МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Острозька академія»
Навчально-науковий інститут інформаційних технологій та бізнесу
Кафедра інформаційних технологій та аналітики даних

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня бакалавра

на тему: «**Розробка інтегрованої системи управління навчальним процесом на основі розкладу занять із функціоналом електронного журналу, завантаженням матеріалів та автоматизованого формування звітів»**

Виконав: студент 4 курсу, групи КН-41
першого (бакалаврського) рівня вищої освіти
спеціальності 122 Комп'ютерні науки
освітньо-професійної програми «Комп'ютерні науки»
Петро Олександрович Бурчак

Керівник: *доктор філософії з прикладної математики, викладач, фахівець-практик*
Богдан Віталійович Красюк

Рецензент: *кандидат технічних наук, доцент, доцент кафедри прикладної математики Донецького національного університету імені Василя Стуса*
Загоруйко Любов Василівна

РОБОТА ДОПУЩЕНА ДО ЗАХИСТУ
Завідувач кафедри інформаційних технологій та аналітики даних
_____ (проф., д.е.н. Кривицька О.Р.)
Протокол № ____ від « ___ » _____ 2026 р.

Острог, 2026

АНОТАЦІЯ
кваліфікаційної роботи
на здобуття освітнього ступеня бакалавра

Тема: *Розробка інтегрованої системи управління навчальним процесом на основі розкладу занять із функціоналом електронного журналу, завантаження матеріалів та веденням документообігу*

Автор: *студент спеціальності* 122 *Комп'ютерні науки освітньо-професійної програми «Комп'ютерні науки*
Бурчак Петро Олександрович

Науковий керівник: *доктор філософії з прикладної математики, викладач, фахівець-практик*
Богдан Віталійович Красюк

Захищена «……»…… 20__ року.

Пояснювальна записка до кваліфікаційної роботи: ***92*** *(кількість сторінок роботи) с.,* ***41*** *(кількість рисунків) рис., 0 (кількість таблиць) табл.,* ***6*** *(кількість додатків) додатків,* ***35*** *(кількість джерел) джерел.*

Ключові слова*: ВЕБЗАСТОСУНОК, ІНТЕГРОВАНА СИСТЕМА, РОЗКЛАД ЗАНЯТЬ, ЕЛЕКТРОННИЙ ЖУРНАЛ, БАЗА ДАНИХ, FASTAPI, VUE.JS, TELEGRAM-БОТ, ВЕБДОСТУПНІСТЬ.*

Короткий зміст праці: *Кваліфікаційна робота присвячена проєктуванню та розробці інтегрованої вебсистеми управління навчальним процесом для закладів вищої освіти. Метою роботи є автоматизація процесів організації навчання та створення єдиного цифрового середовища для адміністраторів, викладачів і студентів. У процесі дослідження проаналізовано існуючі освітні платформи та обґрунтовано необхідність розробки власного рішення. Спроєктовано багаторівневу клієнт-серверну архітектуру на базі концепції REST API. Серверну частину реалізовано за допомогою асинхронного фреймворку FastAPI та реляційної бази даних PostgreSQL. Клієнтський застосунок створено на базі фреймворку Vue.js 3 із суворим дотриманням міжнародних стандартів вебдоступності WCAG 2.1 AA. Як альтернативний мобільний інтерфейс успішно інтегровано Telegram-бот. Практично реалізовано програмний продукт, що включає інструменти управління часовою сіткою, ведення інтерактивного електронного журналу, модуль перевірки завдань та генерації звітності. Система повністю готова до впровадження у реальний освітній процес.*

ABSTRACT
of the qualification
work for a Bachelor's degree

Theme : *Development of an integrated system for managing the teaching process based on a timetable, featuring an electronic register, the ability to upload materials, and document management*

Author: *Student of degree programme 122 Computer Science within the vocational education programme 'Computer Science'*
Petro Oleksandrovych Burchak

Supervisor*: PhD in Applied Mathematics, Lecturer, Practising Specialist*
Bohdan Vitaliyovych Krasiuk

Defended on "……"…… 20__.

Explanatory note to the qualification work*:* ***92*** *(number of pages) pp.,* ***41*** *(number of figures) figs.,* ***0*** *(number of tables) tables,* ***6*** *(number of appendices) appendices,* ***35*** *(number of sources) sources.*

Keywords*: WEB APPLICATION, INTEGRATED SYSTEM, TIMETABLE, ELECTRONIC REGISTER, DATABASE, FASTAPI, VUE.JS, TELEGRAM BOT, WEB ACCESSIBILITY.*

Abstract*: This thesis is devoted to the design and development of an integrated web-based system for managing the educational process in higher education institutions. The aim of the work is to automate the processes of organising teaching and to create a unified digital environment for administrators, lecturers and students. The research analyses existing educational platforms and justifies the need to develop a proprietary solution. A multi-tier client-server architecture based on the REST API concept has been designed. The server-side has been implemented using the FastAPI asynchronous framework and a PostgreSQL relational database. The client application was built using the Vue.js 3 framework, with strict adherence to the WCAG 2.1 AA international web accessibility standards. A Telegram bot has been successfully integrated as an alternative mobile interface. A software product has been practically implemented, comprising tools for timetable management, maintaining an interactive electronic register, a task verification module, and a reporting generation module. The system is fully ready for implementation in the actual educational process.*

ЗМІСТ

ВСТУП

У сучасних умовах цифрової трансформації освіти ефективне управління навчальним процесом вимагає надійних та зручних інформаційних систем. Процес інформатизації навчальних закладів супроводжується необхідністю створення комплексних рішень, що забезпечують централізоване управління навчальним процесом, автоматизацію рутинних операцій та забезпечення доступу до інформації для всіх учасників освітнього процесу. Існуючі рішення часто мають обмежений функціонал, недостатню інтеграцію між компонентами або незручний інтерфейс для користувачів, що знижує ефективність їх використання.

Розробка інформаційної системи управління розкладом та університетською інформацією є актуальною задачею, оскільки вона дозволяє вирішити проблему розрізненості даних про навчальний процес, забезпечити єдине інформаційне простір для студентів, викладачів та адміністрації, а також автоматизувати процеси обліку відвідуваності, оцінювання та надання навчальних матеріалів. Використання сучасних веб-технологій та мобільних додатків забезпечує доступність системи з будь-якого пристрою в будь-який час, що є важливим аспектом в умовах сучасної освіти.

Мета та задачі дослідження

Метою дослідження є створення програмного продукту у вигляді інформаційної системи управління розкладом та університетською інформацією для закладів освіти, яка забезпечує централізоване управління навчальним процесом, автоматизацію обліку оцінок та відвідуваності, а також забезпечення доступу до інформації через веб-інтерфейс та Telegram бота.

Для досягнення поставленої мети вирішуються наступні **задачі дослідження**.

- **А**наліз предметної області, що включає вивчення існуючих рішень для управління навчальним процесом, аналіз вимог користувачів, визначення функціональних та нефункціональних вимог до системи, а також дослідження сучасних технологій та інструментальних засобів для розробки подібних систем. **Деталі:**
    - Вивчення рішень для управління навчальним процесом в сучасних навчальних закладах

    - Аналіз вимог користувачів Національного університету "Острозька академія" методом опитування чи всебічного аналізу потреб.
    - Конкретні функціональні вимоги (управління розкладом, журнал оцінок та відвідуваності, система завдань, новини та події, волонтерство)
    - Нефункціональні вимоги (продуктивність, масштабованість, безпека, доступність)
- Проектуванні інформаційної системи, що включає планування роботи учасників команди на різних етапах, розробку архітектури системи, визначення структури бази даних, проектування API ендпоінтів, розробку структури веб-інтерфейсу та Telegram бота, а також визначення інтеграцій із зовнішніми сервісами. **Деталі:**
    - Планування роботи команди з використанням Епіків та User Stories. Оформлення на обраній платформі Miro.
    - Архітектура клієнт-сервер з трьома компонентами (REST API, веб-інтерфейс, Telegram бот)
    - Структура бази даних PostgreSQL з конкретними таблицями
    - API ендпоінти для різних функцій
    - Структура веб-інтерфейсу на Vue.js
    - Структура Telegram бота на aiogram
    - Інтеграції з Google Drive API та Telegram API
- **В**ибір інструментарію, методів реалізації та тестування програмного продукту, що включає вибір технологічного стеку для кожного компонента системи, визначення методів тестування, вибір інструментів для автоматизації тестування та забезпечення якості програмного продукту. **Деталі:**
    - Технологічний стек (FastAPI, Vue.js, aiogram, PostgreSQL, SQLAlchemy, Pydantic)
    - Методи тестування (unit, integration, e2e мануальне)
    - Інструменти автоматизації (pytest, testcontainers)
    - Забезпечення якості (middleware, валідація, захист від вразливостей)

Об'єкт та предмет дослідження

Об'єктом дослідження є інформаційна система управління розкладом та університетською інформацією, що складається з трьох основних

компонентів, а саме серверної частини на базі REST API, веб-інтерфейсу для адміністративного управління та Telegram бота для забезпечення мобільного доступу до інформації.

Предметом дослідження є інструментальні засоби дослідження об'єкта роботи, які включають математичні моделі для організації даних та їх зв'язків, методи та інформаційні технології для реалізації системи, а також інструменти для розробки, тестування та забезпечення якості програмного продукту.

РОЗДІЛ І. ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1. Опис предметного середовища (функціональної моделі, процесу діяльності)

Предметним середовищем даної роботи є сфера управління навчальним процесом у вищих навчальних закладах, зокрема автоматизація процесів розробки розкладу занять, ведення електронного журналу, обліку відвідуваності студентів, управління навчальними матеріалами та забезпечення мобільного доступу до інформації через месенджери. Управління навчальним процесом є комплексною задачею, що вимагає інтеграції різних компонентів для забезпечення ефективної роботи освітнього закладу та створення єдиного інформаційного простору для всіх учасників освітнього процесу.

Сучасні умови цифрової трансформації освіти характеризуються активним впровадженням інформаційних технологій у всі аспекти навчального процесу. Це відповідає державній політиці, зокрема концептуальним засадам цифровізації освіти та **Положенню про Єдину державну електронну базу з питань освіти (затвердженому постановою КМУ від 24 лютого 2021 р. № 872)**, що створює підґрунтя для переходу навчальних закладів до цифрових форматів, формування єдиних освітніх середовищ та автоматизації управління навчальним процесом». Тому цей процес вимагає створення комплексних інформаційних систем, що забезпечують централізоване управління всіма аспектами навчального процесу.

Управління навчальним процесом можна визначити як сукупність взаємопов'язаних процесів, спрямованих на організацію, координацію та контроль навчальної діяльності в закладі освіти. Ці процеси включають планування навчального процесу, розробку розкладу занять, ведення обліку успішності студентів, контроль відвідуваності, управління навчальними матеріалами та забезпечення комунікації між учасниками освітнього процесу. Основною метою управління навчальним процесом є створення умов для ефективної організації навчальної діяльності та забезпечення якості освітніх послуг.

Сучасні тенденції в області управління навчальним процесом характеризуються переходом від традиційних паперових методів до цифрових платформ, інтеграцією з хмарними сервісами, використанням мобільних додатків та месенджерів для забезпечення доступності інформації. Зростання популярності дистанційного та змішаного

навчання вимагає нових підходів до організації навчального процесу, що включають автоматизацію рутинних операцій, миттєвий обмін інформацією та забезпечення доступу до даних з будь-якої точки світу.

1.1.1.Архітектурні особливості та нормативно-правова база

Багатокомпонентні інформаційні системи управління навчальним процесом являють собою складні системи, що складаються з кількох взаємопов'язаних компонентів, кожен з яких виконує специфічні функції. До таких компонентів належать серверна частина (backend), що забезпечує обробку бізнес-логіки та роботу з даними; веб-інтерфейс (frontend), що забезпечує взаємодію з користувачем; та інтеграційні компоненти, що забезпечують взаємодію з зовнішніми системами та сервісами, включаючи Telegram бота для мобільного доступу. Розробка таких систем вимагає комплексного підходу, що охоплює всі компоненти та їх взаємодію між собою.

Нормативно-правова база в області управління навчальним процесом включає міжнародні та національні стандарти, що визначають вимоги до якості інформаційних систем та процесів їх розробки. Основним міжнародним стандартом є ISO/IEC 25010:2023 "Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models", який визначає модель якості програмного продукту, що включає вісім основних характеристик: функціональна придатність, продуктивність, сумісність, зручність використання, надійність, безпека, супроводжуваність та переносимість. Цей стандарт служить основою для розробки інформаційних систем управління навчальним процесом.

Нормативні документи Міністерства освіти і науки України встановлюють вимоги до ведення електронної документації в навчальних закладах, зберігання даних про студентів та забезпечення захисту персональних даних. Ці документи підкреслюють важливість створення єдиного інформаційного простору для управління навчальним процесом.

1.1.2. Класифікація систем управління навчальним процесом

Класифікація інформаційних систем управління навчальним процесом базується на різних критеріях, включаючи функціональне призначення, масштаб впровадження та архітектурний підхід. За функціональним призначенням виділяють системи управління розкладом, електронні журнали оцінок, системи обліку відвідуваності, платформи електронного навчання, системи управління навчальними матеріалами та

адміністративні панелі. Кожен тип систем має свої особливості та вимоги до реалізації.

За масштабом впровадження інформаційні системи поділяються на локальні системи (в межах одного закладу), галузеві системи (в межах регіону) та національні системи (в межах країни). Локальні системи забезпечують управління навчальним процесом в межах одного закладу, галузеві системи координують роботу кількох закладів в межах регіону, а національні системи забезпечують єдиний підхід до управління освітою на рівні країни.

За архітектурним підходом виділяють монолітні системи, мікросервісну архітектуру та хмарні рішення (SaaS). Монолітні системи мають єдину кодову базу та базу даних, мікросервісна архітектура розбиває систему на незалежні сервіси, а хмарні рішення забезпечують доступ до системи через інтернет без необхідності встановлення програмного забезпечення на локальних комп'ютерах.

1.1.3. Функціональна модель управління навчальним процесом

Функціональна модель процесу управління навчальним процесом включає кілька основних етапів: планування навчального процесу, розробка розкладу, ведення обліку успішності, контроль відвідуваності, управління навчальними матеріалами та забезпечення комунікації. Планування навчального процесу включає визначення навчальних дисциплін, розподіл навчального навантаження між викладачами, формування навчальних груп та визначення аудиторного фонду.

Розробка розкладу є одним з найскладніших етапів, що вимагає врахування множини обмежень та факторів: доступність аудиторій, розклад викладачів, навчальні плани груп, перерви між заняттями, транспортну доступність аудиторій та інші фактори. Автоматизація цього процесу дозволяє значно скоротити час на розробку розкладу та забезпечити оптимальне використання ресурсів закладу.

Ведення обліку успішності включає реєстрацію оцінок студентів, розрахунок середнього балу, формування звітів про успішність та забезпечення доступу до інформації про оцінки для студентів та викладачів. Електронний журнал дозволяє автоматизувати розрахунок статистичних показників, генерувати звіти та забезпечувати прозорість процесу оцінювання.

Контроль відвідуваності студентів є важливим аспектом управління навчальним процесом, що дозволяє відстежувати присутність студентів на

заняттях, виявляти проблеми з відвідуваністю та вживати заходів для їх вирішення. Автоматизований контроль відвідуваності дозволяє формувати звіти про відвідуваність, аналізувати причини пропусків та забезпечувати оперативне інформування батьків та адміністрації.

Управління навчальними матеріалами включає зберігання лекцій, методичних матеріалів, завдань для студентів та інших навчальних ресурсів. Інтеграція з хмарними сервісами, такими як Google Drive, дозволяє забезпечити надійне зберігання матеріалів та легкий доступ до них з будь-якого пристрою.

Забезпечення комунікації між учасниками освітнього процесу є важливим аспектом сучасних інформаційних систем. Використання Telegram бота дозволяє забезпечити миттєве інформування про зміни в розкладі, надсилання сповіщень про нові завдання, надання доступу до оцінок та відвідуваності через мобільний пристрій.

1.1.4. Основні актори (користувачі) та їхні ролі

До основних акторів (учасників процесу) належать адміністратор системи, викладачі, студенти, батьки студентів та автоматизована система. Адміністратор системи забезпечує технічну підтримку, управління користувачами та налаштування системи. Викладачі ведуть електронний журнал, виставляють оцінки, контролюють відвідуваність та надають навчальні матеріали. Студенти отримують доступ до розкладу, оцінок, завдань та навчальних матеріалів. Батьки можуть контролювати успішність та відвідуваність своїх дітей. Автоматизована система обробляє запити, зберігає дані та забезпечує функціонування всіх компонентів.

Процес починається з планування навчального процесу, що включає визначення навчальних дисциплін, формування навчальних груп та розподіл навчального навантаження. Наступним етапом є розробка розкладу занять з урахуванням всіх обмежень та факторів. Після розробки розкладу проводиться налаштування системи, включаючи створення користувачів, завантаження навчальних планів та налаштування параметрів системи.

Після налаштування системи починається основний цикл управління навчальним процесом, що включає ведення розкладу, облік успішності, контроль відвідуваності та управління навчальними матеріалами. Викладачі виставляють оцінки в електронному журналі, відмічають

відвідуваність студентів та завантажують навчальні матеріали. Студенти отримують доступ до інформації через веб-інтерфейс або Telegram бота.
Процес є ітераційним і включає постійний моніторинг стану системи, аналіз ефективності її використання та внесення змін для покращення функціональності. Адміністратор системи забезпечує технічну підтримку, вирішує проблеми користувачів та проводить оновлення системи.

1.1.5. Специфіка реалізації та загальні висновки

Специфіка управління різними аспектами навчального процесу вимагає застосування різних підходів та інструментів. Управління розкладом вимагає алгоритмів оптимізації для знаходження оптимального розподілу занять в часі та просторі. Ведення електронного журналу вимагає надійної системи збереження даних та механізмів валідації оцінок. Контроль відвідуваності вимагає інтеграції з системами ідентифікації студентів та механізмів автоматичної фіксації присутності.
Управління навчальними матеріалами вимагає інтеграції з хмарними сервісами для забезпечення надійного зберігання та легкого доступу до матеріалів. Розробка Telegram бота вимагає врахування специфіки роботи месенджерів, обробки асинхронних повідомлень та забезпечення безпеки передачі даних.
Особливістю управління навчальним процесом в сучасних умовах є необхідність забезпечення доступності системи з різних пристроїв, включаючи комп'ютери, планшети та смартфони. Це вимагає розробки адаптивного веб-інтерфейсу, оптимізації для мобільних пристроїв та забезпечення швидкого доступу до інформації через Telegram бота.
Таким чином, предметне середовище управління навчальним процесом являє собою складну систему, що вимагає комплексного підходу, включаючи різні компоненти, спеціалізовані інструменти та врахування специфіки кожного аспекту навчального процесу. Сучасні тенденції в області цифрової трансформації освіти спрямовані на створення єдиних інформаційних просторів, автоматизацію рутинних операцій та забезпечення високої якості освітніх послуг.

1.2. Огляд наявних аналогів

Огляд наявних аналогів є важливим етапом аналізу предметного середовища, що дозволяє виявити переваги та недоліки існуючих рішень, визначити прогалини в функціональності та сформулювати вимоги до розроблюваної системи. Сучасний ринок інформаційних систем управління навчальним процесом представлений значною кількістю як

комерційних, так і відкритих рішень, що мають різний функціонал, архітектуру та вартість впровадження. Проведення детального аналізу існуючих аналогів дозволяє уникнути повторення помилок, використати кращі практики та розробити систему, що відповідає сучасним вимогам.

1.2.1. Вітчизняні рішення для управління навчальним процесом.

Серед вітчизняних рішень варто відзначити низку систем, що розроблені українськими компаніями та використовуються в навчальних закладах України. Одним з найбільш відомих рішень є система "Розклад КПІ", що використовується в Київському політехнічному інституті для управління розкладом занять. Ця система забезпечує перегляд розкладу для студентів та викладачів, має веб-інтерфейс та мобільну версію. Проте функціонал цієї системи обмежений лише управлінням розкладом і не включає електронний журнал, облік відвідуваності та інші компоненти, що необхідні для комплексного управління навчальним процесом.

Іншим прикладом вітчизняного рішення є система "Деканат", що є комплексною системою управління навчальним процесом для вищих навчальних закладів. Ця система включає модулі для управління розкладом, ведення електронного журналу, обліку відвідуваності, управління навчальними матеріалами та адміністративних функцій. Система "Деканат" має веб-інтерфейс та підтримує роботу з базами даних. Проте ця система має обмежені можливості інтеграції з зовнішніми сервісами, відсутній мобільний додаток для забезпечення доступу через смартфони, а інтерфейс потребує модернізації відповідно до сучасних вимог usability.

Система "ВУЗ-Україна" є ще одним прикладом вітчизняної розробки, що призначена для управління навчальним процесом в університетах. Ця система забезпечує ведення контингенту студентів, управління навчальними планами, розробку розкладу та ведення електронного журналу. Система має модульну архітектуру та підтримує налаштування під специфіку конкретного навчального закладу. Проте ця система має обмежені можливості масштабування, відсутня інтеграція з популярними хмарними сервісами, а система сповіщень про зміни в розкладі реалізована недостатньо ефективно.

Аналіз вітчизняних рішень показав, що більшість з них мають обмежений функціонал, недостатню інтеграцію між модулями, обмежені можливості масштабування та відсутність сучасних механізмів забезпечення мобільного доступу до інформації. Більшість систем орієнтовані на

вирішення окремих задач (наприклад, лише управління розкладом або лише ведення електронного журналу) і не забезпечують комплексний підхід до управління навчальним процесом.

1.2.2. Міжнародні рішення та платформи дистанційного навчання (LMS)

На міжнародному ринку представлено значно більше рішень для управління навчальним процесом, що мають розширений функціонал та використовуються в навчальних закладах по всьому світу. Одним з найпопулярніших рішень є Moodle — система управління навчанням (Learning Management System), що є відкритим програмним забезпеченням та використовується в тисячах навчальних закладів по всьому світу. Moodle забезпечує широкий функціонал, включаючи управління курсами, ведення електронного журналу, систему завдань, форуми, чати та інші інструменти для організації навчального процесу. Система має модульну архітектуру, що дозволяє розширювати функціонал за допомогою плагінів. Проте Moodle має складний інтерфейс, що потребує тривалої адаптації користувачів, відсутня вбудована система управління розкладом, а інтеграція з зовнішніми сервісами потребує додаткової налаштування.

Canvas є іншим популярним комерційним рішенням, що використовується в багатьох університетах США та Європи. Canvas забезпечує сучасний інтерфейс, широкий функціонал для управління навчанням, інтеграцію з різними зовнішніми сервісами та мобільний додаток для доступу з смартфонів. Система має вбудовані інструменти аналітики успішності студентів та забезпечує високу продуктивність. Проте Canvas є платним рішенням з високою вартістю ліцензії, що робить його недоступним для багатьох навчальних закладів, особливо в Україні. Крім того, система не має вбудованого модуля управління розкладом, що вимагає використання додаткових рішень.

Blackboard є ще одним комерційним рішенням, що використовується в університетах по всьому світу. Ця система забезпечує широкий функціонал для управління навчанням, включаючи управління курсами, систему завдань, форуми, чати та інструменти для співпраці. Система має розширений функціонал для аналітики успішності студентів та підтримує інтеграцію з різними зовнішніми сервісами. Проте Blackboard має дуже високу вартість ліцензії, складний інтерфейс, що потребує тривалої

адаптації, а система має високі вимоги до апаратного забезпечення для забезпечення належної продуктивності.

Google Classroom є безкоштовним рішенням від Google, що використовується в багатьох навчальних закладах для організації дистанційного навчання. Система забезпечує створення класів, розсилку завдань, перевірку робіт та спілкування між викладачами та студентами. Google Classroom має простий інтерфейс, інтегрується з Google Workspace (раніше G Suite) та забезпечує безкоштовний доступ до основних функцій. Проте Google Classroom має обмежений функціонал, відсутній модуль управління розкладом, відсутній облік відвідуваності, а система не призначена для комплексного управління навчальним процесом в університеті, а орієнтована на шкільну освіту.

Microsoft Teams for Education є безкоштовним рішенням від Microsoft, що забезпечує інструменти для організації дистанційного навчання, включаючи відеоконференції, чати, спільну роботу над документами та управління навчальними матеріалами. Система інтегрується з Microsoft 365 та забезпечує широкі можливості для співпраці. Проте Microsoft Teams не є спеціалізованою системою для управління навчальним процесом, відсутній модуль управління розкладом, відсутній електронний журнал та облік відвідуваності, а система має високі вимоги до апаратного забезпечення.

1.2.3. Відкриті (Open-Source) системи управління навчанням.

Серед відкритих рішень варто відзначити Open edX — платформу для масових відкритих курсів (MOOC), що використовується університетами для створення онлайн-курсів. Система забезпечує розширені можливості для створення інтерактивних курсів, включаючи відеолекції, тестування, форуми та інструменти аналітики. Проте Open edX не призначена для управління розкладом, відсутній модуль обліку відвідуваності, а система має складну архітектуру, що вимагає значних ресурсів для розгортання та обслуговування.

Chamilo є відкритою системою управління навчанням, що розроблена на PHP та використовується в навчальних закладах по всьому світу. Система забезпечує широкий функціонал для управління курсами, систему завдань, форуми та інструменти для спілкування. Chamilo має простий інтерфейс та вимагає менше ресурсів порівняно з іншими системами. Проте система має обмежені можливості інтеграції з зовнішніми

сервісами, відсутній модуль управління розкладом, а система не забезпечує мобільний доступ через спеціалізовані додатки.

ILIAS є відкритою системою управління навчанням, що використовується в університетах Європи. Система забезпечує розширений функціонал для управління навчанням, включаючи управління курсами, систему завдань, форуми, інструменти для співпраці та аналітики успішності. ILIAS має модульну архітектуру та підтримує налаштування під специфіку конкретного навчального закладу. Проте система має складний інтерфейс, що потребує тривалої адаптації, відсутній модуль управління розкладом, а система має високі вимоги до апаратного забезпечення.

1.2.4. Порівняльний аналіз та виявлені недоліки існуючих рішень

Порівняльний аналіз існуючих рішень показує, що більшість систем мають як переваги, так і недоліки. Комерційні системи (Canvas, Blackboard) забезпечують широкий функціонал та сучасний інтерфейс, але мають високу вартість ліцензії, що робить їх недоступними для багатьох навчальних закладів. Безкоштовні рішення (Google Classroom, Microsoft Teams) мають обмежений функціонал та не призначені для комплексного управління навчальним процесом. Відкриті рішення (Moodle, Open edX, Chamilo, ILIAS) забезпечують доступність, але часто мають складний інтерфейс, обмежені можливості інтеграції та високі вимоги до апаратного забезпечення.

Спільною проблемою для більшості існуючих рішень є відсутність вбудованого модуля управління розкладом, що є критично важливим компонентом для управління навчальним процесом в університетах. Більшість систем орієнтовані на управління навчанням (LMS), але не забезпечують комплексне управління розкладом, відвідуваністю та іншими аспектами навчального процесу. Крім того, більшість систем не забезпечують ефективний мобільний доступ через спеціалізовані додатки або інтеграцію з популярними месенджерами, такими як Telegram.

Іншою проблемою є обмежені можливості інтеграції з зовнішніми сервісами, зокрема з хмарними сервісами для зберігання навчальних матеріалів, такими як Google Drive. Більшість систем мають власні системи зберігання матеріалів, що вимагає додаткового дискового простору та не забезпечує гнучкості у виборі сервісів зберігання.

Таким чином, аналіз існуючих аналогів показує, що на ринку відсутні рішення, що забезпечують комплексний підхід до управління навчальним процесом з включенням управління розкладом, електронного журналу,

обліку відвідуваності, управління навчальними матеріалами та забезпечення мобільного доступу через Telegram бота. Це обумовлює необхідність розробки нової інформаційної системи, що враховує недоліки існуючих рішень та забезпечує розширений функціонал для комплексного управління навчальним процесом.

1.3. Постановка задачі

Постановка задачі дослідження є критично важливим етапом, що визначає напрямок розробки інформаційної системи, критерії успіху та очікувані результати. На основі проведеного аналізу предметного середовища та огляду існуючих аналогів сформульовано мету та завдання дослідження, що спрямовані на вирішення виявлених проблем та створення ефективної інформаційної системи управління навчальним процесом.

1.3.1 Мета дослідження

Метою дослідження є створення програмного продукту у вигляді інформаційної системи управління розкладом та університетською інформацією для закладів освіти, яка забезпечує централізоване управління навчальним процесом, автоматизацію обліку оцінок та відвідуваності, а також забезпечення доступу до інформації через веб-інтерфейс та Telegram бота. Розроблювана система має вирішити проблему розрізненості даних про навчальний процес, забезпечити єдине інформаційне простір для студентів, викладачів та адміністрації, а також автоматизувати процеси обліку відвідуваності, оцінювання та надання навчальних матеріалів.

Для досягнення поставленої мети вирішуються наступні завдання дослідження, що поділяються на теоретичні, практичні та експериментальні.

1.3.2 Теоретичні завдання

Теоретичні завдання включають дослідження сучасних підходів до управління навчальним процесом, аналіз міжнародних стандартів якості програмного забезпечення та розробку математичних моделей для оцінки якості системи. Дослідження сучасних підходів до управління навчальним процесом включає вивчення методологій Agile та DevOps, які інтегрують всі етапи розробки та забезпечують швидку адаптацію до змін вимог. Ці методології передбачають безперервну інтеграцію, безперервне тестування та автоматизацію процесів розробки, що є критично важливим для створення сучасних інформаційних систем.

Аналіз міжнародних стандартів якості програмного забезпечення включає вивчення стандартів ISO/IEC 25010, що визначають модель якості програмного продукту, включаючи такі характеристики як функціональна придатність, продуктивність, сумісність, зручність використання, надійність, безпека, супроводжуваність та переносимість. Ці стандарти служать основою для розробки стратегії тестування та забезпечення якості розроблюваної системи. Особливу увагу приділено стандартам доступності WCAG 2.1, що визначають вимоги до забезпечення доступності веб-інтерфейсів для людей з обмеженими можливостями.

