Полтавський університет економіки і торгівлі
Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп’ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____Олена ОЛЬХОВСЬКА
(підпис)

«____»_____2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему
«РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ЕЛЕКТРОННОГО ЖУРНАЛУ ВИКЛАДАЧА ІЗ РОЗШИРЕНИМ ФУНКЦІОНАЛОМ АНАЛІТИКИ ТА ЗВІТНОСТІ»

зі спеціальності 122 «Комп’ютерні науки»
освітня програма «Комп’ютерні науки»
ступеня бакалавр

Виконавець роботи Крамаренко Костянтин Геннадійович

_____«__»_____202_р.
(підпис)

Науковий керівник к.ф.-м.н., доц., Парфьонова Тетяна Олександрівна

_____«__»_____202_р.
(підпис)

Рецензент _____

ПОЛТАВА 2026

ЗМІСТ

ВСТУП

Цифрова трансформація освітнього середовища, посилює вимоги до якості інформаційного забезпечення навчального процесу, оперативності обробка академічних даних та повноти аналітичної підтримки управлінських рішень. Робота викладача вже давно не обмежується лише фіксацією поточних оцінок, відміток про присутність та підсумкових результатів навчання, оскільки професійна діяльність в сучасному закладі освіти, пов’язана з постійним моніторингом динаміки успішності, виявленням проблемних тенденцій, підготовкою звітних матеріалів для адміністрації, студентів та батьків, а також, з необхідністю швидкого доступу до структурованої інформації в різних розрізах. Відповідно, особливої ваги набуває розробка програмних засобів, здатних поєднати облік, аналіз, візуалізацію та формування звітності в єдиному інформаційному просторі [1].

Потреба в електронному журналі викладача з розширеним функціоналом аналітики та звітності, зумовлена не лише загальною тенденцією цифровізації освіти, а й практичними труднощами, які виникають під час роботи з фрагментованими програмними рішеннями. В багатьох випадках, журнальні дані зберігаються у відокремлених таблицях, локальних файлах або спеціалізованих сервісах з мінімальним набором функцій, придатних насамперед для реєстрації оцінок та відвідуваності. Подібна організація інформації ускладнює побудову узагальнених висновків щодо академічної активності студентів, не забезпечує належного рівня автоматизації підготовки довідок і звітів, створює додаткове навантаження на викладача під час повторного введення однакових відомостей в різні документи [2].

Актуальність обраної теми визначається необхідністю створення інформаційної системи, яка забезпечує комплексне ведення електронного журналу викладача, підтримує накопичення та структурування академічних даних, виконує аналітичне опрацювання показників успішності та

відвідуваності, автоматизує формування звітних матеріалів для різних категорій користувачів. Освітні установи потребують не лише інструменту збереження даних, а й програмного продукту, здатного допомагати у виявленні проблем навчальної діяльності, порівнянні результатів окремих груп, дисциплін і періодів навчання, визначенні ризиків зниження успішності та підготовці обґрунтованих управлінських висновків. Розробка інформаційної системи має практичну значущість для вдосконалення організації навчального процесу, скорочення часу на рутинні операції та підвищення достовірності підсумкової інформації [3].

Стан розробки програмних засобів електронного обліку в освіті, свідчить про наявність значної кількості окремих рішень, проте не всі наявні продукти однаково добре задовольняють потреби викладача в сфері розширеної аналітики та багатоформатної звітності. Частина систем орієнтована переважно на базову фіксацію оцінок, інша частина акцентує увагу на адміністративному управлінні освітнім середовищем, тоді як інструменти глибшого аналізу навчальних показників, часто потребують окремих модулів, додаткового налаштування або зовнішніх сервісів [4].

Мета роботи полягає в розробці інформаційної системи електронного журналу викладача з розширеним функціоналом аналітики та звітності, призначеної для ведення академічних даних, аналізу показників успішності та відвідуваності, автоматизованого формування звітних матеріалів для підтримки освітнього процесу.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1. дослідити теоретичні основи побудови інформаційних систем навчальної аналітики та з’ясувати роль електронного журналу в структурі освітнього інформаційного середовища;

2. проаналізувати підходи до опрацювання показників успішності та відвідуваності, визначити можливості їх використання в межах програмної системи;

3. сформувати функціональні та нефункціональні вимоги до інформаційної системи електронного журналу викладача;

4. спроєктувати архітектуру програмної системи, інформаційну модель даних та логіку їх обробки;

5. розробити модель функціонування аналітичної підсистеми та механізмів формування звітності;

6. реалізувати основні програмні компоненти інформаційної системи електронного журналу викладача;

7. реалізувати алгоритми аналізу академічних показників та генерації звітних документів;

8. провести експериментальну перевірку роботи системи та виконати аналіз одержаних результатів.

Об’єктом дослідження є процес інформаційного супроводу діяльності викладача, пов’язаної з веденням обліку результатів навчання, контролем відвідуваності, аналізом освітніх показників та підготовкою звітної документації.

Предметом дослідження є методи, моделі, алгоритми та програмні засоби розробки інформаційної системи електронного журналу викладача з розширеними можливостями аналітичного опрацювання академічних даних та формування звітності.

Методологічну основу роботи становлять методи системного аналізу для виявлення структури предметної області та взаємозв’язків між її компонентами, методи аналізу вимог для формалізації функціональних та нефункціональних характеристик програмного продукту, методи моделювання даних для побудови логічної структури інформаційного наповнення системи, методи алгоритмізації для реалізації процедур обробки показників успішності та відвідуваності, методи об’єктно-орієнтованого та структурного проєктування для створення програмної архітектури. Для оцінювання результативності розробленого продукту, використовуються методи експериментальної

перевірки, тестування, порівняльного аналізу та узагальнення отриманих результатів.

Наукова новизна одержаних результатів полягає в розробці інформаційної системи, де функції ведення електронного журналу, аналітичного опрацювання академічних показників і формування звітності інтегровані в рамках єдиної логіки обробки даних.

Практичне значення одержаних результатів полягає в можливості використання розробленої інформаційної системи в діяльності викладачів закладів вищої та фахової передвищої освіти для ведення електронного журналу, збереження та систематизації даних про успішність та відвідуваність, оперативного виявлення проблемних аспектів навчальної діяльності, для підготовки поточних та підсумкових звітів. Програмний продукт може бути основою для впровадження в освітнє середовище, модернізації внутрішніх процедур обліку та подальшого розширення функціоналу, пов’язаного з прогнозуванням академічних результатів, інтеграцією з іншими інформаційними ресурсами та підтримкою ухвалення управлінських рішень.

Структура роботи включає, вступ, чотири розділи, висновки, рекомендації, список літератури та додатки. В першому розділі представлено постановку задачі, розкрито зміст завдання та наведено модель задачі разом з її основними характеристиками. Другий розділ присвячено інформаційному огляду, де проаналізовано наявні роботи, визначено їх позитивні риси, виявлено наявні недоліки та обґрунтовано необхідність розробки за обраною тематикою. В третьому розділі представлено теоретичні аспекти виконання роботи, зокрема проєктування архітектури програмного забезпечення, графічне подання архітектурних рішень та обґрунтування вибору програмних засобів реалізації. Четвертий розділ містить практичну частину, де наведено опис процесу програмної реалізації, характеристику програми, результати перевірки валідності та дослідження можливостей програмної реалізації. В додатках надано програмний код.

1 ПОСТАНОВКА ЗАДАЧІ

1.1 Змістновна постановка

В сучасному освітньому середовищі діяльність викладача супроводжується постійним опрацюванням значного обсягу академічної інформації, пов’язаної з фіксацією результатів навчання, контролем відвідування занять, аналізом поточної успішності та підготовкою різних видів звітної документації. Традиційні форми ведення обліку, що спираються на паперові журнали, електронні таблиці або розрізнені цифрові сервіси, не забезпечують належної цілісності даних, ускладнюють узагальнення відомостей і збільшують витрати часу на повторне введення однакової інформації.

Змістовна постановка задачі полягає в розробці програмного засобу, призначеного для автоматизації основних напрямів роботи викладача з електронним журналом. Йдеться про створення системи, в якій забезпечується ведення відомостей про навчальні групи, дисципліни, студентів, заняття, оцінки, відмітки про присутність, коментарі щодо навчальної активності, а також формування аналітичних узагальнень і звітних матеріалів на основі накопичених записів. Зміст задачі не обмежується перенесенням паперового журналу у цифровий формат, оскільки ключове значення має підтримка процедур аналізу, які дозволяють викладачеві швидко оцінювати стан навчального процесу, виявляти зміни в успішності та отримувати впорядковані результати для подальшого використання [5].

Предметна область включає процес інформаційного супроводу освітньої діяльності в частині поточного та підсумкового обліку результатів навчання. Значення мають не лише окремі записи, а й зв’язки між ними, оскільки саме узгоджене поєднання дат, тем занять, балів, статусів присутності та належності студента до певної групи формує інформаційну основу для подальшого аналізу.

Майбутній програмний продукт повинен бути орієнтований на кілька категорій користувачів. Ключове місце посідає викладач, який працює з журналом, додає заняття, вносить оцінки, фіксує відвідування, переглядає динаміку результатів і формує звіти. Окрему роль виконує адміністратор, відповідальний за підтримку довідкових даних, створення користувачів, ведення інформації про групи, студентів, дисципліни та журнали. Студент потребує доступу лише до власних результатів без можливості втручання в журнальні записи. Поділ користувачів за ролями зумовлює потребу в чіткому розмежуванні доступу до функцій системи та обмеженні перегляду академічних відомостей відповідно до повноважень.

Зміст задачі передбачає опрацювання кількох взаємопов’язаних інформаційних потоків. Перший потік охоплює довідкові дані, до складу яких входять облікові записи, ролі, відомості про викладачів, студентів, академічні групи та дисципліни. Другий потік формується під час поточної роботи з журналом і містить інформацію про заняття, дати проведення, тематику, результати оцінювання, присутність або відсутність студентів. Третій потік утворюють похідні дані, що виникають унаслідок аналітичного опрацювання первинних записів. До них належать середні значення, рейтингові позиції, показники відвідування, динаміка результатів, розподіл оцінок та інші узагальнені характеристики. Четвертий потік пов’язаний зі звітною документацією, що формується на основі вже збережених і проаналізованих відомостей [6].

Робота з системою починається з автентифікації, після чого відкривається доступ до функцій, визначених роллю користувача. Для викладача подальший сценарій охоплює вибір журналу, перегляд або створення занять, внесення оцінок і відміток про присутність, збереження змін та перехід до аналітичних і звітних матеріалів. Для адміністратора, пріоритетними є операції з довідковими сутностями, створення необхідного середовища для роботи викладача і підтримка цілісності інформаційного наповнення. Для студента логіка взаємодії обмежується переглядом власних показників і динаміки результатів.

Внесені оцінки та відмітки про присутність повинні одразу ставати основою для обчислення середніх показників, визначення частки відвідування, побудови рейтингу та формування інших узагальнень.

Окремою складовою задачі виступає формування звітності. В діяльності викладача виникає потреба у підготовці поточних і підсумкових відомостей, рейтингових списків, аналітичних зведень і матеріалів для подальшого перегляду або експорту. За відсутності інтегрованої звітної підсистеми значна частина часу витрачається на ручне перенесення інформації до зовнішніх електронних таблиць або текстових документів [7].

Постановка задачі повинна враховувати і вимоги до достовірності та безпеки. Академічна інформація містить персональні та службові відомості, тому система має забезпечувати контроль доступу до записів, підтримку ролей користувачів, фіксацію змін і перевірку коректності введених значень. Помилки в збереженні оцінок, дублювання записів або порушення зв’язків між заняттям, студентом і журналом можуть призвести до викривлення аналітичних результатів і зниження довіри до програмного засобу.

1.2 Модель задачі та її характеристики

Формування вимог до інформаційної системи належить до ключових етапів проєктування програмного продукту, оскільки саме на рівні вимог, закладаються межі майбутнього функціоналу, визначається логіка взаємодії користувача з системою, уточнюються характеристики даних, встановлюються параметри якості програмного рішення. Програмний продукт повинен поєднувати облікові процедури, засоби аналітичного опрацювання, механізми формування звітності, рольову модель доступу та інтерфейс, придатний для регулярного практичного використання в освітньому середовищі. За відсутності належно сформульованої системи вимог, розробка ризикує перетворитися на набір несистемних рішень, які не забезпечують ані цілісності функціонування, ані стійкості до змін предметної області [8].

Функціональні вимоги охоплюють перелік дій, операцій та сервісів, які програмний продукт повинен підтримувати для розв’язання прикладних завдань користувача. Нефункціональні вимоги відображають якісні характеристики функціонування системи, серед яких важливе місце посідають продуктивність, надійність, безпека, зручність використання, масштабованість та супроводжуваність [9].

Функціональні вимоги для інформаційної системи електронного журналу повинні відображати логіку роботи викладача з академічними даними в межах навчального процесу. Насамперед, йдеться про автентифікацію користувача, доступ до персоналізованого робочого простору, можливість вибору дисципліни, академічної групи, навчального періоду та окремого заняття. Без реалізації зазначених механізмів, неможливо забезпечити впорядковану роботу з інформацією, оскільки академічні записи мають чітку прив’язку до конкретного контексту. Також важливими є підтримка операцій створення, редагування, перегляду та збереження записів, пов’язаних з оцінюванням, відвідуваністю, тематикою занять та супровідними коментарями викладача.

Важливу частину функціональних вимог формують аналітичні можливості, призначені для перетворення первинних академічних даних на узагальнену інформацію, придатну для інтерпретації. Система має обчислювати середній бал за студентом, групою, дисципліною або визначеним періодом, аналізувати показники відвідуваності в абсолютному та відносному вираженні, формувати рейтингові зрізи, відображати зміни результатів у часі та надавати засоби порівняння між різними сукупностями даних [10].

Викладач, в реальному освітньому процесі працює не лише з поточними записами, а й зі зведеними відомостями, підсумковими оцінками, вибірками за конкретними періодами, узагальненими характеристиками успішності та відвідуваності, а іноді, з друкованими або електронними формами, які передаються адміністрації або використовуються під час внутрішнього контролю. Система повинна підтримувати автоматичне створення звітів за заданими параметрами, відбір записів за групами, дисциплінами або часовими

рамками, представлення результатів в зручній для перегляду та подальшого використання формі [14]. Основні функціональні вимоги до інформаційної системи електронного журналу викладача наведено в таблиці 1.1.

Таблиця 1.1 – Основні функціональні вимоги до інформаційної системи електронного журналу викладача

Функціональна область	Зміст вимоги	Практичне призначення
Керування доступом	підтримка входу до системи, розмежування прав користувачів, персоналізація робочого середовища	забезпечує безпечну роботу з академічними даними та рольову організацію доступу
Ведення журналу	внесення, редагування, перегляд і збереження оцінок, відміток про присутність, тем занять і коментарів	підтримує щоденний облік навчального процесу
Робота з довідковими даними	обробка відомостей про дисципліни, групи, студентів, навчальні періоди	формує інформаційну основу для коректного пов’язування записів
Аналітичне опрацювання	розрахунок середніх показників, рейтингів, динаміки успішності та відвідуваності	забезпечує перехід від обліку до аналітики
Формування звітів	створення підсумкових, поточних, порівняльних і вибіркових звітів	спрощує підготовку документів для викладача й адміністрації
Пошук і фільтрація	відбір записів за студентом, групою, дисципліною, датою, видом контролю	прискорює роботу з великими масивами інформації
Візуальне подання результатів	відображення зведених таблиць, діаграм, індикаторів активності та результативності	підвищує наочність сприйняття аналітичної інформації
Збереження історії змін	фіксація оновлень оцінок, виправлень відвідуваності та часу редагування	підтримує прозорість роботи й контроль коригувань

Наведена система функціональних вимог засвідчує, що електронний журнал викладача охоплює кілька взаємопов’язаних напрямків діяльності. Перший напрямок стосується безпосереднього ведення академічних записів та забезпечення їх цілісності [11]. Другий напрямок, пов’язаний з аналітичним

опрацюванням, яке повинно виконуватися на основі накопичених даних без додаткового ручного перенесення інформації до сторонніх інструментів. Наступний напрямок охоплює звітність та представлення результатів у форматі, придатному для швидкого сприйняття та практичного застосування. Наявність всіх названих складових, підвищує цінність системи для викладача та скорочує витрати часу на супровід навчального процесу [12].

Поряд з функціональними характеристиками визначальну роль відіграють нефункціональні вимоги, оскільки саме вони формують якість експлуатації програмного продукту. Навіть за умови коректної реалізації базових операцій, система може виявитися малопридатною для практичного використання, якщо вона працює повільно, має перевантажений інтерфейс, не забезпечує захисту даних або складно підтримується в процесі подальшого розвитку [13]. Для електронного журналу викладача, нефункціональні вимоги мають не менше значення, ніж функціональні, оскільки робота з академічною інформацією потребує одночасно стабільності, точності, швидкодії та зручності взаємодії.

Програмний продукт повинен забезпечувати збереження академічних записів без втрат, підтримувати коректність операцій редагування та запобігати пошкодженню даних у разі нештатних ситуацій. Для освітнього середовища, надійність має особливе значення, оскільки будь-яка втрата відомостей про оцінювання, відвідуваність або підсумкові результати здатна спричинити організаційні ускладнення, конфлікти або викривлення аналітичних висновків. Тому потрібно забезпечити функції резервування, перевірки коректності записів, журналювання змін та підтримки цілісності бази даних.

Інформаційна система електронного журналу оперує персональними та академічними даними, доступ до яких повинен бути обмежений відповідно до ролі користувача. Викладач має працювати з власними навчальними курсами та групами, студент повинен бачити лише індивідуальні результати, а адміністративний персонал, переглядати узагальнені або службові дані в межах визначених повноважень. Реалізація вимог безпеки передбачає наявність авторизації, автентифікації, контрольованого доступу до функцій, захисту

облікових записів, механізмів, що зменшують ризик несанкціонованого втручання в академічні записи [14].

Система повинна швидко реагувати на дії користувача під час відкриття журналу, завантаження списків студентів, збереження оцінок, побудови аналітичних зрізів та формування звітів. Повільне опрацювання запитів, істотно знижує практичну цінність програмного продукту, особливо у випадках, коли викладач працює з великим обсягом записів. Вимога продуктивності охоплює не лише швидкість відгуку інтерфейсу, а й ефективність роботи з базою даних, раціональну організацію запитів та коректний розподіл навантаження між компонентами системи.

Інтерфейс електронного журналу повинен бути інтуїтивно зрозумілим, логічно впорядкованим та не перевантаженим другорядними елементами. Викладач, працюючи з системою, має швидко переходити між групами, темами занять, оцінками та аналітичними матеріалами без тривалого пошуку потрібних функцій. Високий рівень зручності досягається завдяки продуманій структурі навігації, чіткому розміщенню елементів керування, наочному поданню підсумкових показників та мінімізації дій, необхідних для виконання типових операцій [15].

Програмний продукт, створений для роботи в освітньому середовищі, повинен зберігати працездатність при збільшення кількості груп, дисциплін, користувачів або аналітичних операцій. Одночасно, архітектура системи має залишатися придатною до подальшого розширення, модернізації та виправлення помилок без необхідності повного перепроєктування. Потрібно передбачати модульну організацію, чітке розмежування рівнів доступу до даних, незалежність окремих функціональних компонентів та зрозумілу внутрішню логіку програмного коду [16]. Нефункціональні вимоги до інформаційної системи електронного журналу викладача наведено в таблиці 1.2.

Таблиця 1.2 – Нефункціональні вимоги до інформаційної системи електронного журналу викладача

Нефункціональна характеристика	Зміст вимоги	Вплив на якість програмного продукту
Надійність	стабільне збереження даних, коректність операцій, захист від втрати записів	підтримує достовірність академічної інформації
Безпека	автентифікація, авторизація, розмежування доступу, захист персональних даних	знижує ризик несанкціонованих змін і витоку відомостей
Продуктивність	швидке відкриття сторінок, обробка запитів, побудова звітів та аналітики	забезпечує комфортну щоденну експлуатацію
Зручність використання	логічна навігація, простота введення даних, наочність результатів	скорочує час роботи користувача та зменшує ймовірність помилок
Масштабованість	підтримка зростання обсягу даних і кількості користувачів без критичного падіння швидкодії	робить систему придатною до ширшого впровадження
Супроводжуваність	можливість оновлення, модифікації й розширення функціоналу без порушення базової логіки	полегшує подальший розвиток програмного рішення
Сумісність	коректна робота у різних програмно-апаратних умовах і можливість інтеграції з іншими сервісами	розширює сферу практичного застосування
Точність обробки	правильність розрахунків, узгодженість результатів і відсутність спотворення показників	підвищує довіру до аналітики та звітності

Сукупність нефункціональних вимог не може розглядатися як другорядне доповнення до функціонального опису. Саме якісні характеристики програмного продукту визначають, чи буде розроблена система реально корисною в повсякденній діяльності викладача. Наприклад, функція

формування звіту втрачає практичну цінність, якщо її виконання займає надто багато часу. Аналогічно, модуль аналітики не забезпечить очікуваного ефекту, якщо користувачеві важко інтерпретувати результати через складний інтерфейс або неочевидну логіку представлення даних. Через подібні причини, в процесі формування вимог необхідно забезпечити узгоджений розгляд обох груп характеристик.

2 ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1 Огляд робіт

Проблематика розробки інформаційної системи електронного журналу викладача з розширеним функціоналом аналітики та звітності перебуває на перетині кількох науково-прикладних напрямів. Один напрям охоплює цифровізацію освіти та побудову інформаційно-освітнього середовища закладу освіти. Інший напрям пов’язаний з навчальною аналітикою, де досліджуються методи збирання, опрацювання й інтерпретації академічних даних. Ще один напрям стосується проєктування інформаційних систем, архітектурних рішень, моделей даних і програмних засобів, придатних для створення веборієнтованих застосунків.

