

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Веб-орієнтована система для планування навчальних
занять»**

Виконав студент групи ІПЗм-22-1
ЗАХАРОВ Дмитро Миколайович

Керівник роботи:
ЛОБАНЧИКОВА Надія Миколаївна

Рецензент:
ВАКАЛЮК Тетяна Анатоліївна

Житомир – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
 ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
 ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
 КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

зав. кафедри інженерії
 програмного забезпечення
 Тетяна ВАКАЛЮК
 «18» вересня 2023 р.

ЗАВДАННЯ на кваліфікаційну роботу

Здобувач вищої освіти: **Захаров Дмитро Миколайович**

Керівник роботи: **Лобанчикова Надія Миколаївна**

Тема роботи: **«Веб-орієнтована система для планування навчальних занять»**
 затверджена Наказом закладу вищої освіти від **«18» вересня 2023 р. № 481/с**

Вихідні дані для роботи: **Предметом дослідження є особливості проектування та обробки даних для підготовки та проведення навчального процесу. Методи розробки веб-систем для організації навчального процесу. Об'єктом дослідження є навчальний процес, його структура та підходи щодо організації.**

Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Лобанчикова Н.М.	22.09.2023р.	22.09.2023р.
2	Лобанчикова Н.М.	22.09.2023р.	22.09.2023р.
3	Лобанчикова Н.М.	22.09.2023р.	22.09.2023р.

РЕФЕРАТ

Кваліфікаційна робота магістра являє собою розроблену веб-орієнтовану систему для планування і організації процесу навчання, а також супроводжуючу документацію у формі пояснювальної записки. Пояснювальна записка має обсяг 81 сторінка. В якості ілюстративного матеріалу використано 22 рисунки, 4 лістинги коду, 15 таблиць, містить 40 використаних літературних джерел.

Мета роботи: є розробка веб-орієнтованої системи для планування навчальних занять для підвищення якості надання освітніх послуг та провадження освітньої діяльності.

В роботі визначено потреби користувачів системи (студенти, викладачі, адміністратори), здійснено збір та аналіз вимог до системи. В ході проектування системи розроблено архітектуру та схему бази даних, визначено вимоги щодо функціональності та графічного веб-інтерфейсу.

Здійснено розробку програмного забезпечення. Реалізовано функціональність для реєстрації та авторизації користувачів. Розроблено модулі для роботи з освітніми компонентами, студентами, керування графіком їх навчання, отриманням звітності. Впроваджено систему ролей та дозволів, зокрема здійснено розподіл функцій та контроль доступу для студентів, викладачів та адміністраторів системи. Розроблено графічний інтерфейс користувача для планування та керування навчальним процесом, формування графіків навчання, отримання звітів по кожному викладачу та студенту.

Виконано етапи тестування та впровадження системи. Виявлено та виправлено помилки. Система впроваджена та активно використовується для навчання студентів.

КЛЮЧОВІ СЛОВА: АНАЛІТИКА, ГРАФІК, ВЕБ-СИСТЕМА, ЗВІТНІСТЬ, КЕРУВАННЯ, НАВЧАННЯ, ОСВІТНІЙ ПРОЦЕС, РОЗКЛАД.

ABSTRACT

The master degree work consists of the developed web-based system for planning and organizing the learning process, as well as a documentation in the form of an explanatory note. The explanatory note has a volume of 81 pages. 22 drawings, 4 code listings, 15 tables, and 40 used literary sources are used as illustrative material.

The purpose of the work is to design, develop and implement a web-oriented information system for planning and managing training courses.

The work identified the needs of system users (students, teachers, administrators), collected and analyzed requirements for the system. In the course of designing the system, the architecture and scheme of the database were developed, the requirements for functionality and the graphical web interface were determined.

Software development was carried out. Functionality for user registration and authorization has been implemented. Modules for working with courses, students, managing their study schedule, and receiving reports have been developed. A system of roles and permissions has been implemented, in particular the distribution of functions and access control for students, teachers and system administrators. A graphical user interface has been developed for planning and managing the educational process, creating study schedules, and receiving lesson reports for each teacher and student.

The stages of testing and implementation of the system have been completed. Bugs were identified and fixed. The system is implemented and actively used for student training.

KEY WORDS: ANALYTICS, EDUCATIONAL PROCESS, LEARNING, MANAGEMENT, REPORTING, SCHEDULE, WEB SYSTEM.

ЗМІСТ

РОЗДІЛ 1. ОГЛЯД ТА АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ ВЕБ-СИСТЕМ ДЛЯ ПЛАНУВАННЯ ТА ОРГАНІЗАЦІЇ НАВЧАЛЬНИХ ЗАНЯТЬ	12
1.1. Постановка основної задачі розробки веб-системи	12
1.2. Порівняння сучасних аналогічних продуктів	15
1.3. Обґрунтування вибору програмних засобів розробки веб-системи	20
Висновки до розділу 1	23
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ДЛЯ ПЛАНУВАННЯ НАВЧАЛЬНИХ ЗАНЯТЬ	24
2.1. Математична модель формування розкладу навчальних занять	24
2.2. Визначення функціональних вимог користувачів та розробка діаграми варіантів використання.....	28
2.3. Розробка діаграм активності, послідовності та класів.....	31
2.4. Проектування та створення реляційної бази даних системи.....	38
2.5. Процес розробки системи	43
Висновки до розділу 2	50
РОЗДІЛ 3. ІНСТРУКЦІЯ КОРИСТУВАННЯ ТА РОЗГОРТАННЯ СИСТЕМИ. РЕЗУЛЬТАТИ ТЕСТУВАННЯ.....	51
3.1. Особливості щодо встановлення веб-системи.....	51
3.2. Керівництво користувача	53
3.3. Проведення та результати тестування веб-системи	61
Висновки до розділу 3	66
ВИСНОВКИ.....	67
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	69
ДОДАТКИ.....	74

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

LMS: Learning Management System – Система керування навчанням

CMS: Course Management System – Система керування курсами

API: Application Programming Interface – Інтерфейс прикладного програмування

SIS: Student Information System – Студентська інформаційна система

UI: User Interface – Користувацький інтерфейс

UX: User Experience – Досвід користувача

CMS: Content Management System – Система керування контентом

SQL: Structured Query Language – мова структурованих запитів

HTML: Hypertext Markup Language – гіпертекстова мова розмітки сторінок

CSS: Cascading Style Sheets – каскадні таблиці стилів

JSON: JavaScript Object Notation – текстовий формат для обміну даними

ВСТУП

Актуальність теми. Розробка веб-систем для планування та організації процесу навчання залишається надзвичайно актуальною у сучасному освітньому середовищі. Ось деякі фактори, які підкреслюють актуальність цього напрямку:

Планування навчального процесу. Система може допомогти у плануванні розкладу занять, аудиторій, матеріалів та інших ресурсів для навчання, як у дистанційній формі та і офлайн.

Покращення організації та ефективності навчального процесу. Забезпечення ефективного планування та організації навчання може допомогти зменшити навантаження на адміністраторів та забезпечити більш якісний навчальний досвід, надаючи засоби для відстеження та оцінки якості навчання.

Дистанційне керування. Навчальні заклади і компанії широко використовують веб-системи для забезпечення безперервного навчання та роботи з дому.

Організація подій. Веб-система може включати можливості для організації подій, семінарів, тренінгів і інших дистанційних або офлайн-заходів.

Реєстрація та управління учасниками навчального процесу. Система може допомогти у веденні списків студентів та викладачів, їх реєстрації та керуванні їхніми даними.

Оцінки, звітність. Система може включати функції оцінювання студентів та створення звітів для кожного викладача або студента.

Комунікація та співпраця. Веб-система може забезпечувати спільноту учасників та інструменти для співпраці, навіть в дистанційній формі або в офлайн-режимі.

Проведення аналітики. Відстеження результатів навчання та аналізу їх для подальшого вдосконалення процесу.

Безпека та контроль доступу. З міркувань безпеки, важливо контролювати доступ до інформації і ресурсів учасників навчання.

З огляду на ці фактори, розробка веб-системи для планування та організації процесу навчання може сприяти покращенню ефективності, доступності та контролю над навчанням.

Метою випускної кваліфікаційної роботи є розробка веб-орієнтованої системи для планування навчальних занять.

При розробці веб-системи важливо визначити головні цілі, які система має вирішувати. Основними цілями, на які варто звернути увагу, є:

- Покращення якості навчання: створення передумов для підвищення якості навчання, підвищення академічних досягнень здобувачів вищої освіти і розвиток їх професійних навичок та компетентностей.
- Підвищення залученості та забезпечення доступності: створити систему, яка заохочує активну участь студентів, сприяє їх мотивації та розвитку, зробити навчання більш доступним широкому колу користувачів, включаючи людей з різними потребами та з різних регіонів.
- Ефективне планування: допомогти навчальним закладам та викладачам ефективно планувати розклади, ресурси та активності, щоб максимізувати використання ресурсів.
- Спрощення адміністративних процесів: зменшити кількість рутинної адміністративної роботи, пов'язану з реєстрацією, відстеженням даних та іншими аспектами організації навчання.
- Забезпечення безпеки даних: захист особистих даних студентів та інших конфіденційних інформаційних ресурсів.
- Аналіз і звітність: надавати інструменти для збору даних та аналізу, щоб здійснювати звітність та приймати рішення на основі даних.
- Гнучкість і адаптивність: забезпечити можливість пристосовувати систему до різних типів навчання і освітніх програм.
- Забезпечення співпраці: створити можливості для співпраці між викладачами, студентами і іншими учасниками навчального процесу.
- Удосконалення процесу оцінювання: забезпечити ефективну систему оцінювання студентів та відстеження їхнього академічного прогресу.

- Підтримка різних видів навчання: створити систему, яка може підтримувати різні види навчання, включаючи лекції, семінари, лабораторні роботи, практикуми і більше.

- Інтеграція з іншими системами: забезпечити можливість інтеграції з іншими системами, такими як системи керування вмістом, бази даних студентів і більше.

- Постійне вдосконалення: забезпечити можливість вдосконалення системи на основі отриманих даних та зворотного зв'язку від користувачів.

Для реалізації поставлених цілей і досягнення мети визначимо *основні завдання* випускної кваліфікаційної роботи:

- визначити актуальність, мету, цілі і завдання розробки;
- визначити функціональні можливості та вимоги системи, провести аналіз потреб користувачів системи;
- розробити математичну модель планування розкладу навчальних занять;
- здійснити проектування системи, а саме розробити архітектуру системи та бази даних, визначити варіанти використання серверної частини та веб-інтерфейсу;
- здійснити розробку програмного забезпечення;
- здійснити розробку системи ролей та дозволів для різних категорій користувачів: студентів, викладачів та адміністраторів;
- провести тестування системи: розробити систему тестів та виконати, виявити та виправити наявні помилки;
- здійснити впровадження системи та технічний супровід, оновлення та виправлення помилок після впровадження;
- оцінити ефективність засобами аналізу використання системи, на основі отриманих даних внести покращення;
- провести узагальнення і систематизацію результатів дослідження.

Об'єктом дослідження є інформаційні технології для планування навчальних занять.

Предметом дослідження є методи проектування та розробки веб-систем для організації навчального процесу, особливості створення та обробки даних для підготовки та проведення навчального процесу.

Методологія роботи. Під час дослідження було використано загальнонаукові та емпіричні наукові методи, що ґрунтуються на системному підході, такі як абстрагування, аналіз, синтез, порівняння, узагальнення, систематизація, методи об'єктно-орієнтованого проектування та моделювання

Для розробки системи було вирішено використати стек технологій LAMP, а саме операційну систему Linux, веб-сервер Apache HTTP Server, систему керування БД MySQL та мову програмування PHP. Для швидкої та якісної розробки було обрано PHP-фреймворк Yii2. Як суміжні технології було використано середовище розробки PHPStorm, систему керування версіями Git, менеджер PHP-розширень Composer.

Наукова новизна дослідження. До найбільш вагомих наукових результатів слід віднести:

- проведений аналіз особливостей провадження освітньої діяльності закладів вищої освіти, професійно-технічних навчальних закладів, загальноосвітніх навчальних закладів;
- порівняльний аналіз функціональних можливостей сучасних LMS-систем, що дозволив визначити спільні та унікальні функції кожної системи та їх варіанти використання для різних типів навчальних закладів;
- математична модель планування розкладу навчальних занять;
- розроблені уніфіковані функціональні діаграми системи, що дозволяють їх використання для навчальних закладів, визначено ролі кожного користувача
- запропоновано уніфіковану діаграму класів та структуру реляційної бази даних

Практичне значення роботи полягає у програмній реалізації запропонованих моделей та функціональних діаграм. Розроблений веб-додаток дозволяє створити передумови для підвищення якості надання освітніх послуг

шляхом підвищення інформативності здобувачів вищої освіти та удосконалення процесів планування, організації та провадження освітнього процесу. Функціональність реалізованої системи підтверджена шляхом проведеного тестування.

Особистий внесок здобувача. Усі результати роботи отримані автором самостійно.

Апробація результатів роботи. Матеріали кваліфікаційної роботи доповідалися на II Всеукраїнській науково-технічній конференції «Комп'ютерні технології: інновації, проблеми, рішення» 23-24 листопада 2023 року [14].

РОЗДІЛ 1. ОГЛЯД ТА АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ ВЕБ-СИСТЕМ ДЛЯ ПЛАНУВАННЯ ТА ОРГАНІЗАЦІЇ НАВЧАЛЬНИХ ЗАНЯТЬ

1.1. Постановка основної задачі розробки веб-системи

Розробка веб-системи для планування та організації процесу навчання – комплексний процес, що включає етапи постановки задачі, проектування бази даних та архітектури системи, написання програмного коду системи, тестування, відлагодження, розгортання та технічний супровід. Дана система може бути корисною для навчальних закладів, компаній, що надають освітні послуги, тренерів.

Основна задача розробки веб-системи для планування і організації процесу навчання – забезпечити зручний і ефективний інструмент для управління розкладом, реєстрацією студентів і викладачів, сповіщеннями та відстеженням навчальних відомостей, формування звітністю та аналітикою. Розробка веб-системи для планування і організації процесу навчання може мати наступні основні **завдання**:

Управління розкладом. Система повинна дозволити створювати і змінювати розклад занять для учасників освітнього процесу. Вона повинна враховувати різні види занять, групи, аудиторії та інші фактори.

Реєстрація студентів і викладачів. Система повинна мати можливість реєстрації нових користувачів (студентів і викладачів) та зберігати їхні дані.

Сповіщення і нагадування. Система може надсилати сповіщення і нагадування користувачам про найближчі заняття, важливі події або зміни у розкладі.

Формування і відстеження успішності і відомостей. Система може дозволити викладачам і студентам вносити і відстежувати оцінки, завдання, рейтинги і інші відомості, пов'язані з навчанням.

Спільнота і комунікація. В системі можуть бути форуми, чати або інші інструменти для спілкування між студентами і викладачами, а також можливість обміну матеріалами та ресурсами для навчання.

Аналітика і звіти. Система може збирати дані про навчання, відвідуваність, успішність і інші параметри, щоб допомогти адміністраторам і викладачам приймати рішення з удосконалення провадження освітньої діяльності.

Безпека та захист від несанкціонованого доступу до особистих даних користувачів і конфіденційної інформації, що є однією з найважливіших функцій.

Інтеграція з іншими системами. Система може інтегруватися з іншими системами, такими як системи управління навчанням (LMS), електронні бібліотеки, системи електронної пошти тощо.

Мобільний доступ. Забезпечення можливості доступу до системи через мобільні пристрої та планшети може бути важливим аспектом для зручності користувачів.

Масштабованість. Система повинна бути здатна розширюватися і працювати ефективно навіть при зростанні обсягу користувачів і даних.

Розробка такої веб-системи вимагає детального аналізу вимог користувачів і планування, а також ефективного програмного забезпечення і забезпечення безпеки для даних користувачів. Сформуємо орієнтовний **план розробки такої системи:**

Етап 1. Аналіз потреб та вимог. Визначимо вимоги і розуміння потреб користувачів, а саме які функції потрібно включити в систему і які основні завдання будуть вирішуватися цією системою.

Етап 2. Проектування архітектури. На даному етапі спроектуємо загальну архітектуру системи. Визначимо, які компоненти будуть включені, і як вони взаємодіятимуть між собою.

Етап 3. Проектування бази даних. На цьому етапі важливо розробити оптимальну структуру для зберігання бази даних для зберігання інформації про дисципліни, студентів, викладачів, розклади, матеріали та інше.

Етап 4. Розробка інтерфейсу користувача. При цьому слід врахувати зручність використання та інтуїтивно зрозумілу навігацію.

Таким чином, для досягнення поставленої мети, і розв'язку задачі є доцільним визначити **функціональні можливості системи**:

- 1) Додавання та редагування освітніх компонент: створення можливості для додавання нових навчальних дисциплін та редагування наявних;
- 2) Формування розкладу та планування: реалізація функцій для створення розкладу занять та планування навчальних активностей;
- 3) Керування студентами: забезпечення можливості додавання та видалення студентів з навчальних дисциплін та груп;
- 4) Звітність: розробка системи формування історії активностей студентів та генерації звітів про їхній прогрес;
- 5) Сповіщення та комунікація: впровадження інструментів для сповіщень студентів та викладачів, а також можливостей для спілкування та обговорення матеріалів;
- 6) Тестування та відлагодження: проведення ретельного тестування системи, виявляючи і виправляючи помилки та недоліки;
- 7) Впровадження: введення системи в експлуатацію, розробка інструкцій для користувачів та адміністраторів;
- 8) Підтримка та оновлення: забезпечення підтримки користувачів та регулярне оновлення системи для виправлення помилок і додавання нових функцій;
- 9) Моніторинг і аналітика: забезпечення можливості відстеження використання системи та аналізу даних щодо навчального процесу;
- 10) Функції безпеки: дотримання високих стандартів безпеки для захисту даних користувачів;

11) Можливість інтеграції системи з іншими інструментами і сервісами, такими як електронні бібліотеки, LMS, аутентифікаційні системи тощо;

12) Регулярна оцінка ефективності системи та внесення необхідних змін та покращень.

Таким чином, поставлена задача, визначений план та функціональні особливості системи є опорними елементами та сприятимуть процесу розробки веб-системи для планування та організації процесу навчання, яка відповідає потребам користувачів та навчального закладу.

1.2. Порівняння сучасних аналогічних продуктів

В даному розділі здійснимо порівняння п'яти поширених веб-систем і систем управління контентом (CMS), які можуть використовуватися для планування і організації процесу навчання: Moodle, Blackboard, Canvas, Schoology, Edmodo [1-28].

Moodle – безкоштовна і відкрита система управління навчанням (LMS), спеціально призначена для освітніх закладів [15]. Має розширений набір інструментів для створення курсів, завдань, тестів і спільнот для навчання. Забезпечує можливість відстеження активності студентів і їхньої успішності.

Серед основних функціональних можливостей Moodle можна виділити:

- *Створення курсів*: викладачі можуть створювати онлайн-курси, додавати матеріали, завдання, тестові завдання та інші навчальні ресурси;
- *Організація розкладу*: Moodle дозволяє створювати розклад занять та управляти ним, визначаючи дні та часи проведення занять;
- *Завдання та оцінювання*: викладачі можуть створювати завдання, виставляти оцінки, а також відстежувати прогрес студентів;
- *Тестування і оцінювання*: Moodle підтримує створення різних видів тестів і опитувань, а також автоматизоване оцінювання;
- *Форуми і спільноти*: учасники курсу можуть взаємодіяти через форуми, обговорювати матеріали та співпрацювати в онлайн-спільнотах;

– *Відстеження успішності*: Moodle дозволяє відстежувати активність студентів, прогрес у навчанні, відвідуваність та інші статистичні дані;

– *Інші функції Moodle*: інтерактивність і мультимедіа, захист даних і безпека, додаткові функції через розширення та плагіни, мобільні застосунки.

Blackboard – популярний LMS, який використовується в навчальних закладах і компаніях, що надають навчальні послуги [16]. Має інструменти для створення курсів, спільнот, відстеження навчальних відомостей та здійснення спілкування між учасниками.