Розробка математичних моделей для оцінки якості системи включає створення моделей для оцінки продуктивності системи під навантаженням, моделей для оцінки ефективності використання ресурсів та моделей для оцінки зручності використання інтерфейсу. Ці моделі дозволяють кількісно оцінити якість системи та порівняти її з аналогами. Особливу увагу приділено моделям для оцінки масштабованості системи, що дозволяють визначити максимальну кількість одночасних користувачів, які можуть працювати з системою без втрати продуктивності.

1.3.3. Практичні завдання

Практичні завдання включають розробку архітектури трьохшарової системи, створення бази даних з оптимізованою структурою та впровадження методів автоматизованого тестування. Розробка архітектури трьохшарової системи включає проектування серверної частини на базі REST API, що забезпечує обробку бізнес-логіки та роботу з даними; веб-інтерфейсу на Vue.js для адміністративного управління та доступу студентів; та Telegram бота на aiogram для забезпечення мобільного доступу до інформації. Архітектура системи повинна забезпечувати модульність, масштабованість та легкість підтримки.

Серверна частина на базі REST API розробляється з використанням фреймворку FastAPI, що забезпечує високу продуктивність, автоматичну документацію API через Swagger та підтримку асинхронного програмування. API забезпечує наступні основні функції: автентифікація та авторизація користувачів, управління розкладом занять, робота з оцінками та відвідуваністю, система завдань, новини та події, волонтерство. Використання REST архітектури забезпечує незалежність клієнта від сервера та дозволяє розширювати функціонал без зміни клієнтської частини.

Веб-інтерфейс на Vue.js розробляється з використанням композиційного API та забезпечує наступні функції: перегляд розкладу з фільтрацією за групами, викладачами, аудиторіями; перегляд оцінок та середнього балу; перегляд відвідуваності; доступ до навчальних матеріалів; перегляд новин та подій; участь у волонтерській діяльності. Інтерфейс має бути адаптивним для роботи з різних пристроїв та відповідати стандартам доступності WCAG 2.1 Level AA.

Telegram бот на aiogram розробляється для забезпечення мобільного доступу до функцій системи та включає наступні функції: команди для перегляду розкладу (/schedule, /today, /tomorrow), отримання сповіщень про зміни в розкладі, перегляд оцінок та середнього балу, інформація про новини та події, меню з callback-кнопками для зручної навігації. Бот забезпечує асинхронну обробку повідомлень та інтеграцію з серверною частиною через API.

Створення бази даних з оптимізованою структурою включає проектування таблиць для зберігання даних про користувачів, групи, викладачів, предмети, розклад, оцінки, відвідуваність, завдання, новини, події та волонтерську діяльність. База даних розробляється на PostgreSQL з використанням ORM SQLAlchemy для забезпечення об'єктно-реляційного відображення. Особливу увагу приділено оптимізації запитів, створенню індексів для прискорення пошуку та забезпеченню цілісності даних через обмеження та тригери.

Впровадження методів автоматизованого тестування включає розробку unit-тестів для перевірки окремих компонентів системи, integration-тестів для перевірки взаємодії між компонентами, e2e-тестів для перевірки системи в цілому та тестів продуктивності для оцінки продуктивності системи під навантаженням. Для автоматизації тестування використовуються інструменти pytest для unit- та integration-тестів, Playwright для e2e-тестів веб-інтерфейсу, K6 для тестування продуктивності API та OWASP ZAP для тестування безпеки.

1.3.4. Експериментальні завдання

Експериментальні завдання включають тестування продуктивності системи під навантаженням, аналіз usability та доступності інтерфейсу та оцінку впливу системи на ефективність навчального процесу. Тестування продуктивності системи під навантаженням включає визначення максимальної кількості одночасних користувачів, які можуть працювати з системою без втрати продуктивності, вимірювання часу відповіді API для

різних типів запитів та оцінку використання ресурсів сервера (CPU, RAM, дискова підсистема). Результати тестування дозволять визначити необхідність масштабування системи та оптимізації запитів до бази даних.

Аналіз usability та доступності інтерфейсу включає проведення тестування з реальними користувачами (студентами, викладачами, адміністраторами) для виявлення проблем у використанні інтерфейсу, аналіз відповідності інтерфейсу стандартам доступності WCAG 2.1 Level AA та оцінку зручності використання інтерфейсу на різних пристроях (комп'ютери, планшети, смартфони). Результати аналізу дозволять внести зміни в інтерфейс для покращення зручності використання та забезпечення доступності для всіх категорій користувачів.

Оцінка впливу системи на ефективність навчального процесу включає аналіз зменшення часу на виконання рутинних операцій (наприклад, розробка розкладу, ведення електронного журналу), оцінку зменшення кількості помилок при обробці даних, аналіз поліпшення комунікації між учасниками освітнього процесу та оцінку задоволеності користувачів системою. Для оцінки впливу проводиться опитування користувачів до та після впровадження системи та аналіз статистики використання системи.

1.3.5. Критерії успіху та очікувані результати

Критеріями успіху дослідження є: створення функціонуючої інформаційної системи з усіма запланованими функціями; забезпечення продуктивності системи (час відповіді API — не більше 200 мс для 95% запитів, підтримка до 1000 одночасних користувачів); забезпечення безпеки системи (захист від SQL-ін'єкцій та XSS-атак, двостороння автентифікація через JWT токени); забезпечення доступності (відповідність стандартам WCAG 2.1 Level AA, підтримка навігації клавіатурою); отримання позитивної оцінки від користувачів системи (задоволеність не менше 80% опитаних користувачів).

Очікувані результати дослідження включають створення інформаційної системи управління навчальним процесом з розширеним функціоналом, що включає управління розкладом, електронний журнал, облік відвідуваності, систему завдань, новини та події, волонтерство; забезпечення мобільного доступу через Telegram бота; інтеграцію з Google Drive для зберігання навчальних матеріалів; розробку методів автоматизованого тестування; отримання результатів тестування

продуктивності, доступності та впливу на ефективність навчального процесу.

Таким чином, постановка задачі дослідження сформульована з урахуванням сучасних вимог до якості програмного забезпечення, міжнародних стандартів та потреб користувачів. Розроблена система має вирішити проблему розрізненості даних про навчальний процес та забезпечити єдине інформаційне простір для всіх учасників освітнього процесу, що підвищить ефективність управління навчальним процесом в закладах освіти.

ВИСНОВКИ ДО РОЗДІЛУ

У результаті виконання першого розділу дипломної роботи проведено детальний аналіз предметного середовища управління навчальним процесом в сучасних навчальних закладах. Встановлено, що в умовах цифрової трансформації освіти управління навчальним процесом вимагає створення комплексних інформаційних систем, що забезпечують централізоване управління всіма аспектами навчальної діяльності, автоматизацію рутинних операцій та забезпечення мобільного доступу до інформації для всіх учасників освітнього процесу.

Аналіз нормативно-правової бази показав, що розробка інформаційних систем управління навчальним процесом повинна відповідати міжнародним стандартам якості програмного забезпечення, зокрема ISO/IEC 25010, що визначає модель якості програмного продукту з восьмома основними характеристиками: функціональна придатність, продуктивність, сумісність, зручність використання, надійність, безпека, супроводжуваність та переносимість. В Україні ці вимоги відображені в національних стандартах ДСТУ, зокрема ДСТУ ISO/IEC 25010:2015, що гармонізований з міжнародним стандартом.

Класифікація інформаційних систем управління навчальним процесом показала, що існуючі рішення можна поділити за функціональним призначенням (системи управління розкладом, електронні журнали, системи обліку відвідуваності, платформи електронного навчання), масштабом впровадження (локальні, галузеві, національні системи) та архітектурним підходом (монолітні системи, мікросервісна архітектура, хмарні рішення). Кожен тип систем має свої особливості та вимоги до реалізації.

Огляд існуючих аналогів показав, що на ринку представлено значну кількість як комерційних, так і відкритих рішень для управління навчальним процесом. Комерційні системи (Canvas, Blackboard) забезпечують широкий функціонал та сучасний інтерфейс, але мають високу вартість ліцензії, що робить їх недоступними для багатьох навчальних закладів. Безкоштовні рішення (Google Classroom, Microsoft Teams) мають обмежений функціонал та не призначені для комплексного управління навчальним процесом. Відкриті рішення (Moodle, Open edX, Chamilo, ILIAS) забезпечують доступність, але часто мають складний інтерфейс, обмежені можливості інтеграції та високі вимоги до апаратного забезпечення.

Спільною проблемою для більшості існуючих рішень є відсутність вбудованого модуля управління розкладом, що є критично важливим компонентом для управління навчальним процесом в університетах. Більшість систем орієнтовані на управління навчанням (LMS), але не забезпечують комплексне управління розкладом, відвідуваністю та іншими аспектами навчального процесу. Крім того, більшість систем не забезпечують ефективний мобільний доступ через спеціалізовані додатки або інтеграцію з популярними месенджерами, такими як Telegram, що є важливим аспектом в умовах сучасної освіти.

Аналіз вітчизняних рішень показав, що більшість з них мають обмежений функціонал, недостатню інтеграцію між модулями, обмежені можливості масштабування та відсутність сучасних механізмів забезпечення мобільного доступу до інформації. Більшість систем орієнтовані на вирішення окремих задач і не забезпечують комплексний підхід до управління навчальним процесом, що вимагається в сучасних умовах цифрової трансформації освіти.

На основі проведеного аналізу сформульовано мету та завдання дослідження. Метою дослідження є створення програмного продукту у вигляді інформаційної системи управління розкладом та університетською інформацією для закладів освіти, яка забезпечує централізоване управління навчальним процесом, автоматизацію обліку оцінок та відвідуваності, а також забезпечення доступу до інформації через веб-інтерфейс та Telegram бота.

Для досягнення поставленої мети вирішуються теоретичні, практичні та експериментальні завдання. Теоретичні завдання включають дослідження сучасних підходів до управління навчальним процесом, аналіз

міжнародних стандартів якості програмного забезпечення та розробку математичних моделей для оцінки якості системи. Практичні завдання включають розробку архітектури трьохшарової системи (REST API, веб-інтерфейс, Telegram бот), створення бази даних з оптимізованою структурою та впровадження методів автоматизованого тестування. Експериментальні завдання включають тестування продуктивності системи під навантаженням, аналіз usability та доступності інтерфейсу та оцінку впливу системи на ефективність навчального процесу.

Критеріями успіху дослідження є: створення функціонуючої інформаційної системи з усіма запланованими функціями; забезпечення продуктивності системи (час відповіді API — не більше 200 мс для 95% запитів, підтримка до 1000 одночасних користувачів); забезпечення безпеки системи (захист від SQL-ін'єкцій та XSS-атак, двостороння автентифікація через JWT токени); забезпечення доступності (відповідність стандартам WCAG 2.1 Level AA, підтримка навігації клавіатурою); отримання позитивної оцінки від користувачів системи (задоволеність не менше 80% опитаних користувачів).

Таким чином, перший розділ дипломної роботи дозволив сформулювати мету та завдання дослідження, визначити критерії успіху та очікувані результати. Розроблена інформаційна система має вирішити проблему розрізненості даних про навчальний процес та забезпечити єдине інформаційне простір для всіх учасників освітнього процесу, що підвищить ефективність управління навчальним процесом в закладах освіти.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Аналіз предметної області (вхідні та вихідні дані)

Аналіз предметної області є критично важливим етапом проектування інформаційної системи, що дозволяє виявити основні об'єкти дослідження, визначити їх взаємозв'язки, побудувати інформаційну модель та описати існуючі обмеження на вхідні та вихідні дані. Предметна область управління навчальним процесом включає широкий спектр об'єктів, процесів та відносин, що мають бути адекватно відображені в інформаційній моделі системи.

2.1.1. Основні об'єкти предметної області

Виділення основних об'єктів дослідження є першим кроком у побудові інформаційної моделі. До основних об'єктів предметної області належать користувачі системи, навчальні групи, викладачі, навчальні дисципліни, аудиторний фонд, розклад занять, оцінки студентів, відвідуваність, завдання для студентів, навчальні матеріали, новини та події, а також волонтерська діяльність. Кожен об'єкт має свої атрибути та взаємозв'язки з іншими об'єктами, що формують структуру предметної області.

Користувачі системи є центральним об'єктом предметної області, оскільки всі інші об'єкти орієнтовані на забезпечення їх потреб. Користувачі поділяються на ролі відповідно до їхніх функцій в навчальному процесі: адміністратори, викладачі, студенти та батьки студентів. Адміністратори мають повний доступ до всіх функцій системи, викладачі мають доступ до управління розкладом, ведення електронного журналу, контролю відвідуваності та управління навчальними матеріалами, студенти мають доступ до перегляду розкладу, оцінок, відвідуваності, навчальних матеріалів та участі у волонтерській діяльності, а батьки мають обмежений доступ до перегляду оцінок та відвідуваності своїх дітей.

Навчальні групи є об'єктом, що об'єднує студентів за спеціальністю, курсом та semester. Кожна група має назву, спеціальність, курс, semester та списку студентів. Групи пов'язані з навчальними дисциплінами через навчальний план, що визначає які дисципліни вивчає група, а також з розкладом, що визначає коли та де відбуваються заняття для групи. Групи також пов'язані з викладачами через розклад, що визначає які викладачі ведуть заняття для групи.

Викладачі є об'єктом, що відповідає за проведення занять, виставлення оцінок та контроль відвідуваності. Кожен викладач має прізвище, ім'я, по батькові, email, телефон, посаду та кафедру. Викладачі пов'язані з навчальними дисциплінами через розклад, що визначає які дисципліни викладає викладач, а також з групами через розклад, що визначає для яких груп проводить заняття викладач. Викладачі також пов'язані з аудиторіями через розклад, що визначає в яких аудиторіях проводить заняття викладач.

Навчальні дисципліни є об'єктом, що відображає зміст навчального процесу. Кожна дисципліна має назву, код, опис, кількість кредитів та тип (лекція, практичне заняття, лабораторна робота). Дисципліни пов'язані з групами через навчальний план, що визначає які дисципліни вивчає група, а також з викладачами через розклад, що визначає які викладачі ведуть дисципліну.

Аудиторія не є окремим об'єктом бази даних у розробленій системі. Замість ведення довідника аудиторного фонду, номер або назва аудиторії вказується менеджером розкладу безпосередньо при формуванні запису розкладу у вигляді довільного текстового значення. Така архітектурна рішення спрощує управління розкладом та забезпечує гнучкість при зазначенні місця проведення заняття без прив'язки до фіксованого переліку приміщень.

Розклад занять є центральним об'єктом предметної області, що визначає коли, де та які заняття відбуваються. Кожен запис розкладу має дату, час початку, час закінчення, групу, викладача, дисципліну, аудиторію та тип заняття. Розклад пов'язаний з усіма іншими об'єктами предметної області та забезпечує їх взаємодію.

2.1.2. Артефакти навчального процесу

Оцінки студентів є об'єктом, що відображає успішність студентів. Кожна оцінка має студента, дисципліну, тип оцінки (модульна, екзаменаційна, поточна), значення оцінки, дату виставлення та викладача, що виставив оцінку. Оцінки пов'язані зі студентами, дисциплінами та викладачами.

Відвідуваність є об'єктом, що відображає присутність студентів на заняттях. Кожен запис відвідуваності має студента, заняття (запис розкладу), статус присутності (присутній, відсутній, поважна причина) та дату. Відвідуваність пов'язана зі студентами та розкладом.

Завдання для студентів є об'єктом, що відображає завдання, що викладачі ставлять перед студентами. Кожне завдання має назву, опис, дату видачі,

дату здачі, дисципліну, групу, викладача, статус (активне, завершене, прострочене) та список завантажених файлів. Завдання пов'язані з дисциплінами, групами, викладачами та навчальними матеріалами.

Навчальні матеріали є об'єктом, що відображає файли з навчальним змістом. Кожен матеріал має назву, опис, тип файлу, посилання на файл (в Google Drive), дату завантаження, автора (викладача) та дисципліну. Матеріали пов'язані з дисциплінами, викладачами та завданнями.

Новини та події є об'єктом, що відображає інформаційні повідомлення для користувачів. Кожна новина має назву, опис, дату публікації, автора, тип (новина, подія, оголошення) та статус (активна, архівна). Новини пов'язані з авторами та можуть бути адресовані певним групам користувачів.

Волонтерська діяльність є об'єктом, що відображає волонтерські заходи та участь студентів у них. Кожна волонтерська подія має назву, опис, дату проведення, місце, організатора, статус (активна, завершена) та список учасників. Волонтерська діяльність пов'язана зі студентами та організаторами.

2.1.3. Інформаційна модель та взаємозв'язки об'єктів

Побудова інформаційної моделі включає визначення атрибутів кожного об'єкта, встановлення взаємозв'язків між об'єктами та визначення обмежень цілісності даних. Інформаційна модель відображає структуру предметної області та служить основою для проектування бази даних. Основні типи взаємозв'язків між об'єктами включають: "один-до-багатьох" (наприклад, один викладач веде багато занять), "багато-до-багатьох" (наприклад, студенти відвідують багато занять, а заняття відвідуються багатьма студентами) та "один-до-одного" (наприклад, кожен студент має один профіль користувача).

2.1.4. Структура вхідних та вихідних даних

Опис існуючих обмежень на вхідні та вихідні дані є важливим аспектом аналізу предметної області. Вхідні дані системи включають інформацію, що вводиться користувачами або отримується від зовнішніх систем. До вхідних даних належать: дані про користувачів (прізвище, ім'я, email, роль), дані про групи (назва, спеціальність, курс), дані про викладачів (прізвище, ім'я, email, кафедра), дані про дисципліни (назва, код, кількість кредитів), дані про розклад (дата, час, аудиторія), дані про оцінки (значення оцінки, дата виставлення), дані про відвідуваність (статус

присутності), дані про завдання (назва, опис, дати) та дані про навчальні матеріали (файли).

Вихідні дані системи включають інформацію, що генерується системою на основі обробки вхідних даних та запитів користувачів. До вихідних даних належать: розклад занять для певної групи, викладача або аудиторії; оцінки студента з певної дисципліни; статистика відвідуваності; список завдань для студента; список навчальних матеріалів для дисципліни; список новин та подій; статистика волонтерської діяльності; звіти про успішність студентів; звіти про відвідуваність.

2.1.5. Обмеження цілісності та валідація даних

Обмеження на вхідні дані включають: валідація email адрес (повинен відповідати формату email); валідація пароля (мінімум 8 символів, включати великі та малі літери, цифри); валідація дат (дата початку не може бути пізніше дати закінчення); валідація оцінок (оцінка повинна бути в діапазоні від 0 до 100 або від 2 до 5 залежно від шкали оцінювання); валідація статусів присутності (тільки дозволені значення: присутній, відсутній, поважна причина); валідація типів файлів (тільки дозволені формати: PDF, DOC, DOCX, PPT, PPTX); валідація розміру файлів (максимальний розмір 50 МБ).

Обмеження на вихідні дані включають: обмеження кількості записів у результатах пошуку (максимум 100 записів на сторінку); обмеження доступу до даних (користувачі мають доступ лише до даних, що стосуються їхньої ролі); обмеження на час формування звітів (звіти повинні формуватися не більше 30 секунд); обмеження на формат дати та часу (використання формату ISO 8601); обмеження на формат чисел (використання крапки як десяткового роздільника).

2.1.6. Вимоги щодо безпеки даних

Особливу увагу приділено обмеженням безпеки даних. До таких обмежень належать: шифрування паролів користувачів (використання bcrypt з фактором роботи не менше 12); захист від SQL-ін'єкцій (використання параметризованих запитів); захист від XSS-атак (екранування вхідних даних); захист від CSRF-атак (використання токенів); обмеження кількості спроб входу (максимум 5 спроб за 15 хвилин); обмеження часу сесії (автоматичне завершення сесії через 30 хвилин бездіяльності); логування всіх операцій з даними (збір інформації про користувача, дату, час та тип операції).

Таким чином, аналіз предметної області дозволив виділити основні об'єкти дослідження, побудувати інформаційну модель та описати існуючі обмеження на вхідні та вихідні дані. Ця інформація є основою для проектування системи та математичного та алгоритмічного забезпечення.

2.2. Проектування системи (проектування бази даних)

Проектування системи є наступним етапом після аналізу предметної області і включає концептуальне проектування системи, побудову концептуальної моделі та проектування бази даних. Концептуальне проектування системи дозволяє визначити архітектуру системи, розподіл функціональності між компонентами та принципи взаємодії між ними. Концептуальна модель дає змогу виявити причинно-наслідкові зв'язки, властивості об'єктів та їх взаємодію в межах визначених цілями дослідження.

2.2.1 Концептуальне проектування та архітектура системи

Концептуальне проектування системи базується на архітектурі клієнт-сервер з трьома основними компонентами: серверна частина (REST API), веб-інтерфейс (frontend) та Telegram бот. Серверна частина відповідає за обробку бізнес-логіки, роботу з базою даних, валідацію даних та забезпечення безпеки. Веб-інтерфейс забезпечує взаємодію користувачів з системою через веб-браузер, відображення даних та збір вхідних даних. Telegram бот забезпечує мобільний доступ до функцій системи через месенджер, відправку сповіщень та виконання команд користувачів.

Архітектура системи базується на принципах розподілу відповідальності, модульності та масштабованості. Серверна частина реалізується як REST API на базі фреймворку FastAPI, що забезпечує незалежність клієнта від сервера та дозволяє використовувати різні клієнтські додатки. Використання REST архітектури забезпечує стандартизований спосіб взаємодії між компонентами через HTTP протокол з використанням методів GET, POST, PUT, DELETE для виконання операцій CRUD.

2.2.2. Клієнтські додатки: веб-інтерфейс та Telegram-бот

Веб-інтерфейс розробляється на базі фреймворку Vue.js 3 з композиційним API та використанням Vue Router для навігації між різними сторінками додатка. Система має 11 основних сторінок (views), що забезпечують різні функції: перегляд розкладу, профіль користувача, сторінка завдань, сторінка новин, сторінка волонтерства, адміністративні

панелі та інші. Кожна сторінка відповідає за певний функціонал і взаємодіє з серверною частиною через REST API.

Веб-інтерфейс забезпечує адаптивний дизайн для роботи з різних пристроїв (комп'ютери, планшети, смартфони), плавну навігацію між сторінками без повної перезавантаження сторінки завдяки Vue Router, а також відповідає стандартам доступності WCAG 2.1 Level AA. Інтерфейс взаємодіє з серверною частиною через HTTP запити з використанням JSON формату для передачі даних.

Додаток використовує Vue Router для маршрутизації між різними сторінками, що дозволяє зберігати стан при переходах між сторінками та забезпечує швидке перемикання між розділами без завантаження нових HTML сторінок. Кожна сторінка (view) відповідає за певний функціонал і може включати власні компоненти для відображення специфічних елементів інтерфейсу.

Telegram бот розробляється на базі бібліотеки aiogram, що забезпечує асинхронну обробку повідомлень, підтримку webhook та довгого опитування, роботу з callback-кнопками та інлайн-клавіатурами. Бот взаємодіє з серверною частиною через REST API для отримання даних та виконання операцій від імені користувача.

2.2.3. Концептуальна модель даних (сутнісно-зв'язкова модель)

Побудова концептуальної моделі включає створення ER-діаграми, що відображає сутності предметної області та їх взаємозв'язки. Основні сутності концептуальної моделі включають: User (користувач), Group (група), Teacher (викладач), Subject (дисципліна), Schedule (розклад), Grade (оцінка), Attendance (відвідуваність), Assignment (завдання), Submission (здача завдання), AssignmentFile (файли завдань), Material (навчальний матеріал), News (новина), NewsPhoto (фотографії новин), Event (подія), Volunteering (волонтерська діяльність), Department (кафедра), Specialty (спеціальність), Class_Type (тип заняття), Class_Number (номер пари), SubjectConfig (конфігурація предмету), NewsCategory (категорія новин), VolunteeringCategory (категорія волонтерства).

Сутність User (користувач) має атрибути: id, name (ім'я), email (електронна пошта), telegram_id (ідентифікатор Telegram), password (пароль), specialty_id (посилання на спеціальність), role_id (посилання на роль), group_id (посилання на групу), google_access_token,

google_refresh_token. Сутність User має взаємозв'язки з Grade, Attendance, Assignment, Submission, Volunteering, News.

Сутність Group (група) має атрибути: id, name (назва), specialty_id (посилання на спеціальність). Сутність Group пов'язана з Schedule, User, SubjectConfig.

Сутність Teacher (викладач) має атрибути: id, name (ім'я), email, department_id (посилання на кафедру), connectionCode. Сутність Teacher пов'язана з Schedule та Department.

Сутність Subject (дисципліна) має атрибути: id, name (назва), desc (опис). Сутність Subject пов'язана з Schedule та SubjectConfig.

Сутність Schedule (розклад) має атрибути: id, date (дата), auditory (аудиторія — текстове поле), connectionCode, lesson_number (номер пари), topic (тема), subject_id, class_type_id, teacher_id, group_id, classnumber_id. Аудиторії зберігаються як текст у полі auditory, без окремої таблиці аудиторій. Сутність Schedule має взаємозв'язки з Group, Teacher, Subject, Class_Type, Class_Number, SubjectConfig, Assignment.

Сутність Class_Type (тип заняття) має атрибути: id, name (назва: лекція, практичне, лабораторна). Пов'язана з Schedule та SubjectConfig.

Сутність Class_Number (номер пари) має атрибути: id, number (номер), time_start (час початку), time_end (час закінчення). Пов'язана з Schedule.

Сутність Department (кафедра) має атрибути: id, name (назва). Пов'язана з Specialty, Teacher, News, Volunteering, Event.

Сутність Specialty (спеціальність) має атрибути: id, department_id, specialty_number, name (назва). Пов'язана з Group, User, Department .

Сутність SubjectConfig (конфігурація предмету) має атрибути: id, group_id, subject_id, class_type_id, total_lessons (загальна кількість занять), current_lesson (поточне заняття), spreadsheet_id (ідентифікатор Google таблиці). Ця сутність зв'язує групу, предмет та тип заняття і містить конфігурацію для ведення журналу. Пов'язана з Group, Subject, Class_Type, Grade, Attendance, Schedule.

Сутність Grade (оцінка) має атрибути: id, subject_config_id, lesson_number, student_id, value (значення оцінки). Пов'язана з SubjectConfig та User. .

Сутність Attendance (відвідуваність) має атрибути: id, subject_config_id, lesson_number, student_id, is_present (присутність). Пов'язана з SubjectConfig та User.

Сутність Assignment (завдання) має атрибути: id, title, description, description_file, description_type, deadline, max_score, submission_type, schedule_id, teacher_id, created_at, updated_at. Пов'язана з Schedule, User, Submission .

Сутність Submission (здача завдання) має атрибути: id, assignment_id, student_id, text_content, file_url, google_doc_id, status (draft, submitted, graded), submitted_at, is_late. Пов'язана з Assignment, User, AssignmentFile .

Сутність AssignmentFile (файли завдань) має атрибути: id, submission_id, filename, file_path, file_size, mime_type. Пов'язана з Submission.

Сутність News (новина) має атрибути: id, name, description, department_id, user_id, newscategory_id, photo_path, created_at. Пов'язана з User, Department, NewsCategory, NewsPhoto.

Сутність NewsPhoto (фотографії новин) має атрибути: id, image_path, news_id. Пов'язана з News.

Сутність NewsCategory (категорія новин) має атрибути: id, name. Пов'язана з News.

Сутність Event (подія) має атрибути: id, department_id, date, created_at. Пов'язана з Department.

Сутність Volunteering (волонтерська діяльність) має атрибути: id, name, date_start, date_end, location, department_id, user_id, goal, desc, volunteeringcategory_id. Пов'язана з User, Department, VolunteeringCategory.

Сутність VolunteeringCategory (категорія волонтерства) має атрибути: id, name. Пов'язана з Volunteering.

Сутність Role (роль) має атрибути: id, name. Пов'язана з User.

2.2.4. Фізичне проектування бази даних (схема таблиць)

Проектування бази даних включає перетворення концептуальної моделі в реляційну модель з визначенням таблиць, стовпців, типів даних, первинних ключів, зовнішніх ключів та обмежень цілісності. База даних розробляється на системі управління базами даних PostgreSQL з використанням ORM SQLAlchemy.

Основні таблиці бази даних: users, groups, teachers, subject, schedules, class_type, class_numbers, departments, specialties, roles, subject_config, attendance, grades, assignments, submissions, assignment_files, news, news_photos, newscategory, events, volunteering, volunteeringcategory.

Таблиця users має стовпці: id (INTEGER PRIMARY KEY), name (STRING), email (STRING UNIQUE), telegram_id (STRING UNIQUE), password (STRING), specialty_id (INTEGER REFERENCES specialties), role_id (INTEGER REFERENCES roles), group_id (INTEGER REFERENCES groups), google_access_token, google_refresh_token.
Таблиця groups має стовпці: id (INTEGER PRIMARY KEY), name (STRING), specialty_id (INTEGER REFERENCES specialties).
Таблиця teachers має стовпці: id (INTEGER PRIMARY KEY), name (STRING), email (STRING), department_id (INTEGER REFERENCES departments), connectionCode.
Таблиця subject має стовпці: id (INTEGER PRIMARY KEY), name (STRING), desc (STRING).
Таблиця schedules має стовпці: id (INTEGER PRIMARY KEY), date (DATE), auditory (STRING), connectionCode, lesson_number (INTEGER), topic (STRING), subject_id (INTEGER REFERENCES subject), class_type_id (INTEGER REFERENCES class_type), teacher_id (INTEGER REFERENCES teachers), group_id (INTEGER REFERENCES groups), classnumber_id (INTEGER REFERENCES class_numbers). Має індекси на group_id+date та date.
Таблиця class_type має стовпці: id (INTEGER PRIMARY KEY), name (STRING).
Таблиця class_numbers має стовпці: id (INTEGER PRIMARY KEY), number (INTEGER), time_start (TIME), time_end (TIME).
Таблиця departments має стовпці: id (INTEGER PRIMARY KEY), name (STRING).
Таблиця specialties має стовпці: id (INTEGER PRIMARY KEY), department_id (INTEGER REFERENCES departments), specialty_number (INTEGER), name (STRING).
Таблиця roles має стовпці: id (INTEGER PRIMARY KEY), name (STRING).
Таблиця subject_config має стовпці: id (INTEGER PRIMARY KEY), group_id (INTEGER REFERENCES groups), subject_id (INTEGER REFERENCES subject), class_type_id (INTEGER REFERENCES class_type), total_lessons (INTEGER), current_lessons (INTEGER), spreadsheet_id (STRING). Має унікальний індекс на group_id+subject_id+class_type_id.