В роботах, присвячених цифровій трансформації освіти, увага зосереджується на переході від локальних і розрізнених способів обліку до інтегрованого інформаційного середовища, де навчальні дані використовуються не лише для фіксації подій, але й для підтримки організаційних і педагогічних рішень. Електронний журнал розглядається як складова ширшої цифрової інфраструктури, що забезпечує збереження академічних записів, підтримує інформаційний обмін між учасниками освітнього процесу та створює підґрунтя для побудови звітних матеріалів [17].

Значна кількість оглянутих робіт присвячена навчальній аналітиці. В них академічні записи інтерпретуються як інформаційна база для виявлення тенденцій успішності, аналізу динаміки навчання, оцінювання рівня активності та контролю відвідуваності. Дослідники наголошують на доцільності використання статистичних узагальнень, порівняльних процедур, рейтингового впорядкування та часових зрізів.

Окрему групу становлять роботи, пов’язані з educational data mining та суміжними методами аналізу навчальних даних. В рамках даних досліджень опрацьовуються питання класифікації, прогнозування, виявлення прихованих

залежностей та побудови структурних узагальнень. Для електронного журналу викладача найбільшу цінність мають не складні прогностичні моделі, а ті положення, які стосуються практичної інтерпретації оцінок, відвідуваності, розподілу результатів та зв’язку між окремими показниками.

Важливе місце займають роботи, присвячені інформаційним системам освітнього призначення. В подібних розробках зазвичай підтримуються ведення списків студентів, внесення оцінок, фіксація пропусків, збереження довідкових відомостей та перегляд підсумкових даних. Частина рішень містить засоби експорту в табличний формат або побудови елементарних зведень. Основні напрями робіт, дотичних до теми роботи наведено в таблиці 2.1.

Таблиця 2.1 – Основні напрями робіт, дотичних до теми роботи

Напрям робіт	Зміст досліджень	Значення для теми роботи
Цифровізація освіти	розгляд інформаційно-освітнього середовища, цифрових сервісів та інтеграції освітніх даних	формує загальний контекст використання електронного журналу у структурі цифрового середовища
Навчальна аналітика	методи узагальнення й інтерпретації академічних даних, виявлення тенденцій, побудова аналітичних зрізів	визначає зміст аналітичного функціоналу програмної системи
Аналіз освітніх даних	статистичне, рейтингове, порівняльне та структурне опрацювання результатів навчання і відвідуваності	слугує основою для реалізації обчислювальних алгоритмів
Інформаційні системи освітнього призначення	електронні журнали, системи обліку успішності, сервіси збереження й експорту навчальних записів	демонструє прикладні підходи до побудови програмного продукту
Програмна інженерія та архітектура систем	формування вимог, моделювання сутностей, проєктування архітектури та вебреалізації	забезпечує методичну основу для створення власної системи

Матеріали з програмної інженерії та проєктування інформаційних систем дозволяють розглядати задачу розробки електронного журналу не лише з прикладного, але й з інженерного погляду [18]. У фаховій літературі підкреслюється важливість формалізації вимог, побудови архітектури, моделювання предметної області, логічного розмежування функціональних модулів та підтримки цілісності даних. Для обраної теми це має принципове значення, оскільки майбутній програмний засіб повинен поєднати кілька різнорідних функцій. Йдеться про ведення журналу, роботу з довідковими сутностями, аналітичне опрацювання академічних показників, формування звітних документів та підтримку рольового доступу.

Огляд робіт, пов’язаних з веборієнтованими інформаційними системами, підтверджує доцільність використання клієнт-серверної моделі з централізованим зберіганням записів в реляційній базі даних. Розглянутий підхід найбільшою мірою відповідає змісту обраної задачі, оскільки дає змогу забезпечити багатокористувацьку роботу, підтримку ролей, єдине джерело достовірної інформації, контроль змін і можливість подальшого формування звітів без дублювання відомостей у зовнішніх файлах. Практика створення подібних систем свідчить, що найбільшого ефекту вдається досягти за умови інтеграції облікових, аналітичних і звітних процедур у межах одного програмного середовища [19].

Проаналізовані роботи дозволяють зробити висновок, що обрана тема має належне теоретичне та прикладне підґрунтя. Наукові джерела формують методологічну основу для аналізу освітніх даних, праці з цифровізації освіти окреслюють місце електронного журналу в сучасному освітньому середовищі, а публікації з програмної інженерії надають інструменти для побудови архітектури й моделі даних. Водночас, огляд засвідчує, що значна частина наявних робіт висвітлює лише окремі аспекти проблеми, тоді як питання поєднання електронного журналу викладача з розширеним функціоналом аналітики та звітності потребує цілісного розгляду.

2.2 Позитивні аспекти оглянутих робіт

Аналіз наукових публікацій і прикладних розробок, пов’язаних з цифровізацією освітнього середовища, навчальною аналітикою та електронними журналами, дозволяє виокремити низку вагомих напрацювань, що становлять методичну й практичну основу для виконання кваліфікаційної роботи. В рамках оглянутих джерел простежується поступовий перехід від сприйняття електронного журналу як засобу фіксації оцінок до розуміння його як компонента цілісної інформаційної системи, придатної для накопичення академічних даних, побудови узагальнень і підтримки роботи викладача [20].

Важливою позитивною рисою оглянутих робіт є увага до освітніх даних як до цінного інформаційного ресурсу. В публікаціях, присвячених навчальній аналітиці, академічні записи розглядаються не ізольовано, а у взаємозв’язку з перебігом навчального процесу, активністю здобувачів освіти, якістю засвоєння матеріалу та результатами контролю. Представлене бачення має велике значення для розробки електронного журналу викладача, оскільки воно зміщує акцент з простого збереження оцінок на подальше інтерпретування відомостей.

Ще однією перевагою оглянутих праць є детальне опрацювання методів аналізу успішності та відвідуваності. В наукових джерелах обґрунтовано використання середніх значень, рейтингового впорядкування, порівняльних процедур, часових зрізів, структурного розподілу оцінок і виявлення взаємозв’язків між окремими показниками.

Позитивним аспектом оглянутих робіт слід вважати і увагу до інформаційно-освітнього середовища закладу освіти. У відповідних дослідженнях наголошується на необхідності централізованого зберігання відомостей, взаємодії цифрових сервісів, підтримки різних категорій користувачів і забезпечення цілісності інформаційного простору. Позитивні аспекти оглянутих робіт за основними напрямами узагальнено в таблиці 2.2.

Таблиця 2.2 – Позитивні аспекти оглянутих робіт за основними напрямами

Напрям огляду	Позитивний аспект	Значення для теми роботи
Цифровізація освіти	обґрунтовано роль цифрових сервісів у супроводі освітнього процесу	формує загальний контекст створення електронного журналу
Навчальна аналітика	освітні дані розглянуто як основу для побудови аналітичних узагальнень	визначає зміст розширеного функціоналу аналітики
Аналіз успішності та відвідуваності	опрацьовано методи статистичного, порівняльного та рейтингового аналізу	надає основу для реалізації алгоритмів у програмній системі
Інформаційно-освітнє середовище	увагу зосереджено на централізованому зберіганні й взаємодії даних	підтримує ідею єдиного інформаційного простору
Освітні інформаційні системи	підтверджено ефективність автоматизації журнального обліку	засвідчує прикладну цінність електронного журналу
Програмна інженерія	розкрито значення архітектурного проєктування, моделювання та формалізації вимог	забезпечує методичну основу для власної розробки

Матеріали з програмної інженерії мають особливу цінність завдяки чіткому опрацюванню вимог до архітектури, логіки моделювання даних та принципів побудови програмного коду. В даних роботах детально висвітлюються питання структурування системи, розмежування функцій між окремими модулями, підтримки цілісності даних і забезпечення супроводжуваності програмного продукту [21].

У веборієнтованих рішеннях, які було розглянуто під час огляду, варто відзначити прагнення до централізації доступу та уніфікації інтерфейсної взаємодії. Використання браузера як клієнтського середовища, серверної обробки запитів та реляційної бази даних дозволяє забезпечити єдине джерело достовірної інформації та підтримати роботу кількох категорій користувачів.

В багатьох дослідженнях підкреслюється значення таблиць, графіків, зведених форм і дашбордів для швидкого сприйняття освітніх показників. Подібний підхід має особливу цінність для викладача, оскільки дозволяє перейти від перегляду великої кількості рядків журналу до стислого узагальненого бачення ситуації.

Певні позитивні висновки можна зробити і щодо організації звітності в оглянутих рішеннях. Навіть в тих роботах і прикладних розробках, де аналітична складова подана обмежено, простежується прагнення до формування підсумкових відомостей, експорту результатів і створення зручних для перегляду документів [22].

Проведений аналіз дозволяє стверджувати, що оглянуті роботи містять низку вагомих позитивних положень, які є корисними для подальшої розробки інформаційної системи електронного журналу викладача. Вони формують теоретичну, методичну та прикладну основу для побудови архітектури, моделі даних, аналітичного блоку й підсистеми звітності. Водночас повноцінне розуміння місця власної розробки у межах уже наявних підходів потребує окремого розгляду тих обмежень, які зберігаються в оглянутих роботах. Саме через це логічним продовженням інформаційного огляду є аналіз вад і невирішених питань, що становитиме зміст наступного підпункту.

2.3 Вади розробок з оглянутих робіт

Поряд з вагомими результатами, виявленими під час аналізу наукових публікацій та прикладних розробок, огляд засвідчив наявність низки обмежень, які ускладнюють повноцінне використання наявних рішень у практиці роботи викладача. Значна частина робіт зосереджується або на загальних питаннях цифровізації освіти, або на окремих методах навчальної аналітики, або на технічних аспектах створення інформаційних систем. Внаслідок цього, питання побудови єдиного програмного середовища, де поєднуються ведення електронного журналу, аналітичне опрацювання та формування звітності, розкривається фрагментарно [23].

Одна з найбільш помітних вад оглянутих робіт полягає в переважанні загальнотеоретичного характеру викладу над описом завершених прикладних рішень. В низці джерел детально пояснюється роль навчальної аналітики, розкриваються переваги цифрових сервісів або характеризуються можливості оброблення освітніх даних, проте не подається завершеного бачення програмної реалізації, придатної до щоденного використання викладачем.

Помітним недоліком прикладних освітніх систем, описаних в літературі та наявних цифрових рішеннях, є зосередження переважно на функціях обліку. Найчастіше подібні сервіси забезпечують ведення списків студентів, фіксацію оцінок і позначення присутності, проте аналітичні засоби або відсутні, або мають спрощений характер. За наявності подібного обмеження викладач змушений переносити дані до сторонніх таблиць або окремих програм для побудови середніх значень, рейтингових зрізів, аналізу динаміки та інших узагальнень.

Ще одна проблема оглянутих розробок пов’язана з недостатньою увагою до звітності. Навіть в тих рішеннях, де підтримується базове формування підсумкових відомостей, звітні документи часто мають обмежений набір параметрів, не забезпечують гнучкого відбору даних і не підтримують архівування результатів [24].

В наукових публікаціях розглядаються корисні статистичні, рейтингові, часові та кореляційні процедури, проте далеко не завжди пояснюється, яким чином названі методи мають бути інтегровані у програмний засіб, орієнтований на повсякденну роботу викладача. Частина досліджень тяжіє до складніших моделей аналізу, що становлять значний інтерес для науки, але не завжди відповідають вимогам простоти, наочності та оперативності, які є критичними у прикладному освітньому середовищі. Внаслідок цього зберігається потреба в побудові системи, де аналітика буде достатньо інформативною, але водночас зрозумілою та зручною для практичного використання.

Суттєвою вадою частини оглянутих робіт є недостатнє опрацювання рольової організації доступу. В теоретичних джерелах визнається потреба

захисту академічних даних і розмежування повноважень, проте у прикладних описах не завжди простежується повний розподіл функцій між адміністратором, викладачем і студентом.

Значна частина оглянутих рішень також не приділяє належної уваги журналюванню змін та контролю коригувань. В межах викладацької практики можливі уточнення оцінок, виправлення технічних помилок, зміна інформації про відвідування або доповнення попередніх записів. За відсутності механізму фіксації оновлень система втрачає прозорість, а довіра до сформованих підсумків знижується [25]. Основні вади оглянутих робіт і прикладних рішень наведено в таблиці 2.3.

Таблиця 2.3 – Основні вади оглянутих робіт і прикладних рішень

Група вад	Сутність обмеження	Наслідок для практичного використання
Фрагментарність викладу	окремо розглядаються цифровізація, аналітика або програмна реалізація без їх цілісного поєднання	ускладнюється побудова завершеної системи, орієнтованої на реальну роботу викладача
Переважання облікових функцій	акцент робиться на внесенні оцінок і відвідуваності без розвиненої аналітики	знижується цінність електронного журналу як інструмента підтримки рішень
Обмежена звітність	звіти мають спрощений характер або відсутня гнучкість формування документів	користувач змушений готувати частину матеріалів поза межами системи
Недостатня адаптація аналітики до практики	наявні методи аналізу не завжди інтегровані у зрозумілий прикладний інтерфейс	аналітичні результати важко використовувати у повсякденній діяльності
Неповне рольове розмежування	не завжди враховано специфіку доступу адміністратора, викладача і студента	виникають ризики надмірного доступу або обмеження потрібних функцій
Відсутність контролю змін	недостатньо опрацьовано аудит редагувань і журналювання оновлень	знижується прозорість роботи з академічними записами

Окремого розгляду потребує питання повноти моделі даних в наявних розробках. Частина рішень орієнтована лише на збереження базового набору атрибутів, через що слабко відображається логічний зв’язок між журналом,

заняттям, студентом, оцінкою та відвідуванням. За подібної побудови дані формально існують в цифровій системі, але їх подальше узагальнення ускладнюється через недостатню структурованість.

В роботах, присвячених освітньому середовищу, також простежується недостатній акцент на потребах безпосереднього користувача системи, тобто викладача. Нерідко увага зосереджується на загальноінституційному рівні, організації цифрової інфраструктури або педагогічних наслідках використання технологій, тоді як практичний сценарій щоденної роботи з журналом висвітлюється менш детально. Внаслідок цього зберігається потреба в розробці програмного засобу, де саме логіка роботи викладача буде поставлена в центр архітектурних та функціональних рішень.

Недоліки можна узагальнити і з погляду їх впливу на майбутню розробку. Частина виявлених обмежень має теоретичний характер, оскільки пов’язана з недостатньою завершеністю моделей або неповнотою опису прикладних сценаріїв. Інша частина належить до функціональних вад і стосується слабкої аналітики, обмеженої звітності та ролей доступу [25]. Окремо можна виділити технічні недоліки, пов’язані з неповною структурою даних, відсутністю журналювання змін і слабкою інтеграцією функціональних модулів. Узагальнення виявлених вад за характером їх впливу представлено в таблиці 2.4.

Проведений аналіз дозволяє стверджувати, що оглянуті роботи, попри безперечну наукову і прикладну цінність, не забезпечують повного розв’язання задачі, пов’язаної зі створенням електронного журналу викладача з розширеним функціоналом аналітики та звітності. Виявлені вади не знижують значення попередніх напрацювань, проте окреслюють межі їх застосовності та вказують на напрями, в яких зберігається потреба у власній розробці.

Таблиця 2.4 – Узагальнення виявлених вад за характером їх впливу

Характер вади	Прояв у оглянутих роботах	Що саме потребує усунення у власній розробці
Теоретичний	неповний опис завершеного прикладного рішення	побудова цілісної моделі системи з чіткою логікою роботи
Функціональний	переважання обліку над аналітикою і звітністю	інтеграція журналу, аналітичного блоку та підсистеми звітності
Організаційний	неповне розмежування ролей користувачів	підтримка окремих сценаріїв для адміністратора, викладача і студента
Інформаційний	недостатньо структурована модель академічних даних	побудова повної реляційної моделі з контрольованими зв’язками
Контрольний	відсутність фіксації змін і коригувань	реалізація журналу аудиту та підтримки прозорості редагувань

2.4 Необхідність та актуальність теми роботи

Проведений інформаційний огляд показав, що проблема цифрового супроводу діяльності викладача вже перебуває в полі уваги дослідників та розробників, проте наявні роботи та прикладні рішення здебільшого зосереджуються на окремих аспектах, не забезпечуючи цілісного поєднання журнального обліку, аналітичного опрацювання та звітності в межах єдиної інформаційної системи. Частина робіт висвітлює методи навчальної аналітики, інша частина розглядає електронні журнали як засоби обліку, а окремі дослідження зосереджені на цифровому освітньому середовищі загалом. За подібних умов, зберігається потреба в створенні прикладного програмного рішення, орієнтованого саме на повсякденні потреби викладача.

Актуальність теми роботи визначається необхідністю автоматизації ведення академічних записів, скороченням трудомісткості рутинних операцій, підвищенням достовірності збереження відомостей та забезпеченням швидкого доступу до узагальнених результатів навчального процесу. Практична значущість розробки полягає в можливості поєднати в одному веборієнтованому застосунку функції ведення електронного журналу, аналізу

показників успішності та відвідуваності, формування звітів і рольового доступу до даних.

3 ТЕОРЕТИЧНА ЧАСТИНА

3.1 Проєктування архітектури програмного забезпечення

Проєктування архітектури програмного забезпечення посідає центральне місце в створенні інформаційної системи електронного журналу викладача, оскільки саме на названому етапі визначається загальна організація програмного продукту, встановлюються межі відповідальності між окремими компонентами, формується логіка взаємодії між інтерфейсом, серверною частиною, базою даних, аналітичним блоком і підсистемою звітності.

Для системи освітнього призначення архітектурне проєктування має особливе значення, адже програмний продукт повинен одночасно забезпечувати щоденне введення академічних записів, підтримку рольового доступу, опрацювання накопичених показників і підготовку підсумкових документів.

Для розроблюваного програмного забезпечення доцільною є веборієнтована клієнт-серверна архітектура, де користувач працює з системою через браузер, а основна логіка оброблення, перевірки і збереження даних виконується на серверному рівні. Водночас подібна модель дозволяє відокремити інтерфейсну взаємодію від процедур аналізу, зберігання та формування звітів, що позитивно впливає на структурованість програмного коду.

Під час архітектурного проєктування увагу зосереджено на модульній організації системи. В межах застосунку виділяються інтерфейсний рівень, серверна логіка, рівень даних, аналітичний модуль, підсистема звітності та блок керування доступом. Кожен з даних компонентів виконує окреме завдання, проте працює не ізольовано, а в складі спільного циклу опрацювання академічних відомостей [26]. Архітектурні рівні програмного забезпечення наведено в таблиці 3.1.

Таблиця 3.1 – Архітектурні рівні програмного забезпечення

Архітектурний рівень	Функціональне призначення	Результат роботи
Клієнтський рівень	відображення сторінок, приймання введених користувачем значень, передавання запитів	забезпечення взаємодії користувача із системою через браузер
Серверний рівень	оброблення запитів, перевірка прав доступу, виконання прикладної логіки	керування роботою журналу, аналітики та звітності
Рівень даних	збереження довідкових, операційних, аналітичних і службових відомостей	підтримка цілісності та доступності академічної інформації
Аналітичний рівень	розрахунок середніх значень, рейтингів, динаміки, розподілів і зв’язків між показниками	формування узагальнених показників для викладача і студента
Рівень звітності	побудова підсумкових документів і архівування результатів	підготовка файлів для перегляду, завантаження і подальшого використання

В структурі програмного забезпечення центральне місце посідає серверна частина, оскільки саме вона координує взаємодію між всіма іншими рівнями. Після надходження запиту від користувача сервер визначає його роль, перевіряє допустимість доступу до конкретної функції, звертається до бази даних, виконує необхідні перетворення й повертає результат до інтерфейсу. Викладач отримує інструменти ведення журналу, адміністратор працює з довідковими сутностями, студент переглядає лише власні результати, а звітна та аналітична підсистеми використовують той самий інформаційний масив без дублювання даних у сторонніх файлах.

Зміст архітектури визначається не лише розподілом за рівнями, а й правильним виокремленням функціональних модулів. В розроблюваній системі до числа базових належать модуль автентифікації, модуль роботи з довідковими даними, модуль ведення електронного журналу, модуль аналітичного опрацювання, підсистема формування звітів, модуль аудиту змін

та модуль індивідуального перегляду результатів [27]. Основні модулі програмного забезпечення узагальнено в таблиці 3.2.

Таблиця 3.2 – Основні модулі програмного забезпечення

Модуль	Основне призначення	Інформація, з якою працює модуль
Модуль автентифікації	перевірка облікових даних і відкриття доступу відповідно до ролі	логіни, паролі, ролі користувачів
Модуль довідкових даних	ведення відомостей про викладачів, студентів, групи, дисципліни та журнали	довідкові сутності системи
Модуль журналу	створення занять, збереження оцінок, відміток про присутність і коментарів	заняття, оцінки, відвідування, службові нотатки
Аналітичний модуль	опрацювання накопичених показників і побудова узагальнень	оцінки, відвідування, часові вибірки, агреговані дані
Підсистема звітності	формування файлів на основі первинних і похідних даних	журнальні записи, аналітичні результати, службові реквізити
Модуль аудиту	фіксація додавання, зміни й видалення записів	історія змін користувачів і службові дані
Модуль студентського перегляду	подання індивідуальних результатів без права редагування	особисті оцінки, відвідування, узагальнені показники

В найпростішому сценарії користувач відкриває сторінку журналу, обирає потрібний контекст роботи, після чого система завантажує довідкову інформацію і поточні записи. Далі введені оцінки або позначки про присутність проходять перевірку, зберігаються в базі даних і стають джерелом для автоматичного оновлення узагальнених показників. За потреби, накопичені результати передаються до модуля аналітики або звітності.

Архітектурне рішення передбачає наявність єдиного сховища даних, де поєднуються довідкові сутності, журнальні записи, результати аналітичних обчислень і службові журнали. Якщо оцінки, відвідування, звітні матеріали та

аналітичні показники зберігались в різних незалежних середовищах, контроль їх узгодженості суттєво ускладнився. Централізована модель, навпаки, дозволяє підтримувати єдине джерело достовірної інформації, що має визначальне значення для освітнього процесу.

