НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри інформаційних систем і технологій**
_____Ігор БОЛБОТ	**_____Михайло Швиденко**
(підпис)	(підпис)
«___» _____2026 р.	«___» _____2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інформаційна система аналізу відвідування занять студентами університету »

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми

к.т.н., доцент **_____Роман РУДЕНСЬКИЙ**

(підпис)

Керівник бакалаврської кваліфікаційної роботи _____ **Максим Мокрієв**

(підпис)

ст.викладач

Виконав **_____Павло Шинкаренко**

(підпис)

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
інформаційних систем і технологій
(назва кафедри)

___к.е.н.,доцент____. Швиденко М.З.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"_06" _січня___ 2026 р. **(дата наказу)**

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студенту

Шинкаренкові Павлу Олександровичу
Спеціальність 122 – «Комп'ютерні науки»

ОП «Комп'ютерні науки»

1. Тема бакалаврської кваліфікаційної роботи «Інформаційна система аналізу відвідування занять студентами» затверджена наказом ректора НУБіП України від 06.01.2026 №46 «С»
2. Термін подання завершеної роботи на кафедру_____2026.05.25_____
(рік, місяць, число)

3. Вихідні дані до роботи: механізм функціонування інформаційної системи збору, збереження та аналізу даних про відвідування студентами занять, включаючи реєстрацію користувачів, фіксацію відвідуваності, формування звітів і взаємодію через веб-платформу.
4. Перелік питань, що розглядаються:
Аналіз взаємодії студентів, викладачів та адміністрації із системою.
-Проєктування інформаційного забезпечення системи моніторингу відвідуваності.
-Розробка програмного забезпечення системи аналізу відвідування занять.
-Впровадження та експлуатація системи в університеті.
Дата видачі завдання "_06"_січня_____ 2026 р. (дата наказу)

Керівник бакалаврської кваліфікаційної роботи _____/ Мокрієв М.В./
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання:_____/ Шинкаренко П.О. /
(підпис) (прізвище та ініціал

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ.....5

ВСТУП.....6

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....5

1.1 Опис предметної області.....5

1.2 Аналіз існуючих рішень для обліку відвідування та занять.....7

1.3 Моделювання предметної області.....11

1.4 Аналіз вимог до інформаційної системи.....14

1.5 Постановка завдання.....16

1.6 Висновки до першого розділу.....17

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....18

2.1 Логічна модель даних у вигляді ER-діаграми.....18

2.2 Фізична модель даних і структура бази UniversityAttendanceDB.....22

2.3 Діаграма компонентів і пакетна структура.....27

2.4 Алгоритмічне забезпечення звітності та аналітики.....30

2.5 Висновки до другого розділу.....36

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....39

3.1 Обґрунтування вибору технологій.....39

3.2 Представлення архітектури системи.....42

3.3 Висновки до третього розділу.....46

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ.....48

4.1 План тестування програмних модулів.....48

4.2 Тестування інтерфейсних модулів.....53

4.3 Розгортання системи, склад інсталяційного пакету.....61

4.4 Висновки до четвертого розділу.....65
ВИСНОВКИ.....67
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....70
ДОДАТОК А.....72
ДОДАТОК Б.....80
ДОДАТОК В.....84

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

- АРМ - автоматизоване робоче місце;
- БД - база даних;
- ВНЗ - заклад вищої освіти;
- ІС - інформаційна система;
- ПЗ - програмне забезпечення;
- СУБД - система керування базами даних;
- ER - Entity-Relationship, модель сутність-зв’язок;
- SQL - Structured Query Language, мова структурованих запитів;
- UML - Unified Modeling Language, уніфікована мова моделювання;
- HTML - HyperText Markup Language;
- CSS - Cascading Style Sheets;
- JS - JavaScript;
- HTTPS - захищений протокол передавання гіпертексту;
- TCP/IP - набір мережевих протоколів для обміну даними;
- UniversityAttendanceDB - база даних інформаційної системи аналізу відвідування;
- Attendance Web App - вебзастосунок для обліку й аналізу відвідування студентів.

ВСТУП

Цифровізація освітнього процесу є одним із важливих напрямів розвитку сучасного університету. Навчальні підрозділи щоденно працюють з великим обсягом організаційної інформації: розкладами занять, академічними групами,

дисциплінами, списками студентів, даними про присутність на заняттях і причинами пропусків. Якщо ці відомості зберігаються у різних журналах, файлах або неузгоджених таблицях, ускладнюється контроль навчальної дисципліни, підготовка звітності та прийняття управлінських рішень.

Особливе значення має облік відвідування занять. Для викладача це інструмент поточного контролю групи, для деканату - джерело інформації про академічну активність студентів, для адміністратора - дані для супроводу навчального процесу, а для студента - можливість бачити власну статистику пропусків. Тому система обліку повинна бути не лише зручною для щоденного використання, а й достатньо надійною, захищеною та придатною для формування аналітичних звітів.

Паперові журнали та розрізнені електронні таблиці не забезпечують належної оперативності. Вони не підтримують автоматичного підрахунку відсотка відвідуваності, не контролюють дублювання записів, не дають швидко сформувати вибірку за групою, дисципліною або періодом. Крім того, у таких підходах складно забезпечити розмежування прав доступу, журналювання дій користувачів і централізоване резервне копіювання.

Актуальність теми полягає в необхідності створення інформаційної системи, яка забезпечує централізований облік занять і відвідування студентами університету, підтримує різні ролі користувачів, автоматизує формування звітів і надає узагальнену аналітику для навчальних підрозділів. Така система зменшує навантаження на викладачів і працівників деканату, підвищує достовірність даних та забезпечує єдиний інформаційний простір для контролю відвідуваності.

Метою кваліфікаційної роботи є проєктування та розроблення інформаційної системи аналізу відвідування і занять студентами університету, яка забезпечує автоматизований облік занять, фіксацію статусів присутності, збереження причин відсутності, формування звітів та аналітичних показників.

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

- проаналізувати предметну область обліку відвідування;
- дослідити існуючі програмні рішення;
- сформувати функціональні, нефункціональні та безпекові вимоги; побудувати UML-моделі;
- розробити логічну і фізичну моделі даних; обґрунтувати вибір технологій;
- описати реалізацію бази даних,
- інтерфейсу та аналітичних модулів; виконати тестування системи й оцінити її ефективність.

Об’єктом дослідження є процес обліку відвідування занять студентами університету та формування звітно-аналітичної інформації для навчальних підрозділів.

Предметом дослідження є методи, моделі та програмні засоби розроблення веборієнтованої інформаційної системи аналізу відвідування і занять студентами університету.

Методи дослідження включають системний аналіз, об’єктно-орієнтоване моделювання UML, ER-моделювання, реляційне проєктування баз даних,

алгоритмізацію процесів формування звітів, методи тестування програмного забезпечення та оцінювання якості користувацького інтерфейсу.

Практичне значення роботи полягає у можливості застосування запропонованої системи для автоматизації обліку відвідування у навчальному підрозділі університету. Розроблені моделі, структура бази даних і алгоритми звітності можуть бути використані як основа для подальшої програмної реалізації або інтеграції з існуючими освітніми сервісами.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Предметна область інформаційної системи охоплює процеси організації навчальних занять, ведення списків академічних груп, обліку студентів, призначення викладачів на дисципліни, фіксації відвідування та підготовки звітної інформації. У межах університету ці процеси взаємопов’язані: розклад визначає набір занять, заняття прив’язуються до групи, дисципліни та викладача, а кожне заняття формує множину записів відвідування для студентів відповідної академічної групи.

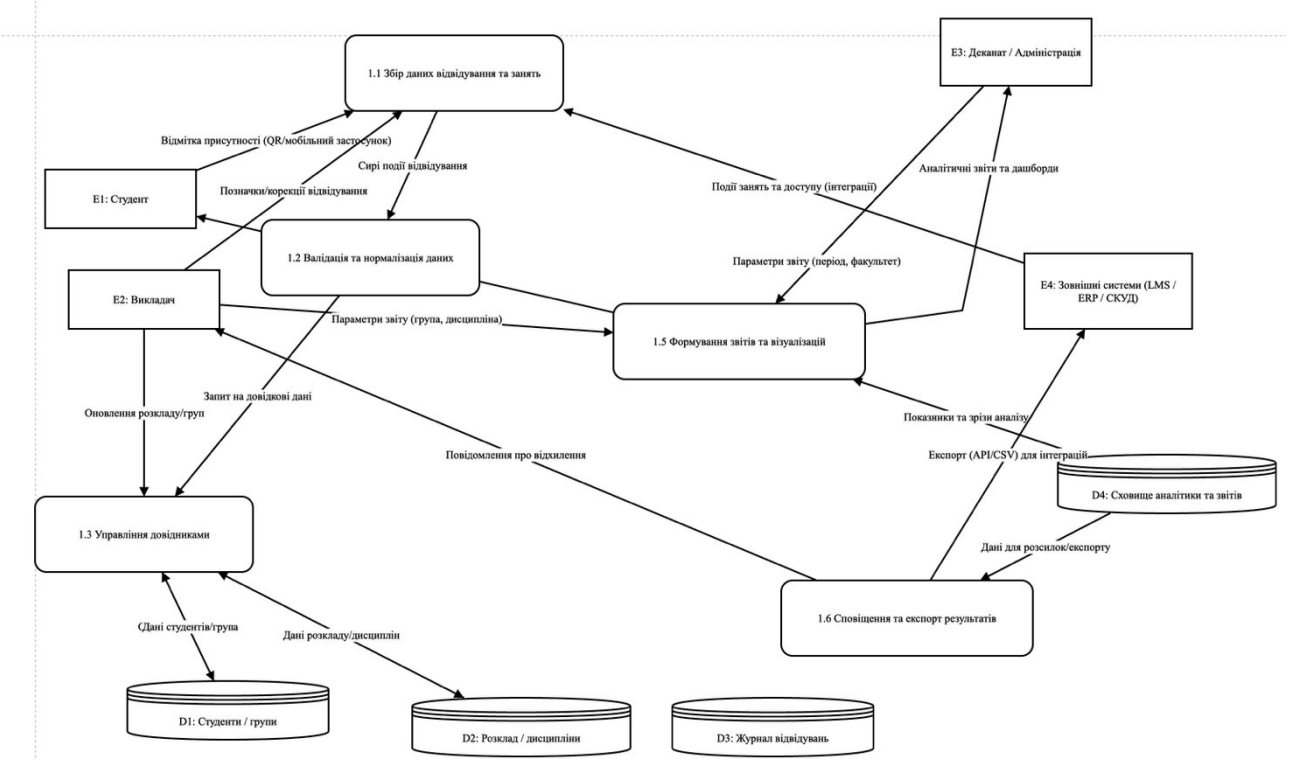

Рисунок 1.1 – Узагальнена структура процесу обліку відвідування студентів університету

Основними учасниками процесу є студент, викладач, працівник деканату та адміністратор системи. Студент переглядає власні записи відвідування, статуси пропусків і причини відсутності. Викладач створює або відкриває заняття, відмічає присутніх, фіксує запізнення та додає примітки. Деканат формує зведені звіти за групами, дисциплінами й періодами. Адміністратор

супроводжує довідники, облікові записи користувачів, права доступу та резервне копіювання даних.

Інформаційна система повинна підтримувати не лише операційний облік, а й аналітичну обробку накопичених даних. Для цього необхідно забезпечити збереження історії занять, статусів присутності, причин відсутності, часу внесення запису та відповідального користувача. Наявність таких даних дозволяє визначати частку пропусків, виявляти студентів із систематичною відсутністю, аналізувати динаміку відвідуваності та формувати підсумкові звіти для навчального підрозділу.

Загальна логіка предметної області полягає в переході від розрізненого ручного обліку до централізованої інформаційної моделі. У такій моделі кожен запис має однозначний зв’язок із заняттям, студентом, статусом відвідування та, за потреби, причиною відсутності. Це зменшує ймовірність дублювання інформації та забезпечує цілісність даних упродовж усього періоду навчання.

Узагальнену структуру процесу обліку відвідування студентів подано на рисунку 1.1. Вона демонструє взаємодію основних учасників із системою та показує, що ключові дані формуються під час заняття, а потім використовуються для звітності й аналітики.

Таблиця 1.1 – Основні інформаційні об’єкти предметної області

Об’єкт	Призначення	Основні дані
Академічна група	Об’єднує студентів за освітньою програмою, курсом і роком навчання	Код групи, курс, факультет, кафедра, навчальний рік
Студент	Є основною одиницею обліку відвідування	ПІБ, номер студентського квитка, група, електронна пошта, статус активності
Викладач	Проводить заняття та фіксує присутність студентів	ПІБ, посада, кафедра, контактні дані
Дисципліна	Визначає навчальний предмет, за яким проводяться заняття	Код, назва, кількість кредитів, форма контролю
Заняття	Фіксує конкретну навчальну подію	Дата, час початку і завершення, аудиторія, тема,

		тип заняття

Продовження таблиці 1.1

Запис відвідування	Зберігає результат відмітки студента на занятті	Студент, заняття, статус, причина відсутності, час фіксації, примітка

Отже, визначені інформаційні об’єкти формують основу предметної області системи аналізу відвідування і занять студентами університету. Їх взаємозв’язок забезпечує цілісне зберігання даних про навчальний процес, дозволяє уникнути дублювання інформації та створює передумови для автоматизованого формування звітів, контролю відвідуваності й подальшої аналітичної обробки результатів навчальної активності студентів.ї

1.2 Аналіз існуючих рішень для обліку відвідування та занять

Для розуміння вимог до розроблюваної системи було розглянуто кілька поширених підходів до обліку відвідування студентів. На практиці використовуються паперові журнали, електронні таблиці, модулі систем дистанційного навчання, сервіси для керування навчальними курсами, аналітичні панелі активності та спеціалізовані електронні журнали. Кожен із цих підходів вирішує частину завдань, однак не завжди відповідає потребам університетського підрозділу, де потрібні структурований облік занять, гнучка звітність і контроль доступу.

Паперовий журнал залишається простим інструментом, але має суттєві обмеження. Він не підтримує автоматичну перевірку коректності введення, не дозволяє швидко сформувати звіт за довільний період і не забезпечує резервного копіювання. Крім того, паперові записи складно використовувати для аналітики, оскільки їх потрібно повторно переносити в електронний вигляд.

Електронні таблиці частково усувають проблему підрахунків, однак не є повноцінною інформаційною системою. У них складно організувати

розмежування доступу, вести історію змін, контролювати зв’язки між групами, дисциплінами, заняттями та студентами. Якщо декілька користувачів редагують один файл, зростає ризик помилок, втрати даних або появи різних версій документа.

Одним із поширених рішень є Moodle з модулем Attendance. За офіційною документацією Moodle, активність Attendance призначена для того, щоб викладачі могли вести облік присутності під час занять, а студенти мали можливість переглядати власні записи. У модулі передбачено статуси на зразок присутній, відсутній, запізнився або звільнений [2]. Приклад такого підходу до обліку відвідування наведено на рисунку 1.2.

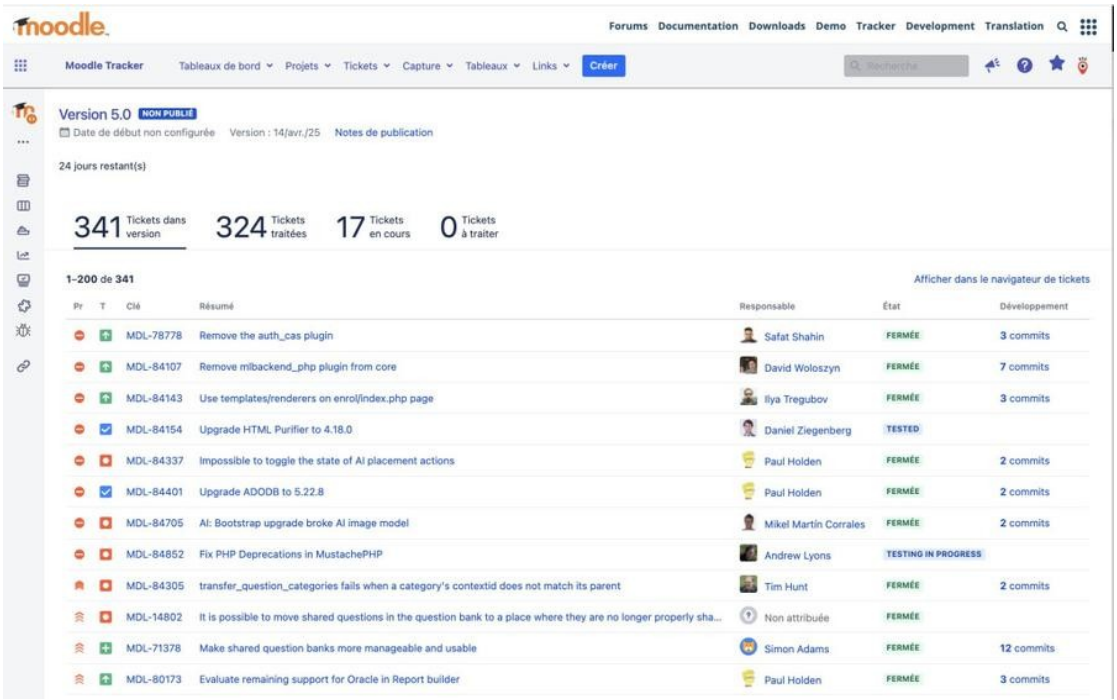

Рисунок 1.2 – Приклад використання модуля Attendance у середовищі Moodle

Перевагою Moodle є інтеграція з електронними курсами, журналом оцінювання та навчальними матеріалами. Водночас система є досить широкою за призначенням, тому її впровадження лише для обліку відвідування може бути надмірним. Також структура звітів і довідників Moodle не завжди повністю відповідає локальним вимогам конкретного факультету або кафедри.

Google Classroom орієнтований на керування навчальними класами, завданнями, матеріалами й комунікацією між викладачем та студентами. Google позиціонує Classroom як центральний простір для керування

навчальними процесами й взаємодії в освітньому середовищі [3]. Приклад інтерфейсу курсу Google Classroom подано на рисунку 1.3.

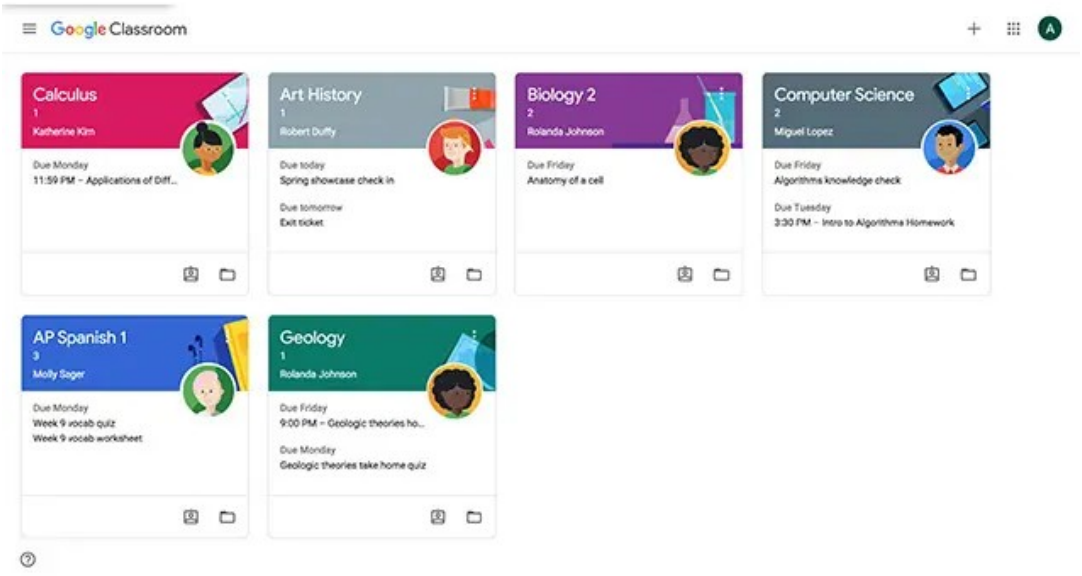

Рисунок 1.3 – Загальний вигляд навчального курсу в Google Classroom

Google Classroom є зручним для організації завдань і навчальних матеріалів, однак засоби фіксації присутності на заняттях не є його основним функціональним напрямом. Для систематичного обліку відвідування за академічними групами, дисциплінами, типами занять і причинами відсутності часто потрібні додаткові форми, таблиці або сторонні інструменти.

Microsoft Teams for Education разом із Education Insights забезпечує інструменти для відстеження активності студентів, виконання завдань, участі в обговореннях та взаємодії в цифровому навчальному середовищі. У документації Microsoft зазначено, що Insights надає викладачам представлення даних, які допомагають відстежувати активність студентів і тенденції їхньої участі [4]. Приклад аналітичної панелі такого типу наведено на рисунку 1.4.