Таблиця attendance має стовпці: id (INTEGER PRIMARY KEY), subject_config_id (INTEGER REFERENCES subject_config), lesson_number (INTEGER), student_id (INTEGER REFERENCES users), is_present (BOOLEAN). Має індекси на subject_config_id, lesson_number, student_id.
Таблиця grades має стовпці: id (INTEGER PRIMARY KEY), subject_config_id (INTEGER REFERENCES subject_config), lesson_number (INTEGER), student_id (INTEGER REFERENCES users), value (STRING). Має індекси на subject_config_id, lesson_number, student_id.
Таблиця assignments має стовпці: id (INTEGER PRIMARY KEY), title (STRING), description (TEXT), description_file (STRING), description_type (STRING), deadline (DATETIME), max_score (INTEGER), submission_type (STRING), schedule_id (INTEGER REFERENCES schedules), teacher_id (INTEGER REFERENCES users), created_at (DATETIME), updated_at (DATETIME). Має індекси на schedule_id, teacher_id, deadline.
Таблиця submissions має стовпці: id (INTEGER PRIMARY KEY), assignment_id (INTEGER REFERENCES assignments), student_id (INTEGER REFERENCES users), text_content (TEXT), file_url (STRING), google_doc_id (STRING), status (STRING), submitted_at (DATETIME), is_late (BOOLEAN). Має індекси на assignment_id, student_id, status.
Таблиця assignment_files має стовпці: id (INTEGER PRIMARY KEY), submission_id (INTEGER REFERENCES submissions), filename (STRING), file_path (STRING), file_size (INTEGER), mime_type (STRING). Має індекс на submission_id.
Таблиця news має стовпці: id (INTEGER PRIMARY KEY), name (STRING), description (STRING), department_id (INTEGER REFERENCES departments), user_id (INTEGER REFERENCES users), newscategory_id (INTEGER REFERENCES newscategory), photo_path (STRING), created_at (DATETIME).
Таблиця news_photos має стовпці: id (INTEGER PRIMARY KEY), image_path (STRING), news_id (INTEGER REFERENCES news).
Таблиця newscategory має стовпці: id (INTEGER PRIMARY KEY), name (STRING).
Таблиця events має стовпці: id (INTEGER PRIMARY KEY), department_id (INTEGER REFERENCES departments), date (DATE), created_at (DATETIME).
Таблиця volunteering має стовпці: id (INTEGER PRIMARY KEY), name (STRING), date_start (DATETIME), date_end (DATETIME), location

(STRING), department_id (INTEGER REFERENCES departments), user_id (INTEGER REFERENCES users), goal (INTEGER), desc (STRING), volunteeringcategory_id (INTEGER REFERENCES volunteeringcategory).

Таблиця volunteeringcategory має стовпці: id (INTEGER PRIMARY KEY), name (STRING).

Таким чином, проектування системи включало концептуальне проектування архітектури, побудову концептуальної моделі (ER-діаграми) та проектування реляційної бази даних відповідно до реальної структури проєкту. Розроблена модель відображає структуру предметної області та забезпечує цілісність даних, ефективність запитів та масштабованість системи.

2.3. Математичне та алгоритмічне забезпечення

Математичне та алгоритмічне забезпечення інформаційної системи управління навчальним процесом включає опис моделей реалізації методів, алгоритмів обробки даних, математичних моделей для розрахунку показників успішності, алгоритмів валідації даних, алгоритмів автентифікації та авторизації, а також алгоритмів для забезпечення продуктивності та масштабованості системи. Ці моделі та алгоритми є основою для програмної реалізації системи та забезпечують коректність, ефективність та надійність її роботи.

2.3.1. Математичні моделі розрахунку успішності та відвідуваності

Математичні моделі для розрахунку показників успішності студентів є важливим компонентом інформаційної системи. Основним показником успішності є середній бал студента, що розраховується на основі всіх оцінок студента за певний період. Математична модель розрахунку середнього балу має наступний вигляд:

$$\overline{G} = \frac{\sum_{i=1}^{n} g_i}{n}$$

де $\overline{G}$ — середній бал студента; g_i — значення оцінки, отриманої на i-му занятті; n — загальна кількість дисциплін, за які студент отримав оцінки. Без врахування вагових коефіцієнт.

Для розрахунку кінцевого балу студент ів з конкретної дисципліни використовується математична модель, що підраховує суму всіх балів з занять які оцінюються, що були виставлені викладачем:

$$G_{total,subject} = \sum_{i=1}^{m} g_i$$

де $G_{total,subject}$ — кінцевий бал студента з певної дисципліни, m — кількість оцінених занять з цієї дисципліни, g_i — значення оцінки за i-те заняття з цієї дисципліни.

Для розрахунку відвідуваності студентів використовується математична модель, що визначає відсоток відвідуваних занять:

$$A = \frac{N_{present}}{N_{total}} \cdot 100\%$$

де A — відсоток відвідуваності, $N_{present}$ — кількість відвіданих занять, N_{total} — загальна кількість занять.

Для визначення статусу відвідуваності використовуються наступні критерії: якщо $A \geq 90\%$, статус відвідуваності — відмінний; якщо $75\% \leq A < 90\%$, статус — добрий; якщо $60\% \leq A < 75\%$, статус — задовільний; якщо $A < 60\%$, статус — незадовільний.

2.3.2. Алгоритми валідації та контролю цілісності даних

Алгоритм валідації даних забезпечує коректність вхідних даних та відповідність їх визначеним обмеженням. Валідація виконується на декількох рівнях:

Валідація на рівні сервера включає перевірку даних на відповідність бізнес-логіці, інтеграційну цілісність даних та захист від ін'єкційних атак.

Валідація дат включає перевірку того, що дата початку заняття не стоїть в минулому:

$$\text{datetime\_valid} = (d_{schedule} >= d_{current}) \wedge (t_{start}) < t_{end})$$

де: $d_{schedule}$ — дата заняття в розкладі; $d_{current}$ — поточна дата; t_{start} — час початку заняття; t_{end} — час закінчення заняття

Для валідації дати створення (щоб не створювати в минулому):

$$\text{creation\_date\_valid} = d_{creation} >= d_{current})$$

де: $d_{creation}$ — дата створення запису розкладу; $d_{current}$ — поточна дата

Валідація оцінок включає перевірку діапазону значень:

$$\text{grade\_valid} = (g_{min} <= g <= g_{max})$$

де g_{min} та g_{max} — мінімальне та максимальне допустимі значення оцінки.

2.3.3. Алгоритми автентифікації та авторизації (безпека доступу)

Алгоритм автентифікації та авторизації забезпечує безпеку доступу до системи та захист даних користувачів. Алгоритм автентифікації включає наступні етапи:

Отримання облікових даних: отримання email та пароля від користувача.

Пошук користувача: пошук користувача в базі даних за email.

Перевірка пароля: порівняння хешу введеного пароля з хешем, збереженим у базі даних, з використанням алгоритму bcrypt.

Генерація токена: генерація JWT токена з інформацією про користувача (id, email, role).

Повернення токена: повернення токена клієнту для подальшого використання.

Алгоритм хешування пароля з використанням bcrypt має наступний вигляд:

$$\text{hash} = bcrypt(password, salt, cost)$$

де salt — випадкове сіль для захисту від таблиць, cost — фактор роботи, що визначає кількість ітерацій (використовується значення 12).

Алгоритм авторизації включає перевірку прав доступу користувача до певних ресурсів на основі його ролі. Математична модель авторизації:

$$\text{access\_granted} = (role \in \text{required\_roles})$$

де $role$ — роль користувача, required_roles — набір ролей, що мають доступ до ресурсу.

2.3.4. Алгоритм обробки запитів до REST API

Алгоритм обробки запитів до API забезпечує ефективну обробку HTTP запитів та повернення відповідей. Алгоритм включає наступні етапи:

Отримання запиту: отримання HTTP запиту від клієнта.

Валідація токена: перевірка валідності JWT токена.

Ідентифікація користувача: ідентифікація користувача на основі токена.

Перевірка прав доступу: перевірка прав доступу користувача до запитуваного ресурсу.

Валідація вхідних даних: валідація параметрів запиту.

Виконання бізнес-логіки: виконання операції з базою даних або зовнішніми сервісами.
Формування відповіді: формування JSON відповіді з результатом операції.
Повернення відповіді: повернення відповіді клієнту з відповідним HTTP статус-кодом.
Забезпечення продуктивності системи досягається через оптимізацію запитів до бази даних за допомогою індексів PostgreSQL та механізму joinedload бібліотеки SQLAlchemy, що дозволяє уникнути проблеми N+1 запитів при отриманні пов'язаних даних. В таблиці schedules створено складені індекси на полях group_id + date та окремо на date, що значно прискорює фільтрацію розкладу за датою та групою. Таблиці attendance, grades, submissions мають індекси на зовнішніх ключах для прискорення агрегаційних запитів журналу.

2.3.5. Алгоритми інтеграційних модулів (Telegram-бот та Google API)

Алгоритм асинхронної обробки повідомлень Telegram бота базується на подієво-орієнтованій моделі бібліотеки aiogram 3 та циклі подій asyncio. Dispatcher автоматично розподіляє вхідні оновлення між зареєстрованими обробниками (роутерами) на основі фільтрів та FSM-станів. Стан діалогу зберігається в MemoryStorage, що забезпечує збереження контексту між повідомленнями одного користувача. Ідентифікація користувача відбувається за полем from_user.id (Telegram ID), яке зіставляється з полем telegram_id у таблиці users бази даних для отримання JWT-токена авторизації.

Алгоритм оновлення токенів доступу до Google API забезпечує безперервну інтеграцію з Google Drive та Google Sheets. Перед кожним викликом Google API система перевіряє стан токена через умову creds.expired and creds.refresh_token. У разі закінчення терміну дії токена виконується його автоматичне оновлення через creds.refresh(Request()), після чого новий токен зберігається в базі даних. Математична модель для перевірки валідності токена:

$$\text{token\_valid} = (t_{current} < t_{expiry})$$

де $t_{current}$ — поточний час, t_{expiry} — час закінчення токена доступу.

ВИСНОВКИ ДО РОЗДІЛУ

У результаті виконання другого розділу дипломної роботи проведено детальний аналіз предметної області, проектування системи та розробку математичного та алгоритмічного забезпечення інформаційної системи управління навчальним процесом.

Аналіз предметної області дозволив виділити основні об'єкти дослідження, включаючи користувачів системи, навчальні групи, викладачів, навчальні дисципліни, розклад занять, оцінки, відвідуваність, завдання, навчальні матеріали, новини та події, волонтерську діяльність. Побудовано інформаційну модель, що відображає структуру предметної області та взаємозв'язки між об'єктами. Описано існуючі обмеження на вхідні та вихідні дані, включаючи валідацію формату даних, довжини рядків, типів даних, діапазонів значень, а також обмеження безпеки даних.

Проектування системи включало концептуальне проектування архітектури, побудову концептуальної моделі (ER-діаграми) та проектування реляційної бази даних. Архітектура системи базується на архітектурі клієнт-сервер з трьома основними компонентами: серверна частина (REST API на FastAPI), веб-інтерфейс (Vue.js з Vue Router) та Telegram бот (aiogram). Розроблено ER-діаграму, що відображає сутності предметної області та їх взаємозв'язки. Спроектовано реляційну базу даних на PostgreSQL з 21 таблицею, включаючи users, groups, teachers, subject, schedules, class_type, class_numbers, departments, specialties, roles, subject_config, attendance, grades, assignments, submissions, assignment_files, news, news_photos, newscategory, events, volunteering, volunteeringcategory. База даних відповідає реальній структурі проєкту, включаючи відсутність окремої таблиці аудиторій (аудиторії зберігаються як текст у таблиці schedules) та наявність додаткових таблиць для конфігурації предметів, здачі завдань, фотографій новин та категорій.

Математичне та алгоритмічне забезпечення включає математичні моделі для розрахунку показників успішності студентів (середній бал та сумарний бал з дисципліни), відвідуваності (відсоток відвіданих занять з визначенням статусу), алгоритми валідації вхідних даних (багаторівнева перевірка на рівні Pydantic-схем та бізнес-логіки), автентифікації та авторизації (хешування паролів з bcrypt, генерація та верифікація JWT-токенів), оптимізації запитів до бази даних (складені індекси PostgreSQL, механізм joinedload SQLAlchemy для уникнення N+1

запитів), асинхронної обробки повідомлень Telegram бота на основі подієво-орієнтованої моделі aiogram з MemoryStorage FSM для збереження стану діалогу, а також інтеграції з Google Drive та Google Sheets API з автоматичним оновленням OAuth 2.0 токенів доступу.

Результатом розділу є побудовані моделі для подальшої програмної реалізації інформаційної системи. Інформаційна модель, концептуальна модель (ER-діаграма), реляційна модель бази даних, математичні моделі та алгоритми забезпечують теоретичну основу для розробки програмного забезпечення та дозволяють перейти до наступного етапу — програмної реалізації системи.

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

Цей розділ присвячено практичній реалізації інформаційної системи управління навчальним процесом. На основі теоретичних моделей та алгоритмів, розроблених у другому розділі, описується конкретний технологічний стек, обґрунтовується вибір інструментів розробки, визначаються вимоги до програмного та технічного середовища, детально описується архітектура програмних модулів та надається керівництво для кінцевих користувачів системи.

Розроблена система складається з трьох незалежних програмних компонентів: серверної частини на базі FastAPI, клієнтського веб-додатку на базі Vue.js 3 та Telegram-бота на базі aiogram 3. Кожен компонент розроблявся як самостійний застосунок із чітко визначеним інтерфейсом взаємодії через REST API, що забезпечує модульність, незалежність масштабування та можливість заміни будь-якого компонента без впливу на інші. Такий підхід відповідає сучасним принципам розробки розподілених інформаційних систем та забезпечує гнучкість у розгортанні та подальшому супроводі.

3.1. Засоби розробки

Вибір засобів розробки є одним із ключових архітектурних рішень при створенні інформаційної системи, оскільки від нього залежать продуктивність, масштабованість, зручність підтримки та термін розробки. При виборі технологій для реалізації системи управління навчальним процесом враховувались наступні критерії: підтримка асинхронної обробки запитів, наявність активної спільноти та документації, відкритий вихідний код, продуктивність при роботі з реляційними базами даних, а також можливість інтеграції з зовнішніми сервісами (Google API, Telegram API).

3.1.1. Засоби розробки серверної частини

Основою серверної частини системи є мова програмування Python версії 3.11+. Вибір Python обґрунтовується широкою екосистемою бібліотек для роботи з базами даних, зовнішніми API та асинхронною обробкою запитів, а також лаконічним синтаксисом, що прискорює розробку та зменшує кількість помилок. Починаючи з версії 3.11, інтерпретатор CPython отримав суттєве прискорення виконання коду (до 60% порівняно з версією 3.10), що позитивно впливає на продуктивність серверних застосунків.

Веб-фреймворк FastAPI обрано як основу для побудови REST API. FastAPI побудований поверх бібліотеки Starlette та підтримує повністю асинхронну обробку запитів через async/await, що дозволяє ефективно обслуговувати велику кількість одночасних підключень без блокування потоку виконання. Ключовою перевагою FastAPI є автоматична генерація інтерактивної документації Swagger UI на основі Pydantic-схем та анотацій типів Python, що суттєво полегшує інтеграцію та тестування API. За результатами незалежних бенчмарків, FastAPI демонструє продуктивність, порівняно з Node.js та Go, що робить його одним із найшвидших Python-фреймворків для побудови API.

Для забезпечення постійності даних використовується PostgreSQL версії 15+ — одна з найрозвиненіших реляційних СУБД з відкритим вихідним кодом. PostgreSQL підтримує повний набір ACID-транзакцій, складні індекси (включаючи складені та часткові), повнотекстовий пошук, а також розширені типи даних. Для даної системи особливо важливою є підтримка зовнішніх ключів із каскадним видаленням (ON DELETE CASCADE) та унікальних складених індексів, що використовуються в таблиці subject_config для забезпечення унікальності конфігурації журналу на рівні СУБД.

Взаємодія з базою даних здійснюється через SQLAlchemy версії 2.x у асинхронному режимі з використанням драйвера asyncpg. SQLAlchemy є найбільш популярним ORM (Object-Relational Mapping) для Python, що надає потужний декларативний синтаксис для визначення моделей, підтримку міграцій через Alembic та гнучку систему запитів. Використання асинхронного режиму SQLAlchemy 2.x у поєднанні з asyncpg забезпечує неблокуючу взаємодію з базою даних, що критично важливо для продуктивності FastAPI-застосунку при одночасній обробці великої кількості запитів.

Міграції схеми бази даних управляються за допомогою Alembic версії 1.12.1 — інструменту міграцій, розробленого авторами SQLAlchemy. Alembic дозволяє відслідковувати зміни схеми бази даних у вигляді версійованих скриптів, що забезпечує відтворюваність розгортання та можливість відкату змін.

Валідація вхідних та вихідних даних реалізована за допомогою Pydantic версії 2. Pydantic забезпечує декларативне визначення схем даних на основі Python-анотацій типів з автоматичною валідацією, серіалізацією та десеріалізацією. Pydantic v2 переписано на мові Rust, що забезпечує

значне прискорення валідації порівняно з першою версією. У системі Pydantic-схеми використовуються як для валідації вхідних даних API-запитів, так і для формування структурованих відповідей.

Автентифікація та авторизація реалізовані на основі JWT (JSON Web Tokens) з використанням бібліотеки PyJWT. JWT-токени генеруються при успішній автентифікації та передаються клієнтом у заголовку Authorization: Bearer <token> при кожному запиті до захищених ресурсів. Хешування паролів користувачів виконується алгоритмом bcrypt з фактором роботи 12, що забезпечує стійкість до атак перебору навіть при витоку бази даних.

Захист від надмірного навантаження забезпечується бібліотекою slowapi — реалізацією rate limiting для FastAPI на основі алгоритму ковзного вікна. Slowapi дозволяє декларативно встановлювати ліміти запитів на рівні окремих ендпоінтів або глобально для всього застосунку.

Інтеграція з хмарними сервісами Google реалізована за допомогою офіційних клієнтських бібліотек: google-api-python-client та google-auth-oauthlib. Бібліотека google-api-python-client надає уніфікований інтерфейс для роботи з Google Drive API v3 та Google Sheets API v4, включаючи завантаження файлів, створення таблиць, керування правами доступу та читання/запис даних у комірки. google-auth-oauthlib забезпечує реалізацію потоку OAuth 2.0 для авторизації користувачів через Google-аккаунт та автоматичне оновлення токенів доступу. ASGI-сервер Uvicorn використовується для запуску FastAPI-застосунку у виробничому середовищі, забезпечуючи підтримку HTTP/1.1 та WebSocket.

3.1.2. Засоби розробки клієнтської частини (Frontend)

Клієнтський веб-додаток розроблено з використанням фреймворку Vue.js версії 3.5.30. Vue.js є прогресивним JavaScript-фреймворком для побудови реактивних користувацьких інтерфейсів, що базується на компонентній архітектурі. У системі використовується Composition API — сучасний підхід до організації логіки компонентів, що забезпечує кращу TypeScript-сумісність, зручне повторне використання логіки через composables та більш зрозумілу організацію коду порівняно з Options API.

Мова програмування TypeScript версії 5.9.3 використовується для статичної типізації всього клієнтського коду. TypeScript є надмножиною JavaScript, що додає систему типів, перевірку типів під час компіляції та покращену підтримку в IDE. Використання TypeScript дозволяє виявляти

помилки на етапі розробки, полегшує рефакторинг та підвищує читабельність коду у великих проєктах.

Управління станом застосунку реалізовано за допомогою Pinia версії 3.0.4 — офіційного сховища стану для Vue 3. Pinia замінила Vuex як рекомендований інструмент управління станом у екосистемі Vue.js, надаючи спрощений API без мутацій, повну TypeScript-підтримку та вбудовану інтеграцію з Vue DevTools. У системі Pinia store auth.ts відповідає за збереження JWT-токена, даних поточного користувача та його ролі між сторінками додатку.

Маршрутизація між сторінками реалізована через Vue Router версії 5.0.3 — офіційний маршрутизатор для Vue.js. Vue Router забезпечує декларативне визначення маршрутів, навігацію без перезавантаження сторінки (SPA-підхід), захист маршрутів через navigation guards (перевірка автентифікації перед доступом до захищених сторінок) та передачу параметрів між маршрутами. Система має 11 визначених маршрутів, кожен з яких відповідає окремому функціональному модулю.

Збірка та розробницький сервер забезпечуються інструментом Vite версії 7.3.1. Vite є сучасним інструментом збірки, що використовує нативні ES-модулі браузера під час розробки для забезпечення миттєвого запуску сервера та швидкого Hot Module Replacement (HMR). Для виробничої збірки Vite використовує Rollup, що забезпечує оптимізовані бандли з tree-shaking та code-splitting.

HTTP-запити до серверного API виконуються через бібліотеку Axios версії 1.14.0. Axios є Promise-базованим HTTP-клієнтом з підтримкою перехоплювачів запитів та відповідей (interceptors), що використовуються для автоматичного додавання JWT-токена до заголовків усіх запитів та централізованої обробки помилок авторизації (401 Unauthorized). Бібліотека marked версії 12.0.0 використовується для рендерингу Markdown-вмісту, зокрема для відображення описів завдань та новин.

3.1.3. Засоби розробки Telegram-бота

Telegram-бот розроблено на мові Python з використанням бібліотеки aiogram версії 3. aiogram є сучасним асинхронним фреймворком для розробки Telegram-ботів, що побудований на базі asyncio та підтримує повний Telegram Bot API. Ключовими перевагами aiogram 3 порівняно з альтернативами є підтримка FSM (Finite State Machine) для управління станом діалогу, фільтри повідомлень, middleware-архітектура та вбудована підтримка callback-кнопок і inline-клавіатур.

Для побудови складних інтерактивних діалогів з кнопками та переходами між станами використовується бібліотека aiogram-dialog, що надає декларативний підхід до визначення діалогових вікон у Telegram. Стан FSM між повідомленнями зберігається в MemoryStorage, що забезпечує збереження контексту діалогу в межах однієї сесії роботи бота. Взаємодія бота з серверним API здійснюється через асинхронні HTTP-запити з використанням бібліотеки aiohttp, що входить до залежностей aiogram 3.

Таким чином, вибір технологічного стеку для розробки інформаційної системи управління навчальним процесом обґрунтовується відповідністю кожного інструменту визначеним критеріям: асинхронна архітектура (FastAPI + asyncpg + aiogram), статична типізація (TypeScript), висока продуктивність (PostgreSQL, Pydantic v2, Vite), а також розвинена екосистема та активна спільнота, що гарантує довгострокову підтримку обраних рішень.

3.2. Вимоги до технічного та програмного забезпечення

Визначення вимог до технічного та програмного забезпечення є необхідним етапом проектування інформаційної системи, що дозволяє забезпечити коректне функціонування всіх її компонентів. Вимоги сформульовано відповідно до архітектури відкритих систем стандарту OSI (Open System Interconnection), розробленого International Standards Organization, з урахуванням трикомпонентної архітектури системи: серверна частина (REST API), клієнтський веб-застосунок та Telegram-бот.

Вимоги до технічного забезпечення

Розробка, тестування та дослідна експлуатація інформаційної системи здійснювалась на персональному комп'ютері з процесором AMD Ryzen 7 5700, що має 8 фізичних ядер та 16 потоків з тактовою частотою від 3.7 до 4.6 ГГц. Обсяг оперативної пам'яті становить 16 ГБ DDR4, накопичувач — SSD NVMe ємністю 512 ГБ. Як операційну систему використовується Windows 11.

На даній апаратній конфігурації всі три компоненти системи — FastAPI REST API, PostgreSQL СУБД та Telegram бот — запускались та функціонували одночасно. Сукупне споживання оперативної пам'яті усіма процесами системи в режимі роботи становить близько 700 МБ, що свідчить про ефективне використання ресурсів завдяки асинхронній архітектурі на базі FastAPI та драйвера asyncpg.

Для розгортання системи у виробничому середовищі мінімальні апаратні вимоги є значно нижчими від конфігурації розробницького середовища. Достатньо наявності двох процесорних ядер з частотою від 2.0 ГГц архітектури x86-64, 2 ГБ оперативної пам'яті (з огляду на фактичне споживання близько 700 МБ) та SSD-накопичувача ємністю від 10 ГБ для розміщення операційної системи, застосунків та бази даних. Обов'язковою умовою є наявність стабільного Інтернет-з'єднання для взаємодії з Google Drive API, Google Sheets API та Telegram API.

Щодо клієнтського технічного забезпечення, веб-інтерфейс системи є Single Page Application та функціонує безпосередньо у браузері без встановлення будь-якого додаткового програмного забезпечення. Система підтримує роботу на настільних комп'ютерах, ноутбуках, планшетах та смартфонах завдяки адаптивному дизайну інтерфейсу.

Вимоги до програмного забезпечення

Відповідно до стандарту OSI, програмне забезпечення системи поділяється на три рівні: системне, інструментальне та функціональне.

Системне програмне забезпечення є базовим рівнем, що забезпечує функціонування всіх інших компонентів. У розробницькому середовищі використовується операційна система Windows 11. Система не має жорсткої залежності від конкретної операційної системи — всі компоненти підтримують запуск на Windows 10/11, Ubuntu 20.04+, Debian 11+ та macOS 12+, що забезпечує переносимість середовища розробки та гнучкість при виборі серверної платформи Для клієнтської сторони вимогою є наявність сучасного браузера: Google Chrome 90+, Mozilla Firefox 88+, Microsoft Edge 90+ або Safari 14+. У розробницькому середовищі використовується Zen Browser браузер на базі Firefox

Інструментальне програмне забезпечення включає середовища виконання та рушії, необхідні для функціонування застосунків. Для серверної частини та Telegram бота необхідним є інтерпретатор мови Python версії 3.11 або вище — ця версія є мінімально необхідною для коректної роботи FastAPI та SQLAlchemy 2.x з підтримкою асинхронних контекстних менеджерів. PostgreSQL версії 15 або вище виступає системою управління реляційними базами даних та повинна бути налаштована з кодуванням UTF-8 для коректного зберігання текстів українською мовою. Node.js версії 20.19 або 22.12+ є необхідним виключно для збірки клієнтського Vue.js застосунку командою npm run build — після отримання готового статичного бандлу Node.js у виробничому

середовищі не потрібен. ASGI-сервер Uvicorn забезпечує запуск FastAPI-застосунку та обробку вхідних HTTP-запитів.

Функціональне (прикладне) програмне забезпечення є безпосередньою реалізацією інформаційної системи та складається з трьох незалежних програмних компонентів. Перший компонент — FastAPI REST API, розміщений у директорії OARestAPI/, — відповідає за всю бізнес-логіку системи та запускається через Uvicorn на порту 8000. Другий компонент — клієнтський Vue.js SPA, розміщений у директорії WebVue/schedule-project/, — у режимі розробки запускається через Vite dev-сервер, а для виробничого середовища збирається у статичний бандл. Третій компонент — Telegram бот на базі aiogram, розміщений у директорії BotOABOT/, — запускається як окремий Python-процес та підключається до Telegram API через механізм long polling.

Для коректного функціонування системи необхідна також наявність та попереднє налаштування трьох зовнішніх сервісів. Google Cloud Project з увімкненими Google Drive API та Google Sheets API надає можливість інтеграції з хмарним сховищем та електронними таблицями — для цього необхідний файл client_secret.json з обліковими даними OAuth 2.0. Telegram Bot API забезпечує роботу бота та потребує токена доступу, що зберігається у змінній середовища TELEGRAM_API_KEY_OA_BOT. PostgreSQL СУБД підключається через рядок підключення DATABASE_URL. Всі чутливі налаштування зберігаються у файлі .env та завантажуються через бібліотеку python-dotenv, що виключає їх потрапляння до системи контролю версій.

Таким чином, система демонструє мінімальні вимоги до апаратного забезпечення — близько 700 МБ оперативної пам'яті для роботи всіх трьох компонентів одночасно, — що підтверджує ефективність обраної асинхронної архітектури та робить систему придатною для розгортання як на потужних серверах, так і на скромних хостинг-конфігураціях.

3.3. Опис програмної реалізації

Програмна реалізація інформаційної системи управління навчальним процесом складається з трьох незалежних компонентів з чітко розмежованими зонами відповідальності: серверної частини на базі FastAPI, клієнтського веб-застосунку на базі Vue.js 3 та Telegram-бота на базі aiogram 3. Кожен компонент є самостійним застосунком, що взаємодіє з іншими виключно через REST API. У цьому підрозділі описується внутрішня структура кожного компонента з погляду

програміста — модулі, класи, методи, принципи їх функціонування та взаємозв'язки між ними.

Перед початком програмної реалізації систему було декомпозовано на сім функціональних модулів (епіків), кожен з яких охоплює окрему предметну область системи. Ця декомпозиція визначила структуру роутерів серверної частини, сторінок веб-застосунку та хендлерів Telegram-бота:

E-AUTH — Аутентифікація та керування обліковими записами. Охоплює всі процеси, необхідні для ідентифікації користувача: реєстрацію, вхід, вихід, відновлення доступу, а також базове управління користувачами для адміністраторів.

E-SCHEDULE — Управління розкладом. Включає всі процеси менеджменту занять відповідно до розпорядку: формування, редагування та перегляд розкладу в розрізі груп та викладачів.

E-MATERIALS — Надання навчальних матеріалів. Охоплює надання викладачем загальних посилань або індивідуальних для кожного заняття в розкладі джерел інформаційних матеріалів для студентів, а також оголошення тем занять.

E-ASSIGNMENTS — Завдання та здача робіт. Включає всі процеси взаємодії між студентом та викладачем: публікацію завдань, здачу виконаних робіт та їх оцінювання.

E-ATTENDANCE — Відвідуваність. Охоплює всі процеси, необхідні викладачу для документування присутності студентів на заняттях → ведення обліку не виходячи за межі системи на сторонні сервіси..

E-GRADING — Оцінювання та рейтинг. Включає всі процеси ведення документу з оцінюванням студентів і перегляду ними своїх оцінок та кінцевого балу з дисципліни.

E-REPORTS — Формування звітів. Охоплює всі процеси автоматизованого формування документації викладача із завантаженням на Google Drive.