Важливе місце в проєктуванні архітектури посідає рольова організація доступу. Програмний продукт повинен підтримувати кілька моделей поведінки залежно від категорії користувача. Для адміністратора пріоритетним є керування довідковими даними і підтримка внутрішньої структури системи. Для викладача основними залишаються операції з журналом, аналітичними матеріалами, звітами і власним профілем. Для студента доступ обмежується переглядом індивідуальних результатів без права редагування.

Архітектура програмного забезпечення для інформаційної системи електронного журналу викладача побудована на принципах централізованого зберігання даних, модульного розподілу функцій, рольового доступу і відокремлення облікових, аналітичних та звітних процедур.

3.2 Графічне представлення архітектури

Архітектурна побудова системи спирається на розмежування кількох логічних рівнів. На рівні представлення, розміщується веб-інтерфейс, через який викладач або адміністратор взаємодіє з програмним продуктом, вводить дані, переглядає журнал, запускає аналітичні процедури та отримує звітні форми. Прикладний рівень охоплює модулі бізнес-логіки, відповідальні за перевірку коректності запитів, обробка академічних записів, обчислення показників успішності та відвідуваності, застосування правил доступу та підготовку даних до відображення. Рівень зберігання даних містить реляційну базу, в структурі якої, акумулюються відомості про користувачів, групи, дисципліни, заняття, оцінки, відмітки про присутність, аналітичні показники та сформовані звіти.

Електронний журнал викладача повинен підтримувати безперервний цикл обробки інформації, який починається з автентифікації користувача,

переходить до вибору навчального контексту, продовжується введенням або коригуванням академічних записів і завершується аналітичним узагальненням та формуванням підсумкових матеріалів. Логіка функціонування побудована не навколо ізольованих сторінок інтерфейсу, а навколо пов’язаних бізнес-процесів, де кожен запис, внесений до журналу, надалі може бути використаний для статистичного узагальнення, порівняльного аналізу, рейтингового впорядкування або включення до звітного документа. Компонентна UML-діаграма архітектури інформаційної системи електронного журналу викладача наведена на рисунку 3.1.

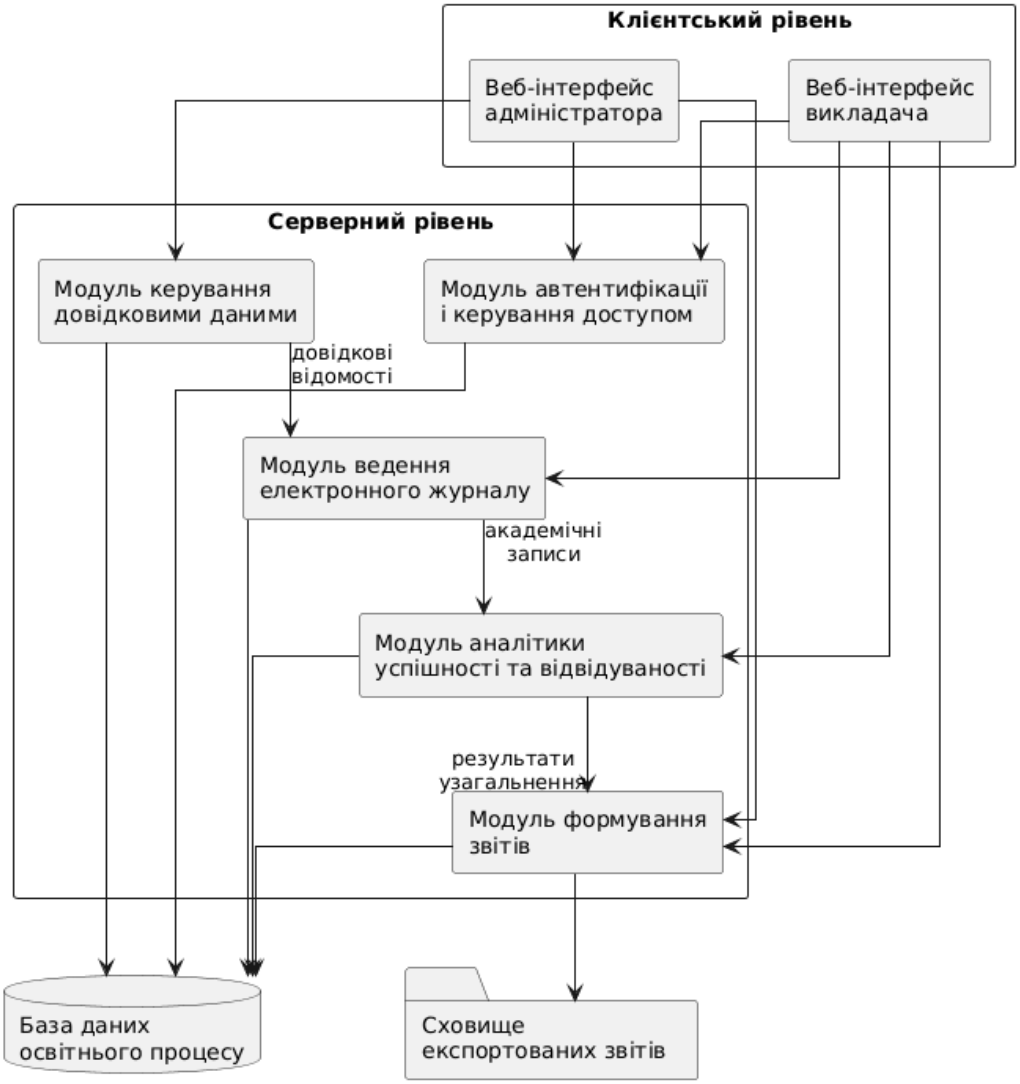

Рисунок 3.1 – Компонентна UML-діаграма архітектури інформаційної системи електронного журналу викладача

Компонентна модель відображає внутрішню організацію системи та дозволяє чітко простежити функціональну роль кожного програмного блоку. Центральне місце в архітектурі посідає модуль ведення електронного журналу, оскільки саме через нього здійснюється основна маса операцій, пов’язаних з поточним обліком навчального процесу. Даний модуль взаємодіє з довідковими даними, отримує інформацію про групи, дисципліни, студентів та навчальні періоди, після чого зберігає первинні записи в базі даних. Аналітичний модуль опрацьовує нагромаджені відомості, розраховує середні та підсумкові показники, визначає тенденції зміни успішності та відвідуваності, готує структури даних для подальшого відображення в зведеному форматі. Модуль звітності використовує як первинні записи, так і результати узагальнення, формуючи документи, придатні для перегляду на екрані, експорту або друку.

Наявність автентифікації та керування доступом модуля зумовлена потребою захисту персональних та академічних відомостей, необхідністю розмежування повноважень між різними категоріями користувачів. Викладач працює насамперед з навчальними курсами, журналами власних груп та аналітикою, яка стосується доступних дисциплін. Адміністратор, відповідає за підтримку довідкової інформації, налаштування системи та контроль коректності загальної структури даних. Рольова організація доступу повинна бути закладена саме на архітектурному рівні, оскільки подальше додавання подібного механізму після завершення основного проєктування зазвичай ускладнює узгодження модулів між собою.

Загальна організація системи передбачає побудову єдиного інформаційного середовища, де кожен програмний модуль працює не автономно, а в складі спільного циклу обробки освітніх даних. Після входу до системи, користувач обирає групу, дисципліну та необхідний часовий інтервал. Далі, виконуються операції внесення або перегляду оцінок, позначення відвідуваності, фіксації тем занять і службових коментарів. Після завершення даного етапу, дані одразу стають доступними для аналітичного опрацювання, що усуває потребу повторного перенесення відомостей до окремих

інструментів. На завершальному етапі, накопичені записи можуть бути подані у вигляді рейтингу, зведеної таблиці, динамічного огляду або звітного документа.

Модуль ведення журналу не повинен містити надмірну кількість аналітичних процедур, оскільки його головне призначення полягає в реєстрації та збереженні академічних записів. Аналітичний блок, навпаки, повинен спиратися на накопичені дані без дублювання функцій введення. Модуль звітності не виконує глибинного опрацювання інформації, а отримує вже підготовлені набори показників та перетворює їх на зручні для перегляду форми. Розмежування сприяє модульності, зменшує взаємну залежність програмних блоків та полегшує супровід системи на подальших етапах розвитку [18]. Діаграма розгортання інформаційної системи електронного журналу викладача наведена на рисунку 3.2.

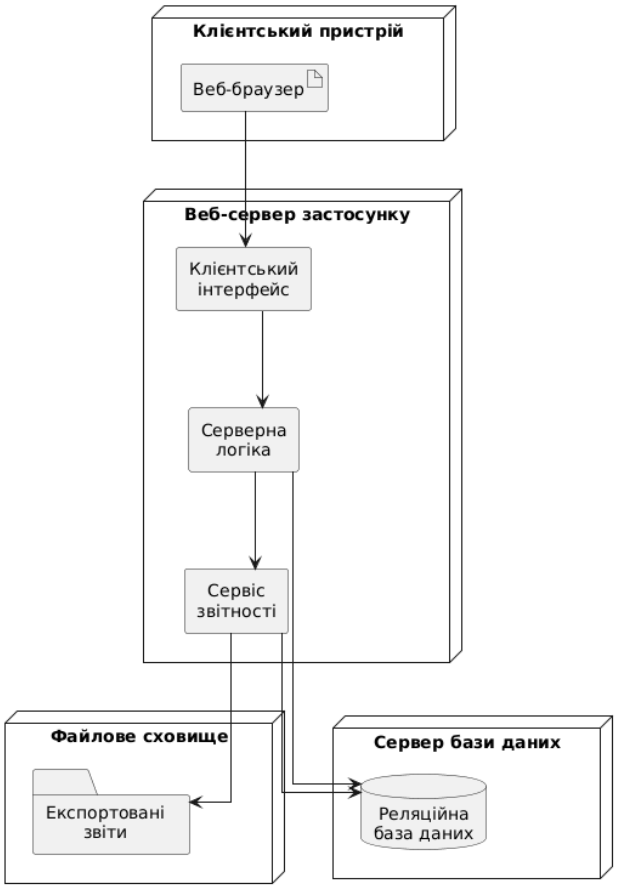

Рисунок 3.2 – UML-діаграма розгортання інформаційної системи електронного журналу викладача

На клієнтському пристрої, користувач взаємодіє з веб-інтерфейсом за допомогою звичайного браузера, що відкидає потребу встановлення окремого локального програмного забезпечення. На веб-сервері розміщується інтерфейсна частина, серверна бізнес-логіка та сервіс звітності. Сервер бази даних відповідає за довготривале зберігання академічних записів, довідкової інформації та результатів аналітичного опрацювання. Окреме файлове сховище використовується для накопичення згенерованих звітів, зокрема у випадках, коли формується архів документів або підтримується повторне завантаження раніше створених файлів.

Важливо підкреслити, що вибір веб-орієнтованої архітектури з централізованою базою даних пов’язаний не лише з технічною доцільністю, а й з особливостями освітнього процесу. Викладач, впродовж семестру постійно працює з оновлюваними даними, а тому, будь-яке локальне розосередження записів між різними файлами або пристроями ускладнює підтримку їх узгодженості. Централізована модель зберігання, дозволяє підтримувати єдине джерело достовірної інформації, спрощує контроль змін, полегшує формування підсумкових звітів та створює основу для масштабування системи в межах кількох академічних груп, кафедри або закладу освіти загалом.

Освітнє середовище характеризується змінністю форм контролю, правил оцінювання, структури навчальних планів, форматів звітності та порядку доступу до академічної інформації. Програмний продукт має бути спроєктований не як жорстко фіксований набір сторінок та процедур, а як модульна система, здатна до розширення через додавання нових сервісів або коригування наявних компонентів.

Багаторівнева клієнт-серверна модель в поєднанні з окремими модулями ведення журналу, аналітики, звітності, довідкових даних і керування доступом найкраще відповідає призначенню розроблюваної системи.

Після визначення архітектурної побудови інформаційної системи, постає потреба у формалізації структури даних, з якими працює електронний журнал викладача. Саме інформаційна модель визначає, яким буде склад сутностей

предметної області, які зв’язки виникатимуть між окремими наборами відомостей, в якій послідовності відбуватиметься накопичення академічних записів та на основі яких правил виконуватиметься подальше аналітичне опрацювання. Якість побудови моделі безпосередньо впливає на коректність функціонування програмного продукту, надійність зберігання результатів навчальної діяльності, точність обчислення показників успішності та відвідуваності і на повноту формування звітних матеріалів.

Інформаційна модель даних для електронного журналу викладача, повинна відображати реальну організацію освітнього процесу та водночас бути придатною до програмної реалізації для реляційної бази даних. Предметна область охоплює кілька взаємопов’язаних груп відомостей. До першої групи належать користувачі системи та їх ролі, оскільки будь-яка операція з академічними записами виконується від імені конкретного суб’єкта з визначеним обсягом повноважень. Друга група містить навчальний контекст, де фіксуються дисципліни, академічні групи, студенти, викладачі, навчальні періоди та журнали. Третя група охоплює операційні записи, серед яких, центральне місце посідають заняття, оцінки та відмітки про відвідування. Окрему групу формують похідні дані, що виникають внаслідок обробки первинних записів та використовуються під час побудови аналітичних зрізів та звітів. Основні сутності інформаційної моделі даних електронного журналу викладача наведено в таблиці 3.3.

Наприклад, значення оцінки саме по собі не має достатнього інформаційного змісту без прив’язки до студента, дисципліни, дати заняття, форми контролю та конкретного журналу. Аналогічна ситуація виникає і з відвідуваністю, оскільки позначка про відсутність набуває аналітичної цінності лише за наявності зв’язку з навчальною групою, періодом навчання та обсягом пропущених занять. Ключовим завданням проєктування стає не просте визначення набору таблиць, а побудова зв’язної системи сутностей, де кожен запис має однозначне місце та функціональне призначення.

Таблиця 3.3 – Основні сутності інформаційної моделі даних електронного журналу викладача

Сутність	Призначення	Основні атрибути
Роль	визначає межі доступу та набір дозволених дій користувача	ідентифікатор ролі, назва ролі, опис повноважень
Користувач	зберігає облікові відомості для входу до системи	ідентифікатор користувача, логін, пароль, стан облікового запису
Викладач	описує професійний профіль особи, яка веде журнал	ідентифікатор викладача, прізвище, ім’я, контактні дані
Студент	містить персональні та навчальні відомості про здобувача освіти	ідентифікатор студента, прізвище, ім’я, номер групи
Академічна група	об’єднує студентів у навчальну одиницю	ідентифікатор групи, назва групи, курс, спеціальність
Дисципліна	відображає навчальний курс, для якого ведеться журнал	ідентифікатор дисципліни, назва, семестр, форма контролю
Журнал	формує контекст ведення записів для викладача, групи та дисципліни	ідентифікатор журналу, навчальний рік, семестр, статус
Заняття	описує окрему навчальну подію в межах журналу	ідентифікатор заняття, дата, тема, вид заняття
Оцінка	зберігає результат контролю знань студента	ідентифікатор оцінки, значення, тип контролю, дата внесення
Відвідування	фіксує факт присутності або відсутності студента	ідентифікатор запису, статус присутності, причина відсутності
Звіт	містить відомості про сформований аналітичний або підсумковий документ	ідентифікатор звіту, тип, період, дата формування

Центральним елементом інформаційної моделі варто вважати сутність журналу. Саме журнал поєднує в собі відомості про викладача, академічну групу, дисципліну та навчальний період. В одному журналі, накопичуються записи про проведені заняття, а вже на рівні окремого заняття зберігаються оцінки студентів і позначки про присутність. Інформаційну модель доцільно будувати на основі первинних та зовнішніх ключів. Кожна сутність повинна мати власний ідентифікатор, який забезпечує однозначне розмежування

записів. Взаємозв’язок між сутностями реалізується через зовнішні ключі, завдяки яким, підтримується цілісність відомостей. Наприклад, запис про оцінку не може існувати без посилання на конкретного студента і конкретне заняття, а заняття, має бути прив’язане до певного журналу. Аналогічні правила стосуються і звітів, які формуються на основі вже наявного масиву записів та повинні зберігати інформацію про параметри побудови, часові межі та джерело даних [21]. Діаграма класів наведена на рисунку 3.3.

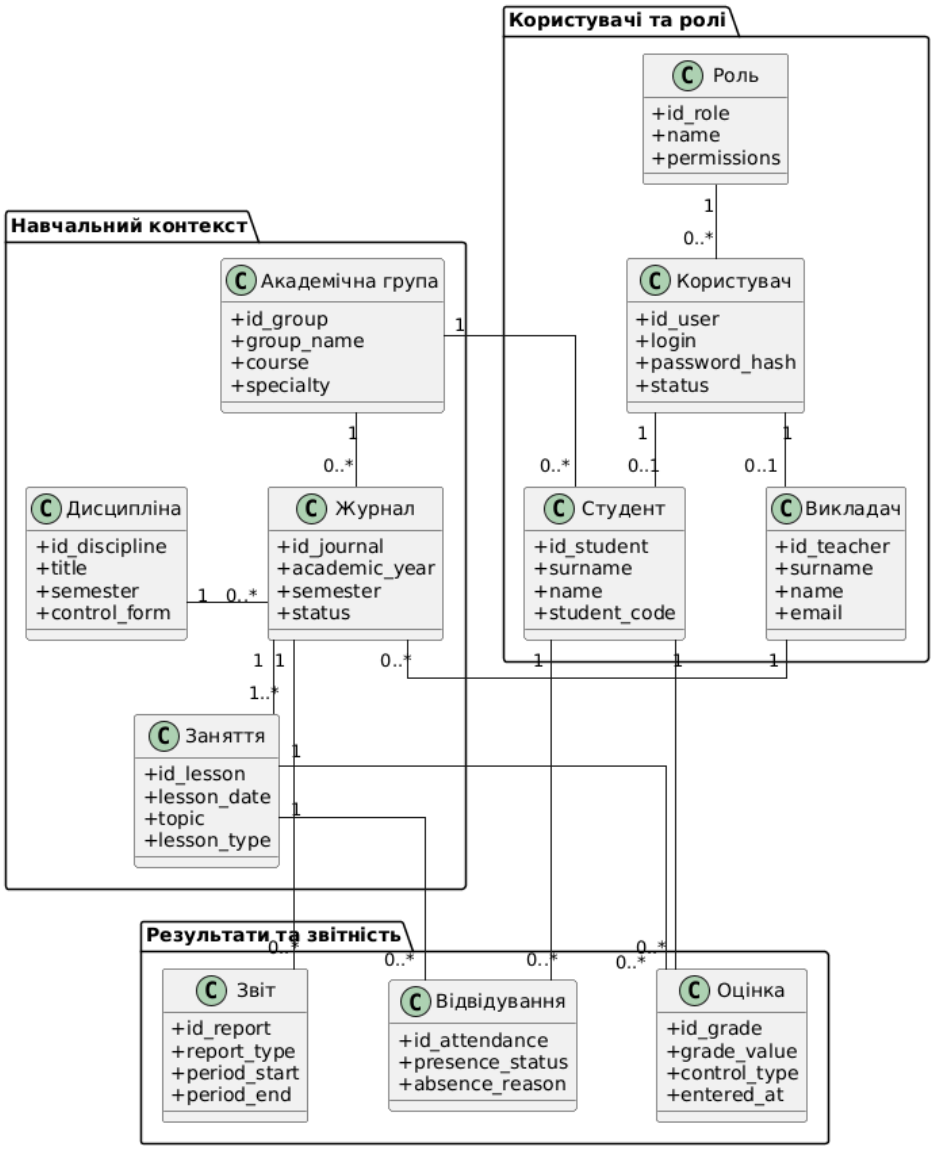

Рисунок 3.3 – UML-діаграма класів інформаційної моделі даних електронного журналу викладача

Діаграма дозволяє простежити загальну логіку побудови інформаційного простору системи. Від верхнього рівня, на якому розміщуються ролі та

користувачі, модель переходить до навчального контексту, а далі, до результатів освітньої діяльності та звітності.

Для електронного журналу логіка обробки даних, повинна враховувати життєвий цикл академічного запису від моменту його створення до включення в підсумковий аналітичний або звітний результат. Робота системи починається з ідентифікації користувача та перевірки його повноважень. Після входу, відкривається доступ до довідкових та операційних відомостей, потрібних для роботи з конкретною дисципліною і групою. Далі, викладач обирає журнал, переходить до переліку занять та виконує внесення, редагування або перегляд записів. На етапі збереження, система повинна перевіряти правильність формату даних, допустимі межі значень, відповідність запису навчальному процесу та наявність обов’язкових зв’язків між сутностями. Лише після проходження перевірки, інформація може бути зафіксована в базі даних та використана для подальших розрахунків.

На рівні програмної логіки, обробка даних повинна здійснюватися за принципом послідовного проходження кількох взаємопов’язаних етапів. Спочатку система отримує вхідний запит користувача. Далі виконується перевірка права доступу до обраного журналу. Після підтвердження повноважень, відбувається завантаження поточних записів та довідкових відомостей, потрібних для відображення форми введення. На наступному етапі, здійснюється валідація значень, серед яких найбільше значення мають рамки оцінювання, унікальність окремих записів, коректність дати заняття та відповідність студента академічній групі. Після успішного завершення перевірки, система зберігає зміни, оновлює пов’язані агреговані показники, фіксує факт редагування в журналі дій та робить оновлені результати доступними для формування аналітики або звітів. UML-діаграму діяльності наведена на рисунку 3.4.