Основні функціональні можливості LMS Blackboard включають в себе:

– *Створення курсів*: викладачі можуть створювати онлайн-курси та завантажувати навчальний матеріал, включаючи тексти, відео, зображення і інші ресурси;

– *Завдання та оцінювання*: існують різні типи завдань, що можуть створюватись викладачами і включають завдання для здачі, тести, опитування та інші види оцінювання;

– *Організація розкладу*: система дозволяє визначити розклад для кожного курсу та інформувати студентів про нього;

– *Спільноти і обговорення*: наявність інструментів для створення форумів, блогів та обговорень, які сприяють взаємодії між студентами та викладачами;

– *Аналітика та звіти*: можливість відстежувати прогрес студентів та аналізувати їхню активність;

– *Відеоконференції*: інтегрована можливість проводити відеоконференції для онлайн-зустрічей та навчальних занять в режимі реального часу;

– *Серед інших функцій Blackboard*: наявність мобільних застосунків, інтеграція з іншими системами, безпека і конфіденційність особистих даних користувачів, безпека системи в цілому.

Canvas – інша популярна LMS, розроблена компанією Instructure. Відома своєю зручністю використання та можливістю інтеграції з іншими інструментами для навчання та відстеження успішності.

Основні функціональні можливості LMS Canvas включають в себе [18]:

- *Створення курсів*: Викладачі можуть створювати та налаштовувати онлайн-курси, додавати матеріали, завдання, тести та інші ресурси для навчання;
- *Оцінювання і звіти*: Canvas надає можливість створювати та оцінювати завдання, тести, опитування і стежити за успішністю студентів;
- *Розклад і календар*: В системі можна створювати розклади занять та подій, що допомагає студентам та викладачам планувати час;
- *Спільноти і обговорення*: Canvas має інструменти для створення форумів, чатів і обговорень, що сприяє взаємодії між користувачами;
- *Відеоконференції*: можливість інтеграції з інструментами для проведення веб-конференцій та віртуальних уроків;
- *Створення завдань*: Canvas надає можливість викладачам створювати завдання та оцінювати їх в режимі реального часу під час занять;
- *Серед інших функцій Canvas*: мультимедіа та інтерактивність, наявність мобільних застосунків для студентів і викладачів, адаптивний дизайн, інтеграція з іншими системами.

Schoology – це платформа для навчання, яка поєднує LMS та систему соціального спілкування для учнів і вчителів [17]. Дозволяє створювати завдання, спільноти, спілкуватися та відстежувати навчальний прогрес.

Функції Schoology включають:

- *Створення курсів*, додавання матеріалів для навчання, завдання та інші ресурси;
- *Завдання та оцінювання*: створення завдань, виставлення оцінок та відстеження прогресу студентів;
- *Спільноти і групи*: можливість створювати спільноти для обговорень, об'єднувати студентів та вчителів в групи для спільної роботи;
- *Інтерактивність і співпраця*: можливість спільної роботи над завданнями та проектами, обміну повідомленнями та обговоренням матеріалів;
- *Оцінювання і аналітика*: наявність інструментів для створення тестів і опитувань, а також аналізу навчальних даних для покращення якості навчання;

– *Засіб для батьків*: надання доступу для батьків для відстеження академічного прогресу своїх дітей;

– *Серед інших функцій системи*: наявність мобільних застосунків, інтеграція з іншими системами (Google Apps, Microsoft 365, LTI-сумісними додатками і багатьма іншими), безпека та конфіденційність, завдання для вчителів для відстеження власного професійного розвитку та навчання.

Edmodo – це безкоштовна платформа для навчання, яка спрямована на роботу зі школами і вчителями. Має інструменти для створення курсів, спільнот, завдань і комунікації між учасниками.

Основні функціональні можливості LMS Edmodo включають в себе [29]:

– *Створення класів і груп*: вчителі можуть створювати онлайн-класи і групи для організації навчання та спільної роботи над завданнями і проектами;

– *Завдання і зміст курсу*: створення завдань, додавання матеріалів для навчання та спільна робота над ними зі студентами;

– *Засоби спілкування*: наявність інструментів для обміну повідомленнями, обговорення матеріалів та спілкування внутрішніми групами;

– *Оцінювання і звіти*: створення та оцінювання завдань вчителями, відстеження успішності студентів;

– *Модуль "Батьки"*: можливість батькам відстежувати академічний прогрес своїх дітей та спілкуватися з вчителями;

– *Бібліотека ресурсів*: наявність бібліотеки навчальних ресурсів, яку вчителі можуть використовувати для підтримки своїх уроків;

– *Серед інших функцій Edmodo*: мобільні застосунки для студентів та вчителів, інтеграція з Google та Microsoft, що спрощує спільну роботу з документами та календарями, забезпечення захисту особистих даних і безпеки користувачів; завдання для вчителів для власного навчання і розвитку.

Наведемо функціональні можливості даних LMS-систем в порівняльній таблиці 1.1.

Порівняння функціональних можливостей сучасних LMS-систем

Функціональність	Moodle	Blackboard	Canvas	Schoology	Edmodo
Створення курсів	+	+	+	+	+
Розклад	+	+	+	-	-
Завдання і оцінювання завдань	+	+	+	+	+
Тестування і оцінювання тестів	+	+	+	+	+
Створення класів і груп	+	+	+	+	+
Засоби спілкування, відеоконференції	+	+	+	+	+
Аналітика та звітність	+	+	+	+	+
Модуль для батьків	-	-	-	+	+
Безпека і конфіденційність	+	+	+	+	+
Завдання для вчителів	-	-	-	+	+

Підсумовуючи результати порівняння, можна зробити висновки про використання тої чи іншої системи. *Moodle* є універсальним рішенням з відкритим кодом і підходить для різних навчальних закладів. Система має надійний набір функцій, але може потребувати технічних знань для налаштування та обслуговування. *Blackboard* – система, яка переважно використовується у вищій освіті. *Canvas* відомий своєю простотою використання, масштабованістю та інтеграцією з іншими інструментами. *Schoology* робить акцент на соціальній взаємодії та співпраці. *Edmodo* – зручна безкоштовна платформа, яка також зосереджена на спілкуванні вчителів, учнів і батьків.

Кожна з даних систем має свої унікальні функціональні можливості і призначена для різних типів освітніх установ та потреб користувачів. Вибір конкретної системи буде залежати від специфіки освітнього закладу і конкретних вимог до функціональності.

1.3. Обґрунтування вибору програмних засобів розробки веб-системи

В даному пункті обґрунтуємо вибір мови програмування PHP [30] та PHP-фреймворку Yii2 [31], а також реляційної бази даних MySQL для розробки веб-орієнтованої системи планування і організації процесу навчання.

Вибір реляційної бази даних MySQL для розробки веб-системи планування і організації навчального процесу можна обґрунтувати наступними важливими факторами:

Надійність і стабільність: MySQL є однією з найбільш поширених та довірених систем управління реляційними базами даних. Вона має довгий і успішний трек-рекорд в щодо стабільності та надійності. У навчальному середовищі важливо, щоб дані були доступними та надійними.

Продуктивність: MySQL славиться своєю продуктивністю, особливо в операціях читання даних (read-heavy). У веб-системі для навчання, де часті операції зчитування (наприклад, завантаження матеріалів, записи студентів або графіки) є розповсюдженими, швидке виконання запитів є важливим для створення комфортного користувацького досвіду.

Підтримка ACID: MySQL гарантує цілісність та стійкість даних завдяки підтримці властивостей ACID (Атомарність, Співмірність, Ізоляція, Довіреність). Це важливо в навчальних системах, де точність та надійність даних мають першочергове значення.

Відкритий код та безкоштовність: MySQL є відкритим продуктом, що робить його безкоштовним для використання. Це може значно знизити витрати на розробку та підтримку веб-системи в освітньому закладі.

Спільнота і підтримка: MySQL має велику та активну спільноту користувачів та розробників. Це означає наявність багато ресурсів, документації та форумів для вирішення проблем і отримання відповідей на питання.

Масштабованість: MySQL може обробляти широкий спектр завдань, від невеликих додатків до великих систем. Вона надає різні рушії збереження та

опції кластеризації, що дозволяє масштабувати систему для освітнього закладу з ростом обсягів даних.

Зручність у використанні: MySQL досить просто проектується та керується, що робить її підходящою для розробників, які можуть не мати великого досвіду адміністрування баз даних. Також існують багато користувацьких інтерфейсів та інструментів, що спрощують завдання управління базами даних.

Захист даних: MySQL надає надійні засоби захисту, такі як аутентифікація користувачів, контроль доступу та опції шифрування. Це важливо для захисту конфіденційних освітніх даних, таких як записи студентів та оцінки.

Вибір PHP як мови програмування та Yii2 як PHP-фреймворку для розробки веб-орієнтованої системи планування та організації навчального процесу може бути обґрунтований наступними перевагами:

Поширеність та надійність: PHP є однією з найпоширеніших серверних мов сценаріїв для веб-розробки. Мова має велику та активну спільноту розробників, що означає доступ до великої бібліотеки ресурсів, документації та плагінів. Yii2, як фреймворк PHP, добре підтримується та відповідає найкращим практикам, що робить його надійним вибором.

Рентабельність: PHP та Yii2 є технологіями з відкритим вихідним кодом, що робить їх рентабельним вибором для навчальних закладів з обмеженим бюджетом.

Легкість у навчанні: PHP відомий своєю простотою та легкістю у вивченні, що може бути вигідним для організацій, в т. ч. навчальних закладів, які хочуть швидко навчати своїх розробників. Yii2, надаючи розширені функції, також більш легший в засвоєнні порівняно з деякими іншими фреймворками.

Швидкість розробки: Yii2 – це високопродуктивний PHP-фреймворк, який підтримує швидку розробку програм. Він надає такі функції, як інструмент генерації коду, віджети для роботи з даними та гнучку реалізацію ActiveRecord, яка може пришвидшити процес розробки для створення навчальних програм.

Функції безпеки: PHP та Yii2 мають надійні функції безпеки. Yii2, зокрема, містить вбудовані інструменти для вирішення поширених проблем безпеки, таких як SQL-ін'єкції, міжсайтовий скриптинг (XSS) і атаки піддробки міжсайтових запитів (CSRF). Це вкрай важливо для захисту конфіденційних освітніх даних.

Архітектура MVC: Yii2 дотримується архітектурного шаблону Model-View-Controller (MVC), який сприяє організації коду, а саме розподілу завдань по обробці запиту серед трьох основних компонент: контролерів, моделей та виглядів. Це особливо корисно для побудови складних освітніх систем, оскільки дозволяє створювати модульний і підтримуваний код.

Розширюваність: Yii2 дуже розширюваний і дозволяє розробникам легко інтегрувати сторонні бібліотеки та компоненти [31]. Ця гнучкість є важливою при створенні налаштованої освітньої системи, яка може вимагати різноманітних функцій та інтеграцій.

Продуктивність: PHP створено для розробки веб-застосунків, а Yii2 розроблено як високопродуктивну структуру. Це сполучення дасть переваги у більш швидкому завантаженні веб-сторінок, забезпечуючи таким чином зручну роботу для студентів, викладачів і адміністраторів.

Спільнота активних розробників: Yii2 має активну спільноту розробників, що дає можливість скористатись великою кількістю розширень, плагінів і підтримкою для вирішення конкретних вимог і завдань у сфері освіти.

Сумісність: PHP сумісний з різними веб-серверами та операційними системами, що робить його універсальним і придатним для розгортання в різних навчальних середовищах.

Таким чином, вибір реляційної СУБД MySQL для розробки і впровадження веб-орієнтованої системи планування і організації навчального процесу дає можливість побудувати надійну, продуктивну і ефективну реляційну базу даних, яка допоможе забезпечити стабільну та надійну роботу системи. Вибір PHP як мови програмування та Yii2 як PHP-фреймворку для освітньої веб-орієнтованої системи має ряд переваг, що включають економічну ефективність, легкість в

засвоєння, безпеку та продуктивність. Ці технології досить поширені у сфері веб-розробки та можуть допомогти оптимізувати процес розробки безпечної та ефективної системи для планування навчального процесу.

Висновки до розділу 1

1. В першому розділі здійснено постановку задачі, визначено план розробки, завдання та функціональні особливості веб-орієнтованої системи, що сприятиме процесу розробки веб-системи для планування та організації процесу навчання у відповідності до потреб користувачів та навчального закладу.

2. Проведено порівняння функціональних можливостей сучасних поширених LMS-систем. Зроблено висновок про спільні та унікальні функції кожної системи та їх варіанти використання для різних типів освітніх установ та потреб користувачів. Вибір конкретної системи буде залежати від специфіки освітнього закладу і конкретних вимог до функціональності.

3. Обґрунтовано вибір мови програмування PHP та Yii2-фреймворку, а також СУБД MySQL для розробки системи. Дані технології досить поширені у сфері веб-розробки та допомагають оптимізувати процес розробки безпечної та ефективної системи для планування навчального процесу.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ДЛЯ ПЛАНУВАННЯ НАВЧАЛЬНИХ ЗАНЯТЬ

2.1. Математична модель формування розкладу навчальних занять

Планування розкладу навчальних занять є досить складним процесом, який поєднує викладачів, студентів (учнів), аудиторії закладу освіти, освітні компоненти, та обумовлює створення складної системи взаємозв'язків між інформаційними таблицями у процесі формування. Складність процесу планування обумовлюється також необхідністю врахування загальних даних завантаженості викладачів (вчителів) та студентів (учнів) з вибором аудиторій, що відповідають певним критеріям, що неможливо здійснити без оптимізаційних методів обробки інформації. Без використання сучасних інформаційних технологій це здійснити доволі важко, тому доволі актуальною є розробка нових засобів автоматизації цих процесів. В навчальному процесі музичної школи передбачені групові, індивідуальні заняття та самостійна робота студентів (учнів). Діти в школі навчаються грі на певних музичних інструментах, вивчають образотворче мистецтво та хореографію. Деякі заняття потребують аудиторій, де знаходяться музичні інструменти (великогабаритні), інші – ні. Крім того в навчальних закладах працюють дитячі творчі колективи та творчі колективи викладачів. Тому оптимізація процесів планування навчальних занять займає особливе місце при розробці даної веб-орієнтованої системи. Кількість занять на тиждень є фіксованою сталою величиною і визначається оптимальним розподілом навантаження. Важливим завдання є також встановлення системи зв'язків між окремими таблицями бази даних, який пропонується реалізувати за єдиним шаблоном для всіх таблиць уніфікованим шаблоном, котрий включає попередній відбір та копіювання даних у робочу таблицю, що відповідає за розклад бази даних. Слід акцентувати увагу, що необхідно забезпечити захист від появи можливих помилок, що можуть викликані оператором через неувважність. Тому при створенні веб-орієнтованої системи для планування навчальних занять пропонується системне впровадження масивів дозволів на

доступ до інформаційного забезпечення бази даних. Такий вибір буде забезпечувати доступ лише до тих освітніх компонент на які буде зареєстрований студент (учень).

Проведемо розробку математичної моделі для планування індивідуального розкладу занять, використовуючи матеріал, що представлений у роботах [34-40].

Одиницею навчального процесу, і відповідно розкладу, є навчальне заняття (урок). У поточному семестрі необхідним є проведення визначеної кількості освітніх компонент. Кожне заняття проводиться призначеним викладачем, що вимагає рознесення його занять по часу. Також накладаються обмеження по часу, кількості занять для студентів (учнів) на тиждень, аудиторного фонду.

Введемо наступні множини:

$T = \{T_1, T_2, \dots, T_n\}$ – множина викладачів (вчителів), де n – кількість викладачів, які задіяні в освітньому процесі поточного семестру, $n = \overline{1 \dots n_{max}}$,

$S = \{S_1, S_2, \dots, S_m\}$ – множина студентів (учнів), де m – кількість студентів (учнів) в поточному семестрі, $m = \overline{1 \dots m_{max}}$;

$D = \{D_1, D_2, \dots, D_h\}$ – множина освітніх компонент (дисциплін), які передбачені навчальним планом освітнього закладу, h – кількість освітніх компонент в поточному семестрі, $h = \overline{1 \dots h_{max}}$.

Кожен студент (учень) може вибрати певну кількість вибіркового освітніх компонент. Значення параметру m визначається кількістю студентів (учнів), яким викладається дана освітня компонента у визначений дискретний момент часу.

Освітні компоненти (дисципліни) в розкладі розрізняються наступними типами, (2.1):

$$D_i = \{Z_{i1}, Z_{i2}, Z_{i3}, Z_{i4}\}, \quad (2.1)$$

де Z_{i1} – теоретичне заняття; Z_{i2} – практичне заняття; Z_{i3} – звітне заняття; Z_{i4} – самостійна робота.

Кожне заняття Z_{ij} , де i – освітня компонента, j – тип заняття, потребує p_{ij} часу і може викладатися викладачем (учителем) з множини $\tau_{ij} \subseteq \{T_1, T_2, \dots, T_n\}$ студентам (учням) з множини $\mu_{ij} \subseteq \{S_1, S_2, \dots, S_m\}$.

Таким чином, пріоритетними напрямками вирішення задачі є концептуальний аналіз основних характеристик процесу створення розкладів для студентів (учнів), зокрема α – характеристика викладачів; β – характеристики студентів (учнів), γ – характеристики освітніх компонент, та реляційне формування цільового функціоналу задачі δ .

На підставі проведених досліджень визначаємо, що поле α складається з двох частин, а саме $\alpha = \alpha_1 \cap \alpha_2$, де $\alpha_1 = P$ – завантаженість викладача (вчителя) з віковими групами, α_2 – ваговий коефіцієнт спеціалізації викладача (вчителя), $\alpha_2 = W$.

Загальний час навантаження викладача (учителя) складає (2.2):

$$p_k = \sum_{i=1}^v \sum_{j=1}^4 q_{ij} \begin{cases} q_{ij}=p_{ij} | (\tau_{ij}=T_k) \cap (\sum_{i=1}^v \sum_{j=1}^4 \mu_{iz} > 0) \\ q_{ij}=0 | (\tau_{ij} \neq T_k) \cup (\sum_{i=1}^v \sum_{j=1}^4 \mu_{iz} = 0) \end{cases} \quad (2.2)$$

Позначимо множину уроків як λ_{du} , робочі дні тижня як $TD \in \{1, 2, 3, 4, 5, 6\}$, номери уроків розкладу занять $TU \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Отримаємо множину часових інтервалів $TR = TD \times TU$.

Запишемо приклади можливих обмежень для k -го викладача (вчителя), (2.3-2.6):

$$x_{k1} = \lambda_{du} | (d \in \{5, 6\}) \cap (7 < u < 10); \quad (2.3)$$

$$x_{k2} = \lambda_{du} | (d = 5) \cap ((u = 9) \cup (u = 10)); \quad (2.4)$$

$$x_{k3} = \lambda_{du} | (d \neq 1) \cap (d \neq 6); \quad (2.5)$$

$$x_{k4} = \lambda_{du} | ((d = 1) \cap (u = 10)) \cup (d > 1, \forall u). \quad (2.6)$$

Характеристичні параметри студентів (учнів) у загальній системі створення розкладу подаються наступним чином: $\beta = \beta_1 \cap \beta_2$, де $\beta_1 = G_i$ – студенти (учні) входять до певних груп, що пропонується музичною школою та передбачено вибраним напрямком підготовки; $\beta_2 = F_i$ – студенти (учні), що

формують низку потоків відповідно до обраного напрямку індивідуальної підготовки (обраного музичного інструменту), (2.7).

$$F = \{F_1, F_2, \dots F_4\}, \quad (2.7)$$

де F_1 – загальний потік студентів (учнів) в музичній школі, F_2 – потік студентів (учнів) для проведення групових занять теоретичної підготовки, F_3 – потік студентів (учнів) для занять з індивідуальної підготовки, F_4 – потік студентів (учнів) для занять із суміжних напрямків підготовки.