3.3.1. Серверна частина (FastAPI REST API)

Файлова структура серверного компонента організована за принципом розділення відповідальності та має наступний вигляд:

```
OARestAPI/src/server/
├── main.py              ← точка входу застосунку
├── routesRegistration.py ← клас реєстрації маршрутів
├── database.py          ← налаштування підключення до БД
```

```
├── models.py          ← 21 SQLAlchemy-модель
├── schemas.py         ← Pydantic-схеми валідації
├── middlewares/       ← шар проміжного ПЗ (3 middleware)
├── routers/           ← 26 модулів маршрутизації
├── crud/              ← операції з базою даних
├── services/          ← сервіси (Google Drive, JWT, Auth)
└── usecases/          ← бізнес-логіка (use cases)
```

Точка входу застосунку — main.py. Файл є точкою запуску FastAPI-застосунку. Його основними функціями є ініціалізація екземпляра FastAPI з метаданими (назва «OAUniBot», версія 1.0.0, документація за адресою /docs), підключення middleware-шарів у визначеному порядку та делегування реєстрації маршрутів до класу RoutesRegistration. Middleware підключаються у зворотному порядку виконання: спочатку реєструється TimeMiddleWare (вимірювання часу обробки запиту), потім AuthMiddleWare (перевірка JWT-токена) та LoggerMiddleware (логування запитів). До застосунку також підключається SlowAPIMiddleware для rate limiting — при перевищенні ліміту запитів система повертає HTTP 429. CORSMiddleware дозволяє крос-доменні запити від клієнтської частини.

Middleware-шар реалізовано у директорії middlewares/ та складається з трьох компонентів. AuthMiddleWare перехоплює кожен вхідний HTTP-запит та перевіряє наявність і валідність JWT-токена у заголовку Authorization. Публічні маршрути (наприклад, /users/login, /users/google) включені до переліку дозволених без автентифікації та пропускаються без перевірки. Для захищених маршрутів middleware декодує токен через PyJWT, витягує ідентифікатор та роль користувача і передає їх у контекст запиту. TimeMiddleWare вимірює час виконання кожного запиту та додає його до заголовку відповіді X-Process-Time, що використовується для моніторингу продуктивності. LoggerMiddleware фіксує метод, URL-шлях, HTTP-статус відповіді та час виконання до системного журналу.

Клас RoutesRegistration реалізує патерн централізованої реєстрації маршрутів. Конструктор класу приймає екземпляр FastAPI-застосунку та ініціалізує залежності для модуля користувачів через Dependency Injection — створює репозиторії UserRepository та RoleRepository, а також об'єкт бізнес-логіки UserCases. Метод register_routes() реєструє 26 роутерів із відповідними URL-префіксами: /schedule, /users, /departments, /teachers, /groups, /journal, /manager, /assignments та інші. Такий підхід забезпечує

модульність — кожен роутер є незалежним модулем зі своїми ендпоінтами, схемами та CRUD-операціями.

Шар маршрутизації (routers/) містить 26 модулів, кожен з яких відповідає за окрему функціональну область відповідно до визначеної декомпозиції на епіки. Модуль journal.py, що реалізує епік E-GRADING та E-ATTENDANCE, забезпечує ендпоінти для отримання даних журналу у розрізі групи та дисципліни, оновлення відвідуваності та виставлення оцінок з синхронізацією до Google Sheets. Вхідними даними є ідентифікатори групи, предмету та типу заняття; вихідними — структурований масив даних по студентах із їх оцінками та статусом присутності на кожному занятті. Модуль manager.py, що охоплює епіки E-SCHEDULE та E-AUTH, є найбільшим роутером системи та реалізує адміністративні CRUD-операції для розкладу, викладачів, груп та дисциплін. Модуль assignment.py, що відповідає за епік E-ASSIGNMENTS, забезпечує повний цикл роботи із завданнями: створення, редагування, видалення, отримання списку завдань та управління здачами включно з оцінюванням та автоматичною перевіркою дедлайну.

Шар сервісів (services/) містить компоненти бізнес-логіки, що не є частиною маршрутизації. googleDriveService.py реалізує всі операції з Google Drive API v3 та Google Sheets API v4, що є основою епіків E-MATERIALS та E-REPORTS: створення папок, завантаження файлів, формування електронних журналів відвідуваності та оцінювання, генерацію звітів навантаження викладача. Перед кожним викликом Google API сервіс перевіряє валідність OAuth 2.0 токена та автоматично оновлює його у разі закінчення терміну дії, зберігаючи новий токен у базі даних. jwtauth.py реалізує генерацію та верифікацію JWT-токенів із визначеними термінами дії: 30 хвилин для access-токена та 7 днів для refresh-токена. userAuth.py надає FastAPI Dependency-функції для отримання поточного користувача, його ідентифікатора та ідентифікатора викладача з контексту запиту.

3.3.2. Клієнтська частина (Vue.js 3)

Файлова структура клієнтського компонента організована наступним чином:

```
WebVue/schedule-project/src/
├── App.vue          ← кореневий компонент, глобальні стилі, тема
├── main.ts          ← точка входу, ініціалізація Vue
```

```
├── router/          ← конфігурація маршрутизації (index.ts)
├── stores/          ← Pinia-сховища (auth.ts)
├── views/           ← 11 сторінок застосунку
├── components/      ← 4 перевикористовувані компоненти
├── composables/     ← composable-функції (useTheme)
└── api/             ← Axios-клієнти для API-запитів
```

Кореневий компонент App.vue є точкою монтування всього застосунку. Він відповідає за дві ключові функції: рендеринг поточного маршруту через директиву <RouterView /> та управління темою оформлення. Система тем реалізована через CSS-змінні (--bg-color, --text-primary, --accent-color тощо), що визначені у :root для темної теми та у класі .light-theme для світлої. Функція applySavedTheme() при монтуванні компонента зчитує збережену тему з localStorage та застосовує її, забезпечуючи персистентність налаштувань між сесіями. Глобальні CSS-стилі встановлюють мінімальні розміри інтерактивних елементів (44×44 пікселі) відповідно до рекомендацій з веб-доступності, підтримку режиму з підвищеним контрастом (prefers-contrast: high).

Маршрутизація налаштована у файлі router/index.ts з використанням режиму HTML5 History API (createWebHistory), що забезпечує чисті URL-адреси без символу #. Визначено 11 маршрутів, кожен з яких відображає URL-шлях на відповідний компонент-сторінку. Навігаційний захист router.beforeEach перехоплює кожен перехід між маршрутами та виконує три перевірки: неавторизовані користувачі перенаправляються на /login; вже авторизовані не можуть відкрити сторінку входу повторно; якщо токен є, але дані користувача ще не завантажені, викликається authStore.ensureUserLoaded() для відновлення сесії. Частина маршрутів містить метадані meta: { requiresAuth: true, roles: ['Teacher', 'Admin'] }, що обмежують доступ за роллю.

Сховище стану Pinia (stores/auth.ts) зберігає JWT-токен авторизованого користувача та його профільні дані (ім'я, email, роль, ідентифікатор групи). Метод ensureUserLoaded() перевіряє наявність токена у localStorage при першому завантаженні застосунку та відновлює сесію без повторного входу. Axios-інтерцептори автоматично додають токен до заголовка Authorization кожного HTTP-запиту до API, а у разі отримання відповіді з кодом 401 — перенаправляють користувача на сторінку входу.

Сторінки застосунку (views/) реалізують функціональні модулі відповідно до визначених епіків. LoginView.vue (E-AUTH) забезпечує

форму автентифікації з підтримкою входу через email/пароль та OAuth 2.0 через Google-акаунт. ScheduleView.vue (E-SCHEDULE, 73 КБ) реалізує перегляд розкладу з фільтрацією по даті та групі, відображенням детальної інформації про кожне заняття. JournalView.vue (E-GRADING + E-ATTENDANCE, 65 КБ) реалізує інтерфейс електронного журналу для викладача — таблицю оцінок та відвідуваності з можливістю inline-редагування та синхронізацією з Google Sheets. StudentJournalView.vue (E-GRADING) надає студентам доступ до власного журналу в розрізі кожної дисципліни. AssignmentsView.vue, AssignmentCreateView.vue та AssignmentEditView.vue (E-ASSIGNMENTS) реалізують повний цикл управління завданнями з боку викладача. SubmissionsView.vue (E-ASSIGNMENTS) забезпечує перегляд та оцінювання здач студентів. LessonManagerView.vue (E-SCHEDULE + E-MATERIALS, 124 КБ) є найбільшим компонентом системи та реалізує адміністративний інтерфейс управління заняттями. ClassNumberManagerView.vue (E-SCHEDULE) забезпечує управління номерами та часом пар.

Компоненти (components/) є перевикористовуваними елементами інтерфейсу. AssignmentSubmissionModal.vue реалізує модальне вікно здачі завдання (E-ASSIGNMENTS) з підтримкою трьох типів відправлення: текстом, файлом або через Google Docs. LessonManagerModal.vue — модальне вікно редагування деталей заняття (тема, відвідуваність, оцінки). ReportModal.vue (E-REPORTS) — модальне вікно генерації звіту навантаження викладача з можливістю збереження на Google Drive. ThemeToggle.vue — перемикач між темною та світлою темою із збереженням вибору у localStorage.

3.3.3. Telegram бот (aiogram 3)

Файлова структура бота організована за принципом відповідності хендлерів функціональним епікам:

```
BotOABOT/bot/
├── telegram_bot.py          ← точка входу, Dispatcher
├── handlers/
│   ├── user_handler.py      ← E-AUTH: реєстрація, авторизація
│   ├── schedule_handler.py  ← E-SCHEDULE: перегляд розкладу
│   ├── attendance_handler.py ← E-ATTENDANCE: відвідуваність
│   ├── grade_handler.py     ← E-GRADING: оцінки
│   ├── drive_handler.py     ← E-MATERIALS: Google Drive
```

```
│   ├── topic_handler.py        ← E-MATERIALS: теми занять
│   ├── report_handler.py       ← E-REPORTS: звіти
│   ├── news_commands.py        ← перегляд новин
│   └── interface_command_handler.py ← головне меню
└── services/
    ├── api_client.py           ← HTTP-клієнт для FastAPI
    ├── user_service.py         ← сервіс роботи з користувачами
    └── logging_service.py      ← логування
```

Точка входу telegram_bot.py ініціалізує Bot з токеном зі змінних середовища, створює Dispatcher з MemoryStorage для збереження FSM-станів та реєструє всі роутери через setup_routers(). Виклик setup_dialogs(dp) підключає бібліотеку aiogram_dialog для підтримки складних діалогових вікон зі станами. Бот запускається у режимі long polling через dp.start_polling(bot), що забезпечує безперервне отримання оновлень від Telegram API без потреби у публічній IP-адресі.

Модуль user_handler.py (E-AUTH) реалізує повний цикл реєстрації та авторизації. Вхідними даними є Telegram ID користувача (from_user.id), що використовується для ідентифікації. FSM-стани управляють покроковим процесом: вибір методу авторизації (Google OAuth або email/пароль), введення облікових даних та отримання JWT-токена від FastAPI. Отриманий токен зберігається у MemoryStorage та використовується у всіх подальших запитах до API від імені користувача.

Модуль schedule_handler.py (E-SCHEDULE) обробляє запити на перегляд розкладу. Вхідними даними є дата, що обирається через вбудований календар або вводиться текстом, та ідентифікатор групи користувача, що витягується з його профілю. Обробник виконує HTTP-запит до ендпоінту /schedule із відповідними параметрами та форматує відповідь у зручний для читання текст з inline-кнопками для навігації між датами.

Модуль attendance_handler.py (E-ATTENDANCE) дозволяє викладачу виставляти відвідуваність через бота. Викладач обирає заняття з розкладу, після чого бот отримує список студентів групи та відображає їх із кнопками «Присутній / Відсутній». Виставлена відвідуваність надсилається до API та синхронізується з Google Sheets. Аналогічну логіку реалізує grade_handler.py (E-GRADING) для виставлення оцінок. topic_handler.py та drive_handler.py (E-MATERIALS) забезпечують відповідно введення теми заняття та роботу з файлами та папками Google

Drive. report_handler.py (E-REPORTS) генерує звіт навантаження викладача та зберігає його у вигляді Google Sheets-таблиці на Drive.

Сервіс api_client.py є HTTP-клієнтом для взаємодії бота з FastAPI. Він інкапсулює всі HTTP-запити до серверного API, автоматично додаючи JWT-токен до заголовка Authorization, обробляючи помилки (401 — не авторизовано, 403 — немає доступу, 404 — не знайдено) та повертаючи структуровані відповіді обробникам.

Таким чином, програмна реалізація системи ґрунтується на чіткому розподілі відповідальності між компонентами відповідно до визначених функціональних епіків: серверна частина управляє бізнес-логікою та даними, клієнтський застосунок забезпечує зручний веб-інтерфейс, а Telegram-бот надає мобільний доступ до ключових функцій через месенджер. Усі три компоненти взаємодіють виключно через REST API, що забезпечує незалежність їх розробки, тестування та масштабування.

3.3.4. Аудит веб-доступності (a11y)

Веб-доступність (accessibility, скорочено a11y) визначає ступінь, до якого веб-застосунок може використовуватися людьми з різними типами обмежень — зоровими, руховими, слуховими або когнітивними. Реалізація доступності у клієнтській частині системи здійснена відповідно до рекомендацій WCAG (Web Content Accessibility Guidelines) та охоплює декілька рівнів.

Семантична розмітка та ARIA-атрибути. Компоненти застосунку використовують розширений набір ARIA-атрибутів для коректної взаємодії зі скрінрідерами. Модальні вікна (LessonManagerModal.vue, ClassNumberManagerView.vue) реалізовані з атрибутами role="dialog", aria-modal="true" та aria-labelledby, що дозволяє допоміжним технологіям коректно ідентифікувати та оголошувати їх вміст. Навігація між вкладками журналу реалізована з повноцінним набором атрибутів ARIA-pattern для tablist: role="tablist", aria-selected, aria-controls, aria-labelledby — що забезпечує навігацію клавіатурою між вкладками. Кнопки закриття модальних вікон мають атрибут aria-label="Закрити модальне вікно", а декоративні іконки — aria-hidden="true", що виключає їх з потоку читання скрінрідера.

Динамічні live-регіони. Для своєчасного оголошення змін стану застосунку без фокусування скрінрідером реалізовані ARIA live-регіони. Повідомлення про успішне збереження даних оголошуються з aria-live="polite" (очікує завершення поточного оголошення), а

повідомлення про помилки — з aria-live="assertive" (переривають поточне оголошення), що відповідає різному рівню терміновості інформації. Стани завантаження даних (role="status", aria-live="polite") також доступні для скрінрідерів. Поле статусу форми збереження оцінок та відвідуваності реалізоване як прихований live-регіон (class="visually-hidden", aria-live="polite"), що повідомляє про результат операції без видимого елемента на сторінці.

Валідація форм та описи полів. Поля введення оцінок мають атрибути aria-invalid (що динамічно встановлюється у true при наявності помилки валідації) та aria-describedby із посиланням на елемент з текстом помилки, що дозволяє скрінрідеру зачитати помилку разом із назвою поля. Поля форм мають підказки, підключені через aria-describedby, що надає контекстну допомогу при фокусуванні.

Управління фокусом. У модальних вікнах реалізована функція trapFocus(), що обмежує переміщення фокусу клавіатурою (клавіша Tab) у межах відкритого модального вікна. Це запобігає небажаному фокусуванню на прихованих за модальним вікном елементах сторінки — стандартна вимога WCAG 2.1 для діалогових вікон (критерій 2.1.2 «No Keyboard Trap»).

Розміри інтерактивних елементів. Усі кнопки та поля введення у глобальних стилях (App.vue) мають мінімальну висоту 44 пікселі на десктопі та 48 пікселів на мобільних пристроях, що відповідає рекомендаціям WCAG 2.5.5 (Target Size) для зручного використання на сенсорних екранах.

Підтримка системних налаштувань доступності. Застосунок підтримує два системних медіа-запити доступності. @media (prefers-reduced-motion: reduce) реалізований у всіх 8 файлах сторінок та у глобальних стилях — при увімкненні відповідного параметра в ОС усі CSS-анімації та переходи вимикаються (transition: none, animation: none). @media (prefers-contrast: high) підвищує контрастність меж та тексту для користувачів, що потребують підвищеного контрасту, коригуючи значення CSS-змінних --border-color та --text-muted.

Система тем та контрастність. Темна тема (за замовчуванням) використовує темний фон #0f172a з текстом #f1f5f9, акцентний колір #3b82f6. Світла тема використовує білий фон з темним текстом #0f172a та акцентом #2563eb. Обидві теми налаштовано з урахуванням коефіцієнта контрастності відповідно до вимог WCAG AA (мінімум 4.5:1 для тексту

нормального розміру). Перемикач тем (ThemeToggle.vue) має динамічний aria-label що оголошує поточну дію: «Переключити на світлу тему» або «Переключити на темну тему».

Таким чином, клієнтська частина системи реалізує комплексний підхід до веб-доступності: семантична ARIA-розмітка, live-регіони для динамічних оновлень, управління фокусом у модальних вікнах, підтримка системних налаштувань ОС (prefers-reduced-motion, prefers-contrast) та мінімальні розміри цільових областей відповідно до рекомендацій WCAG 2.1.

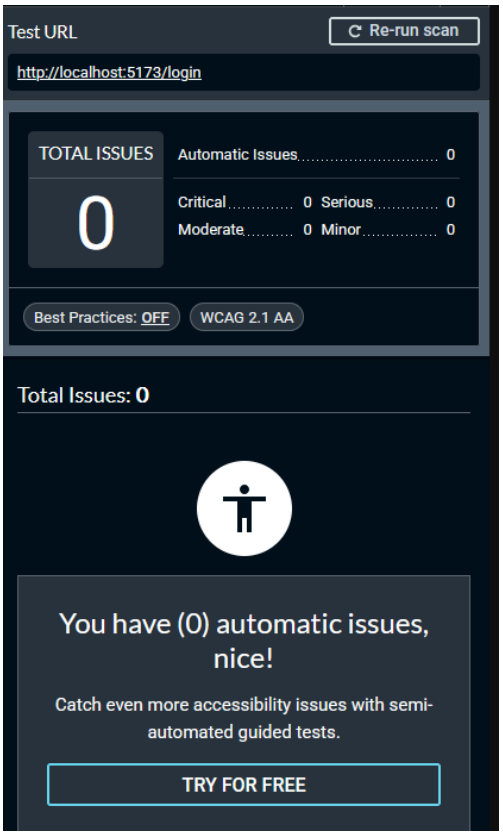

Рис. 1. Результат сканування вебсторінки через інструмент Axe DevTools

Для перевірки ефективності реалізованих рішень з веб-доступності було проведено автоматизоване тестування інтерфейсу за допомогою інструменту Axe DevTools. Результати сканування вебсторінки на відповідність критеріям міжнародного стандарту WCAG 2.1 рівня AA засвідчили повну відсутність помилок доступності (0 автоматично виявлених проблем, див. рис. 1). Це підтверджує коректність використання ARIA-атрибутів, правильну семантичну структуру документа та достатній рівень контрастності, що робить розроблену систему доступною для користувачів із допоміжними технологіями.

3.4. Керівництво користувача

Керівництво користувача описує порядок взаємодії із системою управління навчальним процесом з точки зору трьох основних ролей: адміністратора, викладача, менеджера розкладом та студента. Доступ до системи здійснюється через два рівноцінних клієнтських інтерфейси: веб-застосунок (для повноцінної роботи з комп'ютера) та Telegram-бот (для швидкого мобільного доступу до ключових функцій).

3.4.1. Початок роботи та аутентифікація

Робота із системою починається з процесу ідентифікації користувача. У веб-застосунку аутентифікація реалізована через єдину сторінку входу (рис. Б.1 - Б.2). Для забезпечення безпеки та зручності користувачів інтегровано технологію Single Sign-On (SSO) через облікові записи Google. Система відразу на етапі входу пропонує користувачу персоналізований досвід, підтримуючи як світлу, так і темну теми оформлення інтерфейсу.

Користувач натискає кнопку «Продовжити через Google», після чого система перенаправляє його на захищену сторінку авторизації Google. Після успішного підтвердження особи, користувач повертається до системи. Якщо електронна пошта належить до домену університету та зареєстрована в базі даних, система автоматично розпізнає роль користувача (студент або викладач) і надає відповідний рівень доступу.

У Telegram-боті процес авторизації відбувається через команду /start (рис. Д.1). Бот пропонує користувачу надіслати свою електронну адресу та пароль, або пройти швидку авторизацію через прив'язку Telegram-акаунта до профілю в системі.

3.4.2 Telegram-бот

Telegram-бот виступає зручним мобільним доповненням до основної веб-системи, його головне призначення — швидке надання актуальної інформації про розклад занять без необхідності відкривати браузер чи вмикати комп'ютер. Робота з ботом розпочинається з команди /start та подальшої авторизації (шляхом прив'язки Telegram-акаунта до існуючого профілю викладача в системі).

Після успішного входу користувач отримує доступ до головного навігаційного меню, яке реалізоване у вигляді інтерактивних inline-кнопок (див. Рис. Г.1). Основний функціонал Telegram-бота

повністю сфокусований на оперативній видачі інформації про навчальне навантаження.

Викладач має можливість швидко переглянути свій графік занять через відповідне меню (див. Рис. Г.2), яке пропонує такі опції:

Розклад на сьогодні: система миттєво формує перелік занять на поточну добу, чітко вказуючи час початку та завершення пари, номер аудиторії, тип заняття, назву дисципліни та закріплену групу.

Розклад на завтра: дозволяє викладачеві швидко зорієнтуватися щодо планування та підготовки до наступного робочого дня.

Пошук за конкретною датою: викладач може вказати довільну дату в календарі, і бот поверне розклад саме на цей день.

Додатково в боті реалізовано можливість пошуку предмета за його назвою. Видача розкладу форматується у вигляді структурованих текстових повідомлень з використанням емодзі, що забезпечує максимальну зручність читання з екрана мобільного телефону. Такий підхід дозволяє викладачеві завжди мати під рукою актуальний графік своїх пар у будь-якому місці та в будь-який час.

3.4.3. Інтерфейс викладача: журнал, завдання та звіти

Після сторінки входу (див. рис. Б.1–Б.2) та успішної авторизації в системі викладач потрапляє на головну сторінку (див. рис. В.1), звідки має доступ до свого розкладу (див. рис. В.2–В.3). У розкладі реалізовано можливість керувати заняттями через модальне вікно, яке розділене на секції для швидкого виставлення оцінок, обліку відвідуваності (див. рис. В.4) та оновлення теми заняття (див. рис. В.5). З головної сторінки також здійснюється перехід до електронних журналів (див. рис. В.6–В.7) та сторінки створення завдань для студентів (див. рис. Б.7–Б.8). Крім того, в інтерфейсі доступні модальне вікно для генерації звіту педагогічного навантаження (див. рис. В.8) та інформаційне вікно «Шкала оцінювання» (див. рис. В.9), що пояснює логіку конвертації числової оцінки у відповідне значення літерної шкали (наприклад, «A — Відмінно» тощо).

Також із головної сторінки здійснюється перехід до модуля управління завданнями (див. рис. В.10–В.11), де викладач може відслідковувати статус створених завдань, терміни їх виконання (дедлайни), кількість студентів, які вже здали роботи, та взаємодіяти з елементами керування. Через кнопку переходу до сторінки перевірки здач (див. рис. В.12–В.13) викладач отримує доступ до робіт студентів. Відкривши надіслану роботу, можна переглянути вміст відповіді: текст, прикріплений файл або

посилання (див. рис. В.13). Якщо завдання виконано успішно, викладач виставляє відповідний бал в окремому модальному вікні (див. рис. В.14). У випадку, коли надіслана відповідь є невалідною або помилковою, викладач має змогу анулювати її — після цього студент отримає можливість повторного завантаження рішення.

Інтерфейс розкладу надає можливість зручної навігації за допомогою вбудованого календаря. Він дозволяє переглядати детальний перелік занять на будь-яку вибрану дату, де для кожної пари чітко зазначено час проведення, номер аудиторії, тип заняття (лекція, практична чи інше) та прив'язану до нього навчальну групу.

Електронний журнал є основним робочим інструментом викладача, що суттєво спрощує рутинне ведення академічної документації. Цей модуль поєднує в єдиному інтерфейсі процеси обліку відвідуваності та моніторингу успішності студентів. Для початку роботи викладачеві достатньо вибрати з випадного списку потрібну дисципліну та відповідну навчальну групу, після чого система генерує інтерактивну зведену таблицю. У цій таблиці стовпцями виступають дати проведених занять, а рядками — прізвища студентів. Завдяки механізмам inline-редагування викладач має змогу вносити зміни (виставляти оцінки чи позначати пропуски) безпосередньо у клітинки таблиці. Будь-які зміни автоматично валідуються системою для уникнення помилок вводу та зберігаються в базі даних із подальшою синхронізацією.

Логічним продовженням функціоналу журналу є модуль створення завдань та їх оцінювання, який фактично інтегрує в систему можливості повноцінних систем управління навчанням (LMS, на кшталт Moodle). Викладач має змогу формувати завдання для самостійної роботи, додавати до них детальні інструкції з підтримкою форматування Markdown, прикріплювати зовнішні ресурси та встановлювати жорсткі терміни виконання (дедлайни). На сторінці оцінювання викладач бачить консолідований список усіх надісланих студентами робіт, має можливість переглянути вміст відповіді (посилання прикріплений файл в гугл диску, посилання або текст), перевірити її на валідність і виставити бал. Отримана за завдання оцінка автоматично переноситься до загального електронного журналу, що виключає необхідність подвійного введення даних.

3.4.4. Інтерфейс студента: розклад, журнал та здача завдань

Студентський інтерфейс веб-застосунку розроблено з фокусом на забезпечення швидкого доступу до навчальних матеріалів та зручного контролю власної успішності. Після успішної авторизації студент потрапляє на головну сторінку(див. Рис. Д.1. - Д.2.) сторінку розкладу (див. Рис. Д.3. - Д.4.). На відміну від викладацького інтерфейсу, розклад студента автоматично фільтрується та адаптується під його академічну групу, відображаючи лише релевантні для нього заняття у вигляді інформативних карток.

Окрім базової організаційної інформації (час проведення, номер аудиторії, ім'я викладача), картка заняття слугує єдиною точкою входу для роботи із самостійними завданнями. Якщо викладач прикріпив до заняття практичну роботу, студент може переглянути її деталі, відкривши спеціальне модальне вікно (див. рис. Д.5. - Д.6.). У цьому вікні здобувач освіти має змогу ознайомитися з детальними інструкціями, кінцевими термінами виконання (дедлайнами) та поточним статусом перевірки. Здача виконаної роботи відбувається безпосередньо у цьому ж вікні: система пропонує гнучкий підхід, дозволяючи студенту ввести текст відповіді у редактор, прикріпити локальний файл або вставити посилання на документ у хмарному сховищі. Після відправки матеріалів статус завдання оновлюється, фіксуючи факт здачі.

Для постійного моніторингу власної успішності реалізовано модуль «Мій журнал» (Student Journal) (див. рис. Д.7. - Д.8.). Ця сторінка забезпечує абсолютну прозорість процесу оцінювання, надаючи студенту цілодобовий доступ до статистики його академічної діяльності. Інтерфейс генерує зведену інформацію за всіма дисциплінами, де відображаються:

- усі отримані бали за роботу на парах та виконані завдання;
- статистика відвідуваності (кількість пропусків);
- поточний підсумковий бал (рейтинг) з кожного предмета.

Така ергономічна організація робочого простору дозволяє студенту ефективно планувати свій час, своєчасно закривати дедлайни по завданнях та об'єктивно оцінювати свої результати навчання без необхідності додаткових звернень до викладачів.

3.4.5. Інтерфейс менеджера розкладом

Роль менеджера розкладу (адміністратора) передбачає глобальне управління навчальним процесом, що вимагає наявності потужного та гнучкого інструментарію з повним доступом до бази даних (CRUD

операції). Після авторизації адміністратор потрапляє на головне вікно застосунку (див. рис. Е.1), звідки здійснюється швидка навігація до ключових модулів управління системою.

Глобальний перегляд розкладу Інтерфейс сторінки «Розклад» для менеджера концептуально відрізняється від студентського чи викладацького. Замість фіксованого персонального графіка, менеджеру доступний окремий елемент управління — селектор вибору академічної групи. При виборі певної групи система динамічно генерує та відображає розклад лише для неї (див. рис. Е.2).

Окрім перегляду, безпосередньо з розкладу менеджер має можливість управління конкретним заняттям. При натисканні на картку пари на екрані з'являється спеціальне модальне вікно (див. рис. Е.3), у якому адміністратор може оперативно відредагувати ключову інформацію про заняття: перенести його на іншу дату, змінити порядковий номер пари або перепризначити аудиторію. Це дозволяє швидко вносити точкові зміни в розклад (наприклад, при раптовій заміні адуторії), не переходячи до загального реєстру занять.

Модуль «Управління заняттями» та перевірка даних Ядром адміністративного інтерфейсу є сторінка «Управління заняттями» (Lesson Manager). Цей модуль структурно поділений на кілька функціональних секцій для поетапної роботи з розкладом.

Першою важливою секцією є моніторинг «Статусу даних». У згорнутому вигляді (без таблиці деталей, див. рис. Е.4) вона надає адміністратору загальне зведення про наявність необхідних довідкових даних (дисциплін, викладачів, груп) у системі. При розгортанні цієї секції з'являється таблиця деталей статусу даних (див. рис. Е.5), яка поглиблено деталізує інформацію та дозволяє швидко виявити, чого саме бракує для коректного формування розкладу. Для коригування цих довідників передбачено вікно редагування даних (див. рис. Е.6), через яке менеджер може оперативно вносити зміни (наприклад, змінити назву дисципліни) без необхідності прямого звернення до бази даних через SQL-запити.

Процес створення занять та конфігурація навчального плану Процес додавання нових пар до розкладу відбувається через спеціальне вікно створення заняття (див. рис. Е.7). У цьому вікні адміністратор обирає всі необхідні атрибути майбутньої пари: дату проведення, дисципліну, викладача, тип заняття (лекція, практика, семінар), цільову групу та аудиторію.