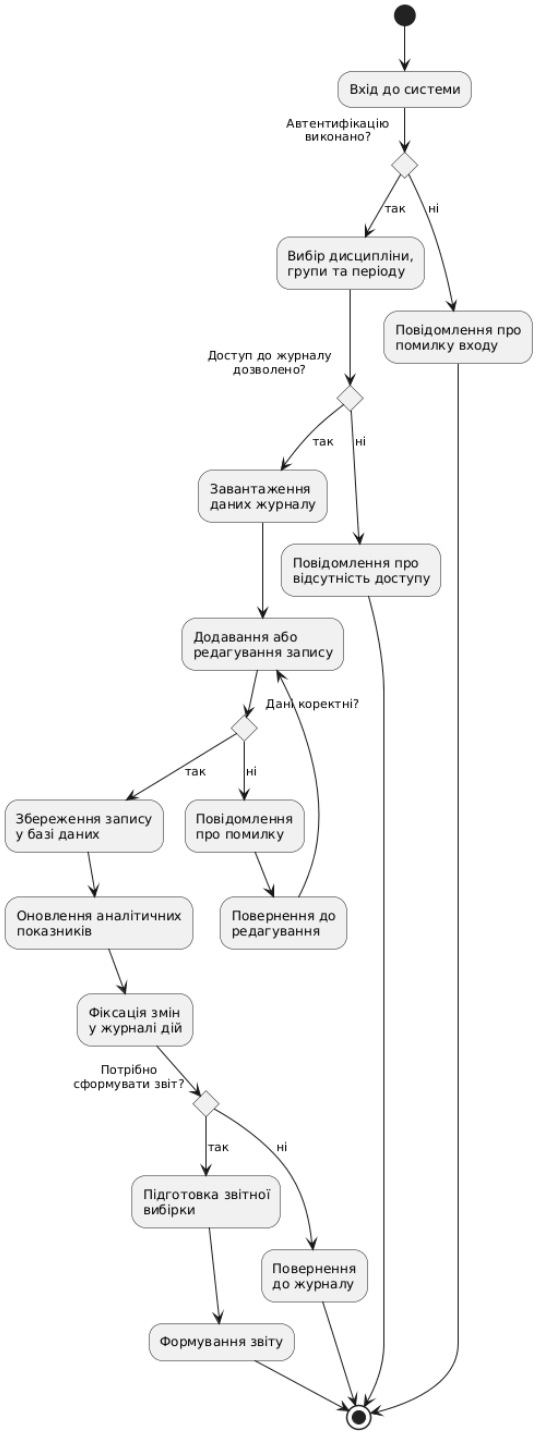

Рисунок 3.4 – UML-діаграма діяльності для логіки обробки даних в системі електронного журналу викладача

Кожна операція супроводжується контрольними процедурами, перевіркою зв’язків та подальшим оновленням похідних характеристик. Саме тому, програмний продукт може гарантувати не лише накопичення відомостей, а й їх придатність до аналітичного використання. При порушенні одного з

правил валідації, система повинна блокувати некоректне збереження, повідомляти користувача про помилку.

Для оцінок, значення має контроль допустимого діапазону балів відповідно до прийнятої шкали оцінювання. Для записів про відвідування важливо виключити дублювання позначок для одного студента в межах одного заняття. Для журналу, суттєвою є унікальність комбінації «викладач – група – дисципліна – семестр – навчальний рік», оскільки порушення даного правила створює ризик розповсюдження однорідних записів між кількома паралельними процесами. Значущим є і контроль каскадних залежностей, а саме, видалення журналу без попереднього опрацювання пов’язаних занять, оцінок і відвідування, може спричинити втрату цінної інформації, тому подібні операції повинні бути або заборонені, або реалізовані через строго контрольовані сценарії адміністрування.

Проєктування інформаційної системи електронного журналу викладача не може обмежуватися лише побудовою структури зберігання академічних записів, оскільки практична цінність програмного продукту визначається здатністю перетворювати нагромаджені відомості на аналітичну інформацію, придатну для педагогічної інтерпретації, управлінського контролю та підготовки звітних матеріалів. Важливим етапом архітектурного опрацювання постає формування моделі функціонування аналітичної підсистеми та механізмів звітності, які забезпечують перехід від первинного журналювання до узагальнення результатів навчального процесу. В структурі розроблюваної системи, аналітичний блок виконує роль інтелектуального ядра, яке акумулює дані з журналу, застосовує визначені правила обробки, розраховує показники та передає підготовлені результати до підсистеми формування звітів.

Аналітична підсистема повинна функціонувати на основі тісного зв’язку з модулем ведення журналу, проте без дублювання його призначення. Модуль журналювання, відповідає за введення, редагування та збереження оцінок, відміток про присутність, тем занять і довідкових позначок, тоді як аналітичний

блок працює з вже накопиченим масивом даних і виконує процедурне узагальнення.

Функціонування аналітичної підсистеми варто розглядати як послідовність взаємопов’язаних етапів. На початковому етапі, виконується отримування вхідних параметрів, серед яких найчастіше використовуються дисципліна, академічна група, часовий інтервал, вид контролю, тип підсумкового зрізу та формат представлення результату. Після одержання параметрів, система формує вибірку первинних даних з бази, перевіряє повноту записів, відкидає дублікати або технічно неприпустимі значення та переходить до розрахункового етапу. На стадії аналітичного опрацювання, виконуються обчислення середніх значень, рейтингових позицій, частки присутності, динаміки зміни результатів, частотних характеристик та порівняльних співвідношень між окремими наборами записів.

Після завершення розрахунків, підсистема передає структурований результат або до візуального представлення на екрані, або до механізму генерації звітного документа.

Частина академічних показників характеризує індивідуальні результати студента, зокрема середній бал, кількість отриманих оцінок, частку присутності та зміну показників в часі. Інша частина, відображає стан академічної групи в цілому, включаючи середній груповий результат, розподіл оцінок за рівнями, сумарну кількість пропусків та частку студентів з низькими або високими результатами. Окрему категорію становлять порівняльні показники, які дозволяють зіставляти результати кількох груп, дисциплін або навчальних періодів [25]. Основні аналітичні показники в системі електронного журналу викладача наведено в таблиці 3.4.

Таблиця 3.4 – Основні аналітичні показники в системі електронного журналу викладача

Аналітичний показник	Джерело формування	Призначення
Середній бал студента	поточні та підсумкові оцінки	відображає індивідуальний рівень навчальних досягнень
Середній бал групи	сукупність оцінок студентів однієї групи	характеризує загальний стан успішності у навчальному колективі
Частка відвідування	записи про присутність і пропуски	демонструє рівень навчальної дисципліни та регулярність участі у заняттях
Рейтинг студентів	інтегровані оцінки за визначений період	забезпечує впорядкування результатів за рівнем досягнень
Динаміка успішності	послідовність оцінок у часовому розрізі	дозволяє виявляти зростання, спад або нестабільність навчальних результатів
Динаміка відвідування	послідовність записів про присутність	дає змогу простежити зміни навчальної активності упродовж семестру
Розподіл оцінок	усі оцінки за обраним зрізом	використовується для структурного аналізу результатів
Порівняльний зріз	дані кількох груп, дисциплін або періодів	підтримує зіставлення академічних показників між різними сукупностями

Аналітична підсистема повинна працювати не з одним універсальним алгоритмом, а з набором обчислювальних процедур, кожна з яких орієнтована на певну управлінську або педагогічну потребу. Для оперативного контролю, важливими є середні значення та частка відвідування, для поглибленого аналізу – динамічні зрізи, структурні розподіли та порівняльні узагальнення, а для підготовки підсумкових матеріалів – рейтинги та інтегровані результати за обраний період.

Найдоцільнішим для розроблюваної системи є автоматизований режим актуалізації, за якого зміна первинного запису призводить до оновлення середніх значень, рейтингових позицій та похідних характеристик без додаткового втручання користувача. Даний механізм підтримує цілісність аналітичного середовища та знижує ризик розбіжностей між журналом та

зведеними матеріалами. UML-діаграма компонентів аналітичної підсистеми та механізмів звітності наведена на рисунку 3.5.

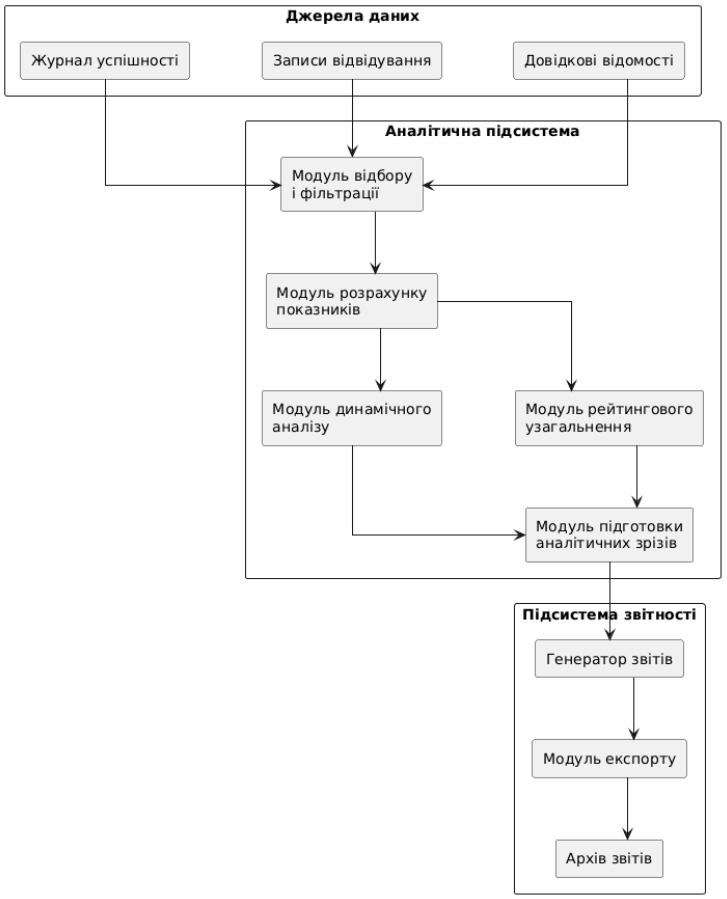

Рисунок 3.5 – UML-діаграма компонентів аналітичної підсистеми та механізмів звітності

Функціонування аналітичного блоку починається з етапу відбору і фільтрації, де з різних джерел даних формується узгоджений набір записів для подальшої обробки. Після завершення відбору, активується модуль розрахунку показників, який виконує базові статистичні процедури та передає результати до спеціалізованих сервісів динамічного аналізу та рейтингового узагальнення. Підсумкова інформація акумулюється в модулі підготовки аналітичних зрізів, звідки надходить до генератора звітів.

Процес формування звітного документа варто будувати за параметризованою моделлю. На першому етапі, користувач визначає тип звіту, часові межі, дисципліну, академічну групу, вид аналітичного зрізу та за потреби додаткові умови відбору. Після одержання параметрів, система

звертається до аналітичної підсистеми або безпосередньо до релевантної вибірки з бази даних, обчислює необхідні показники, впорядковує результат, накладає шаблон представлення та генерує підсумковий документ. Підсистема звітності не повинна повторно обчислювати складні аналітичні показники за наявності вже підготовлених зрізів. UML-діаграма послідовності формування звіту в системі електронного журналу викладача наведена на рисунку 2.6.

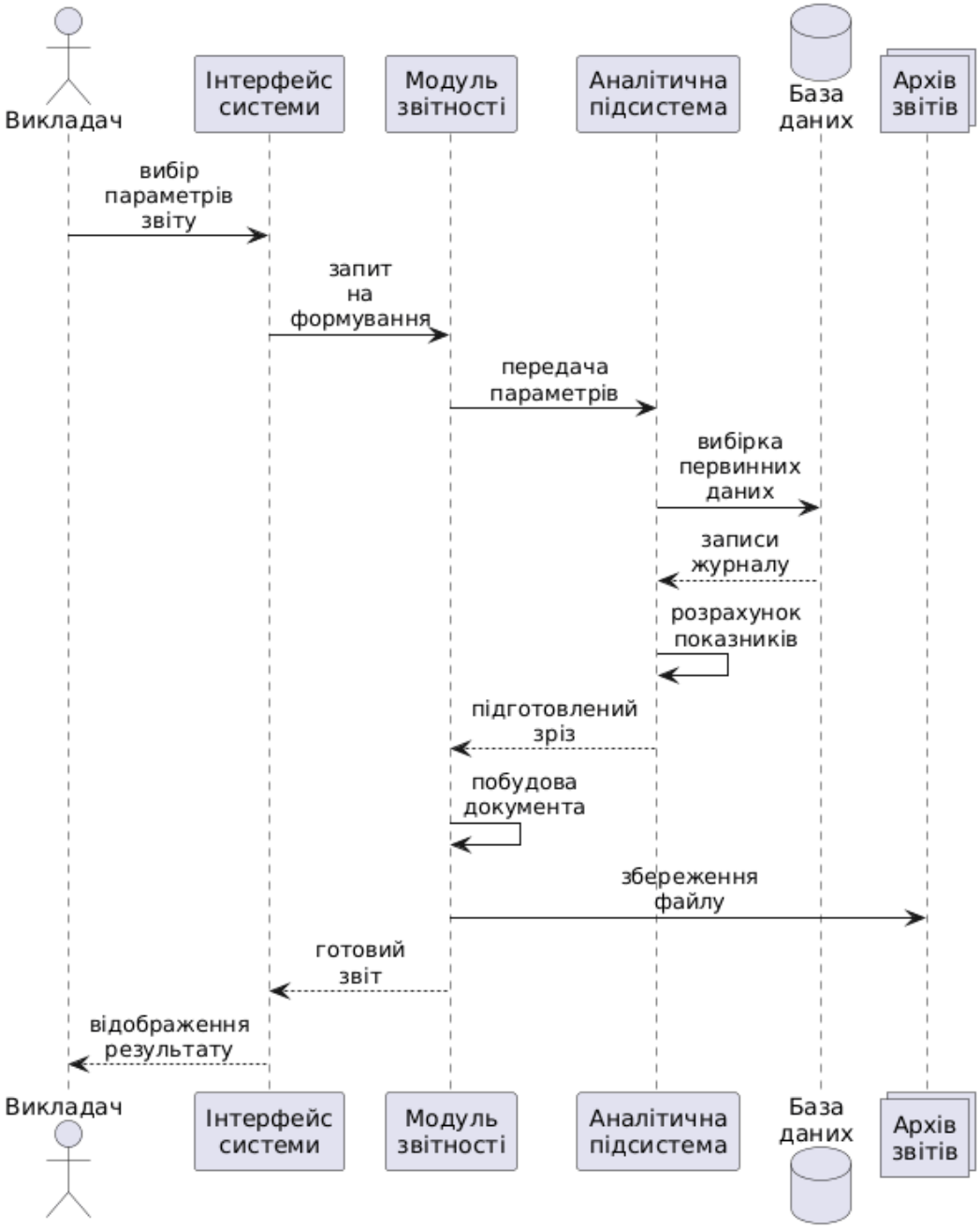

Рисунок 2.6 – UML-діаграма послідовності формування звіту в системі електронного журналу викладача

Ініціатором процесу виступає викладач, який задає параметри майбутнього документа через інтерфейс системи. Після надходження запиту, модуль звітності звертається до аналітичної підсистеми, а та, отримує первинні записи з бази даних та формує розрахунковий зріз. Далі, генератор звітів створює підсумковий документ у визначеному форматі, забезпечує його збереження в архіві та повертає готовий результат до інтерфейсу.

Будь-який сформований звіт має відображати не лише числові показники, а й контекст їх отримання, включаючи часовий інтервал, назву дисципліни, академічну групу, тип аналізу та момент формування документа. Наявність реквізитів полегшує перевірку коректності підсумкових матеріалів та знижує ймовірність помилкового використання зведених даних. Збереження звітів в архіві надає можливість повторного звернення до раніше сформованих документів без повторного запуску повного циклу обробки.

3.3 Обгрунтування вибору програмних засобів для реалізації завдання роботи

Реалізований програмний продукт побудовано мовою Python з використанням мікрофреймворку Flask, реляційної бази даних SQLite, шаблонів HTML, таблиць стилів CSS та клієнтських сценаріїв JavaScript для відображення візуальних аналітичних матеріалів.

Центральним серверним модулем виступає файл app.py, де зосереджено маршрутизацію запитів, логіку взаємодії з базою даних, механізми автентифікації, обробки журналу, запуск аналітичних обчислень, формування звітів та службові процедури контролю змін. Серверна частина взаємодіє з базою journal.db, де збережено ролі, облікові записи користувачів, відомості про викладачів та студентів, академічні групи, дисципліни, журнали, заняття, оцінки, відмітки про відвідування, записи аудиту та архів звітів.

Програмний продукт складається з кількох взаємопов’язаних компонентів, кожен з яких реалізує окремий напрям прикладної логіки. Один

блок відповідає за вхід в систему та рольове розмежування доступу. Інший блок забезпечує роботу викладача з електронним журналом, створення занять, збереження оцінок, відміток про присутність та коментарів. Окремо функціонує модуль аналітичного узагальнення, що формує середні значення, рейтингові зрізи, динаміку та розподіли. Підсистема звітності виконує генерацію файлів в форматі CSV та підтримує архів сформованих документів. Довідковий блок дозволяє адміністратору працювати з групами, дисциплінами, студентами та журналами. Для студента реалізовано індивідуальний перегляд власних результатів. Основні компоненти реалізованої програмної системи представлено в таблиці 3.6.

Базовим компонентом взаємодії користувача з системою є модуль автентифікації. Після запуску вебзастосунку, відкривається форма входу, де користувач вводить логін та пароль. Перевірка облікових даних виконується на серверному рівні з використанням хешованих паролів. Після успішної автентифікації, система зберігає ідентифікатор користувача в сесії та визначає подальший сценарій роботи залежно від ролі.

Таблиця 3.6 – Основні компоненти реалізованої програмної системи

Компонент	Програмна реалізація	Призначення
Серверна логіка	app.py	обробка маршрутів, взаємодія з базою даних, керування сесією, аналітика, формування звітів
База даних	journal.db	зберігання довідкових, операційних, аналітичних та службових даних
Шаблон базового інтерфейсу	templates/base.html	формування спільної структури сторінок, шапки, навігації та повідомлень
Модуль входу	templates/login.html	автентифікація користувача та запуск рольового сценарію роботи
Панель викладача	templates/dashboard_teacher.html	перегляд журналів викладача та швидкий перехід до журналу, аналітики та звітів
Модуль журналу	templates/journal.html	додавання занять, редагування оцінок, відвідуваності та коментарів
Модуль аналітики	templates/analytics.html, static/charts.js	відображення середніх значень, рейтингу, динаміки, розподілів і кореляції
Підсистема звітності	templates/reports.html, папка reports	генерація, збереження та повторне завантаження звітів
Адміністративний модуль	templates/admin_reference.html	робота з довідковими даними та створення журналів
Студентський модуль	templates/student_view.html	індивідуальний перегляд власних результатів та динаміки навчання
Засоби оформлення	static/style.css	стилізація сторінок та забезпечення єдиного візуального вигляду

4 ПРАКТИЧНА ЧАСТИНА

4.1 Опис процесу програмної реалізації

Після запуску програмного продукту користувач переходить на сторінку входу, де вводить особистий логін і пароль, та здійснює вхід до системи. Сторінка входу до інформаційної системи електронного журналу викладача наведена на рисунку 4.1.

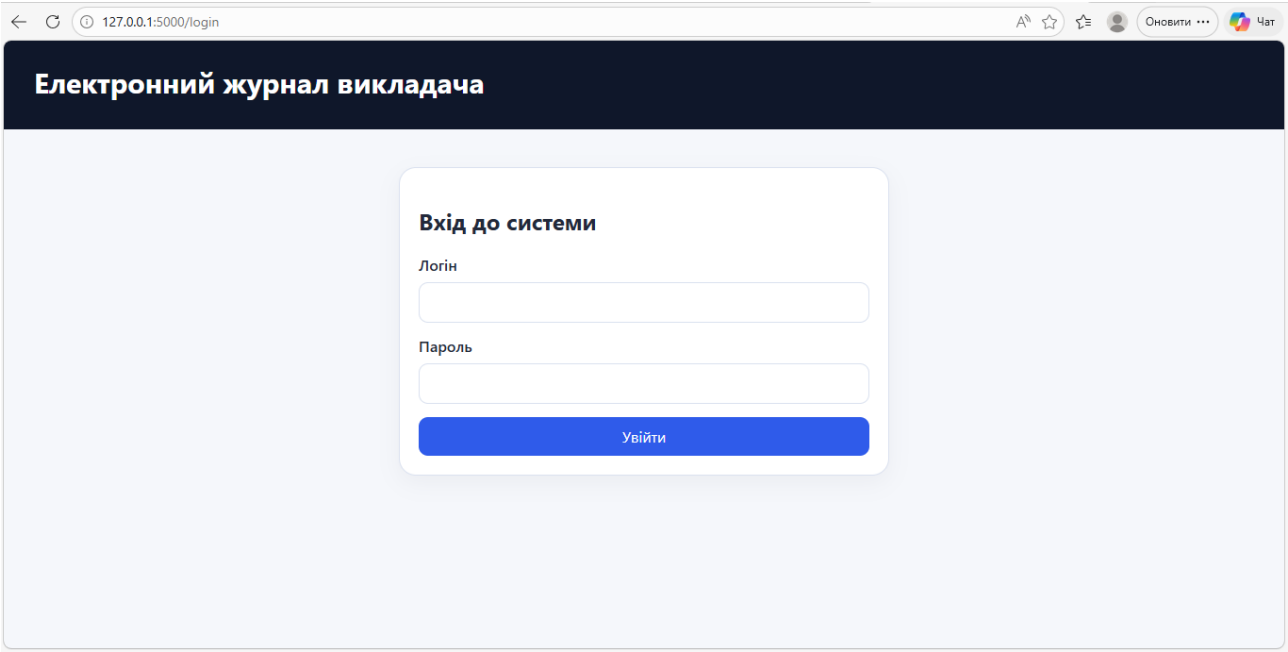

Рисунок 4.1 – Сторінка входу до інформаційної системи електронного журналу викладача

Після входу під роллю викладача відкривається персоналізована панель, де відображаються доступні журнали та порівняльний зріз за закріпленими дисциплінами та групами. На панелі реалізовано короткий маршрут до журналу, аналітики та звітності, завдяки чому скорочується час переходу між найвживанішими режимами [28]. Серверна частина отримує перелік журналів викладача з бази даних, після чого шаблон dashboard_teacher.html формує зведене представлення з табличною структурою. Панель викладача з переліком журналів та порівняльним зрізом наведена на рисунку 4.2.