Запропоноване класифікаційне угруповання є відкритим і може коригуватися у відповідності до об'єкту впровадження. Поле γ спрямоване на наступні характеристики: $\gamma_1 = f_{ij}$ – освітня компонента, що належить до плану підготовки та викладається в довільному порядку.

В свою чергу час занять для k -го студента (магістранта) за індивідуальним розкладом визначається з виразу, (2.8):

$$r_k = \sum_{i=1}^v \sum_{j=1}^4 q_{ij} \begin{cases} q_{ij}=p_{ij} | (\mu_{ik}>0); \\ q_{ij}=0 | (\mu_{ik}=0). \end{cases} \quad (2.8)$$

$\gamma_1 = R$ – освітня компонента належить до множини дисциплін, вибраних студентом (учнем). Множину належності дисципліни до індивідуального плану підготовки студента (учня) можна представити у наступному вигляді, (2.9):

$$R \subseteq D \left| p_i = \sum_{k=1}^m q_{ik} > 0 \right| \begin{cases} q_{ik}=p_{ik} | \mu_{ik}>0 \\ q_{ik}=0 | \mu_{ik}=0 \end{cases} \quad (2.9)$$

для $i = \overline{1 \dots h}$.

$\gamma_3 = J$ – певний вид занять освітньої компоненти, що викладаються одним або кількома викладачами (вчителями).

Множина належності освітньої компоненти до індивідуального плану роботи викладача визначається $\tau_{ijk} \in \{0,1\}$, де $i = \overline{1 \dots h}$, $j = \overline{1 \dots 4}$, $k = \overline{1 \dots n}$, (2.10):

$$J \subseteq D \left| \sum_{k=1}^n \tau_{ijk} > 0 \right| p_{ij} > 0, \forall i, \forall j \quad (2.10)$$

Таким чином, на наявному аудиторному навантаженні p_{ij} будь-якого типу заняття для всіх освітніх компонент визначаються згідно розподілу навантаження для проведення ij -того заняття.

$\gamma_3 = B = B_1 \cup B_2 \cup B_3$ – розподіл освітніх компонент за видом занять згідно кратності тижнів, де B_1 – множина щотижневих занять; B_2 – множина занять, що відбуваються один раз на два тижні; B_3 – множина освітніх компонент, кількість яких не кратна половинній кількості тижнів (планування таких занять здійснюється за визначеними датами).

$\gamma_3 = C$ – розподіл типів занять освітньої компоненти, враховуючи обмеження, що зумовлені психолого-педагогічними особливостями викладання та сприйняття освітніх компонент студентами (учнями): $c_k \in \{-20, \dots, 20\}$, $k = \overline{1 \dots a}$, тут a – кількість психолого-педагогічних обмежень.

Отже, цільова функція автоматизованого формування розкладу δ з врахуванням необхідних умов з максимальною кількістю балів матиме вигляді, (2.11):

$$\delta = \max \left(\sum_{k=1}^h w_k \sum_{j=1}^{m_x} \frac{v_{kj} \cdot \pi_{kj}}{\sum_{i=1}^{m_x} \pi_{ki}} + \sum_{k=1}^a n_k \cdot c_k \right), \quad (2.11)$$

де $v_{kj} \in \{0,1\}$, $j = \overline{1 \dots m_x}$ – ознака врахування додаткової умови для викладача (вчителя) для викладання відповідної освітньої компоненти, π_{kj} – пріоритет реалізованої умови, w_k – ваговий коефіцієнт рейтингу викладача (спеціалізації), π_{ki} – пріоритети комплексу різних умов для відповідного викладача (вчителя).

Проведе дослідження щодо створення математичної моделі формування розкладу навчальних занять є підґрунтям для подальшої програмної реалізації веб-орієнтованої системи для планування навчальних занять.

2.2. Визначення функціональних вимог користувачів та розробка діаграми варіантів використання

В першому розділі ми сформулювали функціональні можливості, типові для сучасних LML-систем. В цьому розділі опишемо детально дані функції з розподілом їх по ролях студента, викладача та адміністратора.

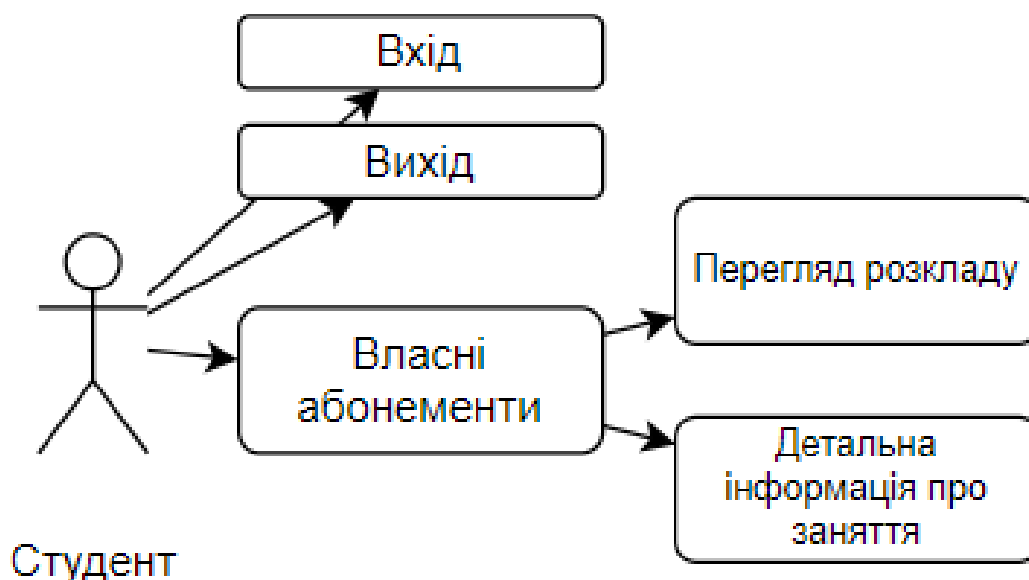


Рис. 2.1. Діаграма прецедентів для студента

Найпростішою роллю в системі є студент – користувач, що має обмежену кількість дозволів (рис.2.1). Зокрема:

- вхід та вихід із системи;
- перегляд даних користувача як студента: розклад, деталі кожного заняття;
- формування і перегляд звітів по навчанню.

Викладач – роль в системі з дозволами для часткового внесення або модифікації даних, рис.2.2. Викладач володіє такими функціями:

- керування заняттями: внесення відміток щодо присутності студентів на заняттях (статус студента щодо заняття);
- перегляд розкладу, як власного так і розкладу інших викладачів (ця опція може бути і недоступною);
- формування і перегляд звітів по навчанню – можливість вибирати місяць та переглядати підсумки та деталі проведеної роботи: виконані замовлення та їх статус, заняття та їх статус, інформація про студентів, що відвідували заняття.

Адміністратор системи має найбільшу кількість дозволів, рис.2.3. Це пояснюється тим, що мета системи – забезпечення інструментів для планування та організації навчального процесу, а ці засоби переважно стосуються саме адміністраторів.

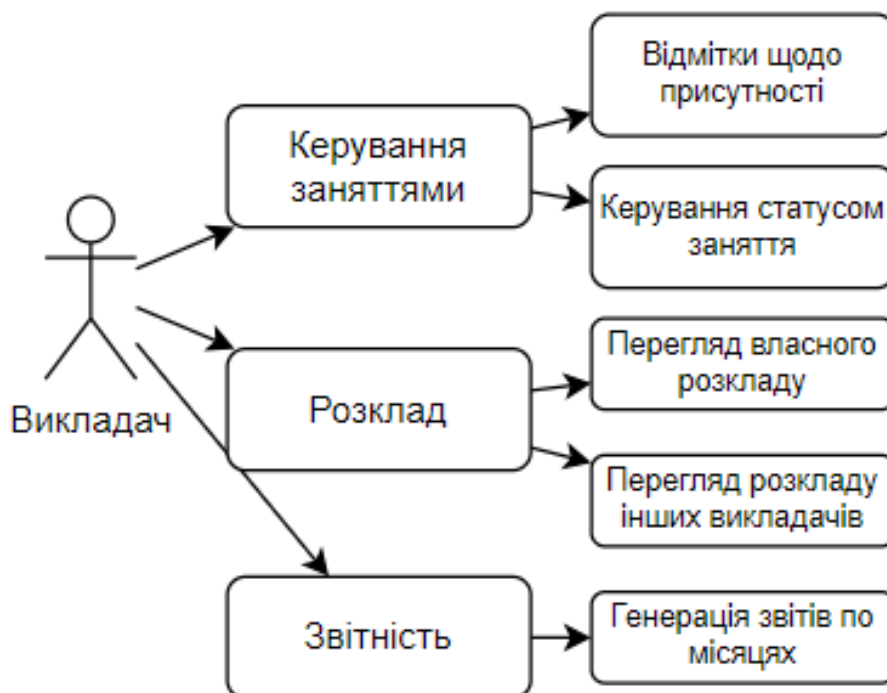


Рис. 2.2. Діаграма прецедентів для викладача

Крім того, роль адміністратора успадковує роль викладача, що дозволяє адміністратору керувати розкладом на всіх рівнях. Функції адміністратора включають:

- ручну реєстрацію користувачів. Передбачається використання однієї форми для реєстрації користувачів з одночасним наданням їм ролі викладача та/або студента;
- створення замовлень та абонементів: замовлення – пакет, що включає в себе назву курсу, студентів, що мають намір пройти даний курс та викладача. Записи про студентів щодо даного замовлення будемо називати «абонементами»;
- керування розкладом, що включає в себе додавання занять, редагування занять та їх статусів, додавання «абонементів» до занять. Заняття – це подія, що включає в себе час початку і час завершення, викладача (вчителя) та «абонементи» студентів (учнів). Таким чином між замовленням та заняттям може існувати відповідність багато до багатьох: замовлення як правило

здійснюється проведенням декількох занять, одне заняття може охоплювати «абонементи» з різних замовлень.

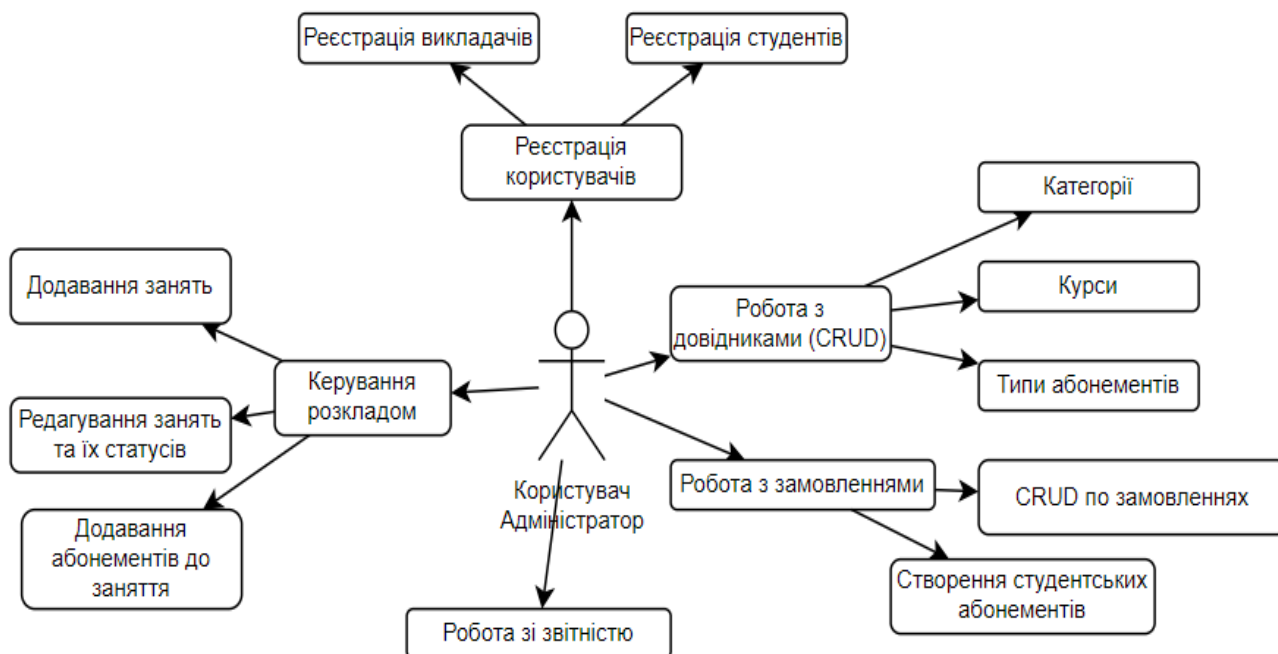


Рис. 2.3. Діаграма прецедентів для адміністратора

– Цей випадок передбачений в системі, проте рекомендується використовувати у виключних випадках. Бажано, щоб одне заняття містило лише «абонементи» конкретного замовлення, оскільки таким чином зберігається певна упорядкованість процесу навчання і немає неузгодженості між викладачами.

2.3. Розробка діаграм активності, послідовності та класів

Здійснимо розробку діаграми діяльності (активності), що ілюструє розподілений між адміністратором та викладачем процес по роботі з замовленнями, «абонеменентами» та розкладом, рис.2.4:

– Початок процесу фіксується виконанням запиту за доменною адресою веб-системи.

– Користувач здійснює вхід, заповнюючи відповідну форму своїми обліковими даними: логін або електронна адреса та пароль.

- Система отримує роль користувача та надає йому відповідні дозволи.
- Адміністратор створює замовлення – запис про наміри студента або групи студентів взяти заняття певної категорії. Кожне замовлення пов'язане з конкретним викладачем.
- Адміністратор на сторінці замовлення додає студента або студентів: створює для них таким чином «абонементи». Кожен «абонемент» відповідає конкретному студенту та конкретному замовленню. Крім того «абонементи» мають крім ідентифікаторів додатково унікальні порядкові номери, які можуть стати продовженням нумерації «абонементів», які реєструвались до впровадження даної системи.
- Адміністратор створює заняття – події, що мають час початку і час завершення (тривалість по замовчуванню можна встановлювати в категоріях курсів). Кожне заняття прив'язане до конкретного викладача.
- У формі додавання/редагування заняття адміністратор додає ті «абонементи», студенти яких відвідують дане заняття. Таким чином система може бути гнучкою: в межах одного замовлення, що містить декілька «абонементів» (студентів), можна створювати заняття з різними студентами. Наприклад, один студент із групи бажає окремо від інших провести додаткове заняття. Тому для нього можна створити окреме заняття і скорегувати ціну.
- Оскільки кожне замовлення пов'язане з викладачем, він має можливість переглянути замовлення, як ті що виконані, так і поточні та майбутні.
- Також викладач має можливість переглянути власний розклад, що формується з даних про заняття.
- Одним із дозволів для викладача є можливість змінювати статус заняття, який він провів: «Заплановано», «Проведено», «Проведено (заміна)», «Не проведено».
- Студенти, що записані на дане заняття, також можуть отримати статус після проведення уроку: «Заплановано», «Присутній», «Відсутній». Викладач має доступ до зміни статусу студента по даному заняттю.

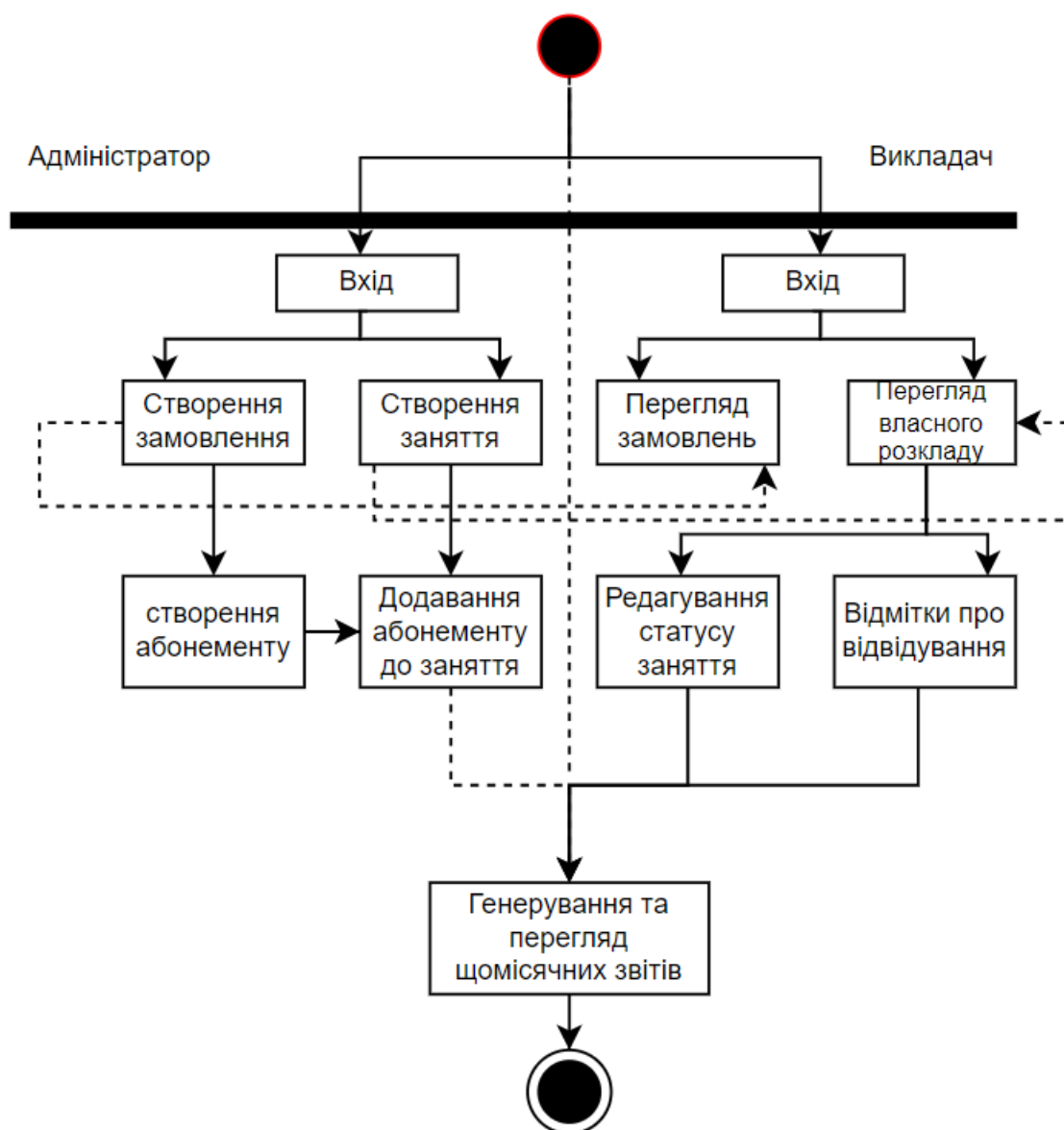


Рис. 2.4. Діаграма активностей, що ілюструє процес створення замовлення та розкладу

– Корисною дією є можливість генерування помісячних звітів для кожного викладача. Звіт може включати перелік замовлень виконувались протягом місяця. Кожне замовлення містить списки «абонементів», а для кожного «абонементу» формується розклад занять, що були заплановані. В кожному занятті виводиться його статус. Як варіант, можна сформувати звіт по заняттях для кожного студента, що відвідував заняття даного викладача.

Діаграма активності для користувача (адміністратора) представлена на рис.2.5.

