Timetable» призначений для впровадження у закладах освіти різного рівня акредитації, які стикаються з проблемою складної логістики навчального процесу. Система проектується як інструмент подвійного призначення. З одного боку, це адміністративна панель для співробітників навчальної частини або диспетчерів, яка дозволяє керувати базою ресурсів закладу: аудиторним фондом, педагогічним складом та навчальними групами. З іншого боку, це інформаційний сервіс для кінцевих споживачів — студентів та викладачів, який надає зручний інтерфейс для перегляду персоналізованого графіку занять.

Впровадження даного програмного продукту дозволяє вирішити низку організаційних проблем, зокрема: усунення комунікаційних затримок при зміні розкладу, зниження навантаження на диспетчерів шляхом автоматизації перевірки конфліктів, а також забезпечення прозорості використання аудиторій. Важливим аспектом призначення системи є її адаптивність до бюджетних обмежень: архітектура застосунку передбачає можливість розгортання на безкоштовних або недорогих хостингових платформах, що робить його доступним для широкого кола навчальних закладів України.

### 2.1.2. Функціональні вимоги до системи

Визначення функціональних вимог здійснювалося на основі аналізу предметної області та потреб потенційних користувачів. Згідно зі стандартом інженерії вимог, систему можна декомпозувати на декілька функціональних блоків, кожен з яких відповідає за специфічний набір операцій.

Першочерговою вимогою є реалізація рольової моделі доступу (Role-Based Access Control — RBAC). Система повинна чітко розмежовувати права користувачів, виділяючи щонайменше дві ролі: «Адміністратор» та «Користувач». Адміністратор повинен мати повний доступ до CRUD-операцій (створення, читання, оновлення, видалення) над усіма сутностями системи, тоді як звичайний користувач має право лише на перегляд інформації (Read-only access). Це забезпечує захист даних від несанкціонованої модифікації. [26]

Критично важливою функціональною вимогою є механізм управління сутностями навчального процесу. Система повинна надавати інтерфейси для ведення реєстрів викладачів, навчальних груп, предметів та аудиторій. При цьому, на відміну від простих текстових редакторів, вебзастосунок зобов'язаний підтримувати реляційну цілісність даних. Наприклад, при видаленні викладача система повинна перевіряти наявність пов'язаних з ним уроків і попереджати адміністратора про можливу втрату даних.

Центральним елементом функціоналу є модуль календаря. Він повинен

забезпечувати візуалізацію розкладу у вигляді часової сітки з можливістю перемикання між режимами перегляду: «Місяць», «Тиждень», «День». Для забезпечення інтерактивності система має підтримувати динамічне завантаження подій без повного перезавантаження сторінки, що досягається використанням технології AJAX. Для зручності сприйняття події в календарі повинні мати кольорове кодування залежно від типу заняття або викладача.

Окрему увагу слід приділити вимозі щодо валідації конфліктів. Система повинна містити алгоритмічний модуль, який під час спроби створення або редагування уроку автоматично перевіряє доступність ресурсів. Зокрема, програмний засіб повинен блокувати операцію збереження, якщо обрана аудиторія вже зайнята іншою групою у вказаний проміжок часу, або якщо викладач вже має призначене заняття. Ця вимога є ключовою для переходу від ручного до автоматизованого управління розкладом.

### 2.1.3. Нефункціональні вимоги та технічні обмеження

Окрім безпосередніх функцій, програмний продукт повинен відповідати низці нефункціональних вимог (якісних атрибутів), визначених стандартом якості програмного забезпечення ISO/IEC 25010. Пріоритетними атрибутами для даної розробки є надійність, продуктивність та сумісність. [27]

Вимога до сумісності (Compatibility) є визначальною для вибору технологічного стека. Враховуючи, що замовником системи може виступати бюджетна установа, вебзастосунок повинен бути спроектований для роботи в умовах обмежених серверних ресурсів. Зокрема, система повинна коректно функціонувати на віртуальних хостингах (Shared Hosting) з підтримкою версії PHP 7.4. Це накладає обмеження на використання деяких сучасних бібліотек, які вимагають новіших версій інтерпретатора, проте гарантує можливість розгортання на таких платформах, як InfinityFree, без додаткових фінансових витрат.

Вимоги до продуктивності (Performance Efficiency) передбачають

оптимізацію швидкості завантаження сторінок. Оскільки значна частина користувачів (студенти) буде звертатися до розкладу з мобільних пристроїв через мобільний інтернет, розмір передаваних даних має бути мінімізованим. Це досягається шляхом використання мініфікованих скриптів та стилів, а також кешування запитів до бази даних. Час відповіді сервера на запит отримання розкладу не повинен перевищувати 1-2 секунди при нормальному навантаженні.

Вимога до зручності використання (Usability) диктує необхідність створення адаптивного інтерфейсу (Responsive Design). Вебзастосунок повинен автоматично підлаштовувати відображення елементів календаря під розмір екрану пристрою користувача — від широкоформатних моніторів у деканаті до екранів смартфонів студентів. Меню навігації, форми введення та сама сітка розкладу повинні залишатися читабельними та функціональними без необхідності горизонтального прокручування екрану.

Окремо слід виділити вимоги до безпеки даних (Security). Система повинна забезпечувати захист облікових записів адміністраторів від несанкціонованого доступу шляхом хешування паролів. Також обов'язковою є реалізація механізмів захисту від поширених веб-атак, таких як міжсайтовий скриптинг (XSS) та підробка міжсайтових запитів (CSRF), що реалізується засобами обраного фреймворку. Загальна схема вимог до системи візуалізована на (Рис. 2.1).

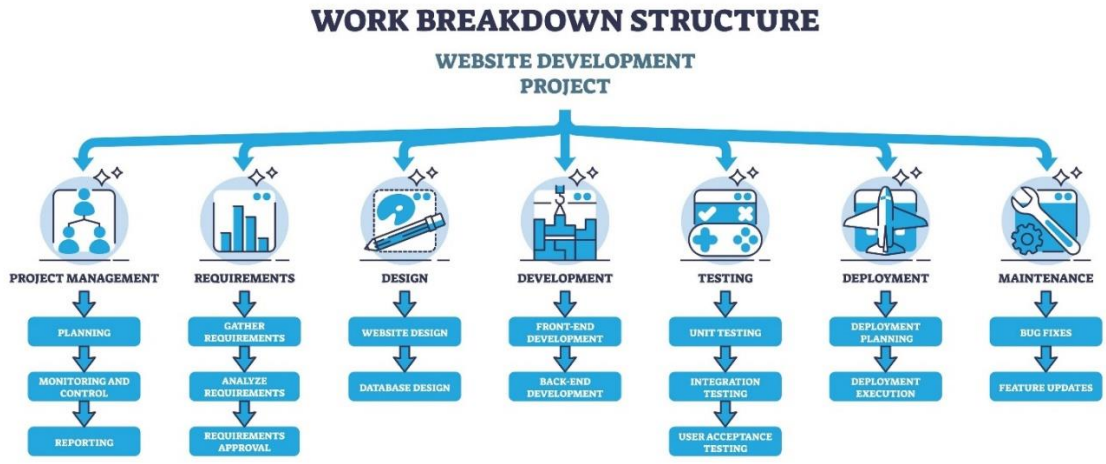



Рисунок 2.1 – Структурна схема вимог до розроблюваного вебзастосунку

Таким чином, сформульована постановка задачі та визначені вимоги окреслюють межі проєктування системи. Розробка ведеться з орієнтацією на створення легкого, надійного та економічно ефективного інструменту, який, на відміну від важких комерційних аналогів, фокусується на вирішенні конкретної проблеми — динамічного управління розкладом з гарантією відсутності колізій. Чітке дотримання цих вимог на етапі реалізації дозволить створити продукт, який повністю відповідає потребам сучасного навчального закладу.

## 2.2. Загальна структура проєкту

Архітектурна організація програмного забезпечення є визначальним фактором його подальшої підтримки та масштабування. Структура розроблюваного вебзастосунку «School Timetable» побудована на основі архітектурного патерну MVC (Model-View-Controller), що є галузевим стандартом для сучасних веб-систем. Цей підхід дозволяє чітко розмежувати компоненти, що відповідають за бізнес-логіку, представлення даних та обробку користувацьких запитів. Фізична структура файлів та каталогів проєкту відповідає стандарту автозавантаження PSR-4 (PHP Standards Recommendations), що забезпечує сумісність компонентів та легкість інтеграції сторонніх бібліотек. [28]

### 2.2.1. Організація файлової системи

Коренева директорія проєкту містить конфігураційні файли та основні системні папки, кожна з яких виконує суворо регламентовану роль у життєвому циклі застосунку. Центральне місце займає директорія app, в якій зосереджена вся бізнес-логіка серверної частини. Всередині цієї папки реалізовано простір імен App\, який містить моделі даних та контролери.

Моделі, розташовані у каталозі app/Models, є об'єктним відображенням таблиць бази даних. Для системи розкладу було створено ключові класи сутностей: Lesson.php (відповідає за зберігання інформації про урок),

Teacher.php (дані про викладачів) та Classroom.php (аудиторний фонд). Така організація дозволяє взаємодіяти з базою даних через ORM Eloquent, не використовуючи прямі SQL-запити в коді програми.

Логіка обробки запитів зосереджена у директорії app/Http/Controllers. Тут розміщуються класи, що приймають вхідні дані від користувача, ініціюють процеси валідації та передають результати у вигляд. Наприклад, Admin\LessonController містить методи для створення, редагування та видалення занять, а також алгоритми перевірки часових колізій. Структуру основних директорій проєкту наведено нижче (Рис. 2.2).

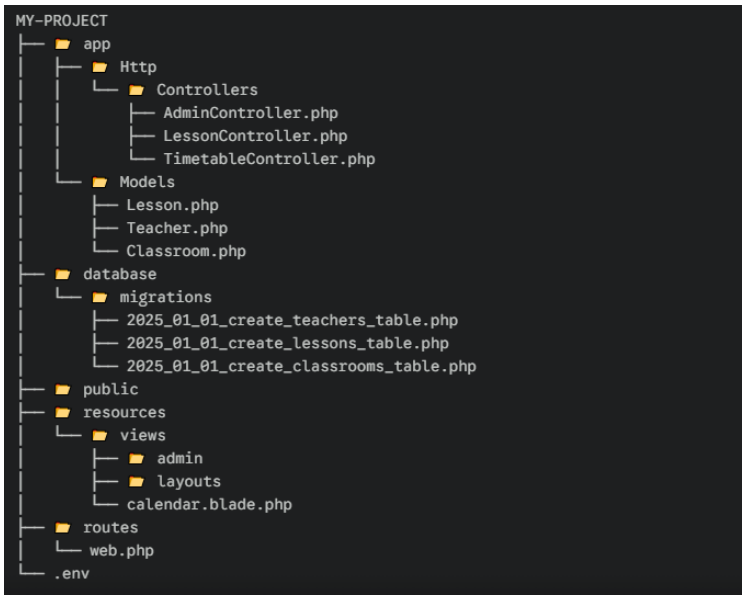



Рисунок 2.2 – Дерево каталогів та файлова структура вебзастосунку

### 2.2.2. Система маршрутизації (Routing)

Важливим елементом структури є система маршрутизації, яка визначає, як URL-адреси співставляються з конкретними контролерами. Всі маршрути веб-інтерфейсу визначені у файлі routes/web.php. Архітектура маршрутів розділена на дві логічні групи: публічні маршрути, доступні всім відвідувачам (наприклад, перегляд головної сторінки календаря), та захищені маршрути адміністративної панелі, які обгорнуті у групу з використанням посередника (middleware) auth. Це гарантує, що доступ до функцій редагування розкладу матимуть лише

авторизовані користувачі.

Програмна реалізація маршрутизації використовує фасад Route, що дозволяє лаконічно описувати endpoints (кінцеві точки) системи. Наприклад, маршрут типу ресурс (Route::resource) автоматично створює всі необхідні шляхи для CRUD-операцій над уроками. Приклад програмного коду, що відповідає за реєстрацію маршрутів, наведено нижче (Рис. 2.3).

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Admin\LessonController;
use App\Http\Controllers\HomeController;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
*/

// Публічний маршрут: головна сторінка з календарем для студентів
Route::get('/', [HomeController::class, 'index'])->name('home');
Route::get('/api/events', [HomeController::class, 'getEvents']);

// Адміністративна панель: доступ тільки для авторизованих користувачів
Route::group(['middleware' => ['auth']], function () {

    // Головна сторінка адмінки
    Route::get('/admin', function () {
        return view('admin.dashboard');
    })->name('dashboard');

    // Маршрути для управління уроками (CRUD)
    Route::resource('lessons', LessonController::class);