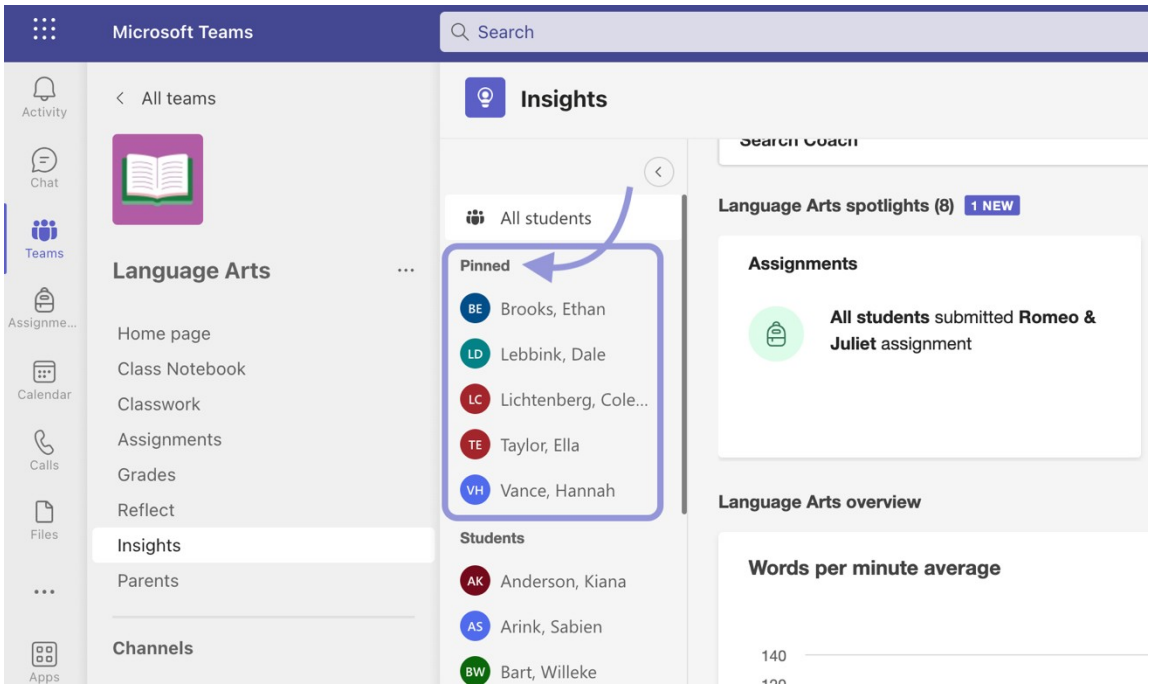

Рисунок 1.4 – Приклад аналітичної панелі Microsoft Education Insights

Такі системи добре підходять для змішаного або дистанційного навчання, але їхня аналітика переважно пов’язана з активністю в цифровому середовищі. Для університету, де потрібно вести формалізований облік аудиторних занять, зберігати причини відсутності та формувати звіти за локальними правилами, доцільно розробити спеціалізовану інформаційну систему.

Порівняння розглянутих підходів подано в таблиці 1.2. Аналіз показує, що розроблювана система повинна поєднати простоту щоденного використання, реляційну цілісність даних, рольову модель доступу, підтримку звітності та можливість подальшого розширення.

Таблиця 1.2 – Порівняння існуючих підходів до обліку відвідування

Рішення	Переваги	Обмеження для задачі роботи
Паперовий журнал	Простота використання, відсутність потреби в технічній інфраструктурі	Немає автоматичних розрахунків, пошуку, резервного копіювання та аналітики
Електронні таблиці	Швидке створення, базові формули, можливість експорту	Слабкий контроль доступу, ризик дублювання, складність роботи з

		великими вибірками

Продовження таблиці 1.2

Moodle Attendance	Підтримка статусів присутності, інтеграція з електронними курсами	Залежність від LMS, обмежена адаптація під локальну модель звітності
Google Classroom	Зручна робота із завданнями, матеріалами та комунікацією	Облік відвідування не є центральною функцією, потрібні додаткові інструменти
Microsoft Teams Education Insights	Аналітика активності студентів у цифровому середовищі	Орієнтація на активність у Teams, а не на повноцінний журнал аудиторних занять
Розроблювана система	Централізована БД, звіти, рольовий доступ, гнучкість під структуру університету	Потребує початкового розгортання та адміністрування

Отже, аналіз існуючих підходів показав, що універсальні навчальні платформи та прості засоби обліку не повністю вирішують завдання централізованого контролю відвідування. Тому розроблення спеціалізованої інформаційної системи є доцільним, оскільки вона дозволяє поєднати облік занять, фіксацію присутності студентів, рольовий доступ, звітність і аналітику в єдиному програмному середовищі.

1.3 Моделювання предметної області

Для формалізації предметної області використано UML-моделювання. Воно дає змогу описати ролі користувачів, основні сценарії роботи, послідовність обміну повідомленнями та логіку виконання процесів. На етапі аналізу побудовано діаграму прецедентів, діаграму послідовності, діаграму активності та модель абстракцій предметної області.

Діаграма прецедентів відображає взаємодію користувачів із системою. Основними акторами є студент, викладач, працівник деканату та адміністратор. Викладач фіксує відвідування, переглядає заняття та формує звіти по своїх групах. Деканат аналізує відвідуваність у розрізі груп і дисциплін.

Адміністратор керує довідниками та правами доступу. Студент переглядає власні записи. Відповідну діаграму наведено на рисунку 1.5.

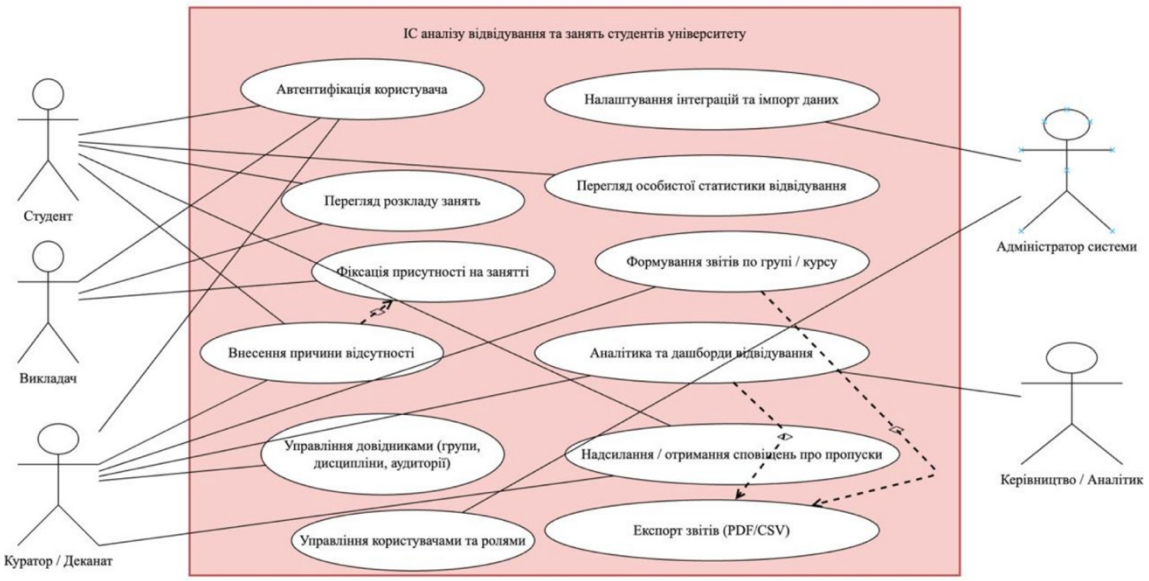

Рисунок 1.5 – Діаграма прецедентів інформаційної системи аналізу відвідування і занять студентами університету

Діаграма послідовності деталізує сценарій обліку відвідування під час заняття. Викладач відкриває форму заняття, система завантажує список студентів групи, користувач встановлює статуси відвідування, після чого сервер перевіряє коректність даних і зберігає записи у базі UniversityAttendanceDB. Цей процес показано на рисунку 1.6.

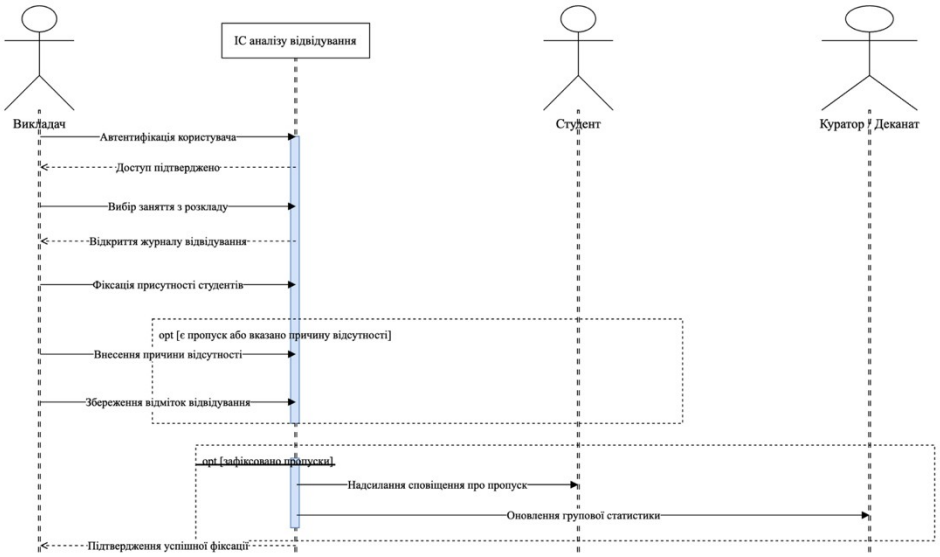

Рисунок 1.6 – Діаграма послідовності процесу фіксації відвідування на занятті

Діаграма активності описує логіку виконання процесу від моменту вибору заняття до формування підсумкового набору записів. Вона відображає перевірку параметрів, вибір студентів, встановлення статусів, збереження результатів і повідомлення користувача про успішне виконання операції або помилки введення. Структуру процесу наведено на рисунку 1.7.

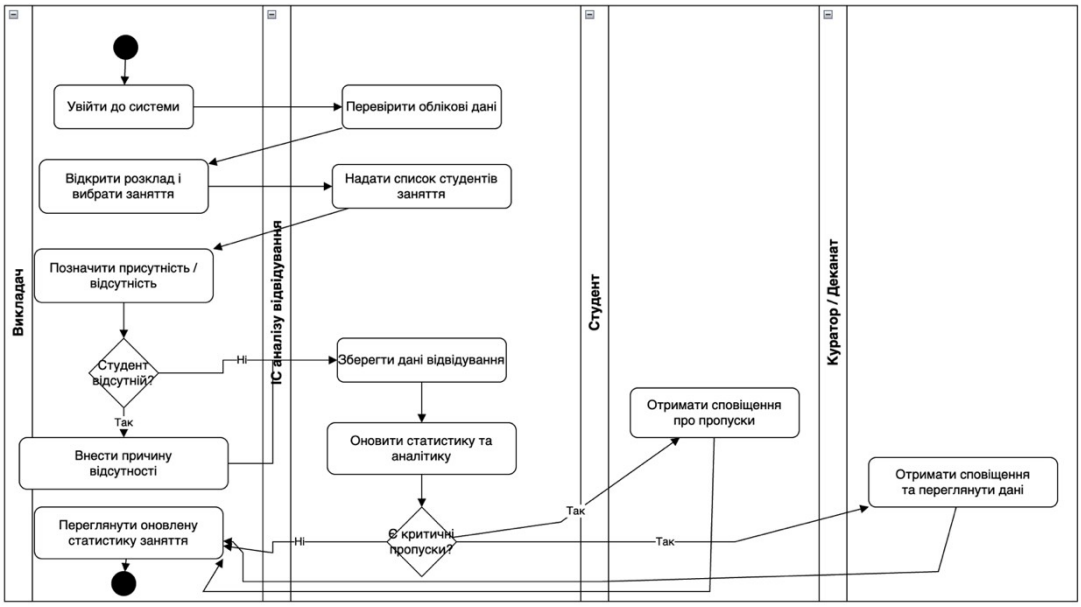

Рисунок 1.7 – Діаграма активності процесу обліку відвідування студентів

Абстракції предметної області дають змогу виокремити головні сутності системи та їхні взаємозв’язки. До них належать академічна група, студент, викладач, дисципліна, заняття, тип заняття, статус відвідування, причина відсутності та запис відвідування. Узагальнена модель абстракцій подана на рисунку 1.8.

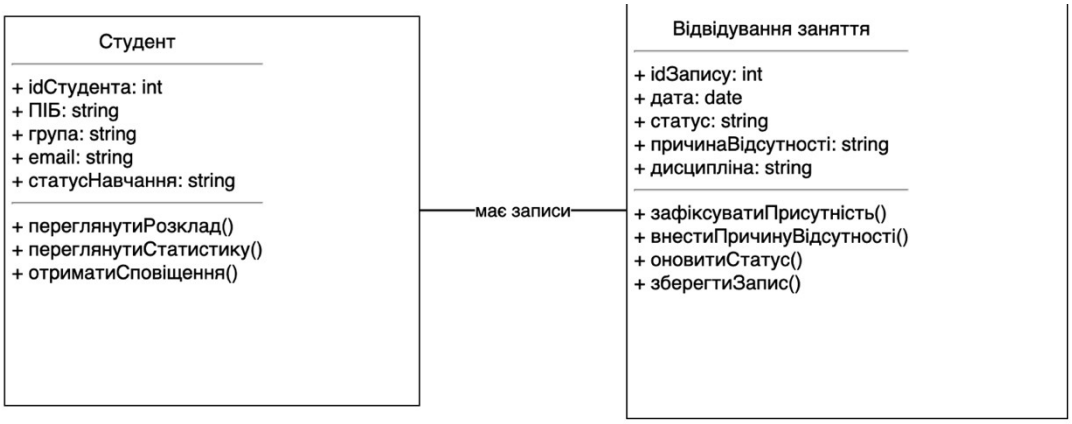

Рисунок 1.8 – Абстракції предметної області інформаційної системи

Отже, UML-моделювання дало змогу формалізувати предметну область інформаційної системи та уточнити логіку її роботи ще на етапі аналізу. Побудовані діаграми відображають основних користувачів системи, їхні функції, порядок фіксації відвідування, взаємодію між інтерфейсом, серверною частиною і базою даних, а також ключові сутності, які використовуються для зберігання навчальної інформації. Це забезпечує цілісне розуміння структури майбутньої системи та створює основу для подальшого проєктування бази даних, програмних модулів і механізмів аналітичної звітності.

1.4 Аналіз вимог до інформаційної системи

Вимоги до інформаційної системи сформовано з урахуванням ролей користувачів, структури навчального процесу та потреб у звітності. Система повинна забезпечувати щоденне внесення даних про заняття і відвідування, підтримувати довідники, виконувати перевірку коректності записів та надавати користувачам зручні засоби пошуку й аналізу.

Функціональні вимоги визначають, які дії повинна виконувати система. До них належать автентифікація користувачів, керування довідниками, створення занять, фіксація відвідування, облік причин відсутності, формування звітів і перегляд статистики. Узагальнений перелік функціональних вимог наведено в таблиці 1.3.

Таблиця 1.3 – Функціональні вимоги до системи

Код	Вимога	Опис реалізації
F1	Автентифікація користувачів	Вхід до системи за обліковими даними з урахуванням ролі користувача
F2	Керування академічними групами	Створення, редагування та перегляд груп, курсу, факультету й року навчання
F3	Керування студентами	Ведення списку студентів, прив’язка до групи, контроль активності
F4	Керування викладачами і дисциплінами	Збереження довідкових даних про викладачів, дисципліни та типи занять
F5	Створення занять	Формування заняття за групою,

		дисципліною, викладачем, датою і часом

Продовження таблиці 1.3

F6	Фіксація відвідування	Встановлення статусів присутності, відсутності або запізнення для кожного студента
F7	Облік причин відсутності	Збереження поважних і неповажних причин пропусків
F8	Формування звітів	Побудова звітів по групах, дисциплінах, студентах і періодах
F9	Аналітика відвідуваності	Розрахунок кількості занять, присутностей, пропусків, запізнень і відсотка відвідуваності
F10	Експорт даних	Вивантаження звітів у табличний формат для подальшого використання

Нефункціональні вимоги визначають якість роботи системи. Вони стосуються продуктивності, надійності, зручності інтерфейсу, безпеки, масштабованості та підтримуваності. Для навчального середовища особливо важливими є швидкий доступ до журналу заняття, стабільна робота з великою кількістю записів і зрозумілий інтерфейс, який не ускладнює роботу викладача під час заняття.

Таблиця 1.4 – Нефункціональні та безпекові вимоги

Група вимог	Зміст вимоги
Продуктивність	Відкриття основних сторінок системи має виконуватися без помітних затримок для користувача; звіти за типовими періодами повинні формуватися протягом кількох секунд
Надійність	Система повинна забезпечувати збереження даних після кожної операції та підтримувати резервне копіювання бази
Зручність	Інтерфейс має бути структурованим, з мінімальною кількістю дій для фіксації відвідування на занятті
Сумісність	Клієнтська частина повинна працювати у сучасних браузерах Google Chrome, Mozilla Firefox, Microsoft Edge і Safari
Безпека	Доступ до функцій має визначатися ролями; передавання даних між клієнтом і сервером повинно виконуватися через HTTPS
Цілісність даних	База повинна містити первинні й зовнішні ключі, унікальні

Група вимог	Зміст вимоги
	обмеження та перевірки допустимих значень
Масштабованість	Структура системи має дозволяти додавання нових груп, дисциплін, користувачів і звітних форм без зміни основної логіки

Отже, сформовані функціональні, нефункціональні та безпекові вимоги визначають основні напрями подальшого проєктування інформаційної системи. Вони охоплюють повний цикл роботи з даними про навчальні заняття та відвідування студентів: від автентифікації користувачів і ведення довідників до фіксації присутності, формування звітів та аналітичної обробки результатів. Урахування вимог до продуктивності, надійності, зручності інтерфейсу, цілісності даних і рольового доступу дає змогу побудувати систему, придатну для щоденного використання викладачами, працівниками деканату, студентами та адміністраторами.

1.5 Постановка завдання

На основі аналізу предметної області та існуючих рішень сформульовано завдання розроблення інформаційної системи аналізу відвідування і занять студентами університету. Система повинна забезпечити централізоване ведення даних про академічні групи, студентів, викладачів, дисципліни, типи занять, заняття, статуси відвідування, причини відсутності та записи відвідування.

Вхідними даними системи є облікові дані користувачів, відомості про групи, студентів, викладачів, дисципліни, параметри заняття, статуси відвідування, причини відсутності та періоди формування звітів. Вихідними даними є списки занять, журнали відвідування, звіти по групах і дисциплінах, індивідуальна статистика студентів, зведені аналітичні показники та експортовані таблиці.

Розроблювана система повинна мати клієнт-серверну архітектуру, вебінтерфейс для користувачів і реляційну базу даних UniversityAttendanceDB. Вебзастосунок Attendance Web App повинен забезпечувати оброблення запитів

користувачів, перевірку введених даних, взаємодію з базою даних і формування звітно-аналітичної інформації.

1.6 Висновки до першого розділу

У першому розділі виконано системний аналіз предметної області інформаційної системи аналізу відвідування і занять студентами університету. Визначено основні процеси, ролі користувачів, інформаційні об’єкти та проблеми традиційного обліку.

Проаналізовано існуючі підходи й програмні рішення, зокрема паперові журнали, електронні таблиці, Moodle Attendance, Google Classroom і Microsoft Teams Education Insights. Установлено, що вони частково вирішують задачу контролю навчальної активності, однак не забезпечують повністю спеціалізований облік занять, причин відсутності та звітність за структурою конкретного університетського підрозділу.

Побудовано UML-моделі предметної області: діаграму прецедентів, діаграму послідовності, діаграму активності та модель абстракцій. Сформовано функціональні, нефункціональні та безпекові вимоги, що є основою для подальшого проєктування інформаційного й програмного забезпечення системи.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Логічна модель даних інформаційної системи аналізу відвідування і занять студентами університету призначена для формалізованого опису основних інформаційних об’єктів предметної області, їхніх атрибутів і зв’язків між ними. Побудова ER-діаграми дає змогу визначити структуру майбутньої бази даних ще до етапу фізичної реалізації, перевірити повноту сутностей, усунути дублювання даних і забезпечити цілісність інформації під час обліку навчальних занять та відвідуваності.

У межах розроблюваної системи основними сутностями є академічна група, студент, викладач, дисципліна, заняття, запис відвідування та причина відсутності. Центральною сутністю моделі є ClassSession, оскільки саме заняття поєднує академічну групу, викладача, дисципліну, дату проведення, номер пари та тип заняття. Для кожного заняття формується множина записів відвідування AttendanceRecord, у яких фіксується статус конкретного студента.

Логічну модель даних інформаційної системи подано на рисунку 2.1.

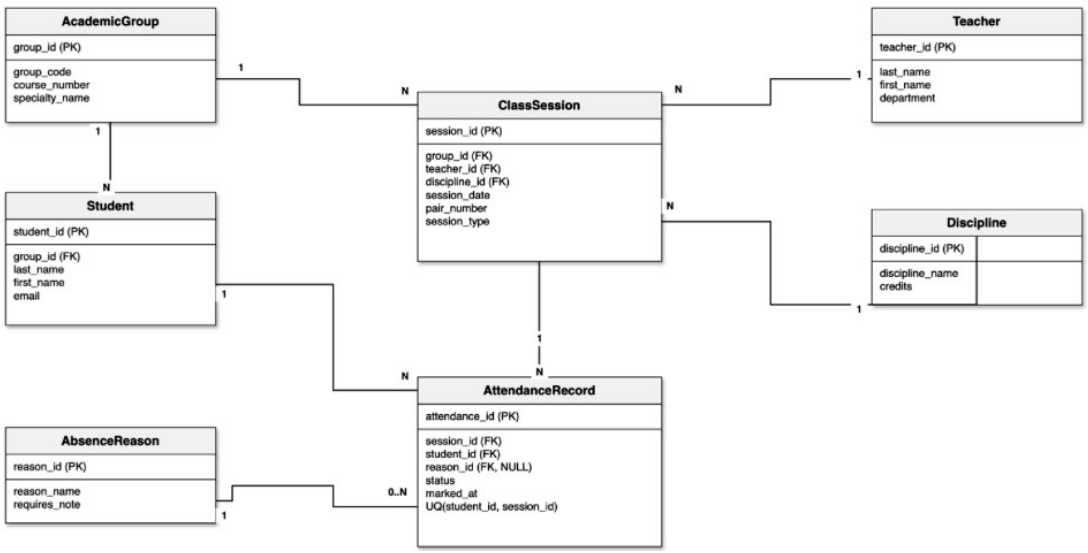

Рисунок 2.1 – Логічна модель даних інформаційної системи аналізу відвідування і занять студентами університету

У моделі AcademicGroup використовується для зберігання даних про академічні групи. Кожна група має власний ідентифікатор, код, номер курсу та назву спеціальності. Зв’язок між AcademicGroup і Student має тип один-до-багатьох, оскільки одна академічна група може містити багато студентів, але кожен студент належить лише до однієї групи. Аналогічно одна академічна група може мати багато занять, що відображено зв’язком AcademicGroup – ClassSession.

Сутність Student описує студента як основний об’єкт обліку відвідування. Вона містить ідентифікатор студента, зовнішній ключ групи, прізвище, ім’я та електронну пошту. Студент пов’язаний із записами відвідування через сутність AttendanceRecord. Такий зв’язок дає змогу зберігати історію присутності кожного студента за всіма заняттями, у яких він бере участь.