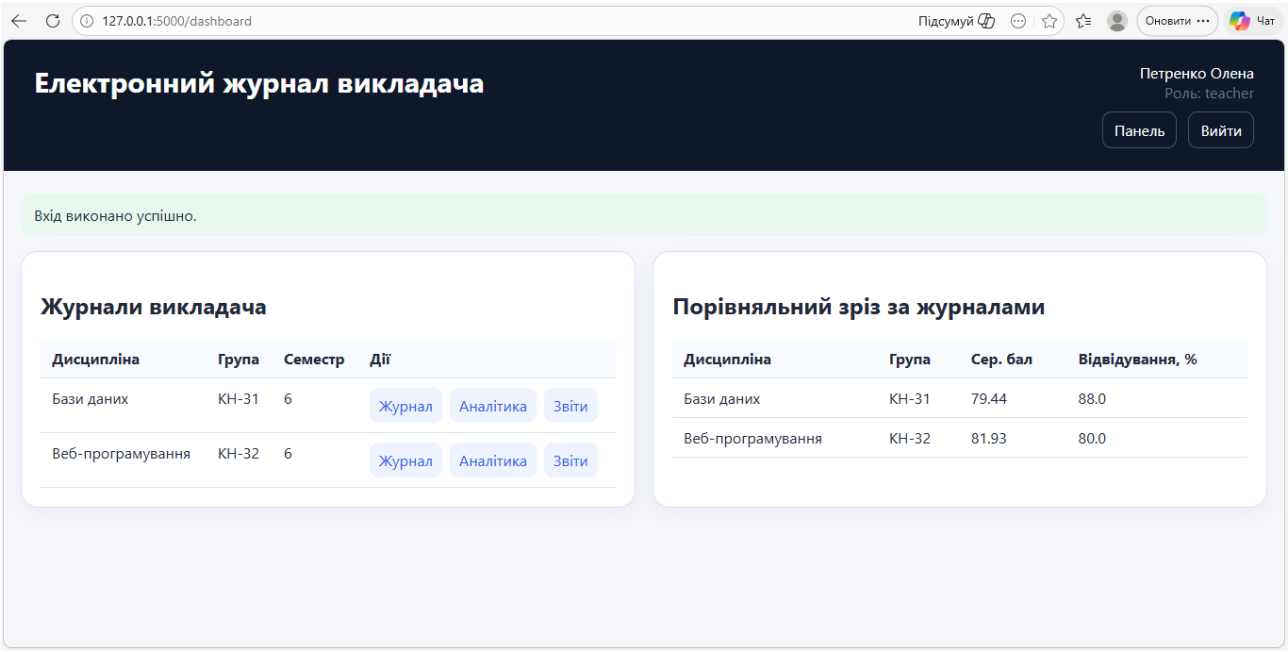

Рисунок 4.2 – Панель викладача з переліком журналів та порівняльним зрізом

Найбільш об’ємним за функціональним наповненням є модуль ведення електронного журналу. Реалізація даного компонента передбачає створення занять, вибір потрібного запису зі списку вже доданих занять, пошук студентів, фільтрацію даних за типом контролю, а також, збереження оцінок, статусів відвідування, причин відсутності та коментарів. Для кожного студента, в рамках обраного заняття формується окремий рядок таблиці, де зосереджено всі основні елементи поточного обліку. Під час збереження, система перевіряє межі балів, наявність обов’язкових атрибутів та коректність зв’язку між заняттям та студентом. Після успішного оновлення записів, до бази даних додаються або змінюються відповідні рядки в таблицях grades та attendance, а також створюється запис в журналі аудиту [28].

Практична цінність журнального модуля полягає в поєднанні двох рівнів роботи з даними. Перший рівень охоплює введення та коригування поточних академічних записів. Другий рівень забезпечує контекстне представлення, де користувач бачить зв’язок між датою, темою, типом заняття, оцінюванням та статусом присутності. Інтерфейс ведення електронного журналу з формою додавання заняття та таблицею записів наведено на рисунку 4.3.

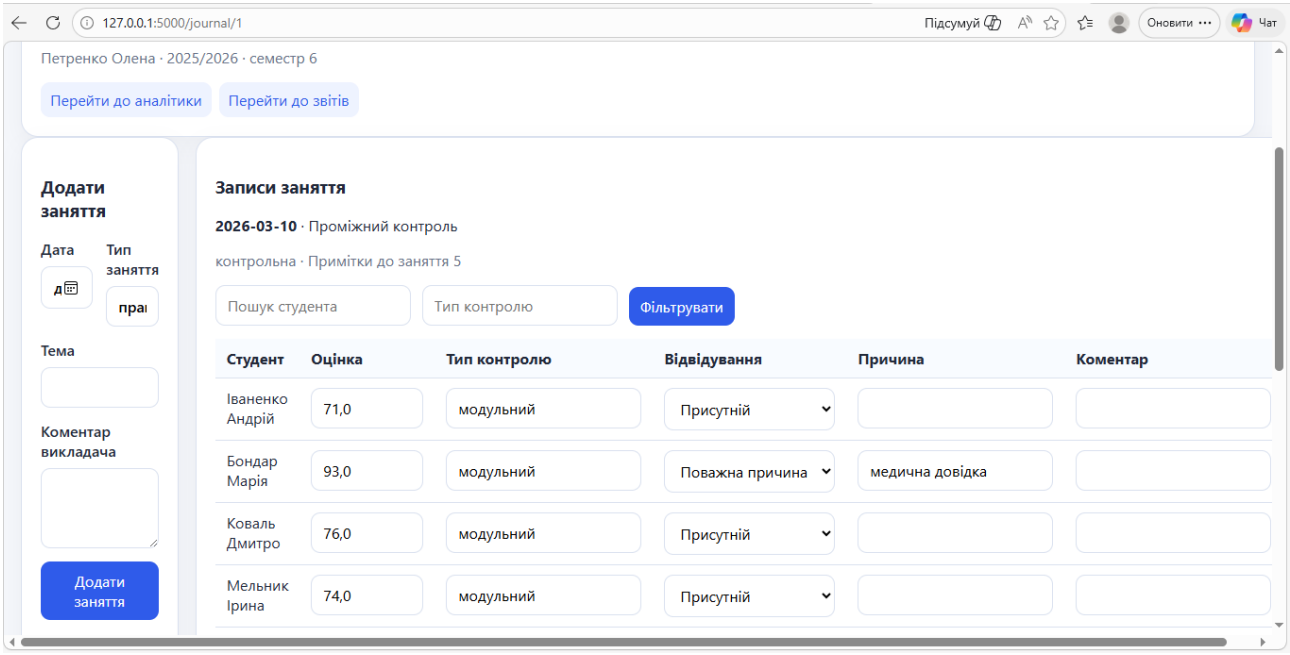

Рисунок 4.3 – Інтерфейс ведення електронного журналу з формою додавання заняття та таблицею записів

Окремим напрямом розробки став адміністративний модуль, призначений для підтримки довідкових даних. В даному блоці реалізовано додавання академічних груп, дисциплін, студентів і журналів. Представлене рішення має важливе значення для повноцінного функціонування системи, оскільки без коректного наповнення довідників, робота з журналом була б неможливою. Адміністратор отримує інтерфейс, де поєднано форми додавання сутностей та представлення поточного стану довідкової інформації. Під час збереження нових записів, використовується серверна перевірка унікальності та цілісності даних, зокрема щодо назв груп, кодів студентів та комбінацій параметрів журналу [28].

Довідкові сутності не існують ізольовано, а формують основу для створення журналів, занять та подальших аналітичних вибірок. Інтерфейс адміністратора орієнтований не лише на внесення нових відомостей, а й на контроль актуального стану довідників. За рахунок спільного шаблонного оформлення та єдиної серверної логіки даний блок органічно інтегрований в загальну систему. Адміністративний модуль роботи з довідковими даними наведено на рисунку 4.4.

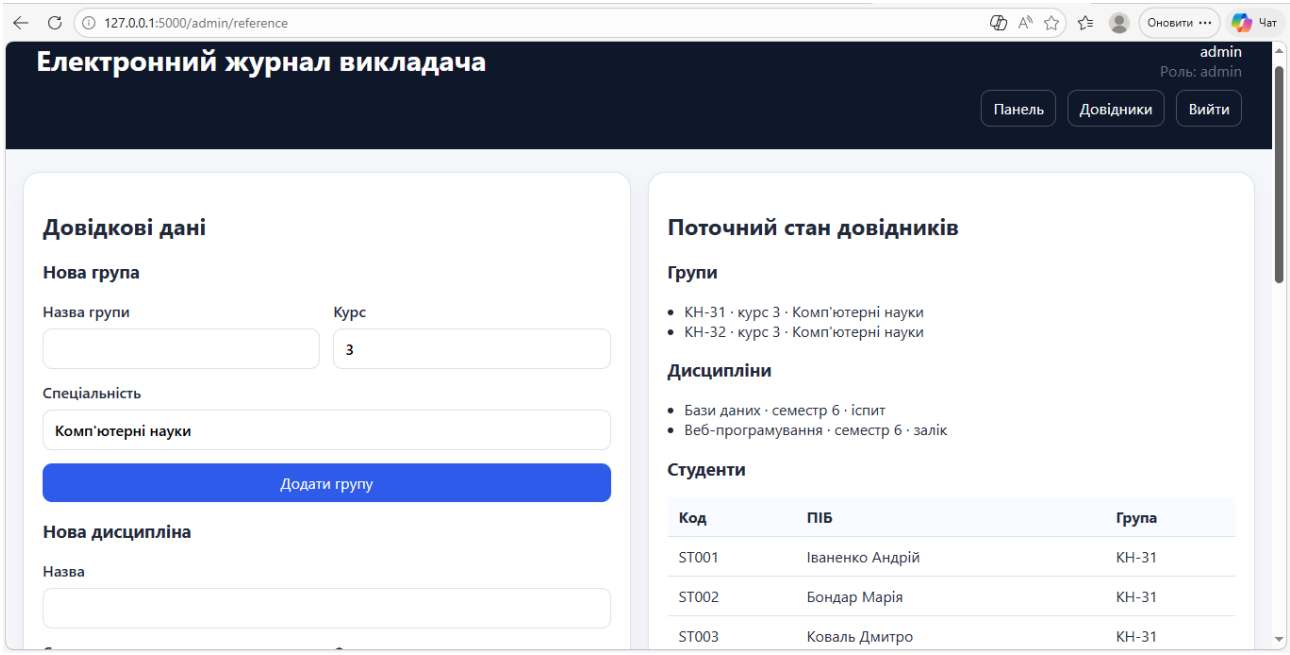

Рисунок 4.4 – Адміністративний модуль роботи з довідковими даними

Ще одним важливим компонентом став модуль перегляду індивідуальних результатів студента. Даний інтерфейс орієнтований на роль з мінімальним набором доступних функцій і не допускає втручання в журнал чи довідкові дані. На сторінці студента відображаються середній бал, відсоток відвідування, графік динаміки результатів та перелік записів, згрупованих за дисциплінами.

Інтерфейсний рівень всієї системи побудовано на основі спільного шаблону base.html, який задає шапку, навігаційний блок, область повідомлень та місце для вмісту окремих сторінок. Оформлення реалізовано у файлі style.css, де описано сіткову організацію сторінок, карткові блоки, таблиці, кнопки, повідомлення та адаптивні відступи [28].

В базі даних наявні таблиці ролей, користувачів, викладачів, студентів, груп, дисциплін, журналів, занять, оцінок, відвідуваності, звітів та аудиту змін. Для підтримки цілісності використано первинні ключі, зовнішні посилання, унікальні обмеження та заборону порушення зв’язків між сутностями. Наприклад, комбінація параметрів журналу для викладача, групи, дисципліни, навчального року і семестру не може дублюватися, а записи оцінювання та відвідування не допускають повторення для одного студента в межах одного заняття.

Окремої уваги заслуговує модуль журналу змін. В рамках серверної логіки реалізовано функцію log_change, яка фіксує користувача, тип сутності, ідентифікатор запису, характер дії, попередній стан та оновлене значення. Підтримка аудиту має прикладне значення для контролю редагувань, оскільки журнал викладача передбачає коригування оцінок і відміток про відвідування впродовж навчального періоду.

Під час розробки основних компонентів значну увагу приділено узгодженню серверного та клієнтського рівнів. Сервер формує дані для сторінок у вигляді структурованих об’єктів, а шаблони перетворюють їх на таблиці, форми та інформаційні блоки. Додатково використано файл charts.js, який обробляє підготовлені набори даних і відображає графічні елементи на сторінках аналітики та студентського перегляду [28]. Сторінка особистих результатів студента з узагальненими показниками та графіком динаміки наведена на рисунку 4.5.

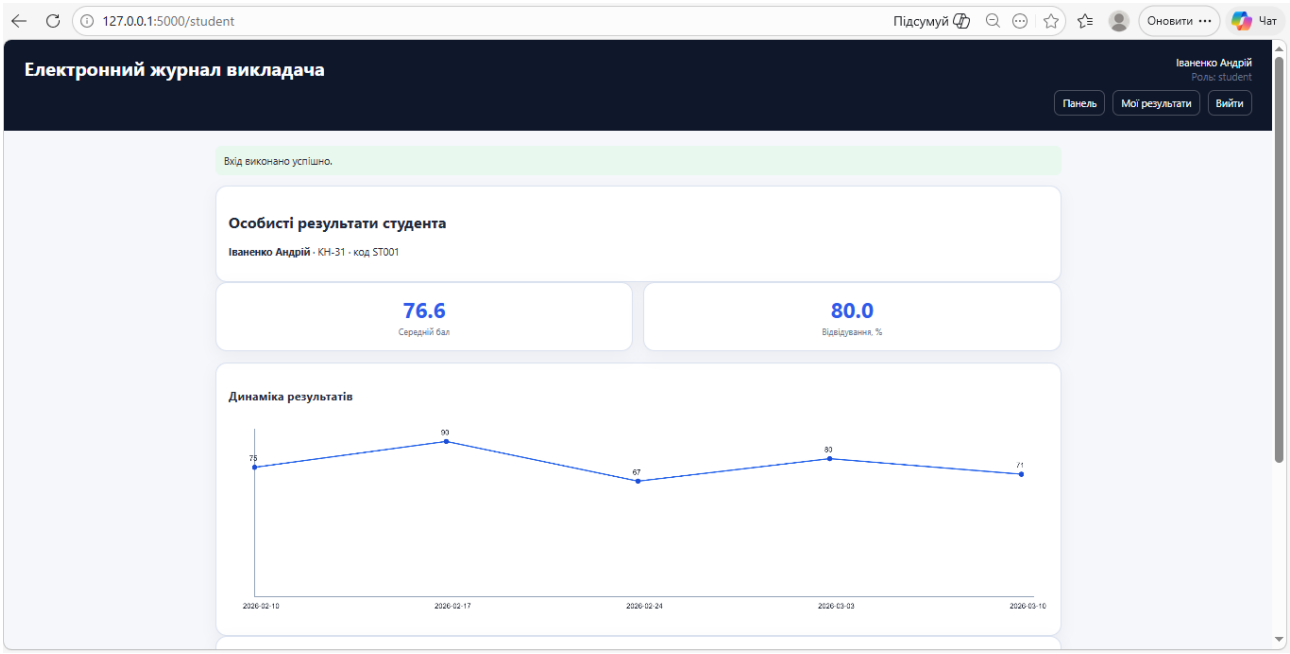

Рисунок 4.5 – Сторінка особистих результатів студента з узагальненими показниками та графіком динаміки

Реалізація основних компонентів програмної системи підтвердила можливість практичного втілення архітектури, моделі даних та рольової структури. Створений застосунок охоплює вхід до системи, персоналізований

доступ за ролями, ведення журналу, підтримку довідкових сутностей, індивідуальний перегляд результатів, механізм аудиту та основу для аналітичних і звітних процедур.

4.2 Опис програми

В рамках створеного вебзастосунку, аналітичні алгоритми реалізовано в серверній частині на Python, де виконується вибірка даних з реляційної бази, їх групування, обчислення похідних показників, підготовка структур для візуалізації та передавання результатів до шаблонів інтерфейсу. Генерація звітів також виконується на серверному рівні, що дозволяє забезпечити єдність логіки обробки, повторюваність результатів та контроль коректності вихідних документів.

Аналітичний модуль в розробленій системі орієнтований на роботу з кількома групами показників. До першої групи належать узагальнення успішності, серед яких середній бал студента, середній бал групи, загальний середній показник за журналом та рейтингове впорядкування результатів. Друга група охоплює аналіз відвідуваності, зокрема кількість відвіданих занять, кількість пропусків та частку присутності у відсотковому відношені. Третя група пов’язана з часовим аспектом та забезпечує побудову динаміки оцінювання та змін навчальної активності впродовж послідовності занять. Наступна група формує структурні та порівняльні зрізи, серед яких розподіл оцінок за категоріями, порівняння результатів різних журналів, оцінювання зв’язку між академічними досягненнями та відвідуваністю [28].

Програмна реалізація аналітичних алгоритмів ґрунтується на поетапному опрацюванні масиву записів. На першому етапі сервер формує SQL-запити до таблиць оцінок, відвідування, занять, студентів і журналів. На другому етапі виконується очищення та впорядкування вибірки, у межах якого відкидаються порожні значення, уточнюється належність записів до визначеного журналу та забезпечується правильна структура. На третьому етапі запускаються обчислювальні процедури, спрямовані на формування середніх, відсотків,

рейтингів, розподілів та кореляційних характеристик. На завершальному етапі підготовлені результати перетворюються на JSON-подібні структури або словники Python, після чого передаються до HTML-шаблонів та клієнтського сценарію charts.js, який відповідає за візуальне відображення графіків.

Для кожного студента з сукупності оцінок формується окремий піднабір значень, після чого виконується обчислення арифметичного середнього. Якщо для конкретного студента у вибірці відсутні оцінки, система не генерує помилкового нульового значення, а подає порожній або нейтральний результат. Після обчислення індивідуальних середніх, сервер формує груповий середній показник, який використовується в аналітичних картках, на інформаційній панелі викладача та в підсумкових звітах.

Програмна логіка опрацьовує записи таблиці attendance, де для кожного студента фіксується статус присутності або відсутності. Під час розрахунку, система визначає загальну кількість занять, для яких існує запис про відвідування, після чого обчислює відношення кількості присутностей до загального обсягу записів. Отриманий результат переводиться у відсоткову форму та використовується як окремий показник в картках аналітики, студентському профілі та звітних документах [28].

Рейтингова аналітика реалізована через впорядкування студентів за інтегральним середнім балом. Після завершення етапу обчислення індивідуальних середніх, система формує список студентів з відповідними числовими значеннями та виконує сортування за спаданням. За рахунок рейтингового узагальнення викладач отримує швидкий доступ до впорядкованої інформації про результати навчання без потреби самостійно переглядати кожен рядок журналу.

Для відображення зміни результатів у часі реалізовано алгоритм побудови динаміки оцінювання. Серверна частина виконує вибірку оцінок в прив’язці до дат занять, після чого сортує їх в хронологічному порядку. На основі впорядкованої послідовності створюються два масиви, перелік дат і

перелік відповідних значень. Далі масиви передаються до клієнтського рівня, де за допомогою JavaScript формуються лінійні графіки.

Структурний аналіз результатів виконано через алгоритм розподілу оцінок за категоріями. В створеній системі оцінки розбиваються на кілька інтервалів, що дозволяє узагальнити картину успішності не лише через середнє значення, а й через внутрішню будову результатів. Сервер підраховує кількість оцінок в кожній категорії, після чого передає підготовлену структуру для побудови стовпчикової діаграми.

Для глибшого опрацювання взаємозв'язку між навчальними досягненнями та відвідуваністю в програмному продукті реалізовано алгоритм кореляційного аналізу. На серверному рівні формуються дві числові послідовності, а саме, індивідуальні середні бали студентів та відсотки їх відвідування. Після підготовки обох масивів застосовується формула коефіцієнту Пірсона, яка дозволяє оцінити характер зв'язку між даними параметрами. При недостатній кількості даних, система уникає хибного числового результату та повертає нейтральне повідомлення про неможливість повноцінного розрахунку. Реалізовані алгоритми аналізу в програмній системі узагальнено в таблиці 4.1.

Таблиця 4.1 – Реалізовані алгоритми аналізу в програмній системі

Алгоритм	Вхідні дані	Результат виконання
Розрахунок середнього бала	оцінки студента, групи або журналу	індивідуальне та групове середнє значення
Обчислення частки відвідування	записи про присутність і відсутність	відсоток відвідування за студентом або групою
Рейтингова обробка	середні бали студентів	ранжований перелік результатів
Побудова динаміки оцінювання	оцінки у хронологічному порядку	масиви для лінійного графіка змін
Розподіл оцінок	повна вибірка балів журналу	кількість оцінок у категоріях
Кореляційний аналіз	середній бал і відсоток відвідування	коефіцієнт зв'язку між параметрами
Порівняльний зріз	показники кількох журналів	таблиця для зіставлення груп і дисциплін

Реалізація аналітичного модуля у вебзастосунку представлена в окремій сторінці аналітики, де зведено ключові показники, графіки та порівняльні дані. На даній сторінці відображаються картки з середнім балом, часткою відвідування, значенням кореляції, таблиця рейтингу, блок розподілу оцінок та графічні матеріали, побудовані на основі підготовлених масивів. Серверна логіка виконує обчислення, а клієнтська частина лише візуалізує вже підготовлені результати. Сторінка аналітики з узагальненими показниками, рейтингом та графіками представлена на рисунку 4.6.