    // Маршрути для управління користувачами та аудиторіями
    Route::resource('users', \App\Http\Controllers\Admin\UserController::class);
    Route::resource('classrooms', \App\Http\Controllers\Admin\ClassroomController::class);
});
```

Рисунок 2.3 – Фрагмент програмного коду файлу маршрутизації web.php

Візуальна частина проєкту, що відповідає за відображення даних користувачеві, розміщена у директорії resources/views. Для організації верстки використано шаблонізатор Blade, який дозволяє наслідувати макети сторінок. Базовий шаблон layouts/app.blade.php містить спільні елементи інтерфейсу: шапку сайту (header), навігаційне меню та підвал (footer). Усі інші сторінки, такі як календар або форма додавання уроку, наслідують цей шаблон і перевизначають лише центральну частину контенту (@section('content')). Така структура дозволяє уникнути дублювання HTML-коду та спрощує внесення змін у дизайн: достатньо змінити один файл макету, щоб оновити вигляд усього

сайту.

### 2.2.3. Структура бази даних та міграції

Управління структурою бази даних реалізовано через механізм міграцій, файли яких знаходяться у каталозі database/migrations. Міграції виступають системою контролю версій для схеми БД, дозволяючи команді розробників синхронізувати структуру таблиць. Кожен файл міграції містить методи up() для створення таблиці та down() для її видалення. Це критично важливо для етапу розгортання: при перенесенні проєкту на хостинг структура бази відтворюється автоматично, що мінімізує ризик людської помилки при ручному створенні таблиць. [29]

Окремо слід виділити файл конфігурації оточення .env, який знаходиться в корені проєкту. Хоча він не потрапляє до системи контролю версій Git з міркувань безпеки, його структура є невід'ємною частиною архітектури. Саме тут визначаються параметри підключення до бази даних, налаштування налагодження (APP_DEBUG) та URL-адреса застосунку. Коректна конфігурація цього файлу є обов'язковою умовою функціонування всієї структури проєкту.

Таким чином, розроблена структура проєкту забезпечує чітке розділення відповідальності між компонентами, високий рівень модульності та безпеки. Використання стандартів PSR та вбудованих механізмів фреймворку створює надійний фундамент для подальшого розширення функціоналу системи.

## 2.3. Вибір моделі розробки

Вибір правильної моделі життєвого циклу програмного забезпечення (SDLC — Software Development Life Cycle) є критичним етапом, що визначає стратегію написання коду, тестування та впровадження системи. Враховуючи специфіку магістерської роботи, яка виконується одним розробником в умовах обмеженого часу та чітких вимог до кінцевого результату, для створення

вебзастосунку було обрано ітеративну модель розробки (Iterative Model) з елементами інкрементного підходу.

На відміну від класичної каскадної моделі («Waterfall»), де перехід до наступного етапу можливий лише після повного завершення попереднього, ітеративний підхід дозволяє створювати систему частинами (інкрементами). Це дає можливість отримати працюючий прототип вже на ранніх стадіях розробки, що є важливим для перевірки гіпотез та раннього виявлення архітектурних помилок. Згідно з рекомендаціями Яна Соммервіля, такий підхід є найбільш ефективним для веб-орієнтованих систем, де вимоги до інтерфейсу та хостингу можуть уточнюватися в процесі реалізації. [30]

Процес розробки системи «School Timetable» було розділено на чотири ключові ітерації, кожна з яких завершувалася отриманням функціонального модуля:

Перша ітерація: Проєктування даних. На цьому етапі було розроблено концептуальну схему бази даних та реалізовано її фізичну модель засобами міграцій Laravel. Головним завданням було забезпечення цілісності зв'язків між сутностями «Вчитель», «Урок» та «Аудиторія». Результатом ітерації стала нормалізована структура бази даних MySQL, готова до наповнення тестовими даними. Використання міграцій дозволило версіонувати структуру БД, що спростило подальше розгортання на хостингу.

Друга ітерація: Реалізація бізнес-логіки (Backend). Цей етап був присвячений написанню серверного коду контролерів та моделей. Було реалізовано механізми CRUD (створення, читання, оновлення, видалення) для основних ресурсів системи. Ключовим досягненням цієї ітерації стала програмна реалізація алгоритму валідації часових колізій, який забороняє створення конфліктних записів у розкладі. Тестування проводилося на рівні API, без графічного інтерфейсу, що дозволило ізолювати логічні помилки від візуальних. [31]

Третя ітерація: Інтеграція інтерфейсу (Frontend). На даному етапі було створено візуальну оболонку вебзастосунку. Використання шаблонізатора Blade

дозволило інтегрувати верстку з серверними даними. Основну увагу було приділено підключенню JavaScript-бібліотеки FullCalendar, яка відповідає за динамічне відображення розкладу. В результаті було отримано повноцінний локальний прототип системи, який працював на сервері XAMPP.

Четверта ітерація: Розгортання та адаптація (Deployment). Фінальна ітерація полягала у перенесенні програмного продукту з локального середовища розробки у публічний доступ. На цьому етапі було виконано конфігурацію веб-сервера Apache через файл .htaccess, налаштування змінних оточення .env для роботи з продуктивною базою даних та адаптацію шляхів до статичних файлів. Успішне завершення цієї ітерації ознаменувало перехід системи у стадію експлуатації. [31]

Обрана модель розробки дозволила мінімізувати ризики, пов'язані з несумісністю версій PHP, оскільки перевірка вимог хостингу здійснювалася паралельно з написанням коду. Такий підхід забезпечив створення стабільного програмного продукту, готового до реального впровадження у навчальний процес.

## 2.4. Обґрунтування вибору інструментальних засобів розробки

Реалізація спроєктованої архітектури вимагає використання комплексу програмних засобів, які забезпечують повний цикл розробки: від написання коду до розгортання на сервері. Базуючись на теоретичному аналізі технологій, проведеному в першому розділі роботи, для практичної реалізації системи «School Timetable» було сформовано інструментальний стек, який складається з фреймворку, мови програмування та середовища виконання. У цьому підрозділі деталізуються особливості використання обраних інструментів безпосередньо в процесі написання коду та налаштування системи.

### 2.4.1. Фреймворк Laravel

У якості основного середовища розробки (Framework) використано Laravel (версія 8.x). На відміну від теоретичного обґрунтування його переваг, у практичній площині вибір цього інструменту дозволив задіяти конкретні вбудовані механізми, що суттєво прискорили процес написання коду.

В ході розробки вебзастосунку було активно використано наступні компоненти фреймворку:

- Консоль Artisan: вбудований інтерфейс командного рядка використовувався для автоматичної генерації шаблонного коду (scaffolding). За допомогою команд php artisan make:model та php artisan make:controller було створено каркас класів, що дозволило уникнути рутинного написання синтаксичних конструкцій та зосередитися на бізнес-логіці розкладу.
- Система міграцій (Migrations): для керування структурою бази даних застосовано механізм міграцій, який дозволяє описувати таблиці та зв'язки мовою PHP. Це вирішило проблему перенесення бази даних: замість ручного створення таблиць через SQL-запити на хостингу, структура розгорталася автоматично за допомогою команди migrate.
- Шаблонізатор Blade: для реалізації клієнтської частини використано двигун Blade, який дозволяє розділити HTML-верстку на модулі (компоненти). Це дало змогу винести спільні елементи інтерфейсу (меню, підвал, підключення скриптів FullCalendar) в окремий макет layouts/app, який наслідується всіма сторінками календаря.
- Middleware (Посередники): для захисту адміністративної панелі використано вбудований механізм auth, який перехоплює HTTP-запити та перевіряє наявність активної сесії користувача перед наданням доступу до контролерів редагування розкладу.

Використання Laravel також дозволило інтегрувати зовнішні залежності через менеджер пакетів **Composer**. Зокрема, для роботи з датами та часом було підключено бібліотеку Carbon, а для генерації тестових даних — бібліотеку

Faker. Таким чином, фреймворк виступив не просто як набір правил написання коду, а як повноцінна екосистема, що забезпечила автоматизацію рутинних процесів розробки.

### 2.4.2. Мова програмування PHP

Для написання серверної логіки вебзастосунку було використано скриптову мову програмування PHP версії 7.4. Вибір саме цієї версії був продиктований технічними обмеженнями середовища розгортання: обраний безкоштовний хостинг InfinityFree забезпечує повну стабільну підтримку саме гілки 7.x. Це вимагало дотримання суворої відповідності синтаксису коду, щоб уникнути помилок сумісності при перенесенні проєкту з локального середовища (де могла бути встановлена новіша версія PHP 8.1) на віддалений сервер.

У практичній площині функціонал мови використовувався для вирішення кількох специфічних задач обробки даних розкладу. По-перше, це робота з асоціативними масивами. Структура розкладу є ієрархічною (дні тижня → аудиторії → уроки). Засоби PHP дозволили ефективно сортувати та групувати вибірки з бази даних, формуючи вкладені масиви перед їх відправкою на клієнтську частину. [32]

По-друге, ключову роль відіграв механізм серіалізації даних. Оскільки клієнтська бібліотека FullCalendar, яка візуалізує розклад, працює з форматом JSON, у контролерах застосунку було використано вбудовану функцію json_encode(). Це дозволило трансформувати об'єкти моделей Eloquent у валідний JSON-рядок, який асинхронно передається браузеру через API-інтерфейс.

По-третє, для забезпечення надійності бізнес-логіки було застосовано механізм типізації (Type Hinting). У сигнатурах методів контролерів та моделей явно вказувалися очікувані типи вхідних параметрів (наприклад, int $id, string $date). Це дозволило виявляти помилки передачі некоректних даних ще на етапі виконання коду, запобігаючи запису спотвореної інформації в базу даних. Також

активно використовувалися функції роботи з датою та часом (клас DateTime та бібліотека-обгортка Carbon), що дозволило реалізувати логіку порівняння часових інтервалів для виявлення колізій між уроками (Рис. 2.4).

```php
<?php

namespace App\Http\Controllers;

use App\Models\Lesson;
use Illuminate\Http\Request;
use Illuminate\Http\JsonResponse;

class LessonController extends Controller
{
    /**
     * Отримання списку занять для календаря (AJAX).
     *
     * @param Request $request
     * @return JsonResponse
     */
    public function index(Request $request): JsonResponse
    {
        // Отримуємо події лише у межах видимого періоду календаря
        $lessons = Lesson::with(['teacher', 'classroom', 'group'])
            ->where('start_time', '>=', $request->get('start'))
            ->where('end_time', '<=', $request->get('end'))
            ->get();

        // Повертаємо дані у форматі JSON для FullCalendar
        return response()->json($lessons);
    }
}
```

Рисунок 2.4 – Фрагмент PHP-коду контролера з використанням типізації та поверненням JSON-відповіді

### 2.4.3. Середовище розгортання InfinityFree та СУБД MySQL

Для розміщення вебзастосунку у мережі Інтернет було обрано хмарну платформу InfinityFree, яка надає послуги віртуального хостингу на базі операційної системи Linux. Вибір цього середовища обумовлений його повною відповідністю архітектурним вимогам стеку LAMP (Linux, Apache, MySQL, PHP), який є нативним для розробленого застосунку. Важливою технічною особливістю даної платформи є використання веб-сервера Apache HTTP Server, який підтримує конфігурацію через децентралізовані файли .htaccess. Це дозволило реалізувати специфічні правила маршрутизації (URL Rewrite), необхідні для коректної роботи єдиної точки входу (Front Controller) фреймворку

Laravel, без необхідності доступу до глобальних налаштувань сервера. [33]

У якості системи управління базами даних (СУБД) використано MySQL версії 5.7, яка надається хостинг-провайдером. На відміну від локального середовища розробки, де керування базою може здійснюватися через консольні команди, робота з СУБД на віддаленому сервері відбувалася через веб-інтерфейс phpMyAdmin. Цей інструмент дозволив виконати імпорт структури бази даних та початкових даних (seed data) з підготовленого SQL-дампу. [34]

Критично важливим аспектом використання MySQL у даному проєкті став вибір рушія таблиць InnoDB. На відміну від застарілого MyISAM, рушій InnoDB підтримує транзакції та зовнішні ключі (Foreign Keys). Це дозволило реалізувати на рівні бази даних каскадні операції видалення та оновлення записів. Наприклад, налаштування зв'язків гарантує, що при видаленні групи автоматично видаляються всі пов'язані з нею записи в розкладі, що забезпечує референційну цілісність даних і запобігає появі "сирітських" записів. Також середовище хостингу накладає певні обмеження на кількість одночасних з'єднань з базою даних (max_user_connections), що вимагало оптимізації SQL-запитів на етапі розробки, зокрема використання "жадібного завантаження" (Eager Loading) у Laravel для вирішення проблеми "N+1 запитів" (Рис. 2.5). [35]

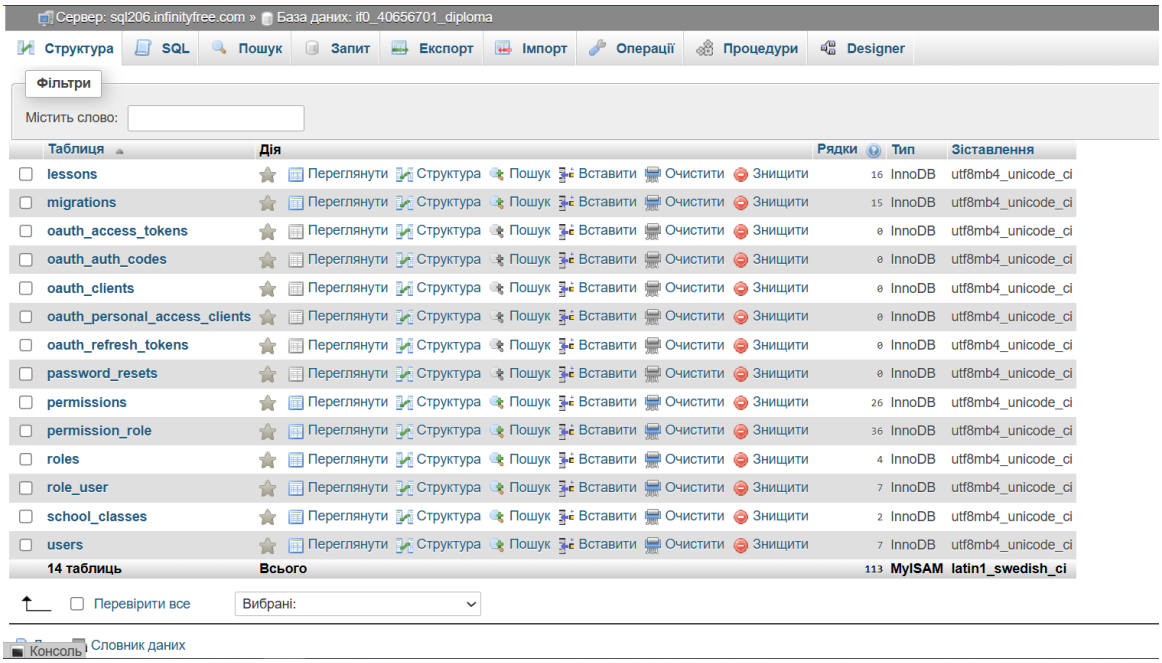



Рисунок 2.5 – Інтерфейс керування базою даних phpMyAdmin на хостингу

## 2.5. Особливості програмної реалізації

Програмна реалізація системи «School Timetable» є результатом трансформації теоретичних моделей та архітектурних рішень, обґрунтованих у попередніх підрозділах, у діючий програмний код. Цей етап розробки включає написання серверної логіки (Backend), яка відповідає за обробку даних та взаємодію з СУБД, створення клієнтських інтерфейсів (Frontend) для відображення інформації кінцевим користувачам, а також реалізацію механізмів асинхронного обміну даними.

Особливістю реалізації є дотримання принципів "чистого коду" (Clean Code) та використання стандартних патернів проєктування, що надаються фреймворком Laravel. Весь програмний код розділено на логічні шари, що забезпечує низьку зв'язність компонентів (Low Coupling) та високу їх згуртованість (High Cohesion). Нижче детально розглянуто програмну реалізацію кожного з компонентів системи.

### 2.5.1. Реалізація серверної архітектури (Backend)

Серверна частина вебзастосунку є ядром системи, яке виконує функції маршрутизації запитів, валідації вхідних даних, взаємодії з базою даних та формування відповідей. Реалізація бекенду розпочалася зі створення моделей даних (Models), які у термінології Laravel Eloquent ORM є об'єктним представленням таблиць бази даних.

Проєктування моделей та зв'язків у директорії app/Models було створено класи Lesson, Teacher, Classroom, кожен з яких наслідує базовий клас Model. Ключовим завданням на цьому етапі було програмне визначення зв'язків між сутностями, що дозволяє отримувати пов'язані дані без написання складних SQL-запитів (JOIN).

Для моделі Lesson (Урок), яка є центральною сутністю системи, реалізовано зв'язки типу «BelongsTo» (належить до), оскільки кожен урок

прив'язаний до конкретного викладача, групи та аудиторії. Програмний код моделі Lesson.php наведено на (Рис. 2.6).

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Lesson extends Model
{
    use HasFactory;

    protected $fillable = [
        'subject',        // Назва предмету
        'start_time',     // Час початку
        'end_time',       // Час завершення
        'teacher_id',     // ID викладача
        'classroom_id',   // ID аудиторії
        'group_id',       // ID групи (класу)
    ];
    public function teacher()
    {
        return $this->belongsTo(Teacher::class);
    }

    public function classroom()
    {
        return $this->belongsTo(Classroom::class);
    }

    public function group()
    {
        return $this->belongsTo(Group::class);
    }
}
```

Рисунок 2.6 – Реалізація моделі Lesson та її зв'язків у файлі Lesson.php

Як видно з лістингу коду, використання властивості $fillable дозволяє захистити модель від вразливості масового призначення (Mass Assignment Vulnerability), чітко вказуючи, які поля дозволено заповнювати через форми (наприклад, weekday, start_time, end_time).