Унікальною архітектурною особливістю розробленої системи є механізм первинної ініціалізації: при першому додаванні нового заняття для певної дисципліни автоматично відкривається вікно створення конфігурації заняття (див. рис. Е.8). У ньому менеджер повинен вказати загальну кількість запланованих занять згідно з навчальним планом. Ця конфігурація закладає ліміти вичитки курсу, що дозволяє системі надалі автоматизувати контроль за виконанням педагогічного навантаження викладачами.

Робота з реєстрами та фільтрація Всі створені пари потрапляють до єдиного реєстру бази даних. Менеджер може переглядати пусту таблицю доки фільтри не будуть застосовані(див. рис. Е.9), що відображає дозволяє покращити перфоменс замість відображення усіх створених пар в університеті. Однак, для зручної роботи з великими масивами даних передбачено використання потужної системи сортування: список занять з фільтром (див. рис. Е.10) дозволяє миттєво відібрати записи за конкретною групою, викладачем або діапазоном дат. Це критично важливо для масового редагування або видалення скасованих чи перенесених пар.

Управління часовою сіткою закладу Останнім ключовим елементом адміністративної панелі є сторінка «Управління парами» (Class Numbers Manager) (див. рис. Е.11). Вона слугує інструментом для налаштування глобального розкладу дзвінків навчального закладу. Інтерфейс надає можливість створювати нові порядкові номери пар (перша, друга тощо), а також редагувати чи видаляти їхні часові межі (час початку та закінчення). Завдяки такій динамічній архітектурі, система не має жорстко закодованих часових слотів і може бути переналаштована менеджером за кілька кліків у разі зміни загального графіка роботи університету (наприклад, при переході на скорочені заняття).

ВИСНОВКИ ДО РОЗДІЛУ

У третьому розділі дипломної роботи було детально розглянуто процеси вибору технологій, архітектурного проєктування та практичної програмної реалізації інтегрованої системи управління навчальним процесом. За результатами виконання цього етапу дослідження можна зробити наступні висновки:

Обґрунтовано вибір технологічного стека. Для створення системи було обрано сучасну модульну архітектуру на базі REST API. Використання фреймворку FastAPI (Python) у поєднанні з реляційною базою даних

PostgreSQL забезпечило високу швидкість обробки запитів, типізацію та надійне транзакційне збереження даних. Клієнтська частина, побудована на базі Vue.js 3 та TypeScript, дозволила створити реактивний клієнтський застосунок, що мінімізує навантаження на мережу. Додатково, для забезпечення багатоканальності доступу, успішно інтегровано Telegram-бота за допомогою асинхронної бібліотеки aiogram.

Здійснено функціональну декомпозицію системи. Процес розробки було розділено на 7 базових епіків (E-AUTH, E-SCHEDULE, E-MATERIALS, E-ASSIGNMENTS, E-ATTENDANCE, E-GRADING, E-REPORTS). Такий підхід дозволив створити гнучку систему, де кожен модуль автономно відповідає за свій бізнес-процес (наприклад, управління розкладом чи генерацію звітів), але при цьому всі вони безшовно взаємодіють між собою через загальну базу даних.

Забезпечено високий рівень вебдоступності (a11y). Клієнтський інтерфейс розроблено з дотриманням міжнародних стандартів інклюзивності WCAG 2.1 AA. Реалізовано підтримку допоміжних технологій (через семантичну розмітку та ARIA-атрибути), управління фокусом клавіатури, контрастні теми оформлення (світла та темна) і системні медіа-запити. Автоматизований аудит інструментом Axe DevTools підтвердив відсутність бар'єрів для користувачів з особливими потребами.

Спроєктовано рольові інтерфейси користувачів. Розроблено та задокументовано у вигляді керівництва логіку взаємодії із системою для трьох ключових ролей: менеджера розкладу, викладача та студента. Інтерфейси адаптовано під специфіку кожної ролі — від інструментів динамічного налаштування розкладу та конфігурацій занять для адміністратора, до інтегрованого журналу, системи перевірки завдань та генерації звітностей для викладача. Дублювання критичного функціоналу у Telegram-боті забезпечило додаткову мобільність учасників процесу.

Отже, розроблений програмний продукт повністю відповідає технічним вимогам та меті дослідження. Створена система здатна ефективно автоматизувати рутинні процеси організації навчання, формує єдиний цифровий простір для викладачів і студентів та є функціонально готовою до впровадження у реальну практику закладу освіти.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне науково-практичне завдання, яке полягає у проєктуванні та розробці інтегрованої програмної системи управління навчальним процесом для закладів вищої освіти.

Проведений на початку дослідження аналіз популярних освітніх платформ, таких як LMS Moodle, Google Classroom та різноманітних локальних електронних журналів, виявив низку їхніх системних недоліків. Серед них ключовими є надмірна перевантаженість користувацького інтерфейсу, фрагментованість даних, відсутність гнучкості в управлінні розкладом та слабка адаптація до мобільного використання. Це повністю обґрунтувало доцільність розробки єдиного, оптимізованого інтегрованого рішення, яке б об’єднало в собі необхідний функціонал та усунуло існуючі бар’єри.

Для реалізації такого рішення було спроєктовано багаторівневу клієнт-серверну архітектуру на базі концепції REST API. За допомогою інструментарію мови моделювання UML змодельовано ключові бізнес-процеси та розроблено нормалізовану схему реляційної бази даних(див. Рис. А.1), здатну ефективно й безперебійно обробляти транзакції, пов'язані з налаштуванням розкладу, обліком відвідуваності, оцінюванням та управлінням завданнями.

З метою забезпечення високої швидкодії та надійності системи було здійснено обґрунтований вибір сучасного технологічного стека. Серверну частину реалізовано за допомогою потужного асинхронного фреймворку FastAPI мовою Python у поєднанні з системою управління базами даних PostgreSQL. Клієнтську частину побудовано як сучасний реактивний веб-застосунок на базі фреймворку Vue.js 3 та мови TypeScript. Додатково, як альтернативний мобільний клієнт, розроблено Telegram-бот.

У рамках практичної реалізації було успішно створено спеціалізовані адаптивні інтерфейси для трьох ключових системних ролей. Для адміністраторів навчального закладу розроблено інструментарій динамічного управління часовою сіткою, конфігурації занять та глобального моніторингу розкладу. Для викладачів створено багатофункціональний інтерактивний електронний журнал, повноцінний модуль створення та перевірки завдань, а також підсистему автоматичної генерації звітів із вивантаженням документів на платформу Google Drive. Інтерфейс студента натомість забезпечує швидкий доступ до

персоналізованого розкладу, механізмів здачі самостійних робіт та цифрового журналу моніторингу власної успішності.

Особливим здобутком виконаної роботи є забезпечення глибокої інклюзивності розробленого вебзастосунку згідно з міжнародними стандартами вебдоступності WCAG 2.1 AA. Крім того, інтеграція Telegram-бота значно розширила мобільність та автономність викладачів, дозволивши їм здійснювати перегляд потрібної інформації прямо в месенджері , що дозволяє за потреби поділитись відразу повідомлення з відповіддю без потреби робитис скріншоти. Отримані результати підтверджують повну працездатність та ефективність системи. Розроблений програмний продукт має високу практичну цінність: він успішно вирішує проблему розрізненого та незручного програмного забезпечення, об'єднуючи розклад, облік успішності та середовище обміну навчальними матеріалами у єдиний цифровий простір. Програма є функціонально готовою до дослідної експлуатації та впровадження у реальний освітній процес. Перспективами подальшого вдосконалення проєкту визначено розширення аналітичного модуля алгоритмами машинного навчання для прогнозування академічної успішності студентів, помічник який буде нотифікувати про велику к-сть пропусків або заборгований завдань, система гейміфікації.

СПИСОК ДЖЕРЕЛ

1. Layered Architecture Pattern [Електронний ресурс] // Microsoft Azure. — Режим доступу: https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design. — Дата звернення: 20.03.2025.
2. RESTful web API design [Електронний ресурс] // Microsoft Azure. — Режим доступу: https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design. — Дата звернення: 21.03.2025.
3. Single-page application [Електронний ресурс] // MDN Web Docs. — Режим доступу: https://developer.mozilla.org/en-US/docs/Glossary/SPA. — Дата звернення: 25.03.2025.
4. Python 3 Documentation [Електронний ресурс] // Python Software Foundation. — Режим доступу: https://docs.python.org/3/. — Дата звернення: 10.04.2025.
5. typing — Support for type hints [Електронний ресурс] // Python Software Foundation. — Режим доступу: https://docs.python.org/3/library/typing.html. — Дата звернення: 28.03.2025.
6. Asynchronous Programming in Python [Електронний ресурс] // Real Python. — Режим доступу: https://realpython.com/async-io-python/. — Дата звернення: 15.04.2025.
7. FastAPI Documentation [Електронний ресурс] // FastAPI. — Режим доступу: https://fastapi.tiangolo.com/. — Дата звернення: 12.04.2025.
8. Starlette: The little ASGI framework that shines [Електронний ресурс] // Encode. — Режим доступу: https://www.starlette.io/. — Дата звернення: 05.04.2025.
9. Pydantic: Data validation using Python type hints [Електронний ресурс] // Pydantic. — Режим доступу: https://docs.pydantic.dev/. — Дата звернення: 18.04.2025.
10. Uvicorn: An ASGI web server, for Python [Електронний ресурс] // Encode. — Режим доступу: https://www.uvicorn.org/. — Дата звернення: 02.04.2025.
11. PostgreSQL: The World's Most Advanced Open Source Relational Database [Електронний ресурс] // PostgreSQL Global Development

Group. — Режим доступу: https://www.postgresql.org/docs/. — Дата звернення: 14.04.2025.
12. SQLAlchemy: The Python SQL Toolkit and Object Relational Mapper [Електронний ресурс] // SQLAlchemy Authors. — Режим доступу: https://docs.sqlalchemy.org/en/20/. — Дата звернення: 22.04.2025.
13. SQLAlchemy: Loading Relationships [Електронний ресурс] // SQLAlchemy Authors. — Режим доступу: https://docs.sqlalchemy.org/en/20/orm/loading_relationships.html. — Дата звернення: 25.04.2025.
14. Alembic: A database migrations tool for SQLAlchemy [Електронний ресурс] // SQLAlchemy Authors. — Режим доступу: https://alembic.sqlalchemy.org/en/latest/. — Дата звернення: 26.04.2025.
15. JSON Web Token (JWT) [Електронний ресурс] // Auth0. — Режим доступу: https://jwt.io/introduction. — Дата звернення: 19.04.2025.
16. PyJWT Documentation [Електронний ресурс] // PyJWT. — Режим доступу: https://pyjwt.readthedocs.io/en/stable/. — Дата звернення: 20.04.2025.
17. Bcrypt: Password hashing function [Електронний ресурс] // Wikipedia. — Режим доступу: https://en.wikipedia.org/wiki/Bcrypt. — Дата звернення: 17.04.2025.
18. Vue.js: The Progressive JavaScript Framework [Електронний ресурс] // Vue Technology LLC. — Режим доступу: https://vuejs.org/guide/introduction.html. — Дата звернення: 10.04.2025.
19. Vue Router: The official router for Vue.js [Електронний ресурс] // Vue Technology LLC. — Режим доступу: https://router.vuejs.org/. — Дата звернення: 11.04.2025.
20. Pinia: The intuitive store for Vue.js [Електронний ресурс] // Vue Technology LLC. — Режим доступу: https://pinia.vuejs.org/. — Дата звернення: 12.04.2025.
21. TypeScript: Typed JavaScript at Any Scale [Електронний ресурс] // Microsoft. — Режим доступу: https://www.typescriptlang.org/docs/. — Дата звернення: 05.04.2025.
22. Vite: Next Generation Frontend Tooling [Електронний ресурс] // Vite. — Режим доступу: https://vitejs.dev/guide/. — Дата звернення: 06.04.2025.

23. Web Content Accessibility Guidelines (WCAG) 2.1 [Електронний ресурс] // W3C. — Режим доступу: https://www.w3.org/TR/WCAG21/. — Дата звернення: 01.05.2025.
24. WAI-ARIA Overview [Електронний ресурс] // W3C Web Accessibility Initiative. — Режим доступу: https://www.w3.org/WAI/standards-guidelines/aria/. — Дата звернення: 02.05.2025.
25. ARIA: dialog role [Електронний ресурс] // MDN Web Docs. — Режим доступу: https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Roles/dialog_role. — Дата звернення: 03.05.2025.
26. Axe DevTools Accessibility Scanner [Електронний ресурс] // Deque Systems. — Режим доступу: https://www.deque.com/axe/devtools/. — Дата звернення: 05.05.2025.
27. Telegram Bot API [Електронний ресурс] // Telegram Messenger LLP. — Режим доступу: https://core.telegram.org/bots/api. — Дата звернення: 28.04.2025.
28. Aiogram 3.x Documentation [Електронний ресурс] // Aiogram. — Режим доступу: https://docs.aiogram.dev/en/latest/. — Дата звернення: 29.04.2025.
29. Finite State Machines in Python [Електронний ресурс] // Real Python. — Режим доступу: https://realpython.com/python-finite-state-machine/. — Дата звернення: 02.05.2025.
30. Google Identity: OAuth 2.0 for Web Server Applications [Електронний ресурс] // Google Developers. — Режим доступу: https://developers.google.com/identity/protocols/oauth2/web-server. — Дата звернення: 08.05.2025.
31. Google Drive API [Електронний ресурс] // Google Developers. — Режим доступу: https://developers.google.com/drive/api/guides/about-sdk. — Дата звернення: 04.05.2025.
32. Google Sheets API [Електронний ресурс] // Google Developers. — Режим доступу: https://developers.google.com/sheets/api/guides/concepts. — Дата звернення: 04.05.2025.

33. Moodle LMS: Open-source learning platform [Електронний ресурс] // Moodle. — Режим доступу: https://moodle.org/. — Дата звернення: 10.03.2025.
34. Google Classroom Help [Електронний ресурс] // Google. — Режим доступу: https://support.google.com/edu/classroom/. — Дата звернення: 12.03.2025.
35. Markdown Guide: Basic Syntax [Електронний ресурс] // Markdown Guide. — Режим доступу: https://www.markdownguide.org/basic-syntax/. — Дата звернення: 06.05.2025.

ДОДАТКИ

Додаток А

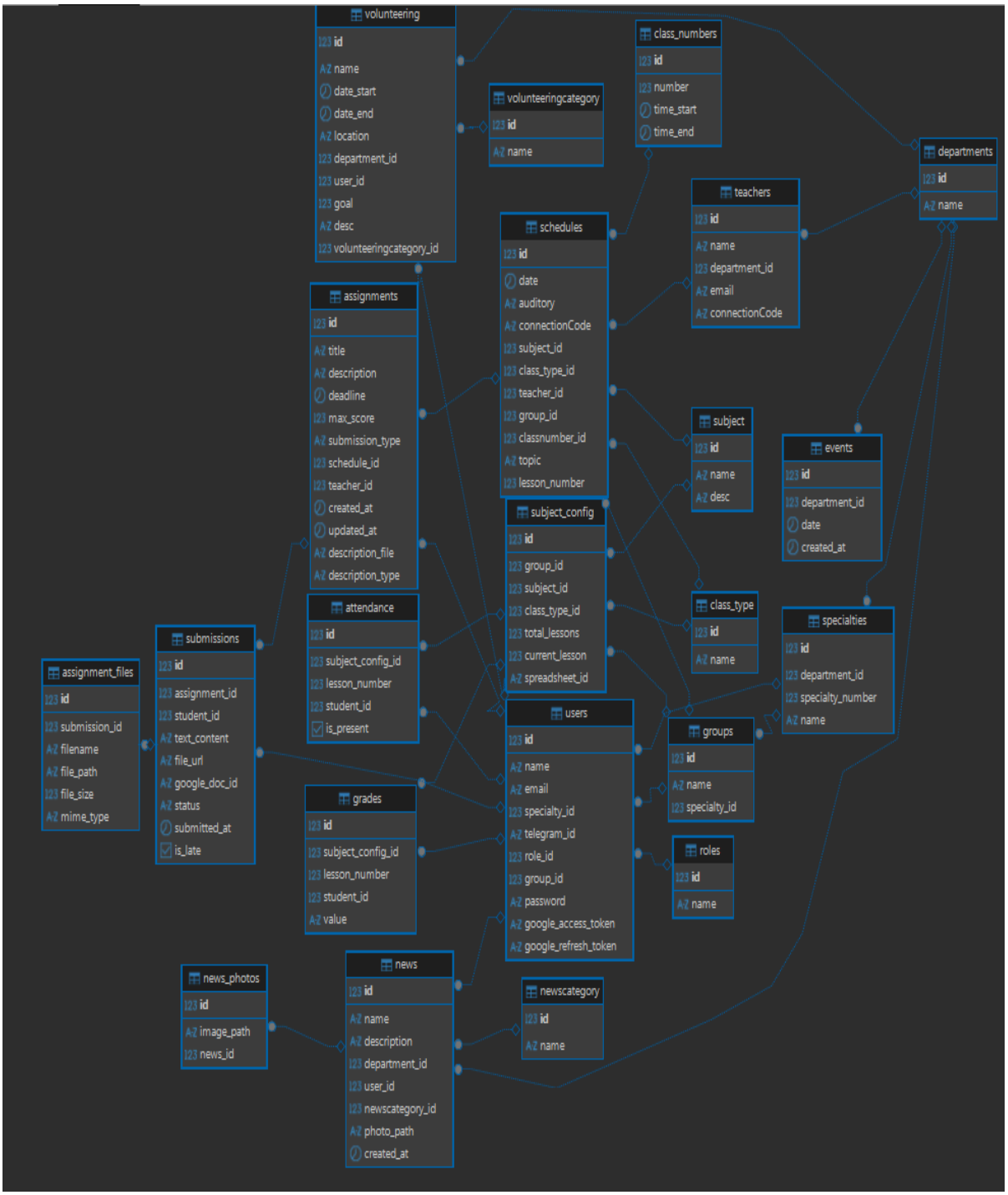

Рис. А.1. ER-діаграма реляційної бази даних інформаційної системи управління навчальним процесом

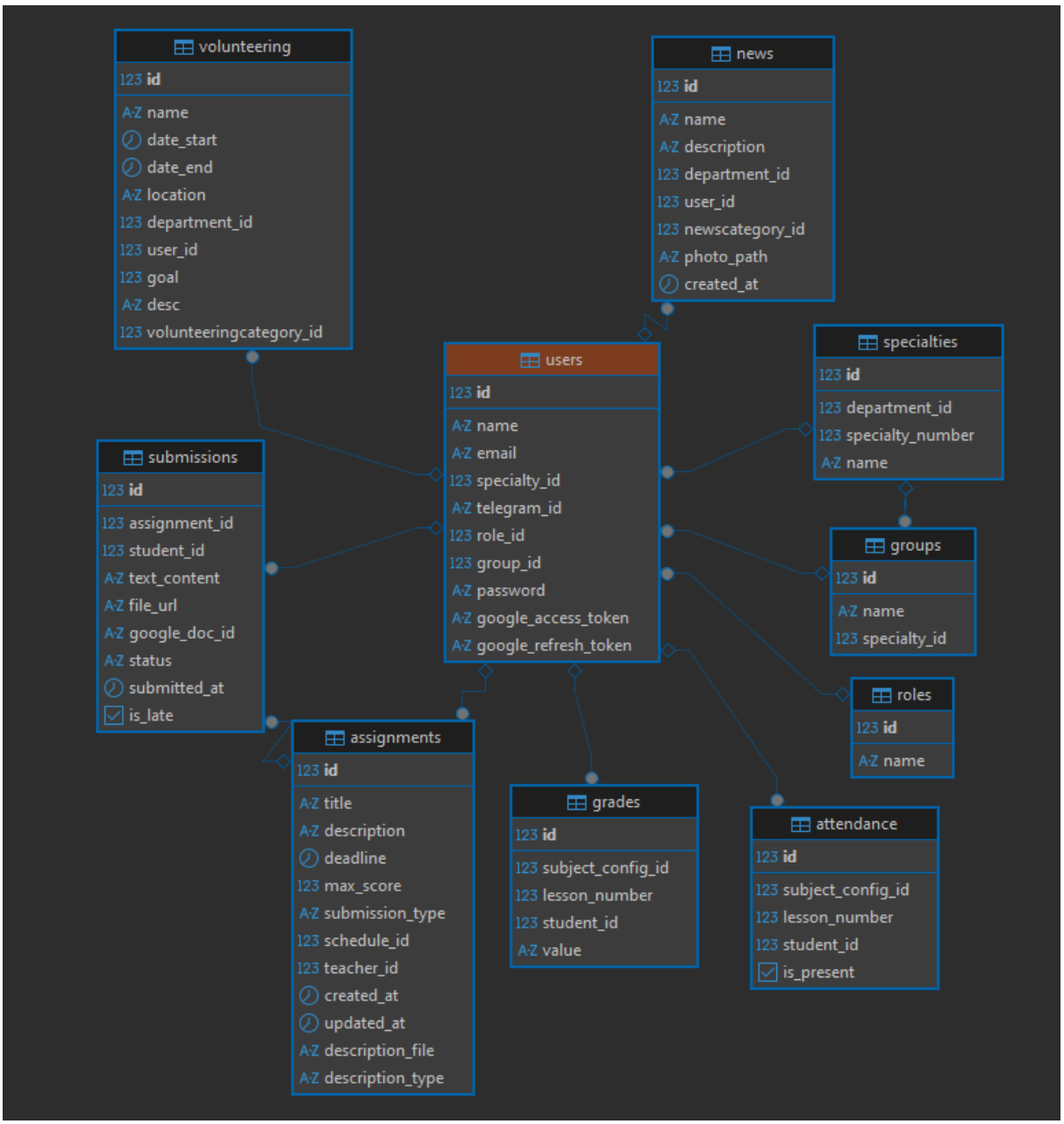

Рис. А.2. Діаграма зв'язків таблиці User.

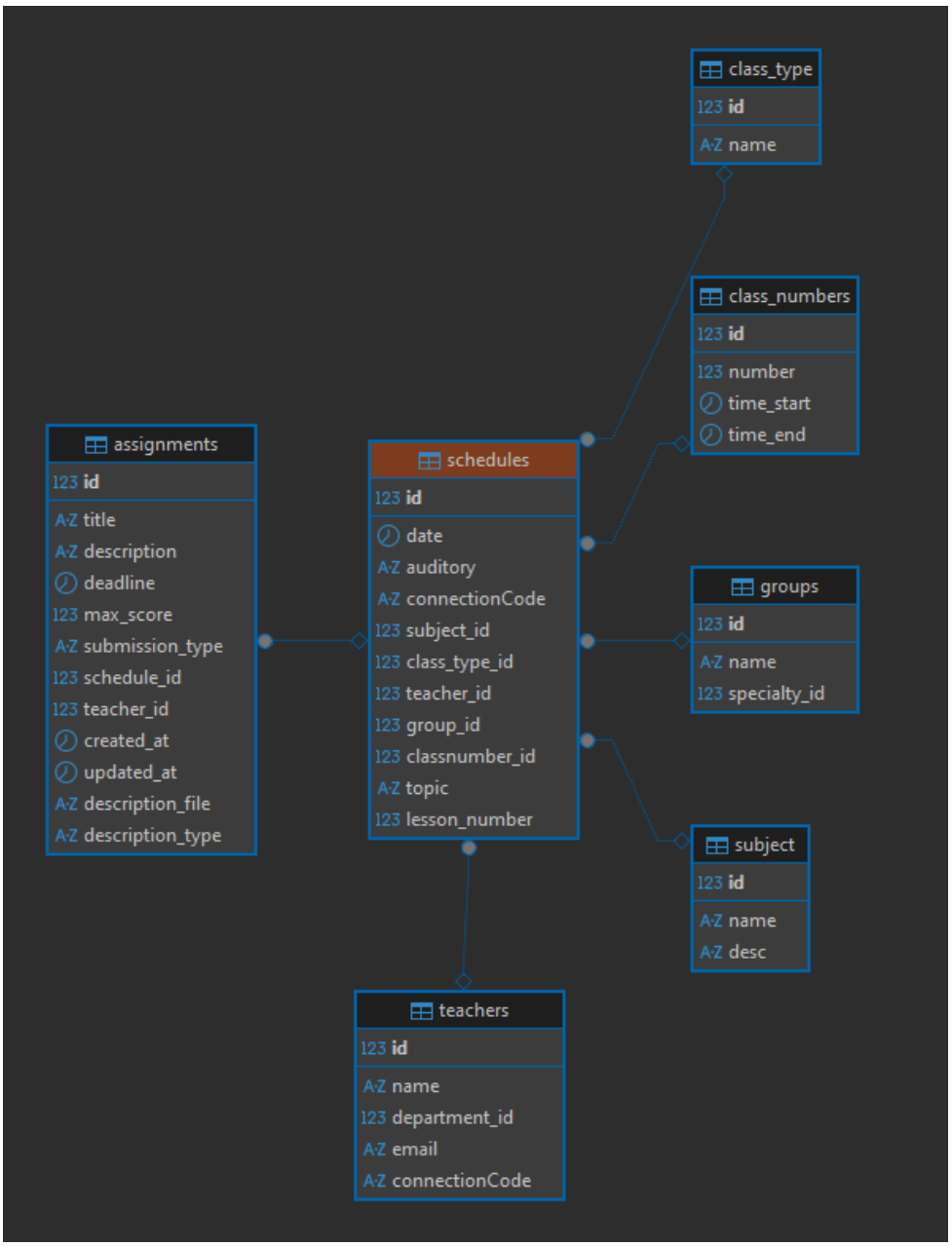

Рис. А.3. Діаграма зв'язків таблиці Schedule.

Додаток Б

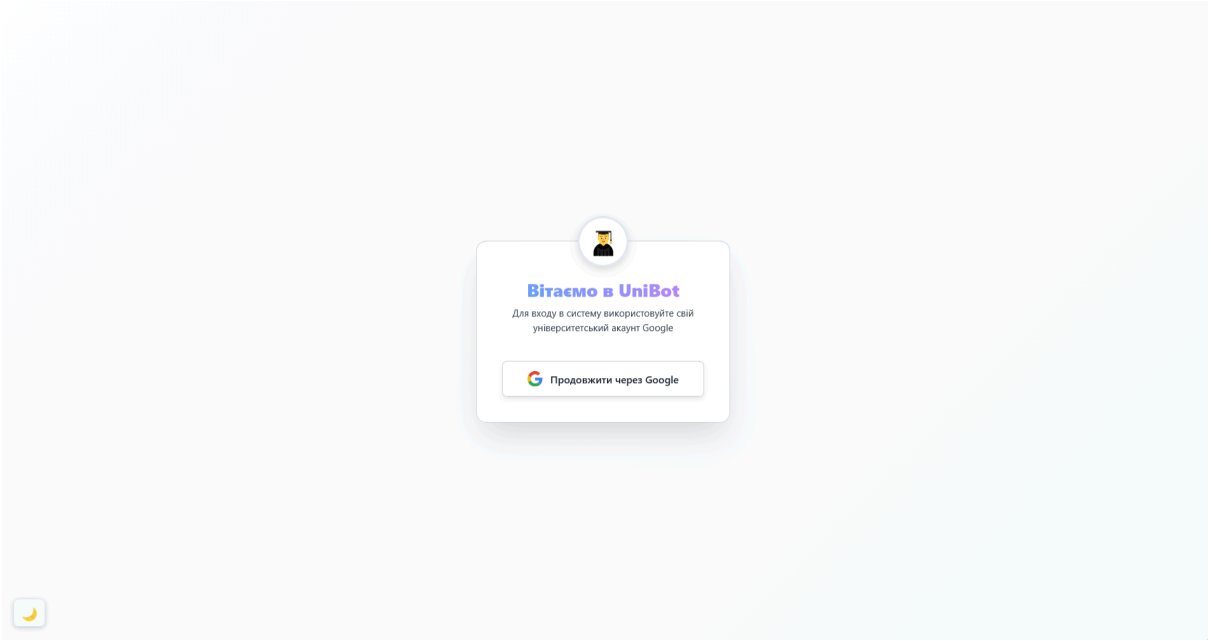

Рис. Б.1. Сторінка входу в систему(Світла тема)

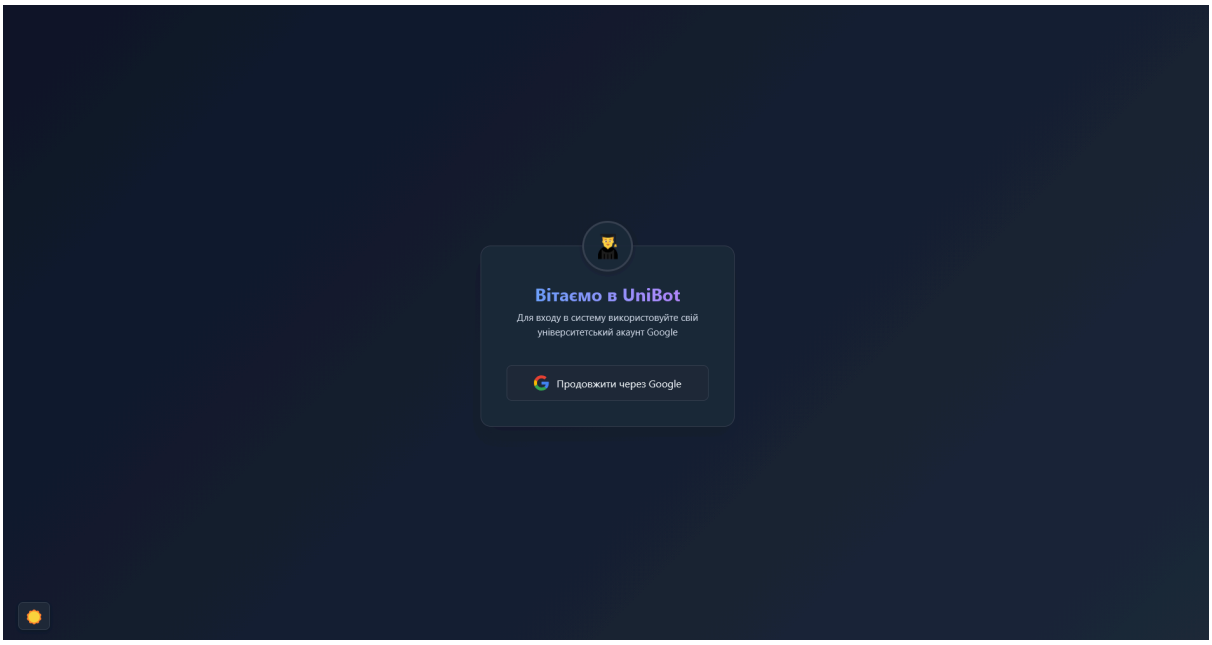

Рис. Б.2. Сторінка входу в систему(Темна тема)