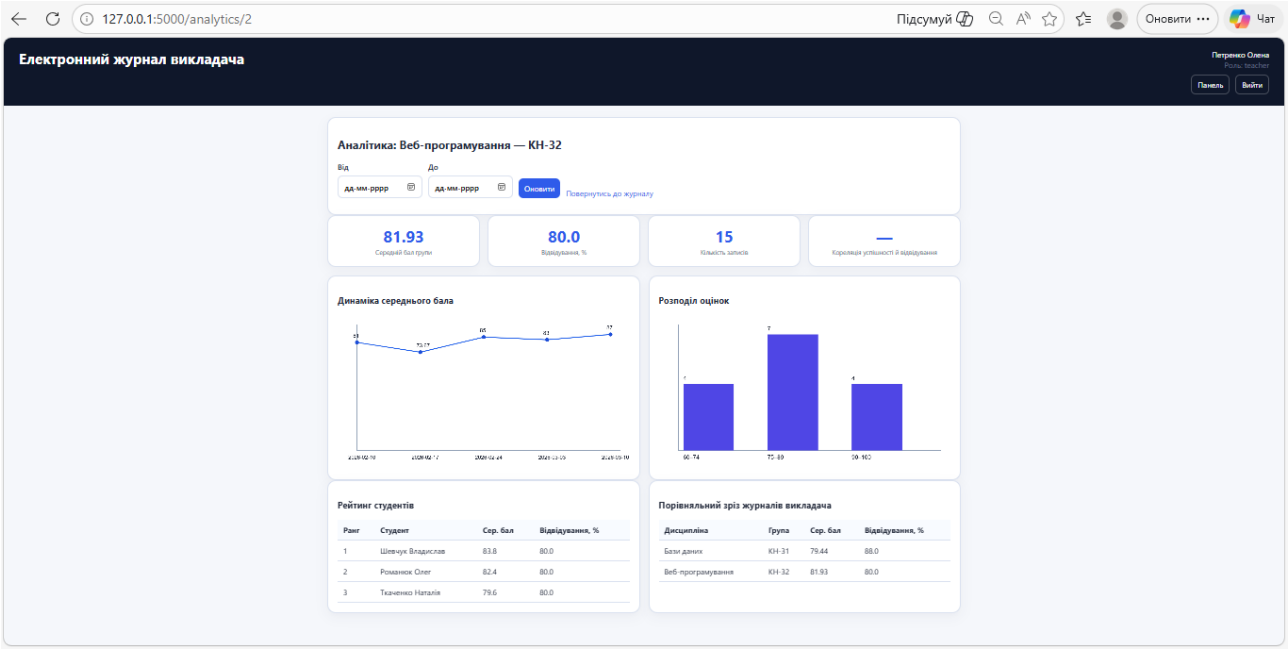

Рисунок 4.6 – Сторінка аналітики з узагальненими показниками, рейтингом і графіками

Наступним важливим напрямом програмної реалізації є підсистема формування звітів. В створеному застосунку, підситсему побудовано на основі генерації CSV-файлів, що дозволяє зберігати звітний документ в табличному форматі, придатному для відкриття в електронних таблицях, подальшого друку або архівуванню. Серверна частина після одержання запиту користувача визначає тип документу, формує відповідну вибірку даних, обчислює потрібні показники, створює файл в папці reports, після чого додає службовий запис до таблиці reports_archive.

Механізм звітності підтримує кілька типів вихідних документів. Один тип охоплює зведений журнал з переліком занять, оцінок та статусів відвідування.

Інший тип спрямований на формування рейтингового звіту за результатами обраного журналу. Окремо реалізовано аналітичний CSV-документ, де подаються основні узагальнення щодо середніх значень, частки присутності та інших похідних характеристик. Вибір конкретного формату виконується через сторінку звітності, де користувач задає параметри побудови, а серверна частина запускає відповідну процедуру підготовки даних.

Після відкриття сторінки звітності користувач обирає потрібний журнал і тип документу. Далі система отримує первинну або аналітично підготовлену вибірку з бази даних, після чого створює файл з табличною структурою рядків та стовпців. В сформований документ включаються не лише числові значення, а й контекстні атрибути, серед яких назва дисципліни, академічна група, дата формування та тип звіту. Після запису файлу на носій, система вносить відомості до архіву, що дозволяє повторно завантажити документ без повторного запуску повного циклу розрахунків.

Кожен звіт отримує запис з зазначенням користувача, журналу, типу документа, шляху до файлу та моменту формування. Наявність архіву має не лише технічне, а й організаційне значення, оскільки дозволяє викладачеві або адміністратору повертатися до раніше створених матеріалів, відстежувати послідовність побудови звітів та підтверджувати факт підготовки підсумкових документів. Реалізовані механізми формування звітів узагальнено в таблиці 4.2.

Таблиця 4.2 – Реалізовані механізми формування звітів

Тип звіту	Джерело даних	Вихідний результат
Зведений журнал	оцінки, відвідування, заняття, студенти	CSV-файл з повним набором журнальних записів
Рейтинг групи	середні бали студентів	CSV-файл з ранжованим переліком студентів
Аналітичний звіт	агреговані показники журналу	CSV-файл з середніми значеннями та узагальненнями
Архів звітів	таблиця службових записів	список доступних для повторного завантаження документів

Інтерфейс сторінки звітності побудовано так, що користувачеві не потрібно виконувати складні технічні дії для створення документу. Достатньо обрати журнал і тип звіту, після чого серверна частина виконує всі подальші процедури автоматично. Після побудови документу сторінка відразу відображає оновлений архів, що забезпечує наочний контроль результату виконання операції. Сторінка формування звітів з вибором типу документа та архівом файлів представлена на рисунку 4.7.

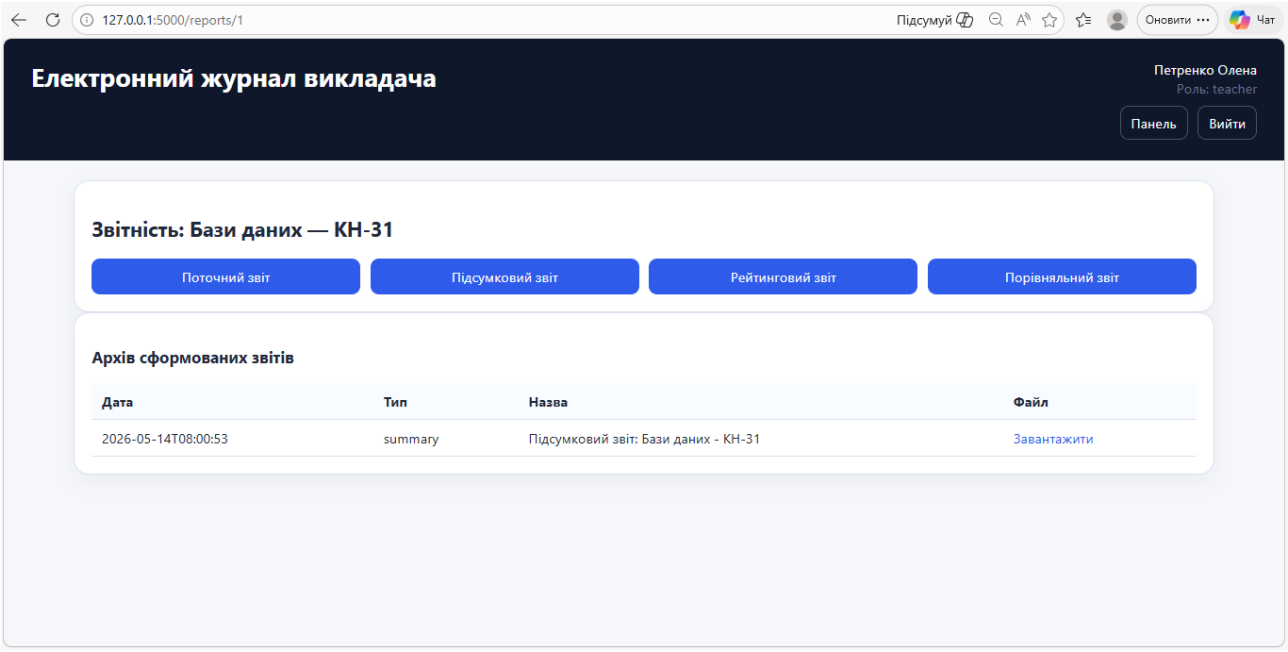

Рисунок 4.7 – Сторінка формування звітів з вибором типу документа та архівом файлів

Програмна реалізація алгоритмів аналізу та звітності тісно пов’язана з модулем аудиту змін. Після оновлення оцінок, відвідування або службових сутностей, система виконує фіксацію операції в журналі дій, що дозволяє співвідносити аналітичні та звітні результати з фактом модифікації первинних даних.

В програмному продукті справді забезпечено відбір вхідних даних, обчислення похідних характеристик, підготовку візуальних матеріалів, побудову структурованих документів та збереження їх в архіві.

4.3 Перевірка валідності. Дослідження можливостей програмної реалізації

Завершальний етап практичної частини був спрямований на перевірку працездатності розробленої інформаційної системи, оцінку коректності виконання аналітичних розрахунків, контроль формування звітних документів. Під час перевірки увагу зосереджено на трьох взаємопов'язаних напрямах: правильність обробки академічних записів, достовірність аналітичних висновків та стабільність функціонування рольової моделі доступу.

Експериментальну перевірку виконано на локальному персональному комп'ютері після розгортання вебзастосунку в середовищі Python та запуску серверної частини Flask [29]. Для зберігання даних використано реляційну базу SQLite, а взаємодія з системою здійснювалася через веббраузер. В базі було підготовлено довідкові записи про ролі користувачів, викладача, студента, академічну групу, дисципліну, журнал та набір занять. Після запуску застосунку виконано серію контрольних дій, пов'язаних з входом під різними ролями, внесенням оцінок, позначенням присутності, коригуванням записів, переглядом аналітики та генерацією звітів [30].

Контрольний набір даних охоплював чотирьох студентів однієї академічної групи, для яких у журналі було зафіксовано результати чотирьох занять. Контрольний набір академічних записів для експериментальної перевірки представлено в таблиці 4.3.

Таблиця 4.3 – Контрольний набір академічних записів для експериментальної перевірки

Студент	Внесені оцінки	Кількість присутностей	Кількість занять	Очікуваний середній бал	Очікувана частка відвідування, %
Студент 1	90, 88, 95, 92	4	4	91,25	100,00
Студент 2	74, 80, 76, 78	3	4	77,00	75,00
Студент 3	60, 65, 70, 62	2	4	64,25	50,00
Студент 4	95, 97, 94, 96	4	4	95,50	100,00

Після введення наведених записів в журнал, виконано контроль розрахунків та порівняння з результатами, отриманими в аналітичному модулі. Розрахунок середнього балу було проведено як арифметичне середнє для кожного студента окремо, а груповий показник визначено як середнє значення по всіх чотирьох студентах. Частка відвідування обчислювалась як відношення кількості відвідувань до кількості занять, помножене на 100. Рейтингова послідовність формувалася за спаданням індивідуального середнього балу. Для оцінювання зв’язку між успішністю та відвідуваністю, виконано кореляційний розрахунок, який продемонстрував виражений позитивний зв’язок між параметрами. Порівняння контрольних розрахунків та результатів, одержаних в системі наведено в таблиці 4.4.

Таблиця 4.4 – Порівняння контрольних розрахунків та результатів, одержаних в системі

Показник	Контрольний розрахунок	Результат в системі
Середній бал студента 1	91,25	91,25
Середній бал студента 2	77,00	77,00
Середній бал студента 3	64,25	64,25
Середній бал студента 4	95,50	95,50
Середній бал групи	82,00	82,00
Частка відвідування студента 2, %	75,00	75,00
Частка відвідування студента 3, %	50,00	50,00
Рейтингова послідовність	4–1–2–3	4–1–2–3
Кореляція між успішністю та відвідуванням	0,991	0,991

Результати засвідчили правильність реалізації основних алгоритмів аналітичного блоку. Розбіжностей між контрольними обчисленнями та значеннями, сформованими програмною системою, не виявлено. Найбільшу практичну вагу має не лише збіг середніх значень, а й коректність рейтингування, побудови відсоткових характеристик відвідування та кореляційного узагальнення, оскільки дані показники формують основу підсистеми аналітики та звітності.

Наступний етап перевірки був спрямований на контроль функціональної повноти системи. Проведено серію тестових сценаріїв, пов’язаних з рольовим доступом, введенням та коригуванням даних, автоматичним оновленням аналітики, генерацією звітів та архівуванням результатів. Під час виконання сценаріїв, оцінювалась не лише наявність потрібної функції, а й логічна завершеність дії, коректність переходу між сторінками, відсутність збоїв та узгодженість отриманих результатів з вмістом бази даних. Результати функціональної перевірки системи наведено в таблиці 4.5.

Таблиця 4.5 – Результати функціональної перевірки системи

Тестовий сценарій	Очікуваний результат
Вхід під роллю викладача	відкриття персоналізованої панелі з доступними журналами
Вхід під роллю адміністратора	відкриття сторінки довідкових даних і журналів
Вхід під роллю студента	відкриття сторінки індивідуальних результатів без прав редагування
Додавання нового заняття	поява заняття в журналі з коректними атрибутами дати, теми й типу
Внесення оцінок та відвідування	збереження записів в базі даних та відображення в таблиці журналу
Коригування оцінки	оновлення запису й фіксація зміни в журналі дій
Перехід до сторінки аналітики	відображення середніх значень, рейтингу, графіків і розподілу
Формування CSV-звіту	створення файлу та поява запису в архіві звітів
Повторне завантаження готового звіту	доступ до збереженого документа з архіву
Спроба студента редагувати журнал	блокування доступу до службових функцій

Рольова модель доступу працює коректно, журнал підтримує створення та коригування академічних записів, аналітичний блок формує узагальнення без порушення логіки предметної області, а підсистема звітності успішно створює та зберігає вихідні документи. Особливо важливим результатом стала коректна синхронізація між редагуванням журналу та автоматичним оновленням

аналітичних показників. Після зміни оцінки або статусу присутності, користувач одразу отримує оновлене середовище аналітики без потреби повторного введення або зовнішнього перерахунку.

Окремий блок експериментальної перевірки стосувався надійності зберігання та прозорості змін. Після коригування оцінок, додавання занять та генерації звітів система формувала записи в журналі аудиту, де зберігалися ідентифікатор користувача, тип сутності, дія, попереднє та нове значення. Наявність журналу змін підтвердила, що програмний продукт підтримує контроль редагувань, а отже, забезпечує додатковий рівень достовірності під час роботи з академічною інформацією.

Під час перевірки підсистеми звітності було встановлено, що система коректно формує файли в форматі CSV для різних типів документів. Згенеровані файли відкривалися в табличному редакторі без втрати структури, містили службові реквізити журналу та відображали вміст, узгоджений з поточним станом бази даних. Архів звітів зберігав інформацію про тип документу, журнал, автора формування та шлях до файлу, завдяки чому забезпечувався повторний доступ до підготовлених матеріалів.

Підсистема ведення журналу забезпечує роботу з оцінюванням, відвідуванням, темами занять та коментарями. Рольова організація доступу підтримує розмежування повноважень між викладачем, адміністратором та студентом. Аналітичний модуль коректно обчислює середні значення, рейтинги, частку присутності, динамічні та кореляційні показники. Підсистема звітності створює структуровані документи та підтримує архівування результатів. Сукупність одержаних даних підтвердила працездатність програмного продукту, логічну завершеність взаємодії його компонентів та придатність до використання як інформаційної системи електронного журналу викладача з розширеним функціоналом аналітики та звітності.

4.4 Необхідна користувачу програми інструкція

Для початку роботи з програмою користувачеві потрібно розгорнути застосунок в локальному середовищі. Після розпакування архіву з проєктом слід відкрити командний рядок в каталозі програми, створити віртуальне середовище Python, активувати його, встановити залежності з файла requirements.txt і запустити головний файл app.py. Після запуску серверної частини програма стає доступною у веббраузері за локальною адресою http://127.0.0.1:5000, де відкривається сторінка входу до системи.

Подальша робота починається з автентифікації. Після введення логіну та паролю користувач потрапляє до робочого середовища, що відповідає його ролі. Адміністратор отримує доступ до довідкових даних, створення викладачів, студентів, груп, дисциплін і журналів. Викладач працює з електронним журналом, додає заняття, вносить оцінки, позначає відвідування, переглядає аналітичні матеріали, формує звіти та за потреби оновлює власні облікові відомості. Студент має доступ лише до особистих результатів без права редагування журналу або довідкових сутностей.

Під час ведення журналу викладач спочатку обирає потрібний журнал, після чого переходить до списку занять. Для нового заняття задаються дата, тема і тип. Далі в табличній формі вносяться оцінки, статус присутності та коментарі щодо кожного студента. Після збереження записи потрапляють до бази даних і одразу використовуються в аналітичному модулі. В розділі аналітики відображаються середні значення, рейтинг, показники відвідування, динаміка результатів і розподіл оцінок. В розділі звітності користувач обирає тип документа, запускає формування файла і за потреби повторно завантажує його з архіву.

Для завершення роботи достатньо скористатися командою виходу з облікового запису в інтерфейсі системи. У разі припинення роботи сервера локальний застосунок зупиняється стандартною командою переривання у вікні командного рядка.

ВИСНОВКИ

Проведене дослідження було спрямоване на розв’язання актуального завдання, пов’язаного з розробкою інформаційної системи електронного журналу викладача, здатної поєднати функції академічного обліку, аналітичного опрацювання результатів навчання, контролю відвідуваності та формування звітних матеріалів. Потреба в створенні подібного програмного продукту зумовлена зростанням обсягу освітніх даних, поширенням цифрових форм організації навчального процесу та необхідністю переходу від фрагментарного збереження записів до цілісного інформаційного середовища підтримки педагогічних рішень.

В ході дослідження встановлено, що електронний журнал викладача слід розглядати не як простий цифровий аналог паперового документу, а як спеціалізовану інформаційну систему, де первинні академічні записи виступають джерелом для подальшого аналізу та документального узагальнення. Теоретичне опрацювання предметної області дозволило визначити місце освітніх даних в структурі сучасного цифрового середовища, виявити їх багаторівневий характер, обґрунтувати значення показників успішності та відвідуваності, з’ясувати роль статистичних, порівняльних, рейтингових, динамічних та кореляційних методів в навчальній аналітиці.

На основі аналізу предметної області сформовано систему функціональних і нефункціональних вимог до програмного продукту, визначено потребу в підтримці автентифікації, рольового доступу, роботи з довідковими даними, ведення журналу, аналітичного опрацювання, пошуку, візуального подання результатів, збереження історії змін та формування звітів. Паралельно встановлено перелік якісних характеристик, серед яких провідне значення мають надійність, продуктивність, безпека, зручність використання, масштабованість, сумісність і супроводжуваність.

В рамках проєктування, побудовано багаторівневу веборієнтовану клієнт-серверну архітектуру, визначено склад основних компонентів, сформовано

інформаційну модель даних та розроблено логіку обробки академічних записів. Центральну роль в структурі системи відведено сутності журналу, яка об’єднує викладача, дисципліну, академічну групу та навчальний період. На основі моделі даних обґрунтовано порядок збереження занять, оцінок, відвідування, звітів і службових відомостей.

Практичний етап завершився створенням працездатного вебзастосунку на основі Python, Flask, SQLite, HTML, CSS і JavaScript. Реалізовано вхід до системи, персоналізовану роботу за ролями адміністратора, викладача та студента, ведення електронного журналу, внесення та коригування оцінок і відміток про присутність, підтримку довідкових сутностей, журнал аудиту, аналітичний модуль і підсистему звітності. В програмному продукті реалізовано алгоритми обчислення середніх значень, рейтингових показників, частки відвідування, динамічних характеристик, структурних розподілів та кореляційних залежностей, а також, побудову звітних документів з архівуванням результатів.

Експериментальна перевірка підтвердила правильність функціонування розробленої системи. Контрольні обчислення повністю збіглися з результатами, сформованими програмним продуктом. Функціональні сценарії показали коректну роботу рольової моделі доступу, стабільність збереження записів, своєчасне оновлення аналітики після редагування даних, успішну генерацію звітів і належну фіксацію змін в журналі дій.

Сукупність одержаних результатів підтвердила досягнення поставленої мети і засвідчила, що створена інформаційна система може використовуватися як практичний інструмент підтримки діяльності викладача в закладах вищої та фахової передвищої освіти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Siemens G. Learning Analytics: The Emergence of a Discipline. *American Behavioral Scientist*. 2013. Vol. 57, No. 10. P. 1380-1400. DOI: 10.1177/0002764213498851.
2. Про освіту : Закон України від 05.09.2017 № 2145-VIII // База даних «Законодавство України» / Верховна Рада України.
3. Про вищу освіту : Закон України від 01.07.2014 № 1556-VII // База даних «Законодавство України» / Верховна Рада України.
4. Long P., Siemens G. Penetrating the Fog: Analytics in Learning and Education. *EDUCAUSE Review*. 2011. Sept./Oct.
5. Арешонков В. Ю. Цифровізація вищої освіти: виклики та відповіді. *Вісник Національної академії педагогічних наук України*. 2020. Т. 2, № 2. DOI: 10.37472/2707-305X-2020-2-2-13-2.
6. Бодненко Д. М., Жильцов О. Б., Лещинський О. Л., Мазур Н. П. *Моніторинг навчальної діяльності : навч. посіб.* Київ : Київський університет імені Бориса Грінченка, 2014. 276 с.
7. Басюк Т. О. та ін. *Цифрова трансформація освіти: теоретико-методичні засади : монографія* / за заг. ред. В. П. Сергієнка ; за наук. ред. Н. П. Франчук. Київ : Вид-во УДУ імені Михайла Драгоманова, 2024. 382 с.
8. Ferguson R. Learning Analytics: Drivers, Developments and Challenges. *International Journal of Technology Enhanced Learning*. 2012. Vol. 4, No. 5/6. P. 304–317. DOI: 10.1504/IJTEL.2012.051816.
9. Buckingham Shum S., Ferguson R. Social Learning Analytics. *Educational Technology & Society*. 2012. Vol. 15, No. 3. P. 3-26.
10. Romero C., Ventura S. Educational Data Mining: A Review of the State of the Art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*. 2010. Vol. 40, No. 6. P. 601-618. DOI: 10.1109/TSMCC.2010.2053532.