Сутність Teacher містить дані про викладачів, які проводять заняття. Один викладач може бути пов’язаний з багатьма заняттями, тому між Teacher і ClassSession встановлено зв’язок один-до-багатьох. Сутність Discipline використовується для зберігання навчальних дисциплін. Одна дисципліна також може мати багато занять, але кожне заняття належить лише до однієї дисципліни.

Сутність AttendanceRecord є зв’язувальною між Student і ClassSession. Вона забезпечує фіксацію факту відвідування конкретного студента на конкретному занятті. Для уникнення дублювання даних у цій сутності передбачено унікальне обмеження за парою полів student_id і session_id. Це означає, що один студент може мати лише один запис відвідування в межах одного заняття.

Сутність AbsenceReason призначена для зберігання причин відсутності. Вона пов’язана із записами відвідування необов’язковим зв’язком, оскільки причина вказується лише тоді, коли студент був відсутній або має відповідну примітку. Для присутніх студентів поле reason_id може залишатися порожнім.

Таблиця 2.1 – Характеристика сутностей логічної моделі даних

Сутність	Призначення	Основні атрибути	Ключові зв’язки
AcademicGroup	Зберігання даних про академічні групи	group_id, group_code, course_number, specialty_name	Пов’язана зі Student і ClassSession
Student	Зберігання даних про студентів	student_id, group_id, last_name, first_name, email	Належить до AcademicGroup, пов’язаний з AttendanceRecord
Teacher	Зберігання даних про викладачів	teacher_id, last_name, first_name, department	Пов’язаний з ClassSession
Discipline	Зберігання даних про навчальні дисципліни	discipline_id, discipline_name, credits	Пов’язана з ClassSession
ClassSession	Опис конкретного навчального заняття	session_id, group_id, teacher_id, discipline_id, session_date, pair_number, session_type	Пов’язана з AcademicGroup, Teacher, Discipline і AttendanceRecord
AttendanceRecord	Фіксація відвідування студента на занятті	attendance_id, session_id, student_id, reason_id, status, marked_at	Пов’язана зі Student, ClassSession і AbsenceReason
AbsenceReason	Зберігання причин відсутності	reason_id, reason_name, requires_note	Пов’язана з AttendanceRecord

У логічній моделі використано первинні та зовнішні ключі. Первинні ключі забезпечують однозначну ідентифікацію записів у кожній таблиці, а зовнішні ключі задають зв’язки між сутностями. Наприклад, поле group_id у таблиці Student вказує на групу студента, поля group_id, teacher_id і discipline_id у ClassSession визначають, до якої групи, викладача і дисципліни належить заняття, а поля session_id і student_id у AttendanceRecord пов’язують запис відвідування з конкретним заняттям і студентом.

Таблиця 2.2 – Основні зв’язки між сутностями

Зв’язок	Тип зв’язку	Зміст зв’язку
AcademicGroup – Student	1:N	Одна академічна група містить багато студентів
AcademicGroup – ClassSession	1:N	Для однієї групи може бути створено багато занять
Teacher – ClassSession	1:N	Один викладач може проводити багато занять
Discipline – ClassSession	1:N	Одна дисципліна може містити багато занять
ClassSession – AttendanceRecord	1:N	Одне заняття має багато записів відвідування
Student – AttendanceRecord	1:N	Один студент має багато записів відвідування
AbsenceReason – AttendanceRecord	1:0..N	Одна причина може використовуватися в багатьох записах відвідування

Запропонована ER-модель відповідає вимогам реляційної структури даних і забезпечує нормалізоване зберігання інформації. Дані про студентів, групи, викладачів, дисципліни, заняття та відвідування зберігаються окремо, що зменшує дублювання та спрощує оновлення довідкової інформації. Унікальне обмеження для пари student_id і session_id у таблиці

AttendanceRecord запобігає повторному внесенню одного й того самого студента в межах одного заняття.

Отже, логічна модель даних визначає основу для подальшої фізичної реалізації бази даних UniversityAttendanceDB. Вона забезпечує цілісне зберігання інформації про навчальні заняття та відвідування студентів, підтримує формування звітів за групами, дисциплінами й періодами, а також створює передумови для аналітичної обробки накопичених даних.

2.2 Фізична модель даних і структура бази UniversityAttendanceDB

Фізична модель даних є наступним етапом після побудови логічної ER-моделі та визначає, у якому вигляді інформація буде зберігатися в базі даних інформаційної системи. Для розроблюваної системи сформовано структуру бази даних UniversityAttendanceDB, яка забезпечує збереження даних про академічні групи, студентів, викладачів, дисципліни, заняття, причини відсутності та записи відвідування.

Фізична модель побудована за реляційним принципом. Кожна сутність предметної області реалізована у вигляді окремої таблиці, а зв’язки між ними задаються за допомогою первинних і зовнішніх ключів. Такий підхід забезпечує цілісність даних, запобігає дублюванню інформації та дозволяє коректно формувати звіти про відвідуваність у розрізі студентів, груп, дисциплін, викладачів і періодів навчання.

Фізичну модель бази даних UniversityAttendanceDB подано на рисунку 2.2.

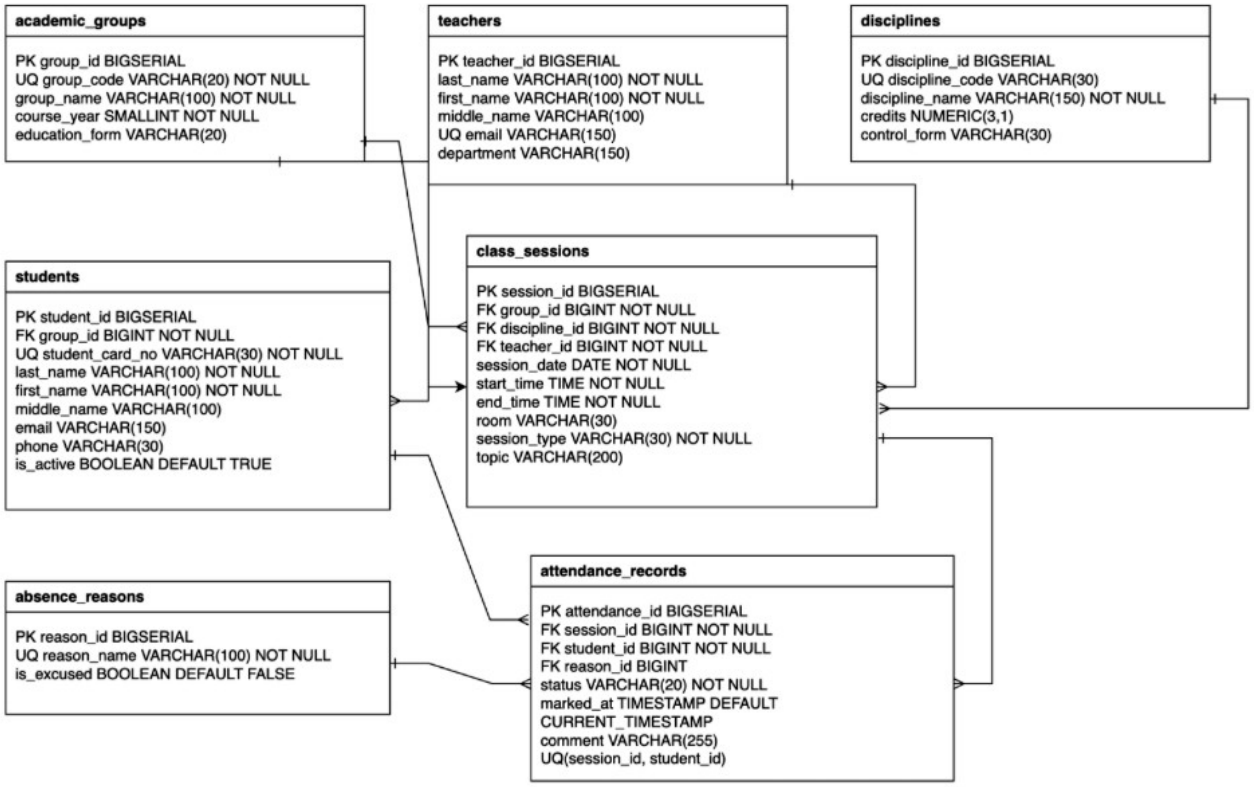

Рисунок 2.2 – Фізична модель даних бази UniversityAttendanceDB

Основу бази даних становлять довідникові та операційні таблиці. До довідникових належать academic_groups, teachers, disciplines і absence_reasons. Вони зберігають відносно сталі дані, які використовуються під час створення занять і заповнення журналів відвідування. Операційні таблиці students, class_sessions та attendance_records забезпечують безпосередню роботу системи з навчальним процесом: облік студентів, створення занять і фіксацію присутності або відсутності студентів.

Таблиця 2.3 – Структура основних таблиць бази даних UniversityAttendanceDB

Таблиця	Призначення	Ключові поля	Особливості зберігання даних
academic_groups	Зберігає дані про академічні групи	group_id, group_code	Код групи є унікальним; додатково зберігаються назва групи, курс і форма навчання
students	Містить інформацію про	student_id, group_id,	Кожен студент прив’язаний до

	студентів	student_card_no	академічної групи; номер студентського квитка є унікальним
teachers	Зберігає дані про викладачів	teacher_id, email	Передбачено збереження ПІБ, кафедри та електронної пошти викладача
disciplines	Містить перелік навчальних дисциплін	discipline_id, discipline_code	Для дисципліни зберігаються назва, кількість кредитів і форма контролю
class_sessions	Зберігає інформацію про проведені або заплановані заняття	session_id, group_id, discipline_id, teacher_id	Заняття прив’язується до групи, дисципліни й викладача; фіксуються дата, час, аудиторія, тип і тема заняття
attendance_records	Фіксує результат відвідування студентом конкретного заняття	attendance_id, session_id, student_id, reason_id	Зберігає статус присутності, час внесення запису та примітку; обмеження UQ(session_id, student_id) не допускає дублювання
absence_reasons	Містить перелік причин відсутності	reason_id, reason_name	Дає змогу розділяти поважні й неповажні причини пропусків

У таблиці academic_groups зберігаються дані про академічні групи, які є базовою одиницею організації навчального процесу. Через поле group_id ця таблиця пов’язується зі студентами та заняттями. Це дає змогу формувати журнали відвідування не для окремих випадкових користувачів, а для конкретних академічних груп.

Таблиця students містить персональні навчальні дані студентів. Для кожного студента зберігається належність до групи, номер студентського квитка, прізвище, ім’я, по батькові, електронна пошта, номер телефону та ознака активності. Поле is_active дозволяє не видаляти студента з бази, а лише змінювати його статус, що важливо для збереження історії відвідування.

Таблиці teachers і disciplines використовуються під час створення занять. Завдяки цьому кожне заняття має чітке посилання на викладача та дисципліну, а система може надалі формувати звіти не тільки по групах, а й за конкретними навчальними предметами або викладачами.

Центральною операційною таблицею є class_sessions. Вона описує конкретне заняття: дату проведення, час початку і завершення, аудиторію, тип заняття та тему. Саме на основі цієї таблиці викладач відкриває журнал і вносить дані про відвідування студентів.

Таблиця attendance_records призначена для збереження результатів обліку відвідування. Один запис у цій таблиці відповідає одному студенту на одному занятті. Для контролю коректності передбачено унікальне обмеження за парою session_id і student_id, яке не дозволяє створити два записи відвідування для одного студента в межах одного заняття. Це є важливою умовою достовірності журналу.

Для фіксації причин пропусків використовується таблиця absence_reasons. Вона дозволяє не вводити причини вручну кожного разу, а використовувати попередньо визначений довідник. Це спрощує статистичну обробку даних і дає змогу окремо аналізувати поважні та неповажні пропуски.

Таблиця 2.4 – Обмеження цілісності у фізичній моделі даних

Тип обмеження	Реалізація в базі даних	Призначення
Первинні ключі	group_id, student_id, teacher_id, discipline_id, session_id, attendance_id, reason_id	Однозначна ідентифікація записів у таблицях
Зовнішні ключі	group_id, teacher_id, discipline_id, session_id, student_id, reason_id	Підтримка зв’язків між таблицями та заборона

		некоректних посилань

Продовження таблиці 2.4

Унікальні обмеження	group_code, student_card_no, email, discipline_code, reason_name	Запобігання дублюванню довідкових і персональних даних
Перевірка обов’язкових полів	NOT NULL для основних атрибутів	Забезпечення повноти критично важливих даних
Значення за замовчуванням	is_active, is_excused, marked_at	Автоматичне заповнення типових значень під час створення записів
Унікальність запису відвідування	UQ(session_id, student_id)	Захист від повторного внесення студента в журнал одного заняття

Фізична структура бази даних орієнтована на щоденну роботу з навчальними журналами. Вона підтримує швидкий вибір списку студентів за групою, перегляд занять за датою, формування звітів за дисциплінами та аналіз відвідуваності окремого студента. Для підвищення продуктивності доцільно передбачити індекси для полів, які найчастіше використовуються у вибірках: group_id, student_id, teacher_id, discipline_id, session_date і status.

Запропонована структура бази UniversityAttendanceDB є достатньою для реалізації основних функцій інформаційної системи: створення занять, ведення списків студентів, фіксації присутності, обліку причин відсутності, формування звітів і побудови аналітики. Водночас модель залишається розширюваною: за потреби до неї можна додати таблиці користувачів, ролей, навчальних років, семестрів, розкладу або журналу дій без порушення основної логіки системи.

Отже, фізична модель даних забезпечує практичну основу для реалізації інформаційної системи аналізу відвідування і занять студентами університету. Вона поєднує довідкові та операційні дані, підтримує цілісність зв’язків між ними та створює умови для надійного зберігання, пошуку й аналітичної обробки інформації про відвідуваність студентів.

2.3 Діаграма компонентів і пакетна структура

Для опису архітектури програмного забезпечення інформаційної системи аналізу відвідування і занять студентами університету побудовано діаграму компонентів та діаграму пакетів. Такі UML-моделі дають змогу показати склад основних програмних модулів, їхню взаємодію між собою та поділ системи на логічні частини. У межах роботи система розглядається як веборієнтований застосунок, що складається з клієнтської частини, серверної частини та рівня зберігання даних.

Діаграма компонентів відображає загальну структуру програмної системи та зв’язки між її функціональними модулями. Клієнтська частина містить веб-інтерфейс користувача, форму входу й реєстрації, журнал відвідування, сторінки звітів та аналітики, а також рольові кабінети студента, викладача, працівника деканату й адміністратора. Через ці компоненти користувачі виконують основні дії в системі: переглядають заняття, фіксують відвідування, формують звіти та працюють із доступними відповідно до ролі даними.

Серверна частина Attendance Web App реалізує бізнес-логіку системи. До її складу входять REST API, сервіс автентифікації, сервіс керування заняттями, сервіс обліку відвідування, сервіс експорту звітних файлів і шар доступу до даних. REST API приймає запити від клієнтської частини та передає їх відповідним сервісам. Сервіс автентифікації перевіряє облікові дані користувачів, сервіс обліку відвідування відповідає за створення й оновлення записів присутності, а шар доступу до даних забезпечує взаємодію з базою UniversityAttendanceDB.

Рівень даних представлено базою UniversityAttendanceDB, у якій зберігаються таблиці academic_groups, students, teachers, disciplines, class_sessions, attendance_records та absence_reasons. Окремо передбачено компоненти резервного копіювання та журналювання дій, що необхідні для підвищення надійності системи й контролю змін у важливих записах. Діаграму компонентів інформаційної системи наведено на рисунку 2.3.

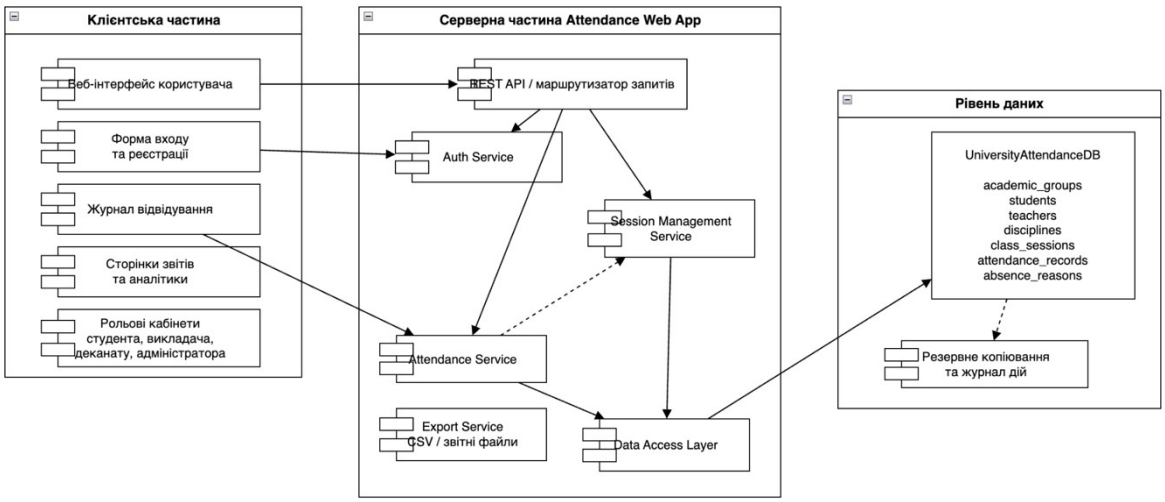

Рисунок 2.3 – Діаграма компонентів інформаційної системи аналізу відвідування і занять студентами університету

Основні компоненти системи та їх призначення подано в таблиці 2.5.

Таблиця 2.5 – Призначення основних компонентів інформаційної системи

Компонент	Призначення
Веб-інтерфейс користувача	Забезпечує доступ користувачів до функцій системи через браузер
Модуль автентифікації	Виконує вхід, реєстрацію та перевірку ролі користувача
Журнал відвідування	Дозволяє викладачу фіксувати присутність, відсутність і запізнення студентів
Сторінки звітів та аналітики	Формують статистику відвідуваності за групами, дисциплінами й періодами
REST API	Приймає запити клієнтської частини та передає їх серверним сервісам
Attendance Service	Реалізує логіку створення й оновлення записів відвідування
Session Management Service	Забезпечує роботу із заняттями, датами, темами та типами занять
Export Service	Формує звітні файли для подальшого використання або збереження
Data Access Layer	Виконує запити до бази даних і повертає результати серверним модулям

UniversityAttendanceDB	Зберігає довідкові, навчальні та операційні дані системи

Пакетна структура деталізує логічну організацію програмного коду. У системі виділено два основні пакети: клієнтську та серверну частини. Клієнтська частина містить веб-інтерфейс, модуль автентифікації та модуль перегляду звітів. Вона відповідає за відображення сторінок, введення користувацьких даних і передавання запитів до серверної частини.

Серверна частина містить сервіс автентифікації, сервіс обліку відвідування, сервіс аналітики та звітності, а також пакет доступу до даних. Такий поділ дозволяє відокремити інтерфейс користувача від бізнес-логіки та роботи з базою даних. Завдяки цьому окремі модулі можна змінювати або розширювати без повної перебудови всієї системи. Пакетну структуру інформаційної системи подано на рисунку 2.4.

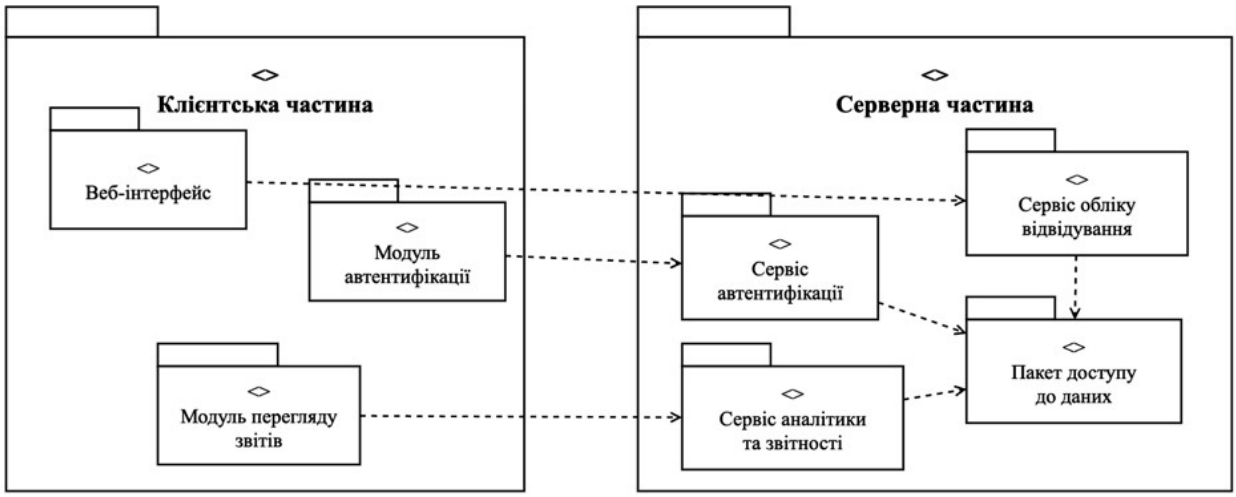

Рисунок 2.4 – Діаграма пакетів інформаційної системи аналізу відвідування і занять студентами університету

Запропонована компонентна й пакетна структура забезпечує модульність програмного забезпечення, спрощує супровід системи та створює умови для її подальшого розвитку. Відокремлення клієнтської частини, серверної логіки та рівня даних дозволяє реалізувати рольовий доступ, централізоване зберігання інформації, формування звітів і стабільну роботу системи з великою кількістю записів відвідування.

2.4 Алгоритмічне забезпечення звітності та аналітики

Алгоритмічне забезпечення інформаційної системи аналізу відвідування і занять студентами університету призначене для автоматизованого відбору, оброблення та узагальнення даних, які зберігаються в базі UniversityAttendanceDB. Основне призначення цих алгоритмів полягає у формуванні звітів для викладачів, працівників деканату та адміністратора системи. За допомогою звітності користувачі можуть швидко отримати інформацію про присутність студентів, кількість пропусків, запізнення, причини відсутності та загальний рівень відвідуваності.