Логіка обробки HTTP-запитів зосереджена у контролерах. Найбільш складним компонентом є адміністративний контролер уроків Admin\LessonController. Він реалізує повний цикл CRUD-операцій. Метод index() відповідає за підготовку даних для відображення списку занять. Для оптимізації кількості запитів до бази даних використано механізм «жадібного завантаження» (Eager Loading) через метод with(). Це дозволяє завантажити дані про викладача та аудиторію для кожного уроку одним запитом, вирішуючи проблему продуктивності "N+1".

Особливу увагу приділено методу store(), який обробляє POST-запит на

створення нового уроку. Перед збереженням даних відбувається їх валідація за допомогою спеціального класу StoreLessonRequest. Це дозволяє винести правила перевірки (наприклад, що час початку має бути меншим за час завершення) за межі контролера, роблячи код чистішим. Фрагмент реалізації методу store наведено на (Рис. 2.7).

```php
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Http\Requests\StoreLessonRequest;
use App\Models\Lesson;

class LessonController extends Controller
{
    /**
     * Збереження нового заняття у базі даних.
     *
     * @param  StoreLessonRequest  $request
     * @return \Illuminate\Http\RedirectResponse
     */
    public function store(StoreLessonRequest $request)
    {
        // 1. Отримуємо провалідовані дані (якщо конфліктів немає)
        $validated = $request->validated();

        // 2. Створюємо новий запис уроку в БД через Mass Assignment
        Lesson::create($validated);

        // 3. Повертаємо адміністратора назад із повідомленням про успіх
        return redirect()->route('admin.lessons.index')
            ->with('success', 'Заняття успішно додано до розкладу.');
    }
}
```

Рисунок 2.7 – Програмна реалізація методу збереження уроку в контролері

Маршрутизація запитів реалізована у файлі routes/web.php. Для організації чистої структури URL-адрес використано групування маршрутів. Всі маршрути адміністративної панелі об'єднані префіксом admin та захищені посередником (Middleware) auth. Це означає, що будь-яка спроба доступу до адреси /admin/lessons неавторизованим користувачем буде автоматично перехоплена системою, яка перенаправить користувача на сторінку входу.

Для ресурсних контролерів використано метод Route::resource, який автоматично генерує сім стандартних маршрутів (index, create, store, show, edit, update, destroy), що суттєво зменшує обсяг коду у файлі маршрутизації (Рис. 2.8).

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Admin\DashboardController;
use App\Http\Controllers\Admin\LessonController;

Route::group(['middleware' => ['auth'], 'prefix' => 'admin', 'as' => 'admin.'], function () {

    // Головна сторінка панелі керування (Dashboard)
    Route::get('/', [DashboardController::class, 'index'])->name('dashboard');

    // Ресурсні маршрути для управління сутностями (CRUD операції)
    Route::resource('lessons', LessonController::class);
    Route::resource('teachers', \App\Http\Controllers\Admin\TeacherController::class);
    Route::resource('groups', \App\Http\Controllers\Admin\GroupController::class);
    Route::resource('classrooms', \App\Http\Controllers\Admin\ClassroomController::class);

});
```

Рисунок 2.8 – Організація захищених маршрутів адміністративної панелі

Окремо реалізовано логіку Service Providers. У файлі AppServiceProvider.php було налаштовано глобальні правила валідації та локалізацію інтерфейсу (використання бібліотеки Carbon для відображення дат українською мовою). Це дозволяє системі автоматично форматувати дати у звичний для користувача формат (наприклад, "Понеділок, 12 травня") у всіх частинах застосунку.

### 2.5.2. Реалізація клієнтського інтерфейсу (Frontend)

Візуальна складова вебзастосунку, або клієнтський інтерфейс (Frontend), відіграє ключову роль у забезпеченні взаємодії між користувачем та програмною логікою системи. Для реалізації цього рівня у проєкті використано вбудований у Laravel шаблонізатор Blade. На відміну від звичайних PHP-шаблонів, Blade надає розробнику потужний набір інструментів для наслідування макетів, роботи з компонентами та відображення даних, не створюючи при цьому додаткового навантаження на сервер, оскільки всі шаблони компілюються у чистий PHP-код і кешуються до моменту внесення змін.

Усі файли, що відповідають за візуалізацію, розміщено у директорії resources/views. Для підтримання чистоти архітектури та зручності навігації було розроблено ієрархічну структуру папок, яка логічно відображає функціональні

модулі системи:

- каталог layouts містить базові макети сторінок, які визначають загальну структуру HTML-документа (шапку, навігаційне меню, підвал);
- каталог admin включає підпапки для кожного ресурсу (lessons, teachers, classrooms), у яких розміщено шаблони для CRUD-операцій (index.blade.php, create.blade.php, edit.blade.php);
- каталог partials використовується для зберігання дрібних повторюваних елементів інтерфейсу, таких як повідомлення про помилки валідації або модальні вікна;
- каталог auth містить автоматично згенеровані шаблони для сторінок входу та реєстрації. [36]

Така організація дозволяє уникнути дублювання коду та спрощує підтримку проєкту: зміна одного файлу в папці partials автоматично відображається на всіх сторінках, де цей елемент підключено.

Центральним елементом frontend-архітектури є головний шаблон адміністративної панелі, розташований у файлі resources/views/layouts/admin.blade.php. Цей файл визначає "скелет" сторінки, підключає зовнішні CSS-стилі (зокрема Bootstrap) та JavaScript-бібліотеки (jQuery, FullCalendar). [37]

Для створення гнучкої структури сторінок використано директиви Blade, які реалізують концепцію наслідування шаблонів:

- директива @yield('content') визначає область, у яку дочірні сторінки будуть вставляти свій унікальний контент. Це дозволяє динамічно змінювати центральну частину сторінки, залишаючи меню та шапку незмінними;
- директива @include('partials.menu') використовується для підключення окремих фрагментів коду, наприклад, бокового меню навігації. Це дозволяє розвантажити головний файл шаблону та зробити код більш читабельним;
- директива @section у дочірніх файлах визначає контент, який буде передано у батьківський шаблон.

На (Рис. 2.9) наведено фрагмент коду головного шаблону, де продемонстровано підключення мета-тегів, CSRF-токена для безпеки AJAX-запитів та визначення основних секцій контенту.

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="csrf-token" content="{{ csrf_token() }}">

    <title>Адмін-панель | Розклад занять</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet">
    <link href="{{ asset('css/admin.css') }}" rel="stylesheet">
</head>
<body>

    <div class="container-fluid">
        <div class="row">

            <nav id="sidebar" class="col-md-3 col-lg-2 d-md-block bg-light sidebar collapse">
                @include('partials.menu')
            </nav>

            <main class="col-md-9 ms-sm-auto col-lg-10 px-md-4">
                <div class="d-flex justify-content-between flex-wrap flex-md-nowrap align-items-center pt-3 pb-2 mb-3 border-bottom">
                    <h1 class="h2">Панель керування</h1>
                </div>

                <div class="content-wrapper">
                    @yield('content')
                </div>
            </main>
        </div>
    </div>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

Рисунок 2.9 – Структура головного шаблону admin.blade.php з використанням директив Blade

Одним із найбільш складних елементів інтерфейсу є форма додавання нового заняття, реалізована у файлі resources/views/admin/lessons/create.blade.php. Ця сторінка наслідує головний макет через директиву @extends('layouts.admin') і наповнює секцію контенту формою введення даних.

Особливістю реалізації форм у Laravel є обов'язкове використання директиви @csrf. Вона генерує приховане поле з унікальним токеном сесії, який перевіряється сервером при отриманні POST-запиту. Це забезпечує захист від атак типу Cross-Site Request Forgery (підробка міжсайтових запитів). Без цього токена сервер відхилить запит з кодом помилки 419.

Також у формі активно використовується директива @error, яка дозволяє

виводити повідомлення про помилки валідації безпосередньо під відповідним полем введення. Якщо користувач спробує зберегти урок у вже зайнятій аудиторії, сервер поверне помилку, і Blade автоматично відобразить її, додавши до поля введення клас стилізації is-invalid.

На (Рис. 2.10) зображено візуальний вигляд форми додавання уроку, а на (Рис. 2.11) – відповідний фрагмент програмного коду шаблону.

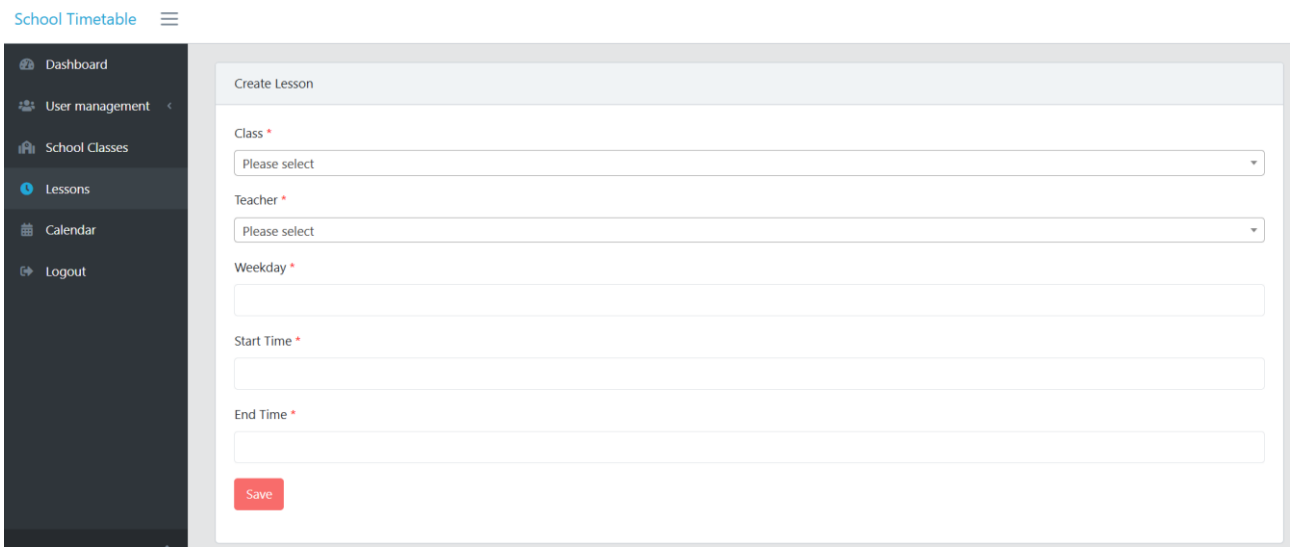



Рисунок 2.10 – Інтерфейс форми додавання нового заняття

```
@extends('layouts.admin')

@section('content')
<div class="card">
    <div class="card-header">
        Create Lesson
    </div>

    <div class="card-body">
        <form action="{{ route('admin.lessons.store') }}" method="POST">
            @csrf <div class="form-group mb-3">
                <label for="class_id" class="required">Class *</label>
                <select name="class_id" id="class_id" class="form-control" required>
                    <option value="">Please select</option>
                    @foreach($classes as $id => $name)
                        <option value="{{ $id }}" {{ old('class_id') == $id ? 'selected' : '' }}>
                            {{ $name }}
                        </option>
                    @endforeach
                </select>
                @if($errors->has('class_id'))
                    <div class="invalid-feedback">
                        {{ $errors->first('class_id') }}
                    </div>
                @endif
            </div>

            <div class="form-group mb-3">
                <label for="teacher_id" class="required">Teacher *</label>
                <select name="teacher_id" id="teacher_id" class="form-control" required>
                    <option value="">Please select</option>
                    @foreach($teachers as $id => $name)
                        <option value="{{ $id }}" {{ old('teacher_id') == $id ? 'selected' : '' }}>
                            {{ $name }}
                        </option>
                    @endforeach
                </select>
                @if($errors->has('teacher_id'))
                    <div class="invalid-feedback">
                        {{ $errors->first('teacher_id') }}
                    </div>
                @endif
            </div>