11. Chatti M. A., Dyckhoff A. L., Schroeder U., Thüs H. A Reference Model for Learning Analytics. *International Journal of Technology Enhanced Learning*. 2012. Vol. 4, No. 5/6. P. 318-331.
12. Papamitsiou Z., Economides A. A. Learning Analytics and Educational Data Mining in Practice: A Systematic Literature Review of Empirical Evidence. *Educational Technology & Society*. 2014. Vol. 17, No. 4. P. 49-64.
13. Sommerville I. *Software Engineering*. 10th ed. Boston : Pearson, 2016.
14. Pressman R. S., Maxim B. R. *Software Engineering: A Practitioner's Approach*. 9th ed. New York : McGraw-Hill, 2020.
15. Dennis A., Wixom B. H., Tegarden D. *Systems Analysis and Design: An Object-Oriented Approach with UML*. 6th ed. Hoboken : Wiley, 2020.
16. Pardo A., Siemens G. Ethical and Privacy Principles for Learning Analytics. *British Journal of Educational Technology*. 2014. Vol. 45, No. 3. P. 438-450. DOI: 10.1111/bjet.12152.
17. Топузов М. О. Проектування інформаційно-освітнього середовища навчальних закладів у сучасному суспільстві. *Український педагогічний журнал*. 2017. № 1.
18. Fowler M. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. 3rd ed. Boston : Addison-Wesley, 2004.
19. Мельник О. М. Узагальнена функціональна модель інформаційно-освітнього середовища закладу загальної середньої освіти. *Фізико-математична освіта*. 2020. Вип. 2(24). С. 94–99. DOI: 10.31110/2413-1571-2020-024-2-013.
20. Silberschatz A., Korth H. F., Sudarshan S. *Database System Concepts*. 7th ed. New York : McGraw-Hill Education, 2019.
21. SQLite. SQLite Home Page.
22. Python Software Foundation. sqlite3 — DB-API 2.0 interface for SQLite databases. *Python Documentation*.
23. Карташова Л. А., Юрженко В. В., Гуралюк А. Г., Липська Л. В., Гуменна Л. С., Зуєва А. Б., Шупік І. М., Ростока М. Л., Шевченко В. Л. *Інформаційно-*

освітнє середовище професійно-технічних навчальних закладів : посібник / за наук. ред. П. Г. Лузана. Київ : ІПТО НАПН, 2017. 124 с.

24. Овчарук О. В., Товканець О. С., Пінчук О. П., Іванюк І. В., Гриценчук О. О., Трикоз С. В. Організаційно-педагогічні умови використання інформаційно-цифрового середовища закладу загальної середньої освіти. *Інформаційні технології і засоби навчання*. 2023. Т. 95, № 3. С. 41-57. DOI: 10.33407/itlt.v95i3.5186.
25. Литвинова С. Г., Мельник О. М., Сухіх А. С. Електронний журнал як інструмент цифрової освітньої комунікації: результати всеукраїнського опитування. *Інформаційні технології і засоби навчання*. 2024. Т. 100, № 2. DOI: 10.33407/itlt.v100i2.5516.
26. Flask Documentation. Welcome to Flask. Version 3.1.x.
27. Flask Documentation. API. Version 3.1.x.
28. Карпенко А. С. Використання сервісів Google Apps у процесі інформатизації закладу вищої освіти. *Інформаційні технології і засоби навчання*. 2019. Т. 71, № 3. С. 183-195. DOI: 10.33407/itlt.v71i3.2350.
29. Myers G. J., Sandler C., Badgett T. *The Art of Software Testing*. 3rd ed. Hoboken : Wiley, 2015.
30. Ліпчевська І. Створення цифрового освітнього середовища як необхідна умова успішності профільної середньої освіти // *Інноваційні педагогічні технології в цифровій школі — 2024 : матеріали VI Міжнародної науково-практичної конференції молодих учених*. 2024. С. 115-118.