У системі передбачено три основні алгоритми аналітичної обробки даних: формування звіту про відвідуваність групи за період, формування звіту про пропущені заняття за дисципліною та побудова зведеної статистики по студентах і групах. Усі ці алгоритми мають спільну послідовність роботи: користувач задає параметри відбору, система перевіряє їх коректність, звертається до бази даних, відбирає потрібні записи, виконує групування та формує результат у вигляді таблиці, аналітичного підсумку або файлу для експорту.

Алгоритм формування звіту про відвідуваність групи за період наведено на рисунку 2.5. Його виконання починається з відкриття форми формування звіту. Користувач задає період, академічну групу, факультет і форму навчання. Після цього система перевіряє, чи всі обов’язкові параметри заповнено правильно. Якщо дані введено некоректно, користувач отримує повідомлення про помилку та повертається до заповнення форми. Якщо параметри задано правильно, система формує запит до бази даних і відбирає інформацію з таблиць academic_groups, students, class_sessions та attendance_records.

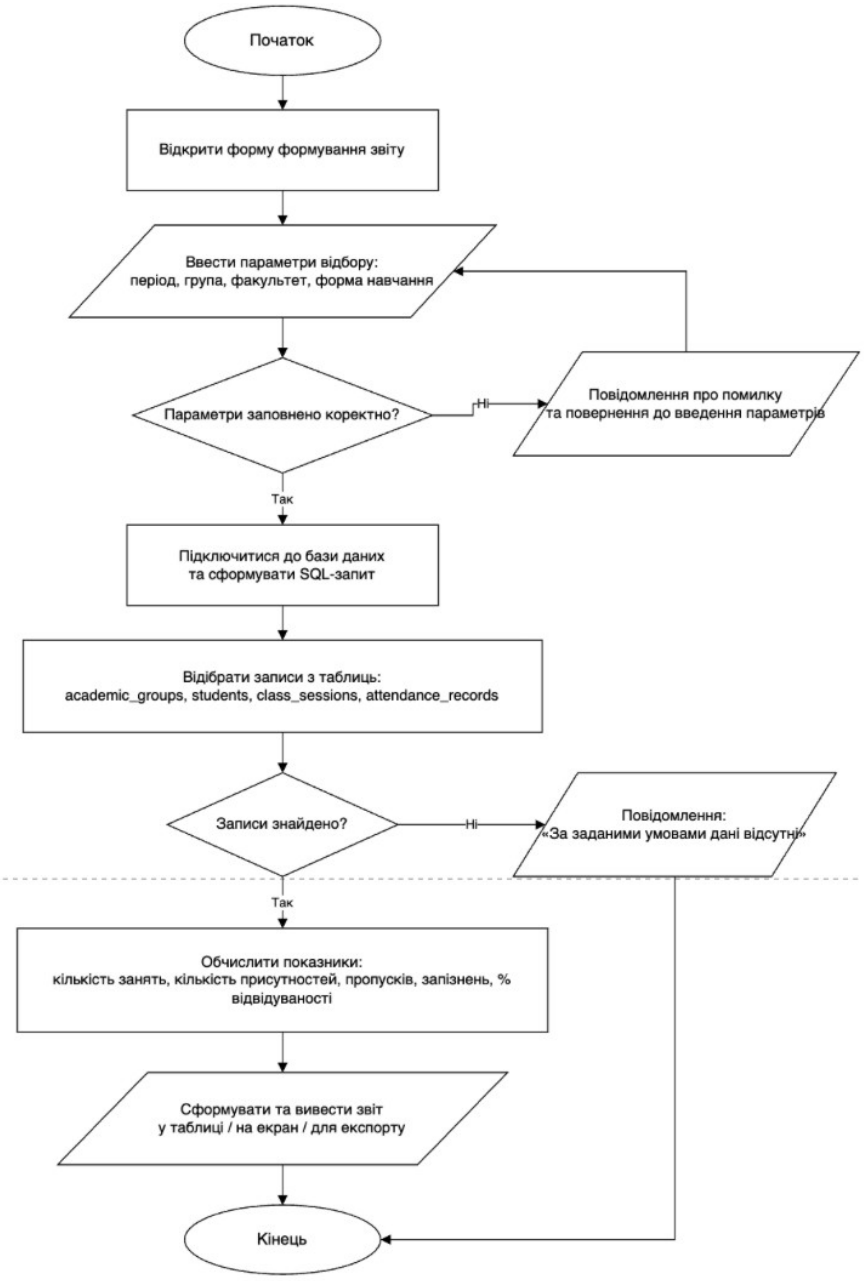

Рисунок 2.5 – Алгоритм формування звіту про відвідуваність групи за період

Після отримання даних система перевіряє, чи існують записи за заданими умовами. Якщо записи відсутні, користувачу виводиться повідомлення про відсутність даних. Якщо дані знайдено, система визначає кількість занять, присутностей, пропусків, запізнень і загальний рівень відвідуваності. Результат подається у вигляді таблиці, яку можна переглянути на екрані або експортувати для подальшого використання.

Алгоритм формування звіту про пропущені заняття за дисципліною подано на рисунку 2.6. Він використовується для пошуку та узагальнення

інформації про відсутність студентів на заняттях з конкретної дисципліни. Користувач обирає режим звіту, задає дисципліну, період, викладача та за потреби причину відсутності. Після перевірки параметрів система звертається до таблиць disciplines, class_sessions, attendance_records, absence_reasons та students.

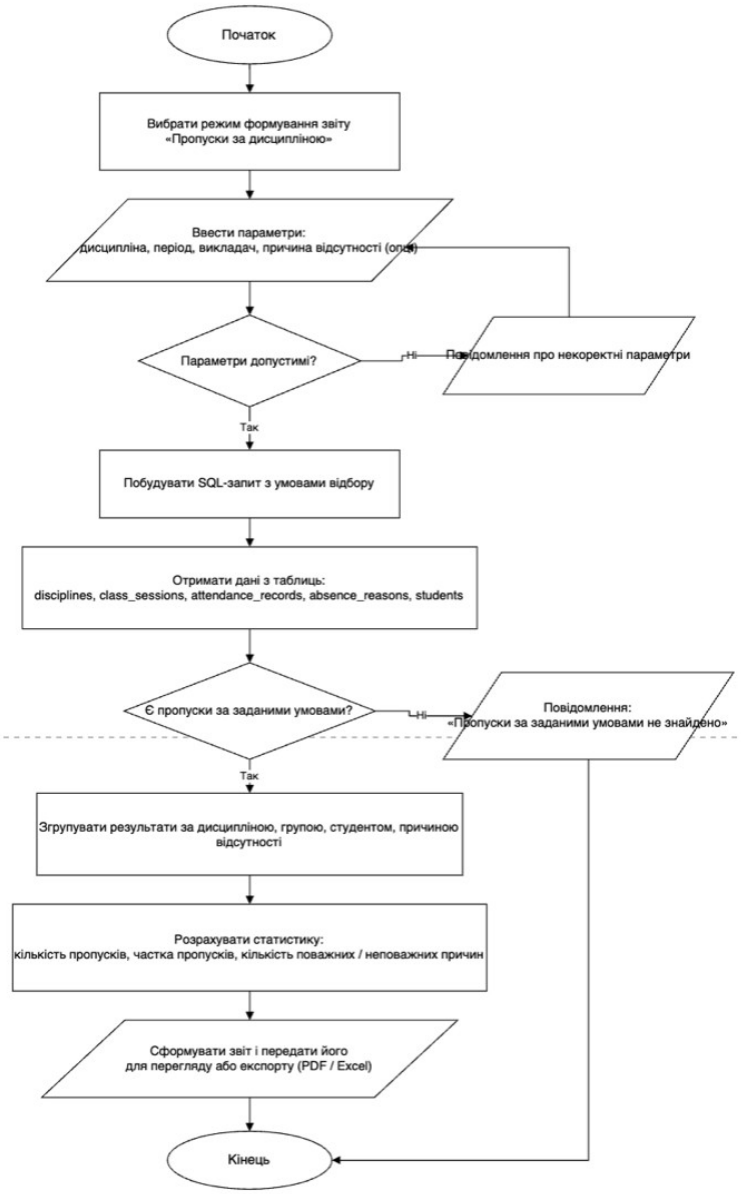

Рисунок 2.6 – Алгоритм формування звіту про пропущені заняття за дисципліною

Якщо за заданими умовами пропусків не знайдено, система повідомляє про це користувача. Якщо записи наявні, результати групуються за дисципліною, академічною групою, студентом і причиною відсутності. На основі відібраних даних формується звіт, у якому відображаються кількість

пропусків, їх характер і належність до поважних або неповажних причин. Такий звіт дає змогу виявляти студентів із систематичними пропусками та аналізувати ситуацію за окремими дисциплінами.

Алгоритм побудови зведеної статистики по студентах і групах наведено на рисунку 2.7. Він призначений для комплексного аналізу відвідуваності за більшим набором даних. Користувач задає критерії відбору: період, факультет, групу, курс і за потреби дисципліну. Система перевіряє введені параметри, виконує запит до бази даних і об’єднує дані з таблиць academic_groups, students, class_sessions, attendance_records та disciplines.

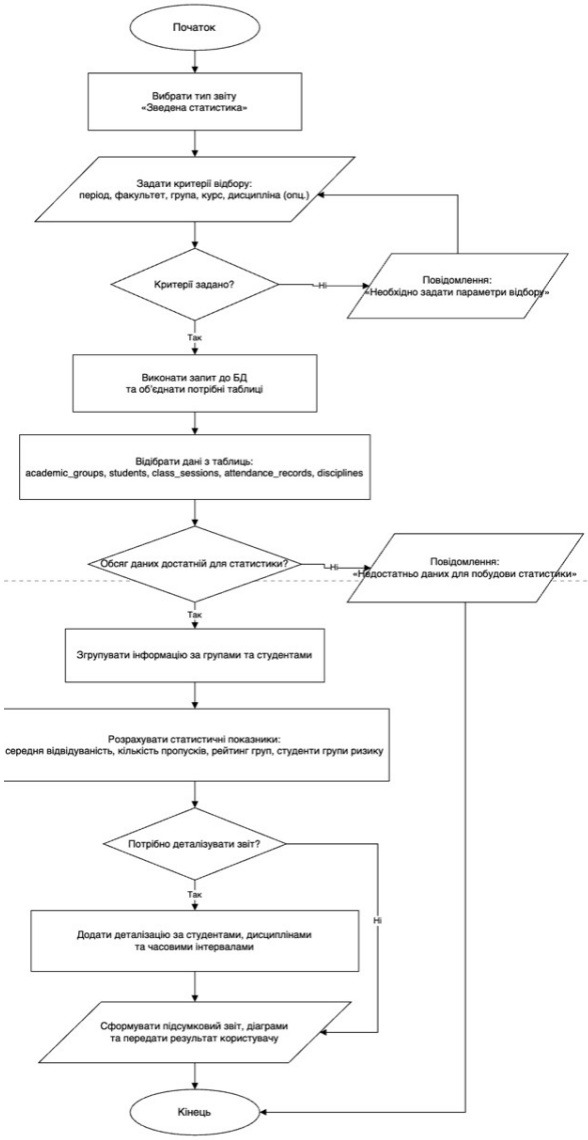

Рисунок 2.7 – Алгоритм формування зведеної статистики по студентах і групах

Після вибірки система визначає, чи достатньо даних для побудови статистики. Якщо даних недостатньо, користувач отримує відповідне повідомлення. Якщо обсяг інформації є достатнім, записи групуються за академічними групами, студентами та дисциплінами. Далі система формує підсумкові показники: середній рівень відвідуваності, кількість пропусків, рейтинг груп, перелік студентів із низькою відвідуваністю та загальну динаміку навчальної активності.

Таблиця 2.6 – Характеристика алгоритмів звітності та аналітики

Алгоритм	Вхідні дані	Основні операції	Вихідний результат
Формування звіту про відвідуваність групи	Період, група, факультет, форма навчання	Перевірка параметрів, вибірка занять і студентів, підрахунок присутностей, пропусків і запізнень	Таблиця відвідуваності групи за вибраний період
Формування звіту про пропущені заняття	Дисципліна, період, викладач, причина відсутності	Вибірка пропусків, групування за студентами, дисциплінами й причинами	Звіт про пропуски за дисципліною
Побудова зведеної статистики	Період, факультет, група, курс, дисципліна	Агрегація даних, узагальнення показників, визначення груп ризику	Зведена статистика по групах і студентах

Важливою частиною алгоритмічного забезпечення є перевірка параметрів відбору. Вона запобігає формуванню некоректних запитів до бази даних і зменшує кількість помилок користувача. Система повинна контролювати заповнення обов’язкових полів, правильність вибору періоду, наявність групи, дисципліни або викладача в базі даних. Такий підхід підвищує стабільність роботи системи та робить процес формування звітів більш зручним для користувача.

Під час аналітичної обробки система виконує групування та підрахунок записів за основними статусами відвідування. Окремо враховуються присутності, пропуски, запізнення, поважні й неповажні причини відсутності. Це дозволяє не лише зафіксувати факт відвідування, а й оцінити загальну дисципліну студентів, визначити проблемні групи та підготувати інформацію для прийняття управлінських рішень.

Таблиця 2.7 – Основні аналітичні показники системи

Показник	Призначення
Загальна кількість занять	Визначає обсяг навчальних подій за вибраний період
Кількість присутностей	Показує участь студента або групи в заняттях
Кількість пропусків	Використовується для контролю відсутності студентів
Кількість запізнень	Дозволяє окремо аналізувати порушення дисципліни відвідування
Рівень відвідуваності	Характеризує загальний стан присутності студента або групи
Кількість поважних причин	Дає змогу відокремити підтверджені пропуски
Рейтинг груп або студентів	Використовується для порівняльної аналітики та виявлення проблемних груп

Запропоновані алгоритми забезпечують перехід від простого зберігання записів відвідування до повноцінної аналітичної обробки навчальних даних. Вони автоматизують типові дії викладачів і працівників деканату, скорочують час підготовки звітів і підвищують достовірність результатів, оскільки всі розрахунки виконуються на основі даних, збережених у централізованій базі.

Отже, алгоритмічне забезпечення звітності та аналітики є важливою складовою інформаційної системи. Воно забезпечує формування звітів, контроль коректності введених параметрів, оброблення записів відвідування та підготовку показників, необхідних для оцінювання навчальної активності

студентів. Така логіка роботи створює основу для практичного використання системи в умовах університету.

2.5 Висновки до другого розділу

У другому розділі виконано проєктування інформаційного, структурного та алгоритмічного забезпечення інформаційної системи аналізу відвідування і занять студентами університету. На основі аналізу предметної області побудовано логічну модель даних у вигляді ER-діаграми, у якій визначено основні сутності системи: академічна група, студент, викладач, дисципліна, заняття, запис відвідування та причина відсутності. Встановлено зв’язки між цими сутностями, що дало змогу формалізувати процес обліку відвідування та забезпечити однозначну структуру зберігання навчальних даних.

Розроблено фізичну модель бази даних UniversityAttendanceDB, яка відображає практичну реалізацію логічної структури в реляційній базі даних. Визначено основні таблиці, первинні та зовнішні ключі, унікальні обмеження, обов’язкові поля та зв’язки між таблицями. Така структура забезпечує цілісність даних, запобігає дублюванню записів і дозволяє коректно зберігати інформацію про групи, студентів, викладачів, дисципліни, заняття та результати відвідування.

Побудовано діаграму компонентів і пакетну структуру системи. Визначено поділ програмного забезпечення на клієнтську частину, серверну частину та рівень даних. Клієнтська частина забезпечує взаємодію користувачів із системою через веб-інтерфейс, серверна частина реалізує бізнес-логіку, автентифікацію, облік відвідування, роботу із заняттями, звітність та експорт даних, а база UniversityAttendanceDB відповідає за централізоване зберігання інформації. Такий підхід підвищує модульність системи та спрощує її подальше розширення.

Також у розділі розроблено алгоритмічне забезпечення звітності та аналітики. Описано алгоритми формування звіту про відвідуваність групи за період, звіту про пропущені заняття за дисципліною та зведеної статистики по

студентах і групах. Передбачено перевірку параметрів відбору, звернення до бази даних, групування записів, підготовку підсумкових показників і формування результатів для перегляду або експорту.

Отже, у другому розділі сформовано основу для подальшої програмної реалізації системи. Запропоновані моделі даних, компонентна структура та алгоритми забезпечують логічну цілісність проєкту, підтримують рольовий доступ, централізоване зберігання даних, автоматизоване формування звітів і аналітичну обробку відвідуваності студентів.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Обґрунтування вибору технологій

Для реалізації інформаційної системи аналізу відвідування і занять студентами університету обрано веборієнтований підхід. Таке рішення є доцільним, оскільки система повинна бути доступною для кількох категорій користувачів: студентів, викладачів, працівників деканату та адміністратора. Використання вебзастосунку дозволяє працювати із системою через звичайний браузер без встановлення окремого клієнтського програмного забезпечення на кожен комп’ютер.

Архітектура системи побудована за клієнт-серверним принципом. Клієнтська частина відповідає за відображення інтерфейсу, введення даних і передавання запитів до сервера. Серверна частина реалізує бізнес-логіку, перевірку прав доступу, обробку даних про заняття та відвідування, формування звітів і взаємодію з базою даних. Такий поділ спрощує супровід системи, підвищує її масштабованість і дає змогу централізовано контролювати доступ до навчальних даних.

Для розроблення клієнтської частини обрано HTML, CSS і JavaScript. HTML використовується для формування структури сторінок, CSS — для оформлення інтерфейсу, а JavaScript — для обробки дій користувача, перевірки введених даних і динамічного оновлення елементів сторінки. Такий набір технологій є достатнім для створення зрозумілого веб-інтерфейсу з формами авторизації, реєстрації, журналом відвідування, сторінками звітів і панелями для різних ролей користувачів.

Для серверної частини доцільно використати Node.js з фреймворком Express.js. Node.js забезпечує виконання серверного JavaScript-коду, а Express.js дає змогу швидко організувати маршрутизацію запитів, обробку форм, реалізацію API та взаємодію між клієнтською частиною і базою даних. Такий підхід добре підходить для навчальної інформаційної системи, оскільки дозволяє реалізувати авторизацію, рольову модель доступу, облік занять,

фіксацію відвідування та формування звітів у межах одного програмного середовища.

Для зберігання даних обрано реляційну базу UniversityAttendanceDB. Її структура відповідає предметній області та містить таблиці академічних груп, студентів, викладачів, дисциплін, занять, причин відсутності та записів відвідування. Реляційна модель є доцільною для цієї задачі, оскільки дані мають чіткі зв’язки: студент належить до групи, заняття пов’язане з викладачем і дисципліною, а запис відвідування прив’язується до конкретного студента і конкретного заняття.

Для роботи з базою даних використовується SQL. Це дає змогу виконувати вибірку даних за періодами, групами, дисциплінами та студентами, формувати аналітичні запити, підраховувати кількість присутностей, пропусків і запізнень. SQL також дозволяє забезпечити цілісність даних за допомогою первинних і зовнішніх ключів, унікальних обмежень та перевірок допустимих значень.

Окрему увагу приділено вибору технологій для авторизації та розмежування доступу. У системі передбачено декілька ролей: адміністратор, деканат, викладач і студент. Кожна роль має власний набір доступних функцій. Адміністратор керує користувачами та довідниками, деканат переглядає звіти й аналітику, викладач працює із власними заняттями та журналами, а студент переглядає тільки особисті дані про відвідування. Такий підхід забезпечує захист інформації та запобігає несанкціонованій зміні записів.

Для експорту звітної інформації передбачено використання табличного формату CSV. Це простий і практичний формат, який підтримується більшістю офісних програм і може бути використаний для подальшого аналізу, збереження або передавання звітів працівникам навчального підрозділу. CSV є достатнім для експорту журналів, статистики відвідування та зведених таблиць.

Таблиця 3.1 – Обґрунтування вибору технологій реалізації системи

Технологія	Призначення в системі	Обґрунтування вибору
HTML	Формування структури вебсторінок	Забезпечує створення форм, таблиць, меню, сторінок журналу та звітів
CSS	Оформлення інтерфейсу користувача	Дає змогу створити зручний і візуально впорядкований інтерфейс
JavaScript	Клієнтська логіка	Використовується для перевірки введення, обробки дій користувача та динамічної роботи сторінок
Node.js	Серверне середовище виконання	Дозволяє реалізувати серверну частину вебзастосунку на JavaScript
Express.js	Маршрутизація та обробка запитів	Спрощує побудову API, обробку форм, авторизацію та взаємодію з клієнтом
SQL	Робота з реляційними даними	Забезпечує вибірку, фільтрацію, групування та оновлення даних відвідування
UniversityAttendanceDB	Зберігання даних системи	Містить структуру для обліку груп, студентів, викладачів, дисциплін, занять і відвідування
CSV	Експорт звітів	Дозволяє вивантажувати результати звітності у формат, придатний для подальшого використання
HTTPS	Захищене передавання даних	Підвищує безпеку обміну інформацією між браузером і сервером

Обраний набір технологій відповідає функціональним вимогам системи та не ускладнює її впровадження. Він дозволяє реалізувати авторизацію, реєстрацію користувачів, рольовий доступ, ведення довідників, створення

занять, фіксацію відвідування, формування звітів і перегляд аналітики. При цьому система залишається достатньо простою для розгортання в навчальному середовищі та може працювати як у локальній мережі університету, так і на віддаленому сервері.

Вибір HTML, CSS, JavaScript, Node.js, Express.js і реляційної бази даних UniversityAttendanceDB є технічно обґрунтованим для реалізації інформаційної системи аналізу відвідування і занять студентами університету. Запропонований технологічний стек забезпечує зручну роботу користувачів, централізоване зберігання даних, розмежування прав доступу, формування звітності та можливість подальшого розвитку програмного забезпечення.

3.2 Представлення архітектури системи

Архітектура інформаційної системи аналізу відвідування і занять студентами університету побудована за клієнт-серверним принципом із розділенням на три основні рівні: клієнтський рівень, серверний рівень та рівень зберігання даних. Такий підхід забезпечує логічне відокремлення інтерфейсу користувача, прикладної логіки та бази даних, що підвищує зручність супроводу системи, спрощує її масштабування та дозволяє централізовано керувати доступом до навчальної інформації.

Загальну архітектуру системи подано на рисунку 3.1.