            <div class="form-group mb-3">
                <label for="weekday" class="required">Weekday *</label>
                <input type="number" name="weekday" id="weekday" class="form-control" placeholder="e.g. 1 for Monday" min="1" max="7" value="{{ old('weekday') }}" required>
                @if($errors->has('weekday'))
                    <div class="invalid-feedback">
                        {{ $errors->first('weekday') }}
                    </div>
                @endif
            </div>
```

Рисунок 2.11 – Фрагмент програмної реалізації форми засобами Blade

Для забезпечення коректного відображення вебзастосунку на різних типах пристроїв (від настільних моніторів до смартфонів) використано CSS-фреймворк Bootstrap 4. Його інтеграція дозволила відмовитися від написання власних медіа-запитів (Media Queries) та використати готову систему сітки (Grid System).

У коді шаблонів активно застосовуються класи адаптивності. Наприклад, для розміщення полів форми у два стовпчики на великих екранах і в один стовпчик на мобільних телефонах використано конструкцію: . Клас col-md-6 вказує браузеру, що на екранах середнього розміру (Medium devices) елемент повинен займати половину ширини контейнера (6 із 12 колонок сітки).

Для стилізації елементів введення використано стандартний клас .form-control, який уніфікує вигляд текстових полів, випадаючих списків та полів вибору дати, забезпечуючи їм стани фокусу та валідації. Кнопки дій стилізовані класами .btn btn-primary (основна дія) та .btn btn-danger (видалення), що створює інтуїтивно зрозумілий інтерфейс для користувача.

Використання готової бібліотеки компонентів дозволило суттєво скоротити час розробки фронтенду та гарантувати кросбраузерну сумісність верстки, що є важливою вимогою до сучасних веб-систем.

### 2.5.3. Інтеграція інтерактивного календаря (FullCalendar.js)

Ключовим функціональним елементом клієнтської частини системи є модуль календаря, який відповідає за візуалізацію розкладу занять у зручному для сприйняття форматі. Оскільки стандартні засоби HTML не надають можливості відображення складних часових сіток, для реалізації цього функціоналу було обрано спеціалізовану JavaScript-бібліотеку FullCalendar. Це потужне рішення з відкритим вихідним кодом, яке дозволяє маніпулювати подіями в DOM-дереві сторінки, забезпечуючи високу швидкість рендерингу та інтерактивність без перезавантаження веб-сторінки.

Інтеграція FullCalendar у середовище Laravel виконувалася через механізм

стеків (Stacks) шаблонізатора Blade. Оскільки скрипти календаря потрібні не на всіх сторінках адміністративної панелі, а лише у розділі «Розклад», їх підключення було реалізовано локально, щоб не перевантажувати інші сторінки зайвим кодом.

У головному шаблоні layouts/admin.blade.php було визначено спеціальні секції @stack('styles') та @stack('scripts'). У дочірньому шаблоні календаря (resources/views/admin/calendar.blade.php) ці секції наповнюються посиланнями на CDN-ресурси бібліотеки. Було підключено три основні модулі:

1. main.css / main.js – ядро бібліотеки, що відповідає за базову логіку відображення.
2. daygrid – плагін для відображення сітки місяця.
3. timegrid – плагін для погодинного відображення дня та тижня.
4. interaction – модуль, що дозволяє користувачеві взаємодіяти з календарем (кліки по подіях, перетягування).

Програмна логіка ініціалізації календаря реалізована мовою JavaScript і розміщена в нижній частині шаблону сторінки. Скрипт спрацьовує після повного завантаження DOM-дерева (DOMContentLoaded), що гарантує наявність HTML-контейнера для рендерингу.

Створення екземпляра календаря відбувається шляхом виклику конструктора FullCalendar.Calendar, який приймає два аргументи: HTML-елемент (контейнер з id="calendar") та об'єкт конфігурації. Об'єкт конфігурації є ключовим елементом налаштування системи, оскільки саме він визначає поведінку інтерфейсу. [38]

Важливим етапом налаштування стало визначення панелі інструментів (headerToolbar). Вона була сконфігурована таким чином, щоб забезпечити швидку навігацію:

- ліва частина (prev,next today) містить кнопки перемикання часових періодів;
- центральна частина (title) відображає поточний місяць або тиждень;
- права частина (dayGridMonth,timeGridWeek,timeGridDay) дозволяє

користувачеві змінювати масштаб відображення.

Особливу увагу було приділено форматуванню часу. За замовчуванням бібліотека використовує 12-годинний формат (AM/PM), який не є характерним для української освітньої системи. Тому в налаштуваннях було явно вказано параметри slotLabelFormat та eventTimeFormat для використання 24-годинного формату (HH:mm). Це забезпечило коректне відображення часу початку та завершення пар (наприклад, "08:30 - 09:50").

На (Рис. 2.12) наведено фрагмент JavaScript-коду, що демонструє ініціалізацію календаря з базовими налаштуваннями.

```
document.addEventListener('DOMContentLoaded', function() {
    // Отримуємо DOM-елемент, де буде розміщено календар
    var calendarEl = document.getElementById('calendar');

    // Ініціалізація об'єкта FullCalendar
    var calendar = new FullCalendar.Calendar(calendarEl, {
        initialView: 'timeGridWeek', // Початковий вигляд: розклад на тиждень

        // Налаштування панелі інструментів (headerToolbar)
        headerToolbar: {
            left: 'prev,next today',
            center: 'title',
            right: 'dayGridMonth,timeGridWeek,timeGridDay'
        },

        locale: 'en',
        firstDay: 1,            // Початок тижня: понеділок
        slotMinTime: '08:00:00', // Початок відображення часу (08:00)
        slotMaxTime: '18:00:00', // Кінець відображення часу (18:00)
        allDaySlot: false,       // Приховати слот "весь день"

        // Джерело даних: завантаження подій через API маршрут
        events: '/api/events',

        // Обробка помилок завантаження
        failure: function() {
            alert('Помилка завантаження розкладу!');
        }
    });

    // Рендеринг календаря на сторінці
    calendar.render();
});
```

Рисунок 2.12 – Програмний код ініціалізації об'єкта FullCalendar

Найважливішим параметром конфігурації є властивість events. Вона відповідає за те, звідки календар отримує дані про уроки. Замість того, щоб жорстко прописувати масив подій у HTML-коді (що зробило б сторінку «важкою» та повільною), було використано динамічне завантаження даних. Властивість events вказує на API-маршрут сервера Laravel (наприклад, /api/calendar).

Коли користувач відкриває сторінку, FullCalendar автоматично відправляє GET-запит на цей маршрут, додаючи параметри start та end (початок і кінець

видимого періоду). Це дозволяє завантажувати з бази даних лише ті уроки, які користувач бачить на екрані в даний момент, суттєво знижуючи навантаження на сервер та споживання трафіку (Рис. 2.13).

Для забезпечення інтерактивності було реалізовано обробники подій (Event Handlers). Зокрема, параметр eventClick дозволяє визначити функцію, яка виконується при кліку на конкретне заняття в сітці розкладу. У розробленій системі цей механізм використано для редагування занять. При кліку на блок уроку скрипт зчитує його унікальний ідентифікатор (info.event.id) та перенаправляє адміністратора на сторінку редагування (/admin/lessons/{id}/edit). [39]

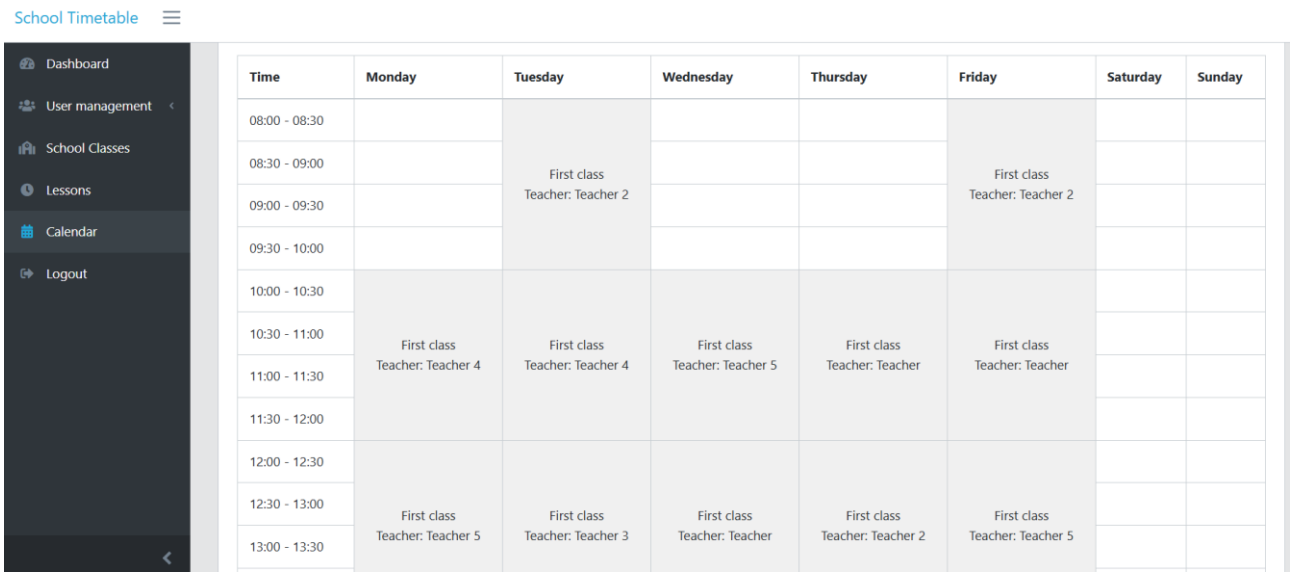



Рисунок 2.13 – Візуальне відображення розкладу занять

Також було налаштовано параметр nowIndicator: true, який відображає червону лінію поточного часу в режимі перегляду дня та тижня. Це допомагає студентам та викладачам швидко орієнтуватися, скільки часу залишилося до початку або завершення пари.

Завдяки використанню FullCalendar вдалося створити інтерфейс, який за функціональністю наближається до настільних додатків (Desktop-like application). Користувачі отримали можливість миттєво перемикатися між датами, бачити структуру тижня та отримувати детальну інформацію про заняття

без зайвих переходів між сторінками.

### 2.5.4. Організація асинхронної взаємодії (AJAX API)

Сучасні веб-інтерфейси вимагають високої реактивності, яка неможлива при класичному підході повного перезавантаження сторінки (Synchronous Request-Response). Для забезпечення плавної роботи календаря у розроблюваній системі реалізовано механізм асинхронного обміну даними за технологією AJAX (Asynchronous JavaScript and XML). Хоча історична назва технології містить абревіатуру XML, у даному проєкті в якості формату обміну даними використано більш сучасний та легкий формат JSON (JavaScript Object Notation), який є нативним для веб-браузерів.

Для передачі даних про розклад із бази даних на клієнтську частину було спроєктовано спеціалізований API-шлюз. У файлі маршрутизації routes/web.php визначено окремий маршрут /api/calendar, який обробляється контролером LessonController. Цей маршрут працює за архітектурним принципом REST (Representational State Transfer), надаючи уніфікований доступ до ресурсів системи.

Особливістю реалізації є те, що бібліотека FullCalendar автоматично додає до GET-запиту параметри часового діапазону: start (початок видимого періоду) та end (кінець видимого періоду). На стороні сервера контролер перехоплює ці параметри та використовує їх для фільтрації вибірки з бази даних. Це критично важливий аспект оптимізації продуктивності: замість завантаження всього розкладу за навчальний рік (що могло б складати тисячі записів), система завантажує лише події поточного місяця або тижня.

На (Рис. 2.14) наведено програмний код методу контролера, який формує JSON-відповідь.

```
public function index(Request $request)
{
    // Створення запиту з можливістю фільтрації даних
    $events = Lesson::with(['group', 'teacher'])
        // Фільтр по викладачу (якщо обрано в select)
        ->when($request->input('teacher_id'), function ($query, $teacherId) {
            return $query->where('teacher_id', $teacherId);
        })
        // Фільтр по групі/класу (якщо обрано)
        ->when($request->input('class_id'), function ($query, $classId) {
            return $query->where('class_id', $classId);
        })
        ->get();

    // Формування масиву подій у форматі, зрозумілому для FullCalendar
    $calendarEvents = $events->map(function ($event) {
        return [
            'id' => $event->id,
            'title' => $event->group->name . ' - ' . $event->subject,
            'start' => $event->start_time,
            'end'   => $event->end_time,
            'backgroundColor' => $event->teacher->color_code ?? '#3788d8',
        ];
    });

    // Повернення даних у форматі JSON
    return response()->json($calendarEvents);
}
```

Рисунок 2.14 – Реалізація API-методу для фільтрації та повернення даних у форматі JSON

Дані, отримані з бази через ORM Eloquent, не можуть бути передані на клієнт у "сирому" вигляді, оскільки FullCalendar очікує специфічну структуру об'єкта події (Event Object). Тому в контролері реалізовано логіку трансформації даних (Data Mapping). Кожен запис уроку з таблиці lessons перетворюється на об'єкт із властивостями:

- id – унікальний ідентифікатор уроку (необхідний для редагування);
- title – заголовок події, який формується динамічно шляхом конкатенації назви предмету, групи та прізвища викладача (наприклад, "Математика (Гр. 101) - Петренко В.В.");
- start та end – часові мітки початку та завершення заняття у форматі ISO 8601 (наприклад, "2023-12-01T08:30:00").

Така трансформація дозволяє відв'язати внутрішню структуру бази даних від формату відображення: якщо в майбутньому зміниться назва поля в таблиці, достатньо буде змінити лише логіку мапінгу в контролері, не переписуючи JavaScript-код.

Перевірка коректності роботи API здійснювалася за допомогою інструментів розробника браузера (Chrome DevTools). У вкладці «Network»

можна відслідкувати XHR-запити, що відправляються календарем. Успішна відповідь сервера має статус 200 OK та містить заголовок Content-Type: application/json.

Реалізація такого механізму взаємодії дозволила досягти високої швидкодії інтерфейсу. Користувач отримує миттєвий відгук при перемиканні дат, оскільки об'єм передаваних даних є мінімальним (кілька кілобайт тексту), на відміну від традиційного підходу, де кожна дія вимагала б завантаження повної HTML-сторінки з усіма стилями та скриптами.

### 2.5.5. Алгоритмічна реалізація валідації та конфлікт-менеджменту

Фундаментальною вимогою до автоматизованої системи розкладу є гарантія відсутності організаційних колізій. На відміну від статичних електронних таблиць, де контроль за накладками покладається на уважність диспетчера, розроблений вебзастосунок реалізує жорстку програмну валідацію даних на етапі їх введення. Алгоритм перевірки базується на математичній моделі інтервальної логіки та реалізований засобами фреймворку Laravel через механізм Form Requests.

Для забезпечення чистоти коду контролерів логіку перевірки вхідних даних було винесено в окремий клас запиту App\Http\Requests\Admin\StoreLessonRequest. Цей підхід дозволяє дотримуватися принципу єдиної відповідальності (Single Responsibility Principle): контролер відповідає лише за маршрутизацію, а клас запиту — за перевірку коректності даних. У методі rules() цього класу визначено набір правил для кожного поля форми: перевірка на обов'язковість заповнення (required), відповідність типам даних (integer, date_format:H:i) та існування посилань у базі даних (exists:teachers,id). [40]

Найскладнішим елементом валідації є перевірка доступності ресурсів у часі. Задача зводиться до визначення перетину двох часових інтервалів:

інтервалу нового уроку $A = [start_A, end_A]$ та інтервалу будь-якого існуючого уроку $B = [start_B, end_B]$.

Для реалізації цієї перевірки було написано кастомне правило валідації (Custom Validation Rule). Алгоритм виконує пошук у базі даних записів, які задовольняють умову перетину. У мові SQL та конструкторі запитів Eloquent ця умова трансформується у запит, що перевіряє три виміри одночасно :

1. Просторова колізія: Чи зайнята аудиторія (Classroom) у вказаний день та час?
2. Персональна колізія: Чи має викладач (Teacher) інше заняття у цей час?
3. Групова колізія: Чи не призначено навчальній групі (Class) інший урок паралельно?

Програмна реалізація цього алгоритму використовує метод where, що групує умови. Логіка перевірки виглядає наступним чином: запис вважається конфліктним, якщо його час початку суворо менший за час завершення нового уроку, І його час завершення суворо більший за час початку нового уроку.

На (Рис. 2.15) наведено фрагмент програмного коду, який демонструє реалізацію цього алгоритму засобами PHP.