Додаток В

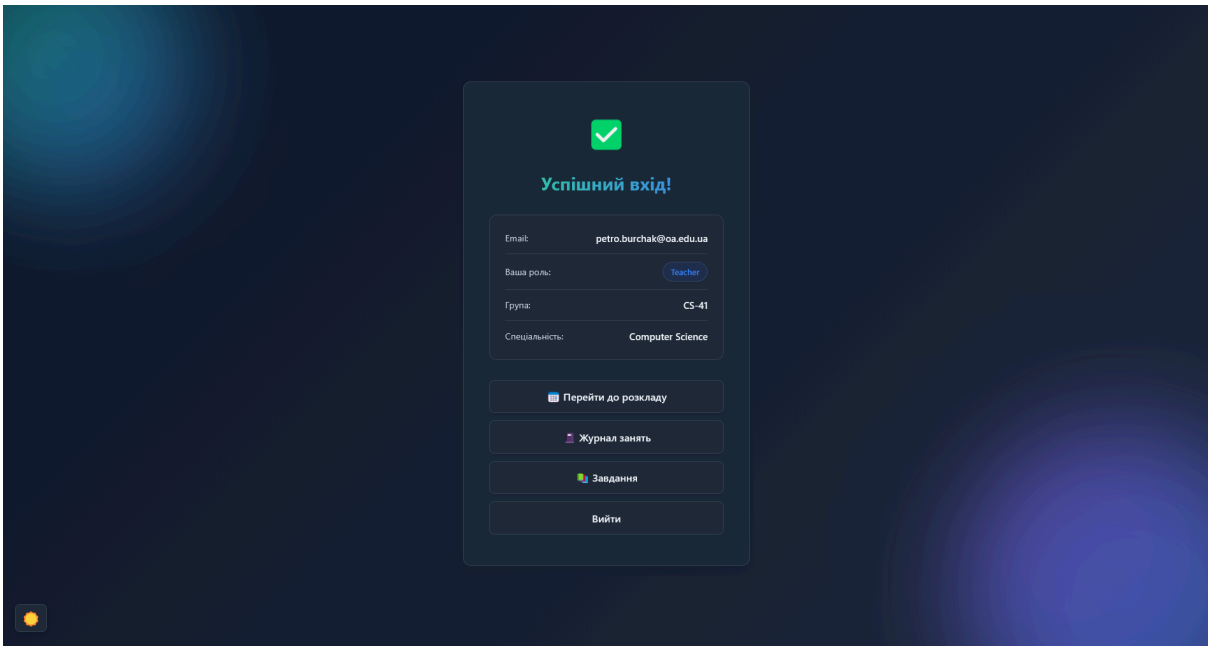

Рис. В.1. Головна сторінка(Роль “Викладач”|Темна тема)

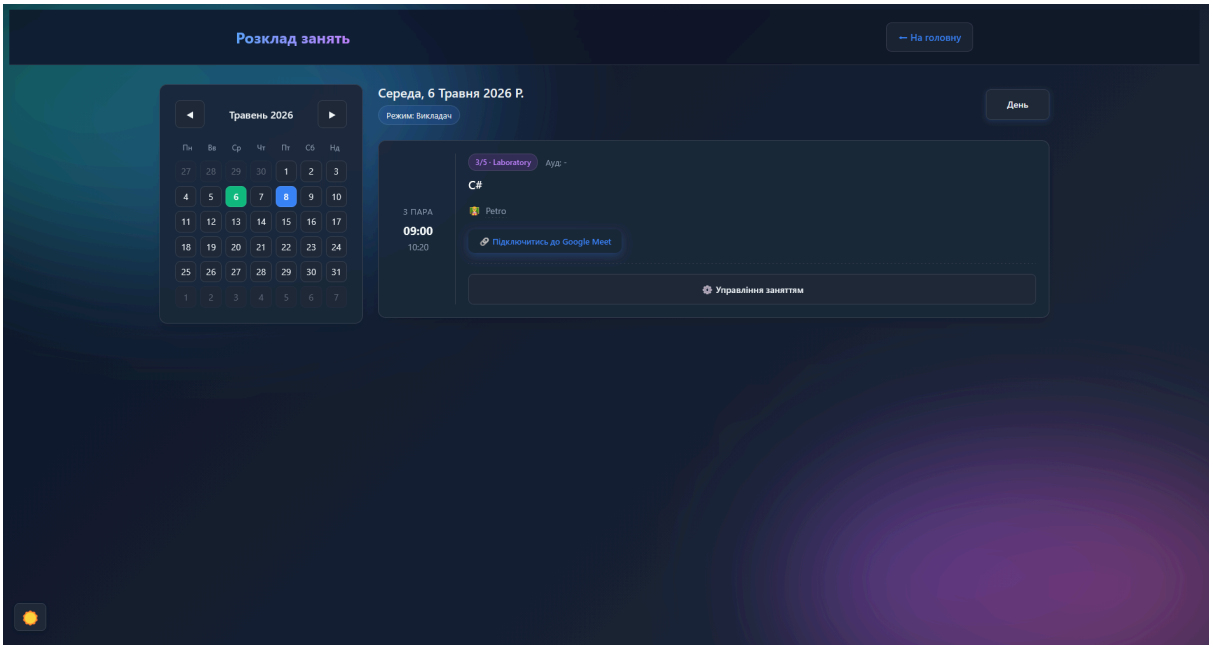

Рис. В.2. Сторінка розкладу системи(Роль “Викладач”|Темна тема)

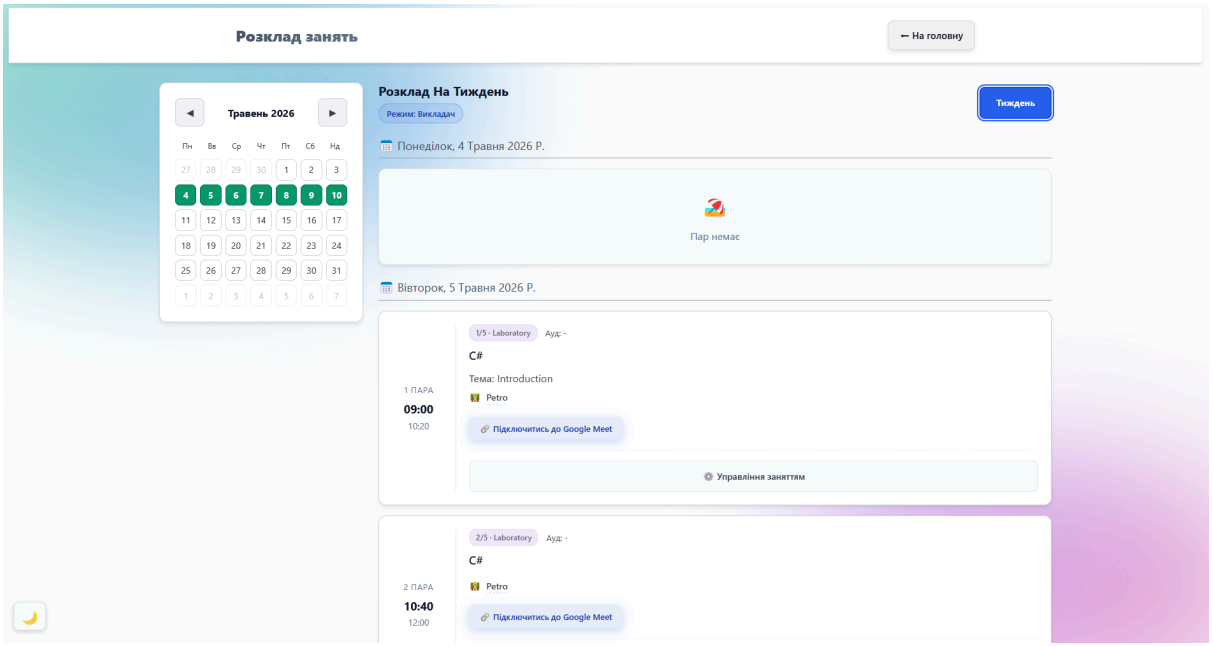

Рис. В.3. Сторінка розкладу системи(Роль “Викладач”|Світла тема)

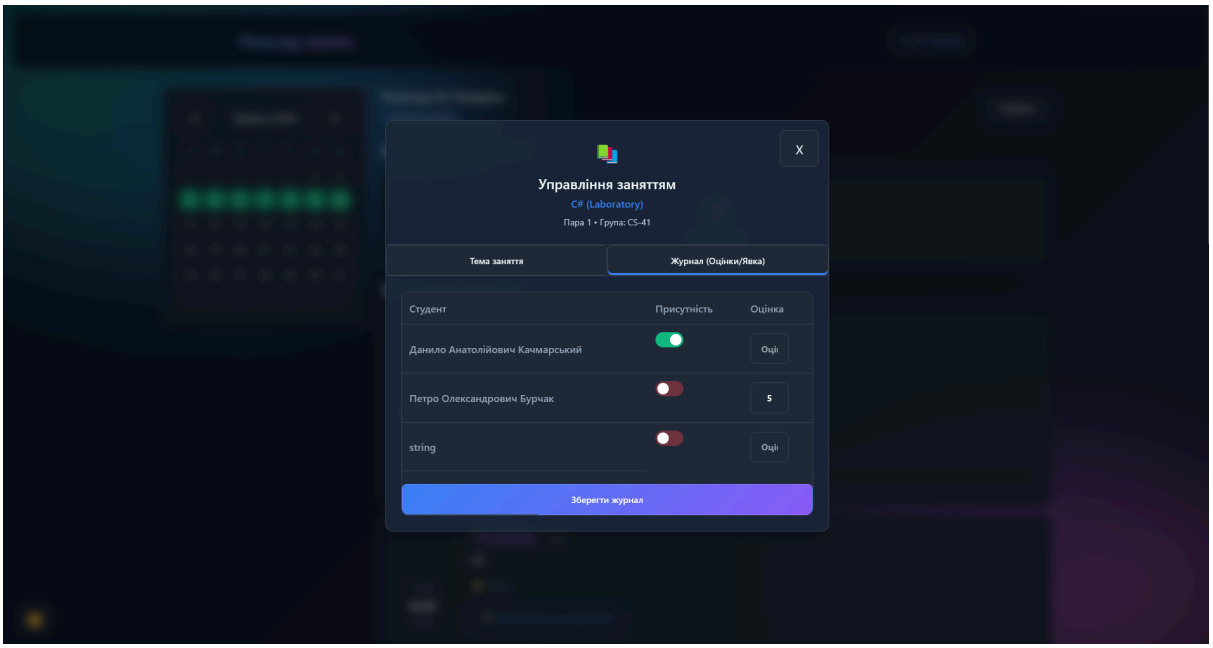

Рис. В.4. Вікно управління заняттям. Секція оцінювання та обліку відвідуваності на сторінці розкладу(Роль “Викладач”|Темна тема)

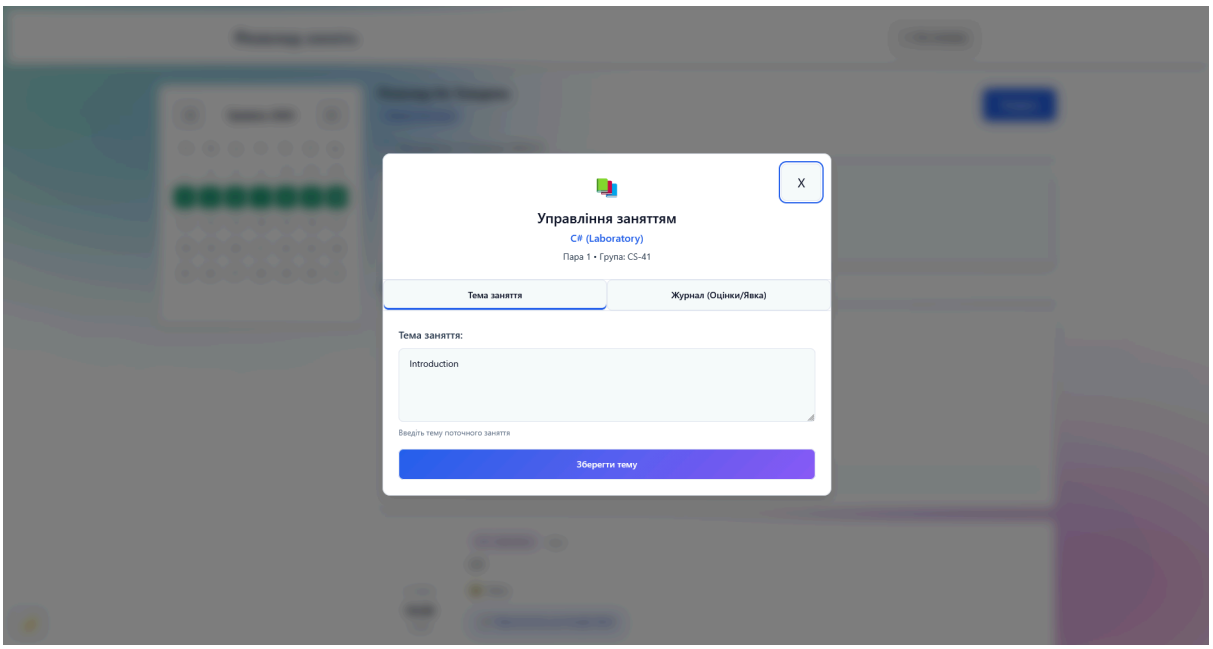

Рис. В.5. Вікно управління заняттям. Секція зміни теми заняття на сторінці розкладу(Роль “Викладач”|Світла тема)

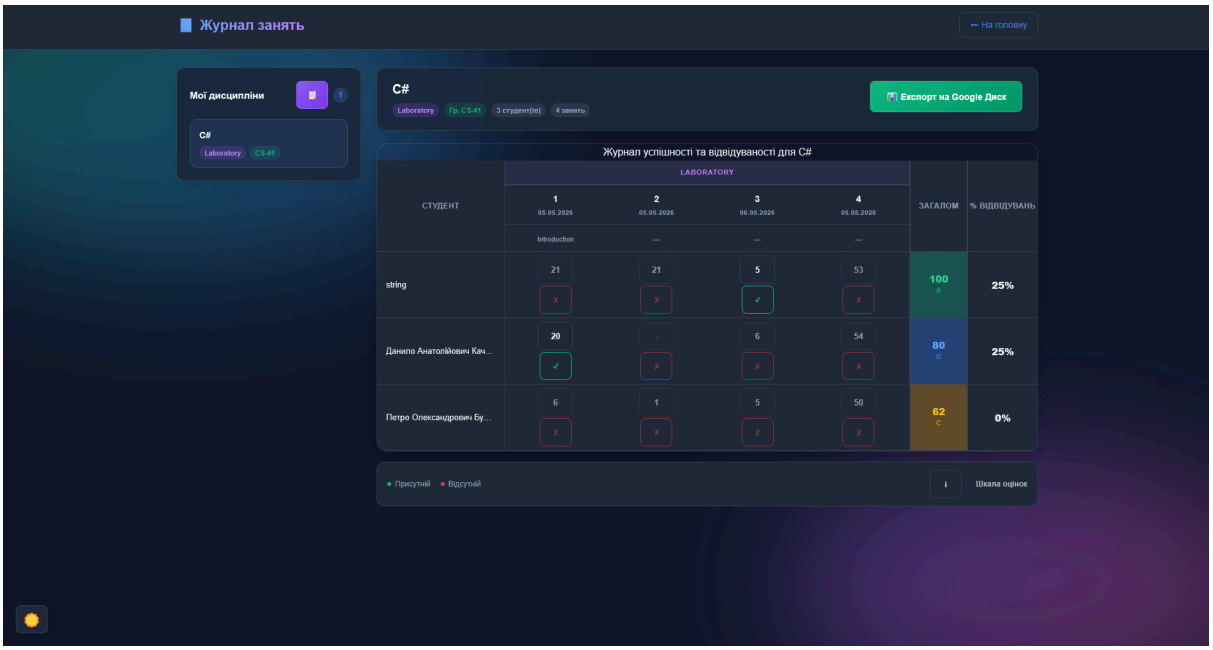

Рис. В.6. Сторінка електронного журналу.(Роль “Викладач”|Темна тема)

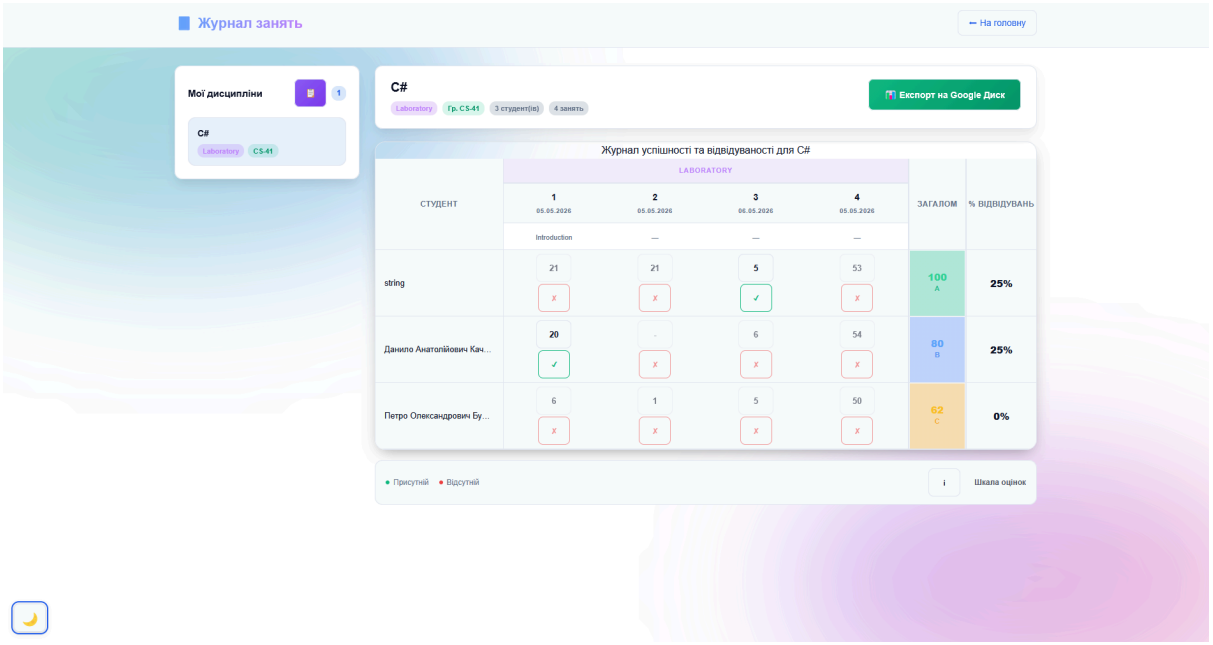

Рис. В.7. Сторінка електронного журналу.(Роль “Викладач”|Світла тема)

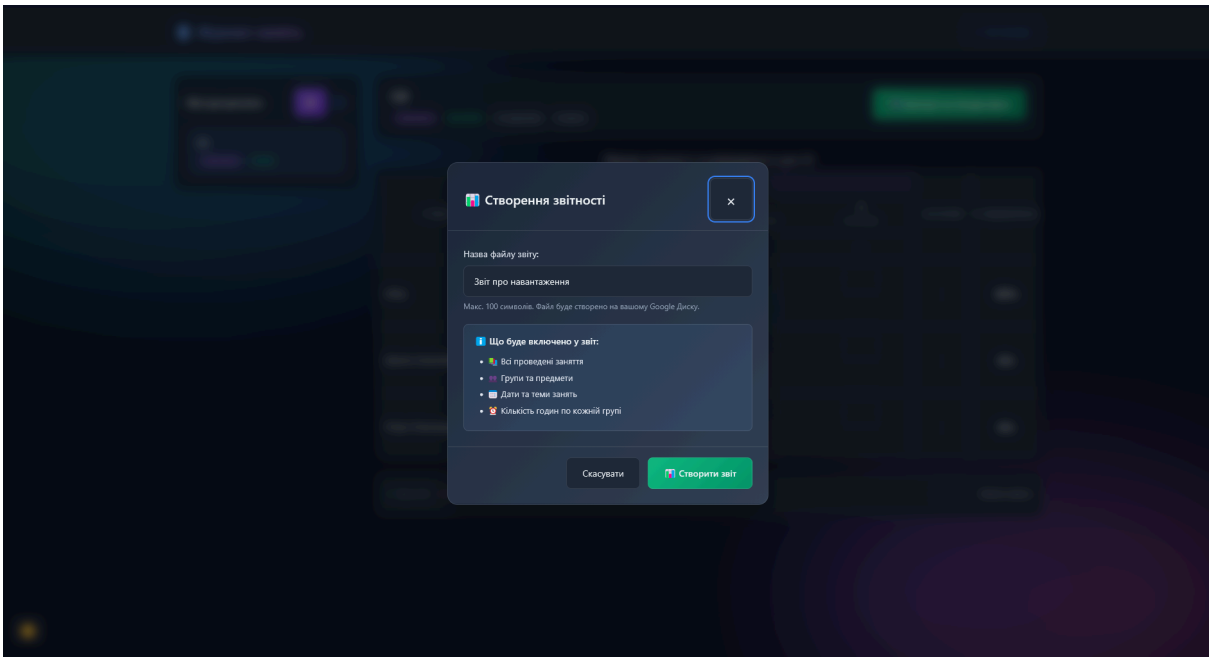

Рис. В.8. Сторінка електронного журналу. Модальне вікно створення документу “Облік педагогічного навантаження” (Роль “Викладач”|Світла тема)

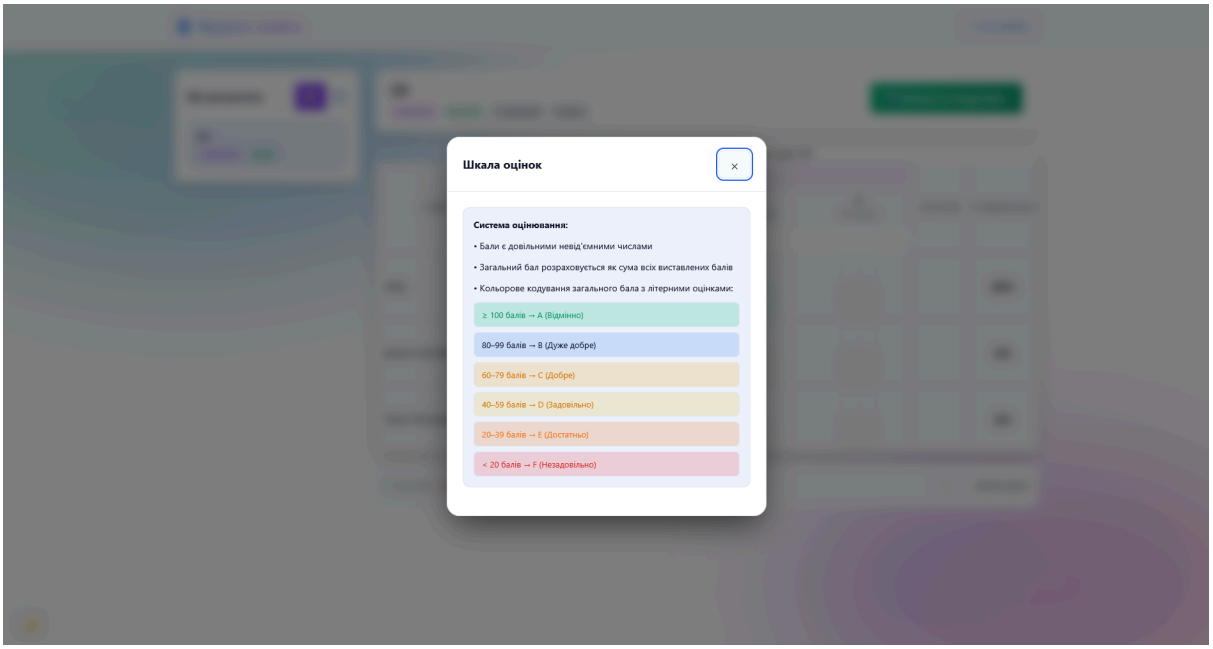

Рис. В.9. Сторінка електронного журналу. Модальне інформаційне вікно про “Шкалу оцінок” (Роль “Викладач”|Світла тема)

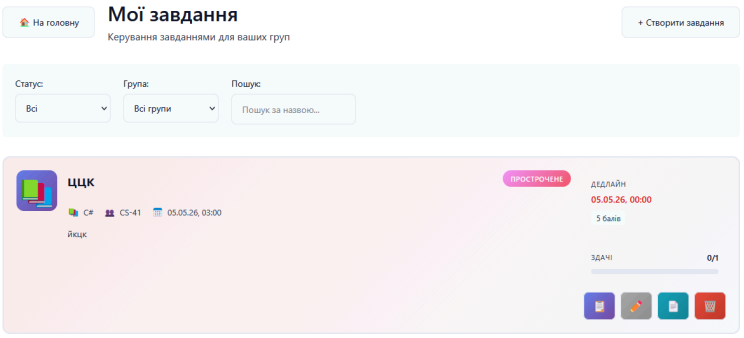

Рис. В.10. Сторінка управління завданнями. (Роль “Викладач”|Світла тема)

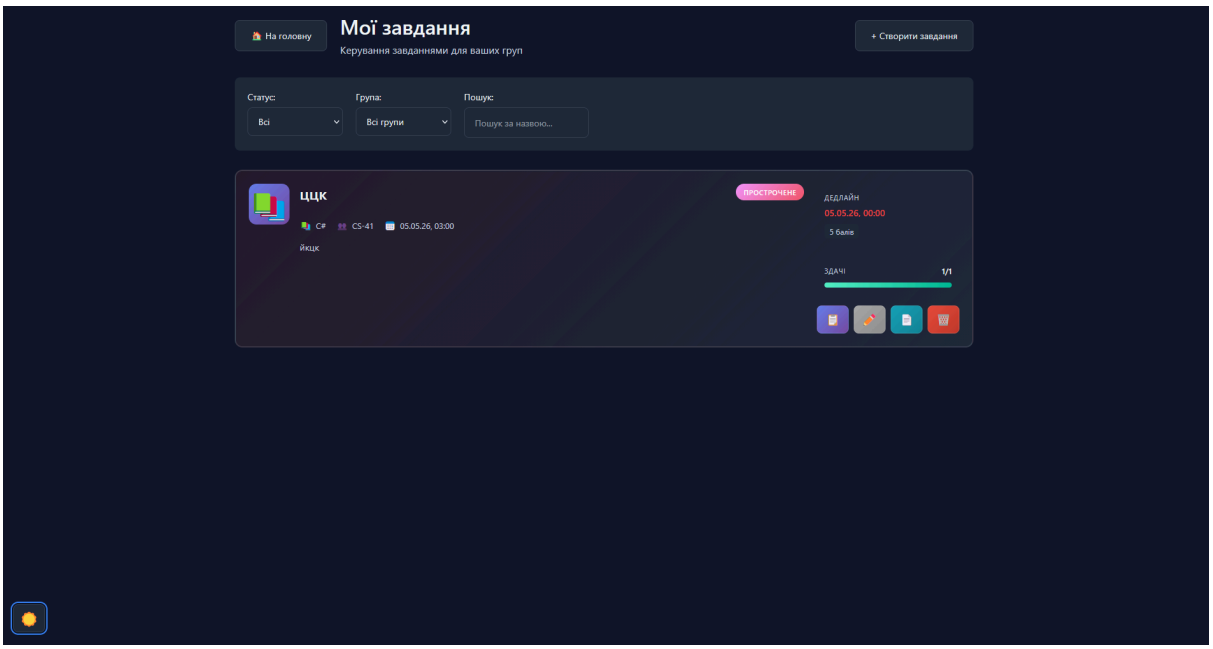

Рис. В.11. Сторінка управління завданнями. (Роль “Викладач”|Темна тема)

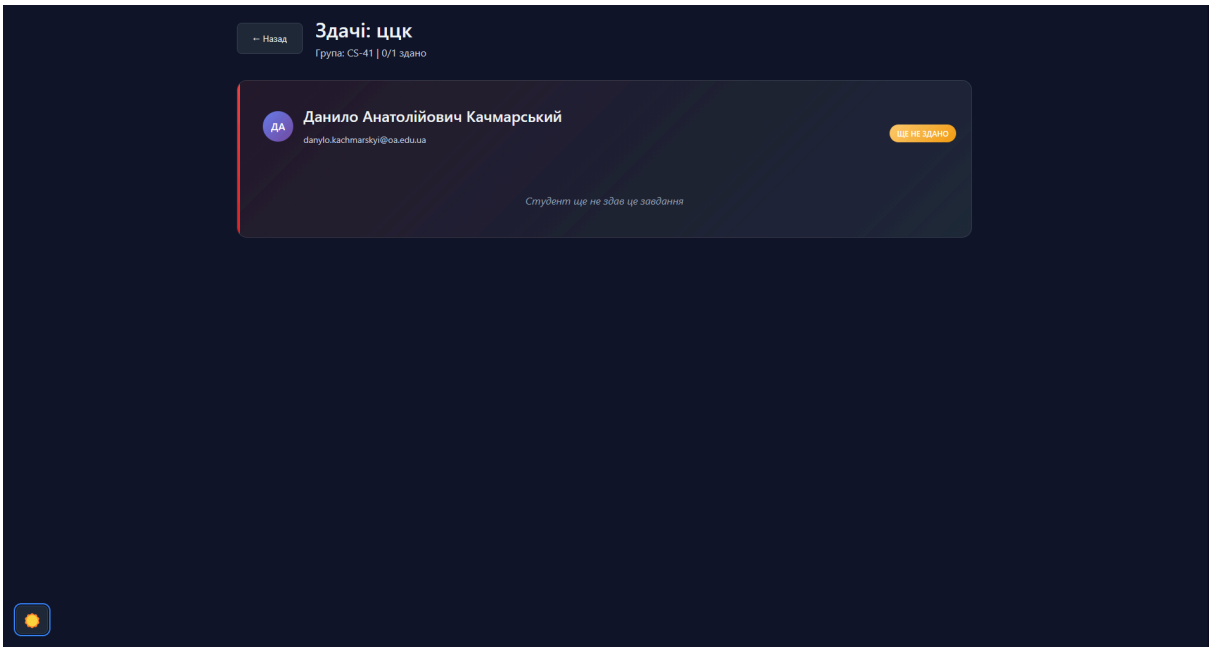

Рис. В. 12. Сторінка здачі завдань студентами. (Роль “Викладач”|Темна тема)