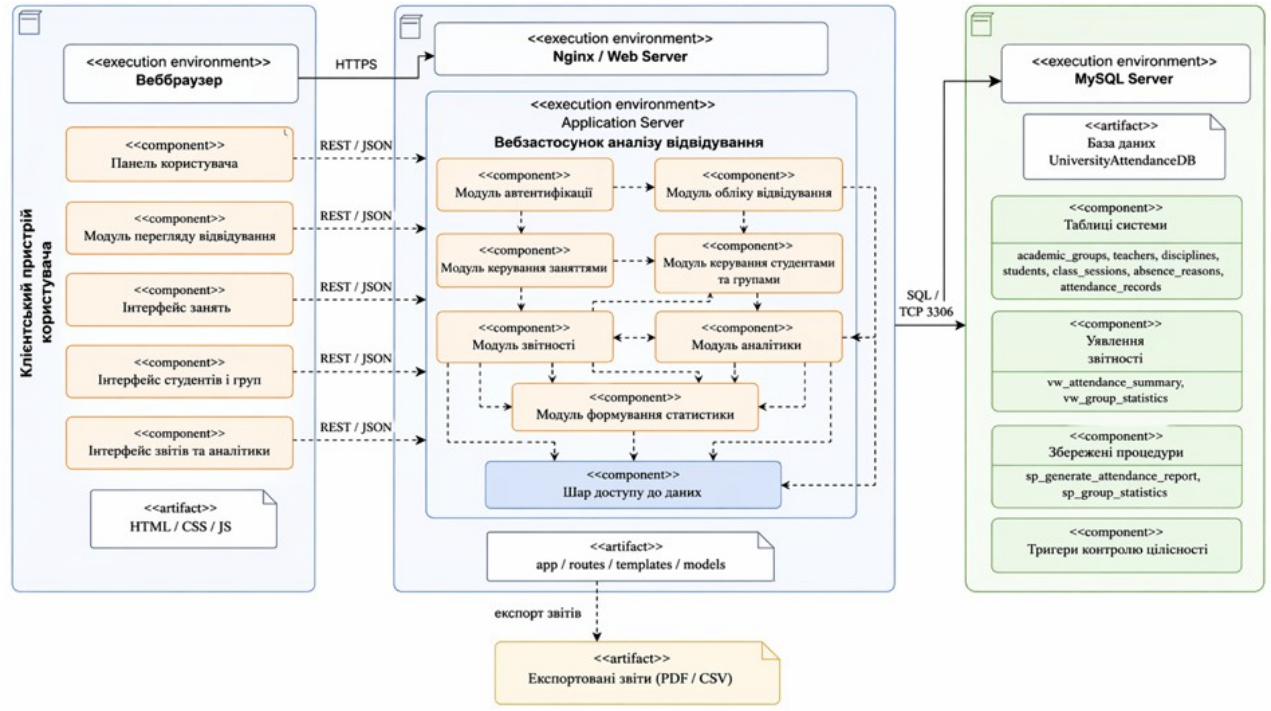

Рисунок 3.1 – Архітектура інформаційної системи аналізу відвідування і занять студентами університету

Клієнтський рівень призначений для взаємодії користувачів із системою через веббраузер. До складу клієнтської частини входять панель користувача, модуль перегляду відвідування, інтерфейс занять, інтерфейс студентів і груп, а також інтерфейс звітів та аналітики. Усі ці компоненти реалізуються засобами HTML, CSS і JavaScript та забезпечують введення даних, перегляд інформації, роботу з журналами відвідування і формування запитів до серверної частини. Передавання даних між клієнтом і сервером виконується через HTTPS у форматі REST/JSON, що забезпечує стандартизований обмін інформацією між вебінтерфейсом і прикладною логікою системи.

Серверний рівень реалізує основну бізнес-логіку інформаційної системи. На цьому рівні працює вебзастосунок Attendance Web App, розміщений у середовищі вебсервера Nginx. Серверна частина приймає запити від клієнта, виконує перевірку користувача, обробляє дані про заняття, студентів, групи та відвідування, формує статистичні показники і передає результати назад до клієнтської частини. У структурі серверного рівня виділено модуль автентифікації, модуль обліку відвідування, модуль керування заняттями,

модуль керування студентами та групами, модуль звітності, модуль аналітики, модуль формування статистики та шар доступу до даних.

Модуль автентифікації відповідає за вхід користувачів до системи, перевірку облікових даних і визначення ролі користувача. Це дозволяє розмежувати функціональні можливості для адміністратора, працівника деканату, викладача та студента. Адміністратор отримує доступ до керування користувачами і довідниками, викладач працює із власними заняттями та журналами, працівник деканату формує зведені звіти, а студент переглядає лише власні дані про відвідування.

Модуль обліку відвідування забезпечує створення та оновлення записів присутності студентів на заняттях. Він обробляє статуси відвідування, причини відсутності, час внесення запису та примітки викладача. Модуль керування заняттями відповідає за створення навчальних подій із прив’язкою до академічної групи, дисципліни, викладача, дати, часу та теми заняття. Модуль керування студентами та групами використовується для роботи з довідковою інформацією про академічні групи, студентів і їхню належність до відповідного навчального підрозділу.

Модуль звітності та модуль аналітики призначені для оброблення накопичених даних і формування підсумкової інформації. Вони дозволяють отримувати звіти за групами, дисциплінами, викладачами, студентами та періодами. Модуль формування статистики виконує узагальнення показників відвідуваності, визначає кількість занять, пропусків, запізнень, присутностей і формує аналітичні підсумки для подальшого перегляду або експорту. Для збереження результатів передбачено формування експортованих звітів у форматах PDF або CSV.

Рівень даних представлений сервером MySQL і базою даних UniversityAttendanceDB. Зв’язок між серверною частиною та базою даних здійснюється через SQL/TCP 3306. У базі зберігаються таблиці academic_groups, students, teachers, disciplines, class_sessions, attendance_records, absence_reasons та інші службові структури. Також на рівні бази даних можуть використовуватися уявлення звітності, зокрема vw_attendance_summary і

vw_group_statistics, збережені процедури для формування звітів та тригери контролю цілісності. Така організація дозволяє підтримувати достовірність даних і зменшувати ризик некоректного внесення інформації.

Таблиця 3.2 – Основні рівні архітектури системи

Рівень архітектури	Основні компоненти	Призначення
Клієнтський рівень	Веббраузер, панель користувача, інтерфейси занять, студентів, груп, звітів та аналітики	Забезпечує взаємодію користувача із системою, введення даних і перегляд результатів
Серверний рівень	Nginx, Attendance Web App, модулі автентифікації, занять, відвідування, звітності та аналітики	Реалізує бізнес-логіку, обробку запитів, перевірку прав доступу та формування відповідей
Рівень даних	MySQL Server, UniversityAttendanceDB, таблиці, уявлення, процедури, тригери	Забезпечує централізоване зберігання, цілісність, вибірку та оброблення навчальних даних

Продовження таблиці 3.2

Рівень експорту	Експортовані звіти PDF/CSV	Дає змогу зберігати та передавати результати звітності за межі системи

Важливою особливістю запропонованої архітектури є рольове розмежування доступу. Користувач не працює безпосередньо з базою даних, а взаємодіє з нею лише через серверну частину. Це дозволяє контролювати права доступу, перевіряти коректність запитів і запобігати несанкціонованій зміні записів. Наприклад, студент може переглядати тільки власну відвідуваність, викладач має доступ до журналів своїх занять, працівник деканату переглядає зведену інформацію, а адміністратор керує системними довідниками та обліковими записами.

Використання окремого шару доступу до даних спрощує супровід програмної системи. Усі запити до UniversityAttendanceDB проходять через спеціалізований компонент, тому зміна структури таблиць або додавання нових

звітів не потребує повної переробки клієнтської частини. Це також дає можливість централізовано реалізувати перевірку даних, оброблення помилок і підготовку результатів для модулів звітності та аналітики.

Запропонована архітектура відповідає вимогам до навчальної інформаційної системи, оскільки забезпечує централізоване зберігання даних, доступ через браузер, розмежування прав користувачів, формування звітів і можливість подальшого розширення. За потреби до системи можна додати нові модулі, наприклад журнал змін, повідомлення студентам, інтеграцію з електронним розкладом або додаткові аналітичні панелі без порушення основної структури застосунку.

Отже, архітектура інформаційної системи побудована як модульний вебзастосунок із чітким поділом на клієнтську частину, серверну бізнес-логіку та рівень бази даних. Такий підхід забезпечує стабільну роботу системи, зручність використання для різних категорій користувачів, захист навчальної інформації та технічну основу для подальшого розвитку програмного забезпечення.

3.3 Висновки до третього розділу

У третьому розділі виконано обґрунтування технологічних засобів і представлено архітектуру інформаційної системи аналізу відвідування і занять студентами університету. Для реалізації системи обрано веборієнтований підхід, що забезпечує доступ користувачів через браузер без потреби встановлення окремого клієнтського програмного забезпечення. Такий варіант є доцільним для університетського середовища, оскільки системою одночасно можуть користуватися викладачі, студенти, працівники деканату та адміністратор.

Обґрунтовано вибір технологій HTML, CSS і JavaScript для створення клієнтської частини, а також Node.js та Express.js для реалізації серверної логіки. Такий стек дозволяє розробити зручний вебінтерфейс, організувати обробку запитів, реалізувати авторизацію, рольовий доступ, роботу з

журналами відвідування, формування звітів і експорт даних. Для зберігання інформації використовується реляційна база даних UniversityAttendanceDB, структура якої відповідає предметній області та забезпечує коректне збереження даних про групи, студентів, викладачів, дисципліни, заняття й записи відвідування.

У розділі представлено архітектуру системи, побудовану за клієнт-серверним принципом із поділом на клієнтський рівень, серверний рівень і рівень даних. Клієнтська частина відповідає за взаємодію користувача із системою, серверна частина реалізує основну бізнес-логіку, а рівень даних забезпечує централізоване зберігання та оброблення навчальної інформації. Такий поділ підвищує надійність системи, спрощує її супровід і створює умови для подальшого розширення функціональних можливостей.

Окрему увагу приділено рольовому розмежуванню доступу. У системі передбачено різні сценарії роботи для адміністратора, працівника деканату, викладача та студента. Це дозволяє обмежити доступ користувачів лише до тих функцій і даних, які відповідають їхнім повноваженням. Завдяки цьому підвищується безпека інформації, зменшується ризик помилкового або несанкціонованого редагування записів і забезпечується контрольоване ведення журналу відвідування.

Отже, у третьому розділі сформовано технічну основу програмної реалізації системи. Обрані технології та запропонована архітектура забезпечують стабільну роботу вебзастосунку, централізоване зберігання даних, підтримку звітності, аналітики, експорту результатів і подальшого розвитку інформаційної системи відповідно до потреб навчального підрозділу.

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

4.1 План тестування програмних модулів

Тестування інформаційної системи аналізу відвідування і занять студентами університету є необхідним етапом перевірки правильності роботи програмних модулів, коректності взаємодії з базою даних UniversityAttendanceDB та відповідності реалізованого функціоналу сформованим вимогам. Основна мета тестування полягає у виявленні помилок у роботі системи до її впровадження в навчальний процес, перевірці стабільності основних сценаріїв і підтвердженні можливості щоденного використання системи викладачами, студентами, працівниками деканату та адміністратором.

Об’єктом тестування є вебзастосунок Attendance Web App, що включає клієнтський інтерфейс, серверну логіку, модулі автентифікації, керування користувачами, обліку занять, фіксації відвідування, формування звітів, аналітики та експорту даних. Окремо перевіряється взаємодія системи з базою даних, оскільки саме в ній зберігаються основні навчальні та операційні дані: академічні групи, студенти, викладачі, дисципліни, заняття, статуси відвідування та причини відсутності.

План тестування сформовано з урахуванням модульної структури системи. Для кожного програмного модуля визначаються тестові дії, очікуваний результат і критерій успішності. Такий підхід дає змогу перевірити систему не лише як єдиний програмний продукт, а й окремо оцінити працездатність її основних функціональних частин.

Таблиця 4.1 – План тестування програмних модулів інформаційної системи

Модуль системи	Мета тестування	Тестові дії	Очікуваний результат
Модуль автентифікації	Перевірка входу користувачів до системи	Ввести коректний логін і пароль для ролей адміністратора, деканату, викладача та студента	Користувач успішно входить до системи та потрапляє до відповідного рольового кабінету
Модуль реєстрації	Перевірка створення нового облікового запису	Заповнити форму реєстрації з коректними даними	Обліковий запис створюється, дані зберігаються, користувач може пройти авторизацію
Модуль перевірки ролей	Контроль розмежування доступу	Увійти під різними ролями та перевірити доступні розділи меню	Кожна роль бачить лише дозволені функції системи
Модуль керування студентами	Перевірка роботи зі списками студентів	Додати, змінити та переглянути дані студента	Дані коректно зберігаються і відображаються у списку студентів
Модуль академічних груп	Перевірка довідника груп	Створити групу, змінити її назву, курс і факультет	Група доступна для прив’язки студентів і занять
Модуль дисциплін	Перевірка довідника навчальних дисциплін	Додати дисципліну, вказати кредити та форму контролю	Дисципліна зберігається та доступна під час створення заняття
Модуль занять	Перевірка створення навчальної події	Створити заняття із зазначенням групи, дисципліни, викладача, дати, часу й аудиторії	Заняття з’являється у журналі та доступне для фіксації відвідування
Модуль обліку	Перевірка	Відкрити заняття,	Записи відвідування

відвідування	фіксації присутності студентів	встановити студентам статуси присутній, відсутній, запізнився	зберігаються без дублювання та відображаються в журналі
Модуль причин відсутності	Перевірка обліку поважних і неповажних причин	Для відсутнього студента вибрати причину пропуску та додати примітку	Причина відсутності зберігається разом із записом відвідування

Продовження таблиці 4.1

Модуль звітності	Перевірка формування звітів	Сформувати звіт за групою, дисципліною, студентом і періодом	Система виводить коректну таблицю з підсумковими показниками
Модуль аналітики	Перевірка розрахунку статистичних показників	Переглянути загальну відвідуваність, кількість пропусків і запізнень	Показники відповідають записам у журналі відвідування
Модуль експорту	Перевірка вивантаження результатів	Виконати експорт сформованого звіту у CSV-файл	Файл створюється та містить дані звіту у придатному для перегляду форматі
Модуль доступу до БД	Перевірка коректності SQL-запитів	Виконати додавання, редагування, вибірку та видалення тестових записів	Дані коректно записуються, оновлюються та зчитуються з UniversityAttendanceDB

Під час тестування особлива увага приділяється модулю автентифікації та рольового доступу, оскільки система працює з навчальними даними, які не

повинні бути доступні всім користувачам у повному обсязі. Студент має переглядати лише власну відвідуваність, викладач - працювати із заняттями, які він проводить, працівник деканату - переглядати зведені звіти, а адміністратор - керувати користувачами, довідниками та системними параметрами.

Для перевірки коректності роботи системи використовуються функціональне, інтеграційне та інтерфейсне тестування. Функціональне тестування спрямоване на перевірку окремих можливостей системи: входу, реєстрації, створення занять, внесення відвідування та формування звітів. Інтеграційне тестування дає змогу перевірити взаємодію клієнтської частини, серверної логіки та бази даних. Інтерфейсне тестування використовується для перевірки зручності форм, таблиць, кнопок, повідомлень і навігації між розділами.

Таблиця 4.2 – Види тестування та їх призначення

Вид тестування	Призначення	Приклад перевірки
Функціональне тестування	Перевірка виконання основних функцій системи	Створення заняття, внесення статусів відвідування, формування звіту
Інтеграційне тестування	Перевірка взаємодії між модулями	Передавання даних від вебінтерфейсу до серверної частини та бази даних
Тестування доступу	Перевірка рольових обмежень	Спроба студента відкрити адміністративний розділ
Тестування введення даних	Перевірка реакції системи на помилкові або неповні дані	Створення заняття без дати або групи
Тестування бази даних	Перевірка цілісності та коректності збереження даних	Заборона дублювання запису відвідування одного студента на одному занятті
Інтерфейсне тестування	Перевірка зручності роботи користувача	Перегляд таблиць, фільтрів, повідомлень і форм введення

Тестування експорту	Перевірка формування зовнішніх файлів	Вивантаження звіту у CSV та відкриття його в табличному редакторі

Критерієм успішного проходження тестування є коректне виконання всіх основних сценаріїв без критичних помилок, правильне збереження даних у базі, відсутність дублювання записів, відповідність функцій ролям користувачів і стабільне формування звітів. Якщо під час тестування виявляються помилки, вони фіксуються, після чого виконується виправлення програмного модуля та повторна перевірка відповідного сценарію.

Таблиця 4.3 – Критерії успішності тестування

Критерій	Очікуване значення
Авторизація користувачів	Користувач входить до системи лише з правильними обліковими даними
Рольовий доступ	Кожна роль має доступ тільки до визначених функцій
Збереження даних	Дані після створення або редагування зберігаються в базі UniversityAttendanceDB
Коректність журналу	Для одного студента в межах одного заняття створюється лише один запис відвідування
Формування звітів	Звіти відображають дані відповідно до заданих параметрів відбору
Стабільність інтерфейсу	Форми, таблиці та кнопки працюють без збоїв у сучасних браузерах

Отже, план тестування програмних модулів охоплює основні функціональні частини інформаційної системи та дозволяє перевірити її готовність до практичного використання. Запропонована методика тестування спрямована на підтвердження коректності авторизації, рольового доступу, роботи з навчальними даними, фіксації відвідування, формування звітності та взаємодії з базою даних. Це забезпечує надійну основу для подальшого тестування інтерфейсу та оцінювання ефективності роботи системи.

4.2 Тестування інтерфейсних модулів

Після перевірки окремих програмних модулів було виконано тестування інформаційної системи в цілому. Метою цього етапу є підтвердження працездатності основних сценаріїв використання: входу до системи, розмежування доступу за ролями, перегляду занять, ведення журналу відвідування, перегляду студентів груп, формування звітної інформації та відображення персональної статистики студента.

Тестування проводилося для чотирьох основних ролей користувачів: адміністратора, працівника деканату, викладача та студента. Основну увагу приділено тому, щоб кожен користувач мав доступ лише до тих функцій, які відповідають його повноваженням. Це важливо для системи обліку відвідування, оскільки викладач повинен працювати лише зі своїми заняттями, студент має переглядати тільки власні записи, а деканат і адміністратор повинні мати доступ до зведеної інформації та керування довідниками.

Першим етапом було перевірено роботу сторінки входу до системи. Користувач вводить логін і пароль, після чого система перевіряє облікові дані та визначає роль користувача. У разі успішної авторизації відкривається відповідний робочий кабінет. Якщо дані введено неправильно, доступ до системи не надається. Вікно входу до системи наведено на рисунку 4.1.

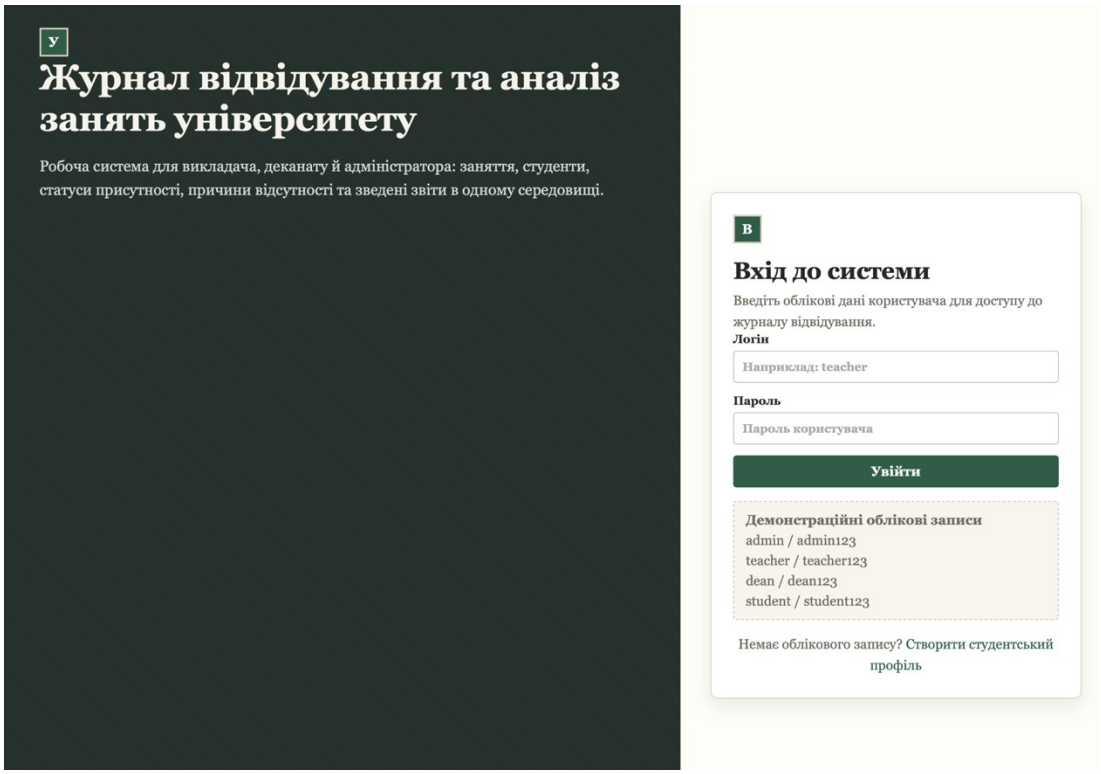

Рисунок 4.1 – Вікно входу до інформаційної системи

Під час тестування входу було перевірено демонстраційні облікові записи адміністратора, викладача, працівника деканату та студента. Система коректно розпізнавала роль користувача і відкривала різні набори функцій. Це підтверджує правильність роботи модуля автентифікації та механізму рольового доступу.

Окремо перевірено робоче місце викладача. Після входу під роллю викладача відкривається головна сторінка з короткими показниками: кількість доступних груп, студентів, дисциплін, занять і середня присутність. Також відображаються останні заняття в журналі та доступні переходи до розділів «Мої заняття», «Студенти моїх груп», «Звіт групи», «Пропуски» й «Аналітика». Робоче місце викладача наведено на рисунку 4.2.