```php
<?php

namespace App\Rules;

use App\Models\Lesson;
use Illuminate\Contracts\Validation\Rule;

class LessonConflict implements Rule
{
    public function __construct(private $weekday, private $startTime, private $endTime, private $teacherId, private $classId, private $classroomId)
    {}

    public function passes($attribute, $value)
    {
        // Шукаємо в базі заняття, які перетинаються з новим
        $conflicts = Lesson::query()
            ->where('weekday', $this->weekday)
            // Логіка перетину часу: (StartA < EndB) AND (EndA > StartB)
            ->where(function ($query) {
                $query->where('start_time', '<', $this->endTime)
                      ->where('end_time', '>', $this->startTime);
            })
            // Перевірка по ресурсах: чи зайнятий Викладач АБО Група АБО Аудиторія
            ->where(function ($query) {
                $query->where('teacher_id', $this->teacherId)
                      ->orWhere('class_id', $this->classId)
                      ->orWhere('classroom_id', $this->classroomId);
            })
            ->exists();

        // Якщо знайдено хоч один запис ($conflicts == true), валідація не проходить
        return !$conflicts;
    }

    public function message()
    {
        return 'Увага! Виявлено конфлікт у розкладі (накладка по часу, аудиторії або викладачу).';
    }
}
```

Рисунок 2.15 – Програмна реалізація перевірки конфліктів у базі даних

У разі виявлення конфлікту система автоматично перериває обробку запиту. Користувач не перенаправляється на сторінку помилки; замість цього відбувається повернення на сторінку форми (Redirect Back) зі збереженням введених даних (Input Flashing) та відображенням повідомлення про помилку. Механізм MessageBag у Laravel дозволяє вивести конкретну причину відмови, наприклад: «Аудиторія 101 зайнята у період 08:30–09:50».

Такий підхід забезпечує цілісність даних на рівні бізнес-логіки застосунку. Навіть якщо два адміністратори спробують одночасно створити конфліктні записи, транзакційні механізми бази даних MySQL у поєднанні з програмною валідацією гарантують, що буде збережено лише перший валідний запис, а другий користувач отримає повідомлення про неможливість операції.

## 2.6. Тестування та налагодження програмної розробки

Етап тестування та налагодження є невід'ємною складовою життєвого циклу розробки програмного забезпечення, що забезпечує відповідність створеного продукту сформульованим вимогам. Для вебзастосунку «School Timetable» було застосовано комбіновану стратегію тестування, яка включала перевірку функціональності (Functional Testing), тестування сумісності середовищ (Environment Testing) та перевірку безпеки доступу.

Першочерговим завданням була перевірка коректності роботи алгоритму запобігання колізіям. Тестування проводилося методом «чорної скриньки» (Black Box Testing), коли тестувальник взаємодіє лише з зовнішнім інтерфейсом системи. Було змодельовано сценарій спроби створення конфліктного запису:

1. Адміністратор створює урок в аудиторії №101 на понеділок, 08:30–09:50.
2. Система успішно зберігає запис.
3. Адміністратор намагається створити інший урок для іншої групи в тій самій аудиторії на той самий час.
4. Очікуваний результат: система блокує збереження та виводить

повідомлення про помилку.

В ході експерименту підтверджено, що валідатор Laravel коректно перехоплює такі запити, повертаючи користувачеві зрозуміле повідомлення про причину відмови (Рис. 2.16).

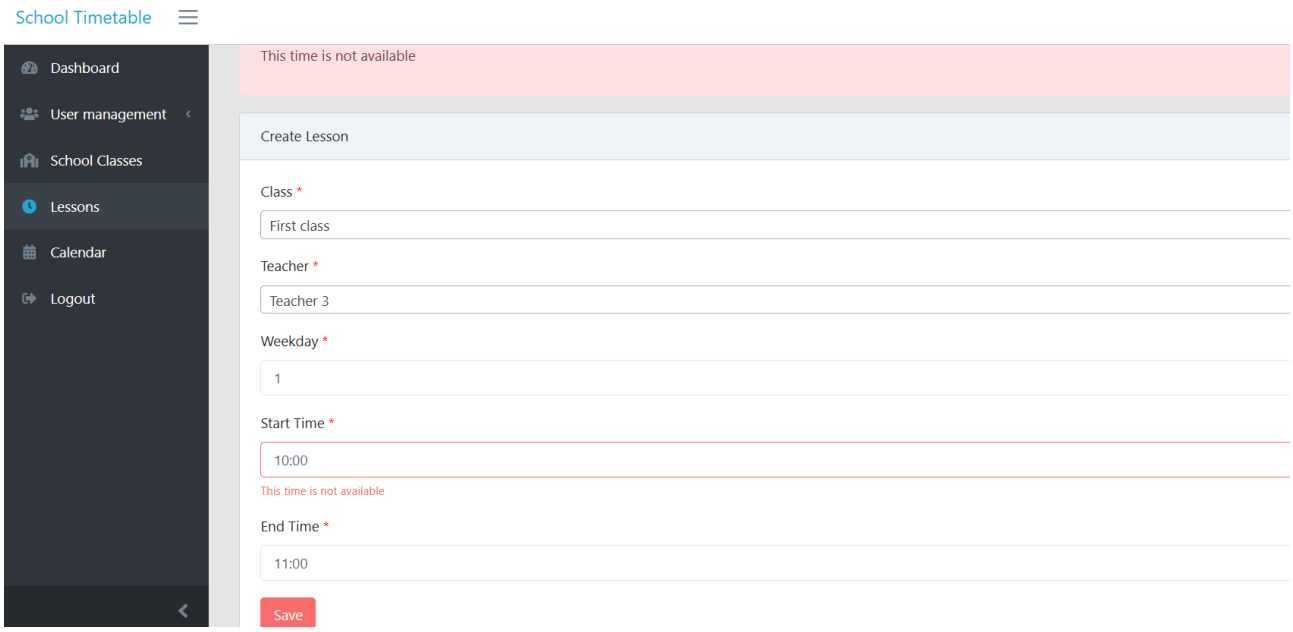



Рисунок 2.16 – Результат роботи системи валідації при спробі створення конфліктного запису

Окремий етап тестування був присвячений перевірці рольової моделі (ACL). Було проведено спробу несанкціонованого доступу до адміністративної панелі шляхом прямого введення URL-адреси /admin/lessons без авторизації. Система коректно спрацювала через механізм Middleware, перенаправивши запит на сторінку входу. Також було перевірено захист від SQL-ін'єкцій шляхом введення спеціальних символів у поля пошуку; ORM Eloquent успішно екранувала шкідливі запити, зберігши цілісність бази даних. [41]

Завершальний етап включав перевірку відображення інтерфейсу на різних пристроях. Використовуючи інструменти розробника Google Chrome (Device Toolbar), було протестовано роботу адаптивної сітки Bootstrap на емуляторах смартфонів (iPhone, Pixel) та планшетів. Виявлено, що календар FullCalendar коректно трансформується у вертикальний список (List View) на малих екранах,

що забезпечує зручність використання для студентів.

Таким чином, комплексне тестування підтвердило стабільність роботи вебзастосунку, коректність виконання бізнес-логіки та готовність системи до експлуатації в реальних умовах навчального закладу.

## 2.7. Рекомендації по використанню та впровадженню програмного засобу

Успішне впровадження інформаційної системи у навчальний процес не обмежується лише технічним розгортанням коду на сервері. Для забезпечення стабільної роботи вебзастосунку «School Timetable» та його ефективного використання адміністративним персоналом було розроблено низку рекомендацій, що стосуються безпеки, обслуговування даних та організаційного регламенту.

**Налаштування виробничого середовища (Production Environment).** Після перенесення проєкту на хостинг InfinityFree критично важливим є переведення застосунку в режим експлуатації. Під час розробки режим налагодження (APP_DEBUG=true) дозволяв бачити детальні звіти про помилки. Однак у публічному доступі це створює серйозну вразливість, оскільки повідомлення про помилки можуть розкрити структуру бази даних або шляхи до системних файлів.

У файлі конфігурації .env на хостингу необхідно встановити параметр APP_DEBUG=false та знизити рівень логування до APP_LOG_LEVEL=error. Це приховає системну інформацію від кінцевих користувачів, замінивши її на стандартні сторінки помилок (404, 503). [42]

**Регламент резервного копіювання даних (Backup Strategy).** Враховуючи використання безкоштовного тарифного плану хостингу, який не гарантує SLA (Service Level Agreement) щодо збереження даних, відповідальність за цілісність інформації покладається на адміністратора системи.

Впровадити регламент щотижневого резервного копіювання. Оскільки InfinityFree не надає доступу до SSH для автоматизації бекапів через CRON, рекомендовано використовувати вбудований інструмент phpMyAdmin для ручного експорту бази даних (Рис. 2.17).

– Формат експорту: SQL.

– Частота: Щоп'ятниці після внесення змін у розклад на наступний тиждень.

– Зберігання: Архіви баз даних повинні зберігатися локально на робочому комп'ютері диспетчера або у хмарному сховищі закладу (Google Drive).

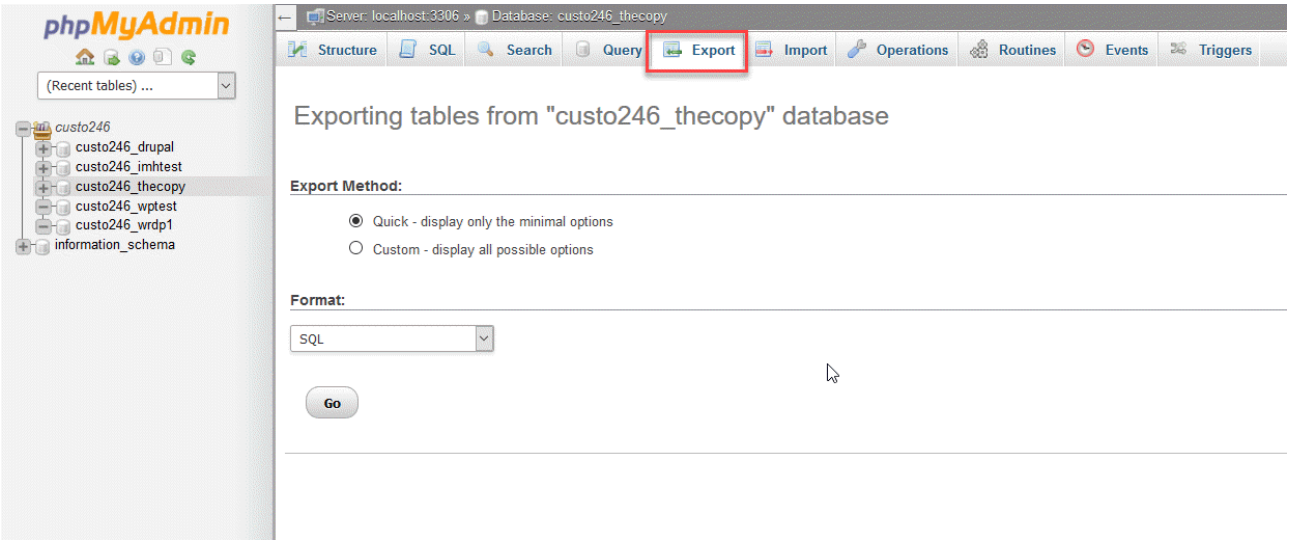



Рисунок 2.17 – Інтерфейс експорту резервної копії бази даних через phpMyAdmin

**Життєвий цикл адміністрування навчального семестру.** Ефективність роботи диспетчера залежить від дотримання правильної послідовності дій при роботі з системою. Пропонується наступний алгоритм роботи з вебзастосунком:

1. **Підготовчий етап (початок семестру):**

– Очищення розкладу попереднього семестру (або архівування бази даних).

– Актуалізація довідників: додавання нових викладачів, видалення

випускних груп, оновлення переліку аудиторій (наприклад, якщо якась аудиторія пішла на ремонт, її слід видалити з бази, щоб валідатор не дозволяв ставити туди пари).

2. **Операційний етап (протягом семестру):**

- Створення шаблону розкладу на тиждень.
- Внесення оперативних змін (перенесення пар, заміни). Система дозволяє робити це в реальному часі, і зміни миттєво відображаються у студентів.

3. **Завершальний етап (сесія):**

- Створення специфічного розкладу іспитів та консультацій, який не підпорядковується правилам регулярних тижнів (чисельник/знаменник).

**Рекомендації щодо масштабування та апаратних ресурсів.** Розроблена система на базі Laravel та MySQL є достатньо продуктивною для обслуговування факультету або коледжу з кількістю студентів до 2000-3000 осіб. При такій кількості користувачів навантаження на базу даних при перегляді кешованого розкладу є мінімальним.

Однак, у разі впровадження системи на рівні великого університету (понад 10 000 студентів), ресурсів shared-хостингу може бути недостатньо через ліміти на кількість одночасних PHP-процесів (Entry Processes).

При масштабуванні системи рекомендовано перехід на виділений віртуальний сервер (VPS) з мінімальними характеристиками: 2 CPU Core, 4 GB RAM. Це дозволить встановити Redis для кешування сесій та черг, що значно підвищить пропускну здатність системи під час пікових навантажень (наприклад, у перший день навчання). [42]