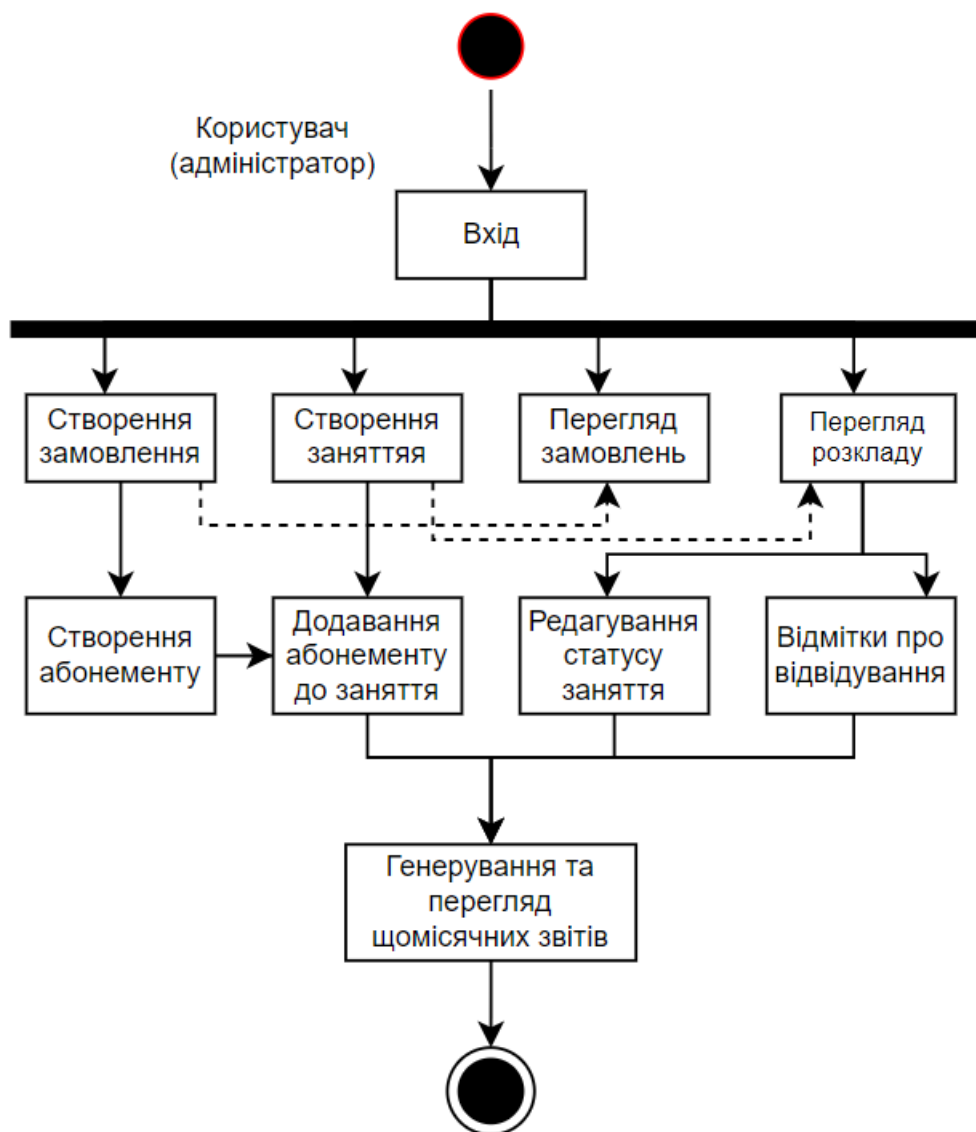


Рис. 2.5. Діаграма активності для користувача (адміністратора)

Наступна діаграма – діаграма послідовностей, рис.2.6 – ілюструє процеси обробки інформаційних даних, впорядковані в часовій послідовності між трьома учасниками: адміністратором, системою та базою даних.

1) Адміністратор створює замовлення через відповідну форму, зазначаючи курс, викладача, дату початку, орієнтовану дату завершення, а також статус замовлення («Заплановано», «В процесі», «Завершено», «Скасовано»);

2) На сторінці замовлення адміністратор додає студентів до даного замовлення – користувачів, що мають намір прослухати заняття по курсу. Запис, що відносить користувача до замовлення, назвемо «абонементом».



Рис. 2.6. Діаграма послідовностей для часової ілюстрації дій адміністратора, виконавчого скрипта (системи) та БД по роботі з замовленнями, «абонементами» та заняттями

3) Маючи створені «абонементи», адміністратор може перейти безпосередньо до планування навчального процесу, а саме створити заняття, скориставшись відповідною формою та візуальним представленням календаря, та додати до кожного заняття «абонементи» студентів.

4) На кінець місяця можна сформулювати та переглянути звіт по кожному викладачу, що включає в себе замовлення, «абонементи», проведені заняття та інформацію про відвідуваність. В звіти можна включити результати, такі як кількість замовлень по статусах («Завершено», «Скасовано» тощо), сітка «студенти – заняття» з датами проведення та відвідуваністю тощо.

Наступна діаграма – діаграма класів, рис.2.7 – візуально відображає структуру (атрибути та методи) та взаємозв’язки класів у системі.

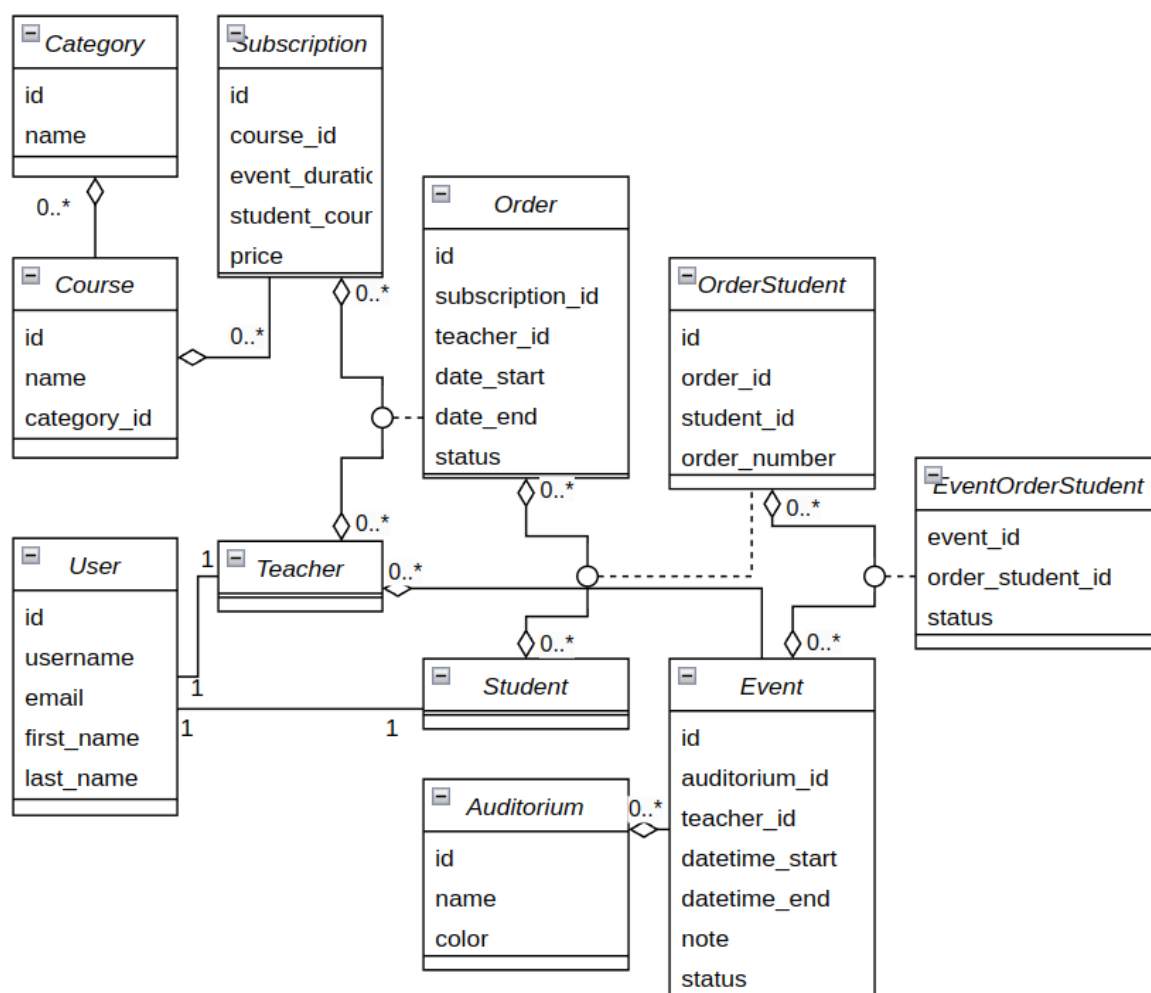


Рис. 2.7 Діаграма класів веб-орієнтованої системи для планування навчальних завдань

Розкриємо структуру даної діаграми, а саме присутні класи та їх відношення між собою.

Клас *User* – базовий клас, що займається обробкою даних про зареєстрованих користувачів. Даний клас пов’язаний відношенням один до одного з класами *Teacher* та *Student*.

Клас *Category* – базовий клас, що імплементує методи для роботи з напрямками освітніх компонент (категоріями).

Клас *Course* – клас для роботи з курсами (освітніми компонентами), тобто

структурованими навчальними програмами, розробленими для навчання студентів конкретній освітній компоненті або компететністями (навичкам).

Клас *Subscription* – клас, що робить курс варіативним, тобто розширяє інформацію про даний курс в контексті кількості занять, студентів, тривалості та вартості заняття. Один курс може мати декілька підписок.

Клас *Order* – клас для обробки замовлень. Замовлення створюється для студента або групи студентів, що мають намір навчатись. Кожне замовлення асоціюється з відповідною підпискою.

Клас *OrderStudent* призначений для додавання студента для даного замовлення. Кожне замовлення може асоціюватись з одним або декількома студентами. Запис про відношення студента до замовлення будемо називати абонементом.

Клас *Event* працює з заняттями – подіями, обмеженими в часі, що мають за мету навчити студентів деякій темі або практичному вмінню. Кожна подія пов'язана з конкретним вчителем, однією аудиторією та асоціюється з одним або декількома абонементами.

Клас *EventOrderStudent* відносить абонемент студентів до заняття. Таким чином, один студентський абонемент може опрацьовуватись декількома освітніми компонентами. Одне заняття може включати декілька абонементів.

Клас *Auditorium* – працює з довідником навчальних кабінетів. Бере участь у формуванні розкладу. Кожне заняття відноситься до конкретної аудиторії.

В даному підрозділі було спроектовано UML-діаграми, що допоможуть при проектуванні архітектури системи та схеми бази даних, а також при безпосередній розробці та користуванні системою.

2.4. Проектування та створення реляційної бази даних системи

В даному підрозділі здійснимо проектування та створення реляційної бази даних системи. База даних є реляційною і містить 11 таблиць, відповідно до статичної структури даних, описаною діаграмою класів вище. На рисунку 2.8 зображена схема бази даних:

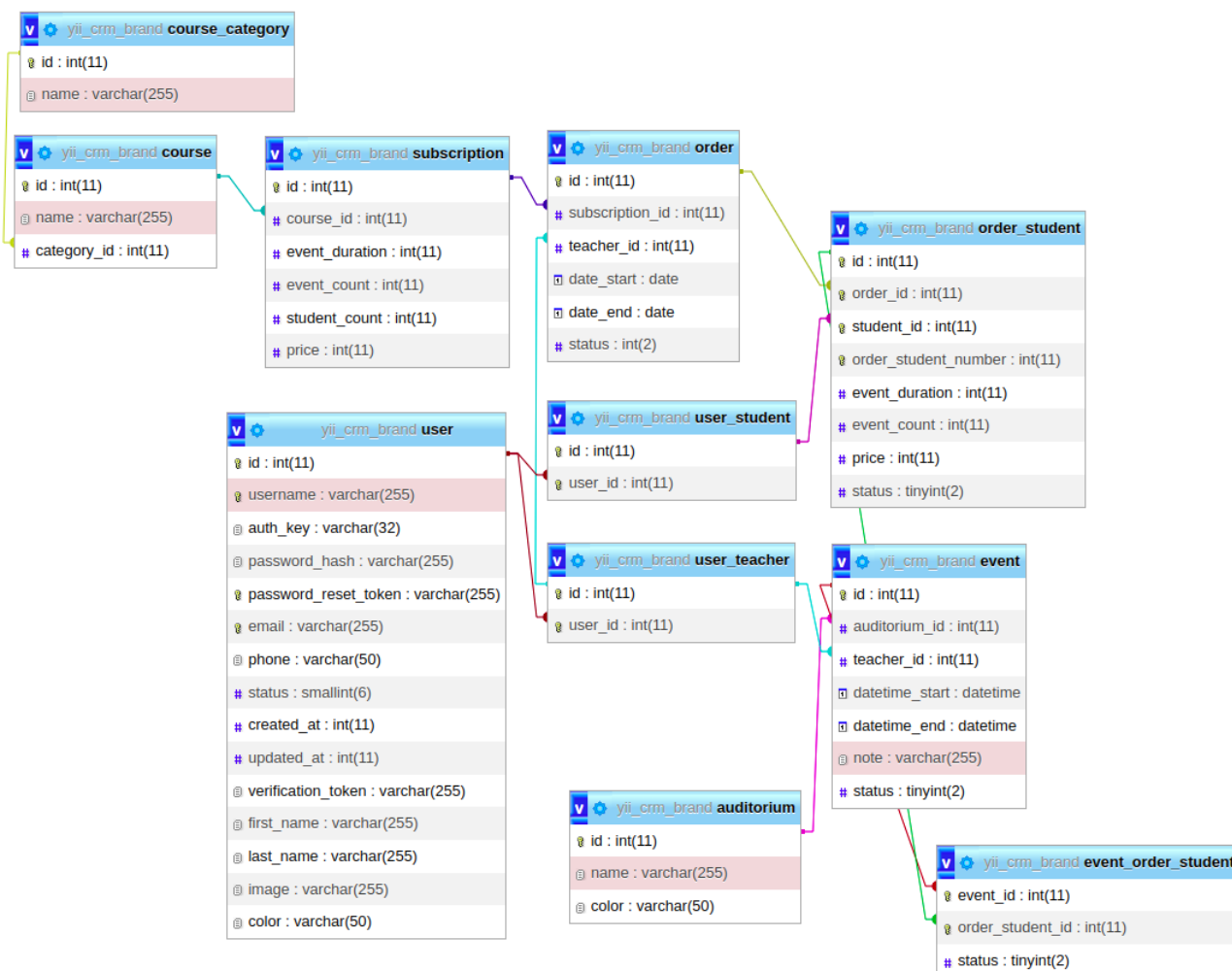


Рис. 2.8. Реляційна база даних веб-орієнтованої системи для планування навчальних занять

В додатку Б наведено дампи бази даних для автоматизованого створення таблиць, індексів та зв'язків між таблицями. З даним sql-файлом база даних імпортується, що дає можливість уникнути ручної роботи по створенню структури БД.

Наведемо опис таблиць бази даних. *User* – таблиця (див. табл. 2.1), що зберігає дані про зареєстрованих користувачів.

Таблиця 2.1

User – таблиця

Поле	Опис
id	Ідентифікатор користувача
username	Логін користувача
auth_key	Ключ автентифікації, що базується на cookie
password_hash	Хеш-рядок пароль
password_reset_token	Ключ для відновлення паролю
email	Електронна адреса
phone	Телефон
status	Статус (активний чи не активний)
created_at	Часова мітка реєстрації користувача
updated_at	Часова мітка останнього оновлення користувача
verification_token	Ключ верифікації облікового запису
first_name	Ім'я користувача
last_name	Прізвище користувача
image	Фото (зображення) користувача
color	Колір користувача

User_student – інформація про студентів: таблиця, що відносить користувачів до студентів (див. табл. 2.2).

Таблиця 2.2

User_student

Поле	Пояснення
id	Ідентифікатор студента
user_id	Ідентифікатор користувача

User_teacher – інформація про викладачів: таблиця, що відносить користувачів до викладачів (див. табл.2.3).

Таблиця 2.3

User_teacher

Поле	Пояснення
id	Ідентифікатор викладача
user_id	Ідентифікатор користувача

Course_category – таблиця категорій курсів (див. табл.2.4).

Таблиця 2.4

Course_category

Поле	Пояснення
id	Ідентифікатор категорії
name	Назва категорії

Course – таблиця курсів, зміст представлено у таблиці 2.5.

Таблиця 2.5

course

Поле	Пояснення
id	Ідентифікатор курсу
name	Назва курсу
category_id	Ідентифікатор категорії

Subscription – таблиця для створення варіацій курсу (підписок), що можуть зробити даний курс більш гнучким, враховуючи тривалість заняття, кількість занять, кількість студентів, ціну (див. табл. 2.6).

Таблиця 2.6

Subscription

Поле	Пояснення
id	Ідентифікатор підписки
course_id	Ідентифікатор курсу
event_duration	Тривалість заняття по замовчуванню
event_count	Кількість років по замовчуванню
student_count	Кількість студентів по замовчуванню
price	Ціна по замовчуванню

Order – таблиця замовлень (див. табл.2.7).

Таблиця 2.7

Order

Поле	Пояснення
id	Ідентифікатор замовлення
subscription_id	Ідентифікатор підписки
teacher_id	Ідентифікатор викладача
date_start	Дата початку
date_end	Дата завершення
status	Статус («очікує», «виконано», «не виконано»)

Order_student – таблиця, що працює з абонементом – записами про студентів, що відносяться до даного замовлення (див. табл.2.8).

Auditorium – таблиця даних про аудиторії, зміст якої представлено у таблиці 2.9. Вона містить ідентифікатор аудиторій, назву аудиторії та колір аудиторії.

Order_student

Поле	Пояснення
id	Ідентифікатор абонементу
order_id	Ідентифікатор замовлення
student_id	Ідентифікатор студента
order_student_number	Зовнішній номер абонементу
event_duration	Тривалість заняття для студента по замовчуванню
event_count	Кількість занять для студента по замовчуванню
price	Вартість даного замовлення для студента
status	Статус абонементу

Таблиця 2.9

Auditorium

Поле	Пояснення
id	Ідентифікатор аудиторії
name	Назва аудиторії
color	Колір аудиторії

Event – таблиця з даними про заняття (події) (див. табл. 2.10).

Таблиця 2.10

Event

Поле	Пояснення
id	Ідентифікатор заняття
auditorium_id	Ідентифікатор аудиторії
teacher_id	Ідентифікатор викладача
datetime_start	Дата і час початку заняття
datetime_end	Дата і час завершення заняття
note	Нотатки
status	Статус

Event_order_student – відношення студентських абонементів до занять (див. табл. 2.11).

Таблиця 2.11

Event_order_student

Поле	Пояснення
event_id	Ідентифікатор події
order_student_id	Ідентифікатор абонементу
status	Статус

В даному розділі автором здійснено проектування структури реляційної бази даних. Наведено зображення схеми бази даних та описано структуру кожної таблиці. Для створення самої бази даних та її таблиць, полів, індексів та зв'язків можна використати інструменти імпорту середовища PHPMyAdmin і дампи бази даних, наведений в додатку В.

2.5. Процес розробки системи

Оскільки в розробці використовується PHP-фреймворк Yii2, то сам процес розробки системи визначається правилами і підходами, що є типовими при використанні фреймворку. Серед таких підходів є використання архітектури MVC, інструменту для генерації моделей даних та CRUD-коду, використання ActiveRecord та віджетів форм і даних, система реєстрації, аутентифікації та авторизації тощо.

Орієнтовний план розробки:

- 1) установка фреймворку, створення проекту, віддаленого та локального репозиторіїв;
- 2) створення віртуального хосту, БД, налаштування з'єднання та семантичних URL;
- 3) створення таблиці user через міграцію, функціонал реєстрації та входу користувачів;

- 4) реалізація механізму авторизації. Створення базової системи ролей та дозволів: ролі Admin, Teacher, Student;
- 5) модуль редагування власного профілю користувача, можливість додавання/редагування власного фото;
- 6) проектування БД;
- 7) установка модуля адміністратора AdminLTE;
- 8) доступ до всіх функцій системи надати лише зареєстрованим користувачам через глобальну конфігурацію;
- 9) надати дозвіл для реєстрації користувачів лише адміністраторами;
- 10) функції для реєстрації користувачів: вчителів та студентів;
- 11) функції для редагування профілю користувача;
- 12) операції по роботі з довідниковими даними: список даних, додавання, редагування чи видалення:
 - операції по роботі з аудиторіями;
 - операції по роботі з категоріями курсів;
 - операції по роботі з курсами;
 - операції по роботі з пропозиціями як варіантами курсу;
- 13) Функції для роботи з даними навчального процесу:
 - операції по роботі з замовленнями;
 - операції по роботі із заняттями (подіями);
 - вивід розкладу у вигляді календаря; реалізація інтерфейсу для роботи з подіями через календар;
 - генерація і отримання звітності по викладачу за місяць.