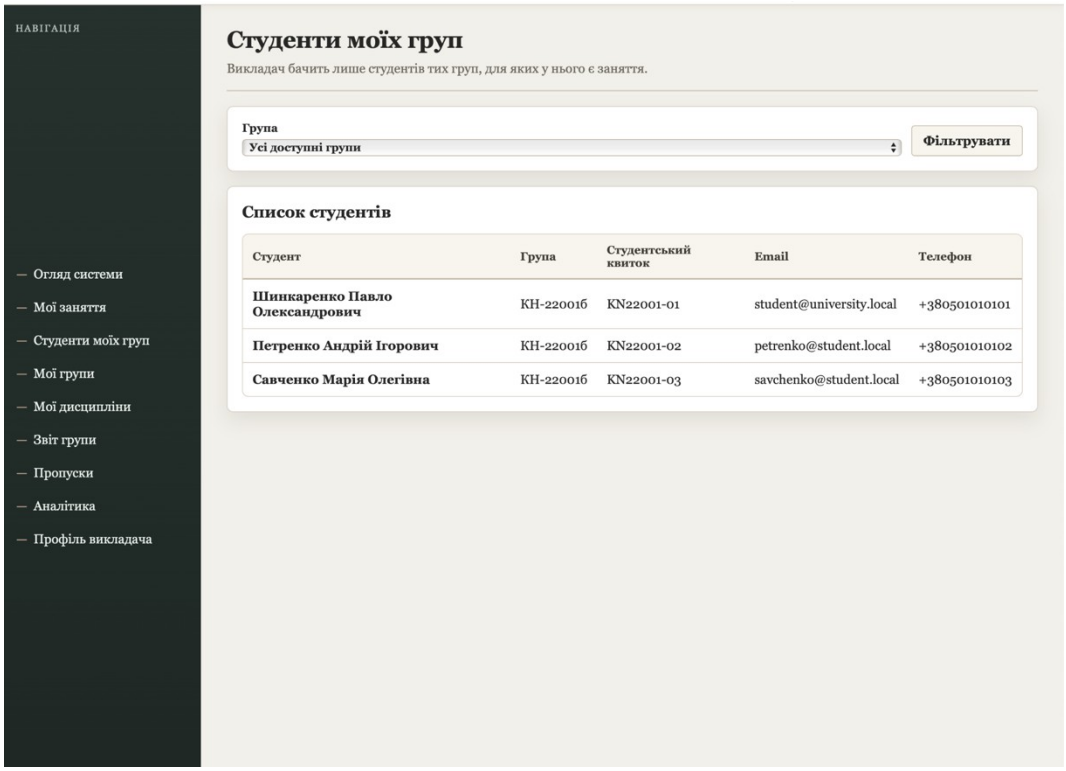

Рисунок 4.2 – Робоче місце викладача в інформаційній системі

Перевірка показала, що викладач бачить лише ті дані, які пов’язані з його заняттями та закріпленими групами. Такий результат відповідає вимогам до розмежування доступу та запобігає перегляду або редагуванню сторонньої навчальної інформації.

Наступним етапом перевірено розділ «Мої заняття». У цьому розділі викладач може переглядати заняття, фільтрувати їх за групою, дисципліною та періодом, а також створювати нове заняття із зазначенням групи, дисципліни, типу заняття, дати, часу, аудиторії та теми. Після створення заняття воно з’являється у загальному переліку та стає доступним для заповнення журналу. Сторінку роботи із заняттями наведено на рисунку 4.3.

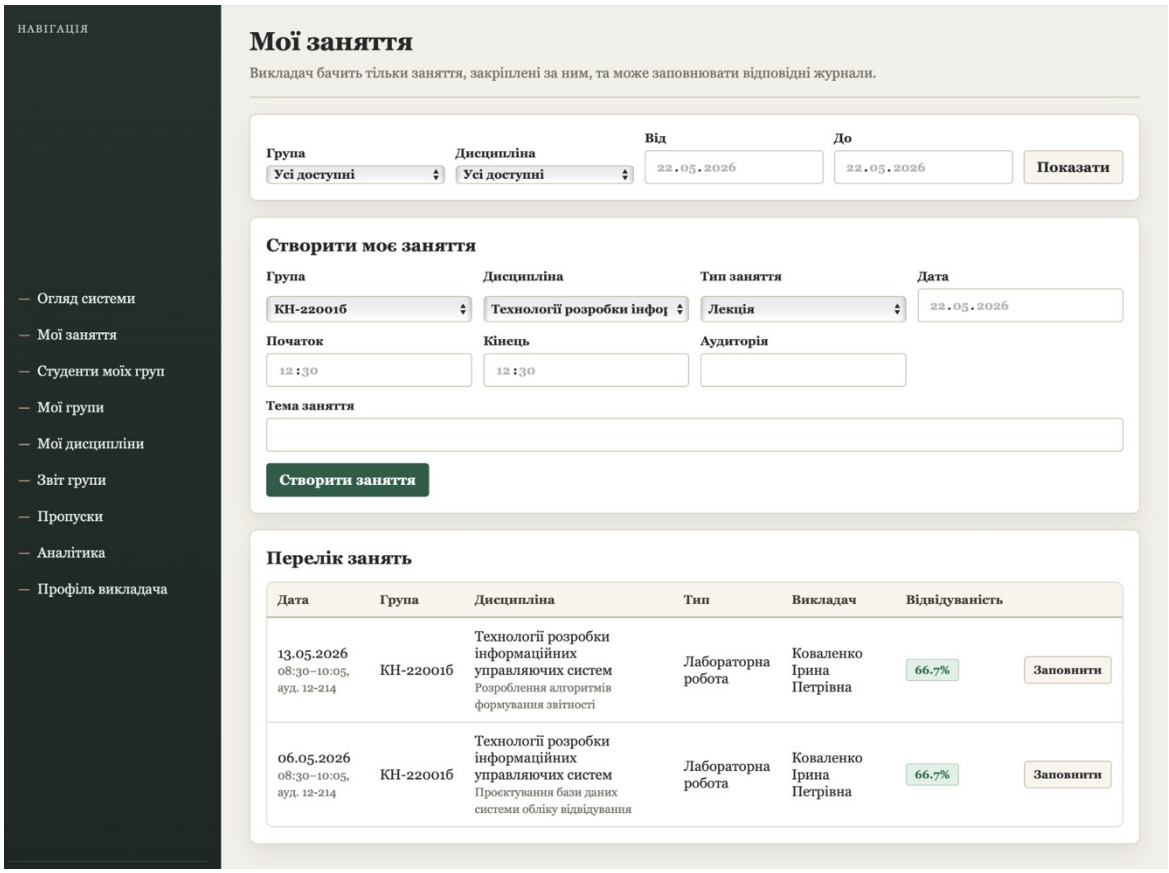

Рисунок 4.3 – Розділ створення та перегляду занять викладача

У процесі тестування було перевірено введення коректних і неповних даних. Якщо всі обов’язкові поля заповнені, система створює запис заняття. Якщо користувач не вказує необхідні параметри, система не повинна створювати некоректний запис. Така перевірка підтверджує правильність роботи форми створення заняття та логіки контролю введених даних.

Також було протестовано перегляд студентів закріплених груп. У цьому розділі викладач бачить список студентів лише тих груп, для яких у нього є заняття. Для кожного студента відображаються ПІБ, академічна група, номер студентського квитка, електронна пошта та телефон. Сторінку перегляду студентів наведено на рисунку 4.4.

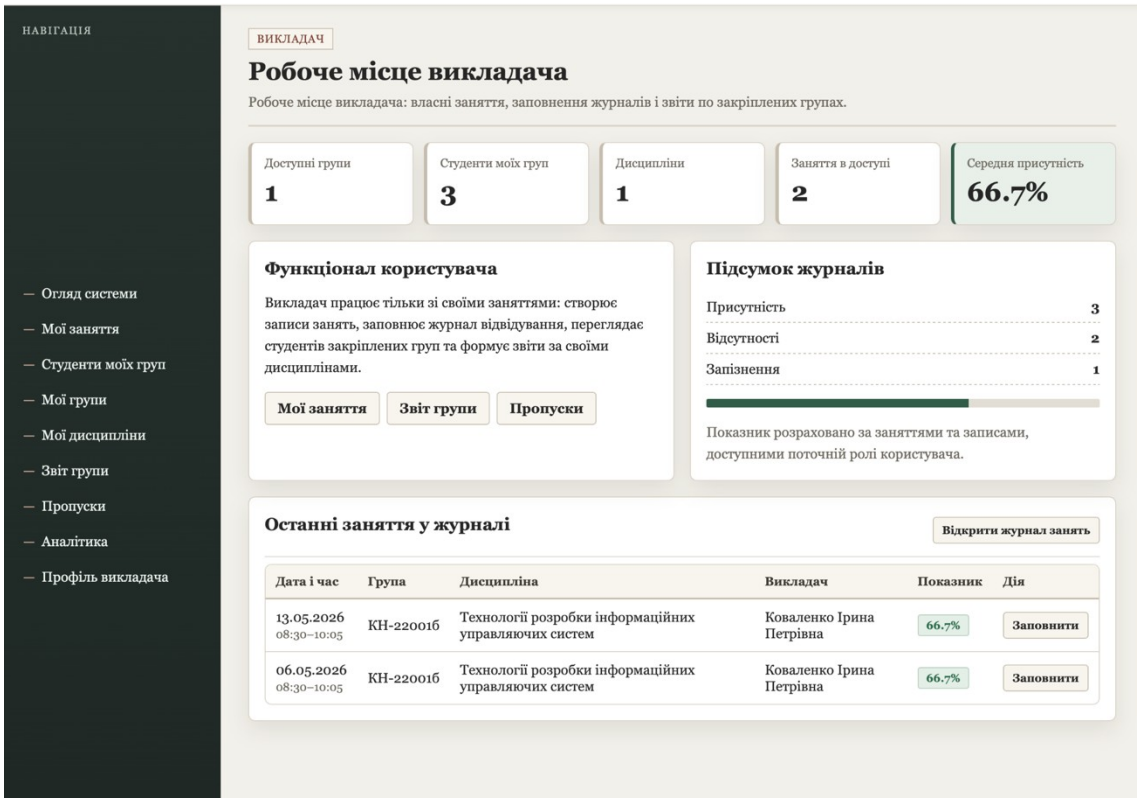

Рисунок 4.4 – Перегляд студентів закріплених груп викладача

Результати перевірки показали, що фільтрація студентів за групами працює коректно. Викладач не отримує доступ до повного списку студентів університету, а працює лише з тими здобувачами, які пов’язані з його навчальними заняттями. Це відповідає принципу мінімально необхідного доступу.

Для студентської ролі було перевірено сторінку розкладу занять. Студент може переглядати заняття своєї академічної групи, використовувати фільтри за групою, дисципліною та датами, а також бачити власну відмітку відвідування. При цьому студент не має доступу до редагування журналу або зміни статусів присутності. Приклад сторінки розкладу занять студента наведено на рисунку 4.5.

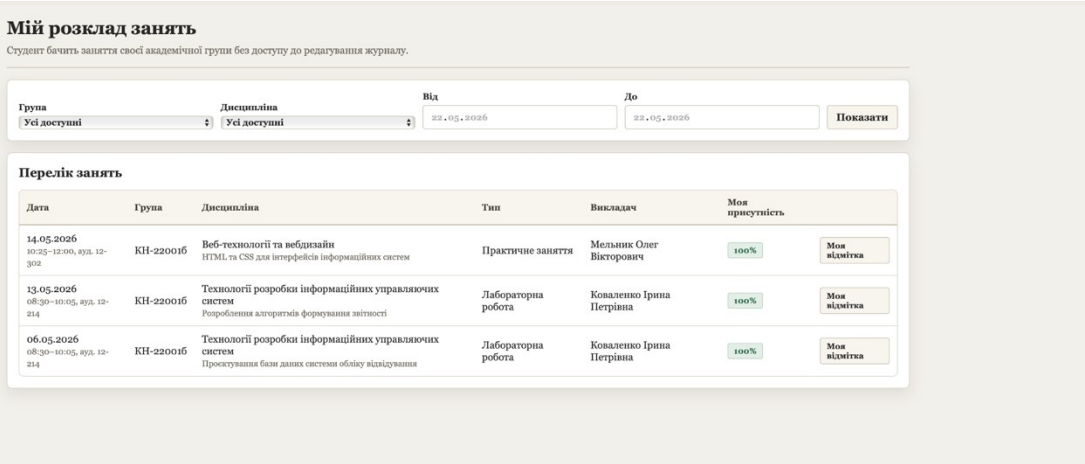

Рисунок 4.5 – Перегляд розкладу занять студентом

Під час тестування підтверджено, що студентський кабінет працює у режимі перегляду. Користувач бачить дату заняття, час, аудиторію, дисципліну, тип заняття, викладача та власний статус присутності. Відсутність кнопок редагування журналу підтверджує коректне обмеження прав для студентської ролі.

Додатково було протестовано сторінку профілю студента. На ній відображаються персональні дані користувача, роль, електронна пошта, а також підсумкові показники відвідування: загальна кількість відміток, кількість присутностей, пропусків і загальний відсоток відвідуваності. Нижче подано таблицю з персональними записами відвідування за дисциплінами. Сторінку профілю користувача наведено на рисунку 4.6.

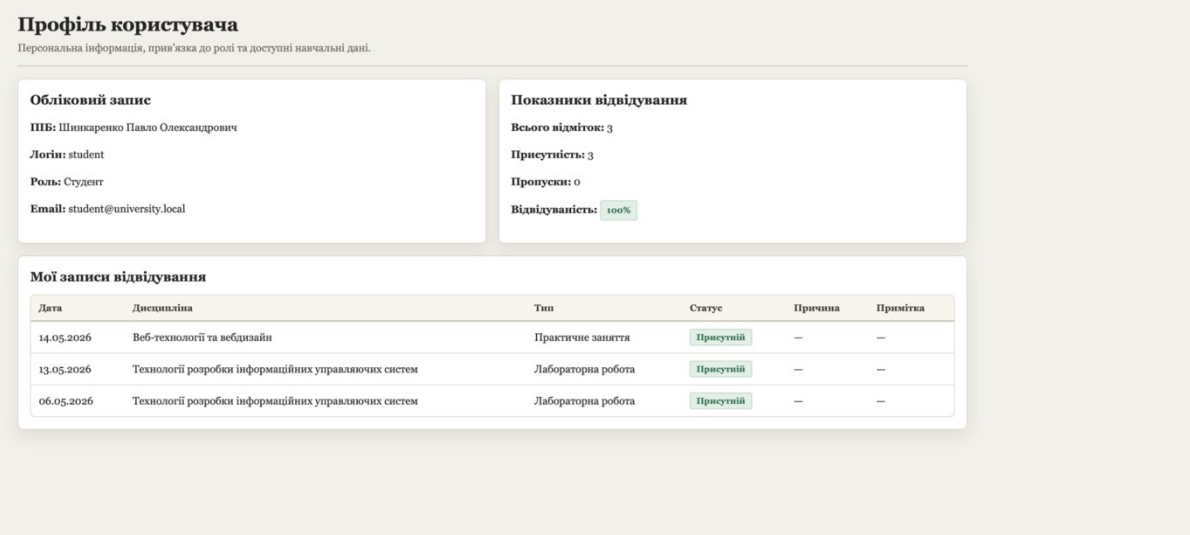

Рисунок 4.6 – Профіль студента та персональні показники відвідування

За результатами перевірки встановлено, що система правильно формує персональну статистику студента на основі записів журналу. Дані у профілі відповідають записам відвідування, що підтверджує коректність зв’язку між заняттями, студентами та статусами присутності.

Результати тестування основних сценаріїв роботи системи наведено в таблиці 4.4.

Таблиця 4.4 – Результати тестування основних сценаріїв роботи системи

Сценарій тестування	Дії користувача	Очікуваний результат	Результат
Вхід до системи	Введення логіна і пароля користувача	Відкривається кабінет відповідної ролі	Виконано
Перевірка неправильного входу	Введення некоректних облікових даних	Доступ до системи не надається	Виконано
Перегляд кабінету викладача	Авторизація під роллю викладача	Відображаються дані лише за закріпленими заняттями	Виконано
Створення заняття	Заповнення форми заняття викладачем	Нове заняття додається до переліку	Виконано
Перегляд студентів груп	Перехід до розділу студентів викладача	Відображаються лише студенти доступних груп	Виконано
Перегляд розкладу студентом	Авторизація під роллю студента	Студент бачить заняття своєї групи без права редагування	Виконано
Перегляд профілю студента	Відкриття персонального профілю	Відображаються особисті дані та статистика відвідування	Виконано
Перевірка рольового доступу	Спроба відкрити недоступні розділи	Система обмежує доступ до заборонених функцій	Виконано
Формування підсумкових показників	Перегляд статистики за журналом	Показники відповідають записам відвідування	Виконано

У процесі тестування також перевірено зручність інтерфейсу. Сторінки системи мають єдину структуру: ліву навігаційну панель, заголовок розділу, фільтри, таблиці даних і кнопки виконання основних дій. Такий підхід спрощує роботу користувача, оскільки всі розділи побудовані за однаковою логікою. Інтерфейс не перевантажений зайвими елементами, а таблиці містять лише ті поля, які потрібні для конкретного сценарію.

Окремо перевірено коректність відображення даних у таблицях. У списку занять правильно відображаються дата, група, дисципліна, тип заняття, викладач і показник відвідуваності. У списку студентів коректно виводяться персональні та навчальні дані. У профілі студента правильно відображаються статуси відвідування та підсумкові показники. Це підтверджує правильність вибірки даних із бази та коректну роботу клієнтської частини.

За результатами тестування встановлено, що основні модулі інформаційної системи працюють відповідно до визначених вимог. Авторизація, рольовий доступ, перегляд занять, робота з даними студентів, студентський кабінет і персональна статистика функціонують коректно. Виявлені під час перевірки незначні недоліки інтерфейсу можуть бути усунені на етапі подальшого вдосконалення, однак вони не впливають на виконання основних функцій системи.

Отже, проведене тестування підтвердило працездатність інформаційної системи аналізу відвідування і занять студентами університету. Система забезпечує коректну роботу з різними ролями користувачів, підтримує перегляд і фіксацію навчальних даних, формує персональні та групові показники відвідування і може бути використана як основа для подальшого впровадження в навчальному підрозділі.

4.3 Розгортання системи, склад інсталяційного пакету

Розгортання інформаційної системи аналізу відвідування і занять студентами університету передбачає підготовку клієнтського середовища, серверної частини вебзастосунку та бази даних UniversityAttendanceDB.

Система реалізована як веборієнтований застосунок, тому користувачі працюють із нею через браузер, а основна обробка даних виконується на сервері. Такий підхід дозволяє централізовано зберігати навчальні дані, контролювати доступ користувачів і не встановлювати окрему програму на кожен комп’ютер.

Схему розгортання інформаційної системи подано на рисунку 4.7.

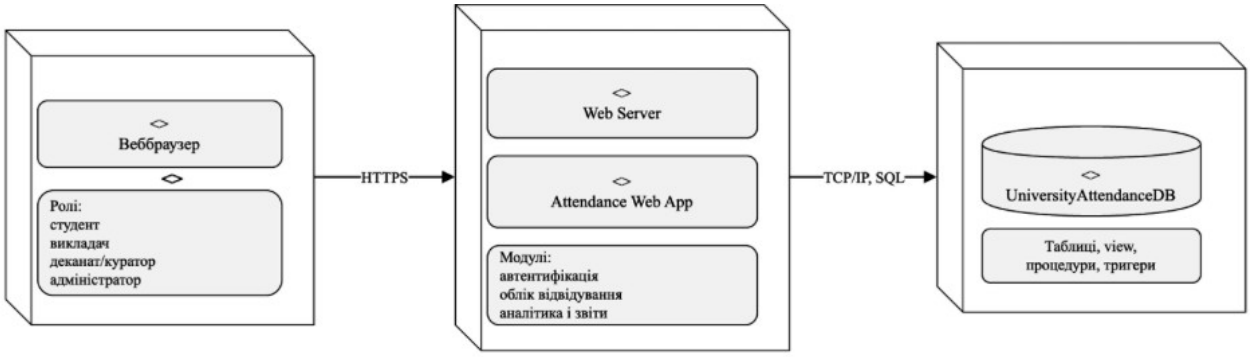

Рисунок 4.7 – Діаграма розгортання інформаційної системи аналізу відвідування і занять студентами університету

Відповідно до діаграми, клієнтський рівень представлений робочими пристроями користувачів, на яких запускається веббраузер. Через браузер із системою працюють студент, викладач, працівник деканату або адміністратор. Передавання даних між клієнтом і сервером виконується через захищений протокол HTTPS. Це дозволяє забезпечити безпечний обмін обліковими даними, параметрами запитів, записами відвідування та результатами звітності.

На серверному вузлі розміщується вебсервер і застосунок Attendance Web App. Вебсервер приймає HTTP/HTTPS-запити від клієнтів і передає їх до серверної частини системи. Attendance Web App реалізує основні програмні модулі: автентифікацію користувачів, облік відвідування, роботу із заняттями, формування звітів та аналітичну обробку даних. Саме серверна частина визначає, які функції доступні конкретному користувачу відповідно до його ролі.

Рівень даних представлений сервером бази даних UniversityAttendanceDB. З’єднання між вебзастосунком і базою даних виконується через TCP/IP із використанням SQL-запитів. У базі зберігаються таблиці студентів, академічних груп, викладачів, дисциплін, занять, причин відсутності та записів

відвідування. Клієнтський браузер не має прямого доступу до бази даних, що підвищує безпеку системи та дозволяє контролювати всі операції через серверну логіку.

Для коректної роботи системи необхідно підготувати серверне середовище. На сервері встановлюється Node.js, менеджер пакетів npm, вебсервер Nginx або Apache, а також сервер бази даних MySQL. Після цього створюється база UniversityAttendanceDB, виконуються SQL-скрипти створення таблиць, ключів, обмежень та початкових довідкових даних. Далі розміщується серверна частина застосунку, встановлюються залежності та запускається вебсервер.