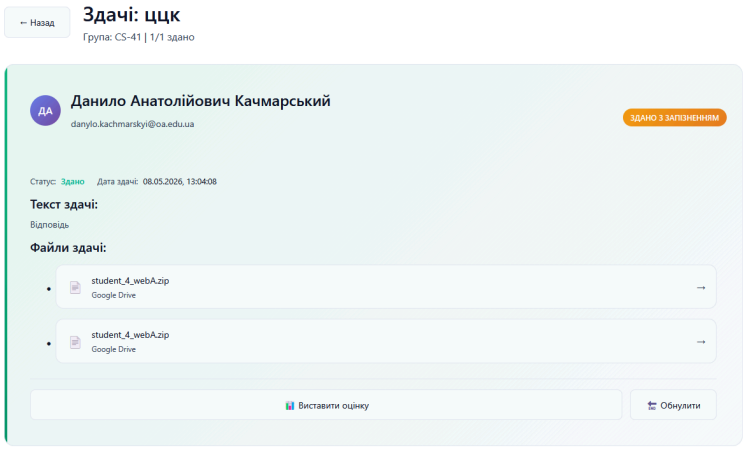

Рис. В. 13. Сторінка здачі завдань студентами. (Роль “Викладач”|Світла тема)

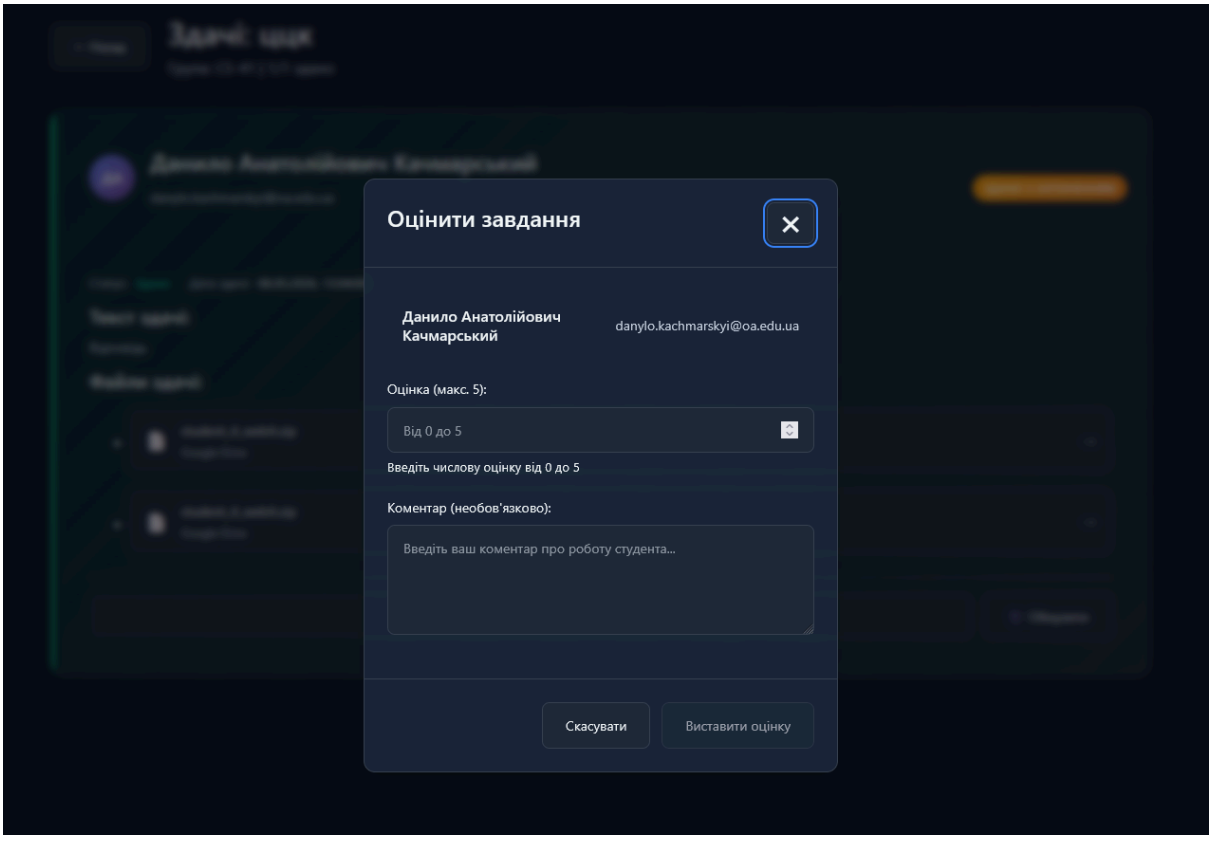

Рис. В. 14. Сторінка здачі завдань студентами. Модальне вікно виставлення оцінки студенту викладачем та додавання опціонального коментаря(Роль “Викладач”|Темна тема)

Додаток Г

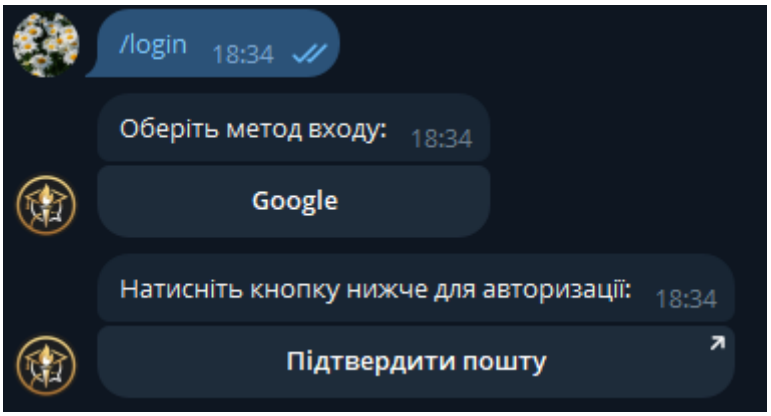

Рис. Г.1. Команди входу в систему через телеграм бота.

Рис. Г.2. Команда розкладу та її результат(справа).Результат конкретної дати в розкладі(зліва).

Додаток Д

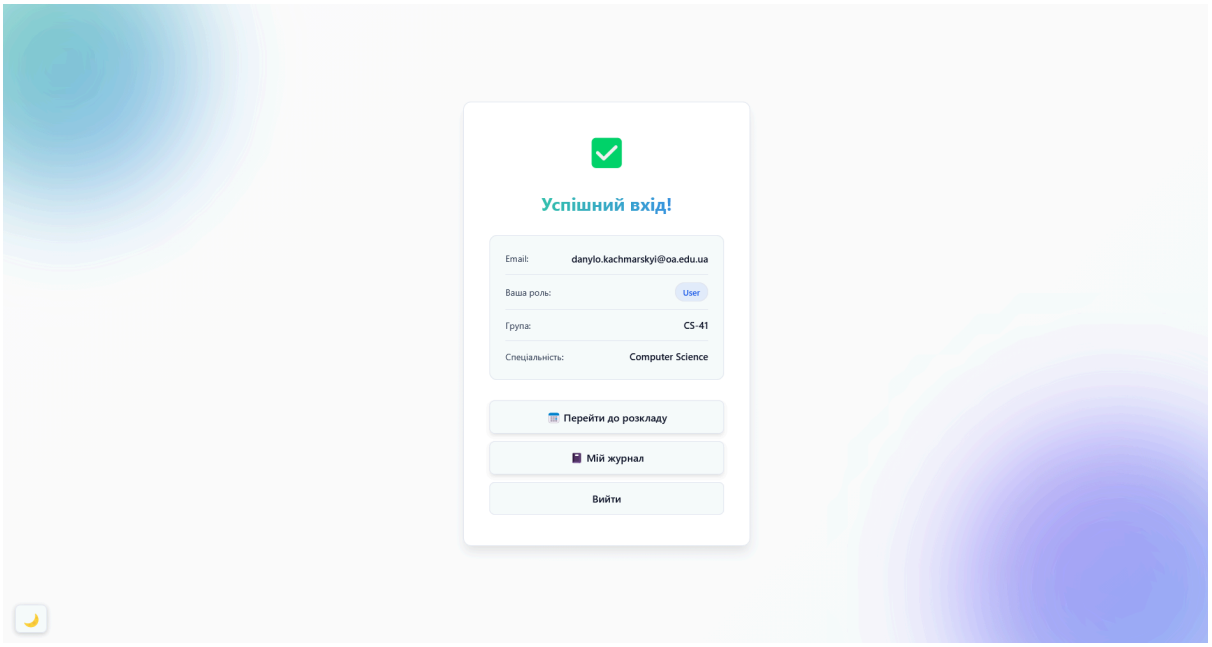

Рис. Д.1. Головна сторінка(Роль “Студент”|Світла тема)

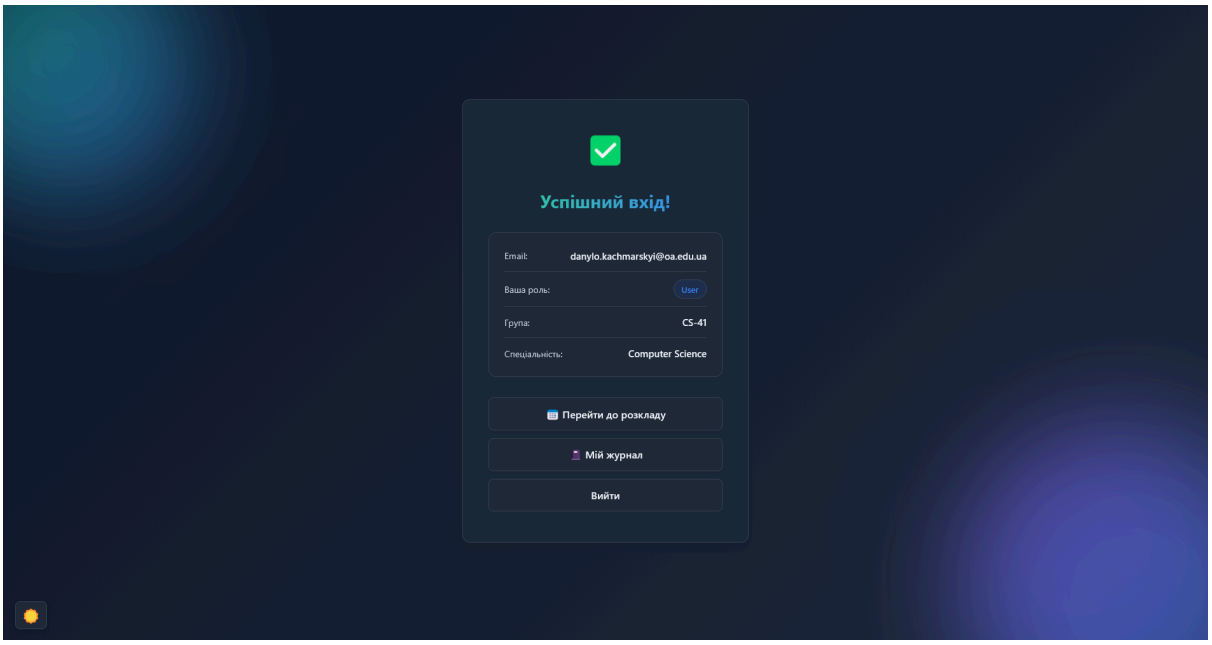

Рис. Д.2. Головна сторінка(Роль “Студент”|Темна тема)

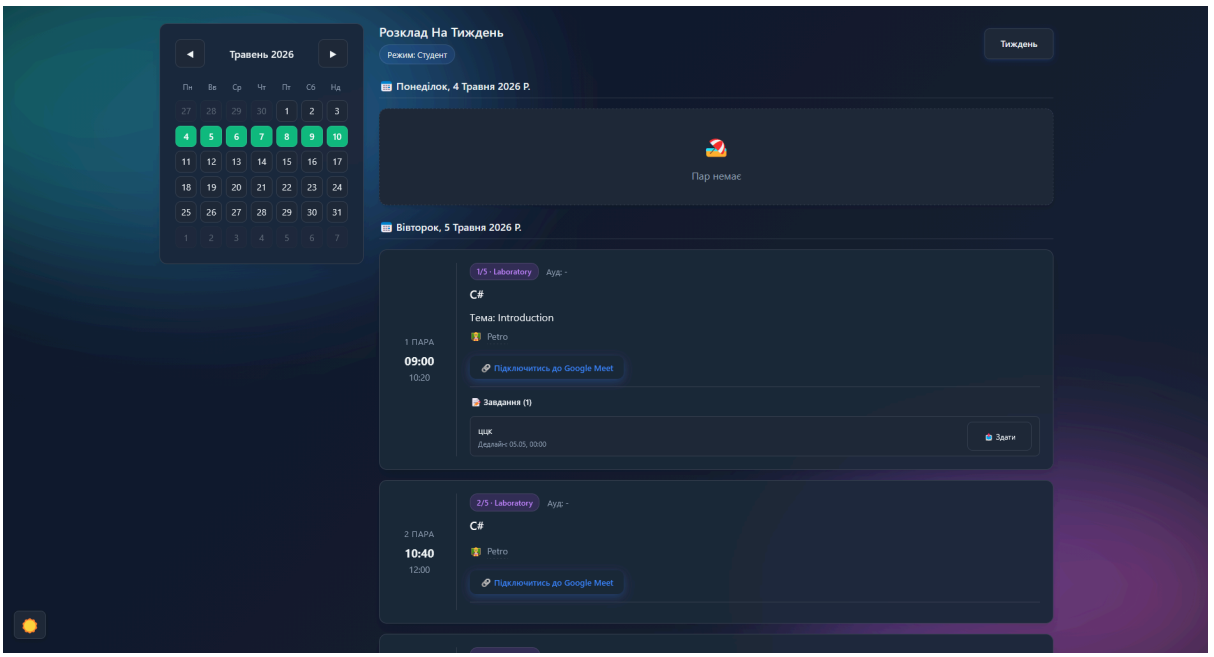

Рис. Д.3. Сторінка розкладу (Роль “Студент”|Темна тема)

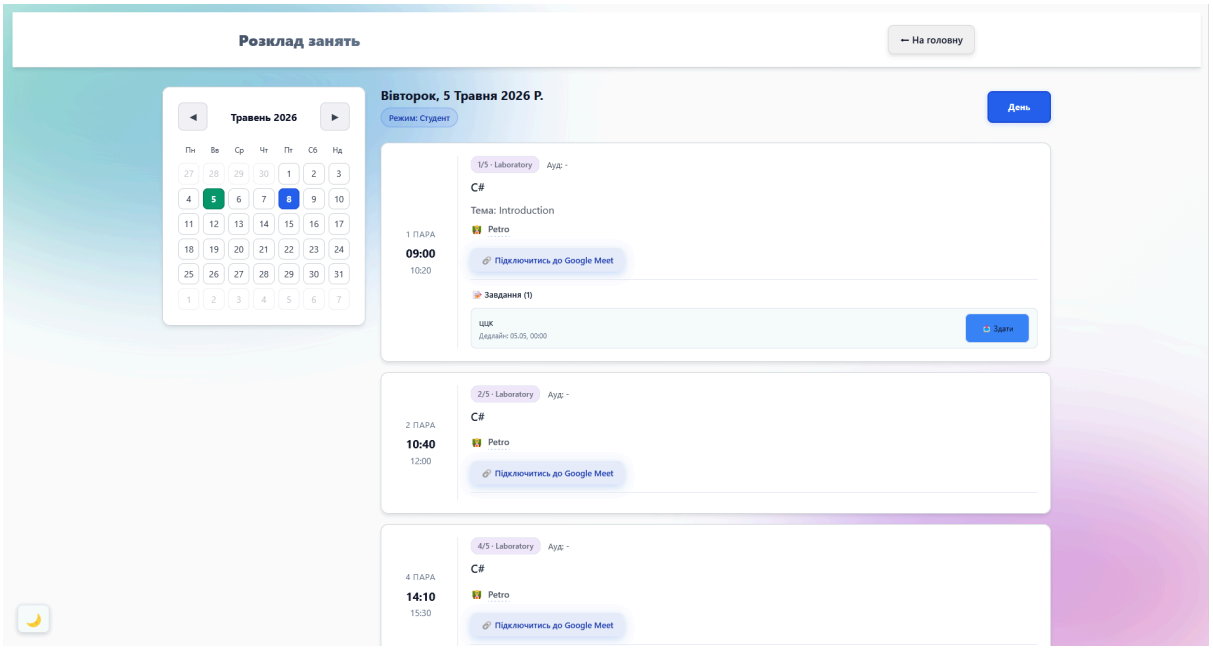

Рис. Д.4. Сторінка розкладу (Роль “Студент”|Світла тема)

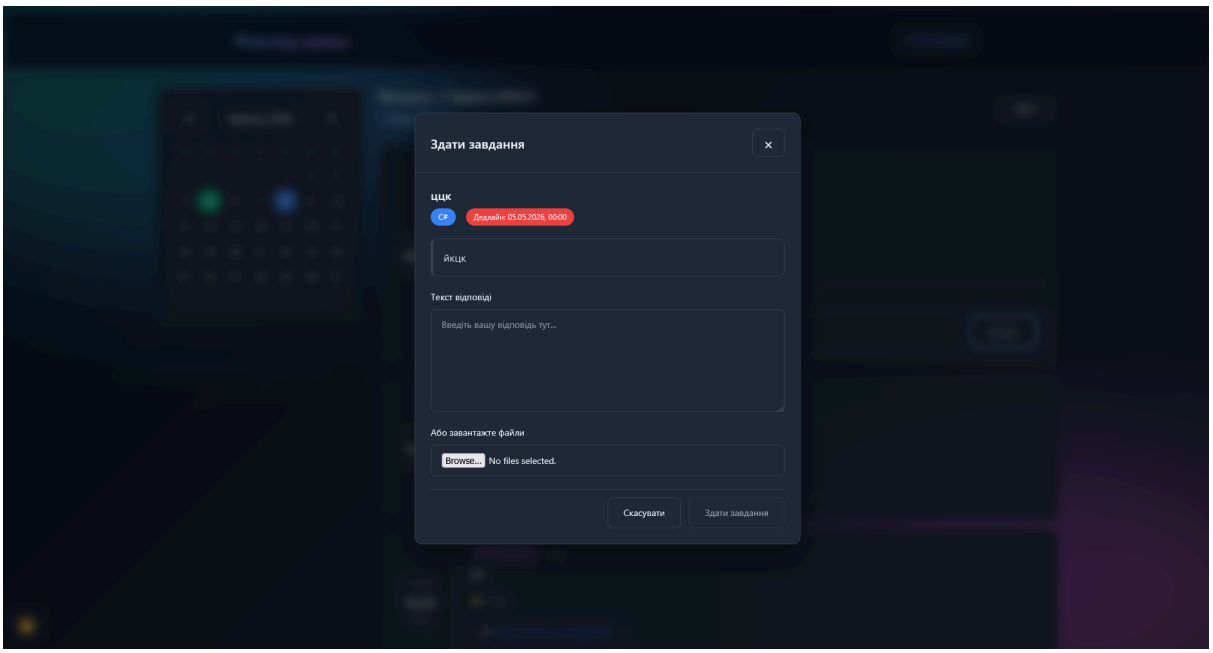

Рис. Д.5. Сторінка розкладу. Модальне вікно “Здати завдання” (Роль “Студент”|Темна тема)

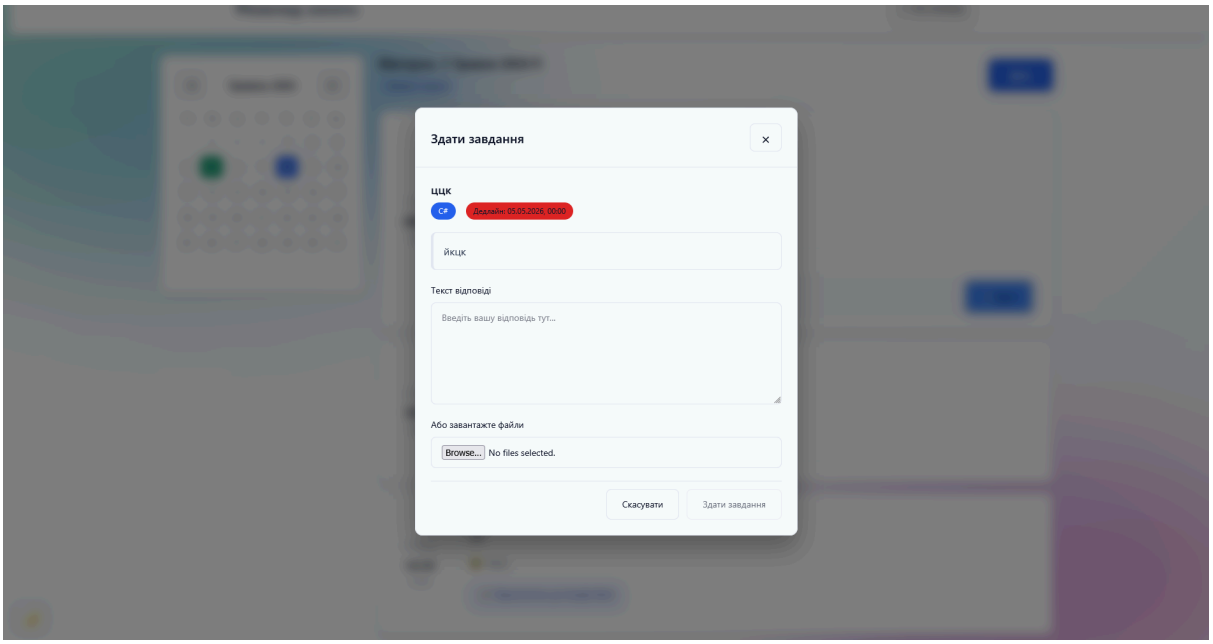

Рис. Д.6. Сторінка розкладу. Модальне вікно “Здати завдання” (Роль “Студент”|Світла тема)

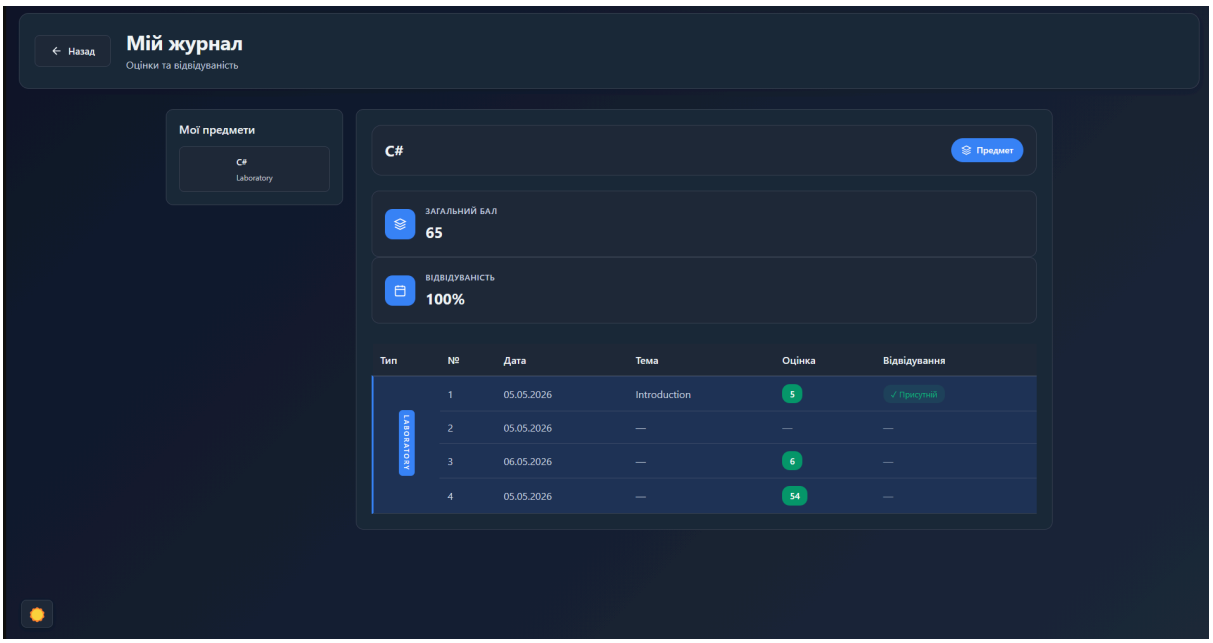

Рис. Д.7. Сторінка мій журнал(Роль “Студент”|Темна тема)

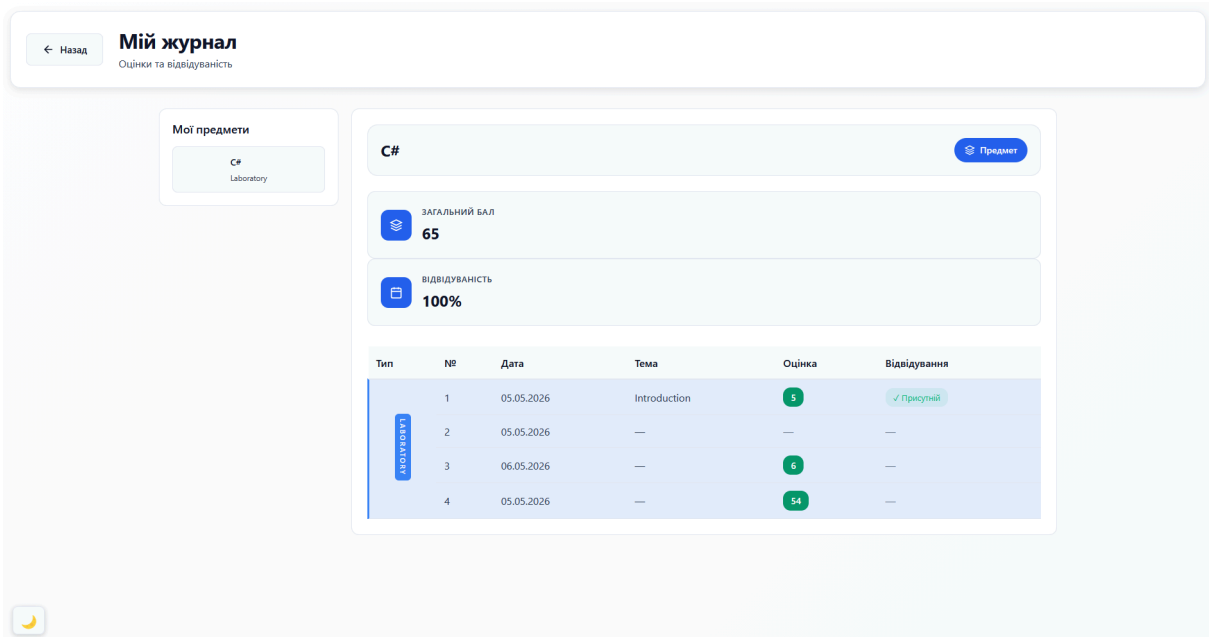

Рис. Д.8. Сторінка мій журнал(Роль “Студент”|Світла тема)

Додаток Е

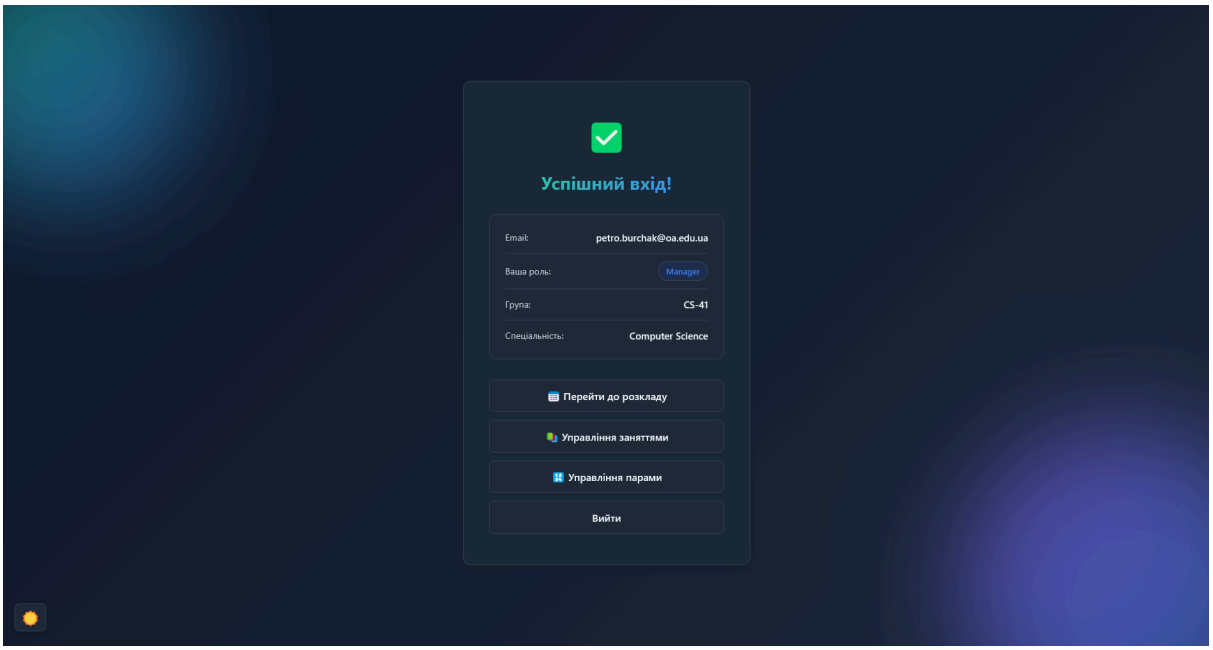

Рис. Е.1. Головна сторінка(Роль “Менеджер”|Темна тема)