**Інструктаж персоналу.** Впровадження програмного засобу вимагає мінімального навчання персоналу. Оскільки інтерфейс адміністративної панелі спроєктовано інтуїтивно зрозумілим, навчання зводиться до пояснення принципів реакції системи на конфлікти. Диспетчер повинен розуміти, що повідомлення про помилку "Аудиторія зайнята" є не збоєм програми, а результатом роботи захисного алгоритму, який запобігає накладкам.

# ВИСНОВКИ

У магістерській роботі вирішено актуальне науково-прикладне завдання автоматизації управління навчальним розкладом шляхом створення спеціалізованого вебзастосунку. Проведене дослідження базувалося на детальному аналізі потреб освітніх закладів, який виявив, що існуючі методи планування (паперові носії, електронні таблиці Excel) не відповідають сучасним вимогам оперативності та мобільності. В умовах динамічного навчального процесу статичні файли швидко втрачають актуальність, що призводить до організаційних колізій та дезінформації учасників навчання.

В ході виконання роботи було проведено ґрунтовний порівняльний аналіз існуючих програмних аналогів. Встановлено, що комплексні системи управління навчанням (LMS) та комерційні платформи часто є надмірно складними, дорогими у впровадженні або перевантаженими зайвим функціоналом. Це обґрунтувало доцільність розробки власного легковагого програмного продукту, який фокусується на вирішенні однієї конкретної проблеми — зручному та безпомилковому веденні розкладу занять.

Важливим теоретичним результатом роботи стало обґрунтування вибору технологічного стека. Вибір мови програмування PHP (версії 7.4), фреймворку Laravel та системи управління базами даних MySQL довів свою ефективність. Така конфігурація забезпечила високу швидкість розробки, надійність роботи системи та, що найважливіше, можливість її розгортання на бюджетних або безкоштовних хостингових платформах. Використання архітектурного патерну MVC (Model-View-Controller) дозволило створити модульну структуру застосунку, де логіка обробки даних чітко відокремлена від інтерфейсу користувача, що значно спрощує подальшу підтримку та масштабування коду.

У практичній частині роботи спроєктовано та реалізовано реляційну базу даних, нормалізовану до третьої нормальної форми. Це дозволило забезпечити цілісність даних та уникнути дублювання інформації. Реалізована система зв'язків між сутностями (викладачі, групи, аудиторії) гарантує коректність

вибірок та надійне зберігання історії змін розкладу.

Ключовим технічним досягненням розробки є реалізація алгоритму автоматичної валідації часових колізій. Створений програмний модуль на етапі введення даних аналізує зайнятість аудиторій, викладачів та навчальних груп. Завдяки використанню інтервальної логіки при побудові запитів до бази даних, система фізично унеможливлює створення конфліктних записів (накладання пар). Це повністю вирішує проблему людського фактора, яка є основною причиною помилок при ручному складанні розкладу.

Для забезпечення якісної взаємодії з користувачем (User Experience) розроблено сучасний клієнтський інтерфейс із використанням технології AJAX та бібліотеки FullCalendar. Асинхронний обмін даними у форматі JSON дозволив реалізувати миттєве оновлення розкладу на екрані без перезавантаження сторінки, що значно підвищило швидкодію системи. Адаптивна верстка на базі Bootstrap забезпечила коректне відображення календаря на будь-яких пристроях — від комп'ютерів у деканаті до смартфонів студентів.

Вагомим результатом роботи є успішне впровадження та тестування розробленого вебзастосунку в реальному середовищі. Розгортання системи на хмарній платформі InfinityFree підтвердило її сумісність з технічними обмеженнями загальнодоступних хостингів. В процесі впровадження було вирішено низку інженерних задач: налаштування веб-сервера Apache, конфігурація змінних середовища та оптимізація роботи з базою даних. Розроблені рекомендації щодо експлуатації та резервного копіювання забезпечують надійність та безпеку даних при тривалому використанні системи.

Підсумовуючи, можна стверджувати, що мета магістерської роботи досягнута. Створений програмний продукт «School Timetable» є завершеним, функціональним інструментом, який готовий до використання. Він дозволяє автоматизувати рутинні процеси планування, підвищити прозорість навчального процесу та забезпечити оперативний доступ до інформації.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Garey M. R., Johnson D. S. Computers and Intractability: A Guide to the Theory of NP-Completeness. San Francisco : W. H. Freeman and Company, 1979. 340 p. URL: https://www.academia.edu/44422461/Computers_and_intractability (дата звернення: 20.11.2025).

2. Schaerf A. A Survey of Automated Timetabling. *Artificial Intelligence Review*. 1999. Vol. 13, No. 2. P. 87–127. URL: https://www.academia.edu/3323794/A_Survey_of_Automated_Timetabling (дата звернення: 20.11.2025).

3. Burke E. K., Petrovic S. Recent research directions in automated timetabling. *European Journal of Operational Research*. 2002. Vol. 140, No. 2. P. 266–280. URL: https://www.academia.edu/Recent_research_directions (дата звернення: 20.11.2025).

4. Saltos R., Maldonado S. Case Article — School Timetabling Problem: A Scheduling Problem for High-School Institutions. *INFORMS Transactions on Education*. 2023. Vol. 24, No. 1. P. 95–99. URL: https://www.academia.edu/117650848/Case_Article_School_Timetabling_Problem_A_Scheduling_Problem_for_High_School_Institutions (дата звернення: 20.11.2025).

5. Цифрова трансформація освіти: теоретико-методичні засади : монографія / Т. О. Басюк [та ін.] ; за заг. ред. В. П. Сергієнка ; за наук. ред. Н. П. Франчук. Київ : Вид-во УДУ імені Михайла Драгоманова, 2024. 382 с. URL: https://lib.iitta.gov.ua/id/eprint/745271/ (дата звернення: 20.11.2025).

6. Організація освітнього середовища : матеріали II Всеукр. наук.-практ. конф. (Запоріжжя, 28 жовт. 2025 р.) / голов. ред. : О. І. Іваницький, О. В. Пономаренко. Запоріжжя : ЗНУ, 2025. URL: https://www.znu.edu.ua/faculty/spp/nauka/ped/zbirnik_konf_oos-2025.pdf (дата звернення: 20.11.2025).

7. Nathaniel J., Danjuma K. J., Oye N. D. A Review of Timetable Scheduling

System Using Genetic Algorithm. *International Journal of Trend in Research and Development*. 2019. Vol. 6, No. 1. URL: https://www.academia.edu/104441476/A_Review_of_Timetable_Scheduling_System_Using_Genetic_Algorithm (дата звернення: 20.11.2025).

8. Nsulangi P. T., Ngongi W. E., Likamba M. R., Sarehe O. B., Mkwande M. A. A Comparative Analysis of Manual and Automatic Timetabling Approaches for Resource Utilisation in Tertiary Higher Learning Institution. *International Journal of Computer Science and Mobile Computing*. 2024. Vol. 13, No. 12. P. 65–76. URL: https://www.academia.edu/127215555/A_Comparative_Analysis_of_Manual_and_Automatic_Timetabling_Approaches_for_Resource_Utilisation_in_Tertiary_Higher_Learning_Institution (дата звернення: 20.11.2025).

9. Panko R. R. What We Know About Spreadsheet Errors. *Journal of Organizational and End User Computing*. 1998. Vol. 10, No. 2. P. 15–21. URL: https://www.academia.edu/112325329/What_We_Know_About_Spreadsheet_Errors (дата звернення: 20.11.2025).

10. Google Workspace for Education: Calendar API Limitations // Google Developers. URL: https://developers.google.com/calendar/api/guides/quota (дата звернення: 10.12.2025).

11. Berényi L. Usability analysis of the Moodle system. *Gradus*. 2025. Vol. 12, No. 1. URL: https://gradus.kefo.hu/archive/2025-1/2025_1_ART_004_Berenyi.pdf (дата звернення: 20.11.2025).

12. Аналітичний вісник у сфері освіти й науки : довідк. бюл. / НАПН України, ДНПБ України ім. В. О. Сухомлинського. Київ : ДНПБ України ім. В. О. Сухомлинського, 2023. Вип. 17. 182 с. URL: https://lib.iitta.gov.ua/id/eprint/735560/1/VNIASO-AHSEduSci-RB-17-2023.pdf (дата звернення: 20.11.2025).

13. TIOBE Index for December 2024 // TIOBE. URL: https://www.tiobe.com/tiobe-index/ (дата звернення: 14.12.2025).

14. Chauksey N., Ghotkar A. S. Extraction of Dependencies from Javascript Files using High Performance Analysis. URL:

https://www.academia.edu/37085921/Extraction_of_Dependencies_from_Javascript_Files_using_High_Performance_Analysis (дата звернення: 20.11.2025).

15. Lubanovic B. Introducing Python: Modern Computing in Simple Packages. Sebastopol : O'Reilly Media, 2014. 460 p. URL: https://dokumen.pub/introducing-python-modern-computing-in-simple-packages-9781449359362-1449359361.html (дата звернення: 20.11.2025).

16. PHP Benchmarks: The definitive guide to PHP performance (2024 Edition). *Kinsta Hosting Research*. URL: https://kinsta.com/blog/php-benchmarks/ (дата звернення: 20.11.2025).