Таблиця 4.5 – Склад середовища розгортання системи

Компонент	Призначення
Клієнтський пристрій	Забезпечує доступ користувача до системи через браузер
Веббраузер	Відображає інтерфейс системи та передає запити до сервера
Вебсервер	Приймає запити клієнтів і забезпечує доступ до вебзастосунку
Attendance Web App	Реалізує авторизацію, рольовий доступ, облік занять, відвідування, звіти й аналітику
MySQL Server	Забезпечує роботу реляційної бази даних
UniversityAttendanceDB	Зберігає навчальні, довідкові та операційні дані системи
HTTPS	Забезпечує захищене передавання даних між клієнтом і сервером
SQL/TCP/IP	Забезпечує обмін даними між серверною частиною і базою даних

Порядок розгортання системи включає кілька основних етапів. Спочатку виконується підготовка серверного вузла: встановлення операційної системи, Node.js, npm, MySQL та вебсервера. Після цього створюється база даних UniversityAttendanceDB і завантажується структура таблиць. Далі на сервер копіюються файли вебзастосунку, встановлюються програмні залежності та

задаються параметри підключення до бази даних. На завершальному етапі виконується запуск системи, перевірка доступності вебінтерфейсу та тестування входу під різними ролями користувачів.

Таблиця 4.6 – Послідовність розгортання інформаційної системи

Етап	Зміст робіт	Очікуваний результат
Підготовка сервера	Встановлення Node.js, npm, MySQL, Nginx або Apache	Сервер готовий до запуску вебзастосунку
Створення бази даних	Виконання SQL-скриптів для UniversityAttendanceDB	Створено таблиці, ключі, зв’язки й обмеження
Розміщення застосунку	Копіювання файлів Attendance Web App на сервер	Файли системи розміщені в робочому каталозі
Встановлення залежностей	Виконання команди встановлення пакетів	Підключено необхідні програмні бібліотеки

Продовження таблиці 4.6

Налаштування підключення	Задання параметрів доступу до бази даних	Серверна частина може працювати з UniversityAttendanceDB
Запуск системи	Старт вебзастосунку та вебсервера	Система доступна користувачам через браузер
Контрольна перевірка	Вхід під ролями адміністратора, викладача, деканату й студента	Підтверджено працездатність основних функцій

Інсталяційний пакет системи повинен містити всі файли, необхідні для встановлення, запуску й початкового налаштування програмного продукту. До його складу входять файли клієнтської частини, серверна логіка, SQL-скрипти бази даних, конфігураційні файли, статичні ресурси, документація користувача та інструкція адміністратора.

Таблиця 4.7 – Склад інсталяційного пакету інформаційної системи

Елемент пакету	Призначення
app / server	Серверна частина вебзастосунку, маршрути, контролери та бізнес-логіка

public / assets	Статичні файли інтерфейсу, стилі, скрипти, зображення та ресурси сторінок
views / templates	Шаблони сторінок інтерфейсу користувача
database / schema.sql	SQL-скрипт створення структури бази даних
database / seed.sql	Початкові дані для демонстраційної роботи системи
config	Налаштування порту сервера, параметрів підключення до БД та службових змінних
package.json	Перелік залежностей і команд запуску Node.js-застосунку
README.md	Коротка інструкція зі встановлення та запуску системи
docs	Опис ролей користувачів, структури системи й порядку роботи
exports	Каталог для збереження сформованих звітів у CSV або PDF

Для запуску системи адміністратор переходить до каталогу проєкту, встановлює залежності та запускає серверну частину. Після старту застосунок стає доступним у браузері за локальною або мережевою адресою сервера. У разі розгортання в локальній мережі університету користувачі можуть підключатися до системи з робочих комп'ютерів кафедри, деканату або комп'ютерних класів. У разі розміщення на віддаленому сервері доступ може бути організований через доменне ім'я та HTTPS-сертифікат.

Після розгортання необхідно виконати первинне налаштування системи: створити обліковий запис адміністратора, додати користувачів, налаштувати ролі, внести академічні групи, студентів, викладачів і дисципліни. Після цього викладачі можуть створювати заняття та заповнювати журнали, студенти — переглядати власну відвідуваність, а деканат — формувати звіти й аналізувати навчальну активність груп.

Для підтримання працездатності системи потрібно регулярно виконувати резервне копіювання бази даних UniversityAttendanceDB. Це дозволяє відновити інформацію у випадку помилки адміністратора, збою сервера або пошкодження даних. Також доцільно вести журнал дій користувачів, особливо

для операцій створення занять, зміни статусів відвідування та редагування довідників.

Отже, розгортання інформаційної системи передбачає підготовку серверного середовища, створення бази даних, встановлення вебзастосунку, налаштування доступу та перевірку основних ролей користувачів. Запропонований склад інсталяційного пакету забезпечує повний цикл встановлення системи та дає змогу використовувати її в локальній мережі університету або на окремому сервері для централізованого обліку відвідування і занять студентів.

4.4 Висновки до четвертого розділу

У четвертому розділі виконано тестування та описано розгортання інформаційної системи аналізу відвідування і занять студентами університету. На першому етапі сформовано план тестування програмних модулів, у якому визначено основні функціональні частини системи: модуль автентифікації, реєстрації, рольового доступу, керування студентами, групами, дисциплінами, заняттями, журналом відвідування, звітністю, аналітикою, експортом даних і взаємодією з базою UniversityAttendanceDB.

Під час тестування перевірено основні сценарії роботи системи для різних ролей користувачів. Встановлено, що адміністратор, викладач, працівник деканату та студент отримують доступ лише до тих функцій, які відповідають їхнім повноваженням. Це підтверджує коректність реалізації рольової моделі та захищає навчальні дані від несанкціонованого перегляду або редагування.

Окремо протестовано роботу інтерфейсних сторінок системи. Перевірено сторінку входу, робоче місце викладача, розділ створення та перегляду занять, список студентів закріплених груп, студентський розклад і профіль користувача. Результати тестування показали, що інтерфейс є зрозумілим, структурованим і придатним для щоденної роботи з журналом відвідування, фільтрами, таблицями та статистичними показниками.

Також перевірено коректність роботи системи з даними. Встановлено, що записи про заняття, студентів, статуси присутності, причини відсутності та підсумкові показники відображаються узгоджено. Система правильно формує персональну статистику студента, відображає заняття відповідної академічної групи та дозволяє викладачу працювати лише з власними журналами.

У розділі описано порядок розгортання системи та склад інсталяційного пакету. Визначено, що для роботи системи необхідні клієнтський пристрій із браузером, серверна частина Attendance Web App, вебсервер, середовище Node.js, СУБД MySQL і база даних UniversityAttendanceDB. Запропонована схема розгортання забезпечує централізоване зберігання даних, доступ через HTTPS, взаємодію серверної частини з базою даних через SQL/TCP та можливість використання системи в локальній мережі університету.

Отже, результати четвертого розділу підтверджують працездатність розробленої інформаційної системи. Проведене тестування показало, що основні функції авторизації, рольового доступу, ведення занять, обліку відвідування, перегляду студентських даних, формування статистики та підготовки звітів працюють коректно. Описаний порядок розгортання і склад інсталяційного пакету створюють практичну основу для впровадження системи в навчальному підрозділі та її подальшого супроводу.

ВИСНОВКИ

У кваліфікаційній роботі вирішено практичне завдання розроблення інформаційної системи аналізу відвідування і занять студентами університету. Запропонована система призначена для централізованого ведення навчальних занять, фіксації присутності студентів, обліку причин відсутності, формування звітів і аналітичної обробки даних про відвідуваність. Її використання дозволяє замінити розрізнений ручний або табличний облік єдиним програмним середовищем із рольовим доступом і структурованим зберіганням даних.

У першому розділі виконано аналіз предметної області та визначено основні процеси, пов’язані з організацією занять і контролем відвідування студентів. Розглянуто учасників системи: студента, викладача, працівника деканату та адміністратора. Проаналізовано існуючі підходи до обліку відвідування, зокрема паперові журнали, електронні таблиці, Moodle Attendance, Google Classroom і Microsoft Education Insights. Встановлено, що наявні рішення або не забезпечують повноцінної аналітики, або є надмірно загальними для задачі локального університетського обліку. Це обґрунтувало доцільність розроблення спеціалізованої інформаційної системи.

Також у першому розділі виконано UML-моделювання предметної області. Побудовано діаграму прецедентів, діаграму послідовності, діаграму активності та модель абстракцій. За допомогою цих моделей уточнено основні сценарії роботи користувачів, порядок фіксації відвідування, взаємодію між інтерфейсом, серверною частиною і базою даних. Сформовано функціональні, нефункціональні та безпекові вимоги до системи, що стало основою для подальшого проєктування.

У другому розділі розроблено інформаційне, структурне та алгоритмічне забезпечення системи. Побудовано логічну модель даних у вигляді ER-діаграми, у якій визначено сутності академічної групи, студента, викладача, дисципліни, заняття, запису відвідування та причини відсутності. На її основі сформовано фізичну модель бази даних UniversityAttendanceDB із таблицями, ключами, зв’язками, обмеженнями цілісності та службовими полями. Така

структура забезпечує коректне зберігання даних і запобігає дублюванню записів відвідування.

У межах другого розділу також розроблено діаграму компонентів і пакетну структуру програмного забезпечення. Систему поділено на клієнтську частину, серверну частину та рівень даних. Описано модулі автентифікації, обліку відвідування, керування заняттями, роботи зі студентами та групами, звітності, аналітики, експорту і доступу до бази даних. Окремо сформовано алгоритмічне забезпечення для звітів: відвідуваність групи за період, пропущені заняття за дисципліною та зведена статистика по студентах і групах.

У третьому розділі обґрунтовано вибір технологій реалізації системи. Для клієнтської частини обрано HTML, CSS і JavaScript, що забезпечують створення вебінтерфейсу, форм введення, таблиць, фільтрів і сторінок звітності. Для серверної частини використано Node.js та Express.js, які дозволяють реалізувати маршрутизацію, авторизацію, рольовий доступ, обробку запитів і взаємодію з базою даних. Для зберігання інформації використано реляційну базу UniversityAttendanceDB, що відповідає структурі предметної області.

Представлено архітектуру системи як веборієнтованого клієнт-серверного застосунку. Клієнтська частина забезпечує роботу користувача через браузер, серверна частина реалізує бізнес-логіку, а база даних зберігає навчальні, довідкові та операційні дані. Передбачено рольове розмежування доступу: адміністратор керує користувачами й довідниками, викладач працює зі своїми заняттями та журналами, деканат формує зведені звіти, а студент переглядає власний розклад і персональну відвідуваність.

У четвертому розділі виконано тестування програмних модулів і системи в цілому. Сформовано план тестування, який охоплює модулі авторизації, реєстрації, рольового доступу, керування студентами, групами, дисциплінами, заняттями, журналом відвідування, звітністю, аналітикою та експортом даних. Перевірено основні сценарії роботи для адміністратора, викладача, працівника деканату та студента. Результати тестування підтвердили, що кожна роль отримує доступ лише до визначеного набору функцій.

Під час інтерфейсного тестування перевірено сторінку входу, робоче місце викладача, розділ створення і перегляду занять, список студентів закріплених груп, студентський розклад і профіль користувача. Встановлено, що інтерфейс має зрозумілу структуру, підтримує фільтрацію, відображає таблиці з навчальними даними та коректно формує персональні показники відвідування. Також описано порядок розгортання системи, склад інсталяційного пакету, вимоги до серверного середовища, бази даних і клієнтського доступу через браузер.

Практичним результатом роботи є програмна система, яка підтримує авторизацію та реєстрацію користувачів, рольовий доступ, роботу з академічними групами, студентами, дисциплінами й заняттями, фіксацію присутності, облік причин відсутності, перегляд персональної статистики, формування звітів та експорт результатів. Система може використовуватися як основа для впровадження в навчальному підрозділі університету або для подальшого розширення функціональності.

Отже, мету кваліфікаційної роботи досягнуто. Розроблена інформаційна система забезпечує автоматизацію обліку відвідування і занять студентами університету, підвищує достовірність навчальних даних, скорочує час підготовки звітності та створює умови для аналітичного контролю відвідуваності. Подальший розвиток системи може бути пов’язаний з інтеграцією з електронним розкладом, додаванням повідомлень для студентів і викладачів, розширенням аналітичних панелей та впровадженням резервного копіювання на рівні адміністративного інтерфейсу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про вищу освіту: Закон України від 01.07.2014 № 1556-VII. URL: https://zakon.rada.gov.ua/laws/show/1556-18.
2. Attendance activity. MoodleDocs. URL: https://docs.moodle.org/en/Attendance_activity.
3. Classroom Management Tools & Resources. Google for Education. URL: https://edu.google.com/workspace-for-education/products/classroom/.
4. Educator’s guide to Insights in Microsoft Teams. Microsoft Support. URL: https://support.microsoft.com/en-us/topic/educator-s-guide-to-insights-in-microsoft-teams-27b56255-90c0-47aa-bac3-1c9f50157181.
5. IT Admin Guide to Education Insights in Microsoft Teams. Microsoft Learn. URL: https://learn.microsoft.com/en-us/microsoftteams/class-insights.
6. HTML: HyperText Markup Language. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/HTML.
7. CSS: Cascading Style Sheets. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/CSS.
8. JavaScript. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript.
9. Node.js Learn. Node.js Foundation. URL: https://nodejs.org/learn.
10. Express routing. Express.js Documentation. URL: https://expressjs.com/en/guide/routing.html.
11. MySQL 8.4 Reference Manual. CREATE TABLE Statement. Oracle. URL: https://dev.mysql.com/doc/en/create-table.html.
12. MySQL 8.4 Reference Manual. CREATE INDEX Statement. Oracle. URL: https://dev.mysql.com/doc/en/create-index.html.
13. Apache HTTP Server Documentation. Apache Software Foundation. URL: https://httpd.apache.org/docs/.
14. OWASP Application Security Verification Standard. OWASP Foundation. URL: https://owasp.org/www-project-application-security-verification-standard/.

15. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлення. Київ: ДП «УкрНДНЦ», 2016.
16. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ: ДП «УкрНДНЦ», 2016.
17. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Addison-Wesley, 2003.
18. Silberschatz A., Korth H., Sudarshan S. Database System Concepts. 7th ed. McGraw-Hill, 2019.
19. Robbins J. Learning Web Design: A Beginner’s Guide to HTML, CSS, JavaScript, and Web Graphics. 5th ed. O’Reilly Media, 2018.
20. Sommerville I. Software Engineering. 10th ed. Pearson, 2016.

ДОДАТОК А

Файл server.js

```
const express = require("express");
const session = require("express-session");
const mysql = require("mysql2/promise");
const bcrypt = require("bcryptjs");
const path = require("path");

const app = express();

const dbConfig = {
    host: "localhost",
    user: "root",
    password: "",
    database: "UniversityAttendanceDB",
    waitForConnections: true,
    connectionLimit: 10
};

const pool = mysql.createPool(dbConfig);

app.use(express.urlencoded({ extended: true }));
app.use(express.json());
app.use(express.static(path.join(__dirname, "public")));

app.use(session({
    secret: "attendance-system-secret-key",
    resave: false,
    saveUninitialized: false,
    cookie: {
        httpOnly: true,
        maxAge: 1000 * 60 * 60 * 4
    }
}));

function requireAuth(req, res, next) {
    if (!req.session.user) {
        return res.status(401).json({ error: "Користувач не авторизований" });
    }
    next();
}

function allowRoles(...roles) {
    return function (req, res, next) {
        if (!req.session.user) {
            return res.status(401).json({ error: "Користувач не
авторизований" });
        }

        if (!roles.includes(req.session.user.role_name)) {
            return res.status(403).json({ error: "Недостатньо прав доступу" });
        }

        next();
    };
}

app.post("/api/register", async (req, res) => {
    try {
        const { username, password, full_name, email, role_name } = req.body;
```

```
        if (!username || !password || !full_name || !role_name) {
            return res.status(400).json({ error: "Не заповнено обов’язкові поля"
});
        }

        const passwordHash = await bcrypt.hash(password, 10);

        await pool.execute(
            `INSERT INTO users (username, password_hash, full_name, email,
role_name)
             VALUES (?, ?, ?, ?, ?)`,
            [username, passwordHash, full_name, email || null, role_name]
        );

        res.json({ message: "Користувача успішно зареєстровано" });
    } catch (error) {
        res.status(500).json({ error: "Помилка реєстрації користувача" });
    }
});

app.post("/api/login", async (req, res) => {
    try {
        const { username, password } = req.body;

        const [rows] = await pool.execute(
            `SELECT user_id, username, password_hash, full_name, email,
role_name
             FROM users
             WHERE username = ? AND is_active = TRUE`,
            [username]
        );

        if (rows.length === 0) {
            return res.status(401).json({ error: "Неправильний логін або пароль"
});
        }

        const user = rows[0];
        const isValid = await bcrypt.compare(password, user.password_hash);

        if (!isValid) {
            return res.status(401).json({ error: "Неправильний логін або пароль"
});
        }

        req.session.user = {
            user_id: user.user_id,
            username: user.username,
            full_name: user.full_name,
            email: user.email,
            role_name: user.role_name
        };

        res.json({
            message: "Авторизацію виконано успішно",
            user: req.session.user
        });
    } catch (error) {
        res.status(500).json({ error: "Помилка авторизації" });
    }
});

app.post("/api/logout", requireAuth, (req, res) => {
    req.session.destroy(() => {
```

```
        res.json({ message: "Вихід із системи виконано" });
    });
});

app.get("/api/me", requireAuth, (req, res) => {
    res.json(req.session.user);
});

app.get("/api/groups", requireAuth, async (req, res) => {
    const [rows] = await pool.execute(
        `SELECT group_id, group_code, group_name, course_year, faculty_name,
department_name, study_year
         FROM academic_groups
         ORDER BY group_code`
    );

    res.json(rows);
});

app.post("/api/groups", allowRoles("admin", "dean"), async (req, res) => {
    const { group_code, group_name, course_year, faculty_name, department_name,
study_year } = req.body;

    await pool.execute(
        `INSERT INTO academic_groups
         (group_code, group_name, course_year, faculty_name, department_name,
study_year)
         VALUES (?, ?, ?, ?, ?, ?)`,
        [group_code, group_name, course_year, faculty_name, department_name,
study_year]
    );

    res.json({ message: "Академічну групу створено" });
});

app.get("/api/students", requireAuth, async (req, res) => {
    const user = req.session.user;

    let sql = `
        SELECT s.student_id, s.student_card_no, s.last_name, s.first_name,
s.middle_name,
               s.email, s.phone, s.is_active, g.group_code, g.group_name
        FROM students s
        JOIN academic_groups g ON g.group_id = s.group_id
    `;

    const params = [];

    if (user.role_name === "teacher") {
        sql += `
            WHERE s.group_id IN (
                SELECT DISTINCT group_id
                FROM class_sessions
                WHERE teacher_id = (
                    SELECT teacher_id FROM teachers WHERE user_id = ?
                )
            )
        `;
        params.push(user.user_id);
    }

    if (user.role_name === "student") {
        sql += ` WHERE s.user_id = ?`;
        params.push(user.user_id);
```

```
    }

    sql += ` ORDER BY g.group_code, s.last_name, s.first_name`;

    const [rows] = await pool.execute(sql, params);
    res.json(rows);
});

app.post("/api/students", allowRoles("admin", "dean"), async (req, res) => {
    const { group_id, student_card_no, last_name, first_name, middle_name,
email, phone } = req.body;

    await pool.execute(
        `INSERT INTO students
         (group_id, student_card_no, last_name, first_name, middle_name, email,
phone)
         VALUES (?, ?, ?, ?, ?, ?, ?)`,
        [group_id, student_card_no, last_name, first_name, middle_name || null,
email || null, phone || null]
    );

    res.json({ message: "Студента додано" });
});

app.get("/api/disciplines", requireAuth, async (req, res) => {
    const [rows] = await pool.execute(
        `SELECT discipline_id, discipline_code, discipline_name, credits,
control_form
         FROM disciplines
         ORDER BY discipline_name`
    );

    res.json(rows);
});

app.post("/api/disciplines", allowRoles("admin", "dean"), async (req, res) => {
    const { discipline_code, discipline_name, credits, control_form } =
req.body;

    await pool.execute(
        `INSERT INTO disciplines (discipline_code, discipline_name, credits,
control_form)
         VALUES (?, ?, ?, ?)`,
        [discipline_code, discipline_name, credits || null, control_form ||
null]
    );

    res.json({ message: "Дисципліну додано" });
});

app.get("/api/sessions", requireAuth, async (req, res) => {
    const user = req.session.user;
    const { group_id, discipline_id, date_from, date_to } = req.query;

    let sql = `
        SELECT cs.session_id, cs.session_date, cs.start_time, cs.end_time,
cs.room_no,
               cs.topic, lt.lesson_type_name,
               g.group_code, d.discipline_name,
               CONCAT(t.last_name, ' ', t.first_name) AS teacher_name
        FROM class_sessions cs
        JOIN academic_groups g ON g.group_id = cs.group_id
        JOIN disciplines d ON d.discipline_id = cs.discipline_id
        JOIN teachers t ON t.teacher_id = cs.teacher_id
```

```
        JOIN lesson_types lt ON lt.lesson_type_id = cs.lesson_type_id
        WHERE 1 = 1
    `;

    const params = [];

    if (group_id) {
        sql += ` AND cs.group_id = ?`;
        params.push(group_id);
    }

    if (discipline_id) {
        sql += ` AND cs.discipline_id = ?`;
        params.push(discipline_id);
    }

    if (date_from) {
        sql += ` AND cs.session_date >= ?`;
        params.push(date_from);
    }

    if (date_to) {
        sql += ` AND cs.session_date <= ?`;
        params.push(date_to);
    }

    if (user.role_name === "teacher") {
        sql += ` AND cs.teacher_id = (SELECT teacher_id FROM teachers WHERE
user_id = ?)`;
        params.push(user.user_id);
    }

    if (user.role_name === "student") {
        sql += `
            AND cs.group_id = (
                SELECT group_id FROM students WHERE user_id = ?
            )
        `;
        params.push(user.user_id);
    }

    sql += ` ORDER BY cs.session_date DESC, cs.start_time DESC`;

    const [rows] = await pool.execute(sql, params);
    res.json(rows);
});

app.post("/api/sessions", allowRoles("admin", "teacher"), async (req, res) => {
    const user = req.session.user;
    const { group_id, discipline_id, lesson_type_id, session_date, start_time,
end_time, room_no, topic } = req.body;

    let teacherId = req.body.teacher_id;

    if (user.role_name === "teacher") {
        const [teacherRows] = await pool.execute(
            `SELECT teacher_id FROM teachers WHERE user_id = ?`,
            [user.user_id]
        );

        if (teacherRows.length === 0) {
            return res.status(400).json({ error: "Для користувача не знайдено
профіль викладача" });
        }
```

```
        teacherId = teacherRows[0].teacher_id;
    }

    await pool.execute(
        `INSERT INTO class_sessions
         (group_id, discipline_id, teacher_id, lesson_type_id, session_date,
start_time, end_time, room_no, topic)
         VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)`,
        [group_id, discipline_id, teacherId, lesson_type_id, session_date,
start_time, end_time, room_no || null, topic || null]
    );

    res.json({ message: "Заняття створено" });
});

app.get("/api/sessions/:id/attendance", allowRoles("admin", "teacher", "dean"),
async (req, res) => {
    const sessionId = req.params.id;

    const [rows] = await pool.execute(
        `SELECT s.student_id, s.student_card_no, s.last_name, s.first_name,
s.middle_name,
                ar.attendance_id, ar.status_code, ar.reason_id, ar.note,
ar.marked_at
         FROM class_sessions cs
         JOIN students s ON s.group_id = cs.group_id
         LEFT JOIN attendance_records ar
                ON ar.session_id = cs.session_id AND ar.student_id =
s.student_id
         WHERE cs.session_id = ?
         ORDER BY s.last_name, s.first_name`,
        [sessionId]
    );

    res.json(rows);
});

app.post("/api/sessions/:id/attendance", allowRoles("admin", "teacher"), async
(req, res) => {
    const sessionId = req.params.id;
    const { records } = req.body;

    const connection = await pool.getConnection();

    try {
        await connection.beginTransaction();

        for (const record of records) {
            await connection.execute(
                `INSERT INTO attendance_records
                 (session_id, student_id, status_code, reason_id, note)
                 VALUES (?, ?, ?, ?, ?)
                 ON DUPLICATE KEY UPDATE
                    status_code = VALUES(status_code),
                    reason_id = VALUES(reason_id),
                    note = VALUES(note),
                    marked_at = CURRENT_TIMESTAMP`,
                [
                    sessionId,
                    record.student_id,
                    record.status_code,
                    record.reason_id || null,
                    record.note || null
```

```
                ]
            );
        }

        await connection.commit();
        res.json({ message: "Журнал відвідування збережено" });
    } catch (error) {
        await connection.rollback();
        res.status(500).json({ error: "Помилка збереження журналу" });
    } finally {
        connection.release();
    }
});

app.get("/api/reports/group", allowRoles("admin", "dean", "teacher"), async
(req, res) => {
    const { group_id, date_from, date_to } = req.query;

    const [rows] = await pool.execute(
        `SELECT g.group_code, s.student_card_no,
                CONCAT(s.last_name, ' ', s.first_name, ' ',
IFNULL(s.middle_name, '')) AS student_name,
                COUNT(ar.attendance_id) AS total_marks,
                SUM(ar.status_code = 'P') AS present_count,
                SUM(ar.status_code = 'A') AS absent_count,
                SUM(ar.status_code = 'L') AS late_count
         FROM students s
         JOIN academic_groups g ON g.group_id = s.group_id
         JOIN class_sessions cs ON cs.group_id = g.group_id
         LEFT JOIN attendance_records ar
                ON ar.session_id = cs.session_id AND ar.student_id =
s.student_id
         WHERE g.group_id = ?
           AND cs.session_date BETWEEN ? AND ?
         GROUP BY g.group_code, s.student_id
         ORDER BY s.last_name, s.first_name`,
        [group_id, date_from, date_to]
    );

    res.json(rows);
});

app.get("/api/reports/discipline-absences", allowRoles("admin", "dean",
"teacher"), async (req, res) => {
    const { discipline_id, date_from, date_to } = req.query;

    const [rows] = await pool.execute(
        `SELECT d.discipline_name, g.group_code,
                CONCAT(s.last_name, ' ', s.first_name) AS student_name,
                ar.status_code, r.reason_name, ar.note, cs.session_date
         FROM attendance_records ar
         JOIN class_sessions cs ON cs.session_id = ar.session_id
         JOIN students s ON s.student_id = ar.student_id
         JOIN academic_groups g ON g.group_id = s.group_id
         JOIN disciplines d ON d.discipline_id = cs.discipline_id
         LEFT JOIN absence_reasons r ON r.reason_id = ar.reason_id
         WHERE cs.discipline_id = ?
           AND cs.session_date BETWEEN ? AND ?
           AND ar.status_code = 'A'
         ORDER BY cs.session_date DESC, g.group_code, s.last_name`,
        [discipline_id, date_from, date_to]
    );

    res.json(rows);
```

```
});

app.get("/api/student/profile", allowRoles("student"), async (req, res) => {
    const [rows] = await pool.execute(
        `SELECT s.student_id, s.student_card_no, s.last_name, s.first_name,
s.middle_name,
                s.email, g.group_code, g.group_name
         FROM students s
         JOIN academic_groups g ON g.group_id = s.group_id
         WHERE s.user_id = ?`,
        [req.session.user.user_id]
    );

    if (rows.length === 0) {
        return res.status(404).json({ error: "Профіль студента не знайдено" });
    }

    res.json(rows[0]);
});

app.listen(3000, () => {
    console.log("Attendance Web App запущено: http://localhost:3000");
});
```