Наведемо приклад програмної реалізації запиту на генерацію звітності по викладачу за деякий місяць.

Крок 1. Рядок URL-запиту визначимо як /order/teacher-work, де order – ідентифікатор контролера, teacher-work – ідентифікатор методу дії;

Крок 2. В контролері OrderController створимо метод дії actionTeacherWork, лістинг 2.1:

```

public function actionTeacherWork()
{
    $data = [];
    $teacherDateForm = new TeacherDateForm();
    $teachers = UserTeacher::getList();

    if (Yii::$app->request->isPjax) {
        $teacherDateForm->load(Yii::$app->request->post());
        if ($teacherDateForm->teacher_id > 0 && !empty($teacherDateForm->date)) {
            $dateTime = new DateTime($teacherDateForm->date);
            $dateTime->modify("+1 month");
            $finish = $dateTime->format('Y-m-d');
            $orders = Order::find()->andWhere(['teacher_id' => $teacherDateForm-
            >teacher_id])
                ->andWhere([">=", 'DATE(date_start)', $teacherDateForm->date])
                ->andWhere(["<", 'DATE(date_start)', $finish])->all();
            foreach ($orders as $order) {
                $data[] = [
                    'id' => $order->id,
                    'name' => $order->getFullName(),
                    'dateStart' => $order->date_start,
                    'dateEnd' => $order->date_end,
                    'studentOrders' => $order->getStudentOrders(),
                ];
            }
        }
    }
    return $this->render('teacher-work', [
        'data' => $data,
        'teacherDateForm' => $teacherDateForm,
        'teachers' => $teachers,
    ]);
}

```

Лістинг 2.1. Метод дії *teacher-work* контролера *Order*.

Даний метод відповідає за обробку запиту користувача за визначеною URL-адресою. Для збору та підготовки даних з моделей визначимо змінну *\$data*, створимо екземпляр моделі для форми *TeacherDateForm* та отримаємо масив всіх викладачів для можливості вибору з випадного списку. Далі метод перевіряє,

якщо запит відбувся асинхронно, тобто користувач відправив дані форми, здійснюється завантаження даних в модель форми. Якщо дані коректні, формуємо межі дат для фільтрації: дату початку і дату завершення (через місяць від дати початку). Далі отримуємо масив замовлень даного викладача між двома датами. Для кожного замовлення формуємо повну інформацію і передаємо її в масив `$data`. В кінці методу підготовлені дані, а саме масив `$data`, екземпляр форми `$teacherDateForm` та масив викладачів `$teachers`, передаємо в шаблон `teacher-work` для відображення даних засобами візуальних компонентів.

Крок 3. Модель `TeacherDateForm`, лістинг 2.2.

```
<?php
namespace app\models;
use yii\base\Model;
class TeacherDateForm extends Model
{
    public $teacher_id = 0;
    public $date = "";
    public function rules()
    {
        return [
            [['teacher_id'], 'integer'],
            [['date'], 'safe'],
        ];
    }
    public function attributeLabels()
    {
        return [
            'teacher_id' => "Вчитель",
            'date' => "Дата",
        ];
    }
}
```

Лістинг 2.2. Клас моделі форми `TeacherDateForm`

Даний клас призначений для створення і обробки форми для генерації звіту по викладачах за вибраний місяць. Атрибутами моделі є `$teacher_id` –

ідентифікатор викладача та \$date – дата початку даного місяця. Клас визначає метод для валідації даних rules() та метод для міток атрибутів attributeLabels().

Крок 4. Файл шаблону teacher-work.php. Компонент форми представлено у Лістингу 2.3.

```
<?php $form = ActiveForm::begin([
    'options' => ['data' => ['ajax' => true]],
    'id' => 'teacher-date-form'
]); ?>
<div class="row">
    <div class="col">
        <?= $form->field($teacherDateForm, 'teacher_id')-
>dropDownList(ArrayHelper::map($teachers, 'id', 'username'), [
    'id' => 'field-teacher-id',
    'prompt' => '--Виберіть вчителя--',
    'onchange' => '$("#teacher-date-form").submit()'
    ]) ?>
    </div>
    <div class="col">
        <?= $form->field($teacherDateForm, 'date')->widget(DatePicker::class, [
    'options' => ['placeholder' => 'Дата початку'],
    'pluginOptions' => [
        'autoclose' => true,
        'format' => 'yyyy-mm-dd',
        'maxViewMode' => 'years',
        'minViewMode' => 'months',
    ],
    'pluginEvents' => [
        "changeDate" => "function(e) {
            $('#teacher-date-form').submit();
        }",
    ]
    ]); ?>
    </div>
</div>
<?php ActiveForm::end(); ?>
```

Лістинг 2.3. Форма для генерування звіту

Дана форма містить два поля: випадний список для вибору потрібного викладача та поле для вибору дати в режимі вибору місяця. При виборі дати на сервер відправляється асинхронний запит для обробки і отримання даних звіту.

Крок 5. Файл шаблону teacher-work.php. Компонент для виводу звіту представлено нижче у Лістингу 2.4.

```
<?php if (!empty($data)): ?>
<table class="table table-bordered">
  <tr>
    <th>Дані замовлення</th>
    <th>Дані студентів</th>
  </tr>
  <?php foreach ($data as $order): ?>
    <tr>
      <td>
        <div><strong>Назва замовлення:</strong> <?= $order['name']; ?></div>
        <div><strong>Дата початку:</strong> <?= $order['dateStart']; ?></div>
        <div><strong>Дата завершення:</strong> <?= $order['dateEnd']; ?></div>
      </td>
      <td>
        <table class="table table-bordered">
          <tr>
            <th># абонементи</th>
            <th>Студент</th>
            <th>Ціна</th>
            <th>Заняття</th>
          </tr>
          <?php foreach ($order['studentOrders'] as $studentOrder): ?>
            <tr>
              <td><?= $studentOrder['orderStudentNumber']; ?></td>
              <td><?= $studentOrder['studentUsername']; ?></td>
              <td>
                <table class="table table-bordered">
                  <tr>
                    <th># заняття</th>
                    <th>Початок</th>
                    <th>Аудиторія</th>
                    <th>Вчитель</th>
                    <th>Статус</th>
                  </tr>
```

Лістинг 2.4. Виведення звіту (частина 1)

```

        <?php foreach ($studentOrder['events'] as $event): ?>
        <tr>
        <td><?= $event['number']; ?></td>
        <td><?= $event['start']; ?></td>
        <td><?= $event['auditorium']; ?></td>
        <td><?= $event['teacher']; ?></td>
        <td style="background-color: <?= $event['statusColor']
?>">

        <?= $event['status']; ?>
        </td>
        </tr>
        <?php endforeach; ?>
        </table>
        </td>
        </tr>
        <?php endforeach; ?>
        </table>
        </td>
        </tr>
        <?php endforeach; ?>
        </table>
        <?php endif; ?>

```

Лістинг 2.4. Виведення звіту (частина 2)

Вивід звіту здійснюється в таблиці на основі даних, що надходять від сервера при надсиланні запитів. Вивід даних групується по замовленнях. Для кожного замовлення виводяться студентські абонементи, а для кожного абонементу формується таблиця з даними про проведені уроки.

В даному розділі було описано план розробки системи, враховуючи можливості і підходи, що пропонуються сучасним РНР-фреймворком. Наведено кроки програмної реалізації щодо обробки запитів на генерацію та виведення звітів про виконання замовлень та проведення занять по викладачам за певний місяць.

Висновки до розділу 2

1 Проведено розробку математичної моделі для планування індивідуального розкладу занять.

2 В розділі здійснено детальний опис функціональних можливостей веб-орієнтованої системи для планування навчальних занять. Розроблено та наведено UML-діаграми варіантів використання, активностей, послідовності та класів, необхідні при проектуванні архітектури системи та схеми бази даних, а також при безпосередній розробці та користуванню системою.

3 Здійснено проектування структури реляційної бази даних. Наведено зображення схеми бази даних та описано структуру кожної таблиці. Для створення самої бази даних та її таблиць, полів, індексів та зв'язків використано інструменти імпорту середовища RНРMyAdmin і дамп бази даних.

4 Описано план розробки системи, враховуючи можливості і підходи, що пропонуються сучасним РНР-фреймворком. Наведено кроки програмної реалізації щодо обробки запитів на генерацію та виведення звітів про виконання замовлень та проведення занять по викладачам за певний місяць.

РОЗДІЛ 3. ІНСТРУКЦІЯ КОРИСТУВАННЯ ТА РОЗГОРТАННЯ СИСТЕМИ. РЕЗУЛЬТАТИ ТЕСТУВАННЯ

3.1. Особливості щодо встановлення веб-системи

В даному підрозділі розробимо інструкцію встановлення системи на локальний комп'ютер або на виділений сервер.

Крок 1. Установка програмного забезпечення. Для цього нам знадобляться дистрибутиви XAMPP (веб-сервер Apache і СУБД MySQL), система керування версіями Git та менеджер розширень для PHP Composer.

Крок 2. Розгортання проекту на локальній машині. Для цього доцільно вибрати папку з вихідним кодом та виконати клонування проекту з репозиторію. Відповідні команди такі:

```
cd /opt/lampp/htdocs  
git clone https://gitlab.com/{username}/lms-system
```

Крок 3. Встановлення залежностей. Для цього знадобиться composer. Перейдемо в папку lms-system та виконаємо команду, яка завантажить всі залежності, зареєстровані в нашому проекті:

```
cd /opt/lampp/htdocs/lms-system  
composer update
```

Крок 4. Створення віртуального хосту. На даному кроці нам потрібно створити доменне ім'я з перенаправленням на локальну IP-адресу та визначити шлях до вихідного коду, тобто створити віртуальний хост. Дані про віртуальний хост містяться в файлі /opt/lampp/etc/extra/httpd-vhosts.conf. Дані про доменне ім'я містяться в файлі /etc/hosts. Відкриємо в редакторі файл hosts і додамо інформацію про доменне ім'я:

```
127.0.1.1 crm-brand.local
```

Відкриємо файл httpd-vhosts.conf та додамо інформацію про віртуальний хост:

```
sudo gedit /opt/lampp/etc/extra/httpd-vhosts.conf
```

Вміст файлу:

```

<VirtualHost *:80>
    ServerName crm-brand.local
    DocumentRoot "/opt/lampp/htdocs/crm-brand/web"
    <Directory "/opt/lampp/htdocs/crm-brand/web">
        RewriteEngine on
        RewriteRule ^index.php/ - [L,R=404]
        RewriteCond %{REQUEST_FILENAME} !-f
        RewriteCond %{REQUEST_FILENAME} !-d
        RewriteRule . index.php
    </Directory>
</VirtualHost>

```

Крок 5. Після розгортання проекту і створення віртуального хосту необхідно запустити веб-сервер і сервер MySQL та виконати запит в браузері <http://crm-brand.local>.

Для запуску сервера відкриємо менеджер XAMPP:

```

cd /opt/lampp
sudo ./manager-linux-x64.run

```

В вікні менеджера запустимо служби Apache Web Server та MySQL Database.

Виконаємо запити

```

http://crm-brand.local/
http://localhost/php-myadmin/

```

Якщо всі кроки виконані правильно, то в браузері повинні з'явитись відповідні сторінки із формою для входу в систему та інструмент RНРMyadmin для роботи з БД.

3.2. Керівництво користувача

Здійснимо розробку інструкції користувача та опис сторінок веб-інтерфейсу. Проектування системи здійснювалося без привязки до конкретного навчального закладу. Однак реалізація програмного інтерфейсу здійснювалася з прив'язкою до музичної школи міста та враховуючи її особливості роботи та побажань адміністрації у плідній співпраці з батьками дітей, що займаються у даному навчальному закладі. Персональні дані користувачів змінено.

Вхід. Для користування системою необхідно здійснити вхід ввівши в форму логін та пароль, рис. 3.1.

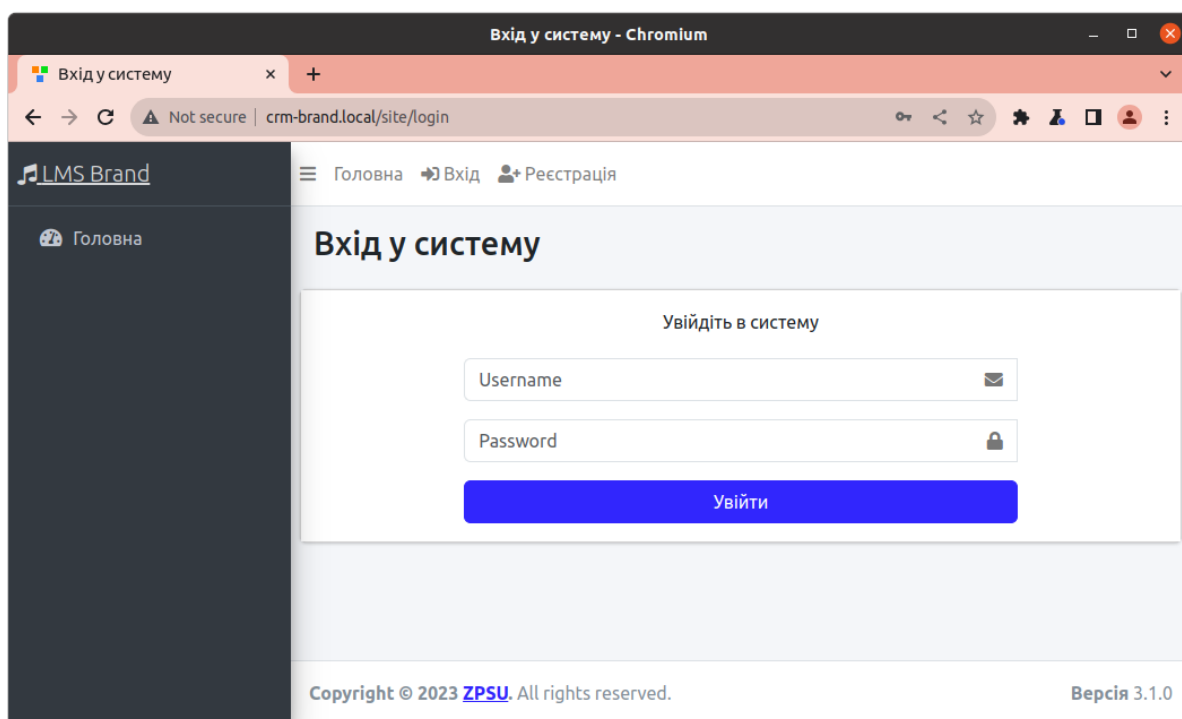


Рис. 3.1. Вхід в систему

Здійснивши вхід під адміністратором, користувач отримує доступ до таких сторінок, рис. 3.2 - 3.15.

Головна сторінка, рис. 3.2. Містить навігаційні блоки по групах «Довідники», «Користувачі» та «Навчальний процес». Всі інші функції розподілені по відповідних посиланнях.

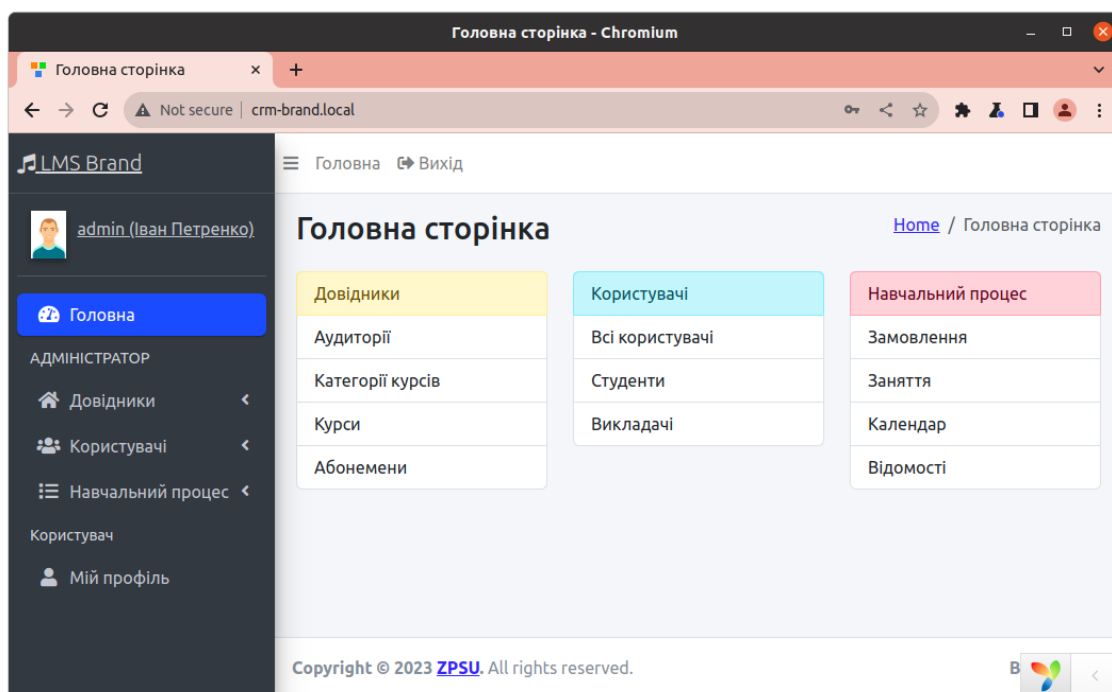


Рис. 3.2. Головна сторінка з навігацією по довідниках, користувачах та навчальному процесу

Сторінка “Аудиторії” (рис. 3.3). Містить довідник аудиторій з можливостями для фільтрації, сортування списку аудиторій, а також посилання на сторінку “Додавання аудиторії”.