17. Williams H. E., Lane D. Web Database Applications with PHP and MySQL. Sebastopol : O'Reilly Media, 2002. 563 p. URL: https://www.academia.edu/1044789/Web_database_applications_with_PHP_and_MySQL (дата звернення: 20.11.2025).

18. Solanki N., Shah D., Shah A. A Survey on different Framework of PHP. *International Journal of Latest Technology in Engineering, Management & Applied Science (IJLTEMAS)*. 2017. Vol. 6, No. 6. P. 155–158. URL: https://www.academia.edu/33899136/A_Survey_on_different_Framework_of_PHP (дата звернення: 20.11.2025).

19. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston : Addison-Wesley, 1994. 395 p.

20. Eloquent: Getting Started // Laravel. URL: https://laravel.com/docs/8.x/eloquent (дата звернення: 20.11.2025).

21. DuBois P. MySQL. 5th ed. Boston : Addison-Wesley Professional, 2013. 1176 p.

22. Date C. J. An Introduction to Database Systems. 8th ed. Boston : Addison-Wesley, 2004. 1328 p. URL: https://www.academia.edu/34528500/Introduction_to_Database_Systems (дата звернення: 20.11.2025).

23. Allen J. F. Maintaining knowledge about temporal intervals. *Communications of the ACM*. 1983. Vol. 26, Issue 11. P. 832–843.

24. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. Sebastopol : O'Reilly Media, 2017. 616 p.

25. A Study of Ajax Template Injection in Web Applications / A. D. Noyon et al. *International Journal of Engineering & Technology*. 2018. Vol. 7, No. 3.13. P. 123–127. URL: https://www.academia.edu/A_Study_of_Ajax_Template_Injection (дата звернення: 20.11.2025).

26. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models // International Organization for Standardization. URL: https://www.iso.org/standard/35733.html (дата звернення: 20.11.2025).

27. Wiegers K., Beatty J. Software Requirements. 3rd ed. Redmond : Microsoft Press, 2013. 672 p. URL: https://olivroqueaprende.com/WDK/Software_Requirements_3rd_Edition.pdf (дата звернення: 20.11.2025).

28. PHP-FIG. PSR-4: Autoloader. *PHP Standards Recommendations*. URL: https://www.php-fig.org/psr/psr-4/ (дата звернення: 16.12.2025).

29. Otwell T. Laravel Architecture Concepts. *Laravel Documentation*. URL: https://laravel.com/docs/8.x/structure (дата звернення: 16.12.2025).

30. Sommerville I. Software Engineering. 9th ed. Boston : Addison-Wesley, 2011. 790 p. URL: https://dn790001.ca.archive.org/0/items/bme-vik-konyvek/Software%20Engineering%20-%20Ian%20Sommerville.pdf (дата звернення: 20.11.2025).

31. Martin R. C. Agile Software Development: Principles, Patterns, and Practices. Upper Saddle River : Prentice Hall, 2002. 529 p. URL: https://desol-tech.com/buckets/Agile%20Software%20Development.pdf (дата звернення: 20.11.2025).

32. PHP Manual. Language Reference: Types. *The PHP Group*. URL: https://www.php.net/manual/en/language.types.declarations.php (дата звернення: 20.11.2025).

33. InfinityFree Knowledge Base. System Constraints and Limits. URL: https://infinityfree.net/support/limits/ (дата звернення: 20.11.2025).

34. Oracle Corporation. MySQL 5.7 Reference Manual: InnoDB Storage Engine. URL: https://dev.mysql.com/doc/refman/5.7/en/innodb-storage-engine.html (дата звернення: 20.11.2025).

35. Apache HTTP Server Documentation. .htaccess files. The Apache Software Foundation. URL: https://httpd.apache.org/docs/2.4/howto/htaccess.html (дата звернення: 20.11.2025).

36. Laravel Documentation. Blade Templates. URL: https://laravel.com/docs/8.x/blade (дата звернення: 20.11.2025).

37. Bootstrap Documentation. Grid System. URL: https://getbootstrap.com/docs/4.6/layout/grid/ (дата звернення: 20.11.2025).

38. FullCalendar Documentation. Initialization. URL: https://fullcalendar.io/docs/initialize-globals (дата звернення: 20.11.2025).

39. MDN Web Docs. DOM API. Mozilla Foundation. URL: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model (дата звернення: 20.11.2025).

40. Laravel Documentation. Validation. URL: https://laravel.com/docs/8.x/validation (дата звернення: 20.11.2025).

41. Laravel Documentation. Error Handling. URL: https://laravel.com/docs/8.x/errors (дата звернення: 20.11.2025).

42. Laravel Documentation. Deployment. URL: https://laravel.com/docs/8.x/deployment (дата звернення: 20.11.2025).

# ДОДАТОК А
# Технічне завдання

## 1. Вступ

Технічне завдання описує вимоги до створення спеціалізованого вебзастосунку для управління навчальним процесом освітнього закладу. Проєкт є програмним комплексом, реалізованим на базі фреймворку Laravel, що надає інструментарій для створення, редагування, валідації та візуалізації розкладу занять у режимі реального часу з використанням веб-інтерфейсу.

## 2. Підстави для розробки

Розробка проводиться на підставі затвердженої теми магістерської кваліфікаційної роботи для спеціальності 014 «Середня освіта (Інформатика)». Робота виконується відповідно до індивідуального плану студента та вимог випускової кафедри.

## 3. Призначення розробки

Метою розробки є створення автоматизованої інформаційної системи для заміни паперового та табличного (Excel) документообігу в навчальній частині. Система призначена для:

- Адміністраторів (диспетчерів): для формування бази даних ресурсів та складання безконфліктного розкладу.
- Викладачів та студентів: для оперативного отримання інформації про час та місце проведення занять через мобільні та стаціонарні пристрої.

## 4. Вимоги до функціональних характеристик

Система повинна забезпечувати реалізацію наступних функцій:

1. **Управління сутностями (CRUD):**
   - Створення, читання, оновлення та видалення записів про Викладачів (ПІБ, контактні дані).

- Управління фондом Аудиторій (номер, місткість, тип).
- Ведення реєстру Навчальних груп та Предметів.

2. **Логіка розкладу:**
   - Призначення занять з прив'язкою до дати, часу, викладача, групи та аудиторії.
   - Підтримка різних типів занять (лекція, практика, лабораторна робота).
3. **Алгоритмічна валідація (Conflict Prevention):**
   - Автоматична перевірка доступності аудиторії на обраний часовий інтервал перед збереженням.
   - Перевірка зайнятості викладача (унеможливлення проведення двох пар одночасно).
   - Перевірка зайнятості навчальної групи.
   - Виведення інформативних повідомлень про причину відмови у бронюванні.
4. **Інтерфейс календаря:**
   - Візуалізація розкладу у вигляді інтерактивної сітки (Month / Week / Day views).
   - Динамічне підвантаження подій без перезавантаження сторінки (AJAX).
   - Кольорова індикація різних типів подій.
5. **Безпека та доступ:**
   - Аутентифікація адміністраторів за логіном та паролем.
   - Захист адміністративної панелі від несанкціонованого доступу (Middleware).
   - Публічний доступ до перегляду розкладу для студентів (режим Read-only).

## 5. Вимоги до надійності

**Цілісність даних:** Система повинна використовувати транзакційні

механізми СУБД MySQL та зовнішні ключі (Foreign Keys) для запобігання появі некоректних зв'язків (наприклад, урок без викладача).

**Відмовостійкість:** Програмний засіб повинен коректно обробляти помилки сервера (404, 500, 503), відображаючи користувачеві зрозумілі повідомлення.

**Безпека:** Забезпечення захисту від поширених веб-атак: SQL Injection (через ORM Eloquent), XSS (через екранування Blade), CSRF (через токени форм).

## 6. Умови експлуатації

**Серверне середовище:** Веб-сервер Apache/Nginx, інтерпретатор PHP версії 7.4 або вище, СУБД MySQL 5.7+.

**Хостинг:** Система має бути оптимізована для роботи на shared-хостингу (наприклад, InfinityFree) з обмеженими ресурсами пам'яті та процесорного часу.

**Режим роботи:** Цілодобовий (24/7) доступ через мережу Інтернет.

## 7. Вимоги до інформаційної і програмної сумісності

**Кросбраузерність:** Інтерфейс повинен коректно відображатися в останніх версіях браузерів Google Chrome, Mozilla Firefox, Safari, Microsoft Edge.

**Адаптивність (Responsiveness):** Вебзастосунок повинен автоматично адаптувати верстку під розмір екрану пристрою (Desktop, Tablet, Mobile) з використанням Grid-системи Bootstrap.

**Формати даних:** Обмін даними між клієнтом і сервером повинен здійснюватися у форматі JSON.

## 8. Вимоги до програмної документації

До складу документації входять:

- Пояснювальна алгоритмів роботи системи.
- Опис архітектури системи.
- Інструкція з розгортання (Deployment Guide).

- Схема бази даних.

## 9. Стадії і етапи розробки

Процес розробки поділяється на наступні етапи:

1. **Аналітичний етап:** Аналіз предметної області, проєктування архітектури MVC та схеми бази даних.
2. **Налаштування середовища:** Встановлення PHP, Composer, Laravel, створення репозиторію.
3. **Розробка Backend:**
   - Створення міграцій та моделей даних.
   - Реалізація контролерів ресурсів.
   - Програмування логіки валідації конфліктів (Form Requests).