ДОДАТОК Б

Файл schema.sql

```
CREATE DATABASE IF NOT EXISTS UniversityAttendanceDB
    CHARACTER SET utf8mb4
    COLLATE utf8mb4_unicode_ci;

USE UniversityAttendanceDB;

CREATE TABLE users (
    user_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    full_name VARCHAR(150) NOT NULL,
    email VARCHAR(150) UNIQUE,
    role_name ENUM('admin', 'dean', 'teacher', 'student') NOT NULL,
    is_active BOOLEAN NOT NULL DEFAULT TRUE,
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE academic_groups (
    group_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    group_code VARCHAR(20) NOT NULL UNIQUE,
    group_name VARCHAR(100) NOT NULL,
    course_year SMALLINT NOT NULL CHECK (course_year BETWEEN 1 AND 6),
    faculty_name VARCHAR(150) NOT NULL,
    department_name VARCHAR(150) NOT NULL,
    study_year SMALLINT NOT NULL CHECK (study_year >= 2020),
    education_form VARCHAR(30) DEFAULT 'денна'
);

CREATE TABLE teachers (
    teacher_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    user_id BIGINT UNIQUE,
    teacher_code VARCHAR(30) NOT NULL UNIQUE,
    last_name VARCHAR(100) NOT NULL,
    first_name VARCHAR(100) NOT NULL,
    middle_name VARCHAR(100),
    position_name VARCHAR(100),
    department VARCHAR(150),
    email VARCHAR(150) UNIQUE,
    phone VARCHAR(30),
    CONSTRAINT fk_teachers_user
        FOREIGN KEY (user_id)
        REFERENCES users(user_id)
        ON DELETE SET NULL
);

CREATE TABLE disciplines (
    discipline_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    discipline_code VARCHAR(30) NOT NULL UNIQUE,
    discipline_name VARCHAR(150) NOT NULL UNIQUE,
    credits DECIMAL(3,1) CHECK (credits IS NULL OR credits BETWEEN 1 AND 20),
    control_form VARCHAR(50)
);

CREATE TABLE lesson_types (
    lesson_type_id TINYINT PRIMARY KEY AUTO_INCREMENT,
    lesson_type_name VARCHAR(50) NOT NULL UNIQUE
);

CREATE TABLE students (
```

```
    student_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    user_id BIGINT UNIQUE,
    group_id BIGINT NOT NULL,
    student_card_no VARCHAR(30) NOT NULL UNIQUE,
    last_name VARCHAR(100) NOT NULL,
    first_name VARCHAR(100) NOT NULL,
    middle_name VARCHAR(100),
    email VARCHAR(150) UNIQUE,
    phone VARCHAR(30),
    is_active BOOLEAN NOT NULL DEFAULT TRUE,
    CONSTRAINT fk_students_user
        FOREIGN KEY (user_id)
        REFERENCES users(user_id)
        ON DELETE SET NULL,
    CONSTRAINT fk_students_group
        FOREIGN KEY (group_id)
        REFERENCES academic_groups(group_id)
);

CREATE TABLE attendance_statuses (
    status_code CHAR(1) PRIMARY KEY,
    status_name VARCHAR(50) NOT NULL UNIQUE,
    is_absence BOOLEAN NOT NULL,
    affects_percentage BOOLEAN NOT NULL DEFAULT TRUE,
    CHECK (status_code IN ('P', 'A', 'L'))
);

CREATE TABLE absence_reasons (
    reason_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    reason_name VARCHAR(120) NOT NULL UNIQUE,
    is_excused BOOLEAN NOT NULL DEFAULT FALSE
);

CREATE TABLE class_sessions (
    session_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    group_id BIGINT NOT NULL,
    discipline_id BIGINT NOT NULL,
    teacher_id BIGINT NOT NULL,
    lesson_type_id TINYINT NOT NULL,
    session_date DATE NOT NULL,
    start_time TIME NOT NULL,
    end_time TIME NOT NULL,
    room_no VARCHAR(30),
    topic VARCHAR(200),
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT fk_sessions_group
        FOREIGN KEY (group_id)
        REFERENCES academic_groups(group_id),
    CONSTRAINT fk_sessions_discipline
        FOREIGN KEY (discipline_id)
        REFERENCES disciplines(discipline_id),
    CONSTRAINT fk_sessions_teacher
        FOREIGN KEY (teacher_id)
        REFERENCES teachers(teacher_id),
    CONSTRAINT fk_sessions_lesson_type
        FOREIGN KEY (lesson_type_id)
        REFERENCES lesson_types(lesson_type_id),
    CONSTRAINT uq_class_session
        UNIQUE (group_id, discipline_id, teacher_id, session_date, start_time),
    CONSTRAINT ck_class_session_time
        CHECK (end_time > start_time)
);

CREATE TABLE attendance_records (
```

```
    attendance_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    session_id BIGINT NOT NULL,
    student_id BIGINT NOT NULL,
    status_code CHAR(1) NOT NULL,
    reason_id BIGINT,
    marked_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    note VARCHAR(255),
    CONSTRAINT fk_attendance_session
        FOREIGN KEY (session_id)
        REFERENCES class_sessions(session_id)
        ON DELETE CASCADE,
    CONSTRAINT fk_attendance_student
        FOREIGN KEY (student_id)
        REFERENCES students(student_id)
        ON DELETE CASCADE,
    CONSTRAINT fk_attendance_status
        FOREIGN KEY (status_code)
        REFERENCES attendance_statuses(status_code),
    CONSTRAINT fk_attendance_reason
        FOREIGN KEY (reason_id)
        REFERENCES absence_reasons(reason_id),
    CONSTRAINT uq_attendance_session_student
        UNIQUE (session_id, student_id)
);

CREATE INDEX idx_students_group
    ON students(group_id);

CREATE INDEX idx_sessions_group_date
    ON class_sessions(group_id, session_date);

CREATE INDEX idx_sessions_teacher_date
    ON class_sessions(teacher_id, session_date);

CREATE INDEX idx_sessions_discipline_date
    ON class_sessions(discipline_id, session_date);

CREATE INDEX idx_attendance_student
    ON attendance_records(student_id);

CREATE INDEX idx_attendance_session
    ON attendance_records(session_id);

CREATE INDEX idx_attendance_status
    ON attendance_records(status_code);

INSERT INTO attendance_statuses (status_code, status_name, is_absence,
affects_percentage)
VALUES
('P', 'Присутній', FALSE, TRUE),
('A', 'Відсутній', TRUE, TRUE),
('L', 'Запізнився', FALSE, TRUE);

INSERT INTO lesson_types (lesson_type_name)
VALUES
('Лекція'),
('Практичне заняття'),
('Лабораторна робота'),
('Семінар');

INSERT INTO absence_reasons (reason_name, is_excused)
VALUES
('Хвороба', TRUE),
('Участь в офіційному заході', TRUE),
```

```
('Сімейні обставини', TRUE),
('Без поважної причини', FALSE);
```

ДОДАТОК В

Файл public/app.js

```
const state = {
    user: null,
    currentSection: "dashboard"
};

async function request(url, options = {}) {
    const response = await fetch(url, {
        headers: {
            "Content-Type": "application/json"
        },
        credentials: "include",
        ...options
    });

    const data = await response.json();

    if (!response.ok) {
        throw new Error(data.error || "Помилка виконання запиту");
    }

    return data;
}

async function login(event) {
    event.preventDefault();

    const form = event.target;

    const payload = {
        username: form.username.value.trim(),
        password: form.password.value.trim()
    };

    try {
        const data = await request("/api/login", {
            method: "POST",
            body: JSON.stringify(payload)
        });

        state.user = data.user;
        renderLayout();
        await loadDashboard();
    } catch (error) {
        showMessage(error.message, "error");
    }
}

async function logout() {
    await request("/api/logout", { method: "POST" });
    state.user = null;
    renderLogin();
}

function renderLogin() {
    document.body.innerHTML = `
        <main class="login-page">
            <section class="login-card">
                <div class="login-header">
                    <h1>University Attendance</h1>
```

```
                    <p>Інформаційна система аналізу відвідування і занять
студентами університету</p>
                </div>

                <form onsubmit="login(event)" class="login-form">
                    <label>Логін</label>
                    <input name="username" type="text" required>

                    <label>Пароль</label>
                    <input name="password" type="password" required>

                    <button type="submit">Увійти до системи</button>
                </form>

                <div id="message"></div>
            </section>
        </main>
    `;
}

function renderLayout() {
    const menu = getMenuByRole(state.user.role_name);

    document.body.innerHTML = `
        <aside class="sidebar">
            <div class="brand">
                <span class="brand-title">Attendance Web App</span>
                <span class="brand-subtitle">${state.user.full_name}</span>
            </div>

            <nav>
                ${menu.map(item => `
                    <button class="menu-item" onclick="openSection('$
{item.key}')">
                        ${item.title}
                    </button>
                `).join("")}
            </nav>

            <button class="logout-button" onclick="logout()">Вийти</button>
        </aside>

        <main class="workspace">
            <header class="topbar">
                <div>
                    <h1 id="page-title">Панель користувача</h1>
                    <p id="page-subtitle">Роль: $
{translateRole(state.user.role_name)}</p>
                </div>
            </header>

            <section id="content" class="content"></section>
        </main>
    `;
}

function getMenuByRole(role) {
    const common = [
        { key: "dashboard", title: "Головна" },
        { key: "profile", title: "Профіль" }
    ];

    if (role === "admin") {
        return [
```

```
            ...common,
            { key: "users", title: "Користувачі" },
            { key: "groups", title: "Групи" },
            { key: "students", title: "Студенти" },
            { key: "disciplines", title: "Дисципліни" },
            { key: "sessions", title: "Заняття" },
            { key: "attendance", title: "Журнал" },
            { key: "reports", title: "Звіти" }
        ];
    }

    if (role === "dean") {
        return [
            ...common,
            { key: "groups", title: "Групи" },
            { key: "students", title: "Студенти" },
            { key: "disciplines", title: "Дисципліни" },
            { key: "sessions", title: "Заняття" },
            { key: "reports", title: "Звіти" }
        ];
    }

    if (role === "teacher") {
        return [
            ...common,
            { key: "sessions", title: "Мої заняття" },
            { key: "students", title: "Студенти моїх груп" },
            { key: "attendance", title: "Журнал" },
            { key: "reports", title: "Звіти" }
        ];
    }

    return [
        ...common,
        { key: "student-schedule", title: "Мій розклад" },
        { key: "student-attendance", title: "Моя відвідуваність" }
    ];
}

async function openSection(section) {
    state.currentSection = section;

    if (section === "dashboard") {
        await loadDashboard();
    }

    if (section === "students") {
        await loadStudents();
    }

    if (section === "sessions") {
        await loadSessions();
    }

    if (section === "reports") {
        renderReports();
    }

    if (section === "student-schedule") {
        await loadStudentSchedule();
    }

    if (section === "profile") {
        await loadProfile();
```

```
    }
}

async function loadDashboard() {
    document.getElementById("page-title").textContent = "Головна панель";
    document.getElementById("content").innerHTML = `
        <div class="dashboard-grid">
            <div class="stat-card">
                <span class="stat-label">Користувач</span>
                <strong>${state.user.full_name}</strong>
            </div>
            <div class="stat-card">
                <span class="stat-label">Роль</span>
                <strong>${translateRole(state.user.role_name)}</strong>
            </div>
            <div class="stat-card">
                <span class="stat-label">Система</span>
                <strong>UniversityAttendanceDB</strong>
            </div>
        </div>
    `;
}

async function loadStudents() {
    document.getElementById("page-title").textContent = "Студенти";

    const students = await request("/api/students");

    document.getElementById("content").innerHTML = `
        <div class="table-card">
            <table>
                <thead>
                    <tr>
                        <th>Група</th>
                        <th>Студентський квиток</th>
                        <th>ПІБ</th>
                        <th>Email</th>
                        <th>Телефон</th>
                    </tr>
                </thead>
                <tbody>
                    ${students.map(student => `
                        <tr>
                            <td>${student.group_code}</td>
                            <td>${student.student_card_no}</td>
                            <td>${student.last_name} ${student.first_name} $
{student.middle_name || ""}</td>
                            <td>${student.email || ""}</td>
                            <td>${student.phone || ""}</td>
                        </tr>
                    `).join("")}
                </tbody>
            </table>
        </div>
    `;
}

async function loadSessions() {
    document.getElementById("page-title").textContent = "Заняття";

    const sessions = await request("/api/sessions");

    document.getElementById("content").innerHTML = `
        <div class="toolbar">
```

```
            ${state.user.role_name === "admin" || state.user.role_name ===
"teacher" ? `
                <button onclick="renderSessionForm()">Створити заняття</button>
            ` : ""}
        </div>

        <div class="table-card">
            <table>
                <thead>
                    <tr>
                        <th>Дата</th>
                        <th>Час</th>
                        <th>Група</th>
                        <th>Дисципліна</th>
                        <th>Тип</th>
                        <th>Викладач</th>
                        <th>Аудиторія</th>
                    </tr>
                </thead>
                <tbody>
                    ${sessions.map(session => `
                        <tr>
                            <td>${formatDate(session.session_date)}</td>
                            <td>${session.start_time} - ${session.end_time}</td>
                            <td>${session.group_code}</td>
                            <td>${session.discipline_name}</td>
                            <td>${session.lesson_type_name}</td>
                            <td>${session.teacher_name}</td>
                            <td>${session.room_no || ""}</td>
                        </tr>
                    `).join("")}
                </tbody>
            </table>
        </div>
    `;
}

function renderReports() {
    document.getElementById("page-title").textContent = "Звіти та аналітика";

    document.getElementById("content").innerHTML = `
        <div class="report-grid">
            <section class="report-card">
                <h2>Відвідуваність групи</h2>
                <p>Формування звіту за академічною групою і періодом.</p>
                <button onclick="loadGroupReport()">Сформувати</button>
            </section>

            <section class="report-card">
                <h2>Пропуски за дисципліною</h2>
                <p>Аналіз відсутності студентів за вибраною дисципліною.</p>
                <button onclick="loadDisciplineAbsences()">Сформувати</button>
            </section>
        </div>

        <div id="report-result"></div>
    `;
}

async function loadStudentSchedule() {
    document.getElementById("page-title").textContent = "Мій розклад";

    const sessions = await request("/api/sessions");
```

```
    document.getElementById("content").innerHTML = `
        <div class="table-card">
            <table>
                <thead>
                    <tr>
                        <th>Дата</th>
                        <th>Час</th>
                        <th>Дисципліна</th>
                        <th>Тип заняття</th>
                        <th>Викладач</th>
                        <th>Аудиторія</th>
                    </tr>
                </thead>
                <tbody>
                    ${sessions.map(session => `
                        <tr>
                            <td>${formatDate(session.session_date)}</td>
                            <td>${session.start_time} - ${session.end_time}</td>
                            <td>${session.discipline_name}</td>
                            <td>${session.lesson_type_name}</td>
                            <td>${session.teacher_name}</td>
                            <td>${session.room_no || ""}</td>
                        </tr>
                    `).join("")}
                </tbody>
            </table>
        </div>
    `;
}

async function loadProfile() {
    document.getElementById("page-title").textContent = "Профіль";

    if (state.user.role_name === "student") {
        const profile = await request("/api/student/profile");

        document.getElementById("content").innerHTML = `
            <div class="profile-card">
                <h2>${profile.last_name} ${profile.first_name} $
{profile.middle_name || ""}</h2>
                <p>Група: ${profile.group_code}</p>
                <p>Студентський квиток: ${profile.student_card_no}</p>
                <p>Email: ${profile.email || ""}</p>
            </div>
        `;
    } else {
        document.getElementById("content").innerHTML = `
            <div class="profile-card">
                <h2>${state.user.full_name}</h2>
                <p>Логін: ${state.user.username}</p>
                <p>Роль: ${translateRole(state.user.role_name)}</p>
                <p>Email: ${state.user.email || ""}</p>
            </div>
        `;
    }
}

function translateRole(role) {
    const roles = {
        admin: "Адміністратор",
        dean: "Деканат",
        teacher: "Викладач",
        student: "Студент"
    };
```

```
    return roles[role] || role;
}

function formatDate(value) {
    if (!value) {
        return "";
    }

    return new Date(value).toLocaleDateString("uk-UA");
}

function showMessage(text, type = "info") {
    const block = document.getElementById("message");

    if (!block) {
        return;
    }

    block.className = `message ${type}`;
    block.textContent = text;
}

async function init() {
    try {
        state.user = await request("/api/me");
        renderLayout();
        await loadDashboard();
    } catch {
        renderLogin();
    }
}

init();
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Шинкаренко Павло Олександрович, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проєкт) на тему «Інформаційна система аналізу відвідування занять студентами Університету» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
    - ○ [ ] інструменти генеративного ШІ не використовувалися.
    - ○ [+] інструменти ШІ (вказати назву: **ChatGPT**, Gemini, Claude, DeepL, _____інші (вказати назву)) були використані для:
        - ■ [ ] перекладу іншомовних джерел;
        - ■ [ ] стилістичного редагування та перевірки граматики;
        - ■ [+] допомоги у написанні/відлагодженні програмного коду;
        - ■ [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«24» травня 2026 р. _____ **Шинкаренко Павло**
(підпис)