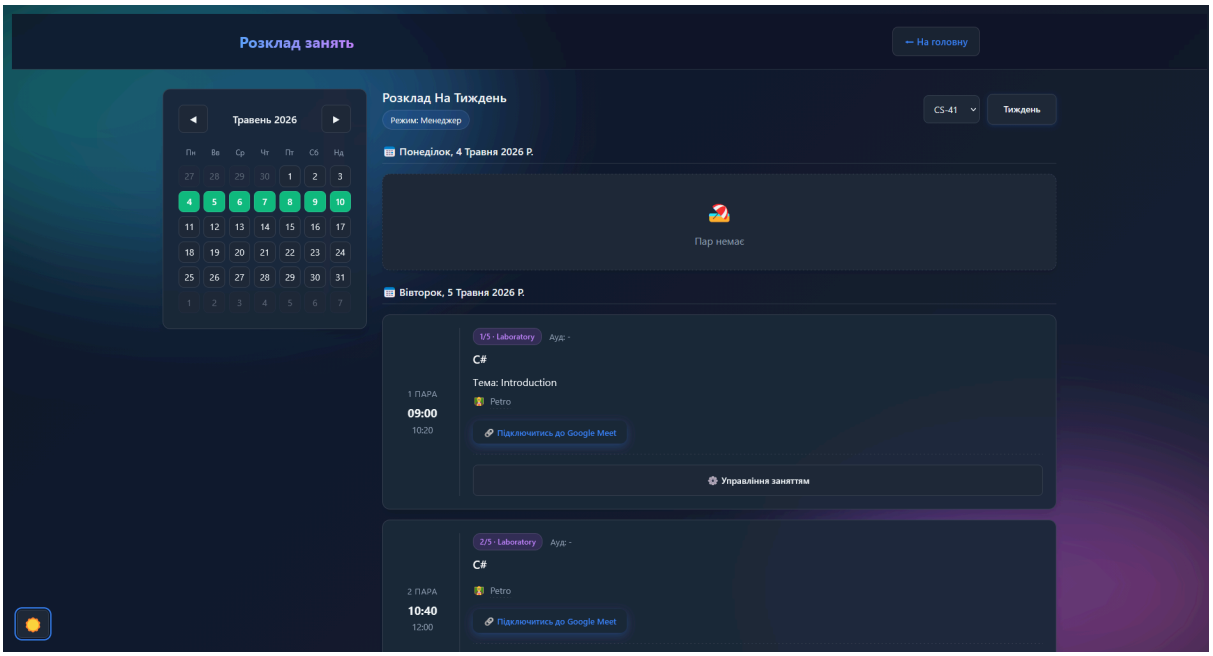

Рис. Е.2. Сторінка розкладу.(Роль “Менеджер”|Темна тема)

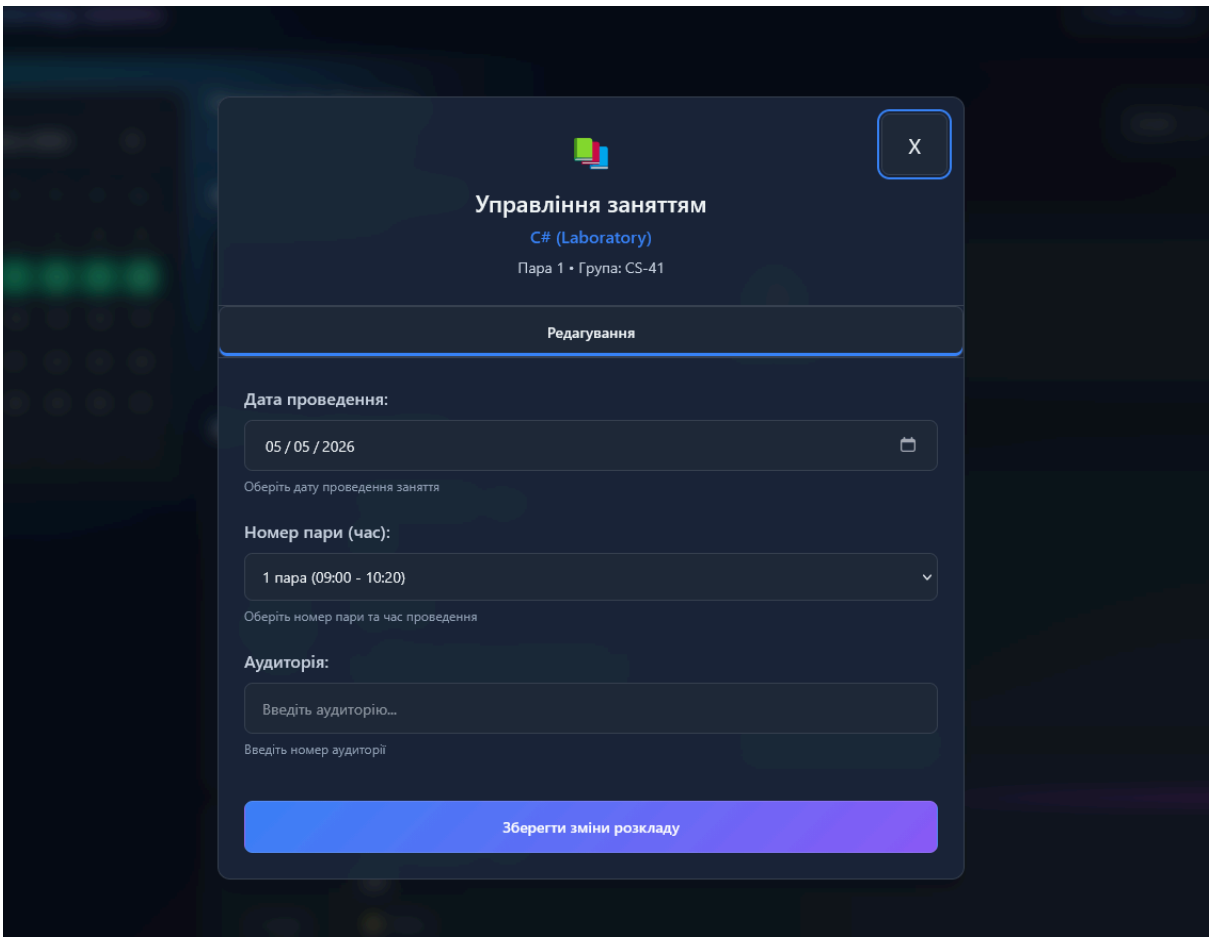

Рис. Е.3. Сторінка розкладу. Вікно редагування конкретного заняття(Роль “Менеджер”|Темна тема)

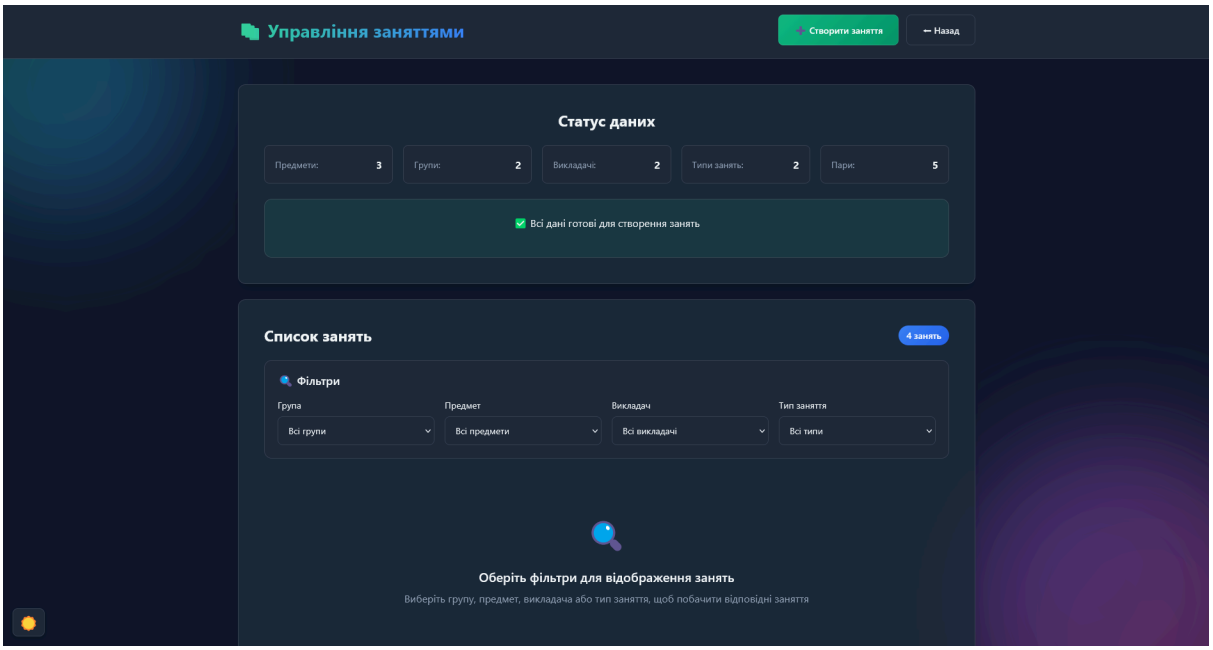

Рис. Е.4. Сторінка управління заняттям.(Роль “Менеджер”|Темна тема)

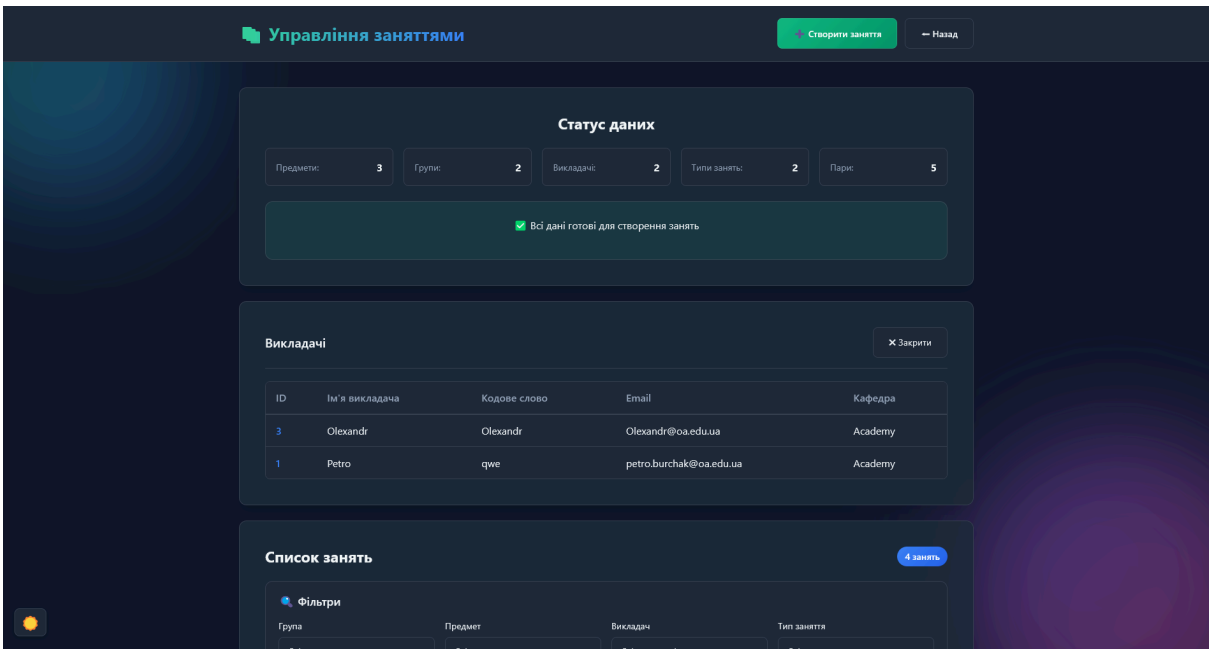

Рис. Е.5. Сторінка управління заняттям.З розгорнутою таблицею деталей(Роль “Менеджер”|Темна тема)

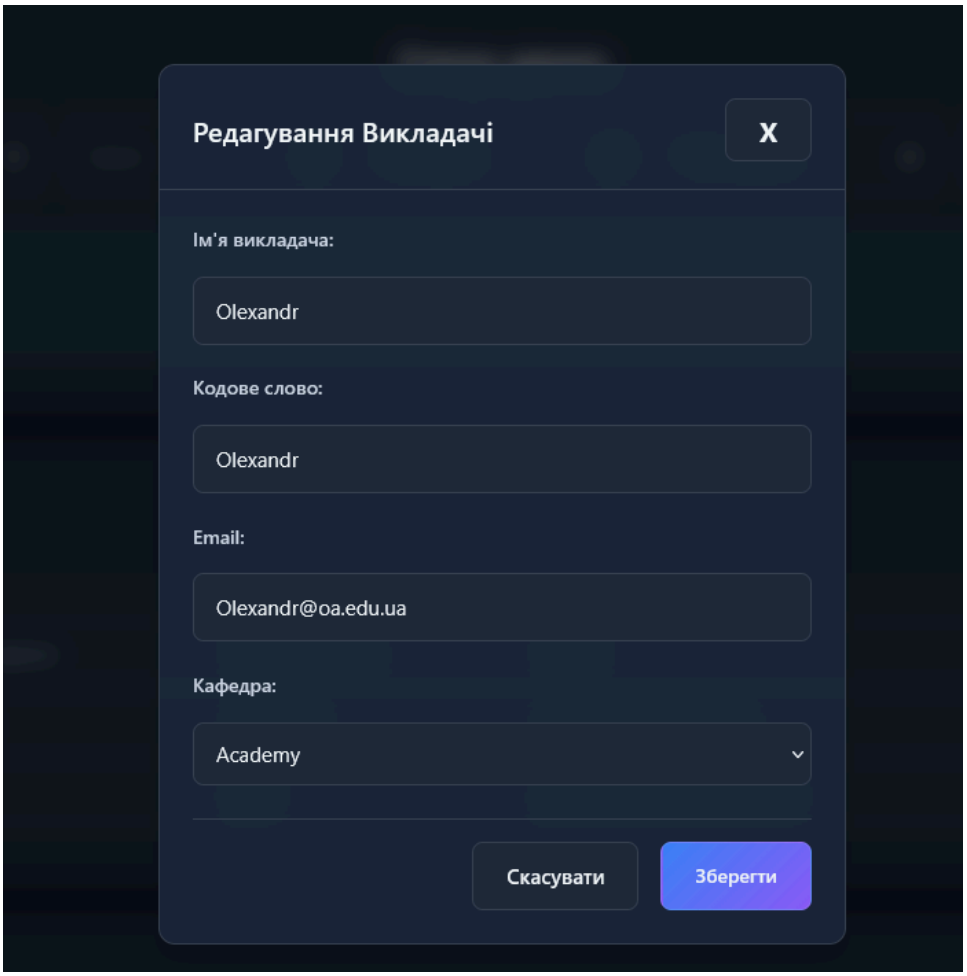

Рис. Е.6. Сторінка управління заняттям.З розгорнутою таблицею деталей. Вікно редагування даних.(Роль “Менеджер”|Темна тема)

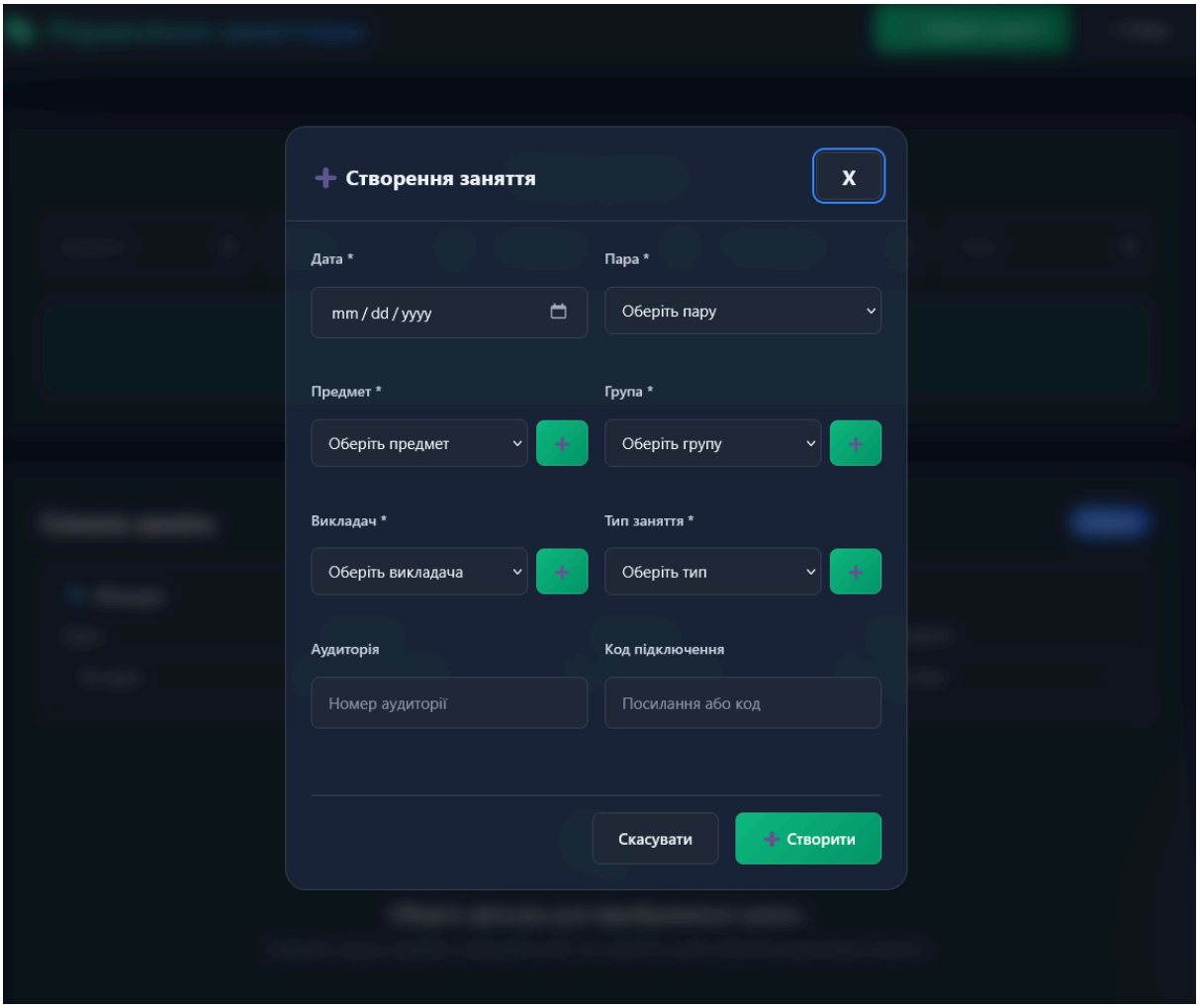

Рис. Е.7. Сторінка управління заняттям. Вікно створення нового заняття(Роль “Менеджер”|Темна тема)

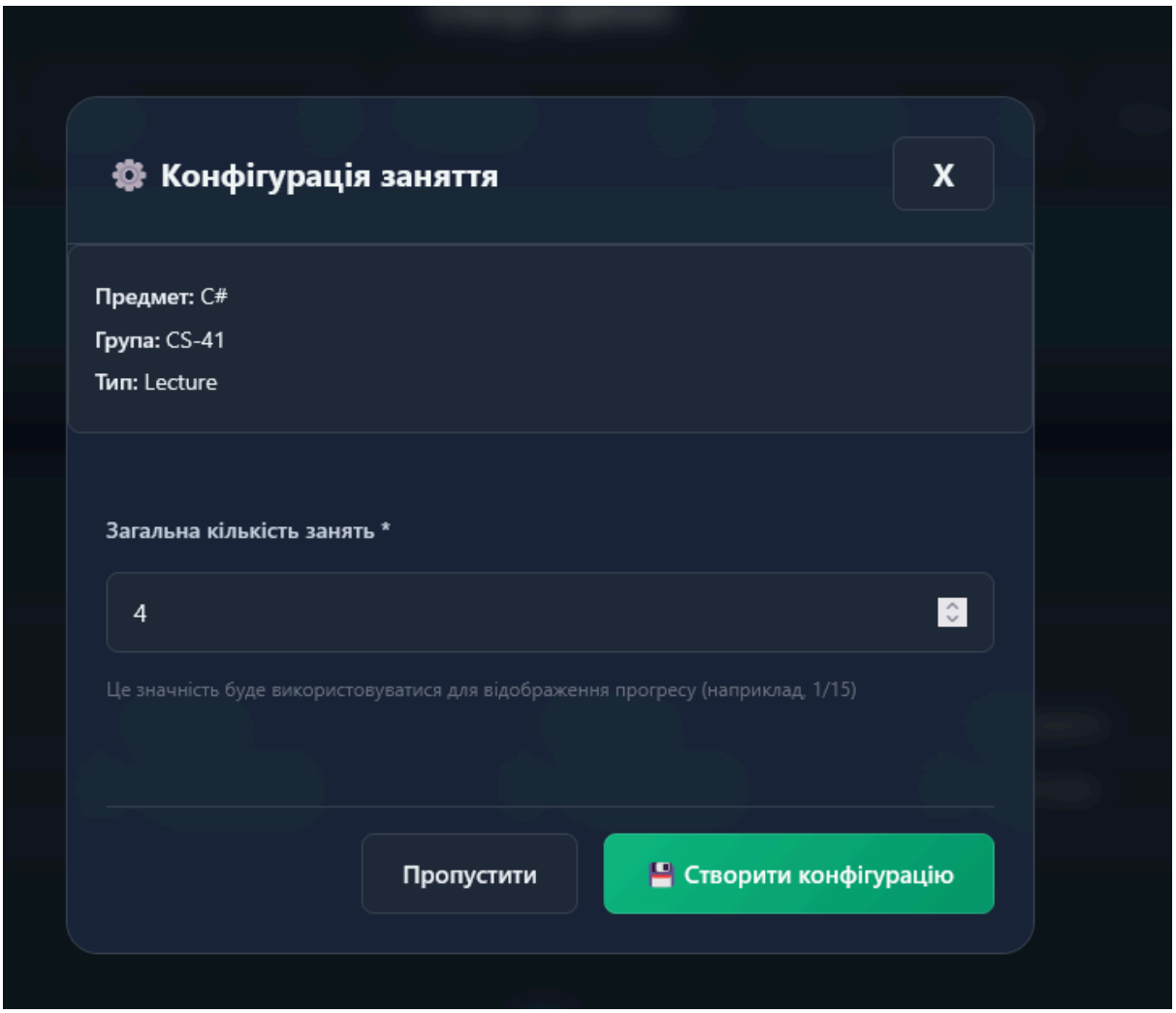

Рис. Е.8. Сторінка управління заняттям. Вікно створення конфігурації для нового(Роль “Менеджер”|Темна тема)

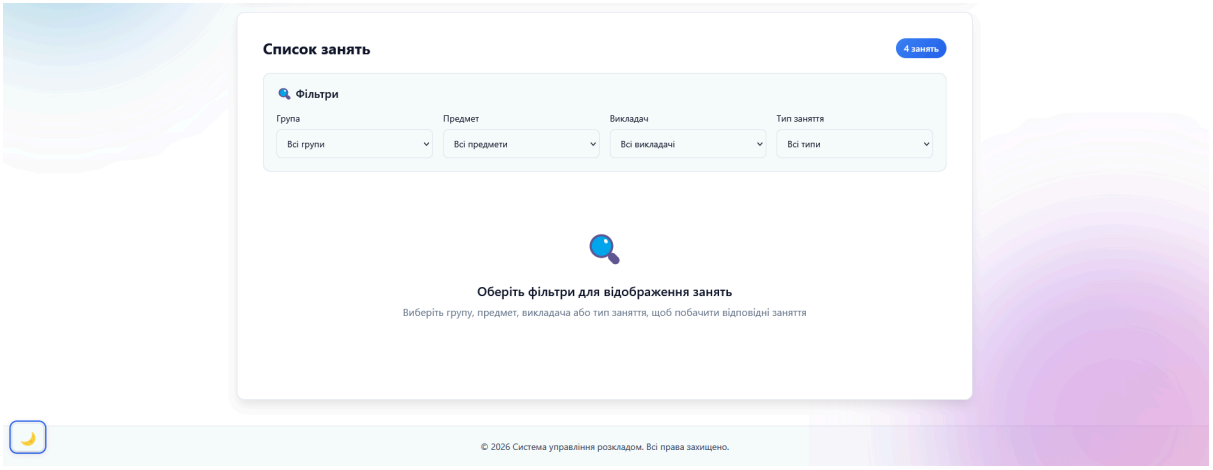

Рис. Е.9. Сторінка управління заняттям. Таблиця загального списку занять без фільтрів.(Роль “Менеджер”|Темна тема)

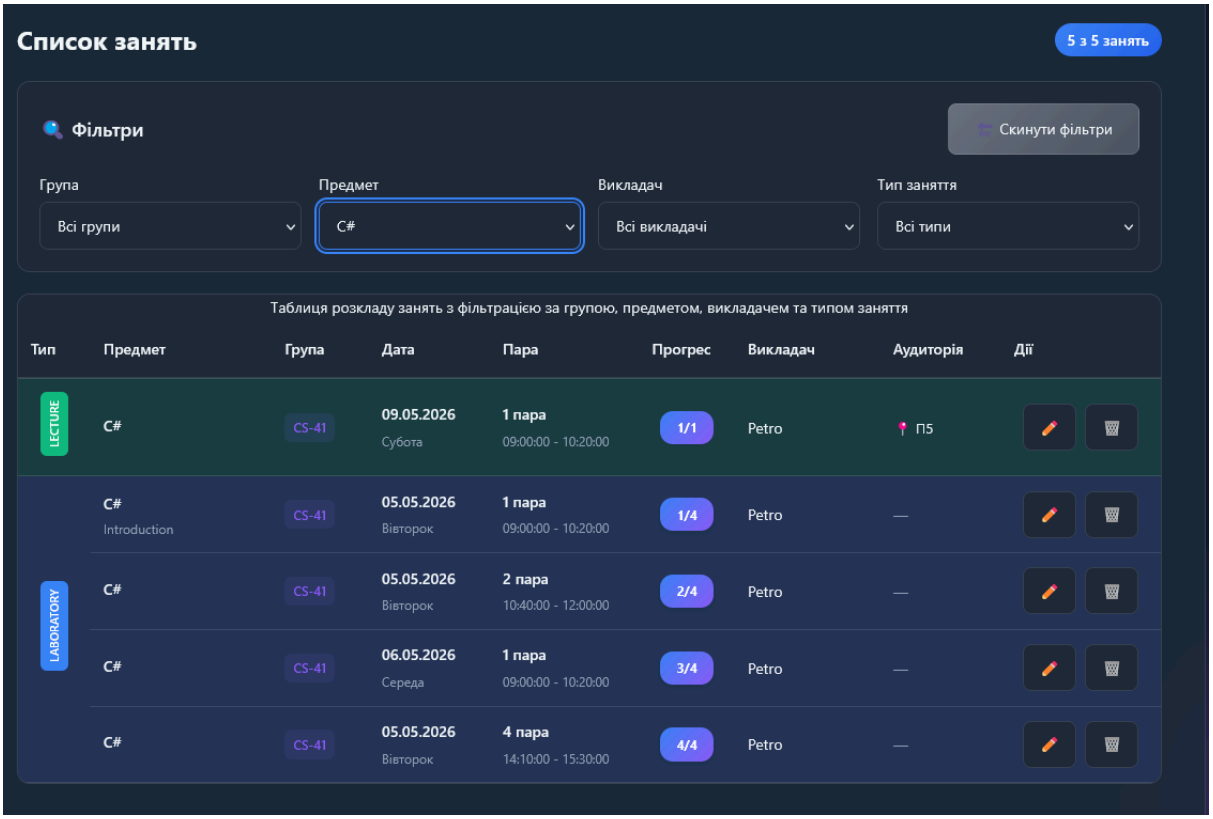

Рис. Е.10. Сторінка управління заняттям. Таблиця загального списку занять після застосуванняфільтрів.(Роль “Менеджер”|Темна тема)

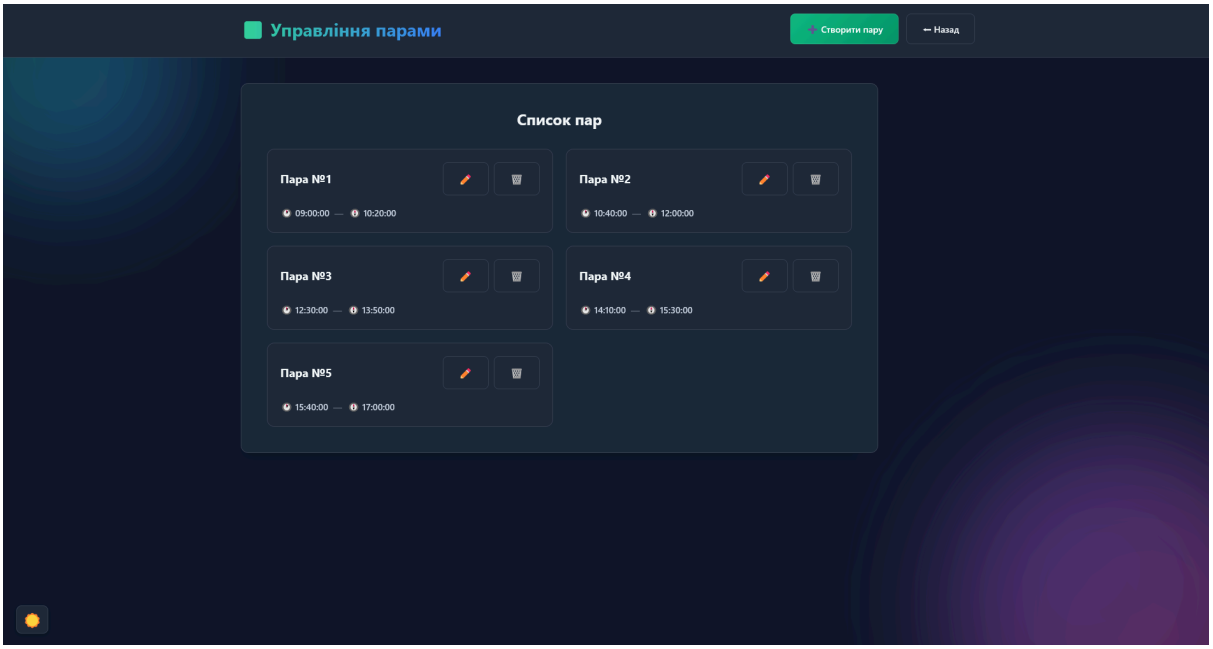

Рис. Е.11. Сторінка управління парами.(Роль “Менеджер”|Темна тема)