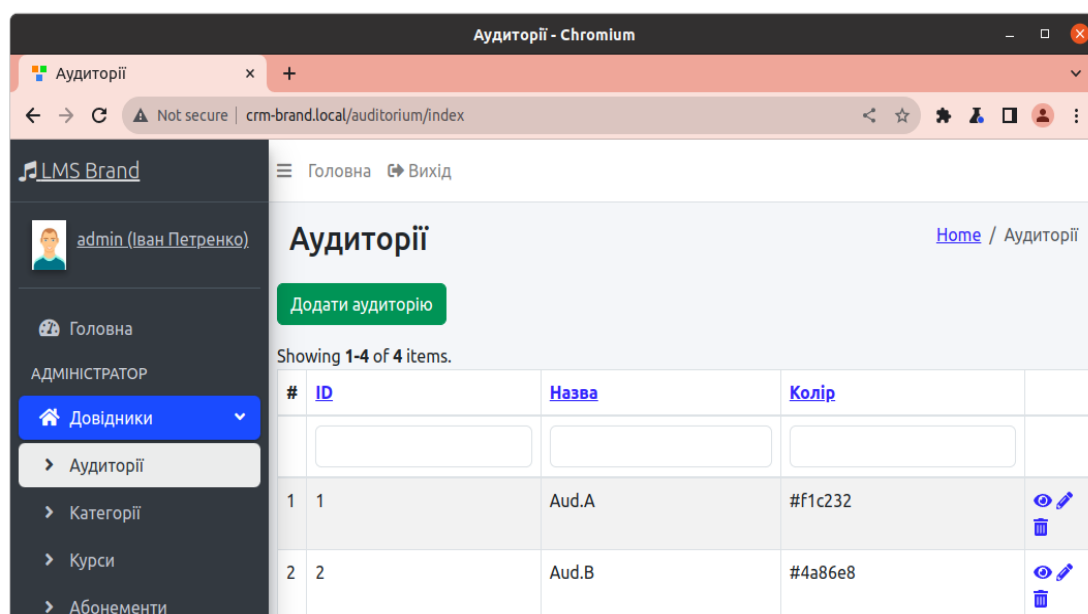


Рис. 3.3. Сторінка «Аудиторії»

Сторінка “Категорії курсів”, рис.3.4. Містить довідник категорій курсів та посилання на сторінку для створення нової категорії:

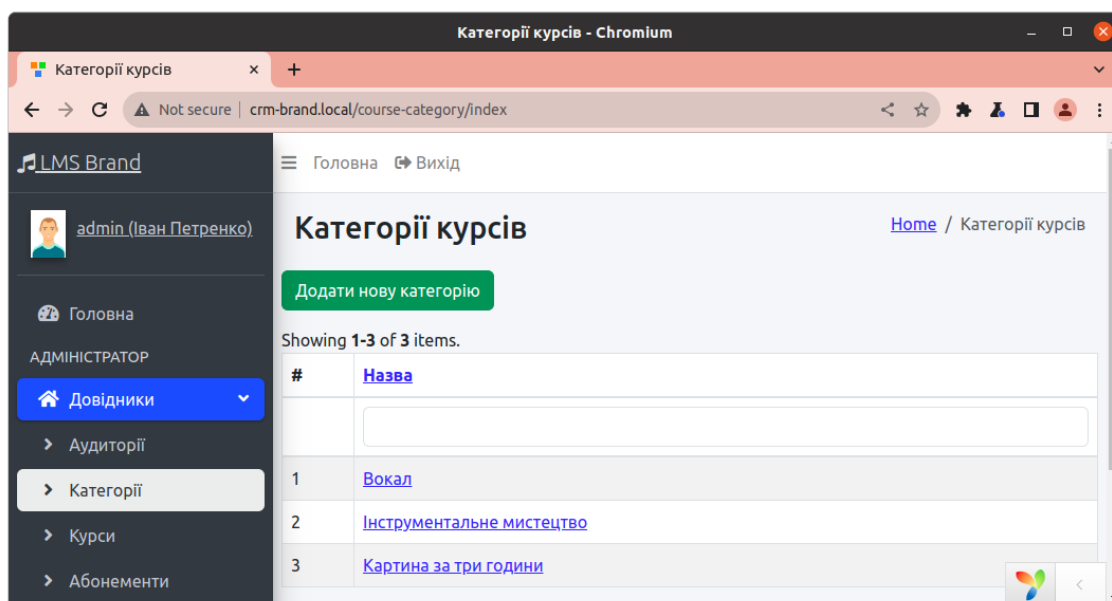


Рис. 3.4. Сторінка “Категорії курсів”

Сторінка «Курси», рис. 3.5. Довідникова сторінка зі списком курсів та можливість додавання нових курсів:

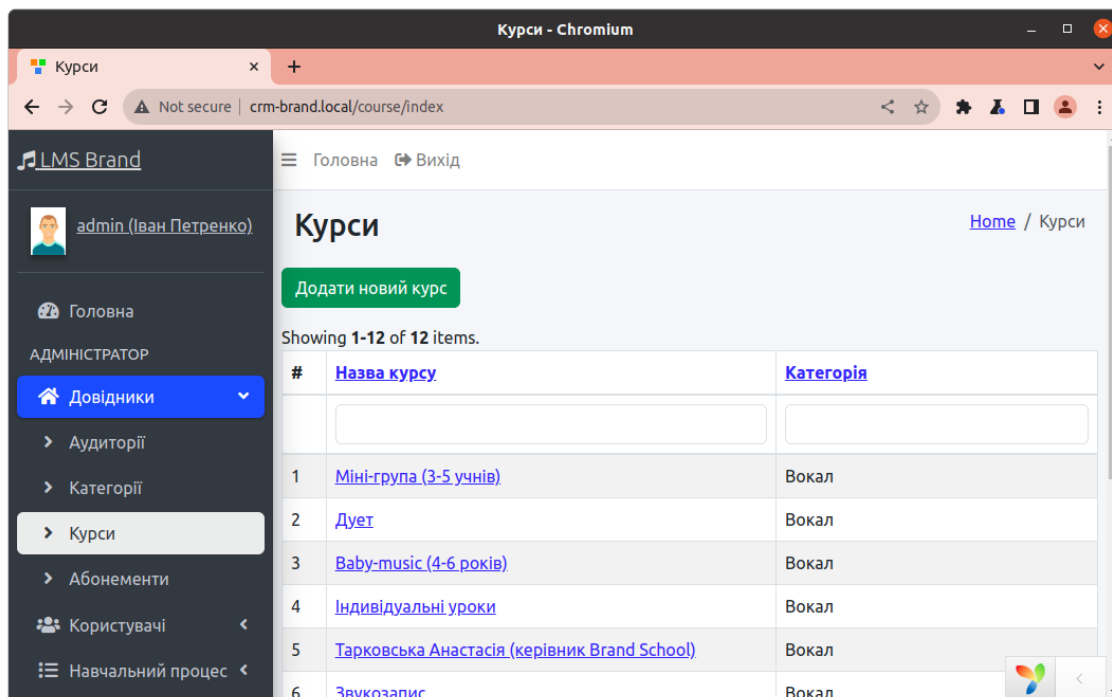


Рис. 3.5. Сторінка “Курси”

Сторінка «Пропозиції», рис. 3.6. Курс може мати різні пропозиції для варіювання параметрів: тривалість заняття, кількість уроків, кількість студентів, ціна. Дана сторінка призначена для перегляду та створення різних варіантів пропозицій:

Пропозиції - Chromium

Пропозиції

Створити пропозицію

Showing 1-20 of 46 items.

#	Курс	Тривалість заняття, хв	Кількість занять	Кількість студентів (макс.)	Ціна	Деталі
1	Вокал - Міні-група (3-5 учнів)	0	4	5	700	Перегляд
2	Вокал - Міні-група (3-5 учнів)	0	8	5	1200	Перегляд
3	Вокал - Міні-група (3-5 учнів)	0	1	5	250	Перегляд
4	Вокал - Дует	0	4	2	1000	Перегляд

Рис. 3.6. Сторінка “Пропозиції”

Сторінка «Користувачі» – сторінка для перегляду списку користувачів і можливістю реєстрації нових користувачів, рис.3.7.

Користувачі - Chromium

Користувачі

Додати

Showing 1-7 of 7 items.

Id	Username	Email	Phone	First Name	Last Name
1	admin	admin@gmail.com	+38(097)123-45-67	Іван	Петренко
8	alexprenko	alexprenko@gmail.com	+38(097)805-02-74	Олександр	Петренко
9	james	james@gmail.com	+38(097)805-02-74	Яків	Пономаренко

Рис. 3.7. Сторінка «Користувачі»

Сторінка “Профіль користувача” – перегляд детальної інформації про користувача, рис.3.8.

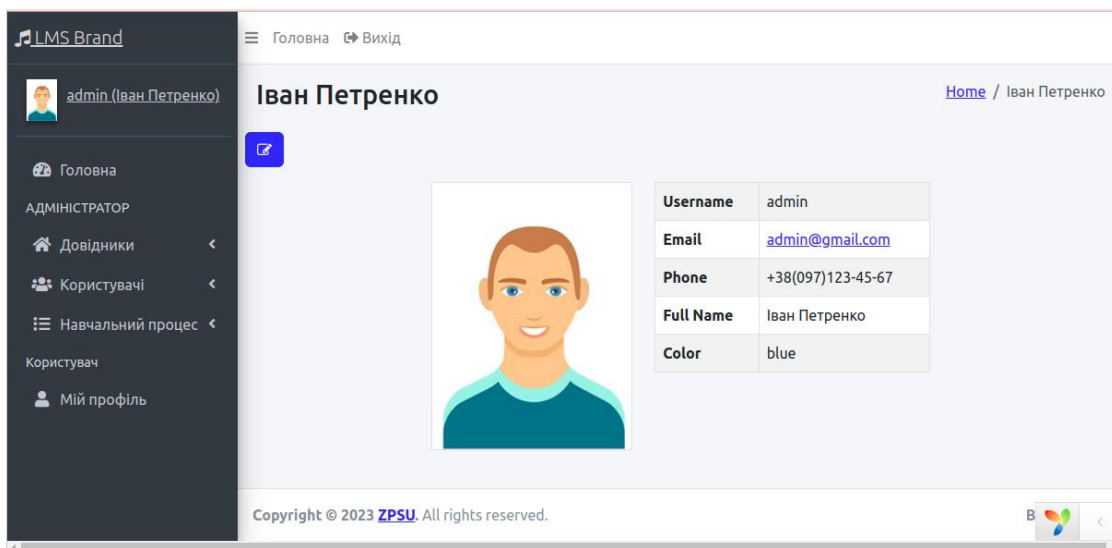


Рис. 3.8. Сторінка «Профіль користувача»

Сторінка «Реєстрація користувача» – форма для додавання нового користувача з одночасним віднесенням його до студентів чи до викладачів, або одночасно до обох груп, рис.3.9.

The screenshot displays the 'Registration' page. The page title is 'Реєстрація' (Registration). Below the title, there is a link 'Home / Реєстрація'. The form includes the following fields and options:

- Для реєстрації заповніть наступні поля: (For registration, fill in the following fields:)
- Логін (Login):
- Email:
- Телефон (Phone):
- Ім'я (Name):
- Прізвище (Surname):
- Пароль (Password):
- Роль (Role): ☒ Студент (Student), ☐ Вчитель (Teacher)
- Колір (Color):

A blue 'Реєстрація' (Registration) button is located at the bottom of the form.

Рис. 3.9. Сторінка “Реєстрація користувача”

Сторінка «Список замовлень» представлена на рисунку 3.10. Дана сторінка дозволяє працювати з замовленнями – записами про наміри студента або групи студентів придбати одну з пропозицій.

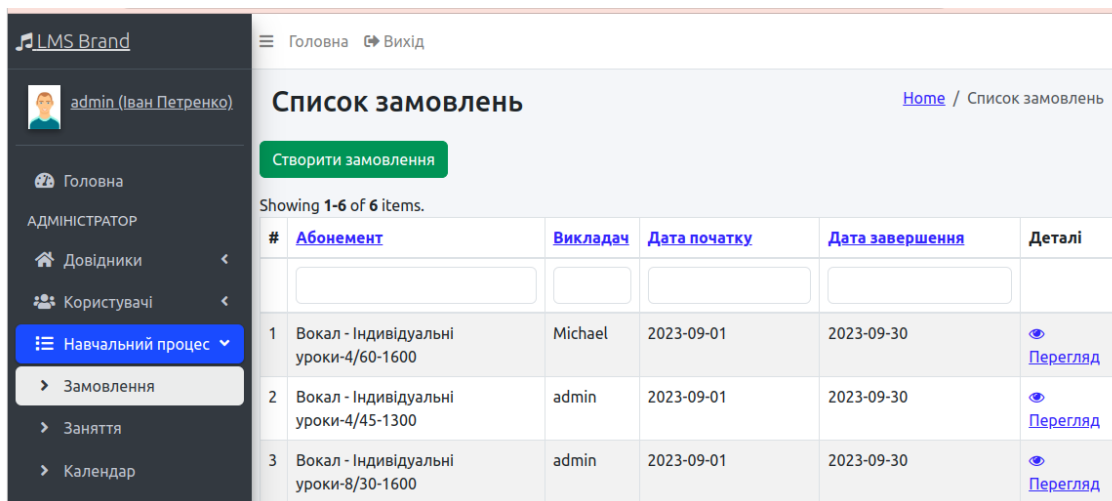


Рис. 3.10. Сторінка «Список замовлень»

Сторінка «Деталі замовлення» (див. рис. 3.11) – можливість переглянути загальну інформацію про замовлення (пропозицію, викладача як виконавця, дати початку та завершення). На даній сторінці є можливість додавання студента (або декількох) до даного замовлення, таким чином можна формувати студентські абонементи — записи про те, що даний студент має намір придбати дане замовлення.

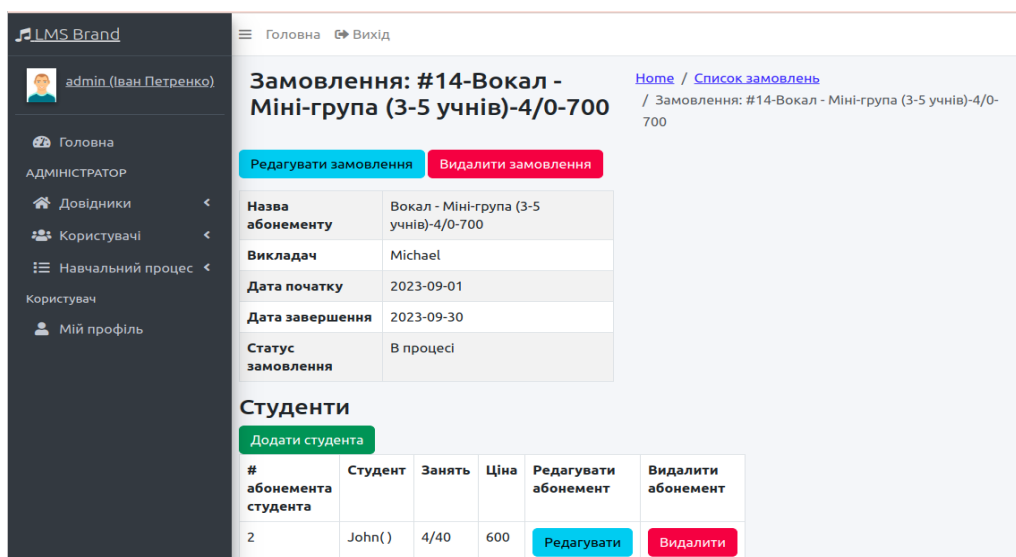
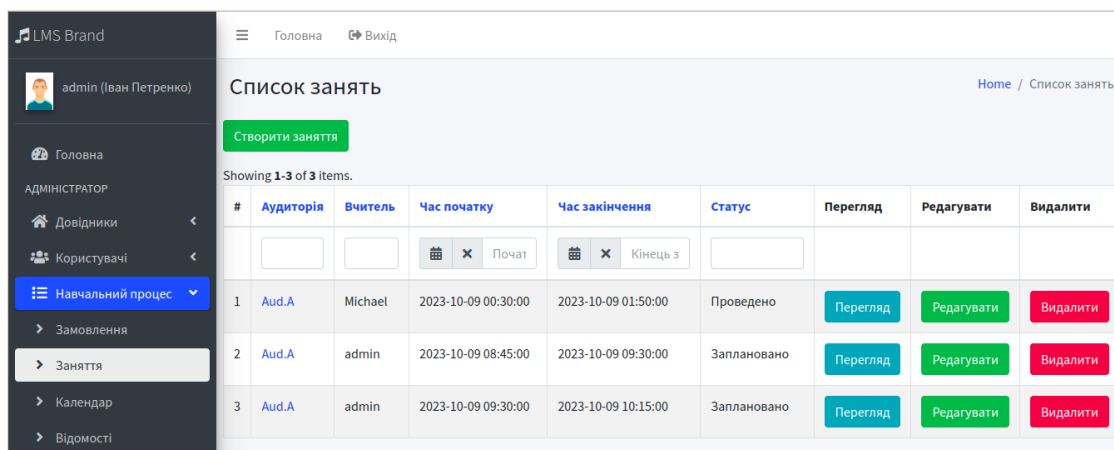


Рис. 3.11. Сторінка «Деталі замовлення»

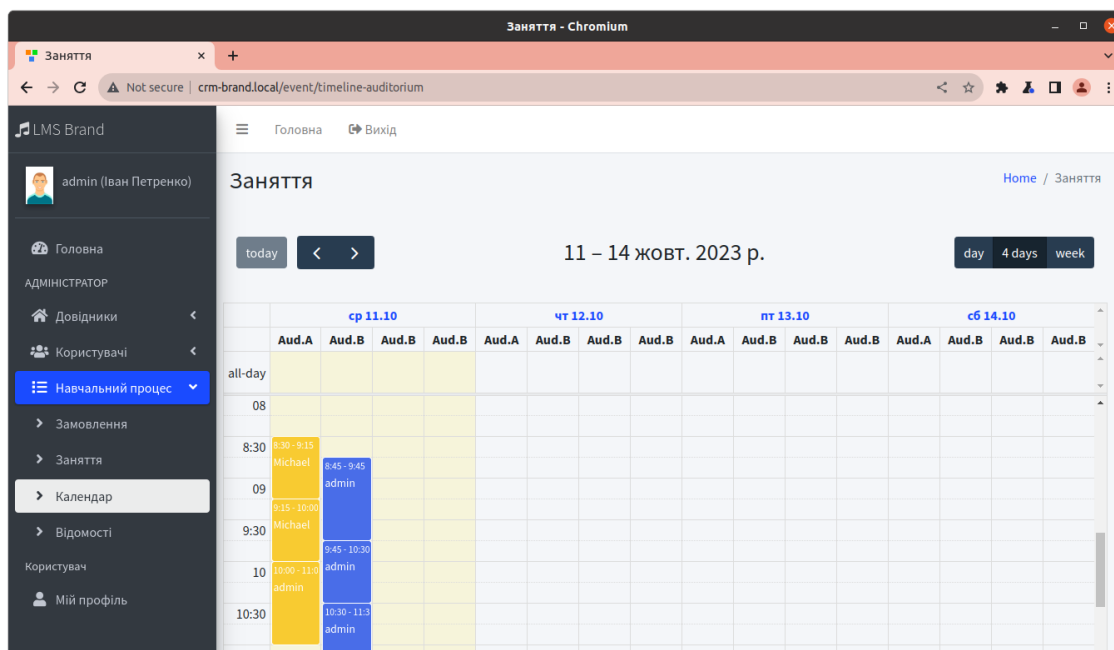
Сторінка «Список занять» (див. рис. 3.12) – перегляд та створення уроків (занять, подій), що мають часовий проміжок, відносяться до певного викладача та певної аудиторії. Кожна подія може асоціюватись з 1-м або декількома студентськими абонементом: таким чином студенти відносяться до створених занять.



#	Аудиторія	Вчитель	Час початку	Час закінчення	Статус	Перегляд	Редагувати	Видалити
1	Aud.A	Michael	2023-10-09 00:30:00	2023-10-09 01:50:00	Проведено	Перегляд	Редагувати	Видалити
2	Aud.A	admin	2023-10-09 08:45:00	2023-10-09 09:30:00	Заплановано	Перегляд	Редагувати	Видалити
3	Aud.A	admin	2023-10-09 09:30:00	2023-10-09 10:15:00	Заплановано	Перегляд	Редагувати	Видалити

Рис. 3.12. Сторінка «Список занять»

Сторінка «Календар занять» (див. рис. 3.13) – вивід розкладу уроків в вигляді календаря з можливістю легкого перегляду та швидкої оцінки розкладу. Календар має також можливість створювати уроки (заняття) (рис. 3.14).