4. **Розробка Frontend:**
   - Верстка макетів Blade.
   - Інтеграція бібліотеки FullCalendar.js.
   - Налаштування асинхронної взаємодії (API).
5. **Тестування:**
   - Модульне тестування алгоритмів.
   - Тестування інтерфейсу на різних пристроях.
6. **Впровадження (Deployment):**
   - Завантаження файлів на хостинг InfinityFree.
   - Налаштування конфігурації .env та .htaccess.
   - Імпорт бази даних.

## 10. Порядок контролю і приймання

Контроль виконання здійснюється науковим керівником. Приймання роботи відбувається під час публічного захисту магістерської роботи.

# ДОДАТОК Б

# Інструкція користувачу

## 1. Загальні відомості

Вебзастосунок «School Timetable» – це спеціалізований програмний засіб, розроблений на базі фреймворку Laravel для автоматизації процесів складання, редагування та розповсюдження навчального розкладу в закладах освіти. Система функціонує як хмарний сервіс, що забезпечує доступ до актуальних даних у режимі реального часу через веб-браузер, усуваючи необхідність використання паперових носіїв або статичних електронних файлів.

## 2. Функціональне призначення

Інструмент призначений для вирішення двох категорій завдань: адміністративних та інформаційних.

- **Для адміністратора (диспетчера):** Програма дозволяє вести облік навчальних ресурсів (аудиторій, викладачів, груп), створювати заняття через графічний інтерфейс, автоматично перевіряти розклад на наявність часових колізій та редагувати події.
- **Для користувачів (студентів та викладачів):** Інструмент забезпечує можливість перегляду персоналізованого розкладу у зручному календарному форматі (місяць/тиждень/день) з будь-якого пристрою, підключеного до мережі Інтернет.

## 3. Опис роботи програми

### 3.1. Початок роботи та авторизація

Для початку роботи з інструментом користувачу необхідно відкрити веб-браузер (Google Chrome, Firefox, Safari тощо) та перейти за URL-адресою розгорнутого застосунку. Система має дві зони доступу: публічну та адміністративну.

- Студенти потрапляють на головну сторінку з календарем без необхідності

реєстрації.

- Для внесення змін у розклад співробітник деканату повинен натиснути кнопку «Вхід» та пройти процедуру автентифікації, ввівши логін та пароль адміністратора.

### 3.2. Налаштування довідників

Перед початком складання розкладу адміністратор повинен наповнити базу даних первинною інформацією. У меню навігації доступні розділи:

- **«Викладачі»:** Додавання профілів педагогів (ПІБ, кафедра, контактні дані).
- **«Аудиторії»:** Формування аудиторного фонду з вказанням номеру кабінету.
- **«Групи»:** Реєстрація навчальних груп та потоків. Коректне заповнення цих довідників є обов’язковим, оскільки ці дані використовуються у випадаючих списках при створенні уроків.

### 3.3. Складання розкладу

Основний робочий процес відбувається у розділі «Календар» або «Список занять». Для додавання нового уроку користувач натискає кнопку «Додати заняття». У формі, що відкрилася, необхідно обрати:

1. Навчальну групу.
2. Викладача.
3. Аудиторію.
4. Дату та часовий інтервал проведення заняття.

### 3.4. Автоматична валідація

Після натискання кнопки «Зберегти» система автоматично запускає алгоритм перевірки конфліктів. Інструмент аналізує зайнятість обраної аудиторії та викладача у вказаний час:

- Якщо конфліктів не виявлено, урок успішно додається до бази даних і

миттєво з’являється у календарі.

– Якщо виявлено накладання (наприклад, аудиторія вже зайнята іншою групою), система блокує збереження та виводить повідомлення про помилку з деталізацією причини конфлікту. Користувач повинен змінити час або місце проведення заняття.

### 3.5. Перегляд та фільтрація результатів

Після формування розкладу кінцеві користувачі можуть переглядати його в режимі інтерактивного календаря. Інструмент підтримує навігацію між тижнями та місяцями. Для зручності передбачено функцію фільтрації: студент може обрати свою групу у випадаючому списку, після чого календар відобразить лише ті заняття, що стосуються обраної групи. Дані завантажуються динамічно без перезавантаження сторінки, що забезпечує швидку роботу навіть при повільному інтернет-з’єднанні.

# ДОДАТОК В

## Скріншоти інтерфейсу

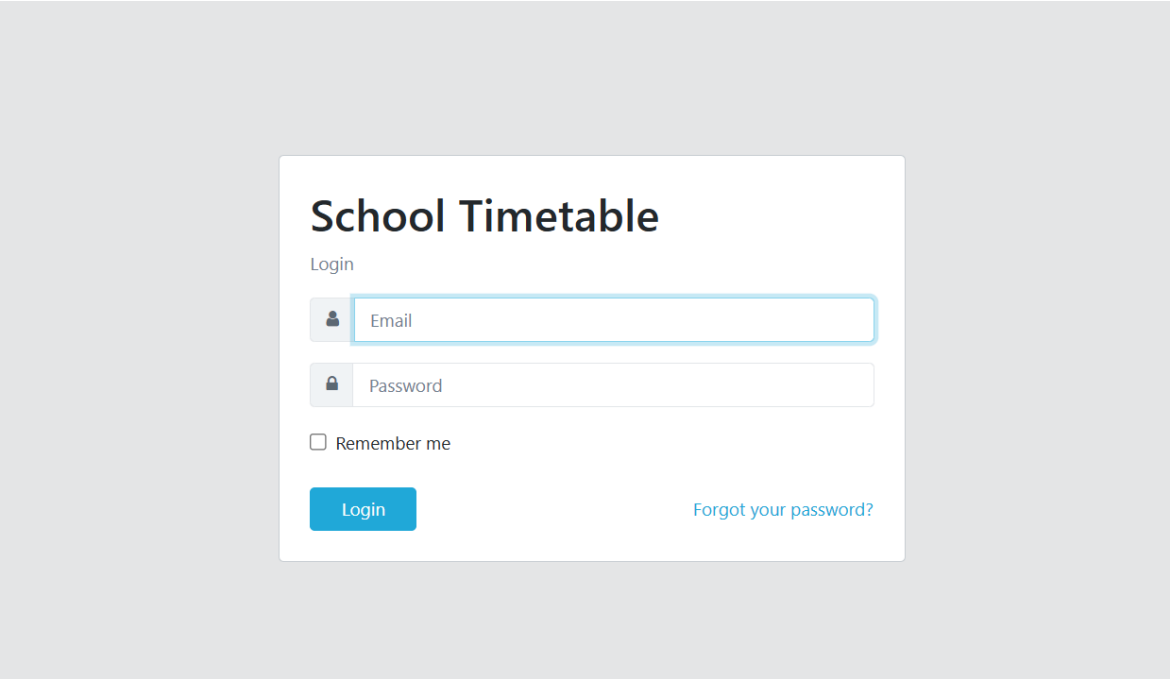



Рисунок В.1. Сторінка входу у особистий кабінет

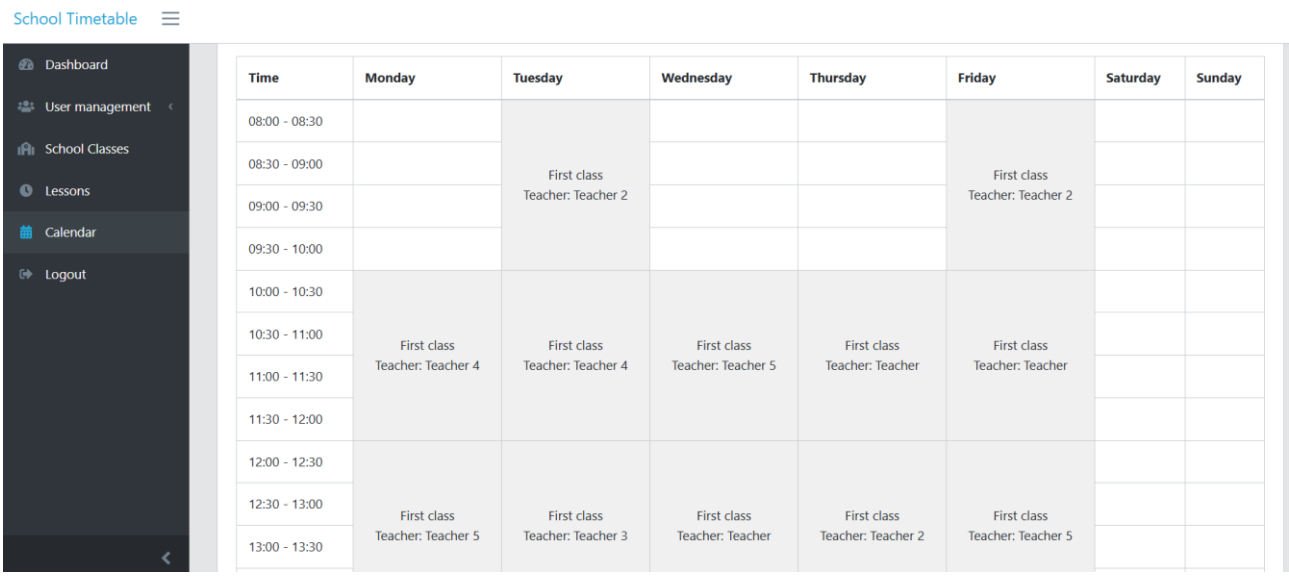



Рисунок В.2. Сторінка з розкладом

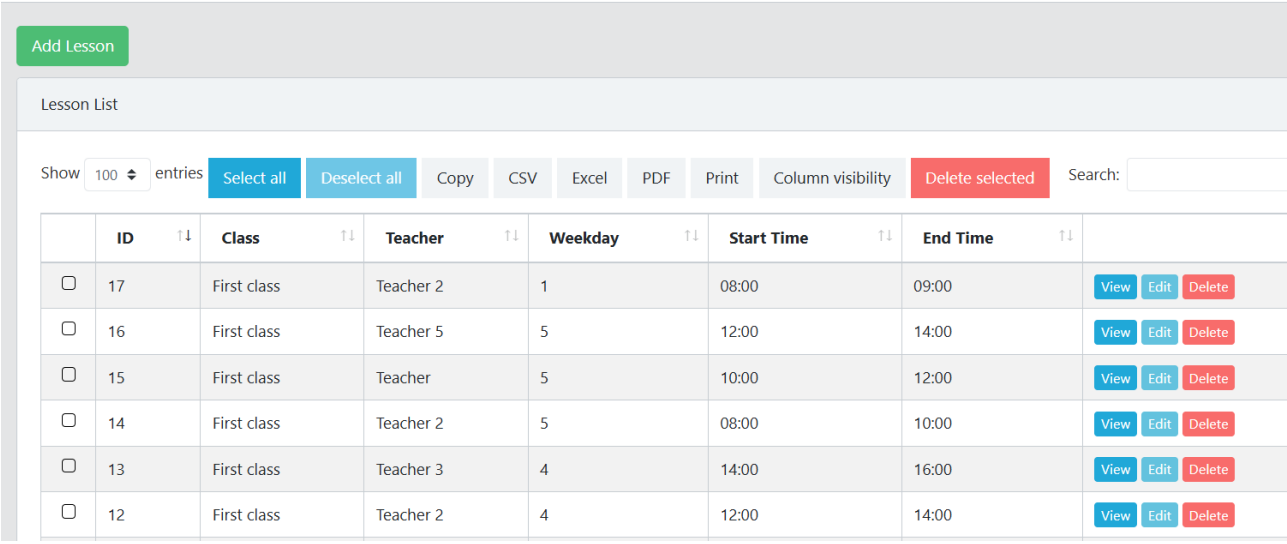



Рисунок В.3. Сторінка адміністрування уроків

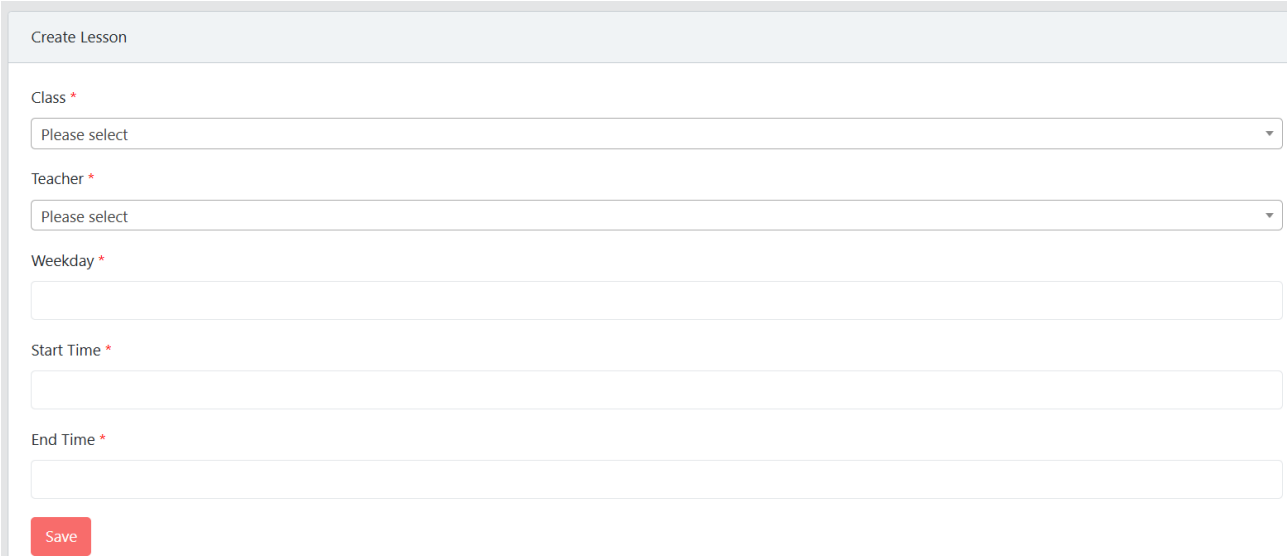



Рисунок В.4. Форма створення нового уроку

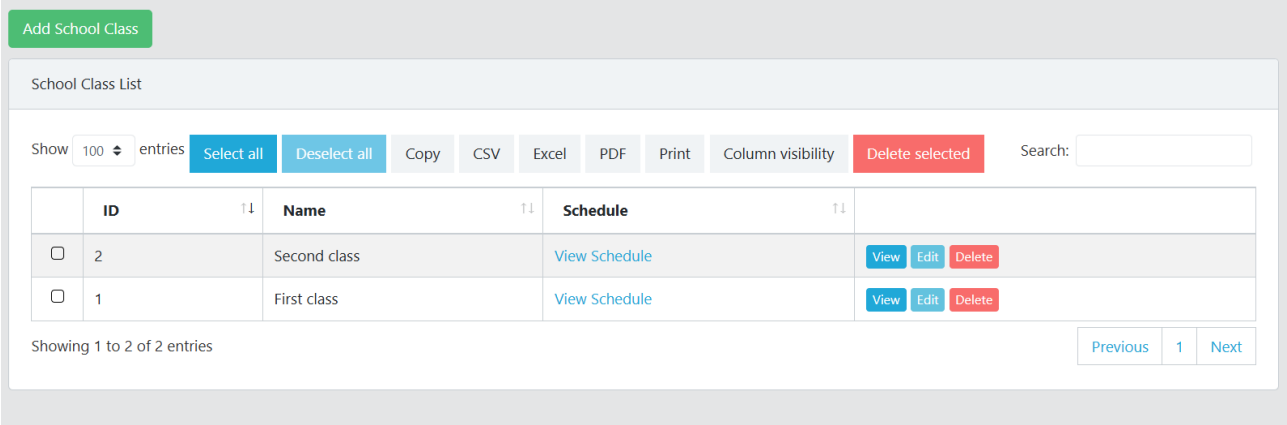



Рисунок В.4. Сторінка управління класами

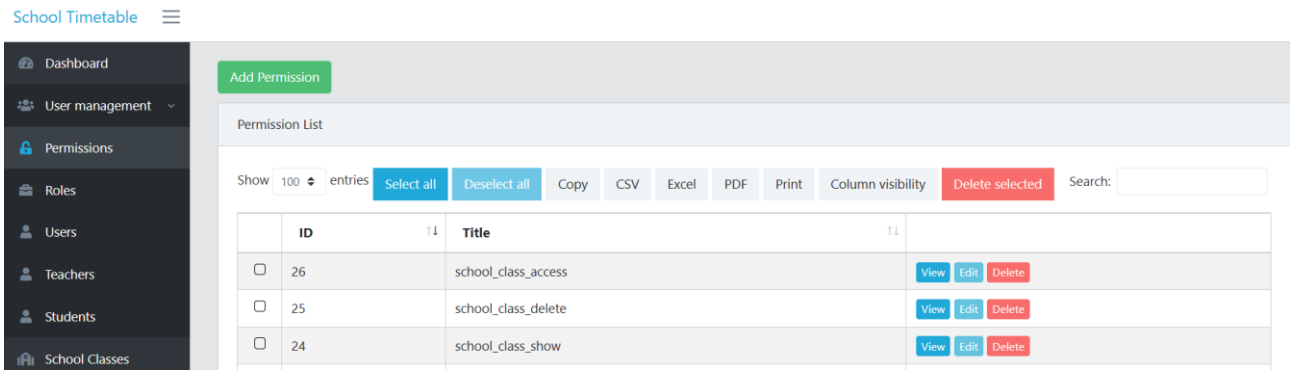



Рисунок В.5. Сторінка керування дозволами

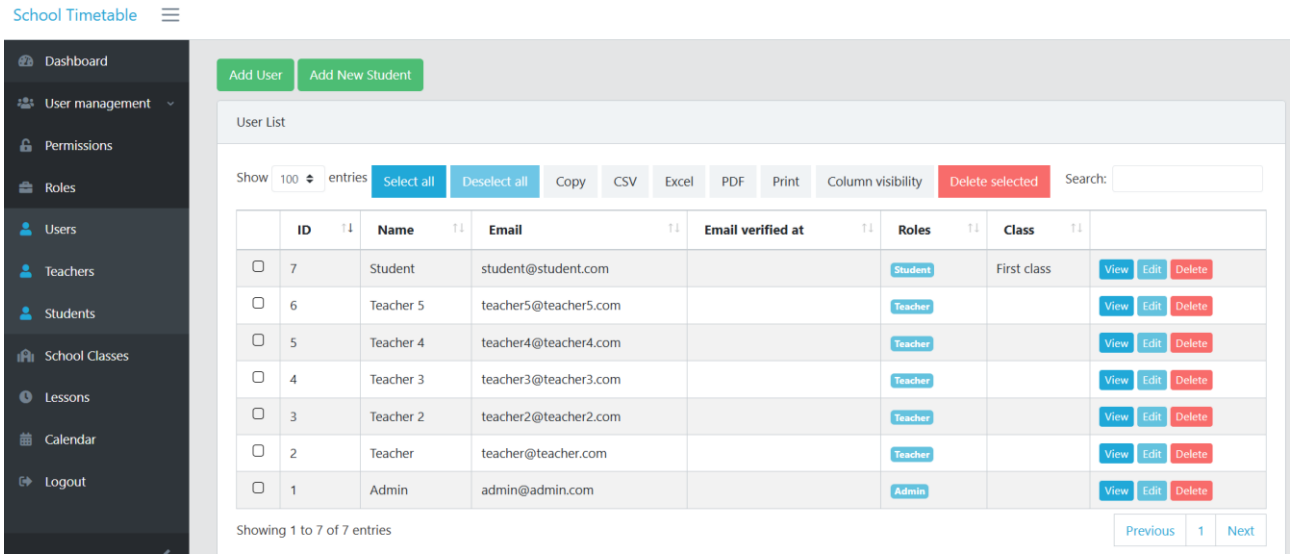



Рисунок В.6. Сторінка управління користувачами

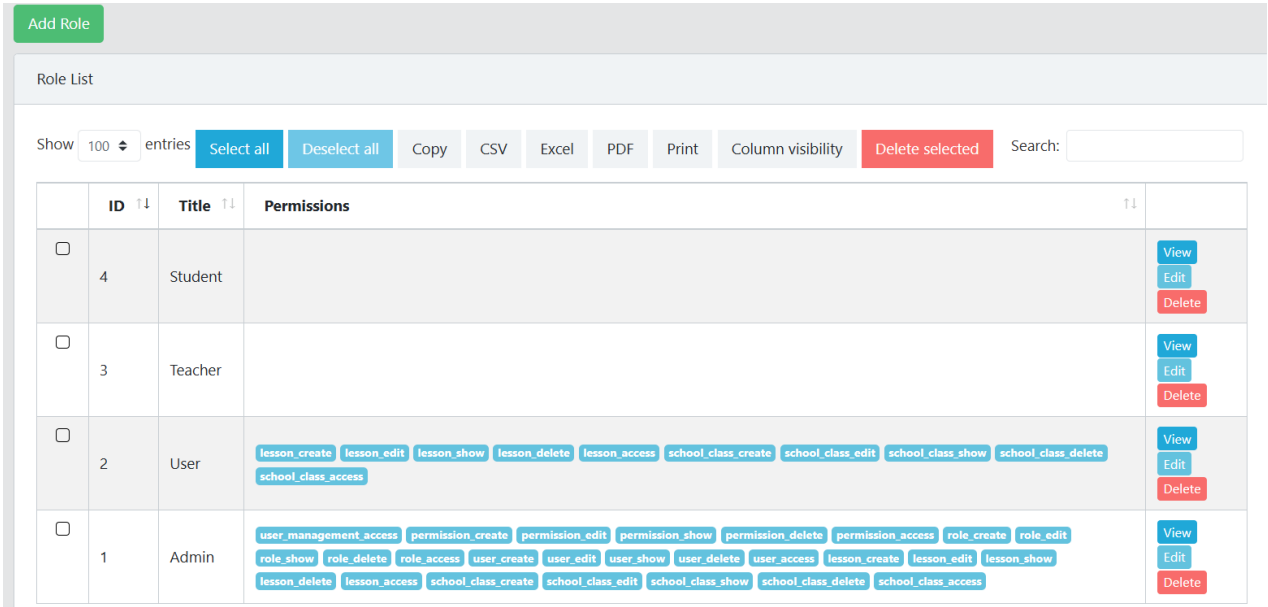



Рисунок В.7. Сторінка призначення ролей

# АНОТАЦІЯ

**Ковальчук В.В. Розробка вебзастосунку для динамічного перегляду та адміністрування розкладу занять навчального закладу**– Рукопис.

Кваліфікаційна робота на здобуття освітнього ступеня «магістр» за спеціальністю 014 Середня освіта (Інформатика). – Волинський національний університет імені Лесі Українки, Луцьк, 2025 р.

Ця кваліфікаційна робота присвячена розробці спеціалізованого вебзастосунку для автоматизації процесів складання та адміністрування розкладу занять на основі фреймворку Laravel. Враховуючи неефективність застарілих методів планування та потребу учасників освітнього процесу в оперативному доступі до актуальної інформації, розробка таких інструментів є надзвичайно актуальною.

У першому розділі досліджено теоретико-методологічні засади автоматизації управління навчальним процесом. Проведено порівняльний аналіз існуючих аналогів. Обґрунтовано вибір технологічного стека (PHP, MySQL, Laravel) та архітектурного патерну MVC для вирішення поставленої задачі.

Другий розділ присвячено проєктуванню, розробці та розгортанню вебзастосунку. Описано постановку задачі, структуру бази даних та інформаційних потоків. Детально розглянуто програмну реалізацію серверної частини, зокрема алгоритму автоматичної валідації часових колізій, а також створення інтерактивного клієнтського інтерфейсу. Описано процес тестування та особливості розгортання системи на хостингу.

Результати дослідження показали, що розроблений інструмент успішно впроваджено на хмарній платформі, що забезпечує зручність та доступність для адміністраторів, викладачів та студентів. Інструмент дозволяє уникати організаційних конфліктів при плануванні та забезпечує відображення розкладу в режимі реального часу.

**Ключові слова:** вебзастосунок, навчальний розклад, Laravel, PHP, автоматизація, валідація даних, MySQL, FullCalendar.

# ABSTRACT

**Kovalchuk V.V. Development of a web application for dynamic viewing and administration of an educational institution's timetable** – Manuscript.

Qualification thesis for the degree of Master in specialty 014 Secondary Education (Informatics). – Lesya Ukrainka Volyn National University, Lutsk, 2025.

This qualification paper is dedicated to the development of a specialized web application for automating the processes of compiling and administering class timetables based on the Laravel framework. Given the inefficiency of obsolete planning methods and the need for educational process participants to have immediate access to up-to-date information, the development of such tools is highly relevant.

The first chapter examines the theoretical and methodological foundations of educational process management automation. A comparative analysis of existing analogues was conducted. The choice of the technology stack (PHP, MySQL, Laravel) and the MVC architectural pattern for solving the set task is substantiated.

The second chapter is devoted to the design, development, and deployment of the web application. The problem statement, database structure, and information flows are described. The software implementation of the server side, particularly the algorithm for automatic validation of time conflicts, as well as the creation of an interactive client interface, are examined in detail. The testing process and the specifics of system deployment on a hosting platform are described.

The research results indicate that the developed tool has been successfully deployed on a cloud platform, ensuring convenience and accessibility for administrators, teachers, and students. The tool allows avoiding organizational conflicts during planning and ensures real-time schedule display.

**Keywords:** web application, school timetable, Laravel, PHP, automation, data validation, MySQL, FullCalendar.