ДОДАТОК А

Код основного серверного модуля інформаційної системи електронного журналу викладача

```
from __future__ import annotations

import csv
import json
import math
import os
import sqlite3
from collections import Counter, defaultdict
from datetime import date, datetime
from functools import wraps
from pathlib import Path
from statistics import mean
from typing import Any

from flask import (
    Flask,
    abort,
    flash,
    g,
    redirect,
    render_template,
    request,
    send_from_directory,
    session,
    url_for,
)
from werkzeug.security import check_password_hash, generate_password_hash

BASE_DIR = Path(__file__).resolve().parent
DATABASE = BASE_DIR / "journal.db"
REPORTS_DIR = BASE_DIR / "reports"
REPORTS_DIR.mkdir(exist_ok=True)

app = Flask(__name__)
app.config["SECRET_KEY"] = "teacher-journal-demo-secret-key"

ATTENDANCE_LABELS = {
    "present": "Присутній",
    "absent": "Відсутній",
    "late": "Запізнення",
    "excused": "Поважна причина",
}
GRADE_BANDS = [
    (90, "90-100"),
    (75, "75-89"),
    (60, "60-74"),
    (0, "0-59"),
]


def get_db() -> sqlite3.Connection:
    if "db" not in g:
        g.db = sqlite3.connect(DATABASE)
        g.db.row_factory = sqlite3.Row
        g.db.execute("PRAGMA foreign_keys = ON")
    return g.db
```

```
@app.teardown_appcontext
def close_db(_: Any) -> None:
    db = g.pop("db", None)
    if db is not None:
        db.close()


def query_db(query: str, params: tuple[Any, ...] = (), one: bool = False) ->
Any:
    cur = get_db().execute(query, params)
    rows = cur.fetchall()
    cur.close()
    return (rows[0] if rows else None) if one else rows


def execute_db(query: str, params: tuple[Any, ...] = ()) -> int:
    db = get_db()
    cur = db.execute(query, params)
    db.commit()
    last_id = cur.lastrowid
    cur.close()
    return last_id


def current_user() -> sqlite3.Row | None:
    user_id = session.get("user_id")
    if not user_id:
        return None
    return query_db(
        """
        SELECT u.id, u.login, u.status, r.name AS role_name,
               t.id AS teacher_id, s.id AS student_id,
               COALESCE(t.surname || ' ' || t.name, s.surname || ' ' || s.name,
u.login) AS full_name
        FROM users u
        JOIN roles r ON r.id = u.role_id
        LEFT JOIN teachers t ON t.user_id = u.id
        LEFT JOIN students s ON s.user_id = u.id
        WHERE u.id = ?
        """,
        (user_id,),
        one=True,
    )


@app.context_processor
def inject_globals() -> dict[str, Any]:
    return {
        "current_user": current_user(),
        "attendance_labels": ATTENDANCE_LABELS,
        "now": datetime.now(),
    }


def login_required(view):
    @wraps(view)
    def wrapped(*args, **kwargs):
        if current_user() is None:
            return redirect(url_for("login"))
        return view(*args, **kwargs)

    return wrapped


def roles_required(*allowed_roles: str):
```

```
    def decorator(view):
        @wraps(view)
        def wrapped(*args, **kwargs):
            user = current_user()
            if user is None:
                return redirect(url_for("login"))
            if user["role_name"] not in allowed_roles:
                abort(403)
            return view(*args, **kwargs)

        return wrapped

    return decorator


def log_change(user_id: int | None, entity_type: str, entity_id: int | None,
action: str,
               old_value: dict[str, Any] | None = None, new_value: dict[str,
Any] | None = None) -> None:
    execute_db(
        """
        INSERT INTO audit_log (user_id, entity_type, entity_id, action,
old_value, new_value, changed_at)
        VALUES (?, ?, ?, ?, ?, ?, ?)
        """,
        (
            user_id,
            entity_type,
            entity_id,
            action,
            json.dumps(old_value, ensure_ascii=False) if old_value is not None
else None,
            json.dumps(new_value, ensure_ascii=False) if new_value is not None
else None,
            datetime.now().isoformat(timespec="seconds"),
        ),
    )


def init_db() -> None:
    db = sqlite3.connect(DATABASE)
    db.execute("PRAGMA foreign_keys = ON")
    schema = """
    CREATE TABLE IF NOT EXISTS roles (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        name TEXT UNIQUE NOT NULL,
        permissions TEXT
    );

    CREATE TABLE IF NOT EXISTS users (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        role_id INTEGER NOT NULL,
        login TEXT UNIQUE NOT NULL,
        password_hash TEXT NOT NULL,
        status TEXT NOT NULL DEFAULT 'active',
        FOREIGN KEY (role_id) REFERENCES roles(id)
    );

    CREATE TABLE IF NOT EXISTS teachers (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        user_id INTEGER UNIQUE,
        surname TEXT NOT NULL,
        name TEXT NOT NULL,
        email TEXT,
        FOREIGN KEY (user_id) REFERENCES users(id)
```

```
);

CREATE TABLE IF NOT EXISTS groups (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    group_name TEXT UNIQUE NOT NULL,
    course INTEGER NOT NULL,
    specialty TEXT NOT NULL
);

CREATE TABLE IF NOT EXISTS students (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER UNIQUE,
    group_id INTEGER NOT NULL,
    surname TEXT NOT NULL,
    name TEXT NOT NULL,
    student_code TEXT UNIQUE NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (group_id) REFERENCES groups(id)
);

CREATE TABLE IF NOT EXISTS disciplines (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    title TEXT NOT NULL,
    semester INTEGER NOT NULL,
    control_form TEXT NOT NULL
);

CREATE TABLE IF NOT EXISTS journals (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    teacher_id INTEGER NOT NULL,
    group_id INTEGER NOT NULL,
    discipline_id INTEGER NOT NULL,
    academic_year TEXT NOT NULL,
    semester INTEGER NOT NULL,
    status TEXT NOT NULL DEFAULT 'active',
    UNIQUE(teacher_id, group_id, discipline_id, academic_year, semester),
    FOREIGN KEY (teacher_id) REFERENCES teachers(id),
    FOREIGN KEY (group_id) REFERENCES groups(id),
    FOREIGN KEY (discipline_id) REFERENCES disciplines(id)
);

CREATE TABLE IF NOT EXISTS lessons (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    journal_id INTEGER NOT NULL,
    lesson_date TEXT NOT NULL,
    topic TEXT NOT NULL,
    lesson_type TEXT NOT NULL,
    notes TEXT,
    FOREIGN KEY (journal_id) REFERENCES journals(id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS grades (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    lesson_id INTEGER NOT NULL,
    student_id INTEGER NOT NULL,
    grade_value REAL,
    control_type TEXT NOT NULL,
    comment TEXT,
    entered_at TEXT NOT NULL,
    UNIQUE(lesson_id, student_id),
    FOREIGN KEY (lesson_id) REFERENCES lessons(id) ON DELETE CASCADE,
    FOREIGN KEY (student_id) REFERENCES students(id)
);

CREATE TABLE IF NOT EXISTS attendance (
```

```
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        lesson_id INTEGER NOT NULL,
        student_id INTEGER NOT NULL,
        presence_status TEXT NOT NULL,
        absence_reason TEXT,
        UNIQUE(lesson_id, student_id),
        FOREIGN KEY (lesson_id) REFERENCES lessons(id) ON DELETE CASCADE,
        FOREIGN KEY (student_id) REFERENCES students(id)
    );

    CREATE TABLE IF NOT EXISTS reports (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        user_id INTEGER NOT NULL,
        report_type TEXT NOT NULL,
        title TEXT NOT NULL,
        params_json TEXT,
        file_path TEXT NOT NULL,
        created_at TEXT NOT NULL,
        FOREIGN KEY (user_id) REFERENCES users(id)
    );

    CREATE TABLE IF NOT EXISTS audit_log (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        user_id INTEGER,
        entity_type TEXT NOT NULL,
        entity_id INTEGER,
        action TEXT NOT NULL,
        old_value TEXT,
        new_value TEXT,
        changed_at TEXT NOT NULL,
        FOREIGN KEY (user_id) REFERENCES users(id)
    );
    """
    db.executescript(schema)
    db.commit()

    roles = [
        ("admin", "керування довідковими даними, журнали дій, перегляд звітів"),
        ("teacher", "ведення журналу, аналітика, звітність"),
        ("student", "перегляд власних результатів"),
    ]
    db.executemany("INSERT OR IGNORE INTO roles (name, permissions) VALUES (?,
?)", roles)

    if db.execute("SELECT COUNT(*) FROM users").fetchone()[0] == 0:
        role_map = {row[1]: row[0] for row in db.execute("SELECT id, name FROM
roles").fetchall()}
        users = [
            (role_map["admin"], "admin", generate_password_hash("admin123"),
"active"),
            (role_map["teacher"], "teacher",
generate_password_hash("teacher123"), "active"),
            (role_map["student"], "student",
generate_password_hash("student123"), "active"),
        ]
        db.executemany(
            "INSERT INTO users (role_id, login, password_hash, status) VALUES
(?, ?, ?, ?)", users
        )
        admin_id = db.execute("SELECT id FROM users WHERE
login='admin'").fetchone()[0]
        teacher_user_id = db.execute("SELECT id FROM users WHERE
login='teacher'").fetchone()[0]
        student_user_id = db.execute("SELECT id FROM users WHERE
login='student'").fetchone()[0]
```

```
        db.execute(
            "INSERT INTO teachers (user_id, surname, name, email) VALUES (?, ?,
?, ?)",
            (teacher_user_id, "Петренко", "Олена", "teacher@example.com"),
        )

        groups = [
            ("КН-31", 3, "Комп'ютерні науки"),
            ("КН-32", 3, "Комп'ютерні науки"),
        ]
        db.executemany("INSERT INTO groups (group_name, course, specialty)
VALUES (?, ?, ?)", groups)
        group_ids = {row[1]: row[0] for row in db.execute("SELECT id, group_name
FROM groups")}

        students = [
            (student_user_id, group_ids["КН-31"], "Іваненко", "Андрій",
"ST001"),
            (None, group_ids["КН-31"], "Бондар", "Марія", "ST002"),
            (None, group_ids["КН-31"], "Коваль", "Дмитро", "ST003"),
            (None, group_ids["КН-31"], "Савчук", "Юлія", "ST004"),
            (None, group_ids["КН-31"], "Мельник", "Ірина", "ST005"),
            (None, group_ids["КН-32"], "Романюк", "Олег", "ST006"),
            (None, group_ids["КН-32"], "Ткаченко", "Наталія", "ST007"),
            (None, group_ids["КН-32"], "Шевчук", "Владислав", "ST008"),
        ]
        db.executemany(
            "INSERT INTO students (user_id, group_id, surname, name,
student_code) VALUES (?, ?, ?, ?, ?)",
            students,
        )

        disciplines = [
            ("Бази даних", 6, "іспит"),
            ("Веб-програмування", 6, "залік"),
        ]
        db.executemany(
            "INSERT INTO disciplines (title, semester, control_form) VALUES (?,
?, ?)", disciplines
        )
        discipline_ids = {row[1]: row[0] for row in db.execute("SELECT id, title
FROM disciplines")}
        teacher_id = db.execute("SELECT id FROM teachers WHERE user_id=?",
(teacher_user_id,)).fetchone()[0]

        journals = [
            (teacher_id, group_ids["КН-31"], discipline_ids["Бази даних"],
"2025/2026", 6, "active"),
            (teacher_id, group_ids["КН-32"], discipline_ids["Веб-
програмування"], "2025/2026", 6, "active"),
        ]
        db.executemany(
            """
            INSERT INTO journals (teacher_id, group_id, discipline_id,
academic_year, semester, status)
            VALUES (?, ?, ?, ?, ?, ?)
            """,
            journals,
        )

        # Seed lessons, grades and attendance
        journal_rows = db.execute(
            "SELECT id, group_id, discipline_id FROM journals ORDER BY id"
        ).fetchall()
```

```
        students_by_group = defaultdict(list)
        for row in db.execute("SELECT id, group_id FROM students").fetchall():
            students_by_group[row[1]].append(row[0])

        lesson_templates = [
            ("2026-02-10", "Вступ до дисципліни", "лекція"),
            ("2026-02-17", "Нормалізація даних", "практичне"),
            ("2026-02-24", "SQL-запити", "лабораторне"),
            ("2026-03-03", "Агрегатні функції", "лабораторне"),
            ("2026-03-10", "Проміжний контроль", "контрольна"),
        ]
        base_grades = [88, 76, 92, 67, 81, 73, 95, 78]
        base_presence = ["present", "present", "present", "absent", "late",
"present", "excused", "present"]

        for j_index, journal in enumerate(journal_rows):
            group_student_ids = students_by_group[journal[1]]
            for l_index, (lesson_date, topic, lesson_type) in
enumerate(lesson_templates, start=1):
                lesson_id = db.execute(
                    "INSERT INTO lessons (journal_id, lesson_date, topic,
lesson_type, notes) VALUES (?, ?, ?, ?, ?)",
                    (journal[0], lesson_date, topic, lesson_type, f"Примітки до
заняття {l_index}"),
                ).lastrowid
                for idx, student_id in enumerate(group_student_ids):
                    grade = max(50, min(100, base_grades[(idx + l_index +
j_index) % len(base_grades)] + (j_index * 2) - (l_index % 3)))
                    presence = base_presence[(idx + l_index) %
len(base_presence)]
                    control_type = "поточний" if l_index < 5 else "модульний"
                    db.execute(
                        "INSERT INTO grades (lesson_id, student_id, grade_value,
control_type, comment, entered_at) VALUES (?, ?, ?, ?, ?, ?)",
                        (lesson_id, student_id, grade, control_type, "",
datetime.now().isoformat(timespec="seconds")),
                    )
                    reason = "медична довідка" if presence == "excused" else (""
if presence != "absent" else "без поважної причини")
                    db.execute(
                        "INSERT INTO attendance (lesson_id, student_id,
presence_status, absence_reason) VALUES (?, ?, ?, ?)",
                        (lesson_id, student_id, presence, reason),
                    )

        db.execute(
            """
            INSERT INTO audit_log (user_id, entity_type, entity_id, action,
old_value, new_value, changed_at)
            VALUES (?, ?, ?, ?, ?, ?, ?)
            """,
            (
                admin_id,
                "system",
                None,
                "seed",
                None,
                json.dumps({"status": "initial demo data created"},
ensure_ascii=False),
                datetime.now().isoformat(timespec="seconds"),
            ),
        )

    db.commit()
    db.close()
```

```
def get_teacher_journals(teacher_id: int) -> list[sqlite3.Row]:
    return query_db(
        """
        SELECT j.id, j.academic_year, j.semester, j.status,
               g.group_name, d.title AS discipline_title,
               d.control_form, d.semester AS discipline_semester
        FROM journals j
        JOIN groups g ON g.id = j.group_id
        JOIN disciplines d ON d.id = j.discipline_id
        WHERE j.teacher_id = ?
        ORDER BY d.title, g.group_name
        """,
        (teacher_id,),
    )


def get_journal_for_user(journal_id: int, user: sqlite3.Row) -> sqlite3.Row |
None:
    if user["role_name"] == "teacher":
        return query_db(
            """
            SELECT j.*, g.group_name, g.specialty, d.title AS discipline_title,
d.control_form,
                   t.surname || ' ' || t.name AS teacher_name
            FROM journals j
            JOIN groups g ON g.id = j.group_id
            JOIN disciplines d ON d.id = j.discipline_id
            JOIN teachers t ON t.id = j.teacher_id
            WHERE j.id = ? AND j.teacher_id = ?
            """,
            (journal_id, user["teacher_id"]),
            one=True,
        )
    if user["role_name"] == "admin":
        return query_db(
            """
            SELECT j.*, g.group_name, g.specialty, d.title AS discipline_title,
d.control_form,
                   t.surname || ' ' || t.name AS teacher_name
            FROM journals j
            JOIN groups g ON g.id = j.group_id
            JOIN disciplines d ON d.id = j.discipline_id
            JOIN teachers t ON t.id = j.teacher_id
            WHERE j.id = ?
            """,
            (journal_id,),
            one=True,
        )
    return None


def get_students_in_group(group_id: int) -> list[sqlite3.Row]:
    return query_db(
        "SELECT * FROM students WHERE group_id = ? ORDER BY surname, name",
        (group_id,),
    )


def get_lessons(journal_id: int) -> list[sqlite3.Row]:
    return query_db(
        "SELECT * FROM lessons WHERE journal_id = ? ORDER BY lesson_date, id",
        (journal_id,),
    )
```

```
def validate_grade(value: str) -> float | None:
    value = value.strip()
    if not value:
        return None
    grade = float(value)
    if grade < 0 or grade > 100:
        raise ValueError("Оцінка повинна бути в межах від 0 до 100")
    return grade


def pearson_correlation(xs: list[float], ys: list[float]) -> float | None:
    if len(xs) != len(ys) or len(xs) < 2:
        return None
    mx = mean(xs)
    my = mean(ys)
    num = sum((x - mx) * (y - my) for x, y in zip(xs, ys))
    den_x = math.sqrt(sum((x - mx) ** 2 for x in xs))
    den_y = math.sqrt(sum((y - my) ** 2 for y in ys))
    if den_x == 0 or den_y == 0:
        return None
    return round(num / (den_x * den_y), 3)


def analytics_for_journal(journal_id: int, date_from: str | None = None,
date_to: str | None = None) -> dict[str, Any]:
    filters = ["l.journal_id = ?"]
    params: list[Any] = [journal_id]
    if date_from:
        filters.append("l.lesson_date >= ?")
        params.append(date_from)
    if date_to:
        filters.append("l.lesson_date <= ?")
        params.append(date_to)

    where_clause = " AND ".join(filters)

    rows = query_db(
        f"""
        SELECT s.id AS student_id, s.surname, s.name, s.student_code,
               l.id AS lesson_id, l.lesson_date, l.topic,
               g.grade_value, g.control_type,
               a.presence_status
        FROM lessons l
        JOIN journals j ON j.id = l.journal_id
        JOIN students s ON s.group_id = j.group_id
        LEFT JOIN grades g ON g.lesson_id = l.id AND g.student_id = s.id
        LEFT JOIN attendance a ON a.lesson_id = l.id AND a.student_id = s.id
        WHERE {where_clause}
        ORDER BY l.lesson_date, s.surname, s.name
        """,
        tuple(params),
    )

    students: dict[int, dict[str, Any]] = {}
    lesson_averages: dict[str, list[float]] = defaultdict(list)
    distribution_counter: Counter[str] = Counter()
    control_type_totals: defaultdict[str, list[float]] = defaultdict(list)

    for row in rows:
        sid = row["student_id"]
        students.setdefault(
            sid,
            {
```

```
                "student_id": sid,
                "name": f"{row['surname']} {row['name']}",
                "student_code": row["student_code"],
                "grades": [],
                "attendance": [],
            },
        )
        if row["grade_value"] is not None:
            grade_val = float(row["grade_value"])
            students[sid]["grades"].append(grade_val)
            lesson_averages[row["lesson_date"]].append(grade_val)
            control_type_totals[row["control_type"]].append(grade_val)
            for band_min, label in GRADE_BANDS:
                if grade_val >= band_min:
                    distribution_counter[label] += 1
                    break
        if row["presence_status"]:
            students[sid]["attendance"].append(row["presence_status"])

    student_metrics: list[dict[str, Any]] = []
    att_for_corr: list[float] = []
    grade_for_corr: list[float] = []
    for item in students.values():
        grades = item["grades"]
        attendance = item["attendance"]
        avg_grade = round(mean(grades), 2) if grades else 0.0
        present_count = sum(1 for value in attendance if value in {"present",
"late", "excused"})
        attendance_pct = round((present_count / len(attendance)) * 100, 2) if
attendance else 0.0
        student_metrics.append(
            {
                "student_id": item["student_id"],
                "name": item["name"],
                "student_code": item["student_code"],
                "average_grade": avg_grade,
                "attendance_pct": attendance_pct,
                "grade_count": len(grades),
                "attendance_count": len(attendance),
            }
        )
        if grades and attendance:
            att_for_corr.append(attendance_pct)
            grade_for_corr.append(avg_grade)

    student_metrics.sort(key=lambda x: (-x["average_grade"], -
x["attendance_pct"], x["name"]))
    for idx, metric in enumerate(student_metrics, start=1):
        metric["rank"] = idx

    group_average = round(mean([m["average_grade"] for m in student_metrics]),
2) if student_metrics else 0.0
    group_attendance = round(mean([m["attendance_pct"] for m in
student_metrics]), 2) if student_metrics else 0.0
    dynamics = [
        {"date": dt, "average": round(mean(values), 2)}
        for dt, values in sorted(lesson_averages.items())
    ]
    control_type_avg = {key: round(mean(values), 2) for key, values in
control_type_totals.items() if values}
    correlation = pearson_correlation(att_for_corr, grade_for_corr)

    return {
        "student_metrics": student_metrics,
        "group_average": group_average,
```

```
        "group_attendance": group_attendance,
        "dynamics": dynamics,
        "distribution": dict(distribution_counter),
        "control_type_avg": control_type_avg,
        "correlation": correlation,
        "record_count": len(rows),
    }


def comparative_journal_analytics(teacher_id: int) -> list[dict[str, Any]]:
    journals = get_teacher_journals(teacher_id)
    data: list[dict[str, Any]] = []
    for journal in journals:
        metrics = analytics_for_journal(journal["id"])
        data.append(
            {
                "journal_id": journal["id"],
                "discipline": journal["discipline_title"],
                "group_name": journal["group_name"],
                "group_average": metrics["group_average"],
                "group_attendance": metrics["group_attendance"],
                "student_count": len(metrics["student_metrics"]),
            }
        )
    return data


def save_csv_report(title: str, headers: list[str], rows: list[list[Any]],
report_type: str, user_id: int,
                    params: dict[str, Any]) -> str:
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    filename = f"{report_type}_{timestamp}.csv"
    file_path = REPORTS_DIR / filename
    with file_path.open("w", newline="", encoding="utf-8-sig") as csvfile:
        writer = csv.writer(csvfile, delimiter=";")
        writer.writerow([title])
        writer.writerow(headers)
        writer.writerows(rows)
    execute_db(
        "INSERT INTO reports (user_id, report_type, title, params_json,
file_path, created_at) VALUES (?, ?, ?, ?, ?, ?)",
        (
            user_id,
            report_type,
            title,
            json.dumps(params, ensure_ascii=False),
            filename,
            datetime.now().isoformat(timespec="seconds"),
        ),
    )
    return filename


def generate_report(report_type: str, journal_id: int, user: sqlite3.Row) ->
str:
    journal = get_journal_for_user(journal_id, user)
    if journal is None:
        abort(403)
    title_base = f"{journal['discipline_title']} - {journal['group_name']}"

    if report_type == "current":
        rows = query_db(
            """
            SELECT l.lesson_date, l.topic, s.surname || ' ' || s.name AS
student_name,
```

```
                    g.grade_value, g.control_type, a.presence_status,
a.absence_reason
                FROM lessons l
                JOIN journals j ON j.id = l.journal_id
                JOIN students s ON s.group_id = j.group_id
                LEFT JOIN grades g ON g.lesson_id = l.id AND g.student_id = s.id
                LEFT JOIN attendance a ON a.lesson_id = l.id AND a.student_id = s.id
                WHERE j.id = ?
                ORDER BY l.lesson_date, student_name
                """,
                (journal_id,),
            )
            csv_rows = [
                [
                    row["lesson_date"], row["topic"], row["student_name"],
                    row["grade_value"] or "", row["control_type"] or "",
                    ATTENDANCE_LABELS.get(row["presence_status"], ""),
row["absence_reason"] or "",
                ]
                for row in rows
            ]
            return save_csv_report(
                f"Поточний звіт: {title_base}",
                ["Дата", "Тема", "Студент", "Оцінка", "Тип контролю",
"Відвідування", "Причина"],
                csv_rows,
                report_type,
                user["id"],
                {"journal_id": journal_id},
            )

        analytics = analytics_for_journal(journal_id)
        if report_type == "summary":
            csv_rows = [
                [
                    item["rank"], item["name"], item["student_code"],
item["average_grade"], item["attendance_pct"]
                ]
                for item in analytics["student_metrics"]
            ]
            return save_csv_report(
                f"Підсумковий звіт: {title_base}",
                ["Ранг", "Студент", "Код", "Середній бал", "Відвідування, %"],
                csv_rows,
                report_type,
                user["id"],
                {"journal_id": journal_id},
            )

        if report_type == "ranking":
            csv_rows = [[item["rank"], item["name"], item["average_grade"]] for item
in analytics["student_metrics"]]
            return save_csv_report(
                f"Рейтинговий звіт: {title_base}",
                ["Ранг", "Студент", "Середній бал"],
                csv_rows,
                report_type,
                user["id"],
                {"journal_id": journal_id},
            )

        if report_type == "comparative":
            compare_rows = comparative_journal_analytics(user["teacher_id"]) if
user["role_name"] == "teacher" else []
            csv_rows = [
```

```
            [row["discipline"], row["group_name"], row["group_average"],
row["group_attendance"], row["student_count"]]
            for row in compare_rows
        ]
        return save_csv_report(
            "Порівняльний звіт за журналами викладача",
            ["Дисципліна", "Група", "Середній бал", "Відвідування, %",
"Кількість студентів"],
            csv_rows,
            report_type,
            user["id"],
            {"journal_id": journal_id, "teacher_id": user["teacher_id"]},
        )

    raise ValueError("Невідомий тип звіту")


@app.route("/")
def index():
    user = current_user()
    if user:
        return redirect(url_for("dashboard"))
    return redirect(url_for("login"))


@app.route("/login", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        login_value = request.form.get("login", "").strip()
        password = request.form.get("password", "")
        user = query_db(
            "SELECT u.*, r.name AS role_name FROM users u JOIN roles r ON r.id =
u.role_id WHERE login = ?",
            (login_value,),
            one=True,
        )
        if user and check_password_hash(user["password_hash"], password):
            session.clear()
            session["user_id"] = user["id"]
            flash("Вхід виконано успішно.", "success")
            return redirect(url_for("dashboard"))
        flash("Невірний логін або пароль.", "danger")
    return render_template("login.html")


@app.route("/logout")
def logout():
    session.clear()
    flash("Сеанс завершено.", "info")
    return redirect(url_for("login"))


@app.route("/dashboard")
@login_required
def dashboard():
    user = current_user()
    assert user is not None
    if user["role_name"] == "teacher":
        journals = get_teacher_journals(user["teacher_id"])
        compare_rows = comparative_journal_analytics(user["teacher_id"])
        return render_template("dashboard_teacher.html", journals=journals,
compare_rows=compare_rows)
    if user["role_name"] == "admin":
        counts = {
            "users": query_db("SELECT COUNT(*) AS c FROM users", one=True)["c"],
```

```
            "students": query_db("SELECT COUNT(*) AS c FROM students",
one=True)["c"],
            "journals": query_db("SELECT COUNT(*) AS c FROM journals",
one=True)["c"],
            "reports": query_db("SELECT COUNT(*) AS c FROM reports",
one=True)["c"],
        }
        recent_logs = query_db(
            """
            SELECT a.changed_at, a.entity_type, a.action, u.login
            FROM audit_log a LEFT JOIN users u ON u.id = a.user_id
            ORDER BY a.id DESC LIMIT 10
            """
        )
        return render_template("dashboard_admin.html", counts=counts,
recent_logs=recent_logs)
    return redirect(url_for("student_view"))


@app.route("/journal/<int:journal_id>", methods=["GET", "POST"])
@login_required
@roles_required("teacher", "admin")
def journal_view(journal_id: int):
    user = current_user()
    assert user is not None
    journal = get_journal_for_user(journal_id, user)
    if journal is None:
        abort(403)

    if request.method == "POST" and request.form.get("form_name") ==
"add_lesson":
        lesson_date = request.form.get("lesson_date", "").strip()
        topic = request.form.get("topic", "").strip()
        lesson_type = request.form.get("lesson_type", "").strip() or "заняття"
        notes = request.form.get("notes", "").strip()
        if not lesson_date or not topic:
            flash("Дата і тема заняття є обов'язковими.", "danger")
        else:
            lesson_id = execute_db(
                "INSERT INTO lessons (journal_id, lesson_date, topic,
lesson_type, notes) VALUES (?, ?, ?, ?, ?)",
                (journal_id, lesson_date, topic, lesson_type, notes),
            )
            log_change(user["id"], "lesson", lesson_id, "create",
new_value={"topic": topic, "lesson_date": lesson_date})
            flash("Заняття додано.", "success")
            return redirect(url_for("journal_view", journal_id=journal_id))

    lesson_id = request.args.get("lesson_id", type=int)
    lessons = get_lessons(journal_id)
    selected_lesson = None
    if lessons:
        if lesson_id is None:
            lesson_id = lessons[-1]["id"]
        selected_lesson = next((lesson for lesson in lessons if lesson["id"] ==
lesson_id), None)

    if request.method == "POST" and request.form.get("form_name") ==
"save_records":
        selected_lesson_id = int(request.form.get("lesson_id", "0"))
        students = get_students_in_group(journal["group_id"])
        try:
            for student in students:
                sid = student["id"]
                grade_key = f"grade_{sid}"
```

```
                attendance_key = f"attendance_{sid}"
                reason_key = f"reason_{sid}"
                control_key = f"control_{sid}"
                comment_key = f"comment_{sid}"

                grade = validate_grade(request.form.get(grade_key, ""))
                attendance_status = request.form.get(attendance_key, "present")
                absence_reason = request.form.get(reason_key, "").strip()
                control_type = request.form.get(control_key, "поточний").strip()
or "поточний"
                comment = request.form.get(comment_key, "").strip()

                existing_grade = query_db(
                    "SELECT * FROM grades WHERE lesson_id = ? AND student_id =
?",
                    (selected_lesson_id, sid),
                    one=True,
                )
                if existing_grade:
                    old_grade = dict(existing_grade)
                    execute_db(
                        "UPDATE grades SET grade_value = ?, control_type = ?,
comment = ?, entered_at = ? WHERE id = ?",
                        (grade, control_type, comment,
datetime.now().isoformat(timespec="seconds"), existing_grade["id"]),
                    )
                    log_change(user["id"], "grade", existing_grade["id"],
"update", old_value=old_grade,
                               new_value={"grade_value": grade, "control_type":
control_type, "comment": comment})
                else:
                    grade_id = execute_db(
                        "INSERT INTO grades (lesson_id, student_id, grade_value,
control_type, comment, entered_at) VALUES (?, ?, ?, ?, ?, ?)",
                        (selected_lesson_id, sid, grade, control_type, comment,
datetime.now().isoformat(timespec="seconds")),
                    )
                    log_change(user["id"], "grade", grade_id, "create",
new_value={"grade_value": grade, "control_type": control_type})

                existing_attendance = query_db(
                    "SELECT * FROM attendance WHERE lesson_id = ? AND student_id
= ?",
                    (selected_lesson_id, sid),
                    one=True,
                )
                if existing_attendance:
                    old_attendance = dict(existing_attendance)
                    execute_db(
                        "UPDATE attendance SET presence_status = ?,
absence_reason = ? WHERE id = ?",
                        (attendance_status, absence_reason,
existing_attendance["id"]),
                    )
                    log_change(user["id"], "attendance",
existing_attendance["id"], "update", old_value=old_attendance,
                               new_value={"presence_status": attendance_status,
"absence_reason": absence_reason})
                else:
                    attendance_id = execute_db(
                        "INSERT INTO attendance (lesson_id, student_id,
presence_status, absence_reason) VALUES (?, ?, ?, ?)",
                        (selected_lesson_id, sid, attendance_status,
absence_reason),
                    )
```

```
                    log_change(user["id"], "attendance", attendance_id,
"create", new_value={"presence_status": attendance_status})
                flash("Записи збережено.", "success")
            except ValueError as exc:
                flash(str(exc), "danger")
            return redirect(url_for("journal_view", journal_id=journal_id,
lesson_id=selected_lesson_id))

    students = get_students_in_group(journal["group_id"])
    lesson_records: list[dict[str, Any]] = []
    if selected_lesson:
        filters = {
            "search": request.args.get("search", "").strip().lower(),
            "control_type": request.args.get("control_type",
"").strip().lower(),
        }
        for student in students:
            grade = query_db(
                "SELECT * FROM grades WHERE lesson_id = ? AND student_id = ?",
                (selected_lesson["id"], student["id"]),
                one=True,
            )
            attendance = query_db(
                "SELECT * FROM attendance WHERE lesson_id = ? AND student_id =
?",
                (selected_lesson["id"], student["id"]),
                one=True,
            )
            record = {
                "student": student,
                "grade": grade,
                "attendance": attendance,
            }
            full_name = f"{student['surname']} {student['name']}".lower()
            control_val = (grade["control_type"] if grade else "").lower()
            if filters["search"] and filters["search"] not in full_name:
                continue
            if filters["control_type"] and filters["control_type"] not in
control_val:
                continue
            lesson_records.append(record)

    return render_template(
        "journal.html",
        journal=journal,
        lessons=lessons,
        selected_lesson=selected_lesson,
        lesson_records=lesson_records,
    )


@app.route("/analytics/<int:journal_id>")
@login_required
@roles_required("teacher", "admin")
def analytics_view(journal_id: int):
    user = current_user()
    assert user is not None
    journal = get_journal_for_user(journal_id, user)
    if journal is None:
        abort(403)
    date_from = request.args.get("date_from") or None
    date_to = request.args.get("date_to") or None
    analytics = analytics_for_journal(journal_id, date_from=date_from,
date_to=date_to)
    return render_template(
```

```
        "analytics.html",
        journal=journal,
        analytics=analytics,
        date_from=date_from or "",
        date_to=date_to or "",
        comparison_rows=comparative_journal_analytics(user["teacher_id"]) if
user["role_name"] == "teacher" else [],
    )


@app.route("/reports/<int:journal_id>", methods=["GET", "POST"])
@login_required
@roles_required("teacher", "admin")
def reports_view(journal_id: int):
    user = current_user()
    assert user is not None
    journal = get_journal_for_user(journal_id, user)
    if journal is None:
        abort(403)

    if request.method == "POST":
        report_type = request.form.get("report_type", "")
        try:
            filename = generate_report(report_type, journal_id, user)
            flash(f"Звіт сформовано: {filename}", "success")
        except Exception as exc:
            flash(f"Не вдалося сформувати звіт: {exc}", "danger")
        return redirect(url_for("reports_view", journal_id=journal_id))

    reports = query_db(
        "SELECT * FROM reports WHERE user_id = ? ORDER BY id DESC",
        (user["id"],),
    )
    return render_template("reports.html", journal=journal, reports=reports)


@app.route("/reports/download/<path:filename>")
@login_required
def download_report(filename: str):
    return send_from_directory(REPORTS_DIR, filename, as_attachment=True)


@app.route("/admin/reference", methods=["GET", "POST"])
@login_required
@roles_required("admin")
def admin_reference():
    user = current_user()
    assert user is not None
    action = request.form.get("action", "") if request.method == "POST" else ""
    try:
        if request.method == "POST" and action == "add_group":
            group_id = execute_db(
                "INSERT INTO groups (group_name, course, specialty) VALUES (?,
?, ?)",
                (
                    request.form.get("group_name", "").strip(),
                    int(request.form.get("course", "1")),
                    request.form.get("specialty", "").strip(),
                ),
            )
            log_change(user["id"], "group", group_id, "create",
new_value={"group_name": request.form.get("group_name")})
            flash("Групу додано.", "success")
        elif request.method == "POST" and action == "add_discipline":
            discipline_id = execute_db(
```

```
                "INSERT INTO disciplines (title, semester, control_form) VALUES
(?, ?, ?)",
                (
                    request.form.get("title", "").strip(),
                    int(request.form.get("semester", "1")),
                    request.form.get("control_form", "").strip(),
                ),
            )
            log_change(user["id"], "discipline", discipline_id, "create",
new_value={"title": request.form.get("title")})
            flash("Дисципліну додано.", "success")
        elif request.method == "POST" and action == "add_student":
            student_id = execute_db(
                "INSERT INTO students (user_id, group_id, surname, name,
student_code) VALUES (?, ?, ?, ?, ?)",
                (
                    None,
                    int(request.form.get("group_id", "0")),
                    request.form.get("surname", "").strip(),
                    request.form.get("name", "").strip(),
                    request.form.get("student_code", "").strip(),
                ),
            )
            log_change(user["id"], "student", student_id, "create",
new_value={"student_code": request.form.get("student_code")})
            flash("Студента додано.", "success")
        elif request.method == "POST" and action == "add_journal":
            journal_id = execute_db(
                """
                INSERT INTO journals (teacher_id, group_id, discipline_id,
academic_year, semester, status)
                VALUES (?, ?, ?, ?, ?, ?)
                """,
                (
                    int(request.form.get("teacher_id", "0")),
                    int(request.form.get("group_id", "0")),
                    int(request.form.get("discipline_id", "0")),
                    request.form.get("academic_year", "").strip(),
                    int(request.form.get("semester", "1")),
                    request.form.get("status", "active").strip(),
                ),
            )
            log_change(user["id"], "journal", journal_id, "create",
new_value={"status": request.form.get("status")})
            flash("Журнал створено.", "success")
    except sqlite3.IntegrityError as exc:
        flash(f"Помилка цілісності даних: {exc}", "danger")
    except ValueError as exc:
        flash(str(exc), "danger")

    groups = query_db("SELECT * FROM groups ORDER BY group_name")
    disciplines = query_db("SELECT * FROM disciplines ORDER BY title")
    students = query_db(
        """
        SELECT s.*, g.group_name FROM students s JOIN groups g ON g.id =
s.group_id
        ORDER BY g.group_name, s.surname, s.name
        """
    )
    teachers = query_db("SELECT * FROM teachers ORDER BY surname, name")
    journals = query_db(
        """
        SELECT j.id, j.academic_year, j.semester, j.status,
               g.group_name, d.title AS discipline_title,
               t.surname || ' ' || t.name AS teacher_name
```

```
        FROM journals j
        JOIN groups g ON g.id = j.group_id
        JOIN disciplines d ON d.id = j.discipline_id
        JOIN teachers t ON t.id = j.teacher_id
        ORDER BY j.id DESC
        """
    )
    audit_logs = query_db(
        "SELECT a.*, u.login FROM audit_log a LEFT JOIN users u ON u.id =
a.user_id ORDER BY a.id DESC LIMIT 25"
    )
    return render_template(
        "admin_reference.html",
        groups=groups,
        disciplines=disciplines,
        students=students,
        teachers=teachers,
        journals=journals,
        audit_logs=audit_logs,
    )


@app.route("/student")
@login_required
@roles_required("student")
def student_view():
    user = current_user()
    assert user is not None
    student = query_db(
        """
        SELECT s.*, g.group_name FROM students s
        JOIN groups g ON g.id = s.group_id
        WHERE s.id = ?
        """,
        (user["student_id"],),
        one=True,
    )
    rows = query_db(
        """
        SELECT d.title AS discipline_title, l.lesson_date, l.topic,
               g.grade_value, g.control_type,
               a.presence_status
        FROM students s
        JOIN groups gr ON gr.id = s.group_id
        JOIN journals j ON j.group_id = gr.id
        JOIN disciplines d ON d.id = j.discipline_id
        JOIN lessons l ON l.journal_id = j.id
        LEFT JOIN grades g ON g.lesson_id = l.id AND g.student_id = s.id
        LEFT JOIN attendance a ON a.lesson_id = l.id AND a.student_id = s.id
        WHERE s.id = ?
        ORDER BY l.lesson_date DESC
        """,
        (student["id"],),
    )
    grouped = defaultdict(list)
    grade_values = []
    present_marks = []
    dynamics = defaultdict(list)
    for row in rows:
        grouped[row["discipline_title"]].append(row)
        if row["grade_value"] is not None:
            grade_values.append(float(row["grade_value"]))
            dynamics[row["lesson_date"]].append(float(row["grade_value"]))
        if row["presence_status"]:
```

```
            present_marks.append(1 if row["presence_status"] in {"present",
"late", "excused"} else 0)
    metrics = {
        "average_grade": round(mean(grade_values), 2) if grade_values else 0.0,
        "attendance_pct": round(mean(present_marks) * 100, 2) if present_marks
else 0.0,
        "dynamics": [{"date": d, "average": round(mean(v), 2)} for d, v in
sorted(dynamics.items())],
    }
    return render_template("student_view.html", student=student,
grouped=grouped, metrics=metrics)


@app.route("/health")
def health():
    return {"status": "ok", "database": str(DATABASE.exists())}


if __name__ == "__main__":
    init_db()
    app.run(debug=True, host="127.0.0.1", port=5000)
```