	ср 11.10			чт 12.10				пт 13.10				сб 14.10			
	Aud.A	Aud.B	Aud.B	Aud.A	Aud.B	Aud.B	Aud.B	Aud.A	Aud.B	Aud.B	Aud.B	Aud.A	Aud.B	Aud.B	Aud.B
all-day															
08															
8:30	8:30 - 9:15 Michael	8:45 - 9:45 admin													
09	9:15 - 10:30 Michael														
9:30		9:45 - 10:30 admin													
10	10:00 - 11:0 admin														
10:30		10:30 - 11:3 admin													
11															

Рис. 3.13. Сторінка “Календар занять”

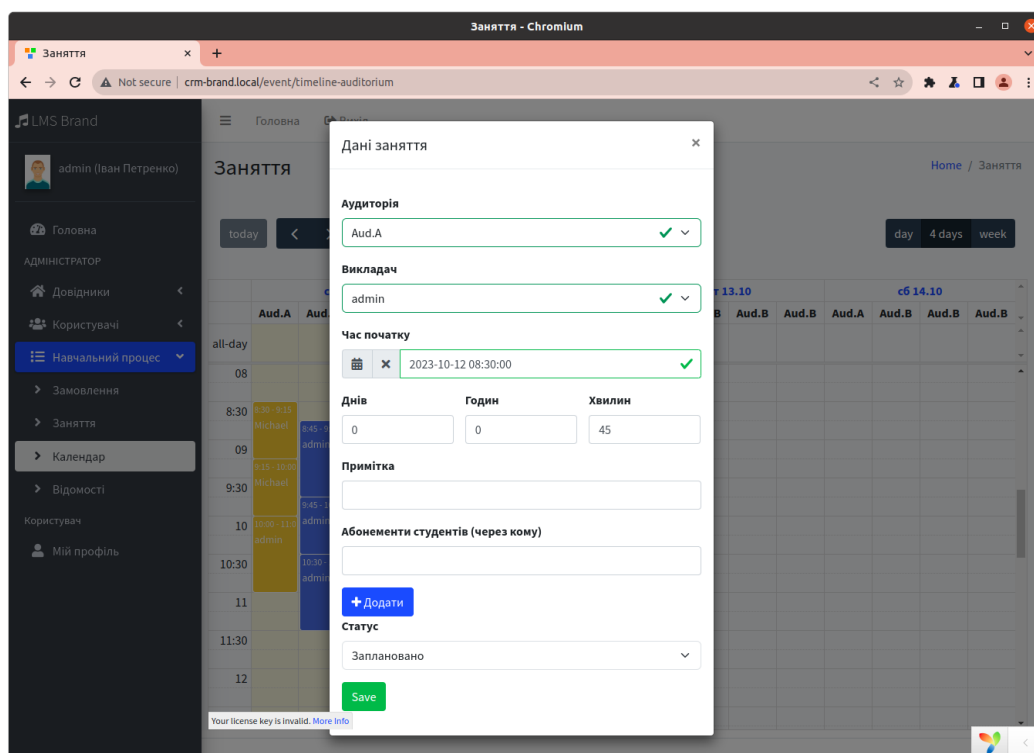


Рис. 3.14. Модальне вікно «Створення/редагування заняття» на сторінці «Календар занять»

Сторінка «Генерація та формування звіту» (див. рис. 3.15) містить форму з полями: випадний список для вибору потрібного викладача та поле для вибору дати в режимі вибору місяця.

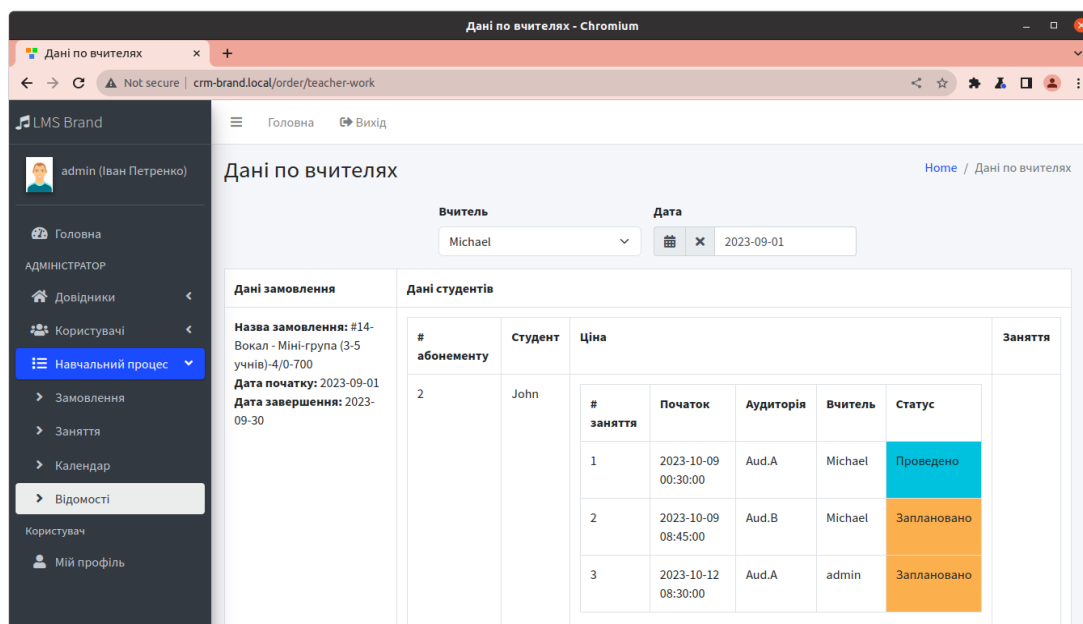


Рис. 3.15. Сторінка для генерації звіту по викладачу за вибраний місяць.

Вивід даних звіту групується по замовленнях. Для кожного замовлення виводяться студентські абонементи, а для кожного абонементу формується таблиця з даними про проведені заняття/уроки.

В якості основної теми веб-інтерфейсу було обрано модуль Yii2 AdminLTE, що імплементує відомий шаблон для адміністрування веб-орієнтованих систем. На лівій бічній панелі всіх сторінок містяться функції, доступні для зареєстрованих користувачів, в основній області сторінок – функції, що стосуються конкретного довідника, користувача чи процесу навчання. Таким чином, нами було створено інструкцію користувача системи, проілюстровано та здійснено опис основних сторінок веб-інтерфейсу.

3.3. Проведення та результати тестування веб-системи

Тестування забезпечення якості (QA) веб-орієнтованої системи планування навчального процесу – важливий і необхідний етап розвитку системи. Основна мета тестування – переконатися у функціонуванні системи належним чином, зручності користування та відповідності вимогам користувачів: студентів, викладачів та адміністраторів. Процес перевірки якості такої системи може містити послідовність таких етапів:

1) Аналіз вимог. Тестування можна розпочати з деталізованого вивчення вимог і специфікацій системи для чіткого розуміння того, що система має робити та як вона повинна працювати.

2) Планування процесу тестування. Пропонується розробка всебічного плану тестування, який окреслює обсяг, цілі, стратегії тестування та ресурси, необхідні для тестування якості. Визначення типів тестів, які планується проводити: перевірка функціональності, зручності використання, продуктивності, безпеки та сумісності.

3) Налаштування тестового середовища. На даному етапі необхідно підготувати середовище тестування, включаючи необхідне обладнання,

програмне забезпечення та мережеві конфігурації, які максимально імітують робоче середовище.

4) Розробка тестів – створення тестів, які охоплюють усі аспекти системи планування та організація навчання: реєстрація користувачів, планування курсів, розподіл навантаження, створення навчального розкладу. Кожен тестовий приклад повинен містити вхідні дані, очікувані результати та чіткий набір кроків для виконання тесту.

5) Функціональне тестування. Рекомендовано розпочати саме з такого типу, щоб переконатися, що всі системні компоненти та функції працюють належним чином. Перевірка коректності введення та пошуку даних, автентифікацію користувача та здатність створювати, редагувати та видаляти навчальні плани.

6) Тестування зручності використання. Даний тип передбачає оцінку інтерфейсу користувача, навігації та загального досвіду користувача. Збір відгуків від потенційних користувачів для визначення модулів чи компонентів, які потрібно покращити.

7) Тестування продуктивності для оцінки її масштабованості та чутливості, що включає тестування навантаження, стрес-тестування, і порівняльний аналіз для порівняння її продуктивності з визначеними показниками.

8) Тестування безпеки: визначення потенційних вразливостей безпеки шляхом тестування на проникнення, сканування вразливостей і перевірку коду, щоб переконатися, що конфіденційні дані захищені, а система стійка до типових загроз безпеці.

9) Тестування на сумісність: перевірка роботи системи на різних веб-браузерах, операційних системах і пристроях, перевірка сумісності системи з різними розмірами екрану та роздільною здатністю.

10) Регресійне тестування – перевірка роботи системи при впровадженні нових функцій чи виправлень для уникнення проблем і стабільної роботи наявних функцій.

11) Звітність щодо тестування – підсумки процесу тестування, що включають результати тестування, виявлені проблеми та рекомендації щодо покращення.

Наведемо деякі результати тестування системи. Результати тестування під користувачем «Гість» предствлено у таблиці 3.1.

Таблиця 3.1

Запити під користувачем «Гість» (користувач, що не здійснив аутентифікацію)

<i>Перевірка доступу для реєстрації та входу</i>		
Сторінка реєстрації	/user/signup	Статус 200. Форма для реєстрації користувача
Реєстрація	POST /user/signup	Статус 201. Перенаправлення на сторінку входу. Успішне повідомлення про реєстрацію
Сторінка входу	/user/login	Статус 200. Форма для входу користувача
Вхід	POST /user/login	Статус 201. Перенаправлення на головну сторінку. Успішне повідомлення про вхід
<i>Всі інші запити під гостем повинні бути недоступні:</i>		
Головна	site/index	Перенаправлення на сторінку входу
Список користувачів	user/index	Перенаправлення на сторінку входу
Перегляд даних про користувача	user/view?id=1	Перенаправлення на сторінку входу

Запити під зареєстрованим користувачем, який не є адміністратором
представлено у таблиці 3.2

Таблиця 3.2

Запити під зареєстрованим користувачем, який не є адміністратором

<i>Вхід та робота з користувачами та їх профілями</i>		
Сторінка входу	/user/login	Перенаправлення на головну сторінку
Мій профіль	/user/my-profile	Сторінка з власною інформацією
Сторінка “Користувачі”	/user/index	Сторінка зі списком всіх користувачів без кнопки “Додати”
Перегляд профілю іншого користувача	/user/view/?id={id}	Сторінка перегляду профілю користувача без кнопки “Редагувати”
Перегляд профілю поточного користувача	/user/view/?id={id}	Сторінка перегляду профілю користувача з кнопкою “Редагувати”
Редагування профілю іншого користувача	/user/update?id=3	Статус 403. Помилка доступу
Редагування власного профілю	/user/update?id=2	Сторінка з формою для редагування власного профілю
Редагування власного профілю (відправка)	POST /user/update?id=2	Перенаправлення на сторінку перегляду власного профілю. Успішне повідомлення про оновлення даних

Реакція системи на запити від користувачем «Адміністратор»
представлено у таблиці 3.3.

Запити під адміністратором

<i>CRUD по роботі з аудиторіями</i>		
Список аудиторій	auditorium/index	Сторінка з даними про аудиторії
Перегляд аудиторії	auditorium/view/?id={auditorium_id}	Детальна інформація про аудиторію
Додавання аудиторії	auditorium/create	Форма для додавання аудиторії
Відправка даних для додавання аудиторії	POST auditorium/create	Статус 201. Перенаправлення на список аудиторій. Успішне повідомлення про додавання
Оновлення даних про аудиторію	auditorium/update	Форма для оновлення аудиторії
Відправка даних оновлення про аудиторію	POST auditorium/update	Перенаправлення на сторінку аудиторії. Успішне повідомлення про додавання
<i>CRUD по роботі зі студентами</i>		
Список студентів	user-student/index	Сторінка зі списком студентів
Додавання студента	user-student/create	Форма з вибором користувачів для додавання студента
Відправка даних при додаванні студента	POST user-student/create	Статус 201. Перенаправлення на список студентів. Успішне повідомлення про додавання
<i>CRUD по роботі із викладачами</i>		
Список викладачів	user-teacher/index	Сторінка зі списком викладачів
Додавання викладача	user-teacher/create	Форма з вибором користувачів для додавання викладача
Відправка даних при додаванні викладача	POST user-teacher/create	Статус 201. Перенаправлення на список викладачів. Успішне повідомлення про додавання

Таким чином, нами було здійснено тестування забезпечення якості веб-орієнтованої системи планування навчання, що включає функціональне тестування, тестування продуктивності, зручності використання, сумісності, безпеки. Варто зазначити, що тестування якості веб-орієнтованої системи планування навчання є безперервним і необхідним процесом для розвитку системи та підтримки її якості і надійності.

Висновки до розділу 3

В даному розділі розроблено інструкцію користувача, наведено ілюстрації та опис сторінок графічного інтерфейсу, а також запропоновано послідовність кроків розгортання системи для локального або віддаленого користування. Запропоновано орієнтовний план процесу тестування забезпечення якості системи планування навчальним процесом. Проведено тестування системи, що включає функціональне тестування, тестування продуктивності, зручності використання, сумісності, безпеки. Система працює в робочому режимі та в повній мірі відповідає поточним вимогам користувачів.

ВИСНОВКИ

В кваліфікаційній роботі магістра виконано розробку веб-орієнтованої інформаційної системи для планування і організації навчального процесу. На першому етапі здійснено постановку задачі, визначено план розробки, завдання та функціональні особливості веб-орієнтованої системи, що сприятиме процесу розробки веб-системи для планування та організації процесу навчання у відповідності до потреб користувачів та навчального закладу.

Проведено порівняння функціональних можливостей сучасних поширених LMS-систем. Зроблено висновок про спільні та унікальні функції кожної системи та їх варіанти використання для різних типів освітніх установ та потреб користувачів. Обґрунтовано вибір мови програмування PHP та Yii2-фреймворку, а також СУБД MySQL для розробки системи.

Запропоновано математичну модель для планування індивідуального розкладу занять.

Здійснено детальний опис функціональних можливостей веб-орієнтованої системи для планування навчальних занять. Розроблено та наведено UML-діаграми варіантів використання, активностей, послідовності та класів, необхідні при проектуванні архітектури системи та схеми бази даних, а також при безпосередній розробці та користуванню системою. Здійснено проектування структури реляційної бази даних. Наведено зображення схеми бази даних та описано структуру кожної таблиці.

Описано план розробки системи, враховуючи можливості і підходи, що пропонуються сучасним PHP-фреймворком. Наведено кроки програмної реалізації щодо обробки запитів на генерацію та виведення звітів про виконання замовлень та проведення занять по викладачам за певний місяць.

Розроблено інструкцію користувача, наведено ілюстрації та опис сторінок графічного інтерфейсу, а також запропоновано послідовність кроків розгортання системи для локального або віддаленого користування. Наведено кроки щодо тестування системи планування навчальним процесом. Проведено тестування

системи, що включає функціональне тестування, тестування продуктивності, зручності використання, сумісності, безпеки. Система працює в робочому режимі та в повній мірі відповідає поточним вимогам користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Awang, Norkhushaini & Darus, Mohamad. (2012). Evaluation of An Open Source Learning Management System: Claroline. *Procedia - Social and Behavioral Sciences*. 67. 416–426. 10.1016/j.sbspro.2012.11.346.
2. Pontoan, Michelle & Sihotang, Jay & Lompoliu, Erienika. (2023). Information Security Analysis of Online Education Management System using Information Technology Infrastructure Library Version 3. *MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*. 22. 207-216. 10.30812/matrik.v22i2.2474.
3. Alvi A. M., Shaon, M. F. I., Das, P. R. Mustafa M. and M. R. Bari, "Automated course management system," 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), Cambridge, UK, 2017, pp. 161-166, doi: 10.23919/ICITST.2017.8356369.
4. Pan, J. Design on College Sports Course Management System. In *Applied Mechanics and Materials* (Vols. 687–691, pp. 2308–2311). 2014. Trans Tech Publications, Ltd. <https://doi.org/10.4028/www.scientific.net/amm.687-691.2308>.
5. Ekwonwune, E. N., & Edebatu, D. C.. Design and implementation of an online course management system. *Journal of Software Engineering and Applications*, 12(2), 2019. pp. 21-33.
6. Maher, D. The use of course management systems in pre-service teacher education. In *Handbook of research on blended learning pedagogies and professional development in higher education*. IGI Global, 2019. p. 196-213.
7. Huang Y., Rex B. Design and Implementation of the Heartful Education System Platform Based on PHP. *International Journal of Information and Education Technology* vol. 13, no. 12, pp. 1924-1931, 2023.
8. Tubagus, Munir & Muslim, Suyitno & Suriani, Suriani. (2020). Development of Learning Management System-Based Blended Learning Model using Claroline in Higher Education. *International Journal of Interactive Mobile Technologies (iJIM)*. 14. 186. 10.3991/ijim.v14i06.13399.

9. Yaremko, Svitlana & Kuzmina, Olena & BEVZ, Svitlana. (2022). DIRECTIONS OF DEVELOPMENT OF LEARNING MANAGEMENT SYSTEMS. Herald of Khmelnytskyi National University. 302. 252-255. 10.31891/2307-5740-2022-302-1-42.

10. Захаров Д.М., Лобанчикова Н.М. Особливості розробки веб-систем для планування та організації процесу. II Всеукраїнська науково-технічна конференція «Комп'ютерні технології: інновації, проблеми, рішення», 23-24 листопада 2023 року. Житомир : «Житомирська політехніка», 2023.

11. Яремко С. А., Кузьміна О. М. Управління бізнес-процесами підприємства засобами інтелектуальних систем. Вісник Хмельницького національного університету. 2018, № 4. С. 43–52.

12. Шумейко Н. Інформаційні системи керування навчальним процесом у вищій школі. [Електронний ресурс] - Режим доступу до ресурсу: <http://www.iconfs.net/en/infocom2016/%D0%86nformatsijni-systemy-keruvannya-navchalnym-protsesom-u-vyschij-shkoli>.

13. Франчук В.М. Використання web-орієнтованих комп'ютерних систем в освітньому закладі// Вища освіта України: теоретичний та науковометодичний часопис. 3 (54) – Тематичний випуск "Педагогіка вищої школи: методологія, теорія, технології", Додаток 2, том 2, 2014 р. – Київ-Кіровоград С.12–14.

14. VorozhbytA. (2018). Web-oriented information-education environment of the institution of general secondary education. Journal of Information Technologies in Education (ITE), (36), 020-029. <https://doi.org/10.14308/ite000671>.

15. LMS Moodle. Docs [Електронний ресурс] - Режим доступу до ресурсу: https://docs.moodle.org/403/en/Main_page.

16. LMS Blackboard. Docs [Електронний ресурс] - Режим доступу до ресурсу: <https://help.blackboard.com/Learn/Instructor>.

17. LMS Schoology. Docs [Електронний ресурс] - Режим доступу до ресурсу: <https://developers.schoology.com/api>

18. LMS Canvas. Docs [Електронний ресурс] - Режим доступу до ресурсу: <https://canvas.instructure.com/doc/api/>.

19. Буйницька, О. П., Варченко-Троценко, Л. О., Терлецька, Т. С., & Настас, Д. Л. (2020). Модернізація системи електронного навчання університету до потреб учасників освітнього процесу. Електронне наукове фахове видання “Відкрите освітнє е-середовище сучасного університету”, (9), 1–14. <https://doi.org/10.28925/2414-0325.2020.9.1>.

20. Lassoued Z., Alhendawi M., Bashitialshaaer R. An Exploratory Study of the Obstacles for Achieving Quality in Distance Learning during the COVID-19 Pandemic. *Educ. Sci.* 2020. 10 (9). P. 1–13. URL: <https://www.mdpi.com/2227-7102/10/9/232> (date of access: 14.10.2023).

21. Seridi A., Dib L., Bourbia R. Modernization of e-learning platforms towards a service-oriented architecture. *Journal of Electrical Systems Systems*. 2019. No. 15. P. 123–132. URL: http://journal.esrgroups.org/jes/papers/15_1_10.pdf (date of access: 17.10.2023).

22. Voronin D., Saienko V., Tolchieva H. Digital Transformation of Pedagogical Education at the University. *Proceedings of the International Scientific Conference “Digitalization of Education: History, Trends and Prospects” (DETP 2020)*. 2020. No. 437. P. 757–763. URL: <https://www.atlantispress.com/proceedings/detp-20/125940270> (date of access: 01.10.2023).

23. Буйницька О., Варченко-Троценко Л., Грицеляк Б. Цифровізація закладу вищої освіти. *Освітологічний дискурс*. 2020. Т. 1, № 28. С. 64–79.

24. Kholiavko N., Medina T., Goshylyk V. Areas of the modernization of the lithuanian higher education system: a case for ukraine. *Studies in Comparative Education*. 2020. 1 (39). P. 14–25.

URL: <http://pps.udpu.edu.ua/article/view/211313/211358> (date of access: 14.10.2023).

25. Гайтан О.М. Дослідження адаптивних методів складання розкладу навчального процесу в університеті: атест. робота магістра [Електронний ресурс] / Харківський національний університет радіоелектроніки. 2020. 102 с. URL: <https://openarchive.nure.ua/server/api/core/bitstreams/bbe30f60-31c2-49dc-aba6-e953baef6807/content>.

26. Bania Kumar Rubul, Duarah, Pinkey. Exam Time Table Scheduling using Graph Coloring Approach // International Journal of Computer Sciences and Engineering, 2018. №6. Pp. 84-93.

27. Морзе Н., Буйницька О., Варченко-Троценко Л. Створення сучасного електронного курсу в системі MOODLE: навчальний посібник. Київ : Київський університет імені Бориса Грінченка, 2016. 232 с. URL: https://ec.europa.eu/programmes/erasmus-plus/project-result-content/e8b0e789-8dee-4ab5-a48e-96d153921d2e/book_Moodle.pdf (дата звернення: 14.10.2023).

28. Морзе Н., Непреєнко Л., Смирнова-Трибульська Є. Шляхи розвитку smart-університету (Досвід київського університету імені Бориса Грінченка). Електронне наукове фахове видання “Відкрите освітнє е-середовище сучасного університету”. 2020. № 8. С. 191–205.

URL: <https://openedu.kubg.edu.ua/journal/index.php/openedu/article/view/2414-0325.2020.8.17> (дата звернення: 07.10.2023).

29. Leite Nuno, Melicio Fernando, Rosa Agostinho. A fast simulated annealing algorithm for the examination timetabling problem // Expert Systems with Applications, 2018. Vol. 122.

30. PHP: Hypertext Preprocessor. Docs [Електронний ресурс] - Режим доступу до ресурсу: <https://www.php.net/docs.php>.

31. Yii2 PHP Framework. Guide [Електронний ресурс] - Режим доступу до ресурсу: <https://www.yiiframework.com/doc/guide/2.0/en>.

32. JS FullCalendar Docs. [Електронний ресурс] - Режим доступу до ресурсу: <https://fullcalendar.io/docs>.

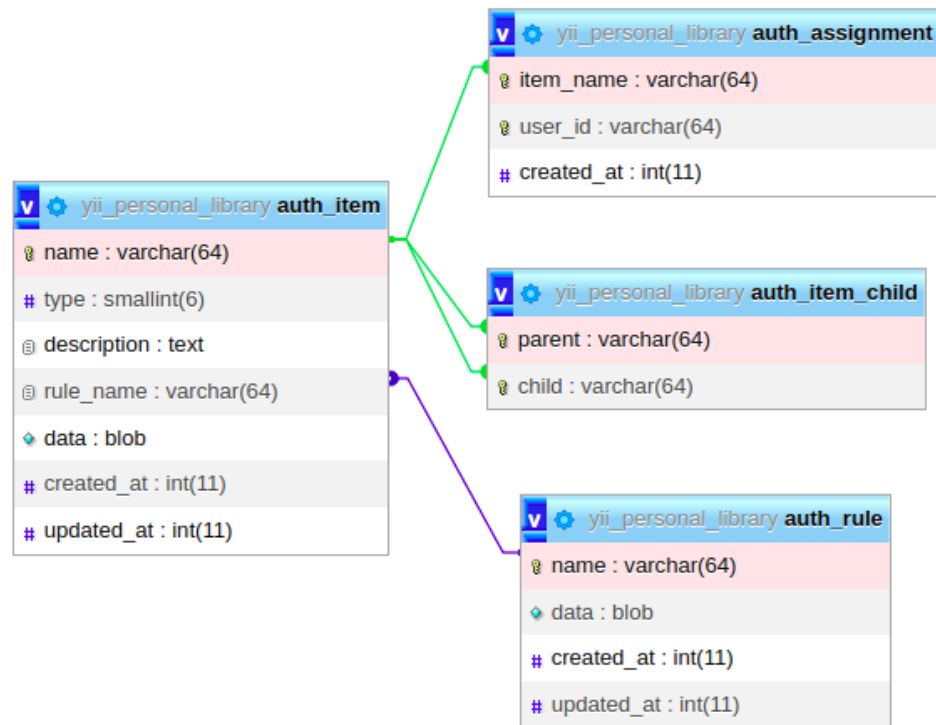
33. CSS Framework Bootstrap. Docs [Електронний ресурс] - Режим доступу до ресурсу: <https://getbootstrap.com/docs/>.

34. Haitan O. Гібридний підхід до розв’язання задачі автоматизованого складання розкладу вищого навчального закладу [Електронний ресурс] / Haitan O., Nazarov O. // Системи управління, навігації та зв’язку. Збірник наукових праць. – Полтава: ПНТУ, 2020. – Т. 2 (60). – С. 60-69. – doi: <https://doi.org/10.26906/SUNZ.2020.2.060>.

35. ITC 2019: International Timetabling Competition [Електронний ресурс]. URL: <https://www.itc2019.org/home> (дата звернення: 10.12.2023).
36. Устенко С.В., Бібко О.О. Використання методу мурашиної колонії для розв'язання оптимізаційних задач // Науковий вісник НЛТУ України. Вип. 25.3, 2015. С. 351-359.
37. Shifton - програма для складання розкладів та змін онлайн: автоматичне та ручне складання графіків роботи [Електронний ресурс] URL: <https://shifton.com/uk/shift-scheduling>.
38. Сікора О.С. Розробка автоматизованої системи «Електронний розклад занять» для середовища електронного навчання на базі LMS ATutor: кваліфікаційна робота магістра за спеціальністю «151 – автоматизація та комп'ютерно-інтегровані технології». Тернопіль: ТНТУ, 2020. 52 с.
URL: <http://elartu.tntu.edu.ua/handle/lib/33306>.
39. Димова, Г. О. (2022). РОЗРОБКА МОДЕЛІ СКЛАДАННЯ РОЗКЛАДУ ЗАНЯТЬ МЕТОДОМ ЕВОЛЮЦІЙНОГО ПОШУКУ. Таврійський науковий вісник. Серія: Технічні науки, (2), 3-9. <https://doi.org/10.32851/tnv-tech.2022.2.1/>
40. Теорія розкладів. 2022. URL: <https://uk.wikipedia.org/wiki/> (дата звернення 20.06.22).

ДОДАТКИ

Таблиці для зберігання даних авторизації: ролей, дозволів, їх відношень, призначень та правил.



JS-функція для ініціалізації та рендерингу розкладу занять засобами інтерактивного календаря

```
function renderCalendar() {
    var calendarEl = document.getElementById('calendar');
    var modalForm = document.getElementById('modal-form');

    var calendar = new FullCalendar.Calendar(calendarEl, {

        selectable: true,
        slotDuration: '00:15:00',
        locale: 'uk',
        now: '<?= date("Y-m-d"); ?>',
        editable: true,
        aspectRatio: 1.8,
        scrollTime: '8:00',
        headerToolbar: {
            left: 'today prev,next',
            center: 'title',
            right: 'resourceTimeGridDay,resourceTimeGridFourDay,resourceTimeGridWeek'
        },
        initialView: 'resourceTimeGridFourDay', // resourceTimelineDay
        datesAboveResources: true,
        views: {
            resourceTimeGridFourDay: {
                type: 'resourceTimeGrid',
                duration: {days: 4},
                buttonText: '4 days'
            }
        },
        navLinks: true,
        resourceAreaWidth: '10%',
        resourceAreaHeaderContent: 'Аудиторії',
        resources: '<?= $resourcesJSON; ?>',
        events: '<?= $eventsJSON; ?>',
        select: function (info) {
            //alert('selected ' + info.startStr + ' to ' + info.endStr);
            // $('#modal-info').html(JSON.stringify(info));
            $('#modal-form #event-id').val("");
            $('#modal-form #event-auditorium-id').val(info.resource.id);
            let dateStart = new Date(info.startStr);
            let dateEnd = new Date(info.endStr);

            let [year, month, day, hours, minutes] = [
                dateStart.getFullYear(),
                ('0' + (dateStart.getMonth() + 1)).slice(-2),
                ('0' + dateStart.getDate()).slice(-2),
                ('0' + dateStart.getHours()).slice(-2),
                ('0' + dateStart.getMinutes()).slice(-2),
            ];

            let mseconds = dateEnd - dateStart;
            let msecondsInMinute = 60000;
            let msecondsInHour = msecondsInMinute * 60;
            let msecondsInDay = msecondsInHour * 24;
            let daysDiff = Math.floor(mseconds / msecondsInDay);
            let hoursDiff = Math.floor(mseconds % msecondsInDay / msecondsInHour);
            let minutesDiff = Math.floor(mseconds % msecondsInHour / msecondsInMinute);

            $('#modal-form #event-datetime-start').val(`${year}-${month}-${day}
            ${hours}:${minutes}:00`);
            $('#modal-form #event-durationDays').val(daysDiff);
            $('#modal-form #event-durationHours').val(hoursDiff);
            $('#modal-form #event-durationMinutes').val(minutesDiff);
        }
    });
}
```

```

        $('#modal-form #event-orderStudentJSON').val('');
        $('#modal-form').modal('show');
    },
    eventClick: function (info) {
        var eventDate = info.event.start;
        console.log(JSON.stringify(info));
        $('#modal-form #event-id').val(info.event.id);
        $('#modal-form #event-auditorium-
id').val(info.event.extendedProps.auditorium_id);
        $('#modal-form #event-teacher-id').val(info.event.extendedProps.teacher_id);
        let dateStart = new Date(info.event.start);
        let dateEnd = new Date(info.event.end);

        let [year, month, day, hours, minutes] = [
            dateStart.getFullYear(),
            ('0' + (dateStart.getMonth() + 1)).slice(-2),
            ('0' + dateStart.getDate()).slice(-2),
            ('0' + dateStart.getHours()).slice(-2),
            ('0' + dateStart.getMinutes()).slice(-2),
        ];

        let mseconds = dateEnd - dateStart;
        let msecondsInMinute = 60000;
        let msecondsInHour = msecondsInMinute * 60;
        let msecondsInDay = msecondsInHour * 24;
        let daysDiff = Math.floor(mseconds / msecondsInDay);
        let hoursDiff = Math.floor(mseconds % msecondsInDay / msecondsInHour);
        let minutesDiff = Math.floor(mseconds % msecondsInHour / msecondsInMinute);

        $('#modal-form #event-datetime-start').val(`${year}-${month}-${day}
${hours}:${minutes}:00`);
        $('#modal-form #event-durationDays').val(daysDiff);
        $('#modal-form #event-durationHours').val(hoursDiff);
        $('#modal-form #event-durationMinutes').val(minutesDiff);

        $('#modal-form #event-
orderStudentJSON').val(info.event.extendedProps.orderStudentJSON);

        $('#modal-form #event-status').val(info.event.extendedProps.status);
        $('#modal-form').modal('show');
    },
});

calendar.render();
}

```

SQL-дамп бази даних для створення таблиць, індексів та зв'язків

```

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";

-- Table structure for table `auditorium`
CREATE TABLE `auditorium` ( `id` int(11) NOT NULL, `name` varchar(255) NOT NULL, `color`
varchar(50) NOT NULL DEFAULT 'blue') ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Table structure for table `course`
CREATE TABLE `course` ( `id` int(11) NOT NULL, `name` varchar(255) NOT NULL, `category_id`
int(11) NOT NULL) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Table structure for table `course_category`
CREATE TABLE `course_category` ( `id` int(11) NOT NULL, `name` varchar(255) NOT NULL)
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Table structure for table `event`
CREATE TABLE `event` ( `id` int(11) NOT NULL, `auditorium_id` int(11) NOT NULL,
`teacher_id` int(11) NOT NULL, `datetime_start` datetime NOT NULL, `datetime_end`
datetime NOT NULL, `note` varchar(255) NOT NULL DEFAULT '', `status` tinyint(2) NOT NULL
DEFAULT 1) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Table structure for table `event_order_student`
CREATE TABLE `event_order_student` ( `event_id` int(11) NOT NULL, `order_student_id`
int(11) NOT NULL, `status` tinyint(2) NOT NULL DEFAULT 1) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4;

-- Table structure for table `order`
CREATE TABLE `order` ( `id` int(11) NOT NULL, `subscription_id` int(11) NOT NULL,
`teacher_id` int(11) NOT NULL, `date_start` date NOT NULL, `date_end` date NOT NULL,
`status` int(2) NOT NULL DEFAULT 1) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Table structure for table `order_student`
CREATE TABLE `order_student` ( `id` int(11) NOT NULL, `order_id` int(11) NOT NULL,
`student_id` int(11) NOT NULL, `order_student_number` int(11) NOT NULL, `event_duration`
int(11) NOT NULL DEFAULT 0, `event_count` int(11) NOT NULL DEFAULT 1, `price` int(11) NOT
NULL, `status` tinyint(2) NOT NULL DEFAULT 1) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Table structure for table `subscription`
CREATE TABLE `subscription` ( `id` int(11) NOT NULL, `course_id` int(11) NOT NULL,
`event_duration` int(11) NOT NULL DEFAULT 0, `event_count` int(11) NOT NULL DEFAULT 1,
`student_count` int(11) NOT NULL DEFAULT 1, `price` int(11) NOT NULL) ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4;

-- Table structure for table `user`
CREATE TABLE `user` ( `id` int(11) NOT NULL, `username` varchar(255) COLLATE
utf8_unicode_ci NOT NULL, `auth_key` varchar(32) COLLATE utf8_unicode_ci NOT NULL,
`password_hash` varchar(255) COLLATE utf8_unicode_ci NOT NULL, `password_reset_token`
varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL, `email` varchar(255) COLLATE
utf8_unicode_ci NOT NULL, `phone` varchar(50) COLLATE utf8_unicode_ci DEFAULT NULL,
`status` smallint(6) NOT NULL DEFAULT 10, `created_at` int(11) NOT NULL, `updated_at`
int(11) NOT NULL, `verification_token` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
`first_name` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL, `last_name` varchar(255)
COLLATE utf8_unicode_ci DEFAULT NULL, `image` varchar(255) COLLATE utf8_unicode_ci
DEFAULT NULL, `color` varchar(50) COLLATE utf8_unicode_ci NOT NULL DEFAULT 'blue')
ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- Table structure for table `user_student`
CREATE TABLE `user_student` ( `id` int(11) NOT NULL, `user_id` int(11) NOT NULL)
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Table structure for table `user_teacher`
CREATE TABLE `user_teacher` (
`id` int(11) NOT NULL,
`user_id` int(11) NOT NULL

```

```

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Indexes for table `auditorium`
ALTER TABLE `auditorium`
  ADD PRIMARY KEY (`id`);

-- Indexes for table `auth_assignment`
ALTER TABLE `auth_assignment`
  ADD PRIMARY KEY (`item_name`,`user_id`),
  ADD KEY `idx-auth_assignment-user_id` (`user_id`);

-- Indexes for table `auth_item`
ALTER TABLE `auth_item`
  ADD PRIMARY KEY (`name`),
  ADD KEY `rule_name` (`rule_name`),
  ADD KEY `idx-auth_item-type` (`type`);

-- Indexes for table `auth_item_child`
ALTER TABLE `auth_item_child`
  ADD PRIMARY KEY (`parent`,`child`),
  ADD KEY `child` (`child`);

--
-- Indexes for table `auth_rule`
ALTER TABLE `auth_rule`
  ADD PRIMARY KEY (`name`);

-- Indexes for table `course`
ALTER TABLE `course`
  ADD PRIMARY KEY (`id`),
  ADD KEY `category_id` (`category_id`);

-- Indexes for table `course_category`
ALTER TABLE `course_category`
  ADD PRIMARY KEY (`id`);

-- Indexes for table `event`
ALTER TABLE `event`
  ADD PRIMARY KEY (`id`),
  ADD KEY `auditorium_id` (`auditorium_id`),
  ADD KEY `teacher_id` (`teacher_id`),
  ADD KEY `status` (`status`);

-- Indexes for table `event_order_student`
ALTER TABLE `event_order_student`
  ADD UNIQUE KEY `event_id_2` (`event_id`,`order_student_id`),
  ADD KEY `event_id` (`event_id`),
  ADD KEY `order_student_id` (`order_student_id`),
  ADD KEY `status` (`status`);

-- Indexes for table `migration`
ALTER TABLE `migration`
  ADD PRIMARY KEY (`version`);

-- Indexes for table `order`
ALTER TABLE `order`
  ADD PRIMARY KEY (`id`),
  ADD KEY `teacher_id` (`teacher_id`),
  ADD KEY `subscription_id` (`subscription_id`),
  ADD KEY `status` (`status`);

-- Indexes for table `order_student`
ALTER TABLE `order_student`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `subscription_order_id_2` (`order_id`,`student_id`),
  ADD UNIQUE KEY `order_student_number` (`order_student_number`),
  ADD KEY `subscription_order_id` (`order_id`),
  ADD KEY `student_id` (`student_id`),
  ADD KEY `status` (`status`);

-- Indexes for table `subscription`

```

```

ALTER TABLE `subscription`
  ADD PRIMARY KEY (`id`),
  ADD KEY `course_id` (`course_id`);

-- Indexes for table `user`
ALTER TABLE `user`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `username` (`username`),
  ADD UNIQUE KEY `email` (`email`),
  ADD UNIQUE KEY `password_reset_token` (`password_reset_token`);

-- Indexes for table `user_student`
ALTER TABLE `user_student`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `user_id` (`user_id`);

-- Indexes for table `user_teacher`
ALTER TABLE `user_teacher`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `user_id` (`user_id`);

-- AUTO_INCREMENT for table `auditorium`
ALTER TABLE `auditorium`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;

-- AUTO_INCREMENT for table `course`
ALTER TABLE `course`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=13;

-- AUTO_INCREMENT for table `course_category`
ALTER TABLE `course_category`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;

-- AUTO_INCREMENT for table `event`
ALTER TABLE `event`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=108;

-- AUTO_INCREMENT for table `order`
ALTER TABLE `order`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=16;

-- AUTO_INCREMENT for table `order_student`
ALTER TABLE `order_student`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=25;

-- AUTO_INCREMENT for table `subscription`
ALTER TABLE `subscription`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=47;

-- AUTO_INCREMENT for table `user`
ALTER TABLE `user`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=14;

-- AUTO_INCREMENT for table `user_student`
ALTER TABLE `user_student`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=18;

-- AUTO_INCREMENT for table `user_teacher`
ALTER TABLE `user_teacher`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=8;

--
-- Constraints for dumped tables
--

ALTER TABLE `course`
  ADD CONSTRAINT `course_ibfk_1` FOREIGN KEY (`category_id`) REFERENCES `course_category`
  (`id`);

ALTER TABLE `event`
  ADD CONSTRAINT `event_ibfk_1` FOREIGN KEY (`auditorium_id`) REFERENCES `auditorium`
  (`id`);

```

```

    ADD CONSTRAINT `event_ibfk_2` FOREIGN KEY (`teacher_id`) REFERENCES `user_teacher`
    (`id`);

ALTER TABLE `event_order_student`
    ADD CONSTRAINT `event_order_student_ibfk_1` FOREIGN KEY (`event_id`) REFERENCES `event`
    (`id`),
    ADD CONSTRAINT `event_order_student_ibfk_2` FOREIGN KEY (`order_student_id`) REFERENCES
    `order_student` (`id`);

ALTER TABLE `order`
    ADD CONSTRAINT `order_ibfk_1` FOREIGN KEY (`subscription_id`) REFERENCES `subscription`
    (`id`),
    ADD CONSTRAINT `order_ibfk_2` FOREIGN KEY (`teacher_id`) REFERENCES `user_teacher`
    (`id`);

ALTER TABLE `order_student`
    ADD CONSTRAINT `order_student_ibfk_1` FOREIGN KEY (`order_id`) REFERENCES `order`
    (`id`),
    ADD CONSTRAINT `order_student_ibfk_2` FOREIGN KEY (`student_id`) REFERENCES
    `user_student` (`id`);

ALTER TABLE `subscription`
    ADD CONSTRAINT `subscription_ibfk_1` FOREIGN KEY (`course_id`) REFERENCES `course`
    (`id`);

ALTER TABLE `user_student`
    ADD CONSTRAINT `user_student_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user` (`id`);

ALTER TABLE `user_teacher`
    ADD CONSTRAINT `user_teacher_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user` (`id`);
COMMIT;

```